KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2010/2011
pod redakcją Tomasza Kubika

KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2010/2011
pod redakcją Tomasza Kubika

Skład komputerowy, projekt okładki
Tomasz Kubik

ISBN 978-83-930823-2-2

Wydawca
Tomasz Kubik

Druk i oprawa
I-BiS sc., ul. Lelewela 4, 53-505 Wrocław

Skład niniejszej książki wykonano w systemie LaTeX

SPIS TREŚCI

Słowo wstępne **11**

1 Udostępnianie informacji w formacie RSS i GeoRSS **13**
1.1. Czym jest RSS? . . . 13
1.1.1. RSS - budowa . . . 15
1.1.2. RSS wersje . . . 16
1.2. Wprowadzenie do GeoRSS . . . 17
1.2.1. GeoRSS Simple . . . 17
1.2.2. GeoRSS-GML . . . 19
1.3. OpenLayers . . . 21
1.4. ExtJS . . . 22
1.5. GeoExt . . . 23
1.6. Przykładowe zastosowanie GeoRSS . . . 24
1.7. Szczegóły implementacyjne . . . 26
1.7.1. Element Mapa . . . 26
1.7.2. Kontrolki mapy . . . 26
1.7.3. Warstwa znaczników . . . 27
1.7.4. Kontener . . . 27
1.7.5. Panel . . . 28
1.7.6. Uwagi na temat wdrożenia . . . 29
1.8. Podsumowanie . . . 29
Literatura . . . 30

2 Wyszukiwanie przestrzenno-czasowe **31**
2.1. Wyszukiwanie dokumentów . . . 31
2.2. Wyszukiwanie w oparciu o metadane . . . 32
2.3. Systemy DMS . . . 33
2.3.1. Czym są systemy DMS . . . 33
2.3.2. Ograniczenia systemów DMS . . . 34
2.3.3. Rozwiązania komercyjne i Open Source . . . 34
2.3.4. LogicalDOC jako przykład systemu DMS . . . 34
2.4. Ontologia *Spime* . . . 35
2.4.1. Podstawowe klasy . . . 36

2.4.2. Podklasy `Place` . . . 37
2.5. Architektura systemu . . . 38
2.6. Aplikacja do przeszukiwania ontologii . . . 39
2.7. Podsumowanie . . . 39
Literatura . . . 41

3 Programowe wsparcie procesu budowy tezaurusa 42
3.1. Wstęp . . . 42
3.1.1. Czym są modele informacyjne? . . . 43
3.1.2. Ontologia jako model informacji . . . 43
3.2. Tezaurus . . . 44
3.2.1. Katalog przedmiotowy . . . 44
3.2.2. Wyszukiwanie w katalogach . . . 45
3.2.3. Proces tworzenia tezaurusa . . . 46
3.2.4. Przykłady tezaurusów . . . 46
3.3. Dokumenty normatywne . . . 47
3.3.1. Wzorce i modele tezaurusów . . . 47
3.3.2. Język SKOS . . . 48
3.4. Przykładowa implementacja tezaurusa . . . 51
3.4.1. SKOS API . . . 52
3.4.2. Interfejs użytkownika . . . 53
3.4.3. Podsumowanie . . . 53
Literatura . . . 55

4 Semantyczne adnotacje dokumentów 56
4.1. Wstęp . . . 56
4.2. RDF i RDFa . . . 57
4.2.1. RDF . . . 57
4.2.2. Bliżej o RDFa . . . 59
4.2.3. Zapis grafów RDF . . . 61
4.3. Mikroformat . . . 65
4.3.1. Czym są i czym nie jest mikroformaty? . . . 65
4.3.2. Charakterystyka i budowa Mikroformatów . . . 66
4.3.3. Różnica pomiędzy HTML, a HTML z mikroformatami . . . 67
4.3.4. Przykłady . . . 68
4.4. Praktyczne wykorzystanie adnotacji semantycznych . . . 70
4.4.1. Założenia . . . 71
4.4.2. Aplikacja . . . 73
4.4.3. Podsumowanie . . . 75
Literatura . . . 76

5 Algorytm PCA w rozpoznawaniu twarzy 77
5.1. Wstęp . . . 77
Wstęp . . . 77
5.2. Architektura systemów rozpoznawania twarzy . . . 78
5.3. Analiza głównych składowych . . . 82

5.4. Przykładowa implementacja . . . 86
5.5. Badania . . . 87
5.6. Podsumowanie . . . 91
Literatura . . . 92

6 Gesty dłoni **93**
6.1. Wstęp . . . 93
6.1.1. Gesty . . . 93
6.1.2. Język migowy . . . 94
6.2. Komputerowe wsparcie w nauce języka migowego . . . 94
6.3. Oprogramowanie . . . 95
6.3.1. Akwizycja obrazu . . . 96
6.3.2. Statyczny detektor skóry . . . 96
6.3.3. Adaptacyjny detektor skóry . . . 97
6.3.4. Odejmowanie tła . . . 98
6.3.5. Wektor cech . . . 99
6.3.6. Klasyfikacja . . . 100
6.3.7. Interfejs . . . 101
6.4. Badania i odbiór aplikacji . . . 101
6.5. Definiowanie własnych gestów . . . 104
6.6. Możliwe kierunki rozwoju . . . 104
Literatura . . . 105

7 Gesty Marszałka **106**
7.1. Wstęp . . . 106
7.2. Podstawy teoretyczne . . . 107
7.2.1. Formalne definicje oraz standardy opisujące gesty marszałka 107
7.2.2. Metody rozpoznawanie gestów . . . 108
7.2.3. Klasyfikacja z wykorzystaniem sztucznych sieci neuronowych . . . 109
7.2.4. Klasyfikacja z wykorzystaniem SVM . . . 111
7.3. Praktyczna realizacja . . . 114
7.3.1. Dlaczego Wiimote? . . . 115
7.3.2. Biblioteki służące do obsługi urządzeń Wiimote . . . 117
7.3.3. Obsługa sprzętowo-programistyczna . . . 118
7.3.4. Uruchomienie systemu . . . 121
7.3.5. Dane do Klasyfikacji . . . 123
7.3.6. Opis działania klasyfikatora SVM . . . 124
7.3.7. Testy walidacyjne dla różnych wartości kerneli . . . 124
7.4. Zastosowanie w dydaktyce . . . 125
7.5. Podsumowanie i dalsze badania . . . 126
Literatura . . . 127

8 Komputerowa pseudolosowość **128**
8.1. Liczby losowe . . . 128
8.2. Liczby pseudolosowe . . . 129

8.3. Zastosowanie liczb pseudolosowych . . . 129
8.4. Generatory liczb losowych . . . 130
8.5. Generatory liczb pseudolosowych . . . 131
8.6. Historia rozwoju metod otrzymywania liczb losowych . . . 132
8.7. Generator liniowy . . . 133
8.7.1. Typy generatorów liniowych . . . 133
8.7.2. Dobór współczynników . . . 134
8.8. Generator kwadratowy BBS . . . 135
8.8.1. Historia powstania algorytmu BBS . . . 135
8.8.2. Charakterystyka algorytmu . . . 136
8.8.3. Warunki bezpieczeństwa algorytmu . . . 138
8.9. Testowanie algorytmów pseudolosowych . . . 140
8.9.1. Cele testowania algorytmów . . . 140
8.9.2. Metody analizy wyników . . . 141
8.10. Przykłady implementacji . . . 142
8.10.1. Wizualizacja . . . 143
8.10.2. Szyfrowanie XOR . . . 143
8.11. Podsumowanie . . . 145
Literatura . . . 146

9 Niepewność w metodach lokalizacji robota 147
9.1. Podstawowe pojęcia z rachunku prawdopodobieństwa . . . 148
9.2. Metody lokalizacji robota . . . 150
9.2.1. Rodzaje czujników . . . 151
9.2.2. Filtr Bayes'a . . . 153
9.2.3. Filtr Kalmana . . . 154
9.2.4. Lokalizacja Markova . . . 158
9.3. Oprogramowanie . . . 160
9.3.1. Postać pliku XML . . . 162
9.3.2. Interfejs . . . 163
9.4. Podsumowanie . . . 167
Literatura . . . 168

10 Wykorzystanie łańcuchów Markova do generowania tekstów 169
10.1. Wstęp teoretyczny . . . 169
10.1.1. Przykład - losowanie kul . . . 169
10.1.2. Łańcuch Markova w ujęciu probabilistycznym . . . 171
10.1.3. Łańcuch Markova w ujęciu teorii automatów . . . 171
10.2. Przykładowe zastosowanie . . . 172
10.2.1. Założenia . . . 173
10.2.2. Implementacja . . . 173
10.2.3. Działanie programu . . . 174
10.2.4. Wnioski . . . 175

11 Odkrywanie reguł asocjacyjnych 176
11.1. Hurtownie danych . . . 176

11.2. Opis problemu . . . 178
11.2.1. Podstawowe pojęcia . . . 178
11.2.2. Zbiory częste . . . 180
11.2.3. Ogólny algorytm . . . 180
11.3. Algorytm Apriori-T . . . 181
11.3.1. Algorytm Apriori . . . 181
11.3.2. Struktura T-drzewa . . . 183
11.4. Algorytm FP-Growth . . . 184
11.4.1. Idea algorytmu . . . 185
11.4.2. Kompresja bazy danych . . . 186
11.4.3. FP-drzewo . . . 187
11.4.4. Eksploracja FP-drzewa . . . 188
11.4.5. Procedura FP-Growth . . . 189
11.5. Generowanie reguł na podstawie zbiorów częstych . . . 190
11.6. Implementacja . . . 190
11.6.1. Obsługa programu . . . 190
11.6.2. Wykonane testy . . . 192
11.6.3. Podsumowanie . . . 192
Literatura . . . 194

12 Interfejsy programowe społecznościowych portali 195
12.1. Portale społecznościowe . . . 195
12.1.1. Podejście socjologiczne . . . 195
12.1.2. Historia . . . 196
12.2. Przykładowe wykorzystanie API . . . 196
12.2.1. Przykładowe funkcje segmentów . . . 197
12.3. Facebook . . . 198
12.3.1. Funkcjonalności Facebook'a . . . 198
12.3.2. Historia Facebook'a . . . 199
12.3.3. Współdziałanie z Facebook'iem . . . 200
12.3.4. Graph API . . . 200
12.3.5. Social plugins . . . 201
12.3.6. Open Graph protocol . . . 201
12.3.7. Autoryzacja . . . 203
12.4. Last.fm . . . 204
12.4.1. Gromadzenie danych przez portal Last.fm . . . 204
12.4.2. Główne funkcjonalności portalu Last.fm . . . 204
12.4.3. Korzystanie z API . . . 205
12.4.4. Żądania REST . . . 205
12.4.5. Żądania XML-RPC . . . 206
12.4.6. Wykorzystanie Last.fm API za pomocą języka Python . . . 207
12.5. Implementacja . . . 207
12.5.1. Menu . . . 208
12.5.2. Widżet . . . 208
12.5.3. Belka przełączników . . . 209

12.5.4. Typy widżetów . . . 209
12.6. Wykorzystane narzędzia . . . 210
12.6.1. Programowanie po stronie serwera . . . 210
12.6.2. Programowanie po stronie klienta . . . 210
12.6.3. JSON . . . 211
12.7. Podsumowanie . . . 211
Literatura . . . 212

13 Technologie agentowe w wyszukiwaniu informacji 213
13.1. Wstęp . . . 213
13.2. Podstawowy teoretyczne systemów agentowych . . . 213
13.2.1. Defincja agenta, cechy agenta, typy agentów . . . 213
13.2.2. Systemy wieloagentowe . . . 214
13.3. FIPA jako standard w systemach agentowych. . . . 215
13.4. Środowiska programowania agentowego . . . 218
13.4.1. JADE . . . 218
13.5. Przykład zastosowania MAS . . . 221
13.5.1. Założenia . . . 221
13.5.2. Narzędzia programistyczne . . . 221
13.5.3. Implementacja . . . 222
13.6. Podsumowanie . . . 223
Literatura . . . 223

14 Metody oceny ryzyka 224
14.1. Wstęp . . . 224
14.2. Polskie normy . . . 225
14.2.1. Ocena ryzyka . . . 225
14.2.2. Metody oceny ryzyka . . . 225
14.3. Bezpieczeństwo na drodze . . . 228
14.3.1. Identyfikacja problemu . . . 229
14.3.2. Ocena bezpieczeństwa na drodze . . . 231
14.4. Strona internetowa – ocena bezpieczeństwa trasy . . . 232
14.4.1. Obliczanie bezpieczeństwa trasy . . . 232
14.4.2. Budowa strony internetowej . . . 233
14.4.3. Format danych . . . 237
14.5. Podsumowanie . . . 238
Literatura . . . 238

15 Komputerowe wsparcie kontroli jakości 239
15.1. Zagadnienia teoretyczne . . . 239
15.1.1. Drzewa decyzyjne . . . 239
15.1.2. C4.5 . . . 241
15.1.3. CART . . . 243
15.1.4. Sortownia owoców . . . 244
15.2. Opis implementacji . . . 248
15.2.1. Moduł analizujący obraz . . . 248

15.2.2. Moduł budowy drzewa . . . 249
15.2.3. Interfejs i obsługa programu . . . 251
15.2.4. Badania . . . 252
15.2.5. Podsumowanie . . . 255
Literatura . . . 256

16 Analiza ryzyka kredytowego 257
16.1. Metody regresji w drążeniu danych . . . 257
16.1.1. Regresja logistyczna, a regresja liniowa . . . 257
16.1.2. Przykład regresji logistycznej . . . 258
16.1.3. Linia regresji logistycznej . . . 259
16.1.4. Funkcja logitowa . . . 259
16.1.5. Estymacja największej wiarygodności . . . 260
16.1.6. Interpretacja wyników regresji logistycznej . . . 260
16.1.7. Interpretacja modelu regresji logistycznej . . . 260
16.2. Modelowanie ryzyka kredytowego z wykorzystaniem regresji logistycznej . . . 261
16.2.1. Zbiór danych . . . 261
16.2.2. Wykorzystanie aplikacji WEKA . . . 262
16.2.3. Model otrzymany z aplikacji WEKA dla zbioru uczącego `german.dat` . . . 263
16.2.4. Wykorzystanie aplikacji do analizy ryzyka kredytowego . . . 265
16.2.5. Podsumowanie . . . 266
Literatura . . . 266

17 Metody wnioskowania w systemach ekspertowych 267
17.1. Zagadnienia teoretyczne . . . 267
17.1.1. Kim jest ekspert? . . . 267
17.1.2. Systemy ekspertowe . . . 268
17.1.3. Metody reprezentacji wiedzy w systemach ekspertowych . . . 269
17.1.4. Klasyfikacja regułowych baz wiedzy . . . 269
17.1.5. Wnioskowanie . . . 270
17.2. Opis projektu . . . 273
17.2.1. Cel . . . 273
17.2.2. Opis działnia programu . . . 273
17.2.3. Instalacja, kompilacja i obsługa programu . . . 273
17.2.4. Opis interfejsu użytkownika . . . 274
17.2.5. Opis implementacji . . . 276
17.2.6. Podsumowanie . . . 281
Literatura . . . 282

SŁOWO WSTĘPNE

Książka *Komputerowe przetwarzanie wiedzy. Kolekcja prac 2010/2011* zawiera dokumentację projektów wykonanych przez studentów V roku Wydziału Elektroniki Politechniki Wrocławskiej, kierunku Automatyka i robotyka, specjalności Robotyka, w semestrze zimowym roku akademickiego 2010/2011, w ramach kursu *Komputerowe przetwarzanie wiedzy*. Historia tego kursu liczy ponad 20 lat i sięga początków specjalności Robotyka; jego twórcą był dr inż. Ireneusz Sierocki, od którego opiekę nad kursem przejął w roku 2005 doc. dr inż. Witold Paluszyński. W latach 2005-2011 prowadzącym kurs był dr inż. Tomasz Kubik. Przy respektowaniu ogólnych wymagań programowych, każdy z prowadzących ten kurs nadał mu indywidualne cechy swojego warsztatu naukowego, co stanowi wielką wartość dydaktyczną. Kurs składa się z wykładu, projektu i seminarium. Celem projektu było przeanalizowanie różnych teorii i algorytmów przetwarzania wiedzy i ich implementacja w postaci programów komputerowych służących do rozwiązania konkretnych zadań. Niniejsza książka obejmuje dokumentację 17 projektów mieszczących się w następujących obszarach tematycznych:

1. Przetwarzania dokumentów
2. Percepcja i ekspresja
3. Losowość jej zastosowanie
4. Bazy danych
5. Systemy ekspertowe
6. Portale społecznościowe
7. Systemy agentowe
8. Analiza ryzyka
9. Kontrola jakości

Treść książki dobrze ilustrują cztery przykładowe projekty.

- P. Batog, M. Pichliński, *Programowe wsparcie procesu budowy tezaurusa*: Autorzy projektu opisali podstawowe zagadnienia związane z budową tezaurusów. Wskazali wykorzystywane standardy i modele tezaurusów oraz dokonali implementacji własnego narzędzia wspierającego proces tworzenia tezaurusa.
- F. Melka, M. Żarkowski, *Gesty dłoni*: Projekt podejmuje temat związany z budową systemu komputerowego wspierającego naukę języka migowego. Autorzy zaproponowali i zaimplementowali własne rozwiązanie (program kompu-

terowy korzystający z kamery internetowej), które pozwala na naukę statycznych gestów dłoni. W implementacji posłużyli się technikami przetwarzania i rozpoznawania obrazów.

- K. Cisek, M. Szlosek, *Gesty marszałka*: Autorzy projektu pokazali wykorzystanie elektronicznego urządzenia zaprojektowanego jako interfejsy gier komputerowych (Wii Remote). Kamera tego urządzenia posłużyła jako element systemu służącego do rozpoznawania gestów wykonywanych przez operatorów na płycie lotniska. W skład tego systemu wchodzi również pałeczka z punktami światła (zaprojektowana wg autorskiego pomysłu) oraz program komputerowy, w którym zaimplementowano algorytmy rozpoznawania i klasyfikacji.
- M. Dubrawski, J. Jewłoszewicz, *Interfejsy programowe społecznościowych portali*: Projekt nawiązuje do najnowszych trendów i technologii wykorzystywanych przy tworzeniu portali społecznościowych. Autorzy zaproponowali i zaimplementowali własne rozwiązanie pozwalające agregować treści pochodzące z różnych serwisów.

Opisy projektów posiadają spójną koncepcję i jednolitą formę, które nadają im charakter zamkniętej całości, od definicji podstawowych pojęć do oceny zastosowanych algorytmów i uzyskanych wyników.

Spodziewam się, że dzięki swojej treści i bardzo starannemu opracowaniu redakcyjnemu książka będzie przydatnym źródłem informacji dla studentów i doktorantów pragnących uzyskać podstawowe rozeznanie w zagadnieniach i metodach komputerowego przetwarzania wiedzy, i stosować je w swojej pracy.

Prof. Krzysztof Tchoń,
opiekun specjalności Robotyka,
Wrocław, wrzesień 2011

ROZDZIAŁ 1

UDOSTĘPNIANIE INFORMACJI W FORMACIE RSS I GEORSS

S. Basiński, P. Michałkiewicz

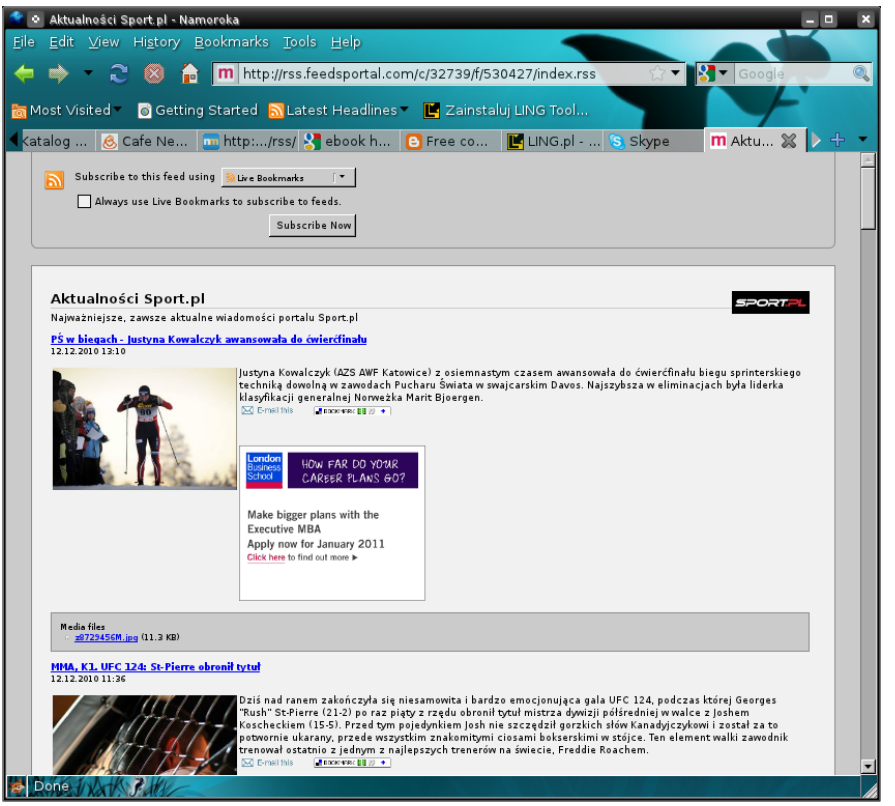
Jednym z ciekawszych procesów związanym z rozwojem ludzkiej cywilizacji jest zdobywanie i przetwarzanie informacji. Powstanie Internetu i technologii sieci web wydatnie przyczyniło się do jakościowej i ilościowej jego zmiany. Rolę tradycyjnych gazet i czasopism oraz radiowych lub telewizyjnych programów informacyjnych coraz częściej i wyraźniej przejmują portale internetowe. Ich użytkownicy, nie wychodząc z domu, mogą za pomocą komputera i połączenia sieciowego publikować samodzielnie przygotowane dokumenty, udzielać się na forach społecznościowych, rozsyłać pocztą elektroniczną wiadomości do znajomych itd. Nigdy przedtem nie można było dotrzeć tak łatwo, szybko i do tak szerokiego kręgu odbiorców. Niestety, łatwość publikowania pociąga za sobą również skutki negatywne. Coraz więcej osób decyduje się na zamieszczenie w sieci Internet przydatnej i niepowtarzalnej, w ich subiektywnym odczuciu, porcji informacji. W ten sposób powstaje *szum informacyjny*, z którego trudno jest odfiltrować użyteczne treści. Na szczęście opracowano metody i technologie, które pozwalają radzić sobie z tym problemem.

W niniejszym rozdziale zostanie opisany jeden z bardzo wygodnych i powszechnie używanych sposobów szerzenia oraz wyszukiwania informacji polegający na wykorzystaniu tzw. kanałów informacyjnych *RSS* i *GeoRSS*. Choć same kanały nie są zjawiskiem nowym, to zwykli użytkownicy Internetu często nie mają pojęcia o ich istnieniu i potencjalnych korzyściach wynikających z ich używania. Oprócz zagadnień podstawowych związanych z RSS i GeoRSS, zaprezentowany zostanie przykład aplikacji stworzonej przez autorów rozdziału, umożliwiający założenie subskrypcji dla stworzonego kanału i wizualizację jego zawartości na mapie.

1.1. Czym jest RSS?

Zanim zacznie używać się jakieś narzędzie wypadałoby zdobyć na jego temat podstawową wiedzę, czyli zaleźć odpowiedź na pytania: *Jak to wygląda? Jak to*

działa? Jakie to ma zastosowanie? Gdy użytkownik posiądzie ten elementarz, w kolejnym kroku może spróbować poszukać odpowiedzi na pytanie: *Jak z tego skorzystać i czy jest to dla mnie przydatne?*. Zacznijmy więc od rzeczy podstawowych. Z pewnością każdy użytkownik Internetu sięgający do jego zasobów za pomocą przeglądarki internetowej miał okazję natknąć się na ikonę: . Jest to *logo RSS*. Ikona ta, jeśli już widnieje na stronie internetowej, prawdopodobnie pełni funkcję przycisku, po kliknięciu na którym użytkownik zostaje automatycznie przeniesiony do *źródła RSS*. Przykładowy widok zawartości takiego kanału pokazano na rys. 1.1:



Rys. 1.1: Widok zawartości kanału RSS w przeglądarce internetowej.

Jedną z podstawowych funkcji RSS jest przechowywanie i szerzenie informacji w sposób prosty i łatwy do obsługi [1]. Do źródeł RSS można odwoływać się za pomocą zwykłych adresów URL i przeglądać ich treść za pomocą przeglądarek internetowych. Jednak nie wszystkie przeglądarki posiadają natywną obsługę RSS. Niektóre wymagają instalacji dodatkowych wtyczek. Istnieją też programy zwane *czytnikami RSS*, np. CafeNews, intraVnews, RSS Reader, RssSpeed, które również pozwalają na dostęp do kanałów informacyjnych. Ostatnio możliwość odczytu

źródeł RSS zaczęto implementować jako funkcję systemów operacyjnych. Patrząc na źródła RSS od strony możliwości publikowanie danych, ich obsługa również wymaga użycia narzędzi programowych. Informacje w kanałach RSS można publikować za pomocą serwisów, aplikacji lub portali internetowych.

Przed przystąpieniem do praktycznego wykorzystania źródeł RSS należy omówić jeszcze jedną, istotną ich własność: możliwość *podziału na kanały tematyczne*. Dlaczego własność ta jest tak istotna? Otóż dzięki niej użytkownik może dokonać selekcji informacji. Selekcja tematów zaś pozwala bronić się przed zalewem niechcianych treści i przytłoczeniem nie tyle ich rangą czy ich istotnością, ale mnogością.

Wprowadzenie mechanizmu tworzenia kanałów tematycznych stało się jedną z największych zalet RSS. Dzięki niemu każdy zainteresowany może dokonać *subskrypcji kanału* za pomocą czytnika lub przeglądarki internetowej. Po tym kroku będzie na bieżąco informowany o wszystkich nowych wiadomościach pojawiających się w subskrybowanym źródle.

1.1.1. RSS - budowa

Przy tworzeniu plików RSS wykorzystywana jest składnia języka XML (*Extensible Markup Language*, [5]) w wersji 1.0. Dlatego pierwsza linijka kodu pliku RSS zawiera informację tej właśnie wersji. Ponadto pliki RSS rozpoznać można po obecności w nich znaczników `<rss>` oraz `</rss>`. W ich obrębie umieszcza się treść i inne znaczniki. Ogólny model zawartości dokumentu XML, będącego dokumentem RSS, przedstawia się następująco:

```
<?xml version="1.0" encoding="UTF-8"?>
<link>http://www.domena.com/main.html</link>
<rss version="2.0">
    ...
</rss>
```

W plikach RSS można zdefiniować jeden lub więcej kanałów informacyjnych. Czyni się to przez zastosowanie specjalnych znaczników. Tak więc każdy element `<rss>` powinien zawierać przynajmniej jeden kanał definiowany za pomocą pary `<channel>` i `/<channel>`. Między tymi znacznikami zamieszczana jest cała treść danego kanału. Z kolei każdy element `<channel>` może zawierać

- `<description>` - element zawierający tekstowy opis zawartości danego kanału,
- `<language>` - element określający język używany w danym kanale,
- `<link>` - element umożliwiający przejście do strony z kanałem,
- `<title>` - element określający tytuł danego kanału.

Dodatkowo `<channel>` można wzbogacić o elementy:

- `<copyright>` - z informacją o prawach autorskich,
- `<docs>` - z linkiem do dokumentacji kanału,

- <image> - z elementami potomnymi, pozwalający załączyć obrazek,
- <item> - przedstawiający stronę lub sekcję strony www. To tu jest zamieszczana właściwa treść kanału. W jednym elemencie <channel> może być do 15 elementów <item>.

Poniżej przedstawiono przykładowy plik źródła RSS z jednym kanałem tematycznym.

```
<?xml version="1.0"?>
<link>http://www.domena.com/main.html</link>
<rss version="2.0">
<channel>
   <title>RSS Title</title>
   <description>Przyklad kodu kanalu RSS</description>
   <link>http://www.domena.com/main.html</link>
   <language>pl</language>
    <image>
      <title>Moje wiadomosci</title>
      <url>http://www.rssmaniak.com/juri/obr1.jpg</url>
      <link>http://www.rssmaniak.com/juri</link>
      <description>Moje wiadomosci</description>
      <width>144</width>
      <height>36</height>
    </image>
    <item>
      <title>Odgarniam snieg</title>
      <description>Znowu spadl snieg</description>
      <link>http://wwww.domena2.com/main.html</link>
    <item>
</channel>
</rss>
```

1.1.2. RSS wersje

Historia RSS sięga roku 1999, a za jego twórcę uważa się Danego Libby z Netscape. W swej niezbyt długiej historii RSS doczekało się trzech różnych kierunków rozwoju. Są nimi:

- Rich Site Sumary (RSS 0.91 i 0.92) – pierwsze popularne wersje powstałe w lipcu 1999 r.
- RDF Site Sumary (RSS 0.90 i 1.0) – wersje powstałe w marcu 1999 r. oraz grudniu 2000 r.. zgodne ze specyfikacją RDF.
- Really Simple Sindycation (RSS 2.0) – wersja wydana we wrześniu 2002 r., obecnie najczęściej używana.

1.2. Wprowadzenie do GeoRSS

Wraz z rozwojem technologii RSS i Atom, coraz ważniejszym stawało się udostępnienie użytkownikom oraz developerom możliwości zamieszczania informacji razem z jej przestrzenną lokalizacją. W odpowiedzi na to zapotrzebowanie stworzono standard GeoRSS. Zapewnia on prosty sposób na załączanie do istniejących kanałów informacji z referencjami przestrzennymi.

Obecnie rozpowszechnione są dwa rodzaje kodowania znaczników GeoRSS: Simple oraz GML. Oba formaty mogą być używane wspólnie z Atomem 1.0, RSS 2.0, RSS 1.0, jak również z każdym innym nie RSS'owym kodowaniem XML. Charakteryzując te dwie wersje można powiedzieć:

- GeoRSS Simple – jest bardzo lekkim formatem, umożliwiającym developerom oraz użytkownikom szybko, prosto i bez większego wysiłku rozszerzyć już istniejące kanały o informacje przestrzenne. Udostępnia on podstawowe obiekty geometryczne, jak: `point`, `line`, `polygon`, `circle`, `box` (odpowiednio: punkt, linia, wielobok, okrąg, prostokąt) oraz obsługuje typowe sposoby kodowania lokalizacji w układzie odniesienia WGS-84.
- GeoRSS GML – jest formatem, w którym wykorzystuje się elementy profilu GML (ang. *Geography Markup Language*) jak: `Point`, `LineString`, `Polygon`, `Envelope`, `CircleByCenterPoint`, odpowiadające elementom z GeoRSS Simple. Elementy te można definiwać w dowolnym układzie odniesienia.

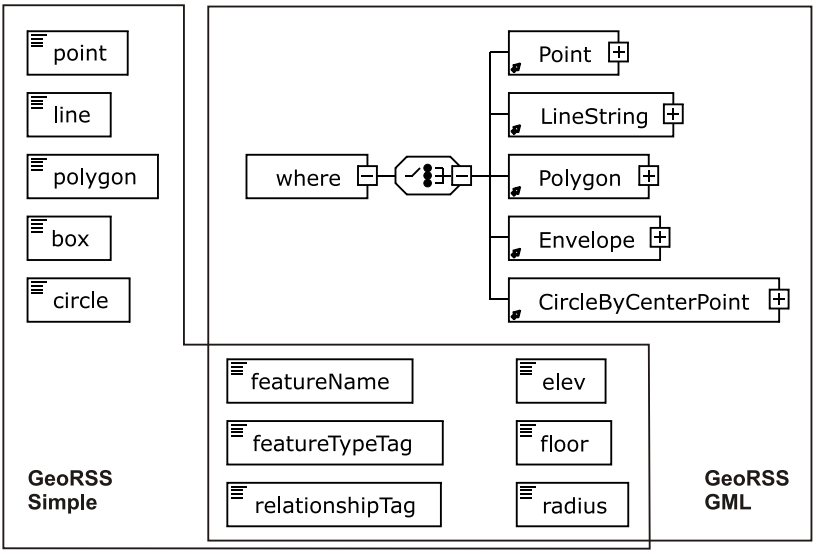
Zarówno w GeoRSS Simple jak i GeoRSS GML można korzystać z dodatkowych elementów:

- `featureType` – pozwala na określenie jednowyrazowym napisem typ znacznika. Na tę funkcję nie nałożono żadnych ograniczeń w celu umożliwienia regionalizacji znacznika. Domyślnie ustawioną nazwą jest `location`,
- `featureName` – umożliwia nadanie nazwy elementowi,
- `relationshipTags` – określa relacje między zawartością publikowanej treści a powierzchnią ziemi. Domyślenie ustawiona jest wartość `is-located-at` co oznacza, że dana treść dotyczy miejsca wskazanego na mapie. Podobnie jak poprzednio nie ma narzuconych żadnych ograniczeń co do treści tego znacznika, prócz przymusu użycia jednoelementowej nazwy,
- `elev` – pozwala na określenie wysokości nad poziomem morza,
- `floor` – określa piętro w budynku,
- `radius` – określa promień lub bufor w metrach wokół geometrycznego obiektu.

Elementy GeoRSS (zgodnie ze schematem `http://www.georss.org/xml/1.1/georss.xsd`) pokazano na rys. 1.2.

1.2.1. GeoRSS Simple

Kodowanie GeoRSS Simple pozwala na maksymalizację zwięzłości zarówno w formatowaniu, jak i opisie danej koncepcji. Każdy obiekt GeoRSS Simle wymaga tylko jednego przestrzennego znacznika. Choć takie podejście prowadzi do utraty zgodności z językiem GML, to jednak istnieją metody pozwalające konwer-



Rys. 1.2: Elementy schematu GeoRSS.

tować format Simple na GML. Należy jednak zauważyć, że dla wielu zastosowań format Simple jest wystarczający i nie ma potrzeby sięgania do bardziej złożonych formatów. Poniżej przedstawiono sposoby zapisu geometrii w kanale GeoRSS:

Punkt: `<georss:point>45.256 -71.92</georss:point>`, para liczb to współrzędne punktu (szerokość i długość geograficzna).

Linia: `<georss:line>45.256 -110.45 46.46 -109.48 43.84 -109.86</georss:line>`, pary liczb oznaczają współrzędne kolejnych punktów tworzących linię.

Box: `<georss:box>42.943 -71.032 43.039 -69.856</georss:box>`, dwie pary liczb oddzielone białym znakiem, wyznaczające prostokątny obszar, pierwsza para to współrzędne lewego, dolnego rogu prostokąta, druga para to współrzędne prawego, górnego rogu.

Wielokąt: `<georss:polygon>45.256 -110.45 46.46 -109.48 43.84 -109.86 45.256 -110.45</georss:polygon>`, musi zawierać co najmniej cztery pary współrzędnych oddzielonych białym znakiem, przy czym ostatnia para musi być taka sama jak pierwsza (wielokąt to obiekt mający przynajmniej trzy różne wierzchołki).

Okrąg: `<georss:circle>42.943 -71.032 500</georss:circle>`, zawiera trzy liczby, z których dwie pierwsze to para określająca współrzędne położenia środka okręgu (szerokość i długość geograficzna), zaś trzecia to wartość jego promienia.

Geometrię można uzupełnić o dodatkową informację:

```
<georss:point>45.256 -110.45</georss:point>
<georss:elev>313</georss:elev>
<georss:point>45.256 -110.45</georss:point>
<georss:floor>2</georss:floor>
<georss:point>45.256 -110.45</georss:point>
<georss:radius>500</georss:radius>
```

1.2.2. GeoRSS-GML

GML jest schematem XML do modelowania i zapisu danych geograficznych. GML dostarcza wiele różnorodnych konstrukcji do przedstawiania obiektów przestrzennych, ich topologii oraz geometrii. Dostarcza też wielu innych możliwości, takich jak deklarowanie systemu współrzędnych odniesienia, czasu, jednostek miar oraz wartości ogólnych. W GeoRSS GML reprezentacja cyfrowa rzeczywistości jest przedstawiana za pomocą serii funkcji. Każda z tych funkcji opisuje jakąś właściwość modelu świata. Przykładem może być zdefiniowanie układu współrzędnych poprzez dodanie do znacznika georss atrybutu `srsName`, np. `srsName=urn:ogc:def:crs:EPSG:6.6:26986`. W wyrażeniu tym `urn:ogc:def:crs` oznacza definicję system odniesienia według standardu OGC, zaś `EPSG:6.6:26986` jest identyfikatorem tego system odniesienia (w tym przypadku STATEPLANE dla Massachusetts Mainland). Poniżej przedstawiono pełny przykład użycia tego systemu odniesienia do opisu wielokąta.

```
<georss:where>
       <gml:Polygon srsName="urn:ogc:def:crs:EPSG:6.6:26986">
          <gml:exterior>
             <gml:LinearRing>
                <gml:posList>
      45.256 -110.45 46.46 -109.48 43.84 -109.86 45.256 -110.45
                </gml:posList>
             </gml:LinearRing>
          </gml:exterior>
       </gml:Polygon>
    </georss:where>
 </entry>
```

Podobnie jak w wersji Simple można definiować podstawowe geometrie, przy czym używa się tu znaczników zaczynają się przedrostkiem `gml`, skojarzonym z przestrzenią nazw GML, oraz zanurza się te znaczniki w elemencie `where`:

Punkt: jest reprezentowany przez znacznik `<gml:Point>`, w którym zagnieżdżony jest element `<gml:pos>` zawierający parę współrzędnych oddzielonych znakiem spacji. Przykładowy zapis punktu w GeoRSS przedstawiono poniżej.

```
<georss:where>
     <gml:Point>
```

```
        <gml:pos>45.256 -71.92</gml:pos>
        </gml:Point>
</georss:where>
```

Linia: jest definiowana z wykorzystaniem znacznika `<gml:LineString>`. Znacznik ten oznacza początek definicji linii, natomiast `<gml:posList>` zawiera przynajmniej dwie pary współrzędnych, przy czym pary nie mogą być oddalone o więcej niż 179 stopni zarówno w szerokości jak i długości geograficznej.

```
<georss:where>
   <gml:LineString>
      <gml:posList>
     45.256 -110.45 46.46 -109.48 43.84 -109.86
      </gml:posList>
   </gml:LineString>
</georss:where>
```

Box: określa prostokątny element. Często jest używany do zaznaczania większych obszarów mapy. Definicja zawiera element `<gml:Envelope>` oraz jego potomków `<gml:lowerCorner>` i `<gml:upperCorner>`. Wartością pierwszego jest para współrzędnych lewego dolnego rogu, natomiast drugi określa współrzędne prawego górnego rogu.

```
<georss:where>
   <gml:Envelope>
      <gml:lowerCorner>42.943 -71.032</gml:lowerCorner>
      <gml:upperCorner>43.039 -69.856</gml:upperCorner>
   </gml:Envelope>
</georss:where>
```

Wielokąt: jest opisywany przy pomocy elementu `<gml:Polygon>` wraz z jego potomkami `<gml:exterior>`, `<gml:LinearRing>` oraz `<gml:posList>`. Lista współrzędnych musi zawierać co najmniej cztery elementy, z których pierwszy i ostatni muszą być takie same. Dodatkowo, żadna z dwóch par nie może różnić się między sobą o więcej niż 179 stopni zarówno w szerokości, jak i długości geograficznej.

```
<georss:where>
   <gml:Polygon>
      <gml:exterior>
         <gml:LinearRing>
            <gml:posList>
   45.256 -110.45 46.46 -109.48 43.84 -109.86 45.256 -110.45
            </gml:posList>
         </gml:LinearRing>
      </gml:exterior>
```

```
    </gml:Polygon>
  </georss:where>
```

1.3. OpenLayers

Do wizualizacji danych geograficznych można wykorzystać bibliotekę `OpenLayers` [2]. Pozwala ona publikować mapy na stronach internetowych przy wykorzystaniu języka `javaScript`. Publikowana mapa powstaje na poprzez nakładanie kolejnych warstw związanych z danymi przestrzennymi lub usługami geoprzestrzennymi. W rozwiązaniach budowanych na bibliotece OpenLayers często wykorzystuje się usługi WMS (ang. *Web Map Server*). Takie podejście, poza ułatwieniem aktualizacji mapy, pozwala na agregację danych pochodzących z rozproszonych źródeł oraz pozwala na implementację cienkiego klienta. W ogólnym zarysie działanie aplikacji z komponentami biblioteki OpenLayers polega na wysyłaniu żądań do usługi przestrzennej z parametrami określającymi obszar zainteresowania na mapie, oraz zobrazowaniu odpowiedzi będącej wybranym wycinkiem mapy. Dzięki takiej komunikacji OpenLayers pozwala dołączyć do mapy dodatkowe elementy w sposób dynamiczny (oraz je usunąć). OpenLayers ponadto umożliwia umieszczanie na mapie różnych znaczników, w tym znaczników definiowanych w GML, KML, GeoRSS. Obsługa znaczników GeoRSS przez bibliotekę OpenLayers jest jak na razie ograniczona tylko do elementu `point`. Podsumowując, OpenLayers umożliwia:

- wybór tekstury mapy,
- obsługę formatów GeoRSS, KML, GML, GeoJSON,
- wykonywanie operacji na mapie, np. przybliżania, przesuwania.

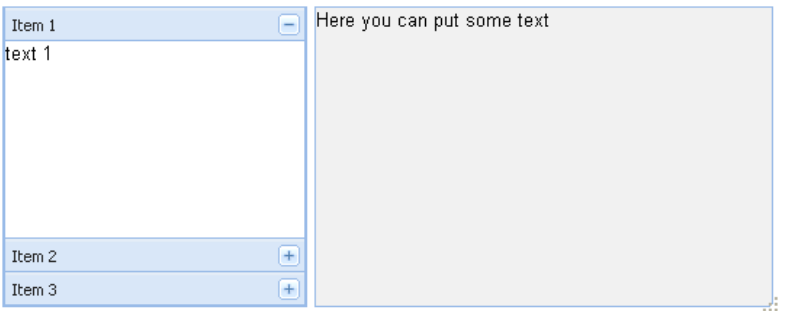
Poniżej przedstawiono przykładowy kod, który pozwala wyświetlić na stronie internetowej mapę złożoną z dwóch warstw: warstwy związanej z usługą WMS oraz warstwy związanej ze źródłem GeoRSS.

```
// tworzy element mapa
  map = new OpenLayers.Map('map', {maxResolution:'auto'});
// tworzy, dodaje i centruje warstwę mapy (standardowa mapa)
  layer = new OpenLayers.Layer.WMS("OpenLayers WMS",
    "http://vmap0.tiles.osgeo.org/wms/vmap0", {layers: 'basic'});
  map.addLayer(layer);
/ map.setCenter(new OpenLayers.LonLat(0, 0), 0);
// dodaje konrolki do włączania/wyłączania warstw
  map.addControl(new OpenLayers.Control.LayerSwitcher());
// tworzy i dodaje warstwę georss z pliku 'georss.xml'
  var newl = new OpenLayers.Layer.GeoRSS('GeoRSS', 'georss.xml');
  map.addLayer(newl);
```

1.4. ExtJS

Biblioteka `ExtJS`, podobnie jak OpenLayers, jest biblioteką javaScript. Pozwala na budowę okien dialogowych i kontrolek w aplikacjach webowych . Biblioteka obsługuje, m.in.:

- pole tekstowe oraz pole wprowadzania tekstu,
- pole daty,
- rozwijane menu,
- radio i checkbox,
- grid,
- suwaki,
- regionalizacja paneli,
- klasy do zarządzania formatami JSON oraz XML.



Rys. 1.3: Wynik wykonania przykładowego skryptu ExtJS.

Na rys. 1.3 pokazano wynik wykonania poniższego skryptu ExtJS:

```
<script type="text/javascript">
  Ext.onReady(function() {
 // tworzy item 1
    var item1 = new Ext.Panel({
          title: 'Item 1', html: 'text 1', cls:'empty'
            });
 // tworzy item 2
    var item2 = new Ext.Panel({
           title: 'Item 2',  html: 'text 2', cls:'empty'
            });
 // tworzy item 3
    var item3 = new Ext.Panel({
          title: 'Item 3', html: 'text 3', cls:'empty'
            });
 // grupuje itemy od 1 do 3 na jednym panelu
    var accPanel = new Ext.Panel({
```

```
          region:'west', margins:'5 0 5 5', split:true,
          width: 210, layout:'accordion',
          items: [item1, item2, item3]
            });
 // wyświetla zgrupowane panele z dodatkowym oknem w centrum
    var viewport = new Ext.Viewport({
          layout:'border',
          items:[
               accPanel, {
                  region:'center',
                  margins:'5 5 5 0',
                  cls:'empty',
                  bodyStyle:'background:#f1f1f1',
                  html:'Here you can put some text'
                }]
            });
    });
</script>
```

1.5. GeoExt

Biblioteka `GeoExt` jest rozszerzeniem biblioteki ExtJS o elementy wspierające obsługę map [3]. Pozwala ona budować aplikacje internetowe z elementami wykorzystującymi między innymi bibliotekę OpenLayers. Jednym z dodanych elementów jest panel `gx_mappanel`. Pozwala on obsługiwać zarówno mapy OpenLayers jak i również GoogleMaps. Przykład demonstrujący wyświetlenie panelu mapy przy pomocy biblioteki GeoExt pokazano na rys. 1.4. Jest to efekt wywołania poniższego skryptu:

```
//rozpoczęcie konstrukcji panelu
Ext.onReady(function() {
  Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
  //tworzenie mapy przy pomocy biblioteki OpenLayers
 var map = new OpenLayers.Map();
    var layer = new OpenLayers.Layer.WMS(
        "Global Imagery",
"http://maps.opengeo.org/geowebcache/service/wms",
        {layers: "bluemarble"}
    );
// dodanie warstwy z mapą
    map.addLayer(layer);
// stworzenie panelu na mapę
    mapPanel = new GeoExt.MapPanel({
```

```
        title: "GeoExt MapPanel",
        renderTo: "mappanel",
        stateId: "mappanel",
        height: 400,
        width: 600,
 // umieszczenie w nim stworzonej przez nas mapy
        map: map,
        center: new OpenLayers.LonLat(5, 45),
        zoom: 4,    });
});
```

Rys. 1.4: Wywołanie przedstawionego przykładowego kodu GeoExt.

1.6. Przykładowe zastosowanie GeoRSS

W celu przetestowania i oceny możliwości wykorzystania znaczników GeoRSS i RSS wykonano przykładowy projekt. Założeniem projektowym było stworzenie strony internetowej ułatwiającej poruszanie się nowych studentów Politechniki Wrocławskiej po jej kampusie. Strona ta miała umożliwiać:

- wczytanie pliku opisanego znacznikami GeoRSS będącego planem zajęć studenta,
- stworzenie okna umożliwiającego wyświetlanie ważnych miejsc z kampusu Politechniki Wrocławskiej z ich nazwami i najważniejszymi informacjami. Założono dodatkowo, że samo okno powinno pozwalać na:

  - sortowanie rekordów,
  - wyłączanie jednej z dwóch kolumn,
  - zmiana kolejności wyświetlania kolumn.
- wyświetlenie ważnych miejsc Politechniki Wrocławskiej na mapie. Zgodnie z założeniami projektowane rozwiązanie miało umożliwiać zmianę rozmiaru okien aplikacji internetowej oraz możliwość przybliżania/oddalania, a także przesuwania mapy.

Jedną z zalet biblioteki OpenLayers jest łatwe operowanie warstwami. Powstała więc aplikacja umożliwiająca wybór jednej z dwóch warstw. Pierwszą z nich była `vmap0`, a drugą `osm`. Na przykładzie tej drugiej pokazano możliwość zmiany układu współrzędnych odniesień geograficznych dla prezentowanej warstwy.

Podczas prac projektowych napotkano na problemy implementacyjne podczas wykorzystywania biblioteki OpenLayers wraz z biblioteką GeoExt. Ich powodem była niedostateczna komunikacja między tymi bibliotekami. Zaobserwowane problemy to, np.:

- przy zmianie współrzędnych niektóre elementy związane biblioteką przestają wyświetlać poprawnie znaczniki (okno `storage` z biblioteki GeoExt nie współpracuje w pełni ze zmianą współrzędnych),
- sprzężenie okna `storage` z mapą nastręczało kłopotów (zablokowanie wyskakujących okienek z opisem znacznika).

Ostatecznie powstała aplikacja pozwalająca na:

- wczytywanie poprzez okno formularza pliku opisanego przy pomocą znaczników GeoRSS/RSS,
- prezentację na mapie znaczników GeoRSS,
- wyświetlanie na mapie oraz w panelu pliku z danymi o budynkach Politechniki Wrocławskiej,
- wybór warstwy mapy między standardową a OSM,
- wyłączanie wyświetlania na mapie wczytanych plików,
- przybliżanie/oddalanie i przesuwanie widocznej części mapy,
- przybliżanie wielkości skali przy pomocy linijki,
- odczytywanie położenia kursora na mapie (klasyczny system opisu współrzędnych),
- szybkie przesuwania widocznej części mapy przy pomocy podglądu z miniaturową mapą.

Powyższe funkcje związane są z obsługą mapy. Jednakże aplikacja dodatkowo pozwalała na:

- wczytywanie plików z adresów spoza serwera aplikacji,
- parsowanie pliku z danymi po tytule i opisie,
- zmianę rozmiaru okien dialogowych,
- zmianę kolejności kolumn w oknie informacyjnym oraz możliwość sortowania informacji,

- wielokrotne uruchomienie poprzez mechanizm kart w przeglądarce internetowej.

1.7. Szczegóły implementacyjne

Aby zrealizować wymagania funkcjonalne w implementacji wykorzystano funkcje opisane poniżej.

1.7.1. Element Mapa

Do zmiany układu współrzędnych geograficznych w jakich wyświetlana jest mapa posłużono się funkcją `Projection`:

```
var wgs = new OpenLayers.Projection("EPSG:4326");
```

Aby stworzyć okno z mapą wystarczy użyć funkcji `Map` jak w następującym przykładzie:

```
map = new OpenLayers.Map("map_box", {
       //ustawia kontrolki mapy na domyślne
        controls: [],
       //używa zdefiniowanego układu współrzędnych
        displayProjection: wgs
});
```

Do stworzenia i dołączenia warstw do mapy wykorzystano następujący kod:

```
// tworzy warstwe pobierana z serwisu Open Street Map
var osm = new OpenLayers.Layer.OSM("OpenStreetMap");

// tworzy warstwe mapy VMAP0 z danego serwisu WMS
layer = new OpenLayers.Layer.WMS( "OpenLayers WMS",
"http://labs.metacarta.com/wms/vmap0",
{layers: 'basic'} );
// dodajemy warstwy do OKNA mapy w zadanej kolejności
map.addLayers([osm, layer]);
```

1.7.2. Kontrolki mapy

Biblioteka OpenLayers umożliwia zarządzanie kontrolkami na mapie. Oto niektóre z możliwych ustawień:

- dodatkowy pasek do przybliżania i oddalania mapy

  ```
  map.addControl(new OpenLayers.Control.PanZoomBar());
  ```

- przełącznik między warstwami (na prawo u góry)

  ```
  map.addControl(
   new OpenLayers.Control.LayerSwitcher({'ascending':false}));
  ```

- linijka ze skalą w lewym dolnym rogu

```
map.addControl(new OpenLayers.Control.ScaleLine());
```

- wyświetlanie pozycji kursora

```
map.addControl(new OpenLayers.Control.MousePosition());
```

- kontrolka umozliwiająca podgląd mapki w szerszym kontekście

```
map.addControl(new OpenLayers.Control.OverviewMap());
```

- kontrolka umożliwiająca skalowanie mapy myszą

```
map.addControl(new OpenLayers.Control.Navigation());
```

1.7.3. Warstwa znaczników

Aby dodać znaczniki np. GeoRSS do mapy należy użyć komendy

```
var newl = new OpenLayers.Layer.GeoRSS( 'GeoRSS','georss.xml' );
```

powoduje ona wczytanie pliku `georss.xml` do mapy i wyświetlenie znaczników GeoRSS na mapie. OpenLayers umożliwia również wczytywanie innego typu znaczników. Oto przykłady wywołania znaczników innego typu:

- Dodawanie znaczników GeJSON do mapy "map".

```
var geojson_format = new OpenLayers.Format.GeoJSON();
var vector_layer = new OpenLayers.Layer.Vector();
map.addLayer(vector_layer);
vector_layer.addFeatures(geojson_format.read(featurecollection));
// gdzie 'featurecollection' jest wcześniej zdefiniowaną
// zmienną przechowującą znaczniki GeoJSON
```

- W tym przykladzie dodano znacznik opisujący wielokąt przy pomocy składni GML. W pliku `polygon.xml` znajduje się opis tego wielokąta.

```
map.addLayer(new OpenLayers.Layer.GML("GML", "polygon.xml"));
```

- Dodawanie znaczników z pliku opisu KML.

```
map.addLayer(new OpenLayers.Layer.GML("KML", "lines.kml",
```

1.7.4. Kontener

Do przechowywania danych GeoExt posiada specjalny obiekt `FeatureStore`.

```
myStore = new GeoExt.data.FeatureStore({
// zmienna przechowujaca wczytywane dane wektorowe
     layer: newl,
     fields: [
 // nazwy i typy parsowane z pliku
{name: 'title', type: 'string'},
```

```
{name: 'description', type: 'string'}
        ],
        proxy: new GeoExt.data.ProtocolProxy({
    protocol: new OpenLayers.Protocol.HTTP({
//adres i format pliku parsowanego
url: myUrl,
format: new OpenLayers.Format.GeoRSS()
            })
}),
   //automatyczna zaladowanie i zapamiętanie danych
      autoLoad: true, autoSave: true,
    });
```

Pozwala on na wczytywanie danych z pliku (z jego parsowaniem) oraz umożliwia ich związanie z mapą w postaci warstwy wektorowej. Poniżej przedstawiono przykład stworzenia warstwy wektorowej.

```
var vector_layer = new OpenLayers.Layer.Vector();
```

1.7.5. Panel

Poniżej przedstawiono przykład utworzenia panelu przy użyciu biblioteki ExtJS. Opis poszczególnych opcji umieszczono w komentarzach.

```
myPanel = new Ext.Panel({
//zdefiniowanie typu układu
layout: 'border',
// item definiuje obiekty użyte do konstrukcji
items: [
{//każdy nowy obiekt jest zamknięty w nawiasie klamrowym
//xtype definiuje typ obiektu 'gx_mappanel' z GeoExt
xtype:'gx_mappanel',
// map odwołuje się do wczytanej mapy z OpenLayers
map: map,
// wyśrodkowanie
region: 'center',
// wycentrowanie mapy
center: new OpenLayers.LonLat(0,0),
// dwukrotne powiększenie mapy
zoom: 2,
// zezwolenie na przesuwanie granic panelu
split: true
},
{// panel typu Grid  we wschodniej części okna
xtype: 'grid',region: 'east',
```

```
// szerokość okna
width: 400,
// przypisuje zmienną z danymi
store: myStore,
// definiuje nazwy pól pobranych ze zmiennej store
columns: [
 {header: 'Title', dataIndex: 'title'},
 {header: 'Description', dataIndex: 'description',width: 296}],
// zezwolenie na przesuwanie granic panelu
split: true,
}],
// wysokość całego okna i unikalna nazwa całego panelu
height:570,renderTo:'panelDiv'
});
});
```

1.7.6. Uwagi na temat wdrożenia

Aplikacja powinna operować na danych otagowanch znacznikami RSS i GeoRSS, zawartych w stronach internetowych. Stąd stosownym było jej zaimplementowanie jako aplikacji WEB'owej. Finalna wersja programu powinna zostać osadzona na serwerze WWW podłączonym do internetu. Ponieważ w stworzonym rozwiązaniu wykorzystywany są zewnętrzne usługi serwujące mapy, koniecznym stało się przygotowanie dla aplikacji odpowiedniej konfiguracja ustawień proxy. Do tego celu można użyć odpowiednich modułów Apache2 oraz plików konfiguracyjnych proxy.cgi.

1.8. Podsumowanie

Kanały RSS i GeoRSS są obecniej jednym z lepszych źródeł krótkich i zwięzłych informacji. Ponadto GeoRSS, jako aplikacja XML, jest stosowany do opisu danych geograficznych przy pomocy prostych znaczników. Dzięki takiemu podejściu możliwym jest dekodowanie informacji o danych geograficznych przez systemy informatyczne. Umożliwienie gromadzenia, obróbki, wizualizacji oraz przetwarzania tego typu danych komputerom prowadzi do budowania systemów autonomicznych wykorzystujących wiedze o położeniu. Taką wiedzę można udostępnić autonomicznym pojazdom do celów nawigacji, jak również budować systemy gromadzące dane i przypisujące im informacje o współrzędnych geograficznych. Ten drugi sposób został użyty przez firmę Google do prezentacji danych o powodzi z wiosny 2010 roku.

Dzięki możliwości przerzucenia części analizy danych geograficznych wraz z ich opisem można przyspieszyć ekstrakcje danych z wielkich baz wiedzy. Takie systemy mogą nie tylko zmniejszyć koszty obróbki danych, ale również ocalić życie ludzkie. Przykładem mogą być systemy gromadzące i przetwarzające dane o aktywności sejsmicznej ziemi. Liczba danych i prędkość ich napływania sku-

tecznie uniemożliwiają na bieżąco wyciąganie z nich informacji przez człowieka. Natomiast systemy wyposażone w metody łatwego i zwięzłego opisu danych wraz z ich opisem mogą znacznie szybciej wyciągnąć wnioski i wystosować odpowiednie komunikaty do użytkownika systemu.

Literatura

[1] S. Holzner: *Sekrety RSS* Wydawnictwo HELION, Warszawa, (2007).
[2] OpenLayer v 3.3.1 Java Script library: http://www.openlayers.org/.
[3] GeoExt v 1.0 Java Script library: http://www.geoext.org/.
[4] OpenGIS Catalogue Services Specification 2.0.2 .
[5] W3C Extensible Markup Language (XML) 1.1 (Second Edition)., (2006).

ROZDZIAŁ 2

WYSZUKIWANIE PRZESTRZENNO-CZASOWE

A. Bernad, Ł. Juszkiewicz

W rozdziale tym zostanie przedstawione zagadnienie wyszukiwania przestrzenno-czasowego w świetle możliwości rozszerzenia funkcji wyszukiwania w systemie zarządzania dokumentami DMS (ang. *Document Management System*). Przedstawione zostaną ogólne zasady wyszukiwania dokumentów oraz rola ontologii czasowo-przestrzennej. Wyjaśniona zostanie znaczenie systemów DMS w codziennym życiu.

2.1. Wyszukiwanie dokumentów

Ze względu na rosnącą w bardzo szybkim tempie ilość informacji przechowywanej w systemach komputerowych przez pojedyncze osoby czy instytucje, efektywne zarządzanie tą informacją oraz możliwość jej analizy w konteście czasowo-przestrzennym jest niezwykle gorącym temat. O ile samo gromadzenie i przechowywanie danych jest stosunkowo proste do zaimplementowania, o tyle skuteczne i szybkie wyszukiwanie danych jest zagadnieniem dużo bardziej skomplikowanym. Obserwując publikacje na ten temat można zauważyć, że w ostatnich latach wykonano wiele badań nad skutecznością istniejących rozwiązań i oraz metodami jej zwiększenia.

W większości przypadków przy wyszukiwaniu potrzebnych informacji stosuje się wyszukiwanie pełnotekstowe. Jest to dobre rozwiązanie, jeśli jakaś fraza jest wyszukiwana w jednym czy kilku tylko dokumentach. Jednak w przypadku dużych zbiorów danych jest to bardzo złożona, a czasami zawodna operacja. Wyniki wyszukiwania mogą być dalekie od oczekiwań, jeśli zapytanie zostanie źle skonstruowane. Chodzi tu o to, że wynik wyszukiwania w większości przypadków zostanie znaleziony na zasadzie prostego dopasowania słów kluczowych w analizowanych tekstach. Przez to wśród rezultatów brak będzie tych dokumentów, które opisują interesujący użytkownika temat, ale za pomocą słów bliskoznacznych. Przykładowo, wynik wyszukiwania dokumentów zawierających słowo „dom" nie będzie zawierał dokumentów, które zawierają słowo „mieszkanie". Zapewne w wielu przypadkach wystarczyłoby zmodyfikować parametry wy-

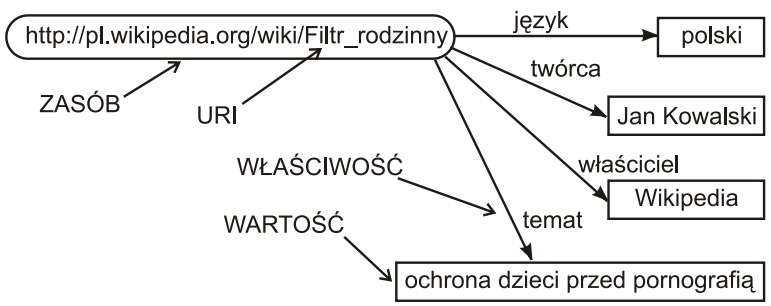
szukiwania, zamieszczając w zapytaniach całą listę bliskoznacznych słów. Jednak konstrukcja takich zapytań może być trudna i nieefektywna.

Podczas przeszukiwania zasobów można również postawić sobie za zadanie wyszukanie tych dokumentów, które zostały stworzone tylko w określonym przedziale czasu czy też dotyczą określonej lokalizacji w przestrzeni. Jeśli dokumenty będą miały w swojej treści informacje o lokalizacji, można spodziewać się poprawnych wyników wyszukiwania, przynajmniej w części związanej z przestrzenią. Ale jak w takim razie wybrać dokumenty z ostatnich 5 zamiast 15 lat? Wypisywanie wszystkich dat dzień po dniu jest dość męczące dla dłuższych interwałów. Dodatkowo należy pamiętać o niejednoznacznym formacie daty. Wydaje się więc, że postawione zadanie nie należy do łatwych. Aby go rozwiązać, zamiast wyszukiwania pełnotekstowego stosuje się wyszukiwanie czasowo-przestrzenne na bazie specjalnie w tym celu utworzonych indeksów.

2.2. Wyszukiwanie w oparciu o metadane

Sposobem na umożliwienie wyszukiwania danych przy pomocy bardziej zaawansowanych kryteriów jest wprowadzenie metadanych, czyli inaczej „danych o danych". Przykładem może tu być katalog biblioteczny zawierający podstawowe informacje o książkach, umożliwiający ich wyszukiwanie. Tagi opisujące krótko treści stron internetowych to również proste metadane.

Możliwości jakie oferowane są przez wyszukiwarkę w dużym stopniu zależą właśnie od metadanych, zawartych w nich informacji oraz ich struktury. Pojawia się tutaj problem w jaki sposób je zapisywać, aby były one czytelne dla programów komputerowych i w łatwy sposób poddawały się szybkiej maszynowej obróbce. Jedną z prób usystematyzowania zapisu mtetadanych jest język RDF (ang. *Resource Description Framework* [1]). Został on opracowany przez WC3, a jego składnia bazuje na XML. w języku RDF każdy zasób opisywany jest za pomocą wyrażenia składającego się z trzech elementów: podmiotu, orzeczenia/predykatu (własności) i dopełnienia/obiektu (wartości), tak jak to pokazno na rys. 2.1.



Rys. 2.1: Przykład grafu RDF zawierającego opis właściwości filtru rodzinnego.

Rozszerzeniem języka RDF jest OWL (ang. *Web Ontology Language*, [2]). OWL jest używany do tworzenia ontologii, czyli formalnej reprezentacji wiedzy, na

którą składają się zbiory pojęć i relacji pomiędzy nimi. Opis bazujący na ontologii stanowi jeden ze sposobów tworzenia metadanych. Jego zaletą jest struktura logiczna, pozwalająca na wnioskowanie, co czyni wyszukiwanie danych „inteligentniejszym".

Jak już wspomniano często zachodzi potrzeba wyszukania danych, na przykład dokumentów, wg kryterium czasowego, przestrzennego lub obu jednocześnie. Znalezienie dokumentu opisującego upadek meteorytu w okolicach Oleśnicy w nocy z 21 na 22 października przy pomocy wyszukiwania pełnotekstowego może być trudne, jeśli w treści dokumentu nie są wprost zawarte informacje o dacie i miejscu. Należy więc skorzystać z systemu, których potrafi indeksować dane w kategoriach czasu i przestrzeni, a następnie dokonywać wyszukiwania przy ich pomocy. Takim systemem jest na przykład portal Filckr, który umożliwia dodawanie zdjęć, a następnie ich opisywanie przy pomocy informacji geograficznych. Pozwala to na łatwe znalezienie zdjęć związanych ze wskazanym miejscem na mapie.

Pożądane jest również wykorzystanie pewnych logicznych własności informacji czasowo-przestrzennej — wyszukiwanie w określonych przedziałach czasu, regionach kraju, świata. Jeśli poszukiwane są dokumenty dotyczące Polski, w wynikach wyszukiwania powinny znaleźć się również dokumenty oznaczone jako dotyczące konkretnych miejsc czy regionów w Polsce: miasta Wrocławia, powiatu lubuskiego itd.

Rozwijanie systemów posiadających wyżej opisaną funkcjonalność jest ostatnio bardzo intensywne. Ilość informacji znajdujących się w sieci Internet oraz najróżniejszych bazach danych jest ogromna. Większe możliwości szybkiego wyszukiwania potrzebnych informacji oznaczają zwykle konkretny zysk ekonomiczny. Jednym z rodzajów systemów, w których można zastosować wyszukiwanie przestrzenno-czasowe, są systemy zarządzania dokumentami.

2.3. Systemy DMS

W obecnych czasach, gdy papierowe dokumenty zostają zamieniane na ich cyfrowe odpowiedniki, tradycyjne archiwum z regałami teczek zastąpił komputer z odpowiednim oprogramowaniem. Systemy DMS to przykład takiego oprogramowania, przeznaczonego do efektywnego zarządzanie dokumentami.

2.3.1. Czym są systemy DMS

Systemy DMS, czyli systemy zarządzania dokumentami w postaci cyfrowej, pozwalają na wydajne administrowanie nawet dużymi zbiorami danych. Zalet wdrożenia takiego systemu jest co najmniej kilka. Bardzo szczegółowe, a jednocześnie wydajne kategoryzowanie informacji powoduje, że odszukanie określonych danych zamiast standardowych minut, godzin i dni zajmuje najwyżej kilkanaście sekund. Po przeniesieniu archiwum firmy z metalowej szafy na dysk komputera skończą się telefony i e-maile do poszczególnych pracowników z pytaniami o miejsce odłożenia jakiejś teczki. Odpowiednie skonfigurowanie reguł dostępu i przydzielanie zadań zwiększy nie tylko poziom bezpieczeństwa danych,

ale także podniesie wydajność pracy. Nie bez znaczenia będzie również możliwość uzyskania dostępu do dokumentów z dowolnego miejsca i o każdej porze.

2.3.2. Ograniczenia systemów DMS

W dzisiejszym stanie prawnym całkowite zrezygnowanie z papierowych dokumentów na rzecz ich elektronicznych wersji nie jest możliwa. Informatyzacja postępuje jednak wytrwale, a już dziś elektroniczna wersja papierowego archiwum może znacznie podnieść komfort pracy i zwiększyć jej wydajność.

Dwoma podstawowymi kwestiami, które mogą wzbudzić wątpliwości przyszłych użytkowników systemów zarządzania dokumentami, są koszty wdrożenia oraz bezpieczeństwo danych. Jakkolwiek w tym pierwszym wypadku odpowiedź jest bardzo prosta - poniesione wydatki zwrócają się szybko w postaci znaczących oszczędności czasu personelu - to problemu zabezpieczenia przechowywanych informacji z pewnością nie można lekceważyć.

Przystępując do korzystania z DMS należy więc przygotować i wdrożyć politykę bezpieczeństwa danych - chodzi tu przede wszystkim o regularne wykonywanie kopii zapasowych oraz regulację fizycznego i zdalnego dostępu do serwera bazodanowego. Wszystko po to, by nie sprawić sobie przykrej niespodzianki, gdy wszystkie dokumenty nagle odejdą w cyfrowy niebyt, na przykład z powodu uszkodzenia dysku.

2.3.3. Rozwiązania komercyjne i Open Source

Rozwiązania komercyjne dostępne na rynku to często produkty kompleksowe - poza samym przechowywaniem cyfrowych wersji dokumentów oferowane są w pakiecie usługi skanowania, katalogowanie, a nawet magazynowania oryginałów w bezpiecznych i chronionych pomieszczeniach. Często są to produkty skalowane i parametryzowane do potrzeb ich użytkowników.

Do pracy grupowej duże przedsiębiorstwa wykorzystują najczęściej takie rozwiązania, jak Microsoft SharePoint, DocPoint, pakiet IBM FileNet P8 Platform czy EMC Documentum. Ich wspólną cechą są jednak wysokie koszty wdrożenia - często nie do zaakceptowania przez małe firmy czy mikroprzedsiębiorstwa. Alternatywą dla takich podmiotów mogą być aplikacje z otwartym kodem źródłowym. Oczywiście należy pamiętać, że i w tym wypadku nieuniknione jest poniesienie pewnych kosztów - ktoś przecież musi system zainstalować, skonfigurować i przeprowadzić szkolenie w zakresie jego obsługi. Wydatki mogą być jednak mniejsze, a korzyści z wdrożenia DMS - porównywalne. Poniżej opisano jeden z programów z tego sektora rynku.

2.3.4. LogicalDOC jako przykład systemu DMS

Program jest dostępny w dwóch wersjach - płatnej oraz Open Source (Community Edition). w porównaniu z wersją płatną aplikacja z otwartym źródłem (na elastycznej licencji LGPL v3) jest uboższa o funkcje: rozpoznawania tekstu (OCR), współpracy z dokumentami MS Office 2007 (ze starszymi wersjami działa poprawnie) oraz zdalnego importu udostępnionych dokumentów. LogicalDOC

został stworzony przy wykorzystaniu technologii J2EE, powinien więc działać na każdym serwerze wyposażonym w ten kontener. Domyślnie oprogramowanie można pobrać wraz z serwerem Tomcat 6.0.16. Jako bazę danych można wykorzystać jeden z popularnych systemów, na przykład MySQL lub Oracle 9i/10i. Instalacja sprowadza się do rozpakowania pliku archiwum, uruchomieniu serwera i ustalenia parametrów połączenia z bazą danych. Program nie został spolszczony, choć architektura systemu pozwala na dodanie samodzielnie wykonanego tłumaczenia dla każdego języka.

Pliki przechowywane są w folderach i opisywane za pomocą prostych oznaczeń, takich jak autor i data utworzenia oraz słów kluczowych. Te ostatnie pozwalają na grupowanie i przeglądanie dokumentów pochodzących z różnych folderów. Podczas dodawania pliku można wybrać opcję automatycznego sugerowania słowa kluczowego, choć jej implementacja pozostawia dużo do życzenia. Obsługa interfejsu jest mało skomplikowana, zdarzają się jednak błędy - na przykład zbyt długie nazwy folderów potrafią doprowadzić do zupełnej dezorganizacji głównego okna programu. Zauważalne są także pewne braki w ergonomii - foldery można co prawda przeglądać w wygodnej strukturze drzewa, ale wymaga to kilku dodatkowych zabiegów.

Standardowo użytkownicy systemu należą do grup, do których przypisywane są prawa dostępu (odczyt, zapis) związane z plikami oraz folderami. Aby skorzystać z konkretnego pliku, należy mieć dostęp do wszystkich folderów, w których jest on umieszczony - na przykład `Umowy -> Umowy kupna -> J.P. Kowalscy s.c.`. Brak dziedziczenia praw może utrudniać codzienne korzystanie z systemu.

LogicalDOC udostępnia wersjonowanie plików. Można przy tym dodać nową wersję główną (np. 1.0-2.0), poboczną (1.0-1.1) albo pozostawić oznaczenie bez zmian. Cały czas zachowuje się oczywiście dostęp do poprzednich wersji. Co ważne, program potrafi odczytać standardowe formaty plików biurowych (pakietów MS Office, OpenOffice, Adobe PDF) - dzięki czemu można w nim korzystać z wyszukiwania pełnotekstowego. Ciekawym pomysłem, pozwalającym zaoszczędzić dużo czasu, jest także importowanie struktury plików w katalogach za pomocą archiwum ZIP.

Społeczność deweloperów LogicalDOC jest dość prężna. Raportowane błędy poprawiane są stosunkowo szybko (najpóźniej w ciągu 30 dni od zgłoszenia), a nowe wersje produktu pojawiają się co kilka miesięcy - ostatnia z nich datowana jest na październik 2008 roku. Mimo to błędy interfejsu, problemy z ergonomią i brak choćby najprostszego systemu zarządzania obiegiem dokumentów nie pozwalają wystawić aplikacji zbyt wysokiej noty.

2.4. Ontologia *Spime*

Ontologia *Spime*, o której będzie mowa w dalszej części tego rozdziału, powstała jako rezultat projektu realizowanego przez studentów Wydziału Elektroniki Politechniki Wrocławskiej. Nazwa ontologii: *Spime* to słowo powstałe z połączenia dwóch słów: *space* i *time*. Ontologia została napisana w języku OWL przy użyciu narzędzia TopBraid Composer. Ontologia ta częściowo opiera się

na OWL-Time. w kolejnych podrozdziałach opisano efekt prac autorów związanych z modyfikacją tej ontologii oraz zastosowaniem do indeksowania czasowo-przestrzennego dokumentów w ramach systemu LogicalDoc.

2.4.1. Podstawowe klasy

- `Document` - jest to podstawowa klasa w ontologii *Spime*. Instancja tej klasy reprezentuje pojedynczy dokument w systemie LogicalDoc. To właśnie klasa `Document` jest fundamentem indeksu czasowoprzestrzennego. Posiada ona następujące właściwości:
  - `documentID` - unikalny numer identyfikujący dokument,
  - `documentDescription` - słowny opis dokumentu,
  - `regardsBuilding` - określa jakiego budynku dotyczy dokument,
  - `regardsCircle` - określa jakiego obszaru ograniczonego okręgiem dotyczy dokument,
  - `regardsContinent` - określa jakiego kontynentu dotyczy dokument,
  - `regardsCountry` - określa jakiego państwa dotyczy dokument,
  - `regardsInstant` - określa jakiego punktu w czasie dotyczy dokument,
  - `regardsInterval` - określa jakiego przedziału czasowego dotyczy dokument,
  - `regardsPointOnEarth` - określa jakiego punktu na Ziemi definiowanego przez parę współrzędnych geograficznych dotyczy dokument,
  - `regardsRectangle` - określa jakiego obszaru ograniczonego prostokątem dotyczy dokument,
  - `regardsTown` - określa jakiego miasta dotyczy dokument.
- `Instant` - jest to klasa reprezentująca pojedynczy punkt w czasie. Jej jedyną właściwością jest:
  - `time` - punkt w czasie
- `Interval` - jest to klasa opisująca przedział czasowy. Posiada dwie właściwości:
  - `hasBeginning` - określa punkt w czasie (`Instant`), będący początkiem przedziału czasowego,
  - `hasEnd` - określa punkt w czasie (`Instant`), będący końcem przedziału czasowego.

  Dopuszczalne jest niezdefiniowanie dla przedziału czasowego jednej lub obu właściwości.
- `Place` - jest to klasa pełniąca nadrzędną funkcję w stosunku do takich klas, jak: `Building`, `Country` itp. Reprezentuje miejsce lub obszar na Ziemi w najogólniejszym sensie.

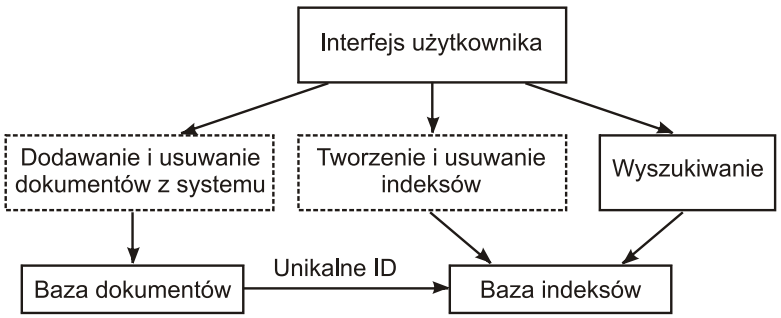
2.4.2. Podklasy `Place`

- `Address` - jest to klasa reprezentująca adres pocztowy obiektu. Ze względu na różne systemy numeracji budynków na świecie, takie właściwości klasy, jak: `flatNumber` czy `houseNumber` są typu `xsd:string`. Klasa `Address` posiada następujące właściwości:
  - `addressTown` - miasto występujące w adresie pocztowym,
  - `flatNumber` - numer mieszkania. Może zawierać litery i inne znaki,
  - `fullAddress` - Pełny adres obiektu w postaci ciągu znaków,
  - `houseNumber` - Numer domu. Może zawierać litery i inne znaki,
  - `postalCode` - kod pocztowy,
  - `street` - nazwa ulicy.
- `Building` - jest to klasa reprezentująca budynek. Budynek może mieć swój adres, tzn. właściwości:
  - `buildingAddress` - adres budynku,
  - `containedInTown` - określa w jakim mieście znajduje się budynek.
- `Circle` - jest to klasa reprezentująca okrąg w terenie, opisany poprzez podanie współrzędnych geograficznych jego środka oraz promienia w metrach. Klasa ta posiada następujące właściwości:
  - `center` - definiuje środek okręgu w postaci pary współrzędnych geograficznych,
  - `radius` - promień okręgu podawany w metrach.
- `Continent` - jest to klasa reprezentująca jeden z siedmiu kontynentów.
- `Country` - jest to klasa reprezentująca państwo. Posiada następującą właściwość:
  - `containedInContinent` - określa na jakim kontynencie leży dane państwo.
- `PointOnEarth` - jest to klasa reprezentująca współrzędne geograficzne w systemie WGS 84. Właściwości tej klasy to:
  - `latitude` - długość geograficzna,
  - `longitude` - szerokość geograficzna.
- `Rectangle` - jest to klasa reprezentująca obszar w terenie ograniczony prostokątem, definiowanym jako dwie pary współrzędnych geograficznych: 'lewy górny' oraz 'prawy dolny' róg.
  - `lowerRightCorner` - dolny prawy róg prostokąta. Jest to para współrzędnych geograficznych w systemie WGS 84,
  - `upperLeftCorner` - lewy górny róg prostokąta. Jest to para współrzędnych geograficznych w systemie WGS 84.
- `Town` - jest to klasa reprezentująca pojedynczy punkt w czasie. Jej jedyną właściwością jest:

– `time` - punkt w czasie.

2.5. Architektura systemu

Architekturę systemu DMS z indeksowaniem opartym na ontologii pokazano na rys. 2.2. Podstawowym elementem jest baza danych o dokumentach. Ma ona zwykle postać klasycznej relacyjnej bazy danych, zawierającej podstawowe informacje o zarządzanych dokumentach typu: ścieżka do pliku, autor, słowa kluczowe itp.



Rys. 2.2: Architektura systemu DMS z indeksowaniem opartym na ontologii.

Każdy dokument musi mieć unikalne ID, które pozwala na połączenie bazy dokumentów z bazą indeksów ontologicznych. Indeksy te mają postać trójek RDF, przypisujących dokumentowi o danym ID określone ustrukturyzowane własności. w tym przypadku są to własności opisujące dokument w kategoriach czasowo-przestrzennych. Ich strukturę determinuje użyta ontologia, którą tutaj jest ontologia *Spime*.

Użycie tego typu metadanych ma oczywiście określony cel. Jest nim umożliwienie czasowo-przestrzennego wyszukiwania dokumentów. Konieczne jest więc stworzenie mechanizmu pozwalającego na zadawanie odpowiednich zapytań do bazy indeksów. Takim mechanizmem jest SPARQL, czyli język zapytań dla RDF. Umożliwia on formułowanie zapytań do baz trójek RDF, podobnie jak język SQL czyni to dla relacyjnych baz danych. Dzięki temu można dokonywać wyszukiwania na podstawie kryteriów czasowo-przestrzennych w bazie indeksów, a następnie, dzięki unikalnemu ID dokumentu, skojarzyć uzyskane wyniki z zawartością bazy dokumentów.

Aby uzyskać w pełni funkcjonalny system należy jeszcze umożliwić dodawanie, indeksowanie i usuwanie dokumentów z systemu. Ważne jest takie zaimplementowanie tych funkcji, aby zachować spójność przechowywanych danych oraz maksymalnie wykorzystać możliwości zastosowanej ontologii. Sporym problemem jest usuwanie indeksów ontologicznych. Przykładowo: kilka dokumentów może być powiązanych z jednym miastem, np. Wrocławiem. Usunięcie z systemu jednego z nich nie może powodować usunięcia trójek opisujących miasto Wrocław. Może tak się stać dopiero po usunięciu ostatniego z tych dokumen-

tów. Konieczne jest więc ciągłe śledzenie powiązań pomiędzy przechowywanymi indeksami, aby nie tracić informacji, ale również nie zaśmiecać bazy niepotrzebnymi elementami.

2.6. Aplikacja do przeszukiwania ontologii

Aplikacja służąca do przeszukiwania plików RDF zgodnie z ontologią *Spime* została stworzona w środowisku Windows. Program został napisany w języku Visual Basic w środowisku Microsoft Visual Studio 2008. Główne okno aplikacji przedstawiono na rys. 2.3. Okno to podzielone jest na trzy główne części odpowiedzialne za: konfigurację programu, parametry wyszukiwania, zapytanie SPARQL (wygenerowane na podstawie zaznaczonych parametró wyszukiwania). Parametry wyszukiwania jakie można ustawić podzielono na trzy grupy: pierwsza - do wyszukiwania po danych adresowych; druga - do wyszukiwania na podstawie współrzędnych geograficznych; trzecia - do wyszukiwania na podstawie danych czasowych. Ostatnia część aplikacji służy do prezentowania zapytania w języku SPARQL które zostanie wygenerowane na podstawie wybranych parametrów.

Program do prawidłowego działania wykorzystuje silnik wyszukiwania JENA zawarty w pakiecie ARQ. w jego konfiguracji należy odpowiednio ustawić ścieżki do: zewnętrznej aplikacji `sparql.bat` (o względnej ścieżce dostępu `/bat/sparql.bat`); pliku RDF zapisanego zgodnie z ontologią *Spime*; pliku do przechowywania zapytań SPARQL; pliku w którym będą przechowywane wyniki działania programu.

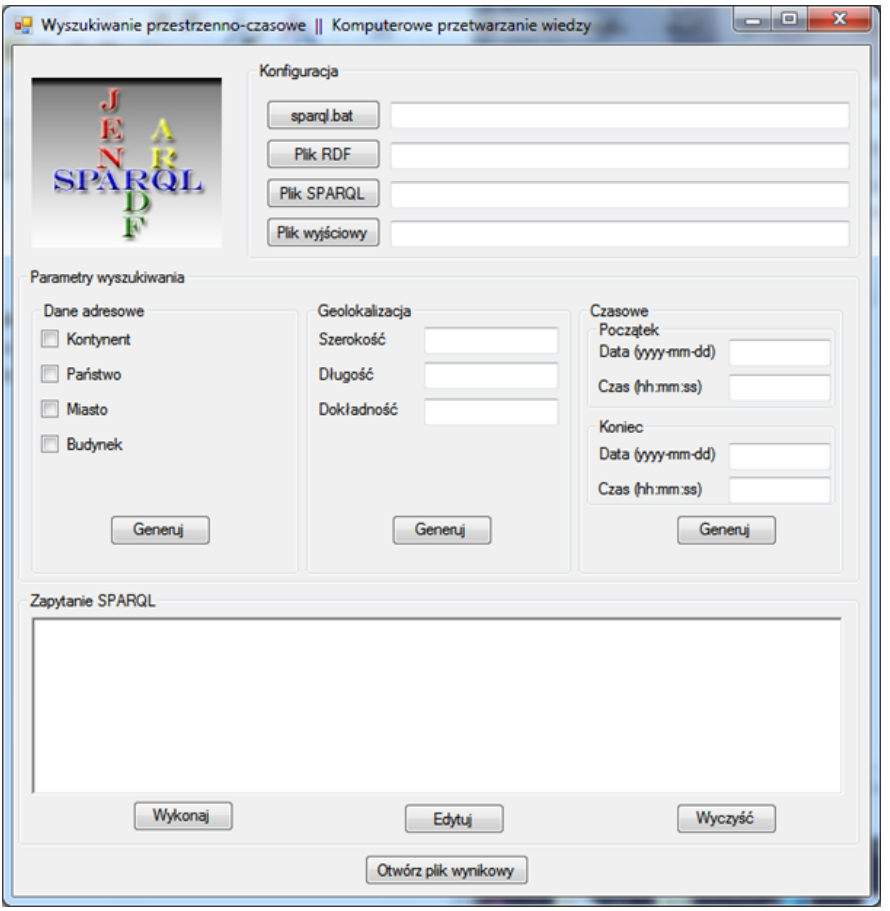
Zapytania SPARQL budowane są fragmentami z wcześniej przygotowanych szablonów dla poszczególnych parametrów. Po kliknięciu jednego z przycisków *generuj* sprawdzane są informacje na temat wybranych przez użytkownika parametrów i ich ustawień, a następnie tworzone jest kawałkami zapytanie SPARQL, z podstawieniem wartości parametrów. Przykładowe zapytanie SPARQL pokazano na rys. 2.4.

Przykładową odpowiedź na zapytanie SPARQL do bazy indeksów ontologicznych pokazano na rys. 2.5.

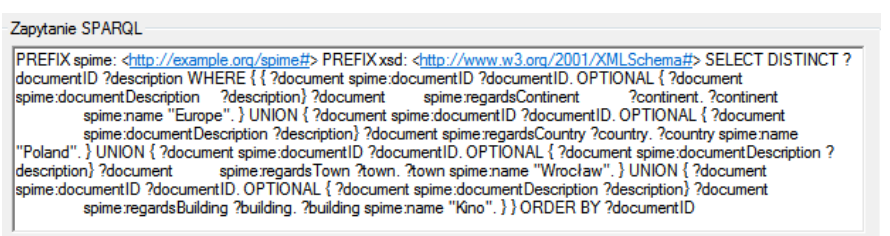
2.7. Podsumowanie

Realizując projekt związany z rozbudową systemu DMS o możliwość wyszukiwania czasowo-przestrzennego na bazie indeksów ontologicznych napotkano na pewne trudności. Podstawowym problemem był brak dostępności do dokumentacji systemu DMS LogicalDoc. w wyniku prac zauważono, że aplikacja, która pierwotnie była Open Source, obecnie zaczyna przybierać charakter komercyjny. Istnieje kilka różnych wersji systemu, co jednoznacznie wskazuje na chęć stworzenia płatnego systemu DMS. Z tego też powodu ostatnim dobrze udokumentowane wydaniem jest wersja 5.1 tego systemu. Dlatego jako cel projektu postawiono zbudowanie aplikacji służącej do intuicyjnego przeszukiwania plików RDF stworzonych zgodnie z ontologią *Spime*. Program spełnił pokładane w nim oczekiwania i pozwolił na utworzenie zalążka być może nowego systemu DMS,

który zajmie miejsce LogicaDoc. Ze względu na początkowe trudności i ograniczony czas realizacji projektu, stworzona aplikacja ma dość ograniczoną funkcjonalność. Brakuje w niej możliwości dodawania nowych trójek i ich zapisanie w plikach RDF. Jest to zadanie, które wymaga opracowania własnych rozwiązań.



Rys. 2.3: Główne okno aplikacji.



Rys. 2.4: Przykładowe zapytanie SPARQL.

documentID	description
3	"Wystapienie Polski z UE."^^xsd:string
4	"Europa zostanie zalana przez Ocean Indyjski."^^xsd:string
6	"I znowu meteoryt. Ale urwal."^^xsd:string

Rys. 2.5: Odpowiedź na zapytanie SPARQL.

Literatura

[1] Resource Description Framework (RDF). http://www.w3.org/RDF/, (2004).
[2] OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/, (2004).

ROZDZIAŁ 3

PROGRAMOWE WSPARCIE PROCESU BUDOWY TEZAURUSA

P. Batog, M. Pichliński

3.1. Wstęp

Rozwój komputerów, sprzętu elektronicznego oraz Internetu sprzyja zwiększaniu zasobów informacyjnych gromadzonych w sieci oraz rozwojowi metod ich udostępniania. W roku 2003 za pomocą wyszukiwarek internetowych można było sięgnąć do 167 terabajtów danych tzw. powierzchniowego Internetu, czyli trzykrotnie więcej niż w 2000 roku [1]. Przedstawiciele IBM w 2006 roku przewidywali, że w 2010 ilość przechowywanej elektronicznej informacji będzie zwiększała się dwukrotnie w przeciągu 11 godzin (niekoniecznie w samej sieci) [2]. Biorąc pod uwagę fakt, że Internet powierzchniowy to dopiero wierzchołek góry lodowej (szacuje się, że tzw. głęboki Internet, czyli taki, którego dane ukryte za strefami zabezpieczeń, jest 500 krotnie większy od Internetu dostępnego z poziomu wyszukiwarek [3]) widać, jak wielki zasób informacji został zgromadzony już w sieciach komputerowych. Z tego też powodu coraz trudniej znaleźć jest potrzebne dane, coraz więcej problemów sprawia samo wyszukiwanie, coraz ciężej jest opanować chaos informacyjny [2]. Te i podobne problemy wynikające z konieczności przetwarzania zasobów wiedzy próbuje się rozwiązać za pomocą modeli danych strukturalnych.

W niniejszej rozdziale zostanie przedstawione ontologiczne podejście do budowy modeli informacyjnych zilustrowane przykłem tworzenia tezaurusa. Omówione zostaną pojęcia modelu informacyjnego oraz ontologii. Następnie przedstawiona zostanie odpowiedź na pytania: Czym jest tezaurus? Jak wygląda proces jego tworzenia oraz jakie ma on zastosowania? Pod koniec rozdziału opisana zostanie specyfikacja SKOS będąca podstawą w implementacji omawianego tezaurusa.

3.1.1. Czym są modele informacyjne?

Modele informacyjne służą do reprezentacji zasobów wiedzy, takich jak: koncepty, relacje, ograniczenia, zasady, operacje. Nadają one znaczenia semantycznego zgromadzonym danym [4]. Dzięki modelom informacyjnym wiedza, która jest w nich przechowywana, nabiera elastyczności pozwalającej na: dzielenie jej pomiędzy różnymi aplikacjami, łatwiejsze wyszukiwanie pożądanych informacji oraz dostosowanie do potrzeb użytkownika [5].

Warto wspomnieć o różnicy pomiędzy modelami informacyjnymi a modelami danych, które mogą być mylnie utożsamiane ze sobą. Celem tych pierwszych jest zarządzanie modelami na poziomie ideowym oraz przedstawienie relacji pomiędzy nimi, niezależnie od konkretnych implementacji bądź protokołów transmisji danych. Model informacyjny jest narzędziem przeznaczonym dla projektantów. Natomiast modele danych odnoszą się niższemu poziomowi abstrakcji, zawierają więcej szczegółów, są one już konkretną implementacją modeli informacyjnych [6].

3.1.2. Ontologia jako model informacji

Celem stworzenia modelu informacji jest nadanie jej wyższego stopienia organizacji. Uporządkowanie ma zaś umożliwić łatwiejsze przetwarzanie informacji i wyciąganie na ich podstawie określonych wniosków. Innymi słowy jest to dodanie abstrakcji umożliwiającej przechodzenie od informacji w kierunku do wiedzy. Budowa modelu informacji jest zagadnieniem z natury heurystycznym. Nie istnieją bowiem ogólne prawa, według których proces budowy takiego modelu miałby zostać przeprowadzony.

Budowa modelu ściśle wiąże się z dziedziną i problemem, którego ma być rozwiązaniem. W dzisiejszych czasach dostępnych jest mnóstwo odpowiednich metod i technik. Z uwagi na różnorodność zastosowań modyfikuje się ogólnie znane metody, bazując na doświadczeniach w konkretnych zagadnieniach i technikach. Taka sytuacje bywa nazywana w literaturze „dżunglą metodologiczną" [7]. W niniejszej pracy skupiono się na podejściu ontologicznym.

Ontologia jest reprezentacją dystrybuowanej konceptualizacji określonej dziedziny. Mówiąc prościej, ontologia definiuje wspólną warstwę pojęciową w danej dziedzinie oraz rozwija konceptualizację poprzez dodanie relacji pomiędzy poszczególnymi podmiotami (obydwa pojęcia mają swoje korzenie w naukach filozoficznych).

- Konceptualizacja - próba określenia ścisłych pojęć, (usunięcie wieloznaczności) niezbędnych do opisu jednoznacznie rozumianego procesu z określonej dziedziny.
- Ontologia - (od greckich ovto - byt i iogos - nauka) najogólniejsza nauka rozważająca pojęcie bytu, do XVII w utożsamiana z metafizyką. Jej początków można szukać u Arystotelesa. Dziedzina metafizyki, związana z badaniem, wyjaśnianiem natury, kluczowych właściwości oraz relacji rządzących wszelakimi bytami oraz głównych zasad i przyczyn bytu.

W praktyce technologii informacyjnych poprzez ontologię rozumie się świadomą, formalną specyfikację konceptów (pojęć) w danej dziedzinie i relacji pomiędzy nimi. Ontologia łączy ze sobą nie tylko wyrażone w niej pojęcia, ale również wiedzę, która może być na jej podstawie wywnioskowana. Jako sposób reprezentacji zintegrowanego modelu wiedzy z danej dziedziny oraz jako instrument badawczy treści informacji tekstowej jest obecnie, razem z innymi technologiami informatycznymi, jednym z kluczowych elementów rozwijającej się idei społeczeństwa informacyjnego. Dowodem na to jest dynamiczny rozwój internetu trzeciej generacji.

3.2. Tezaurus

Tezaurus w ogólnym rozumieniu stanowi słownik terminów bliskoznacznych. Słowo tezaurus pochodzi z greckiego $\Theta\eta\sigma\alpha\upsilon\rho\omega\varsigma$, co oznaczało zbiór rzeczy o wielkiej wartości (magazyn, skarbiec). W technologii informacyjnej i bibliotekoznawstwie tezaurus jest ściśle określonym rozwinięciem słownika. Jest to zbiór semantycznie i hierarchicznie powiązanych terminów, ułatwiający wyszukiwanie informacji.

Tezaurus jest pewnego rodzaju zastosowaniem podejścia ontologicznego w celu sklasyfikowania słów języka naturalnego. Jak wiadomo, język naturalny zawiera wiele słów o wielu znaczeniach, ale relacje pomiędzy słowem a jego znaczeniem nie są jednoznaczne. Jedno słowo bowiem może mieć wiele znaczeń, a jedno pojęcie może być wyrażane przez wiele słów. Co więcej, często relacje te zmieniają się w zależności od kontekstu użycia jak i interpretacji samych osób je wypowiadających i czytających. Człowieka interesuje przeważnie znaczenie, jednak musi się on posługiwać słowami.

Idealnym tezaurusem byłby system komputerowy, który pobierając pewne słowa zdołałby odpowiednio je zinterpretować i zmapować na koncepty odzwierciedlające sedno poszukiwań, dzięki czemu możliwe byłoby zwrócenie dokładnie wszystkich wyników interesujących użytkownika, ale bez żadnego nadmiaru. Przykładowo, na zapytanie „lodówka" system wyświetli także to, co było opisane jako „chłodziarko-zamrażarka" ale już nie jako „chłodnie".

Rozumienie relacji zachodzących pomiędzy słowami i znaczeniami jest częścią ludzkiej inteligencji. Nauka tych relacji to proces, który trwa przez całe życie. Jest to nietrywialne zagadnienie, o czym świadczą tak zwane nieporozumienia występujące podczas wymiany informacji, gdy odbiorcy i nadawca inaczej rozumieją znaczenie przekazywanych słów. Świadczy o tym także szacunek dla ludzi, którzy opanowali umiejętność jasnego i zwięzłego wypowiadania się. Jak jednak nauczyć tego maszynę, jak zaimplementować komputerowe przetwarzanie wiedzy i wyszukiwanie informacji w ogromnych zbiorach danych?

3.2.1. Katalog przedmiotowy

Pierwszą próbę semantycznego usystematyzowania terminów podjęli już dawno bibliotekarze. Było nią utworzenie katalogu przedmiotowego. W takim systemie każda z książek opisana jest jednym lub wieloma hasłami z katalogu.

Zakłada się, że opisywanie zawartości tematycznej książek, jak i wyszukiwanie, odbywa się przy użyciu tego samego zbioru słów. Niestety katalog taki jest mało elastyczny i odporny, a efektywne korzystanie z niego jest rzeczą wymagającą odpowiedniego przygotowania. O ile doświadczony bibliotekarz na co dzień obcujący z katalogiem nie ma problemów z określeniem odpowiednich kategorii, to przypadku przeciętnego użytkownika część zasobów pozostanie ukryta, bowiem jego intuicje co do tego do jakiej kategorii został zaklasyfikowany poszukiwany przez niego temat mogą być mylne. Co więcej, odwołując się do hasła np. *Rzeki w Polsce* można znaleźć jedynie pozycje omawiające ogólnie *Rzeki*, natomiast już nie odnajdzie się monografii o Odrze, Wiśle itd. ani książek opisujących ogólnie rzeki w Europie. Użytkownik musiałby odnaleźć w katalogu pojęcia węższe i szersze a następnie przeszukać listy książek im odpowiadające, co z pewnością byłoby czasochłonne.

Komputeryzacja katalogu zwiększa wygodę użytkowania i efektywność systemu, m.in. poprzez znaczne zwiększenie ilości kategorii możliwych do przeszukania w tym samym czasie, jednak nie zmienia zasady działania, i nie usuwa jego zasadniczych wad. Co więcej, z komputerowych katalogów korzystają z reguły bez pomocy bibliotekarzy sami czytelnicy, przeważnie nie uświadamiający sobie opisanych wad katalogów, co może zniechęcać do odwiedzania bibliotek i prowadzić do frustracji.

3.2.2. Wyszukiwanie w katalogach

W ostatnich latach do opisywania danych coraz częściej wykorzystuje się technologię słów kluczowych. Dzięki temu możliwe jest „wyszukiwanie po słowach kluczowych". Z uwagi na ich „luźny" charakter należy stwierdzić, że jest to technologia komplementarna w stosunku do słownictwa kontrolowanego [8]. Choć oczywiście jest ona bardzo użyteczna, posiada jednak szereg wad takich jak:

- konieczność podawania synonimów,
- niepełny opis,
- brak usystematyzowania,
- zwracanie setek bezużytecznych trafień (wyszukiwarki internetowe),
- brak relacji,
- trudności ze zbudowaniem systemu inteligentnego.

Zamiast ograniczać się do samego opisu danego podmiotu, można rozważyć jako alternatywę odwołanie się do pełnego tekstu - książki, artykułu itp. Jest to „wyszukiwanie pełnotekstowe". Niestety, nie wszystko posiada swój „pełny tekst". Obiektem kategoryzacji i wyszukiwania są nie tylko książki i artykuły, ale także dzieła sztuki, utwory muzyczne, artykuły sklepowe i wiele innych. Warto zauważyć, że sama zawartość bibliotek jest rzadko w pełni zdigitalizowana a pełny tekst dostępny w formie elektronicznej. Ponadto wyszukiwanie pełnotekstowe ma swoje ograniczenia, m.in.:

- znacznie dłuższy czas trwania,

- zwracanie znacznie większej ilości bezużytecznych trafień,
- ograniczenie do jednego języka naturalnego.

3.2.3. Proces tworzenia tezaurusa

Tezaurus w uproszczeniu można rozumieć jako zbiór słownictwa kontrolowanego i zbiór relacji zachodzących pomiędzy słowami. Słownictwo kontrolowane jest ujednoznacznione (poprzez jawne zdefiniowanie pojęć lub rozróżnienie pomiędzy różnymi znaczeniami tego samego słowa). Przeważnie zbiór słów jest ograniczony do jakiejś konkretnej dziedziny, w której będzie wykorzystywany. Do opisu zasobów wolno posługiwać się tylko słowami które są w systemie.

Następnym krokiem w organizacji wiedzy jest określenie reguł relacyjnych zachodzących pomiędzy danymi konceptami. Koncept jest pojęciem abstrakcyjnym, może odpowiadać mu wiele słów, ale może też nie istnieć właściwe słowo. Najczęściej spotyka się implementację następujących reguł:

- *Hiperonimia* - terminy nadrzędne (koncepty ogólniejsze),
- *Hiponimia* - terminy podrzędne (koncepty bardziej szczegółowe),
- *Synonimia* - termin bliskoznaczny (wymienienie *Deskryptorów* - terminów preferowanych i *Askryptorów*),
- *Homonimia* - różne znaczenie tego samego słowa (stworzenie odrębnych konceptów),
- *Asocjacja* - skojarzenie ze sobą pokrewnych konceptów.

Poprzez dodanie relacji hierarchicznych i skojarzeniowych, otrzymuje się narzędzie, które umożliwia stworzenie inteligentnego systemu wyszukiwawczego oferującego znacznie większe możliwości niż zwykły katalog. Jadwiga Woźniak-Kasperek [9] wymienia następujące etapy budowy tezaurusa:

1. Określenie zakresu języka.
2. Wskazanie źródeł, z których (lub na podstawie których) będzie pobierane słownictwo.
3. Robocze określenie struktury części rzeczowej (systematycznej) tezaurusa.
4. Wybór metody gromadzenia słownictwa.
5. Zgromadzenie słownictwa.
6. Ustalenie zasad tworzenia ciągów synonimicznych.
7. Ustalenie, jak będą „oddzielane" różne znaczenia wyrażeń wieloznacznych.
8. Wybranie sposobu zapisu deskryptorów, askryptorów, ew. modyfikatorów, przyjęcie notacji, za pomocą której będą oznaczane typy relacji łączących hasła w tezaurusie.
9. Opracowanie kilku wzorcowych przykładów artykułów deskryptorowych i askryptorowych (z komentarzem).
10. Opracowanie tezaurusa, w tym części systematycznej.

3.2.4. Przykłady tezaurusów

Od XVI w. nazwa tezaurus występuje jako oznaczenie słownika lub leksykonu. W Polsce, pierwszym dziełem tego typu był *Thesaurus Polono-Latino-Graecus*

Grzegorza Knapiusza (Kraków 1621-32). Twórcą pierwszego nowożytnego, usystematyzowanego semantycznie tezaurusa był Peter Mark Roget. *Thesaurus of English Words and Phrases*, dzieło jego życia, ukazało się w 1852 r. w Londynie i jest używane także dziś, choć oczywiście nieustannie aktualizowane. Tezaurus Rogeta dotyczył języka angielskiego i zawierał 6 poziomów hierarchicznych. Tezaurusy jako język wyszukiwawczo-informacyjny zaczęły rozwijać się intensywnie w drugiej połowie XX w. i trend ten trwa właściwie do dzisiaj. Pierwszy polski tezaurus ukazał się w 1969 r. i był wykazem terminów z zakresu urządzeń budowlanych i transportu bliskiego [10]. Budową (także wielojęzycznych) tezaurusów, m.in. w zakresie dziedzictwa kulturowego, zajmuje się wiele ośrodków naukowych w Europie (korzystając także ze wsparcia Unii Europejskiej). Przykładowe tezaurusy dostępne *on-line*:

- Roget Thesaurus `http://www.roget.org/`
- WordNet `http://wordnetweb.princeton.edu/perl/webwn`
- Visual Thesaurus `http://www.visualthesaurus.com/`
- Tezaurus Dziedzictwa Kulturowego `http://historiasztuki.uni.wroc.pl/tezaurus.html`

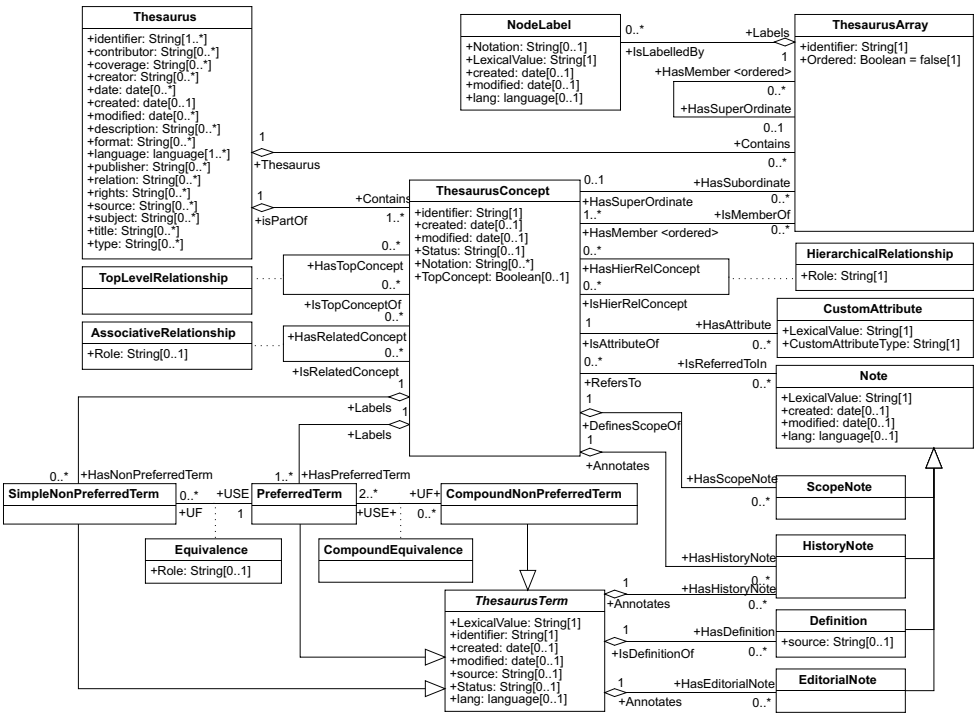
Tezaurusy mogą być stosowane wszędzie tam, gdzie zachodzi potrzeba wyszukiwania lub „zrozumienia" tekstu przez komputer. Takimi miejscami mogą być: biblioteki, archiwa, muzea, bazy danych, rozbudowane sklepy internetowe i serwisy aukcyjne, systemy przetwarzania i rozumienia mowy. Tezaurusy mogą rozszerzać opcje wyszukiwania wyszukiwarek.

3.3. Dokumenty normatywne

3.3.1. Wzorce i modele tezaurusów

Za wzorzec tezaurusa uważa się opublikowany w 1967 r. *Thesaurus of Engineering and Scientific Terms* (TEST), który powstał w ramach tzw. projektu LEX, realizowanego w latach sześćdziesiątych wspólnie przez *Office of Naval Research amerykańskiego Department of Defence* (DOD) oraz *Engineering Joint Council* (EJC). Miał on formę tzw. alfabetyczno-hierarchicznego wykazu deskryptorów i askryptorów, w którym zastosowano symetryczne oznaczenia relacji ekwiwalencji (*use – use for*), hierarchicznych (*broader term* oraz *narrower term*) oraz asocjacyjnych (*related term*) używanych do dzisiaj [10]. Na podstawie właśnie tego systemu powstała w USA pierwsza na świecie norma dotycząca budowy tezaurusów - ANSI Z39.19-1974. Pierwszym międzynarodowym standardem była norma ISO 2788 *Documentation – Guidelines for the Establishment and Development of Monolingual Thesauri* [11], przyjęta w Polsce jako PN/N-09008. Z początkiem XXI w., w związku z koniecznością zapewnienia odpowiedniej elastyczności (interoperacyjności pomiędzy tezaurusami i innymi systemami organizacji wiedzy), wydane zostały nowe normy: amerykańska Z39.19-2005 *Guidelines for Construction, Format, and Management of Controlled Vocabularies* [12], brytyjska BS 8723 *Structured Vocabularies for Information Retrieval* [13, 14]. W trakcie tworzenia jest międzynarodowa norma ISO 25964 (`http://www.niso.org/`

`workrooms/iso25964/`). Diagram klas, wchodzących w skład tezaurusa według normy BS 8723, został umieszczony na rys. 3.1.



Rys. 3.1: Klasy wchodzące w skład tezaurusa, według normy BS 8723.

3.3.2. Język SKOS

Simple Knowledge Organization System jest językiem, który pozwala na tworzenie systemów organizacji wiedzy jak tezaurusy, schematy klasyfikacji, systemy nagłówkowe oraz taksonomie. Modele, które są zapisane za pomocą SKOS, mogą być łatwo przetwarzane przez maszyny, wymieniane pomiędzy aplikacjami komputerowymi oraz publikowane w internecie.

Modele informacyjne utworzone w języku SKOS spełniają warunki *OWL Full ontology* i mogą one zawierać również informacje zapisane za pomocą OWL (*Web Ontology Language*). SKOS, podobnie jak OWL, powstał w oparciu o definicje RDF (*Resource Description Framework*) i formalna składnia jaką się posługuje to RDF/XML oraz Turtle [15].

Słowa wykorzystywane w języku SKOS zostały zdefiniowane w przestrzeni nazw: `http://www.w3.org/2004/02/skos/core#`, reprezentowanej w niniejszym dokumencie przedrostkiem `skos:`. Na podstawie [16] dostępne są następujące słowa kluczowe, które definiują klasy (słowa zaczynające się z dużej litery) oraz właściwości (są pisane z małej litery):

- `skos:Concept` - jednostka myśli, będąca abstrakcyjną encją, niezależną od pojęć użytych do jej etykietowania; semantyczna zawartość danego konceptu, może zostać wyrażona za pomocą kombinacji innych konceptów [17].
- Etykiety - są cechami konceptów, które służą jako odnośniki do nich wyrażone w języku naturalnym. SKOS pozwala na tworzynie etykiet w różnych językach w postaci: `"treść etykiety"@skrót_języka`. Istnieją trzy rodzaje etykiet, które wzajemnie się wykluczają (niedozwolone jest istnienie etykiet innego typu o tej samej treści):
  - `skos:prefLabel` - preferowana etykieta, może być tylko jedna dla danego konceptu; nie jest zabronione, by różne koncepty zawierały tę samą etykietę, jednak zaleca się, by takich przypadków nie było
  - `skos:altLabel` - alternatywna etykieta, przydatna do określenia synonimów, jak również wyrazów bliskoznacznych oraz akronimów
  - `skos:hiddenLabel` - ukryta etykieta, dostępna jedynie wewnętrznie dla aplikacji w celu indeksowania i wyszukiwania, nie jest ona widoczna dla użytkownika; może służyć do uwzględniania literówek podczas poszukiwania danego konceptu
- Relacje - służą do wiązania konceptów pomiędzy sobą. SKOS dostarcza następujące typy relacji:
  - `skos:narrower` - hierarchiczne połączenie z konceptem bardziej szczegółowym na przykład:

    ```
    ex:figuraGeometryczna rdf:type skos:Concept;
      skos:narrower ex:kwadrat.
    ex:kwadrat rdf:type skos:Concept.
    ```

  - `skos:broader` - łączy dany koncept z konceptem bardziej ogólnym:

    ```
    ex:kwadrat rdf:type skos:Concept;
      skos:broader ex:figuraGeometryczna.
    ex:figuraGeometryczna rdf:type skos:Concept.
    ```

  - `skos:related` - łączy z innym konceptem bez ustalania hierarchii, nie jest relacją tranzytywną.

  Pojęcia `skos:broader` i `skos:narrower` są przeciwieństwami, więc wystarczy określić tylko jedną z tych relacji, druga natomiast zostanie automatycznie przypisania w procesie wnioskowania OWL. Dodatkową cechą wyżej wymienionych relacji jest brak ich tranzytywności, czyli:

  ```
  ex:figuraGeometryczna rdf:type skos:Concept;
    skos:narrower ex:prostokat.
  ex:prostokat rdf:type skos:Concept;
    skos:narrower ex:kwadrat.
  ex:kwadrat rdf:type skos:Concept.
  ```

  nie implikuje relacji:

```
ex:figuraGeometryczna skos:narrower ex:kwadrat.
```

Jednak w [15] opisane są tranzytywne wersje relacji *broader* i *narrower*: `skos:broaderTransitive`, `skos:narrowerTransitive`, które umożliwiają z powyższej deklaracji wnioskować:

```
ex:figuraGeometryczna skos:narrowerTransitive ex:kwadrat.
```

Należy pamiętać również o tym, że poszczególne typy relacji wzajemnie się wykluczają.

- Przypisy dokumentacyjne - informacje w postaci zrozumiałej dla człowieka, przypisane do konceptu, w celu jego opisania:
  - `skos:definition` - zawiera pełne wyjaśnienie znaczenia danego konceptu,
  - `skos:example` - zawiera przykłady użycia konceptu,
  - `skos:scopeNote` - dostarcza częściowego znaczenia konceptu w szczególności określa zakres jego użycia,
  - `skos:historyNote` - opisuje znaczące zmiany jakie zaszły w znaczeniu, bądź formie konceptu,
  - `skos:editorialNote` - informacja od autora, umieszczana w celu ułatwienia utrzymania konceptu,
  - `skos:changeNote` - informacja o zmianach w koncepcie, umieszczona w celach administracyjnych.
- Schematy konceptów - służą do grupowania, pojedynczych konceptów, w celu ułatwienia indeksowania i tworzenia np. tezaurusów:
  - `skos:ConceptScheme` - kontener będący schematem konceptów, zawierającym inne koncepty,
  - `skos:hasTopConcept` - przypisuje danemu schematowi, najwyższy w hierarchii koncept; może być kilka najwyższych konceptów,
  - `skos:topConceptOf` - przypisuje danemu konceptowi schemat konceptów, w którym będzie on najwyższy w hierarchii,
  - `skos:inScheme` - przypisuje danemu konceptowi schemat konceptów, w którym się znajduje, bez określenia hierarchii.

  Niestety w [15] określone jest, że relacje pomiędzy konceptami (tzn. `skos:broader`, `skos:narrower` itp.) nie są przenoszone na schemat konceptów, czyli:

```
ex:tezaurusAstronomiczny rdf:type skos:ConceptScheme;
  skos:hasTopConcept ex:cialoNiebieskie.
ex:cialoNiebieskie rdf:type skos:Concept;
  skos:narrower ex:gwiazda.
ex:gwiazda rdf:type skos:Concept.
```

  nie implikuje:

```
ex:gwiazda skos:inScheme ex:tezaurusAstronomiczny
```

- Mapowanie konceptów pomiędzy schematami - SKOS dostarcza odpowiednich narzędzi, pozwalających mapować między sobą poszczególne koncepty w różnych schematach:
  - `skos:exactMatch` - dokładne dopasowanie, koncepty są identyczne,
  - `skos:closeMatch` - dopasowanie częściowe, które oznacza, że koncepty mogą być używane zamiennie, jednak w ograniczonym zakresie,
  - `skos:broaderMatch` - dopasowanie konceptu o bardziej ogólnym znaczeniu,
  - `skos:narrowerMatch` - dopasowanie konceptu o bardziej szczegółowym znaczeniu,
  - `skos:relatedMatch` - dopasowanie konceptu o powiązanym znaczeniu z danym konceptem bez wyróżnienia hierarchii.
- Kolekcje - służą do zgrupowania konceptów, które mogą zostać wspólnie dzielić tą samą etykietę:
  - `skos:Collection` - kolekcja konceptów,
  - `skos:OrderedCollection` - kolekcja konceptów, których kolejność jest znacząca, jak np. kolekcja „wykształcenie", która może zawierać: podstawowe, średnie, wyższe,
  - `skos:member` - przypisuje do danej kolekcji, wybrany koncept, użycie:

    ```
    <kolekcja> skos:member <koncept>
    ```

  - `skos:memberList` - przypisuje do danej kolekcji, listę konceptów, użycie:

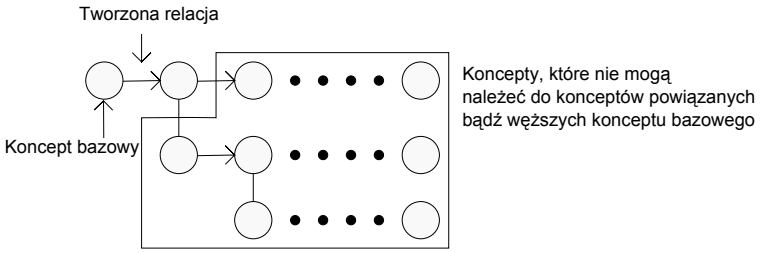
    ```
    <kolekcja> skos:memberList (<koncept1> <koncept2>)
    ```

3.4. Przykładowa implementacja tezaurusa

Tworzenie aplikacji operujących na tezaurusach nie jest łatwym zadaniem. Jednym z ważniejszych problemów jest obsługa bardzo dużych modeli. Próba wczytania pokaźnych rozmiarów tezaurusa w całości może zakończyć się zużyciem całej dostępnej pamięci. W celu uniknięcia tego problemu należy ładować do programu tylko tą część pliku, która jest w rzeczywistości potrzebna.

Kolejnym poważnym problemem jest utrzymanie spójności modelu. Przykładowo dodanie szerszej relacji wiąże się ze sprawdzeniem, czy koncept, który ma być wyżej, nie zawiera żadnego powiązanego, bądź bardziej ogólnego konceptu takiego samego jak jakikolwiek koncept węższy lub powiązany w stosunku do słowa bazowego. Co więcej należy to rekurencyjnie powtarzać na tych gałęziach, aż do momentu, gdy zostaną osiągnięte liście drzewa. Całość została przedstawiona na rys. 3.2. Takie sprawdzanie wiąże się z tym, że przy dużym tezaurusie stworzenie kolejnych relacji będzie uniemożliwione poprzez długi czas weryfikacji poprawności modelu.

Przeszukiwanie również należy do zadań, które mogą sprawić problemy. Samo znalezienie danego słowa w danym modelu nie jest trudne. Jednak jeśli celem jest odnalezienie wszystkich konceptów powiązanych z danym elementem,



Rys. 3.2: Przykład sprawdzenia dodawanej relacji.

możliwe jest powstanie cykli i pojawienie się nieskończonych pętli. Wymusza to śledzenie odwiedzanych słów, co zwiększa narzut pamięciowy i czasowy. W kolejnych podrozdziałach opisano założenia oraz sposób implementacji własnego tezaurusa.

3.4.1. SKOS API

Do implementacji własnego tezaurusa można wykorzystać bibliotekę „SKOS API" napisaną w języku JAVA (więcej informacji o tej bibliotece można znaleźć pod adresem `http://skosapi.sourceforge.net/`) lub inne biblioteki dostarczające metod do przetwarzania dokumentów RDF, repozytoriów trójek itp.

W rozwiązaniu, które opisano w dalszych częściach niniejszego rozdziału, wykorzystano autorską bibliotekę napisaną w języku C++. Stworzony interfejs do języka SKOS bazuje na bibliotece *Soprano - the Qt/C++ RDF framework* oraz *Qt*. Pierwsza z nich zapewnia parsowanie i serializację plików. Dzięki parserowi, dokument zawierający definicję tezaurusa przetwarzany jest na listę trójek RDF. Na podstawie tych stwierdzeń tworzone są obiekty klas z języka SKOS oraz zostają im przypisane odpowiednie właściwości określone przez specyfikację na stronie W3C. W trakcie wczytywania pliku dodawane są kolejne elementy tezaurusa, do modelu, który ma zawierać wszystkie definicje. Każdy koncept, bądź relacja, przed umieszczeniem w tworzonej bazie jest sprawdzana pod względem spójności. W przypadku naruszenia poprawności modelu dodawany element zostaje zignorowany.

Biblioteka *Qt* została wykorzystana do stworzenia odpowiednich kontenerów na wszystkie koncepty, relacje, etykiety itd. znajdujące się w modelu. Nie użyto pojemników dostarczanych przez *STL*, by zachować spójność z *Soprano* oraz interfejsem użytkownika (stworzonym również w *Qt*). Użycie `QList` umożliwiło łatwe rozszerzanie modelu np. o nowe koncepty, dzięki czemu został zredukowany problem zarządzania pamięcią.

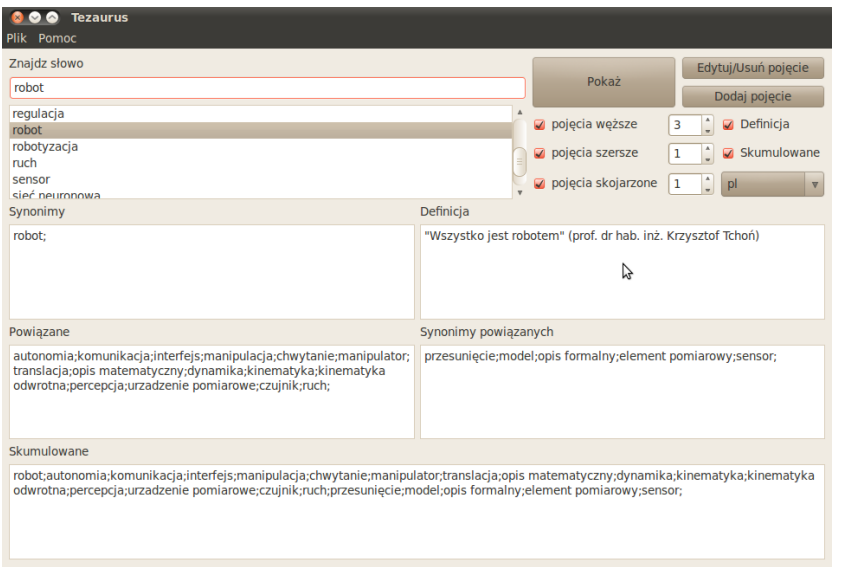
Ze względu na ograniczony czas na realizację projektu autorska biblioteka SKOS API nie implementuje całego zasobu słownictwa SKOS. W aplikacji obsługiwane są: `skos:Concept`, `skos:ConceptScheme`, `skos:prefLabel`, `skos:altLabel`, `skos:hiddenLabel`, `skos:definition`, `skos:broader`, `skos:narrower`, `skos:related`.

3.4.2. Interfejs użytkownika

Interfejs użytkownika składa się zasadniczo z dwóch okien: przeglądarki oraz edytora. W pierwszym znajduje się podgląd zasobu słownictwa w tezaurusie (lista etykiet wszystkich pojęć) oraz pola umożliwiające przedstawienie wybranego pojęcia, tzn. jego etykiet-synonimów, definicji oraz konceptów z nim powiązanych. Użytkownik może wybrać język wyświetlanego słownictwa oraz odpowiednio skonfigurować wyświetlane pola. W szczególności może wybrać rodzaj uwzględnianych relacji oraz ustawić głębokość przeszukiwania grafu powiązań. Okno graficzne przeglądarki przedstawiono na rys. 3.3.

W oknie edytora możliwa jest edycja etykiet oraz definicji w wybranym języku (domyślnie polskim). Można także usunąć relację spośród już istniejących lub dodać relację określonego rodzaju z konceptem będącym już w bazie. Podczas dodawania relacji sprawdzona będzie spójność modelu. Przy usuwaniu pojęcia z bazy usunięte zostaną również odpowiednie relacje.

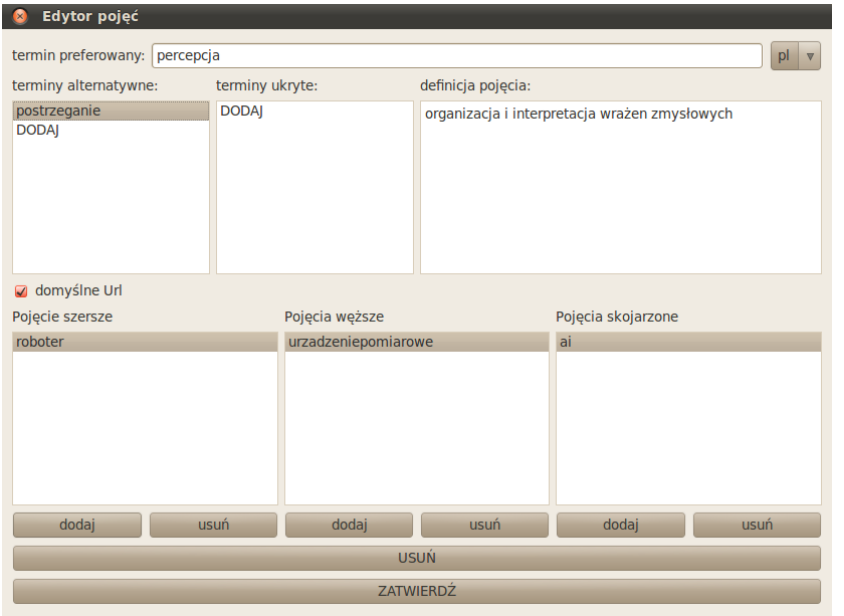
Do tworzenia nowych konceptów służy również okno edytora, uruchamiane w odpowiedni sposób z poziomu przeglądarki. Domyślnie nazwą (unikalnym identyfikatorem w bazie) nowotworzonego konceptu jest etykieta preferowana konceptu. Użytkownik może jednak zmienić ID według swego uznania. Okno graficzne edytora przedstawiono na rys. 3.4.



Rys. 3.3: Okno przeglądarki.

3.4.3. Podsumowanie

Zadaniem opisywanej aplikacji jest ułatwienie budowania tezaurusa zgodnie ze standardem SKOS. Aplikacja pozwala więc użytkownikowi na: odczytanie istniejącego tezaurusa (czyli bazy powiązanych słów z pliku), jego przeglądanie,



Rys. 3.4: Okno edytora.

modyfikację oraz zapisywanie zmian. Podczas implementacji aplikacji główny nacisk położono na dbałość o spójność tworzonego modelu. Dzięki bieżącemu sprawdzaniu poprawności wprowadzanych danych niwelowane są błędy, które może popełnić osoba budująca tezaurus.

Problemy z przeglądaniem i wyszukiwaniem konceptów powiązanych zostały rozwiązane dzięki ograniczeniu głębokości przeszukiwania, zapisanym przez użytkownikaw odpowiednim parametrze (jego wartość to maksymalna liczba przeglądanych węzłów w głąb). Ten prosty zabieg zapobiega problemom z cyklami w grafie, jednocześnie nie ograniczając funkcjonalności aplikacji.

Funkcjonalnym ograniczeniem stworzonej aplikacji jest brak obsługi modeli tezaurusów zapisanych w formacie *Turtle*, choć warto zauważyć, że istnieją ogólnie dostępne aplikacje służące do tłumaczenia poszczególnych formatów serializacji RDF.

Literatura

[1] P. Layman and H. R. Varian: How Much Information. `http://www.sims.berkeley.edu/how-much-info-2003`, (2003).

[2] C. M. Paul Coles, Tony Cox and S. Richardson: The Toxic Terabyte., (2006).

[3] A. Varde, F. Suchanek, R. Nayak, and P. Senellart: Knowledge Discovery over the Deep Web, Semantic Web and XML. In *DASFAA '09 Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, (2009).

[4] Y. T. Lee: Information Modeling: From Design to Implementation. In *Proceedings of the Second World Manufacturing Congress*, (1999).
[5] J. T. Hackos: What is an Information Model & Why do You Need One?. *The Gilbane Report*, 10(1), (2002).
[6] A. Pras and J. Schoenwaelder: RFC 3444: On the Difference between Information Models and Data Models., (2003).
[7] T. F. Verhoef: *Effective information modelling support* Technische Universiteit Delft, (1993).
[8] J.Woźniak-Kasperek: Organizacja informacji w internetowych serwisach kontrolowanej jakości. In *Opracowanie przedmiotowe dokumentów z zakresu nauk ścisłych: matematyczno-przyrodniczych i technicznych.Język haseł przedmiotowych KABA: teoria, praktyka, przyszłość*, Kazimierz Dolny, (2006).
[9] J.Woźniak-Kasperek: *Podstwy budowy tezaurusa* Wydawnictwo SBP, Warszawa, (2005).
[10] B. Sosińska-Kalata: Tezaurusy w zmieniającym się środowisku wyszukiwania informacji. *Informacja w sieci. Problemy, metody, technologie*, (2006).
[11] Documentation – Guidelines for the establishment and development of monolingual thesauri (ISO 2788:1986). Paperback, (2007).
[12] ANSI/NISO Z39.19-2005: Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies (R2010)., (2005).
[13] Structured Vocabularies for Information Retrieval - Guide - Definitions, symbols and abbreviations (BS 8723-1:2005). British Standard, (2005).
[14] Structured Vocabularies for Information Retrieval - Guide - Thesauri (BS 8723-2:2005). British Standard, (2005).
[15] A. Miles and S. Bechhofer: SKOS Reference., (2009).
[16] A. Isaac and E. Summers: SKOS Simple Knowledge Organization System Primer., (2009).
[17] S. D. Clarke, A. Gilchrist, R. Davies, and L. Will: Glossary of terms relating to thesauri and other forms of structured vocabulary for information retrieval.

ROZDZIAŁ 4

SEMANTYCZNE ADNOTACJE DOKUMENTÓW

R.Wojtyna, Ł.Walukiewicz

4.1. Wstęp

Rozwój technologii informatycznych wymusza wymianę informacji pomiędzy oprogramowaniem oraz urządzeniami różnych typów. Problemem nie jest jedynie to, że urządzenia ze względu na swoją odmienność nie „mówią tym samym językiem". W wielu wypadkach są one także zmuszone korzystać z danych stworzonych jedynie przez człowieka dla drugiego człowieka. Łatwo sięgnąć po ogólnie znane przykłady, jak choćby wyszukiwarki internetowe, które zmuszane są do rozumienia zawartości stron WWW. Ich efektywność zależy od tego, w jakim stopniu potrafią to robić. Najpopularniejsze portale, jak Google czy Yahoo, od niedawna radzą sobie z tym problemem poprzez obsługe dwóch potężnych standardów: RDFa i Microformats. Standardy te służą semantycznemu adnotowaniu dokumentów XHTML. Adnotacje te wykorzystują istniejące już treści w dokumencie, dzięki czemu dokumenty nie rozrastają się objętościowo i nie ma konieczności duplikowania informacji.

Ideą tworzenia standardów typu RDFa lub Microformats jest próba stworzenia namiastki sieci semantycznej, co w konsekwencji ma pokazać, w jakim kierunku powinni kierować się twórcy stron internetowych w przyszłości, aby sieć internet odzwierciedlała sieć semantyczną. Sieć semantyczna w założeniu ma posiadać budowę warstwową, składającą się z następujących elementów: Unicode, URI, XML, RDF i Microformats, OWL, mechanizmy wnioskowania, mechanizmy certyfikacji i zaufania.

Warstwa pierwsza, najniższa - Unicode pozwala na wyrażenie w języku maszyn dowolnego znaku pisanego i dowolnego języka. Dzięki czemu zostanie wyeliminowany problem z różnymi czcionkami i alfabetami. Kolejna warsttwa URI ma zapewniać jednoznaczność pojęć - jest to identyfikator danego pojęcia, który informuje maszynę (w przypadku natknięcia się na URI w kodzie) o czym jest dany tekst, poprzez możliwe odwołanie do słownika pojęć. Następnie jest standard XML, pozwalający na strukturyzowanie danych. Kolejną wartstwą są semantyczne adnotacje, które zostaną opisane w poniższych podrozdziałach. Dalej jest OWL, który pozwala na formalne definiowanie ontologii. Kolejne dwie warstwy to

mechanizmy wnioskowania (na razie działają mało efektywanie) i mechanizmy certyfikacji i zaufania. Ta ostatnia warstwa, która praktycznie jeszcze nie istnieje, pozwalałaby na zestandaryzowanie i rozwiązanie problemów autoryzacji użytkowników, identyfikacji ich zasobów, a także określenia praw, na jakich te zasoby są przesyłane i mogą być udostępniane.

4.2. RDF i RDFa

Koncepcja RDFa oparta jest na funkcjonującym standardzie RDF. Poniżej zebrano podstawowe informacje na temat obu tych standardów.

4.2.1. RDF

RDF (ang. *The Resource Description Framework*) jest modelem i językiem służącym do reprezentacji zasobów dostępnych w sieci Internet. Został on po części zaprojektowany do reprezentacji opisu zasobów za pomocą metadanych, w których zawarte są informacje o ich autorze, licencjonowaniu, czasem modyfikacji itp. Można go jednak użyć właściwie do opisu wszystkiego, z czym możemy spotkać sie w Internecie. Opis ten umożliwia maszynom (a więc oprogramowaniu) przetwarzanie danych, które oryginalnie były zrozumiałe jedynie dla człowieka.

Idea RDF oparta jest na używaniu sieciowych znaczników URI (ang. *Uniform Resource Identifiers*) oraz opisywaniu rzeczy poprzez właściwości i ich wartości. Przykładowym zdaniem na temat strony www może być:

http://www.example.org/index.html ma twórcę, którym jest John Smith

W zdaniu tym wyróżnić można trzy elementy:

- rzecz, którą to zdanie opisuje (identyfikowaną przez adres sieciowy),
- właściwość tej rzeczy (ma stwórcę),
- wartość tej właściwości (John Smith).

Dla RDF te trzy elementy określane są, odpowiednio, jako *temat* (ang. *subject*), *predykat* (ang. *predicate*) i *obiekt* (ang. *object*). Obiekty w RDF mogą być określane poprzez stałe (np. *John Smith*) zwane literałami albo poprzez obiekty typu URIref (ang. *URI reference*), takie jak np. adresy www. Zestawy trójek postaci `<temat, predykat, obiekt>` tworzą graf.

Aby zapisać w RDF następujące *twierdzenia*:

```
http://www.example.org/index.html
   has a creator whose value is John Smith.
http://www.example.org/index.html
   has a creation-date whose value is August 16, 1999.
http://www.example.org/index.html
   has a language whose value is English.
```

należy sprowadzić je do poniższej postaci (fragmentu grafu RDF) wykorzystując zdefiniowane wcześniej i oznaczone identyfikatorami URI pojęcia: `creator`, `creation-date`, `language`.

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>.

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date>
 August 16, 1999.

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language>
 en .
```

Zapis trójek byłby znacznie krótszy, gdyby dało się zrezygnować z pisania pełnych adresów URI. Jest to możliwe poprzez tworzenie prefiksów. Prefiksy reprezentują przestrzenie nazw, dzięki czemu można je wykorzystać do skracania adresów. Koncepcja ta wywodzi się od XML'owego *qualified name* (`QName`). Jeśli więc prefiksowi `foo` przypisana będzie przestrzeń nazw `http://example.org/somewhere/`, wtedy wyrażenie typu `foo:bar` będzie skrótem dla dłuższego URI-ref `http://example.org/somewhere/bar`. W tab. 4.1 zamieszczono zestawienie kilku „dobrze znanych" przestrzeni nazw i ich prefiksów. Definiując dodatkowo

Tab. 4.1: Zestawienie popularnych przestrzeni nazw i ich prefiksów

prefiks	URI przestrzeni nazw
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
dc:	http://purl.org/dc/elements/1.1/
owl:	http://www.w3.org/2002/07/owl#
ex:	http://www.example.org/
xsd:	http://www.w3.org/2001/XMLSchema#

prefiksy jak w tab. 4.2 trójki z powyższczego przykładu można zapisać następu-

Tab. 4.2: Zestawienie własnych przestrzeni nazw i ich prefiksów

prefiks	URI przestrzeni nazw
exterms:	http://www.example.org/terms/
exstaff:	http://www.example.org/staffid/
ex2:	http://www.domain2.example.org/

jąco:

```
ex:index.html  dc:creator             exstaff:85740 .
ex:index.html  exterms:creation-date  "August 16, 1999" .
ex:index.html  dc:language            "en" .
```

Jest to zdecydowanie bardziej wygodna i krótka forma niż ta, która została użyta na początku.

4.2.2. Bliżej o RDFa

RDFa (ang. *RDF in attributes*) jest sposobem na zamieszczanie w znacznikach dokumentu XHTML twierdzenia w języku RDF. Aby tego dokonać RDFa używa *atrybutów* obecnych w języku XHTML, ale również dostarcza kilku własnych. Typowe atrybuty dla XHTML nie zmieniają znaczenia, jednak może ulegać zmianie część ich składni. Wykorzystywanymi atrybutami są:

- `@rel` Lista CURIE (compact URI) oddzielonych spacją. Służy do wyrażania relacji pomiędzy zasobami (predykatami w terminologii RDF).
- `@rev` Lista CURIE oddzielonych spacją. Określa odwrotną relację pomiędzy zasobami (także predykatami).
- `@content` Zawiera zawartość czytelną dla maszyny (literał).
- `@href` URI dla wyrażenia zasobu partnerskiego relacji.
- `@src` URI dla wyrażenia zasobu partnerskiego relacji, gdy ten zasób jest osadzony.

Natomiast nowe atrybuty, charakterystyczne dla RDFa, to:

- `@about` URI lub SafeCURIE (ang. *safe compact URI*) używane do określenia, o czym jest zasób (*subject* w terminologii RDF). Może nim być cała strona, jednak niekoniecznie. Przeważnie jest to jej część. Jeśli istnieje atrybut `@about`, wszystkie zagnieżdzone w nim atrybuty są do niego odnoszone. Jeśli `@about` nie istnieje, wszystkie atrybuty są odnoszone do aktualnej strony. Przy użyciu tego argumentu można określać wlaściwości i relacje dla dowolnych elementów dokumentu.
- `@property` lista CURIE oddzielona spacją używana do określenia relacji między tematem i tekstem (*predicate*). Jeśli nie występuje wraz z artybutem `@content`, wtedy wartość będzie pobierana jako tekst elementu, np.:

  ```
  Author: <em property="dc:creator">Mark Birbeck</em>
  ```

  Nastąpi automatyczne wskazanie na wartość `Marka Birbeck` jako nazwę autora. Jeśli w tym samym miejscu zostanie użyty atrybut `@content`, jego wartość będzie ważniejsza.

  ```
  <img src="img1.png" property="dc:created" content="2009-03-22"/>
  ```

  W takim wypadku nie ma innej możliwości i data przekazana jest poprzez atrybut `@content`.
- `@resource` URI lub SafeCURIE używane do określenia partnera w relacji, którego nie można kliknąć (obiekt).
- `@datatype` CURIE reprezentujacy typ danych dla wyrażenia typu danych literału.
- `@typeof` CURIE oddzielone spacjami, określające typ danych do powiązania z tematem.

Terminologia RDFa

Twierdzenia (ang. *statements*) to znormalizowane jednostki informacji, stworzone w określonym formacie pozwalającym na ich łatwe przetwarzanie. Dla przykładu typowy fragment informacji:

```
Albert was born on March 14, 1879, in Germany.
There is a picture of him at the web address:
http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

można przetworzyć na zestaw faktów:

```
Albert was born on March 14, 1879.
Albert was born in Germany.
Albert has a picture at
http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

Trójki `<temat, predykat, obiekt>` pozostają niezmienionym pojęciem. Temat określa, o czym są twierdzenia. W powyższym przykładzie tematem wszystkich twierdzeń jest `Albert`. Predykat mówi o opisywanej właściwości. W przykładach predykatami są: miejsce urodzenia (`was born in`), czas urodzenia (`was born on`) i zdjęcie. Natomiast obiekt to wartość właściwości, czyli `"March 14, 1879"`, `"Germany"` i adres http.

Referencje URI mogą być wykorzystywane w twierdzeniach, o czym wspomniano podczas charakterystyki RDF. Natomiast jeszcze nic nie powiedziano o sposobie ich wykorzystania. Odwołując się do przykładu opisującego Alberta można zauważyć, że w zasadzie twierdzenia zostały sformułowane tak, aby czytający je człowiek domyślił się, że `Albertem` jest *Albert Einstein*. Maszyna (czy też program) nie może jednak wysuwać takich domysłów. Dodatkowo nie ma prawa wiedzieć, że predykat `"was born in"` jest równoważny z `"birthplace"`, który mógłby istnieć w innej przestrzeni nazw. Ten właśnie problem rozwiązywany jest poprzez użycie URIref.

Aby jasno wskazać na Einsteina można skorzystać z zasobów dbpedia.org udostępniających dane RDF Wikipedii. Można też przy okazji jasno sprecyzować kraj, w którym się urodził.

```
<http://dbpedia.org/resource/Albert_Einstein>
   has the name
   "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
   was born on
   "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
   was born in
   <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
   has a picture at
```

```
   <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

Idąc dalej, można wskazać, że "birthplace", "place of birth", "Lieu de naissance" itd. to te same pojęcia:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name>
  "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
  "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/birthPlace>
  <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/depiction>
  <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

Gładkim literałem (ang. *plain literal*) jest zwykłym ciągiem znaków bez informacji o typie ani języku. W poniższym przykładzie takim literałem jest "Albert Einstein".

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name>
  "Albert Einstein".
```

Typowane literały (ang. *typed literals*) posiadają URI definiujący ich typ. Zwykle typy te pochodzą z XML Schema Datatypes.

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
   "1879-03-14"^^<http://www.w3.org/2001/XMLSchema#date>.
```

4.2.3. Zapis grafów RDF

Grafy RDF można zapisywać w różny sposób. Normatywnym formatem jest RDF/XML, ale obok niego funkcjonują inne formaty, jak np. N-Triples czy Turtle.

Notacja Turtle jest bardzo wygodnym, czytelnym dla człowieka sposobem zapisu grafów RDF. W notacji tej wykorzystuje się przestrzenie nazw i prefiksy RDF, co pozwala skrócić zapis długich ciągów URI. Przykład z Albertem zapisany w notacji Turtle ma następującą postać:

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
dbr:Albert_Einstein dbp:dateOfBirth "1879-03-14"^^xsd:date .
```

```
dbr:Albert_Einstein foaf:depiction
<http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg> .
```

CURIE jest skrótem od ang. *Compact URI*. Deklaracja takiego „kompaktowego" adresu w dokumencie XHTML może wyglądać następujaco:

```
<div xmlns:db="http://dbpedia.org/">
  ...
</div>
```

Przestrzenie nazw wykorzystuje się przy zamieszczania atrybutów RDFa w dokumentach XHTML:

```
<div xmlns:db="http://dbpedia.org/">
  <div about="[db:resource/Albert_Einstein]">
    ...
  </div>
</div>
```

Warto wspomnieć, że deklaracje przestrzeni nazw mają swój zakres. Możliwym jest więc użycie tego samego CURIE w dwóch różnych znaczeniach:

```
<div xmlns:dbr="http://dbpedia.org/resource/">
  <div about="[dbr:Albert_Einstein]">
    ...
  </div>
</div>
<div xmlns:dbr="http://someotherdb.org/resource/">
  <div about="[dbr:Albert_Einstein]">
    ...
  </div>
</div>
```

Możliwe zastosowania CURIE i URI w RDFa przedstawiono poniżej.

- `@about` i `@resource` - obsługuje URI i CURIE
- `@href` i `@src` - obsługuje tylko URI
- `@property`, `@datatype` i `@typeof` - obsługuje tylko CURIE
- `@rel` i `@rev` - obsługuje XHTML link types i CURIE.

Przykład opisu semantycznych adnotacji

W przykładzie poniżej pokazano, jak można zamieszczać semantyczne adnotacje w dokumencie HTML.

```
<html xmlns:dc="http://purl.org/dc/terms/">
 <head> <title>RDFa: Now everyone can have an API</title> </head>
 <body> <h1>RDFa: Now everyone can have an API</h1>
  Author: <em property="dc:creator" content="Mark Birbeck">
```

```
    Mark Birbeck</em>
  Created: <em property="dc:created" content="2009-05-09">
    May 9th, 2009</em>
  License: <a rel="license"
   href="http://creativecommons.org/licenses/by-sa/3.0/">
    CC Attribution-ShareAlike</a>
  Previous version: <a rel="dc:replaces" href="rdfa.0.8.html">
    version 0.8</a>
 </body>
</html>
```

Na samym początku przykładu zdefiniowano prefiks `dc` dla przestrzeni `http://purl.org/dc/terms/`, co pozwoliło użyć krótszej formy identyfikatorów. Dalej można zauważyć kilka szczególnych właściwości.

W oryginalnej składni HTML atrybut `@content` mógł być użyty jedynie w znacznikach `meta` sekcji `head`. W dokumentach z RDFa może on być użyty dla dowolnego elementu. Zrezygnowano z używania `@name` w dalszych częściach dokumentu. Wprowadzono za to nowy atrybut `@property`. Podobnie w przypadku `@rel, @href`, które teraz mogą także wskazywać na relacje pomiędzy obrazkiem i innymi częściami dokumentu. Ponadto wiele powiązań i relacji można jednocześnie opisywać dla jednego elementu:

```
<img src="image1.png" rel="license"
      href="http://creativecommons.org/licenses/by-sa/3.0/"
      property="dc:created" content="2009-05-01" />
```

W przykładzie poniżej pokazano użycie atrybutu `@about` dla dowolnego elementu dokumentu:

```
<a
  about="http://www.slideshare.net/fabien_gandon/rdfa-in-a-nutshell-v1"
  rel="license" href="http://creativecommons.org/licenses/by/2.5/"
  property="dc:creator" content="Fabien Gandon">
  <img src="http://i.creativecommons.org/l/by/2.5/80x15.png" />
</a>
```

Zastosowany zapis pozwala w efektywny sposób określić dwie rzeczy. Pierwsza: *Prezentacja na slideshare pod określonym adresem ma licencję CC BY* oraz druga: *Prezentacja na slideshare pod określonym adresem została stworzona przez Fabiena Gandona.*

Powiązania

RDFa daje możliwość tworzenia powiązań pomiędzy twierdzeniami. Dzięki temu można wykorzystywać istniejące już pojęcia i określenia w dokumencie bez konieczności ich kolejnego przepisywania. Sposoby zamieszczania twierdzeń w kodzie XHTML zademonstrowano w kolejnym przykładzie.

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">
  1879-03-14</span>
  <div rel="dbp:birthPlace" resource=
 "http://dbpedia.org/resource/Germany">
    <span property="dbp:conventionalLongName">
 Federal Republic of Germany</span>
  </div>
</div>
```

Powyższy zapis oznacza, że: *Albert Einstein urodził się w Niemczech, których dłuższą nazwą jest "Federal Republic of Germany"*.

Możliwe jest również tworzenie zagnieżdżonych powiązań o odwrotnym działaniu niż zaprezentowane.

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <div rel="dbp:citizenship"
  resource="http://dbpedia.org/resource/Germany"></div>
  <div rel="dbp:citizenship"
  resource="http://dbpedia.org/resource/United_States"></div>
</div>
```

Powyższy przykład mówi, że Albert posiada jednocześnie niemieckie i amerykańskie obywatelstwo.

Można także stworzyć niekompletną trójkę, która będzie uzupełniona (dwukrotnie) zagnieżdżanymi informacjami.

```
<div about="http://dbpedia.org/resource/Albert_Einstein"
  rel="dbp:citizenship">
  <span about="http://dbpedia.org/resource/Germany"></span>
  <span about="http://dbpedia.org/resource/United_States"></span>
</div>
```

Efektem powyższego zapisu jest utworzenie trójek:

```
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/Germany> .
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/United_States> .
```

Wymagania odnośnie budowy dokumentu

Poprawnie zredagowany dokument XHTML+RDFa powinien spełniać wymagania opisane w standardzie.

- Dokument musi posiadać poprawna deklaracje namespace dla URI:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

- Dokument musi posiadać poprawny atrybut `@version` z odpowiednim numerem wersji

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
    version="XHTML+RDFa 1.0" xml:lang="en">
  <head> <title>Virtual Library</title> </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.
    </p>
  </body>
</html>
```

- Dokument musi spełniać wymagania określane definicji typu dokumentów XHTML+RDFa zawartej pod adresem:

```
http://www.w3.org/TR/rdfa-syntax/#a_xhtmlrdfa_dtd
```

4.3. Mikroformat

4.3.1. Czym są i czym nie jest mikroformaty?

Mikroformaty są wykorzystywane sę przy tworzeniu stron internetowych. Pozwalają skorzystać ze strukturalnego opisu wiedzy. Wszystko po to, aby dokument np. w języku HTML, posiadał składnię rozumianą przez maszyny, które przetwarzają informacje w nim zawarte. Dokładniej mówiąc, mikroformaty są ogólnie przyjętym ustandaryzowanym zestawem elementów (głównie klas CSS i elementów języka XHTML) przystosowanych do przetwarzania przez w maszyny w jednoznaczne i zrozumiały sposób. Innymi słowy, jest to taki zbiór sposobów stosowania klas CSS i elementów języka XHTML, który pozwala zapisać w dokumencie wiedzę w sposób strukturalny i przejrzysta nie tylko dla człowieka, ale co najważniejsze, dla maszyny.

Stosowanie mikroformatów w dokumentach (np. na stronach internetowych) wzbogaca ich semantykę kodowania, ułatwia maszynowe przetwarzanie treści i umożliwia przenoszenie informacji z dokumentu do aplikacji. Mikroformaty są rodzajem metadanych. Jak można przeczytać pod adresem: `http://microformats.org/about`: *Designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards.*

Przez mikroformaty można także rozumieć zestaw zasad projektowania opisu wiedzy, dzięki którym wiedza zapisana w dokumencie staje się przejrzysta. Chodzi przede wszystkim o budowę mikroformatów, jak np. mikroformat **hCard** przeznaczony do opisywania kontaktów (ludzi, firm, organizacji itd.). Opisując ludzi przy użyciu **hCard** wiadomo, że należy użyć odpowiednich klas do nazwy - **fn**, czy adresu - **addr**. Przestrzeganie zasad projektowania opisu wiedzy przy użyciu

mikroformatu jest bardzo istotne z punktu widzenia aplikacji przetwarzających dokument, gdyż jeżeli będzie on zawierał błędy na szczeblu projektowym, to program przetwarzający dane osiągnie słabsze rezultaty.

Idąc dalej, mikroformaty są obiektami zaadoptowanymi do aktualnych wzorców i sposobów korzystania. Nowe mikroformaty tworzone są tylko w celu wykorzystania w istniejących już aplikacjach. Korzystanie z mikroformatów w kodach źródłowych stron prowadzi do utworzenia **lower-case semantic web** (małych sieci semantycznych). Główne ich cechy to:

- małe fragmenty łączące się ze sobą,
- znaczniki semantyczne dodawane są do istniejących stron
- wiedza musi być najpierw zrozumiała dla człowieka, a następnie dla maszyny.

W odpowiedzi na pytanie „Czym są mikroformaty?" można uzyskać także odpowiedź, że są one ewolucyjną rewolucją *Ąn evolutionaly revolution."* [1]. W stwierdzeniu tym chodzi przede wszystkim o możliwość stworzenia globalnej sieci semantycznej, w której mikroformaty mają zapewnić przejrzystość dokumentów dla człowieka i dla maszyny oraz jednoznaczność interpretacji informacji zawartej w znacznikach.

Bardzo ważną cechą tego standardu jest to, iż nie jest on osobnym językiem, czy też dodatkiem do jakiegoś języka. Standard ten opisuje zastosowanie klas CSS i elementów języka xHTML. Nie potrzeba żadnego dodatkowego kompilatora, dlatego też mikroformaty są praktycznie kompatybilne z wszystkimi istniejącymi już stronami - nie wymagają żadnych technicznych zmian w istniejącej infrastrukturze.

Mikroformaty nie są także rozszerzalnymi w nieskończoność strukturami. Nie opisują one całego świata, dlatego też nie narzucają nikomu swoich narzędzi służących do opisu informacji. Z założenia mikroformaty nie są także podejściem do opisu wiedzy, który odrzucałby całkowicie wszystko, co było przed nimi. Jest jasno powiedziane, że wiedza nie posiadająca żadnej semantyki może być wzbogacona o znaczniki mikroformatu.

4.3.2. Charakterystyka i budowa Mikroformatów

Główne założenia

Głównym założeniami przyjętymi przy tworzeniu mikroformatów były:

- Prosty, powtarzalny format, gdyż ma on być łatwy do nauki i użytkowania.
- Tworzenie mikroformatów do funkcjonowania w istniejących aplikacjach.
- Możliwość implementacji w językach skryptowych takich, jak PHP, Python i Perl.
- Użytkowanie mikroformatów przez webmasterów, ponieważ tylko wtedy będzie on rozpowszechniany.

Mikroformaty spełniające te założenia to:

- RSS - używany w przesyłaniu nagłówków informacji,
- xfn - do określania relacji międzyludzkich

- GeoURL - używany do lokalizacji
- hCalendar - pozwala na określanie dat wydarzeń
- hCard - służy do tworzenia książki adresowej
- XOXO - związany z tworzeniem zarysów i subskrypcji
- Attention.XML - pozwala opisywać najczęściej odwiedzane strony.

Charakterystyka

Mikroformaty można stosować w takich technologiach, jak: HTML, XHTML, XML, RSS, ATOM. Mikroformaty występują w następujących atrybutach: `class`, `rel`, `rev`.

Często stosuje się ⟨*SPAN*⟩ i ⟨*DIV*⟩ do zaznaczenia fragmentów, które dotyczą danej informacji. Aktualnie istnieje mniej niż 20 przyjętych i używanych mikroformatów. Najpopularniejsze w użyciu są **hCard** i **hCalendar**. Są zgodne z zapisem formatów odpowiednio **vCard** i **vCalendar** w xHTML-u, umożliwiając jednocześnie zapis jednego formatu w drugim (zagnieżdżenie). Oba powyższe formaty używane są często przez zewnętrzne programy do synchronizacji informacji.

Inne ważne mikroformaty to **rel-nofollow**, **rel-license** i **rel-tag** - polegają na określeniu atrybutu **rel**. Pierwszy z nich określa, że roboty przeszukujące daną stronę nie powinnny podążać za danym linikiem. Kolejny informuje na jakich zasadach została opublikowana treść strony. Natomiast ostatni mówi, że dany link jest odnośnikiem do podstrony ze słowem kluczowym.

Oprócz powyższych standardów występują też takie, których zastosowanie jest ukierunkowane wyłącznie na dany typ strony, np. **VoteLinks** - strony z sondami lub **XFN** (ang. *HTML Friends Network*) - do sieci społecznościowych lub blogów, aby opisać zależności ze stronami innych internautów.

4.3.3. Różnica pomiędzy HTML, a HTML z mikroformatami

Różnicę pomiędzy zwykłymi plikami HTML, a plikami HTML wzbogaconymi o mikroformaty najlepiej omówić jest na przykładie

```
<div>
   <img src="www.photo.pl/robertkowalski" />
   <strong>Robert Kowalski</strong>
   Inzyniet konstruktor firmy Urzadzenia AGD
   ul. techniczna 154
   Raciborz, woj. slaskie, 12-345
</div>
```

Powyżej znajduje się kod dotyczący osoby napisany w zwykłym języku HTML, bez użycia metadanych.

```
<div class="vcard">
   <img class="photo" src="www.photo.pl/robertkowalski" />
   <strong class="fn">Robert Kowalski</strong>
   <span class="title">Inzynier konstruktor</span> firmy
   <span class="org">urzadzenia AGD</span>
   <span class="adr">
```

```
      <span class="street-address">ul. techniczna 154</span>
      <span class="locality">Raciborz</span>,
      <span class="region">woj. slaskie</span>,
      <span class="postal-code">12-345</span>
   </span>
</div>
```

Powyższy kod dotyczy tych samych informacji, co poprzedni z tym, że do opisu znaczeniowego użyto mikroformatów.

W interpretacji powyższych przykładów pomocne są następujące wyjaśnienia:

- Całość została objęta w mikroformat **hCard**, który w języku HTML zapisywany jest jako **vcard**.
- Właściwości jakie zostały przypisane do mikroformatu **hCard** to: `class=photo` - zdjęcie, `class=fn` - nazwa, `class=title` - stanowisko, `class=adr` - adres.
- Właściwość `class=adr` posiada trzy właściwości podrzędne: `class=street-address` - ulica, `class=locality`, `class=postal-code` - kod pocztowy.

Plikiem HTML z użyciem mikroformatów jest bardzo dokładnie opisany przez metadane, dzięki czemu roboty odwiedzające strony z takim kodem w znacznie łatwiejszy sposób mogą dokonywać przetwarzania zawartej na niej wiedzy.

4.3.4. Przykłady

Poniżej znajduje się wyjaśnienie, w jaki sposób należy interpretować kod korzystający z **hCard**.

```
<div class="vcard">
  <span class="fn n">
    <a class="url" href="http://kowalski.pl">
      <span class="given-name">Jan</span>
      <span class="family-name">Kowalski</span>
    </a>
  </span>
  <span class="nickname">kowal</span>
  <a class="email" href="mailto:kowal@kowal.net">
    <span class="type">pref</span><span>erred email</span>
  </a>
  <span class="org">Firma</span>
  <span class="adr">
    <abbr class="type" title="dom">Polska</abbr>
    <span class="type">dom</span> address
    <abbr class="type" title="postal">mail</abbr> and
    <abbr class="type" title="parcel">shipments</abbr>:
```

```
    <span class="street-address">12 Mickiewicza</span>
    <span class="locality">Kalisz</span>
    <span class="postal-code">52-509</span>
    <span class="country-name">Polska</span>
  </span>
  <span class="geo">
    <abr class="latitude" title="52.816607">N 52 81.6607</abr>
     <abr class="longitude" title="12.365667">E 12 36.5667</abr>
  </span>
</div>
```

- W pierwszymi wierszu występuje atrybut `class=vcard`. Oznacza on, że cały blok informacji będzie dotyczył kontaku. Mikroformat `hCard` w kodzie HTML jest oznaczony jako obiekt `vcard`.
- Blok `kontakt` zawiera takie właściwości jak:
  - atrybut `class=fn n` - nazwa składająca się z kilku elementów, posiada właściwości podrzędne, jak imię i nazwisko kontaktu, oraz adres jego strony internetowej.
  - atrybut `class=nickname` - pseudonim, posiada właściwość adresu poczty internetowej.
  - atrybut `class=Org` - organizacja z której wywodzi się dany kontakt
  - atrybut `class=adr` - adres danego kontaktu, posiada najwiecej właściwości podrzędnych: `class=street-address` - ulica, `class=locality` - miejscowość, `class=postal-code` - kod pocztowy i `class=country-name` - kraj.
  - atrybut `class=geo` - określenie lokalizacji danego kontaktu.

Poniżej znajduje się wyjaśnienie, w jaki sposób należy interpretować kod korzystający z **hCalendar**.

```
<div class="vevent">
  <h5 class="summary">
    Mecz pilkarski FC Barcelona - Real Madryt. </h5>
  <div>Posted on: <abbr class="dtstamp" title="19970901T1300Z">
    September 1, 1997</abbr></div>
  <div class="uid">20101129T203000Z-123402@host.com</div>
  <div>Dates:
          <abbr class="dtstart" title="20101129T203000Z">
          November 29, 2010, 20:30 UTC </abbr>-
    <abbr class="dtend" title="20101129T203000Z">
                November 29, 2010, 22:30 UTC</abbr>
  </div>
</div>
```

- W pierwszym wierszu znajduje się atrybut `class=vevent`, który oznacza, że cały blok informacji będzie dotyczył wydarzenia.

- W drugim wierszy znajduje się atrybut `class=summary` po którym występuje temat określający opisywane wydarzenie. Kolejną właściwością znajdująca się w bloku opisywanego wydarzenia jest czas rozpoczęcia `class=dtstart` i zakończenia `class=dtend` wydarzenia.

Poniżej znajduje się wyjaśnienie, w jaki sposób należy interpretować kod korzystający z **hReviewer**

```
<div class="hreview">
   <span class="item">
     <strong class="item">Opinia na temat trenera
       <span class="fn">Dotychczasowa praca trenera</span>
     </strong>
   </span>
   <span class="reviewer vcard">
  Autor: <span class="fn">Jan Kowalski</span>,
    <span class="title">programista</span>serwisu
    <span class="org">Firma</span>
   </span>
  Ocena: <span class="rating">4.5</span> na 5
   <span class="description">
      Uwazam, ze trener prowadzi druzyne
      w dobrym kierunku i w przyszlosci pojawia sie sukcesy.
   </span>
</div>
```

- W pierwszym wierszu powyższego kodu jest atrybut `class=hreview`, który oznacza, że tekst zawarty w całym bloku dotyczy opinii.
- W celu określenia autora opinii można użyć atrybutu `class=reviewer`. W powyższym przykładzie w celu określenia dodatkowych informacji użyto dwóch poleceń pisanych w jednym wierszu i oddzielonych znakiem spacji - `class=reviewer vcard`.
- Atrybuty w bloku `reviewer vcard` opisują imię i nazwisko autora opinii - `fn`, tytuł autora opinii - `title` oraz firmę zatrudniającą go - `org`.
- Atrybuty w bloku `hreview` dotyczą najpierw nazwy opinii - atrybut `class=fn`, a następnie oceny przyznanej danemu tematowi. Na samym końcu pojawia się właściwość dotycząca opisu (komentarz) oceniającego na temat ocenianego obiektu - `class=description`.

4.4. Praktyczne wykorzystanie adnotacji semantycznych

W celu przetestowania możliwości stosowania adnotacji semantycznych zbudowano prototyp aplikacji internetowej pozwalającej na kontekstowe wyszukiwanie informacji na wskazanych stronach internetowych. Poniżej opisano przyjęte założenia oraz sposób jej implementacji.

4.4.1. Założenia

Bardzo ważnym etapem w cyklu projektowania aplikacji jest wydzielenie wymagań funkcjonalnych i niefunkcjonalnych. Jest to niezbędny element każdego projektu, bez którego trudno określić założony cel. Dla aplikacji, która miała posłużyć jako ilustracja sposobu wykorzystania semantycznych adnotacji w praktyce zdefiniowano następujące wymagania funkcjonalne:

- użytkownik musi mieć narzędzie do wyszukiwania semantycznego,
- użytkownik musi mieć możliwość wybrania strony internetowej, na której zostanie przeprowadzone wyszukiwanie semantyczne,
- użytkownik musi mieć możliwość wpisania terminu, który będzie wyszukiwany,
- użytkownik powinien móc wprowadzić głębokość przeszukiwania (ilość podstron),
- wyniki powinny być wizualizowane w sposób graficzny,
- interakcja pomiędzy aplikacją a użytkownikiem powinna zachodzić z wykorzystaniem Internetu.

Pierwszy funkcjonalny wymóg projektu informuje projektantów, że efektem finalnym powinna być wyszukiwarka semantyczna, czyli taka, która nie szuka ilości (statystyki) wystąpienia słowa kluczowego na danej stronie, tylko korzystając z ontologii (bazy wiedzy) dokonuje porównania pomiędzy nią, a zawartością strony. Skłania to odrazu do wniosku, że wyniki wyszukiwania semantycznego w dużym stopniu zależą od metadanych zawartych na stronach, które tworzy się używając znaczników opisane w rozdziałach 4.2 i 4.3. Jest to także jeden z celów nadrzędnych projektu - uwidocznienie różnic w wyszukiwaniu semantycznym i statystycznym, a przede wszystkim zwrócenie uwagi na jakość wyszukiwania, która przecież jest najważniejsza.

Oczywiście celem projektu nie jest stworzenie narzędzia do przeszukiwania całego internetu, bądź tworzenie baz danych i indeksowanie wszystkich stron podobnie do znanych wyszukiwarek internetowych Google czy Yahoo. Wspomniane wyszukiwarki wypuszczają w internet swoje „robaki", które „przegryzają się" przez strony, indeksując je w swoich bazach danych, co w dużej mierze ułatwia wyszukiwanie statystyczne (zarówno Google jak i Yahoo wyposażone są w silniki wyszukiwania kontekstowego). Tak więc w realizowanym zadaniu przeszukiwanie całego internetu nie miałoby sensu.

Możliwość przeglądania strony wskazanej przez użytkownika pojawiło się więc jako drugi wymóg funkcjonalny. Użytkownik powinien wpisać ręcznie adres strony startowej, od której zacząć się ma wyszukiwanie. W przypadku niewskazania adresu strony startowej, aplikacja powinna wyświetlić stronę domyślną, wyświetlając przy tym komunikat, iż nie została podana żadna strona startowa (bez podania jej adresu wyszukiwanie nie rozpocznie się). Podobna sytuacja powinna zdarzyć się z użyciem jakiegoś terminu jako parametru wyszukiwania. W sytuacji, gdy użytkownik nie określi terminu, wyszukiwanie powinno zostać uruchomione z parametrem domyślnym lub powinien zostać wyświetlony komunikat o braku parametru wyszukiwania i użytkownik powinien zostać poproszony o jego wpi-

sanie. Intuicyjnie wydaje się, iż powinna być także opcja wpisania dwóch lub też więcej parametrów wyszukiwania (słów kluczowych), które powinny być traktowane jako całość.

Głębokość przeszukiwania jest ważnym parametrem, który określa jak daleko aplikacja ma „zakopać się" w linki znalezione na stronie bazowej i w podstronach strony bazowej. Wartość tego parametru powinien określać użytkownik na podobnej zasadzie co dwa poprzednie parametry. Czyli w przypadku braku wprowadzenia jakiejkolwiek wartość (głębokości przegladania) powinna zostać ustawiona wartość domyślna lub powinien zostać zwrócony komunikat o błędzie i prośba o wprowadzenie takiej wartości.

Kolejnym istotnym wymaganiem jest zwracania rezultatów w sformatowany sposób - najlepiej, gdyby zostały one wyświetlone w jakiejś graficznej repzrezentacji, aby były czytelne. Może to być graficzne drzewo z gałęziami reprezentującymi znalezione dopasowanie (jedna gałąź - jedno dopasowanie) w taki sposób, że im lepsze dopasowanie, tym dłuższa gałąź. Ewentualnie można wyszukane dopasowania wyświetlać w sposób procentowy (procentowe dopasowanie do wzorca z ontologii).

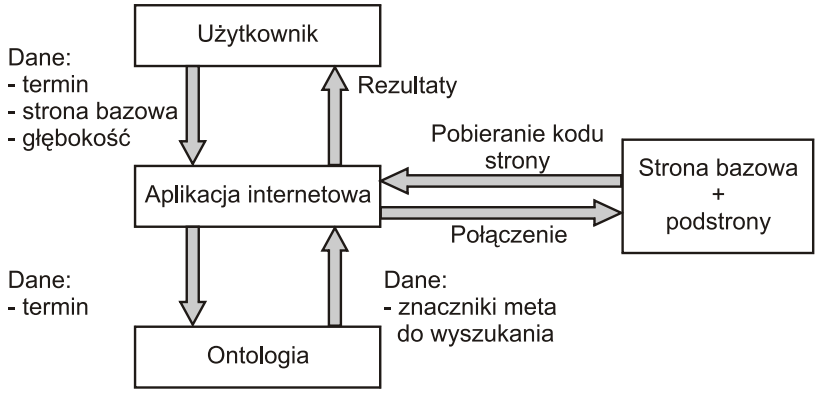
Ostatnim wymaganiem jest niejako podsumowanie wszystkich powyższych, czyli interakcyjność aplikacji. Aplikacja musi zawierać zgrabny interfejs graficzny pomiędzy użytkownikiem, a silnikiem wyszukiwarki, pozwalający na swobodną i łatwą komunikację pomiędzy obydwoma podmiotami.

Wymagania niefunkcjonalne dla testowej aplikacji można zdefiniować jak następuje:

- aplikacja webowa powinna być zaimplementowana jako dynamiczna strona internetowa,
- do semantycznego wyszukiwania powinny być wykorzystane bazy wiedzy (ontologie).
- rozwiązanie powinno mieć architekturę warstwowa,
- wdrożenie powinno odbyć się na serwerze HTTP.

Stworzenie aplikacji webowej wydaje się optymalną formą realizacji postawionych założeń. Dzięki niej łączenie się z przeszukiwaną stroną, pobierania kodu źródłowego tej strony itp. nie powinno sprawiać problemów. Co więcej, implementując aplikację w języku skryptowym jak **Perl** lub **PHP** uzyskuje się rozwiązanie, które nie wymaga kompilacji (niweluje problem z przenośnością oprogramowania) i jest ogólnie dostępne dla każdego użytkownika (dzięki umieszczeniu strony w internecie). Wymienione języki posiadają mocne wsparcie pod postacią bibliotek obsługi połączeń sieciowych oraz parsowania tekstu, co w odgrywa kluczową rolę w szybkim stworzeniu prototypowej aplikacji.

Implementując aplikację webową należy również pochylić się nad problemem wykorzystania istniejącej lub stworzeniem nowej ontologii, czyli stworzenia bazy wiedzy, na podstawie której zostanie dokonane kontekstowe przeszukanie wybranej przez użytkownika strony internetowej. Ontologia dostarcza wzorców, które aplikacja powinna odnaleźć w kodzie parsowanej strony.



Rys. 4.1: Zarys architektury aplikacji.

Na rys. 4.1 przedstawiono zarys architektury budowanej aplikacji. Użytkownik poprzez warstwę dostępową (najwyższą) uruchamia cały projekt, wprowadza dane i otrzymuje rezultaty. Dane od użytkownika przekazywane są do aplikacji, która uruchamia warstwę najniższą - ontologię i odwołuje się do niej poszukując tam terminu wpisanego przez użytkownika. Następnie, jeżeli w bazie wiedzy zostanie odnaleziony dany termin, to meta znaczniki odwołujące się do niego zostają przesłane do warstwy wyższej - aplikacji. Istnieje jeszcze warstwa poboczna, z której aplikacja pobiera kod do przeszukania pod kątem wcześniej wspomnianych meta znaczników. Jeżeli przeszukiwanie zakończy się pozytywnie, to na stronie internetowej (w warstwie dostępowej aplikacji) zostaną wyświetlone rezultaty.

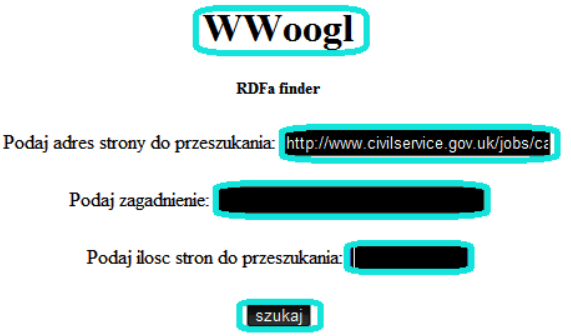
Aplikacja webowa powinna działać w środowisku serwera HTTP, którym może być Apache. Ważne jest, aby serwer obsługiwał skrypty CGI (ang. *Common Getaway Interface*), gdyż aplikacja będzie miała właśnie postać takiego skryptu.

4.4.2. Aplikacja

Do implementacji aplikacji ostatecznie wybrano język Perl. Uzasadniając jego wybór należy zwrócić uwagę na to, iż pierwotnie został on stworzony do pracy z danymi tekstowymi - posiada ogromne wsparcie w postaci bibliotek do parsowania tekstów. Kolejnym ważnym czynnikiem, który przemawiał za tym językiem jest fakt, iż jest on darmowy i dostępny na wiele platform. Do pracy z programami napisanymi w tym języku wystarczy jego interpreter, gdyż pliki nie są kompilowane do postaci binarnej. W celu spełnienia założeń funkcjonalnych zbudowano dynamiczną stronę internetową, generowaną przez skrypt CGI.

Opis użytkowania

Na rys. 4.2 przedstawiono widok interfejsu użytkownika z zaznaczonymi kluczowymi jego elementami:



Rys. 4.2: Widok interfejsu użytkownika.

- Pole z *nazwą przeglądarki* – nazwa przeglądarki *WWoogl* powstała jako zlepek pierwszych liter nazwisk jej autorów (*W*ojtyna i *W*alukiewicz) oraz przyrostka *oogl* (nawiązującego do przeglądarki firmy Google, gdyż przeglądarka ta wspiera wyszukiwanie semantyczne).
- Pole *Podaj adres strony* – służy do wpisania adresu strony internetowej, na której zostanie przeprowadzone wyszukiwanie kontekstowe. Domyślnie jest to strona `http://www.civilservice.gov.uk/jobs/careers-detail.aspx?JobId=14224`, która posiada semantyczne adnotacje wg standardu RDFa.
- Pole *Podaj zagadnienie* – przeznaczone jest do wpisania terminu, który poddany zostanie wyszukaniu.
- Pole *Ilość stron* – służy do określenia głębokości przeszukiwania (ilość stron, które aplikacja może przeszukać).
- Przycisk *Szukaj* – służy do uruchomienia algorytmu przeszukiwania semantycznego.

Podczas działania aplikacji jej sterowanie przechodzi przez kilka warstw:

- interfejs użytkownika - pobranie danych wejściowych,
- baza wiedzy - przeszukiwanie pod kątem wystąpienia terminu,
- wczytana strona - przeszukiwanie pod kątem wyników znalezionych w bazie wiedzy.

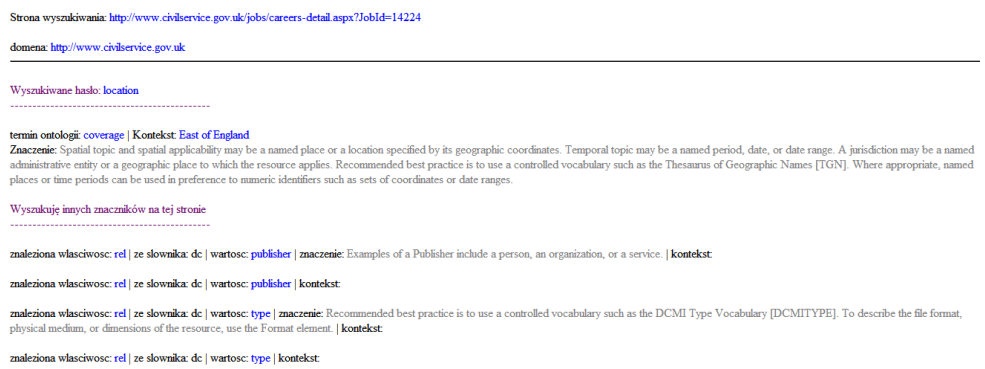
Dokładniej mówiąc, podczas działania aplikacji wykonywane są następujące operacje:

- pobranie danych od użytkownika,
- połączenie się ze stroną, która ma zostać przeszukana
- pobranie kodu strony,
- połączenie się z ontologią `http://purl.org/dc/terms`, która zapisana jest w pliku po stronie serwera http,
- przeszukanie opisu słów w ontologii pod kątem wystąpienia wpisanego przez użytkownika terminu,
- zapamiętanie wszystkich słów (tematów) z ontologii, w których opisach został znaleziony uprzednio podany termin,

- przeszukanie kodu strony pod kątem słów zapamiętanych z ontologii,
- wyświetlenie rezultatów na stronie (interfejsie użytkownika).

Przykładowe rezultaty semantycznego wyszukiwania za pomocą aplikacji WWoogl pokazano na rys. 4.3. Wyszukiwanie odbyło się na stronie domyślnej `http://www.civilservice.gov.uk/jobs/careers-detail.aspx?JobId=14224`, zaś parametrem wyszukiwania był termin `location`. Aplikacja zwróciła następujące rezultaty:

- dla terminu `location` w ontologii wyszukano temat `coverage` (po polsku - obszar)
- temat ten ma właściwość o wartości `East of England`
- aplikacja zwróciła także opis tematu `coverage` zaczerpnięty z ontologii.



Rys. 4.3: Przykładowe rezultaty wyszukiwania.

Zaimplementowana wyszukiwarka po zwróceniu (wypisaniu na stronie) rezultatów dla wyszukiwanego terminu zwraca wszystkie znaczniki RDFa występujące na przeszukiwanej stronie. Procedura wypisania jest zawsze taka sama, a mianowicie najpierw wypisywany jest atrybut określający właściwość np. `rel`, następnie określona jest ontologia, temat, jego znaczenie i na końcu, jeżeli występuje to zwrócony zostaje kontekst.

4.4.3. Podsumowanie

Stworzona aplikacja jest tylko namiastką wyszukiwarki semantycznej. Pokazuje ona w najprostszy sposób na czym polega wyszukiwanie semantyczne. Ze względu na mało zaawansowaną implementację zwracane przez nią rezultaty nie są oszałamiające. Mimo wszystko jednak dzięki niej można zorientować się, na czym polega wykorzystanie wiedzy zawartej na stronach internetowych, a także ukazuje możliwą kategoryzację tej wiedzy.

W trakcie implementacji i testów aplikacji stwierdzono, że znaczniki RDFa obecnie są mało wykorzystywane i dlatego tylko na niektórych stronach wyszukiwanie semantyczne w oparciu o ten standard ma sens. Zauważono, że baza

wiedzy odgrywa taką samą rolę jak stosowanie przez projektantów meta znaczników na stronach. Bogata bazy wiedzy zapewnia rzetelne przeanalizowanie terminów zadanych przez użytkownika oraz skuteczniejsze przeszukiwanie kodu strony pod kątem semantycznym.

Zaimplementowana aplikacja nie wspiera wyszukiwania w oparciu o drugi standard opisu wiedzy, tj. Microformats, dlatego nie dokonano porównania zalet i wad obu standardów w praktyce.

Literatura

[1] An Evolutionary Revolution. http://theryanking.com/entries/2005/04/07/an-evolutionary-revolution/

ROZDZIAŁ 5

ZASTOSOWANIE METODY ANALIZY GŁÓWNYCH SKŁADOWYCH W ROZPOZNAWANIU I KLASYFIKACJI TWARZY

D. Dudek, S. Tomaszewski

5.1. Wstęp

Identyfikacja osób na podstawie obrazu twarzy od dawna budzi duże zainteresowanie. Jest to najbardziej naturalna i najczęściej stosowana przez ludzi forma weryfikacji tożsamości. Począwszy od wczesnych lat 90-tych zagadnienia rozpoznawania twarzy wyraźnie zyskują na popularności. Obecnie – po ponad 30 latach od pojawienia się pierwszych prac z tej dziedziny – odpowiednio wydajny sprzęt komputerowy stał się powszechnie dostępny, jednocześnie pojawiło się duże zapotrzebowanie na stosowanie tej technologii w praktyce.

Konieczność ochrony przed zamachami terrorystycznymi, do których w ostatnim czasie przywiązuje się szczególną wagę, z pewnością przyczyniła się do wzrostu zainteresowania urządzeniami zdolnymi do wykrywania twarzy osób podejrzanych, pojawiających się w miejscach publicznych, takich jak lotniska, dworce, stadiony czy stacje metra. Z drugiej strony, atrakcyjność rozpoznawania twarzy wynika również z faktu wprowadzenia dodatkowego biometrycznego zabezpieczenia, które nie jest okupione obniżeniem komfortu użytkowania systemu. Chociaż inne metody identyfikacji, takie jak analiza odcisków palców czy skanowanie źrenicy, są bardziej niezawodne, wymagają one jednak pewnego zaangażowania ze strony użytkownika. Systemy identyfikujące ludzi na podstawie obrazów twarzy często są w stanie działać sprawnie bez udziału, a nawet bez wiedzy, osoby rozpoznawanej. Dzięki temu wyposażony w moduł rozpoznawania twarzy telewizor mógłby automatycznie blokować określone kanały „widząc", że przed ekranem znajduje się dziecko. Z kolei system identyfikacji osób dokonujących zakupów przez Internet byłby bezpieczniejszy, gdyby – oprócz skontrolowania, czy wprowadzono odpowiednie numery karty kredytowej – mógł sprawdzić, czy przy komputerze siedzi rzeczywiście osoba będąca właścicielem karty. W takich dziedzinach jak wirtualna rzeczywistość czy gry komputerowe rozpoznawanie twarzy

może umożliwić na przykład zmiany zachowania wirtualnych postaci w zależności od tego, kto pojawi się przed monitorem komputera. Przykładowe obszary zastosowań systemów rozpoznawania twarzy to:

- karty z danymi biometrycznymi
  - prawa jazdy, upoważnienia, karty wstępu
  - dokumenty imigracyjne, paszporty, rejestracja wyborców
  - ochrona przed nadużyciami w pomocy społecznej oraz służbie zdrowia
- zabezpieczenie informacji
  - kontrola rodzicielska dostępu do kanałów TV, logowanie do urządzeń osobistych i kont
  - ochrona do dostępu do programów, baz danych, plików
  - bezpieczeństwo połączeń internetowych, kontrola dostępu do Internetu, ochrona danych medycznych
  - bezpieczne zakupy w siecie
- wymiar sprawiedliwości, ochrona budynków
  - zaawansowane systemy ochrony, telewizja przemysłowa
  - kontrola dostępu, analiza danych do śledztwa
  - eliminowanie kradzieży w sklepach, śledzenie podejrzanych, prowadzenie dochodzeń
- rozrywka
  - gry komputerowe, wirtualna rzeczywistość, programy szkoleniowe
  - roboty społeczne

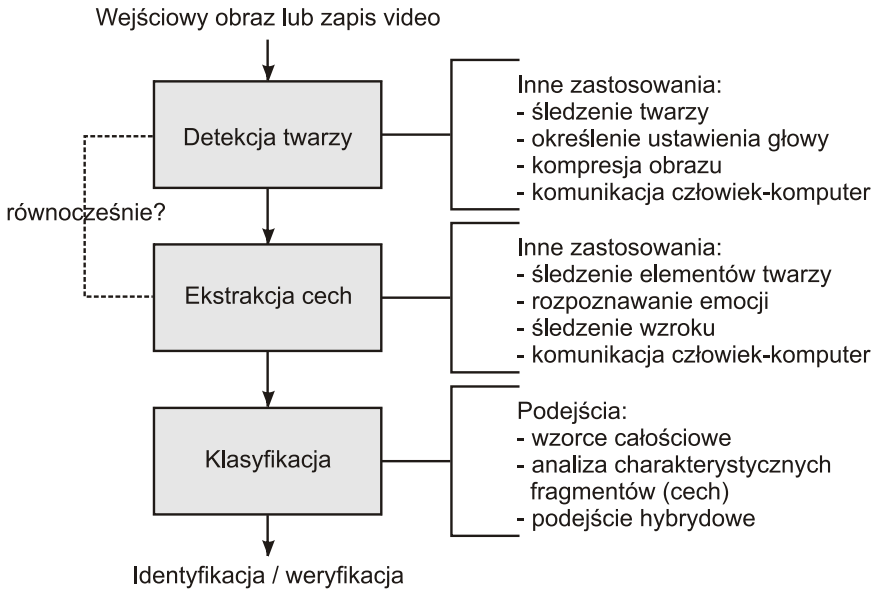
Podczas prac przedstawianych w niniejszym rozdziale wykorzystano bazę danych FERET zawierającą zjęcia twarzy zebrane w ramach programu FERET, sponsorowanego przez DOD Counterdrug Technology Development Program Office (*Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office.*).

5.2. Architektura systemów rozpoznawania twarzy

Zautomatyzowanie procesu rozpoznawania twarzy wymaga (rys. 5.1): a) przeprowadzenia odpowiedniej segmentacji obrazu (detekcji twarzy uchwyconej na zdjęciu); b) wydobycia cech z fragmentu obrazu reprezentującego twarz; c) dokonania klasyfikacji pozyskanych danych, czyli identyfikacji osoby. Czasami wymienione etapy przetwarzania nie są rozłączne – na przykład cechy twarzy używane do identyfikacji często są wykorzystywane już w procesie detekcji. Tak więc lokalizowanie twarzy na obrazie może być połączone z ekstrakcją jej cech.

W przypadku rozpoznawania twarzy problem ekstrakcji cech ma fundamentalne znaczenie. Należy przy tym zaznaczyć, że jest to zagadnienie szczególnie trudne i złożone, bowiem obraz twarzy tej samej osoby może się znacząco zmie-

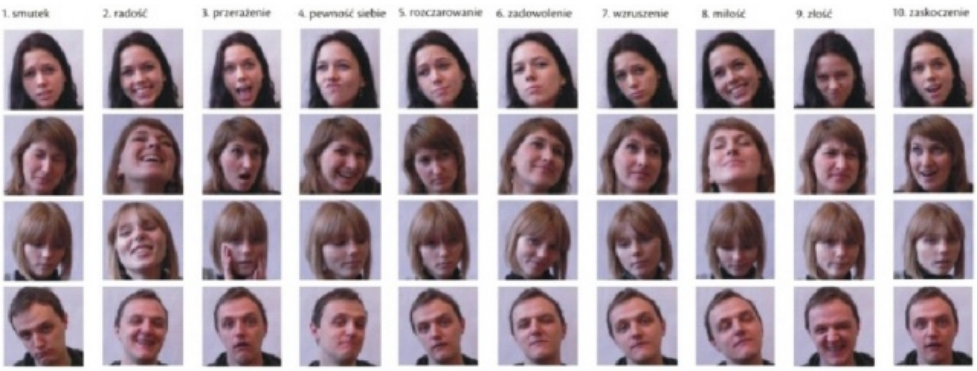
niać w zależności od kąta obserwacji, rodzaju i kierunku oświetlenia czy też nastroju. Twarz człowieka uśmiechniętego wygląda inaczej niż twarz kogoś zdziwionego bądź znudzonego (rys. 5.2). W praktyce istotne okazują się również takie czynniki jak fryzura, makijaż, zarost, noszenie okularów czy nakrycie głowy. Nie bez znaczenia są też zmiany, nierzadko bardzo poważne, powodowane przez upływ czasu.



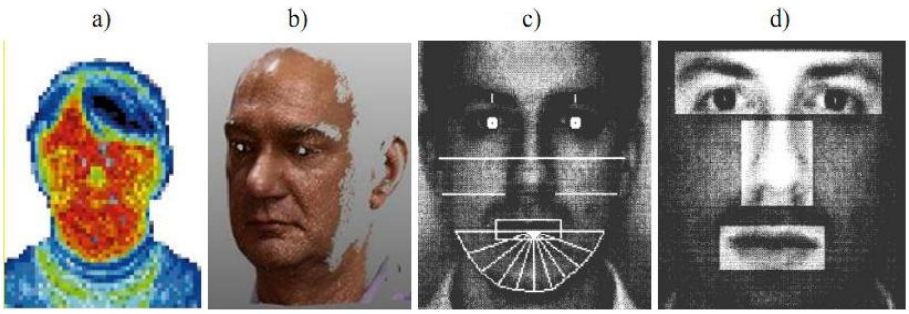
Rys. 5.1: Ogólny schemat systemu rozpoznawania twarzy.

Ekstrakcja cech w automatycznym rozpoznawaniu twarzy odgrywa szczególnie ważną rolę, trzeba bowiem z całego zakresu zmienności danych wychwycić informację inwariantną, zawierającą cechy istotne z punktu widzenia identyfikacji [1]. Można tu wyróżnić pięć najważniejszych typów rozwiązań:

- Podejście oparte o specjalistyczny sprzęt, umożliwiający wydobywanie cech innych niż uzyskiwane z typowych fotografii i wykorzystujący np. techniki termograficzne (rys. 5.3a). Potrzebne są jednak do tego odpowiednie urządzenia, poza tym tego rodzaju rozwiązania są nieprzydatne do celów sortowania czy wyszukiwania zdjęć. Odrębną grupę stanowią metody operujące na obrazie trójwymiarowym, uzyskiwanym za pomocą stereowizji lub specjalizowanych skanerów. Na ogół jednak rozpoznawanie twarzy wymaga przeprowadzania analizy obiektu trójwymiarowego na podstawie dwuwymiarowego obrazu (rys. 5.3b). Już sama natura problemu jest zatem źródłem licznych trudności i ograniczeń.
- Podejście „geometryczne” (rys. 5.3c) – twarz opisywana jest przez zbiór wartości kątów, odległości czy pól, wyznaczonych w oparciu o charakterystyczne



Rys. 5.2: Przykład zmienności danych opisujących twarz tej samej osoby (smutek, radość, przerażenie, pewność siebie, rozczarowanie, zadowolenie, wzruszenie, miłość, złość, zaskoczenie).



Rys. 5.3: Przykłady różnych sposobów ekstrakcji twarzy: a) obraz termograficzny; b) dane ze skanera 3D; c) cechy geometryczne; d) wydzielone fragmenty.

punkty, takie jak środki oczu, koniec nosa itd. Problemem jest w tym przypadku precyzyjne zlokalizowanie odpowiednich fragmentów twarzy. Staje się to szczególnie trudne lub wręcz niemożliwe, gdy zdjęcie nie jest wykonane dokładnie od przodu. Wówczas niektóre istotne części twarzy mogą być nawet całkowicie niewidoczne. Sporządzanie opisu matematycznego poprzez wyznaczanie geometrycznych parametrów było jednym z pierwszych, najbardziej intuicyjnych pomysłów na ekstrakcję cech w rozpoznawaniu twarzy. Obecnie jednak podejście to praktycznie nie jest już rozwijane.

- Podejście bazujące na wzorcach – z twarzy wydzielane są najistotniejsze fragmenty, takie jak oczy, nos czy usta (rys. 5.3d), następnie badana jest zgodność (korelacja) wzorca przechowywanego w bazie z rozpoznawanym obrazem i właśnie wartość korelacji stanowi tu cechę będącą podstawą rozpoznawania. W tym przypadku również niezbędny jest algorytm pozwalający na wydzielenie z fotografii odpowiednich fragmentów. Pewna modyfikacja tego podejścia

wykorzystuje techniki przekształcania obrazów – sprawdzane jest, jak bardzo trzeba zmienić obraz wzorcowy, by uzyskać obraz rozpoznawany. Im większa modyfikacja, tym mniejsze jest podobieństwo obrazów. Wykazano, że takie rozwiązanie ma przewagę nad podejściem geometrycznym, pomimo znacznie większej złożoności obliczeniowej.

- Podejście, w którym wykorzystuje się bezpośrednio informacje o jasności poszczególnych punktów, składających się na obraz twarzy (lub jej charakterystycznego elementu). W tym przypadku zdjęcie traktowane jest jak macierz, do której można stosować różne przekształcenia matematyczne, mające na celu wydobycie informacji. Cechami pierwotnymi są tu po prostu wartości intensywności kolejnych pikseli, które następnie mogą podlegać różnorodnym transformacjom. Szczególnie dużą popularnością cieszy się metoda tzw. „twarzy własnych" (ang. *eigenfaces*), wykorzystująca analizę głównych składowych (ang. *Principal Component Analysis*, PCA), która została przedstawiona w poniższym rozdziale. Poza tym wykorzystywane są też techniki dyskryminacyjne, autokorelacje, falki i wiele innych metod. Ten właśnie sposób wydobywania cech twarzy zyskał w ostatnich latach ogromną popularność i dominuje wśród metod opisywanych w aktualnej literaturze.
- Podejście oparte o deformowalne modele, które zawierają pewną informację ogólną, potrafią jednak samodzielnie się modyfikować, tak aby uzyskać najlepsze dopasowanie do danego obiektu -– w tym przypadku twarzy. Informacje niskiego poziomu, związane z jasnością poszczególnych pikseli są tu wykorzystywane tylko pośrednio, zasadniczą rolę odgrywa pewna wiedza zawarta w strukturze modelu. Cechy twarzy mogą być określone np. poprzez parametry modelu dopasowanego do rzeczywistego obrazu. Rozwiązania wykorzystujące modele są wciąż w stadium badań i eksperymentów. Wiąże się z nimi nadzieję na przełamanie ograniczeń, jakie wykazują inne podejścia, w praktyce jednak ich stosowanie nastręcza wielu trudności.

Należy wspomnieć również o dużej grupie rozwiązań wykorzystujących sieci neuronowe. Tego typu metody występują jednak na ogół w połączeniu z jednym z podejść opisanych powyżej. Z rozpoznawaniem twarzy wiążą się też specyficzne problemy dotyczące selekcji cech i klasyfikacji. Przede wszystkim, liczba wykorzystywanych cech jest na ogół bardzo duża. Nawet wówczas, gdy są to wartości pomiarów geometrycznych, bierze się pod uwagę kilkadziesiąt parametrów. Natomiast w przypadku, gdy jasność każdego z pikseli stanowi odrębną cechę, dla zdjęcia o wymiarach np. 100×100 uzyskujemy wektor, który ma 10000 składowych. Jednocześnie, liczba obrazów uczących (będących do dyspozycji) jest najczęściej niewielka, ponieważ każdą osobę reprezentuje na ogół tylko kilka zdjęć. Prowadzi to do wystąpienia tzw. problemu małego zbioru danych (liczba cech jest niewspółmiernie wielka w stosunku do liczby zdjęć dostępnych na etapie uczenia).

5.3. Analiza głównych składowych

Jednym ze sposobów na wydobycie informacji zawartej w obrazach twarzy, tworzących pewien zbiór uczący, może być uchwycenie najbardziej charakterystycznych różnic występujących wśród tych obrazów. Z matematycznego punktu widzenia optymalną metodą wydobycia i zakodowania informacji o wariancji w zbiorze obrazów jest zastosowanie analizy głównych składowych, określanej również mianem rozwinięcia Karhunena-Loevego [2].

Załóżmy, że obraz twarzy $I(x,y)$ jest dwuwymiarową macierzą N na N w 8-bitowej skali szarości (0-255). Z drugiej strony taki obraz można traktować jako wektor w przestrzeni N^2, zatem typowy obraz rozmiaru 256 na 256 jest wektorem rozmiaru 65 536 lub równoważnie punktem w przestrzeni 65 536 wymiarowej. Wtedy zespół obrazów jest zbiorem punktów w ogromnej przestrzeni.

Operowanie w tak dużej przestrzeni jest trudne. Ideą metody analizy głównych składowych PCA jest znalezienie takich wektorów, które najlepiej będą opisywać obrazy twarzy w takiej przestrzeni [3]. Te wektory tworzą podprzestrzeń, którą nazywaną „przestrzenią twarzy". Każdy wektor jest długości N^2, opisującym obraz rozmiaru $N \times N$ i jest liniową kombinacją oryginalnych obrazów twarzy. Ponieważ te wektory są wektorami własnymi macierzy kowariancji utworzonej z oryginalnych obrazów twarzy i ponieważ są one odzwierciedleniem twarzy, zostały nazwane „twarzami własnymi" lub jak się pojawiają w literaturze „*eigenfaces*" [4, 5].

Niech treningowy zbiór twarzy (przykładowy zbiór przedstawiono na rys. 5.4) będzie oznaczony przez

$$\Gamma_1, \Gamma_2, \ldots, \Gamma_M \tag{5.1}$$

gdzie Γ_i to kolejny obraz. Tak zwana „twarz średnia" zdefiniowana będzie poprzez

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n \tag{5.2}$$

Każda twarz różni się od średniej o wektor

$$\Phi_i = \Gamma_i - \Psi \tag{5.3}$$

W metodzie głównych składowych poszukiwany jest zbiór M ortonormalnych wektorów u_i, które najlepiej opisują rozrzut danych. Wektor i-ty jest wybierany, aby zmaksymalizować kryterium

$$\lambda_i = \frac{1}{M} \sum_{n=1}^{M} (u_i^T \Phi_n)^2 \tag{5.4}$$

Ponieważ wektory u_i tworzą bazę ortonormalną, to zachodzi warunek

$$\forall 1 \leq l,k \leq M \qquad u_l^T u_k = \delta_{lk} \begin{cases} 1, & l = k \\ 0, & l \neq k \end{cases} \tag{5.5}$$

Rys. 5.4: Przykład zbioru twarzy.

Z drugiej strony wektory u_i oraz skalary λ_i są wektorami własnymi oraz wartościami własnymi macierzy kowariancji

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T, \tag{5.6}$$

gdzie macierz A dana jest przez

$$A = [\Phi_1\Phi_2\ldots\Phi_M] \tag{5.7}$$

Na rys. 5.5 przedstawiono przykład twarzy średniej oraz twarzy własnych dla danych z rys. 5.4.

Macierz kowariancji C jest wymiaru N^2 na N^2 i posiada N^2 wektorów i wartości własnych. Jest to duży zbiór danych, który należałoby zawęzić, aby możliwe było jego przetwarzać w rozsądnym czasie. Jeśli ilość danych punktów jest mniejsza niż rozmiar przestrzeni ($M < N^2$) to wtedy będzie tylko $M-1$ głównych wektorów własnych. Pozostałe odpowiadają wartościom własnym równym 0. Zatem zamiast rozpatrywać N^2 wektorów własnych można rozpatrywać wektory własne macierzy M na M i wtedy brać do analizy liniowe kombinacje obrazów twarzy Φ_i. Rzeczywiście biorąc wektory własne v_i macierzy A^TA takie, że

$$A^TAv_i = \mu_i v_i \tag{5.8}$$

i przemnażając lewostronnie przez A otrzymujemy

$$AA^TAv_i = \mu_i Av_i \tag{5.9}$$

Jak widać, Av_i jest wektorem własnym macierzy kowariancji $C = AA^T$. Idąc dalej tym tropem można skonstruować macierz rzędu M na M o następującej postaci $L = A^TA$, gdzie $L_{mn} = \Phi_m^T\Phi_n$. Następnie dla L można znaleźć M wektorów własnych v_l. Te wektory determinują liniową kombinację M treningowego zbioru do

Rys. 5.5: Zdjęcie w lewym górnym rzędzie przedstawia uśrednioną twarz dla zdjęć z rys. 5.4. Pozostałe dwa zdjęcia w górnym rzędzie przedstawiają *eigenfaces* dla znaczących wartości własnych, natomiast dolny rząd *eigenfaces* dla trzech najmniejszych wartości własnych.

formy twarzy własnych *eigenfaces* u_l.

$$u_l = \sum_{k=1} M v_{lk} \Phi_k, \quad l = 1, 2, \ldots, M. \tag{5.10}$$

Dzięki tym przekształceniom można znacznie zredukować ilość wykonywanych obliczeń, przechodząc z wymiaru obrazu N^2 na ilość obrazów zbioru uczącego M. W praktyce M jest dużo mniejsze od N^2, a co za tym idzie, czas wykonywanych obliczeń jest akceptowalny.

Dodawanie nowej twarzy (Γ) polega na przetransformowaniu jej w przestrzeń twarzy za pomocą operacji

$$\omega_k = u_k^T (\Gamma - \Psi), \quad k = 1, 2, \ldots, M. \tag{5.11}$$

Po tym zabiegu tworzony jet wektor wag

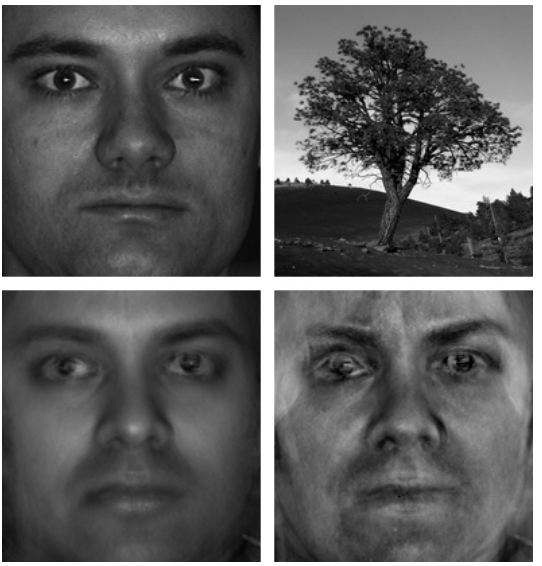
$$\Omega^T = [\omega_1, \omega_2, \ldots, \omega_M] \tag{5.12}$$

który opisuje rzuty obrazu twarzy na poszczególne wektory bazowe zbioru uczącego. Ten wektor wykorzystywany jest w standardowej klasyfikacji, odpowiadającej na pytnie, do której ze wcześniej predefiniowanych klas należy badany obraz. Najprostszym sposobem doboru klasy jest skorzystanie z normy euklidesowej:

$$\epsilon_k^2 = ||\Omega - \Omega_k||^2, \tag{5.13}$$

gdzie Ω_k jest wektorem opisującym k-tą klasę. Ta klasa, która będzie minimalizowała normę będzie determinować klasę badanego obrazu. Twarz jest zaklasyfikowana do klasy k jeśli norma euklidesowa jest mniejsza niż pewna wartość progowa θ_ϵ. W przeciwnym wypadku rezultatem klasyfikacji jest wynik nieznany

lub opcjonalnie nowa twarz jest użyta do stworzenia nowej klasy. Ponieważ tworzenie wektora wag jest równoważne rzutowaniu oryginalnego obrazu twarzy na przestrzeń mało wymiarową, wiele obrazów (niektóre nawet nie wyglądające jak twarz) mogą być omyłkowo podobne to wzorcowego obrazu, co pokazano na rys. 5.6.



Rys. 5.6: Zdjęcie oryginalne (górny rząd) oraz ich transformacje w przestrzeń twarzy (dolny rząd).

Dystans pomiędzy obrazem i przestrzenią twarzy jest równy dystansowi między różnicowym obrazem wejściowym $\Phi = \Gamma - \Psi$ a wektorem $\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$

$$\epsilon^2 = ||\Phi - \Phi_f||^2 \tag{5.14}$$

Istnieją cztery możliwości położeniem obrazu wejściowego względem wzorcowego wektora:

- blisko przestrzeni twarzy i blisko przestrzeni klasy twarzy - wtedy następuje dopasowanie obrazu do klasy,
- blisko przestrzeni twarzy a daleko przestrzeni klas twarzy - wtedy obraz zostaje zakwalifikowany jako nieznany,
- daleko od przestrzeni twarzy i blisko przestrzeni klas twarzy - obraz nie jest twarzą, ale poprzez rzutowanie system może błędnie go zakwalifikować jako pewną twarz,
- daleko od przestrzeni twarzy oraz daleko od przestrzeni klas - obraz nie jest twarzą.

Podsumowując, przedstawiony algorytm rozpoznawania twarzy za pomocą *eigenfaces* składa się z następujących etapów:

1. Znalezienie zbioru indywidualnych twarzy. Ten zbiór powinien zawierać pewną liczbę twarzy każdej osoby z różnymi wariantami oświetlenia lub pozycji (np. 4 zdjęcia 10 osób, stąd $M = 40$).
2. Wyznaczenie macierzy L rozmiaru $M \times M$ (40×40) i znalezienie jej wartości oraz wektorów własnych. Należy wybrać M' wektorów odpowiadające najwyższym wartością własnym (niech $M' = 10$).
3. Znormalizowanie zbioru treningowego. W wyniku otrzymuje się M' ($M' = 10$) twarzy własnych (*eigenfaces*).
4. Wyliczenie wagowego wektora klasy Ω_k dla każdej osoby, poprzez uśrednienie wektorów wzorcowych twarzy własnych Ω wyliczonych z (czterech) oryginalnych obrazów twarzy. Należy przy tym określić próg Θ_ϵ oznaczający maksymalną odległość od przestrzeni klasy twarzy oraz próg Θ_ϵ oznaczający maksymalną odległość od przestrzeni twarzy.
5. Wyliczenie wektora wagowego Ω_k, odległości ϵ_i od każdej klasy oraz odległość ϵ od przestrzeni twarzy dla każdej nowej twarzy, która ma być rozpoznana. Jeśli minimalny dystans $\epsilon_k < \Theta_\epsilon$ oraz odległość $\epsilon < \Theta_\epsilon$ wtedy wejściowy obraz twarzy klasyfikowany jest do klasy odpowiadającej wektorowi wzorcowemu Ω_k. Jeśli $\epsilon > \Theta_\epsilon$ ale $\epsilon < \Theta_\epsilon$ wtedy twarz jest klasyfikowana jako nieznana i opcjonalnie tworzona jest nowa klasa (odpowiadająca temu obrazowi).
6. Jeśli twarz został zakwalifikowana do pewnej klasy to może być użyta do stworzenia nowej klasy przestrzeni twarzy (wymaga to powtórzenia kroków 1-4). Daje to możliwość powiększania przestrzeni twarzy i lepszej klasyfikacji, jako że system posiada w bazie więcej obrazów twarzy.

5.4. Przykładowa implementacja

Algorytm rozpoznawania twarzy może stanowić część systemu kontroli dostępu. Zaimplementowane mechanizmy mogą umożliwiać porównanie i rozpoznanie osoby na podstawie analizy zdjęć z bazy programu. W celu przetestowania samego algorytmu zbudowano prototyp systemu w oparciu o następujące założenia:

- architektura modułowa systemu,
- rozpoznawanie twarzy w oparciu o fotografie (brak możliwości analizy obrazów uzyskiwanych w czasie rzeczywistym),
- fotografie twarzy monochromatyczne i znormalizowane,
- algorytm rozpoznawania twarzy oparty o PCA,
- klasyfikacja twarzy w oparciu o normę euklidesową,
- prototypowanie w środowisku Matlab/Simulink,
- produkt końcowy w C++ i Qt4,
- produkt końcowy dostarczony wraz z dokumentacją, instrukcją instalacji i użytkownika,
- docelowe środowisko uruchomieniowe Windows.

5.5. Badania

Badania działania algorytmu przeprowadzono z wykorzystaniem środowiska `Matlab/Symulink` oraz bazy obrazów FERET [6, 7]. Dla bazy wiedzy składającej się z 10 obrazów twarzy różnych osób zrealizowano trzy scenariusze:

- pojawienie się tego samego zdjęcia,
- pojawienie się zdjęcia osoby będącej w bazie ale z innym wyrazem twarzy,
- pojawienie się nieznanej twarzy.

Podczas każdego z testów badany obraz zapisywano w bazie przestrzeni obrazów zbioru uczącego, rozpinanej przez wyznaczone *eigenfaces*, a następnie badano, z wykorzystaniem normy euklidesowej, dystans pomiędzy obrazem a przestrzenią twarzy.

	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5
ω_1	-5.87e-008	-1.09e-007	-5.38e-008	-5.18e-008	-4.13e-008
ω_2	-3.50e+006	-2.14e+006	4.07e+005	-5.92e+005	-3.42e+006
ω_3	-9.97e+006	7.01e+006	-8.89e+006	4.02e+006	-1.34e+006
ω_4	-6.49e+006	5.85e+006	-1.13e+007	7.43e+006	-4.58e+006
ω_5	1.24e+007	-7.77e+006	-3.23e+006	1.01e+007	-2.26e+006
ω_6	-2.04e+007	1.17e+007	1.87e+007	1.06e+007	-3.30e+006
ω_7	8.20e+006	3.02e+007	-6.79e+004	-1.88e+007	7.29e+006
ω_8	-1.17e+007	-5.75e+006	6.97e+006	-3.78e+007	-1.89e+007
ω_9	1.23e+007	8.79e+006	-4.26e+007	-5.72e+006	6.15e+006
ω_{10}	-6.65e+007	-4.40e+007	-1.55e+007	-2.52e+007	1.40e+008
	Ω_6	Ω_7	Ω_8	Ω_9	Ω_{10}
ω_1	5.76e-008	1.27e-007	-2.37e-008	1.21e-007	3.32e-008
ω_2	-1.01e+007	-4.09e+006	2.25e+006	3.70e+006	1.74e+007
ω_3	3.90e+005	1.17e+007	5.80e+006	-1.26e+007	3.83e+006
ω_4	4.33e+006	-1.13e+007	9.22e+006	1.16e+007	-4.77e+006
ω_5	-2.38e+007	1.01e+007	1.39e+007	2.93e+006	-1.23e+007
ω_6	-1.14e+007	6.17e+006	-1.31e+007	9.84e+006	-8.88e+006
ω_7	-1.41e+007	-9.03e+006	4.35e+006	-4.36e+006	-3.75e+006
ω_8	7.80e+006	2.32e+007	2.07e+007	2.09e+007	-5.36e+006
ω_9	-5.61e+006	2.82e+007	-4.11e+007	3.22e+007	7.39e+006
ω_{10}	-2.00e+007	-3.38e+006	2.55e+007	2.05e+007	-1.17e+007

Tab. 5.1: Tabela wag dla poszczególnych obrazów.

Postępując zgodnie z algorytmem opisanym w punkcie 5.3 dla zbioru uczącego pokazanego na rys. 5.7 wyznaczono średnią twarz (rys. 5.8). Następnie dla

każdego z obrazów policzono jego różnicę z twarzą średnią, co przedstawiono na rys. 5.9. Kolejnym krokiem było wyznaczenie wektorów twarzowych (rys. 5.10) i wektora wag dla każdej z twarzy z bazy (tab. 5.1). Mając wyznaczoną bazę dla przestrzeni twarzy zrekonstruowano twarze na podstawie wyliczonych wag. Wynik tej operacji pokazano na rys. 5.11.

Rys. 5.7: Zbiór uczący wykorzystany przy badaniach (obrazy pochodzą z bazy zdjęć FERET).

Rys. 5.8: Twarz średnia dla przyjętego zbioru uczącego.

Analizując otrzymane rekonstrukcje obrazów twarzy można zauważyć ciekawą prawidłowość. Zdjęcia osób, których twarze mają wyraźnie inny odcień niż tło zostały odtworzone lepiej. Spowodowane jest to małą ilością punktów wyróżniających daną twarz w stosunku do tła (punktów charakterystycznych). W celu poprawienia rezultatów należałoby zmodyfikować obraz twarzy tak, by uwidocznić różnice pomiędzy twarzą a tłem. Dobrym pomysłem do sprawdzenie wydaje się być dopasowanie histogramu barw.

Mając wyznaczone wektory wag dla poszczególnych twarzy oraz bazę przestrzeni obrazów twarzy przystąpiono do badań według zaproponowanych sce-

Rys. 5.9: Różnica każdej z twarzy zbioru uczącego od twarzy średniej.

Rys. 5.10: Wyznaczona baza przestrzeni obrazów.

nariuszy. Każdy z nowych obrazów rozpisano w bazie rozpinającej przestrzeń twarzy (równanie 5.11). Następnie dla wyliczonego wektora wag Ω wyznaczono wektor odległości od znanych systemowi twarzy, korzystając z równania 5.12.

1. **Pojawienie się tego samego zdjęcia**
   Przyjęto, że nowym zdjęciem pojawiającym się na wejściu systemu jest fotografia numer 5. Otrzymano następujące odległości od obrazów ze zbioru uczącego:

nr	1	2	3	4	5	6	7	8	9	10
$\epsilon_a[10^8]$	2.08	1.87	1.67	1.70	**0.00**	1.66	1.54	1.33	1.32	1.55

   Zgodnie z przewidywaniami najmniejsza odległość wystąpiła dla twarzy numer 5.
2. **Pojawienie się zdjęcia osoby będącej w bazie ale z innym wyrazem twarzy**
   Sprawdzono reakcję systemu przy pojawieniu się na wejściu systemu zdjęcia

Rys. 5.11: Zrekonstruowane obrazy twarzy na podstawie wyznaczonego wektora wag i *eigenfaces*.

osoby zapisanej w bazie ale mającej inny wyraz twarzy (inną minę). Obraz dla jakiego przeprowadzono badanie przedstawiono na rys. 5.12a. Odległości od znanych systemowi twarzy wyrażają się następująco:

nr	1	2	3	4	5	6	7	8	9	10
$\epsilon_b[10^7]$	8.28	6.46	**1.42**	5.82	16.2	4.94	7.39	5.70	8.29	5.31

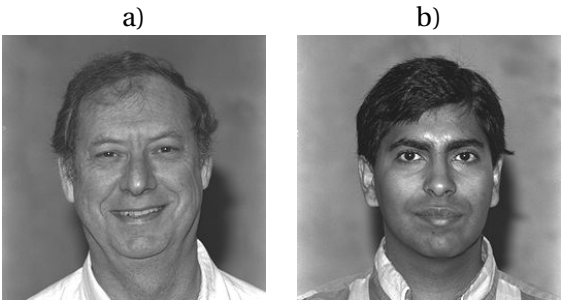
Jak widać współczynnik odległości przyjmuje najmniejszą wartość dla trzeciej twarzy z zestawu uczącego. System poprawnie zakwalifikowałby osobę.

3. **Pojawienie się nieznanej twarzy**
   Sprawdzono również odpowiedź systemu dla nieznanej twarzy. Do testu wykorzystano zdjęcie pokazane na rys. 5.12b. Uzyskano odległości:

nr	1	2	3	4	5	6	7	8	9	10
$\epsilon_c[10^7]$	13.5	11.3	9.21	10.6	8.47	8.90	7.78	5.95	**5.88**	8.09

   Wartość minimalną normy euklidesowej wektora wag badanego obrazu i wag obrazów z bazy otrzymano dla dziewiątego obrazu. Jej wartość jest kilkukrotnie większa od wartości uzyskanej dla innego zdjęcia tej samej osoby, $\min\{\epsilon_c\} = 5.88 \cdot 10^8 > 1.42 \cdot 10^7$, co było prezentowane w poprzednim badaniu. Należałoby przeprowadzić dużo większą liczbę testów i w ich wyniku określić pewną wartość progową ϵ_{gr} przy przekroczeniu której obraz byłby traktowany jako zdjęcie nowej twarzy.

Wyniki przeprowadzonych testów algorytmu rozpoznawania twarzy opartego na dyskryminacyjnej analizie danych pokazują, że dla odpowiednio dobranego zbioru uczącego działanie algorytmu jest bardzo obiecujące. W przypadku rzeczywistej implementacji systemu kontroli dostępu opartego o analizowany algorytm należy jednak wziąć pod uwagę całą gamę czynników, które w niniejszym



Rys. 5.12: Zdjęcia wykorzystane do testu działania algorytmu: a) fotografia osoby, której zdjęcie znajduje się w zbiorze uczącym (trzecie zdjęcie w pierwszym wierszu, rys. 5.7), ale z innym wyrazem twarzy; b) fotografia spoza zbioru uczącego.

studium pominięto. Przyjęte założenie o skorzystaniu z gotowych fotografii było znacznym uproszczeniem. Zdjęcia zostały wykonane w warunkach laboratoryjnych z jednolitym tłem i przy stałym oświetleniu. Jakkolwiek system w przeprowadzonych testach pokazał poprawną identyfikację osoby na podstawie fotografii innej niż ta zapisana w bazie, ciągle była to fotografia o bardzo zbliżonych parametrach, bo wykonana w tych samych warunkach i krótkim odstępie czasowym. Dla pełnej oceny algorytmu należałoby przeprowadzić serię testów dla fotografii uzyskanych w różnych warunkach oświetleniowych, przy różnej kondycji fotografowanej osoby (stopień zmęczenia, czystość powłok skórnych, fryzura itp.). Należałoby również rozszerzyć zestaw uczący o taki zestaw fotografii dla każdej osoby.

5.6. Podsumowanie

Mimo 30 lat badań i rozwoju rozpoznawanie twarzy wciąż nie jest technologią, której można bezkrytycznie ufać i którą można masowo stosować. Niezawodne algorytmy automatycznej lokalizacji i rozpoznawania twarzy, działające w każdych warunkach, nie istnieją. Naukowcy są często nieobiektywni, a zaangażowanie instytucji komercyjnych, zainteresowanych osiągnięciem finansowych zysków, jeszcze pogarsza sytuację. Mimo wszystko postęp w dziedzinie rozpoznawania twarzy jest duży. Sukcesy odnotowano zwłaszcza przy zagadnieniach minimalizowania wpływu oświetlenia na wyniki rozpoznawania. Wydaje się też, że obiecujące perspektywy mają rozwiązania wykorzystujące specjalistyczny sprzęt, taki jak skanery trójwymiarowe.

Literatura

[1] M. Smiatacz and W. Malina: „Automatyczne rozpoznawanie twarzy – metody, problemy, zastosowania”., (2007).

[2] H. Moon and P. J. Phillips: „Computional and perfomence aspects of PCA-based face recognition algorithms”., (200).

[3] W. Zhao, R. Chellappa, and A. Krishnaswamy: „Discriminant Analysis of Principal Components for Face Recognition”. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG ’98, pp. 336–, Washington, DC, USA IEEE Computer Society, (1998).

[4] M. Turk and A. Pentland: „Eigenfaces for Recognition”. *Journal of Cognitive Neuroscience*, 3(1), (1991).

[5] A. Pentland, B. Moghaddam, and T. Starne: „View-based and Modular Eigenspaces for Face Recognition”. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, (1994).

[6] P. Phillips, H. Wechsler, J. Huang, and P. Rauss: „The FERET database and evaluation procedure for face recognition algorithms”. *Image and Vision Computing J*, 16(5):295–306, (1998).

[7] P. Phillips, H. Moon, S. Rizvi, and P. Rauss: „The FERET Evaluation Methodology for Face Recognition Algorithms”. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1090–1104, (2000).

ROZDZIAŁ 6

GESTY DŁONI

F. Melka, M. Żarkowski

6.1. Wstęp

Ludzkie dłonie są dobrym medium do przekazu prostych komend oraz nieskomplikowanych idei. Gesty dłoni, które są reprezentacją unikalnych kształtów oraz układów palców, mogę pełnić rolę interfejsu, który może zostać wykorzystany jako alternatywna metoda sterowania różnymi urządzeniami np. komputerami [1]. Wiąże się to z potrzebą komunikacji z otaczającymi nas maszynami w sposób bardziej zbliżony do naturalnego. Wzrost popularności rzeczywistości rozszerzonej (ang. *augmented reality*) [2] także sprzyja popularyzacji tego zagadnienia. Trend ten wyraźnie widać na rynku konsol do gier. Najwięksi producenci zaczynają produkować kontrolery do swoich urządzeń oparte na systemie kamer. Przykładami są *PlayStation Eye* firmy Sony [3] oraz *Kinect* Microsoftu [4]. Amatorsko powstają także aplikacje na komputery PC umożliwiające rozszerzenie funkcjonalności myszki o dodatkowe tzw. *gesty* [5] – wzorowane na systemie multitouch, zaimplementowanym w mobilnych urządzeniach firmy Apple.

Rozpoznawanie gestów dłoni jest także wykorzystywane do wspomagania procesów komunikacji osób z upośledzonym narządem mowy lub słuchu [6, 7, 8]. Translatory języka migowego na język pisany wypełniłyby lukę komunikacyjną, pomiędzy osobami głuchoniemymi, a w pełni sprawnymi. Ponadto programy służące do nauki języka migowego umożliwiłyby łatwiejszy dostęp do języka migowego osobom sprawnym, chcących się go nauczyć.

6.1.1. Gesty

Gestem nazywamy dowolny ruch wykonany świadomie lub nieświadomie, mający określone znaczenie. Stereotypowe myślenie nierozłącznie wiąże je z dłońmi i rękoma. W rzeczywistości może on zostać wykonany dowolną częścią ciała lub być wyrażony poprzez mimikę twarzy, postawę ciała etc.

Zazwyczaj gesty są rozumiane przez osoby należące do grupy, która przebywa w danym środowisku lub kręgu kulturowym (jak gesty pokazane na rys. 6.1). Niektóre z nich, dzięki mediom oraz prostocie przekazu, upowszechniły się na ca-

Rys. 6.1: Przykładowe gesty dłoni (źródło: `http://bi.gazeta.pl/im/1/5640/z5640551N.jpg`, `http://bi.gazeta.pl/im/6/5640/z5640626N.jpg`).

łym świecie. Przykładowo gest w postaci kciuka uniesionego do góry, w kulturze zachodniej ma kontekst pozytywny i oznacza wszystko w porządku. Natomiast w kulturze Bliskiego Wschodu ma on bardzo pejoratywny wydźwięk o kontekście seksualnym.

Gesty służą do komunikacji pozawerbalnej i niosą ze sobą informacje na temat podstawowych stanów emocjonalnych, intencjach, oczekiwaniach wobec rozmówcy, pozycji społecznej, pochodzeniu, wykształceniu, samoocenie, cechach temperamentu itp.

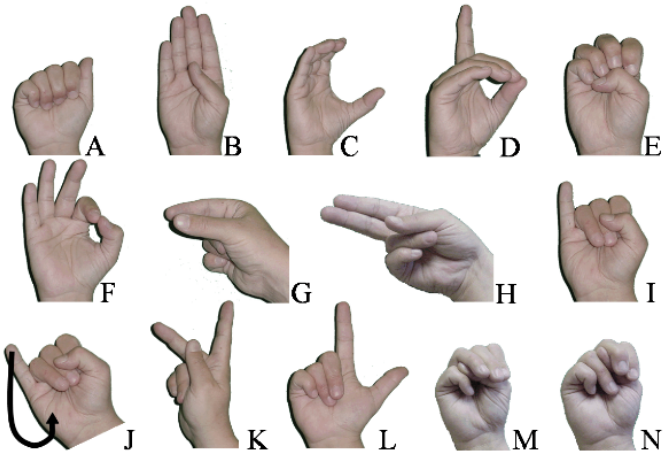
6.1.2. Język migowy

Czy język gestów można nazwać językiem migowym? Nie. Język gestów służy do wspomagania komunikacji werbalnej, natomiast język migowy służy do budowania wypowiedzi i komunikacji interpersonalnej. Docelową grupą, korzystającą z tego języka, są osoby z upośledzeniem narządu mowy oraz słuchu. Miganie umożliwia im uniknąć wykluczenia ze społeczeństwa związanego z brakiem możliwości komunikacji werbalnej.

W tym języku występują zarówno dynamiczne jak i statyczne gesty. W większości przypadków te pierwsze reprezentują całe słowa, natomiast drugie (z kilkoma wyjątkami) są wykorzystywane w jednoręcznym alfabecie palcowym służącym do migania pojedynczych liter (zobacz rys. 6.2).

6.2. Komputerowe wsparcie w nauce języka migowego

Stworzona aplikacja ma za zadanie wspomagać uczenie się gestów języka migowego. Jednym z podstawowych założeń jest rozpoznawanie statycznych układów dłoni. W związku z tym litery, których reprezentacja wymaga ruchu, są ignorowane. Pomijane są także niektóre statyczne gesty, które mogłyby wprowadzić element niepewności w klasyfikacji np. ze względu na zbliżony do innej litery układ dłoni. Domyślnie program działa prawidłowo dla dziewiętnastu znaków amerykańskiego alfabetu palcowego, jednak może on operować na dowolnej bazie gestów. Co więcej, można ją swobodnie edytować i podmieniać na własną.



Rys. 6.2: Przykłady liter w jednoręcznym alfabecie palcowym (źródło: `http://en.wikipedia.org/wiki/File:ABC_pict.png`).

W obecnej wersji program dysponuje trzema metodami nauczania gestów języka migowego. Pierwsza metoda *od człowieka do komputera – podstawowa* polega na tym, że użytkownik miga sekwencję liter, którą wyświetla komputer. Następnie na podstawie poprawności wykonania gestów oraz przeprowadzonych statystyk zostaje wyświetlona ogólna ocena. Aplikacja wyświetla podpowiedź w formie zdjęcia oczekiwanego gestu.

Druga metoda *od człowieka do komputera – zaawansowana* jest modyfikacją metody pierwszej. Użytkownik musi skorzystać ze swojej wiedzy, ponieważ aplikacja przestaje wyświetlać podpowiedzi.

Trzecia metoda *od komputera do człowieka* — komputer wyświetla zdjęcia różnych liter alfabetu palcowego, a użytkownik wprowadza ich odpowiedniki na klawiaturze. To podejście jest pomocne dla osób, które dopiero zaczynają naukę i nie znają jeszcze żadnych układów dłoni.

W każdej metodzie możliwa jest regulacja poziomu trudności poprzez zmianę czasu oczekiwania na zamiganie gestu. Pobranie gestu może odbywać się w ciągu całego czasu oczekiwania lub dopiero pod koniec — uniemożliwia to poprawę nieudanego gestu. Gesty mogą wyświetlać się w wyznaczonej kolejności lub losowo. Możliwa do modyfikacji jest także ilość testowanych gestów.

6.3. Oprogramowanie

Do wykonania aplikacji wykorzystano język *C#* wraz z biblioteką do przetwarzania obrazów *EmguCV*. Zaimplementowano w nim trzy metody segmentacji obrazu dłoni: statyczny detektor skóry, adaptacyjny detektor skóry oraz metodę wykorzystującą technikę odejmowania tła. Użytkownik może sam zdecydować, która z nich ma być wykorzystywana podczas przetwarzania obrazu. Klasyfikacja odbywa się za pomocą algorytmu *k najbliższych sąsiadów*.

6.3.1. Akwizycja obrazu

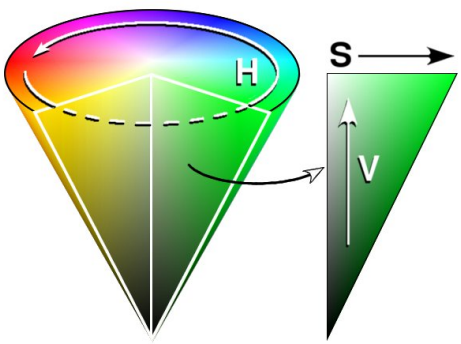
Do przechwytywania obrazu wykorzystano zwykłą kamerę internetową. Rozdzielczość nie jest parametrem krytycznym, więc każdy sprzęt takiego rodzaju powinien wystarczyć do prawidłowej pracy programu. Wskazane jest, aby firmware kamery umożliwiał regulację następujących parametrów: *ekspozycja*, *wzmocnienie*, *nasilenie kolorów*. Zalecane jest także wyłączenie wszelkich automatycznych funkcji poprawy obrazu, np. dynamicznej korekcji jasności, ponieważ może to zakłócić pracę detektora skóry.

6.3.2. Statyczny detektor skóry

Metoda ta wykorzystuje przestrzeń barw *HSV* (ang. *Hue Saturation Value*). Jest to model przestrzeni barw zaproponowany w 1978 roku przez Alveya Raya Smitha.

Model *HSV* nawiązuje do sposobu, w jakim widzi ludzki narząd wzroku, gdzie wszystkie barwy postrzegane są jako światło pochodzące z oświetlenia. Według tego modelu wszelkie barwy wywodzą się ze światła białego, gdzie część widma zostaje wchłonięta a część odbita od oświetlanych przedmiotów.

Nazwa modelu to skrót od pierwszych liter nazw składowych opisu barwy: H – barwa światła (ang. *Hue*) wyrażona kątem na kole barw przyjmująca wartości od 0° do 360°. Model jest rozpatrywany jako stożek, którego podstawą jest koło barw (rys. 6.3). Wymiary stożka opisuje składowa S – nasycenie koloru (ang. *Saturation*) jako promień podstawy oraz składowa V – (ang. *Value*) równoważna nazwie B – moc światła białego (ang. *Brightness*) jako wysokość stożka.



Rys. 6.3: Stożek przestrzeni barw HSV (źródło: `http://pl.wikipedia.org/w/index.php?title=Plik:HSV_cone.jpg&filetimestamp=20050904031115`).

Barwa skóry ludzkiej stanowi dość istotny czynnik. Powszechnie uważa się, że ludzie różnych ras mają odmienny *kolor skóry*. Jednak udowodniono, że różnica tkwi w intensywności, a nie w barwie. Poprzez przejście do odpowiedniej

przestrzeni barw w której te dwie składowe są podane wprost można zniwelować różnice pomiędzy różnymi odcieniami skóry.

Rozkład barwy skóry ludzkiej jest zależny od odpowiedniej reprezentacji, stąd też wybór właściwego modelu barw jest niezwykle istotny. Ostatecznie do reprezentacji kolorów wybrano paletę barw HSV, ponieważ umożliwia ona redukcję wpływu oświetlenia oraz koloru karnacji podczas rozpoznawania skóry.

Algorytm Zaimplementowany algorytm wykrywania skóry jest dość prosty. Wykrywanie odbywa się poprzez filtrację pikseli o określonej wartości składowych H, S i V. Ich wartości są dobierane przez użytkownika każdorazowo po uruchomieniu programu. Pomimo swej prostoty metoda nie jest dostatecznie odporna. Wystarczy, że na etapie ustalania progów zostanie uwzględniony jeden piksel, który nie odpowiada kolorowi skóry (artefakt na obrazie, zabrudzenie na dłoni) i podczas dalszego przetwarzania wszystkie piksele o takich samych parametrach, jak ten niechciany, zostaną sklasyfikowane jako poprawne i utrudnią lub uniemożliwią poprawną klasyfikację gestu.

6.3.3. Adaptacyjny detektor skóry

Nazwa modelu RGB powstała ze złożenia pierwszych liter nazw barw w nim uwzględnionych (rys. 6.4): R – czerwonej (ang. *Red*), G – zielonej (ang. *Green*) i B – niebieskiej (ang. *Blue*) . Jest to model wynikający z właściwości odbiorczych ludzkiego oka, w którym wrażenie widzenia dowolnej barwy można wywołać przez zmieszanie w ustalonych proporcjach trzech wiązek światła o barwie czerwonej, zielonej i niebieskiej.

Z połączenia barw RGB w dowolnych kombinacjach ilościowych można otrzymać szeroki zakres barw pochodnych. Do przestrzeni RGB ma zastosowanie synteza addytywna, w której wartości najniższe oznaczają barwę czarną, najwyższe zaś białą. Model RGB jest jednak modelem teoretycznym a jego odwzorowanie zależy od urządzenia (co po angielsku określa się jako *device dependent*). Oznacza to, że w każdym urządzeniu każda ze składowych RGB może posiadać nieco inną charakterystykę widmową, a co za tym idzie, każde z urządzeń może posiadać własny zakres barw możliwych do uzyskania.

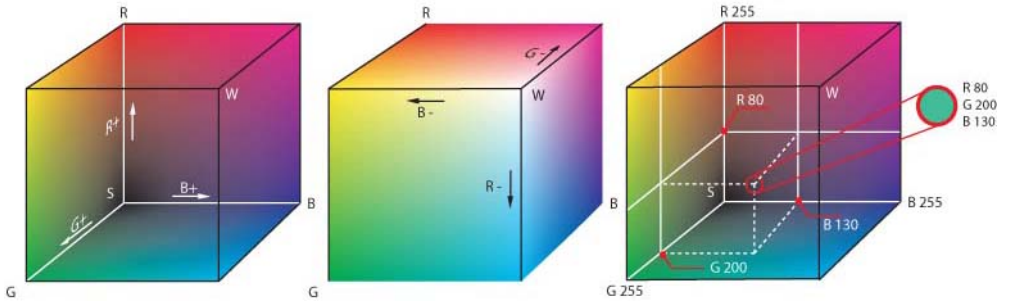
W adaptacyjnym detektorze skóry wykorzystano znormalizowaną paletę RGB, dzięki czemu uzyskano odporność na zmienne warunki oświetleniowe, a także wyeliminowano cienie na obrazie.

$$\text{podstawa} = R+G+B, \quad \text{r} = \frac{R}{podstawa}, \quad \text{g} = \frac{G}{podstawa}, \quad \text{b} = \frac{B}{podstawa}$$

Dzięki zabiegowi normalizacji zmniejsza się wymiar przestrzeni kolorów, ponieważ b jest redundantny. Można go zapisać w postaci

$$\text{b} = 1 - \text{r} - \text{g}$$

Algorytm Działanie adaptacyjnego detektora skóry, [9], zasadniczo jest podobne do działania jego statycznego poprzednika. Detektor adaptacyjny ma jednak większą odporność na błędy związane z danymi wejściowymi do określenia



Rys. 6.4: Sześcian przestrzeni barw RGB (źródło: `http://pl.wikipedia.org/w/index.php?title=Plik:RGB_farbwuerfel.jpg&filetimestamp=20041114064931`).

progów dla koloru skóry. Wiąże się to z charakterystyką jego działania. Nie traktuje on danych wejściowych *indywidualnie* tylko traktuje je jak pewien zbiór dla którego liczy wartość średnią oraz odchylenie standardowe. Tak otrzymane wartości są wykorzystane do obliczenia poszczególnych progów dla klasyfikatora:

$$\begin{aligned} lb_R &= 0.055 + 0.75(Rskin_{śr} - Rskin_\sigma) \\ ub_R &= -0.098 + 1.385(Rskin_{śr} + Rskin_\sigma) \\ lb_G &= -0.597 + 2.857(Gskin_{śr} - Gskin_\sigma) \\ ub_G &= -0.17 + 1.6(Gskin_{śr} + Gskin_\sigma) \\ lb_{podstawa} &= -45.26 + 0.79(Pskin_{śr} - Pskin_\sigma) \end{aligned} \tag{6.1}$$

gdzie,
$lb_R, lb_G, lb_podstawa$ – dolna granica przedziału (ang. *lowerbound*) dla koloru czerwonego, zielonego i podstawy (odpowiednio); ub_R, ub_G – górna granica przedziału (ang. *upperbound*) dla koloru czerwonego i zielonego (odpowiednio); $Rskin_{śr}, Gskin_{śr}, Pskin_{śr}$ – wartości średnie składowej czerwonej, zielonej i podstawy (odpowiednio) dla próbki danych pochodzących z obszaru testowego; $Rskin_\sigma, Gskin_\sigma, Bskin_\sigma$ – odchylenie standardowe składowej czerwonej, zielonej i podstawy (odpowiednio) dla próbki danych pochodzących z obszaru testowego.

6.3.4. Odejmowanie tła

W tej metodzie obrazy reprezentowane są w 256 stopniowej skali szarości. Każdy piksel zawiera tylko jedną informację na temat swojej jasności.

Algorytm Na początku użytkownik musi pobrać manualnie tło, które w kolejnych iteracjach algorytmu jest odejmowane od każdej następnej klatki. Program umożliwia odejmowanie pobranego obrazu od tła, bądź tła od obrazu, w zależności od tego czy tło jest jasne, czy ciemne. W efekcie otrzymywany jest obraz, na którym piksele o wartości różnej od 0 wskazują miejsca występowania różnic.

Kolejnym krokiem jest operacja progowania, dzięki której uzyskuje się binarną reprezentacje obiektów, które pojawiły się na danej klatce.

Następnie obraz uzyskany z tej operacji poddaje się działaniu filtru medianowego, o możliwej do ustalenia wielkości okna. Pozwala to na usunięcie drobnych zakłóceń i wygładzenie kształtu dłoni.

6.3.5. Wektor cech

Jako wektor cech często wykorzystuje się momenty geometryczne zwykłe oraz centralne. Momenty centralne są odpowiednikami momentów zwykłych, jednakże odniesione są do środka sylwetki, dzięki czemu zyskują one niezależność od położenia. Moment geometryczny zwykły rzędu $p+q$, opisujący sylwetkę U, definiuje się następująco:

$$m_{p,q} = \sum_{x=0}^{M-1} \sum_{y-0}^{N-1} \chi_U(x,y) x^p y^p \tag{6.2}$$

gdzie: M, N – rozmiar obrazu: szerokośc, wysokość, x, y – współrzędne punktu na obrazie, χ_u – funkcja charakterystyczna sylwetki U.

Podobną definicję posiada moment geometryczny centralny:

$$\eta_{p,q} = \sum_{x=0}^{M-1} \sum_{y-0}^{N-1} \chi_U(x,y)(x-x_s)^p (y-y_s)^q \tag{6.3}$$

gdzie: M, N – rozmiar obrazu: szerokośc, wysokość, x, y – współrzędne punktu na obrazie, x_s, y_s – współrzędne środka obrazu, χ_u – funkcja charakterystyczna sylwetki U.

W przypadku momentów centralnych można także zdefiniować momenty centralne znormalizowane, które są niezależne od położenia oraz skali badanego obiektu:

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma+1}}, \quad \text{gdzie} \quad \gamma = \frac{p+q}{2} \tag{6.4}$$

Istnieje także definicja momentów, które są niezależne od skali, przesunięcia oraz rotacji. Są to tzw. momenty Hu:

$$\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{6.5}$$

Rozważono dwie koncepcje kształtu wektora cech. Pierwsza z nich opiera się o momenty centralne znormalizowane, natomiast druga o momenty Hu. Dodatkowo wspólną cechą zawartą w obu wektorach jest współczynnik zwartości $\gamma = \frac{4\pi A}{P^2}$, gdzie A – pole powierzchni obszaru, P – obwód tego obszaru.

Rozważając oba powyższe przypadki zdecydowano się na domyślne wykorzystanie momentów normalnych scentralizowanych. Postawiono hipotezę, że ich brak odporności na rotację umożliwi lepsze rozróżnienie próbek. Owo założenie okazało się słuszne.

6.3.6. Klasyfikacja

Do klasyfikacji gestów wykorzystano algorytm *k najbliższych sąsiadów*. Jest to jeden z algorytmów regresji nieparametrycznej używanych w statystyce do prognozowania wartości pewnej zmiennej losowej. Może również być używany do klasyfikacji. Przyjęto też następujące założenia:

- Dany jest zbiór uczący zawierający obserwacje z których każda ma przypisany wektor zmiennych objaśniających $X_1 \dots X_n$ oraz wartość zmiennej objaśnianej Y.
- Dana jest obserwacja C z przypisanym wektorem zmiennych objaśniających $X_1 \dots X_n$ dla której chcemy prognozować wartość zmiennej objaśnianej Y.

W przypadku rozwiązywanego problemu, wektor X odnosi się to wektora cech uzyskanego przy pomocy ekstrakcji, a wektor Y reprezentuje klasy gestów (litery), które występują w programie. Algorytm polega na:

1. porównaniu wartości zmiennych objaśniających dla obserwacji C z wartościami tych zmiennych dla każdej obserwacji w zbiorze uczącym,
2. wyborze k (ustalona z góry liczba) najbliższych do C obserwacji ze zbioru uczącego,
3. uśrednieniu wartości zmiennej objaśnianej dla wybranych obserwacji, w wyniku czego uzyskujemy prognozę.

Definicja *najbliższych obserwacji* w punkcie drugim sprowadza się do minimalizacji pewnej metryki, mierzącej odległość pomiędzy wektorami zmiennych objaśniających dwóch obserwacji. Zwykle stosowana jest tu metryka euklidesowa lub metryka Mahalanobisa. Można również zamiast średniej arytmetycznej stosować np. medianę.

Klasyfikator wymaga wcześniejszego uczenia. Do tego celu została wykorzystana biblioteka gestów, zawierająca ok. 100 zdjęć dla każdego ułożenia dłoni. Uczenie jest przeprowadzane przed każdym wykorzystaniem aplikacji, co pozwala na łatwą zamianę bazy gestów lub wykorzystanie innego współczynnika k. Program po otrzymaniu serii uczącej analizuje te obrazy, dokonuje ekstrakcji cech zgodnie z ustawieniami zadanymi przez użytkownika, a następnie na ich podstawie uczy klasyfikator.

Wartą wspomnienia zaletą klasyfikatora *k najbliższych sąsiadów* jest jego odporność na rozrzucone punkty należące do poszczególnej klasy. Punkty nie muszą znajdować się w jednym skupisku, jak w przypadku innych klasyfikatorów.

Różne kształty dłoni i różne ułożenia światła mogą powodować wystąpienia wielu oddalonych od siebie grup punktów, każda symbolizująca tą samą klasę. Zakładając, że te skupiska mają liczebność przynajmniej k, pozwalają one na poprawną klasyfikację gestu nawet w przypadku, jeśli te grupy są rozłączne.

6.3.7. Interfejs

Wygląd okna stworzonej aplikacji został zaprezentowany na rys. 6.5 oraz rys. 6.6. Pokazano na nich zawartość dwóch zakładek: pierwsza dotyczy opcji segmentacji, natomiast druga - samego treningu i jego ustawień. Nazwy wszystkich elementów są intuicyjne, więc interfejs zostanie opisany zgrubnie.

W oknie można wyróżnić jedną wspólną część (nr 1 na obu rysunkach). Zawiera ona przyciski umożliwiające włączenie przechwytywania obrazów, trening klasyfikatora oraz tworzenie nowych zestawów danych uczących (patrz rozdział 6.5). Poniżej jest umiejscowiona konsola zwracająca parametry aktualnie zmieniające się w systemie. Widoczny jest także podgląd obrazu z kamery oraz powiększonego obszaru zainteresowania (czerwona ramka na podglądzie). W tym miejscu można także wybrać metodę segmentacji obrazu.

Na rysunku 6.5 jest pokazana zakładka z ustawieniami segmentacji. Obszar nr 2 zawiera podgląd klasyfikatora HSV, poszczególnych wartości barwy, nasycenia i jasności. Przycisk *Catch* umożliwia pobranie próbki skóry, na podstawie której będzie się odbywała klasyfikacja. Jest ona pobierana z obszaru w niebieskiej ramce (jest ona umieszczona na podglądzie obrazu z kamery).

W obszarze nr 3 jest zawarty podgląd adaptacyjnego detektora skóry. Przycisk *Catch* pełni taką samą funkcję jak w poprzednim klasyfikatorze.

Ostatni, czwarty obszar z tej zakładki zawiera informacje na temat segmentacji z wykorzystaniem metody odejmowania tła. Kolejne okna podglądu pokazują różnice na obrazach, obraz wejściowy po binaryzacji oraz tło, które jest odejmowane. Przycisk *Catch* umożliwia przechwycenie tła. Dodatkowo można wybrać kierunek odejmowania obrazów.

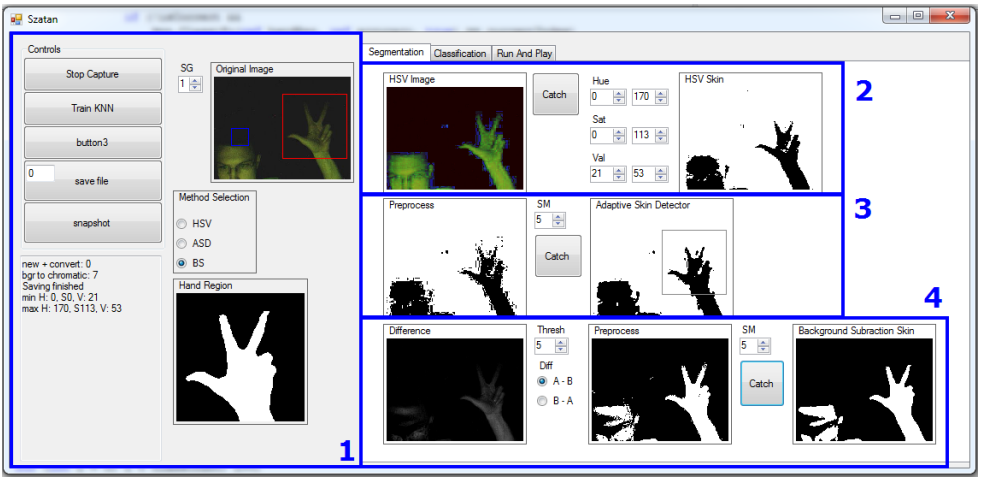
Na rysunku 6.6 pokazano, że na obszarze nr 5 dostępne są opcje związane z treningiem użytkownika. Może on wybrać rodzaj ćwiczenia, czas oczekiwania na gest, ilość gestów w sekwencji. Można także włączyć pokazywanie rezultatów klasyfikacji, wybrać rodzaj przechwycenia gestu – czy ma być pobrany na pod koniec czasu przeznaczonego na pokazanie gestu, czy szybciej. Ostatnia opcja umożliwia wybór kolejności pojawiających się liter pomiędzy rosnącą, a losową.

Obszar nr 6 zawiera podgląd litery do pokazania, zdjęcia gestu, a także jego zbinaryzowaną wersję.

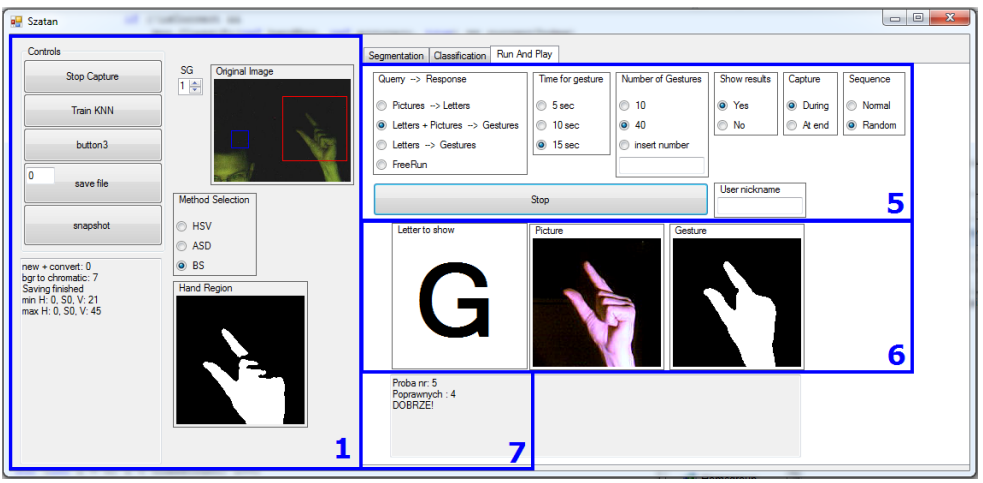
Ostatni obszar nr 7 to prosta konsola tekstowa zwracająca użytkownikowi informację na temat klasyfikacji zamiganego gestu oraz statystykę skuteczności.

6.4. Badania i odbiór aplikacji

Aplikacja wywarła bardzo dobre wrażenie na osobach testujących. Aby sprawdzić jak szybko przebiega proces uczenia przeprowadzono dwa eksperymenty na różnych osobach, nie mających wcześniej kontaktu ani z językiem migowym ani



Rys. 6.5: Interfejs programu – zakładka *Segmentation*.



Rys. 6.6: Interfejs programu – zakładka *Run And Play*.

alfabetem palcowym. Wykorzystano trzy metody: 1) Litera + Obrazek → Gest; 2) Litera → Gest; 3) Obrazek → Litera.

Proces uczenia przebiegał następująco:

- konfiguracja aplikacji przez osobę z nią zaznajomioną,
- wyjaśnienie celu ćwiczenia odbiorcy,
- uruchomienie programu dla 20 losowych gestów i metody nr 1 — sesja treningowa,
- podczas interakcji z aplikacją osoba znająca aplikację pomagała osobie badanej w poprawnym ułożeniu dłoni,
- uruchomienie dla 20 losowych gestów i metody nr 3 — test wprowadzający,

- ponowne uruchomienie metody nr 1, tym razem z gestami w kolejności alfabetycznej i szczególnym zwróceniu uwagi na gesty sprawiające trudność,
- powtórzenie nauki liter opartej na metodzie nr 3 — test właściwy,
- wyłączenie podpowiedzi — metoda nr 2 — test zaawansowany.

Poniżej przedstawione są logi z nauczania, wskazana jest liczba poprawnych trafień i liczba prób dla każdej litery oraz wynik sumaryczny.

```
2011-01-20 10:59:04 PicturesToGestures 20 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
02 01 01 01 03 01 00 01 00 02 00 00 00 01 03 00 00 01 03
02 01 01 01 03 01 00 01 00 02 00 00 00 01 03 00 00 01 03

2011-01-20 11:03:58 PicturesToLetters 14 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
00 04 02 02 00 00 00 00 00 01 01 00 01 00 00 01 01 00 01
00 04 02 02 01 00 02 00 00 01 01 00 01 00 00 01 01 03 01

2011-01-20 11:10:02 PicturesToGestures 19 19
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

2011-01-20 11:11:43 PicturesToLetters 20 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
01 00 00 02 01 01 02 01 01 01 00 03 00 00 02 00 00 01 04
01 00 00 02 01 01 02 01 01 01 00 03 00 00 02 00 00 01 04

2011-01-20 11:13:23 LettersToGestures 19 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
00 04 02 01 01 00 00 01 01 00 01 00 03 00 02 01 00 00 02
00 04 02 02 01 00 00 01 01 00 01 00 03 00 02 01 00 00 02
```

Pierwsza badana osoba po 15 minutach nauki była w stanie przejść test na poziomie zaawansowanym z 95% skutecznością.

```
2011-01-20 11:01:23 PicturesToGestures 19 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
01 01 01 00 00 02 02 01 00 00 00 00 05 02 01 01 01 00 01
01 01 01 01 00 02 02 01 00 00 00 00 05 02 01 01 01 00 01

2011-01-20 11:08:07 PicturesToLetters 07 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
01 00 00 00 00 00 00 00 00 00 03 00 00 00 00 01 00 00 02
01 02 00 01 02 00 01 00 00 00 03 02 03 02 00 01 00 00 02
```

```
2011-01-20 11:15:13 PicturesToGestures 19 19
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

2011-01-20 11:18:02 PicturesToLetters 13 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
00 00 04 00 01 00 02 00 01 01 00 01 00 00 00 01 01 00 01
00 00 04 00 02 00 03 00 01 01 00 02 02 00 02 01 01 00 01

2011-01-20 11:20:13 LettersToGestures 18 20
 A  B  C  D  F  G  H  I  K  L  O  P  Q  R  U  V  W  X  Y
02 01 01 01 00 01 00 01 00 03 02 02 01 01 01 00 00 00 01
02 01 01 01 01 01 01 01 00 03 02 02 01 01 01 00 00 00 01
```

Druga osoba miała początkowe problemy ze skojarzeniem gestu z konkretną literą, mimo to nauka przebiegała równie szybko. Po 20 minutach test zaawansowany zaliczyła z 90% skutecznością. Zauważono problemy w kojarzeniu obrazu gestu z literą, pomimo, że ćwiczenie w stronę litera – gest, przebiegło bardzo dobrze.

Autorzy aplikacji są pozytywnie zaskoczeni szybkością i skutecznością nauczania aplikacji, nie spodziewano się tak dobrych wyników w tak krótkim czasie.

6.5. Definiowanie własnych gestów

W prosty sposób można nauczyć klasyfikator rozpoznawania nowych gestów. W tym celu należy podmienić predefiniowane gesty na inne. Algorytm postępowania jest następujący: w polu tekstowym na przycisku *Save file* trzeba wpisać numer klasy, jaki będzie reprezentowany przez zarejestrowane próbki, następnie umieścić dłoń w regionie zainteresowania (czerwony prostokąt na obrazie z kamery) i powyższym przyciskiem uruchomić akwizycję obrazu. Program przechwyci serię obrazów, które zostanę wykorzystane do nauki klasyfikatora. Po ukończeniu tego zadania należy wykonać zdjęcie nowego gestu, które będzie wyświetlane podczas nauki jako wzór. W tym celu trzeba kliknąć przycisk *Snapshot* (z ręką w rejonie zainteresowania). Następnie należy udać się do folderu *NewTraining* znajdującego się w folderze z aplikacją. Kolejnym krokiem jest odszukanie pliku `base.txt` i zmieniana w bazie zapytań litery skojarzonej z danym gestem na nową. Po ponownym uruchomieniu aplikacji gest będzie już dostępny do nauki.

6.6. Możliwe kierunki rozwoju

Wygląd skóry na różnych obrazach może być zupełnie różny. Jednym z najbardziej znaczących czynników wpływających na to są warunki oświetlenia sceny. Aby zmniejszyć ilość błędów identyfikacji ze względu na powyższy czyn-

nik, są prowadzone badania nad algorytmami umożliwiającymi wyeliminowanie wpływu światła lub jego znormalizowanie. Jedno z efektywnych rozwiązań jest opisane w [10]. Jego implementacja poprawiłaby jakość działania aplikacji.

Aplikacja jest otwarta na wszelkie zmiany bazy gestów. Jest przygotowana do implementacji jednoczesnej obsługi różnych baz. Oprogramowanie może stanowić bibliotekę rozpoznającą gesty, udostępniającą swoje funkcje innym aplikacjom. Skojarzenie gestów z akcjami klawiatury, czy ruchami i przyciskami myszki może stanowić nietypowe i alternatywne sterowanie komputerem.

Literatura

[1] Y. Y. Pang, N. A. Ismail, and P. L. S. Gilbert: A Real Time Vision-Based Hand Gesture Interaction. In *Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on*, pp. 237 –242, (2010).

[2] M. Billinghurst, H. Kato, and S. Myojin: R. ShumakerAdvanced Interaction Techniques for Augmented Reality Applications. In :, editor, *Virtual and Mixed Reality*, volume 5622 of *Lecture Notes in Computer Science*, pp. 13–22 Springer Berlin / Heidelberg, (2009).

[3] Sony: PlayStation Eye. `http://pl.playstation.com/ps3/peripherals/detail/item114974/PlayStationEye/`, (2010).

[4] Microsoft: Kinect. `http://www.xbox.com/en-US/kinect`, (2010).

[5] A. Dąbrowski: Czy gesty zastąpią myszkę?. `http://wiedza.hoga.pl/Wiadomosc.aspx?id=444`, (2010).

[6] H. Tauseef, M. Fahiem, and S. Farhan: Recognition and Translation of Hand Gestures to Urdu Alphabets Using a Geometrical Classification. In *Visualisation, 2009. VIZ '09. Second International Conference in*, pp. 213 –217, (2009).

[7] J. Marnik: Rozpoznawanie znaków polskiego alfabetu palcowego., (2003).

[8] S. Myślinski: *Rozpoznawanie obrazów dłoni za pomocą gramatyk klasy ETPL(k) w systemach wizyjnych analizy języka migowego.* PhD thesis, Akademio Górniczo-Hutnicza w Krakowie, (2009).

[9] M. Wimmer and B. Radig: Adaptive Skin Color Classificator., (2005).

[10] Y. Tu, F. Yi, G. Chen, S. Jiang, and Z. Huang: Skin color detection by illumination estimation and normalization in shadow regions. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pp. 1082 –1085, (2010).

ROZDZIAŁ 7

GESTY MARSZAŁKA

K. Cisek, M. Szlosek

7.1. Wstęp

Jednym ze sposobów komunikacji międzyludzkiej jest posługiwanie się gestami - lokalnie jednoznacznymi wizyjnymi komunikatami niewerbalnymi. Gesty pozwalają na wymianę informacji tam, gdzie komunikacja werbalna zawodzi lub jest niepożądana ze względu na fakt rozchodzenia się fali dźwiękowej albo innych powodów. Prawdopodobnie największą grupę społeczną stosującą komunikację niewerbalną na co dzień stanowią osoby, które utraciły możliwość posługiwania się aparatem mowy. Języki oparte na gestach, używane przez osoby niepełnosprawne, nazywane są migowymi oraz miganymi. Migowy jest rodzajem gestów pokazującym całe słowa albo krótkie wyrażenia, a migany jest systemem pokazywania pojedynczych liter bądź zgłosek. Co ciekawe, i nielogiczne, języki migowe i migane nie są językami międzynarodowymi. Specyficzną grupą użytkowników gestów są ludzie o profesjach, w których potrzebna jest szybka, skuteczna, jednoznaczna oraz niewymagająca specjalistycznego sprzętu komunikacja. Przykładowymi grupami zawodowymi posługującymi się gestami są: specjaliści sterujący programami/maszynami/systemami, żołnierze, płetwonurkowie, pracownicy wysokościowi, pracownicy kolei, sędziowie sportowi, organizatorzy oraz uczestnicy zawodów sportowych, żeglarze, marszałkowie.

Marszałek (od ang. *Marshal's Hand Signals* lub *Aircraft Marshalling*) jest członkiem personelu lotniczego odpowiedzialnym za kierowanie ruchem po płycie lotniska/lotniskowca, który nawiązuje bezpośredni kontakt wzrokowy z pilotem statku powietrznego. Marszałek może również dawać wskazówki pilotowi do lądowania oraz startu w przypadku, gdy statek powietrzny jest przystosowany do pionowego startu lub/i lądowania (śmigłowce, wiatrakowce, niektóre samoloty). Osoba taka pracuje najczęściej na cywilnych lotniskach kontrolowanych oraz na wojskowych lotniskach, lądowiskach oraz lotniskowcach.

W niniejszym rozdziale opisano sposób implementacji systemu rozpoznawania gestów marszałka. Pomysł podjęcia takiego tematu ma ciekawą genezę. Je-

den z autorów [1] podczas podstawowego szkolenia samolotowego na lotnisku Poznań Ławica zetknął się z koniecznością rozpoznawania takich gestów. Stało się to inspiracją do opracowania założeń narzędzia programowego wspomagającego proces szkolenia. W założeniach system ten miał rozpoznawać sfilmowane gesty i porównywać je ze wzorcem. Finalne rozwiązanie, opisane w dalszej części rozdziału, zrealizowano korzystając z algorytmu SVN. Za wyborem tego algorytmu przemawiały jego właściwości oraz natura badanego problemu. Okazało się, że odczyty sensoryczne można przedstawić w przestrzeni stuwymiarowej, w której SVN radzi sobie bardzo dobrze. Ponadto zadowalające wyniki z SVN można osiągnąć w stosunkowo szybkim czasie, przy dość prostej implementacji i sposobie uczenia.

7.2. Podstawy teoretyczne

Mimo upływu lat klawiatura i myszka pozostają wciąż podstawowymi urządzeniami wejściowymi w komunikacji człowiek-komputer. Nie są to jednak jedyne urządzenie służące do przekazywania poleceń/danych. Wraz z rozwojem technologii pojawiły się kolejne rozwiązania, jak np.: ekrany dotykowe, kamery, czujniki przyśpieszenia/prędkości/położenia. Obecnie można sobie wyobrazić rozwiązanie, w którym człowiek za pomocą samych tylko gestów byłby w stanie sterować maszyną, np. bezzałogową koparką albo quadrocopterem. Wystarczy, aby maszyna była wyposażona w kamery rejestrujące gesty oraz komputer przetwarzający obrazy z kamer i wyznaczający sterowania. Wiąże się to oczywiście z budową systemu wizyjnego, implementacją algorytmów do przetwarzania obrazów itp.

W prostym przypadku rozpoznawanie gestów może polegać na śledzeniu położenia znaczników na sekwencji obrazów przechwyconych przez kamery oraz interpretacji zaobserwowanych przemieszczeń. Znacznikami mogą być: kolorowe plansze, odbłyśniki, źródła światła widzialnego, źródła światła podczerwonego i inne elementy fizycznie związane z obserwowanym obiektem.

Jednym z urządzeń, które na swoim pokładzie gromadzi kilka różnych czujników, jest *Wii Remote* firmy Nintendo. W urządzeniu tym zamontowano kamerę podczerwieni, czyjnik ruchu, przyciski. Choć zaprojektowano je do celów rozrywkowych, jednak z powodzeniem można je wykorzystać do budowy systemu rozpoznawania gestów.

7.2.1. Formalne definicje oraz standardy opisujące gesty marszałka

Istnieje około 68 [1] gestów marszałka, przy czym mogą to być gesty statyczne (ustalona pozycja) oraz dynamiczne (sekwencja ruchów). Gesty są standardowe dla wielu organizacji zrzeszających różne typy lotnictwa:

- North Atlantic Treaty Organization (NATO)
- Air Standardization Coordinating Committee
- International Civil Aviation Organization (ICAO) [2, str. 26]

[1] Krzysztof Cisek, instruktor pilot szybowcowy

- Federal Aviation Administration (FAA)

W ciągu dnia gesty najczęściej wykonywane są za pomocą rąk. W nocy lub przy ograniczonej widzialności taka praktyka jest nieskuteczna, w związku z czym stosuje się specjalne podłużne lampy informacyjne.

Marszałek podczas wydawania poleceń za pomocą gestów powinien stać twarzą do statku powietrznego w jednej z dwóch pozycji:

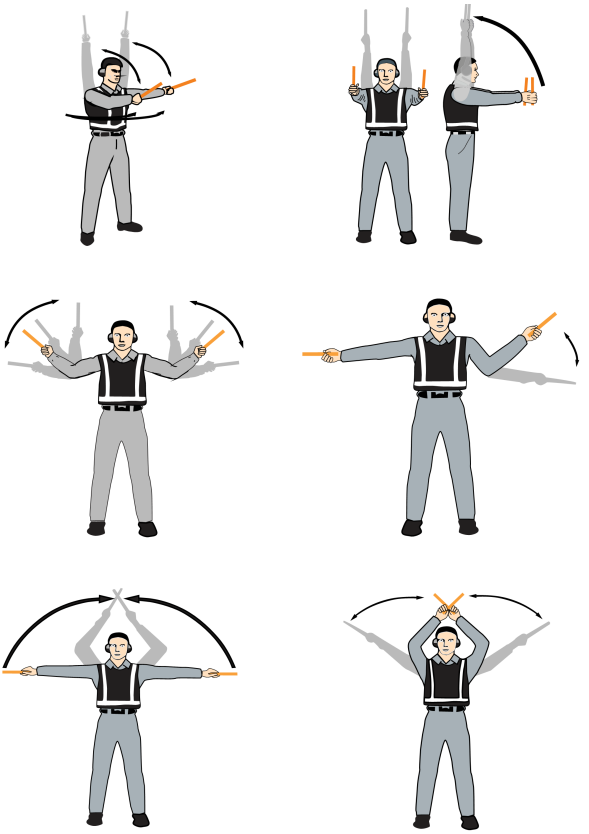
- Dla statków powietrznych o stałym płacie (np. samoloty) marszałek powinien znajdować się z przodu po lewej stronie patrząc od strony pilota maszyny, tak aby te dwie osoby się widziały wzajemnie.
- Dla śmigłowców, marszałek powinien się znajdować z przodu maszyny patrząc od strony pilota.

Marszałek posługuje się w nocy jednokolorowymi podłużnymi lampami. W momencie kołowanie w przypadku awarii jednej lub obu takich lamp, pilot statku powietrznego ma obowiązek zatrzymania się. Przykładowe gesty przedstawiono na rys. 7.1 i rys. 7.2.

7.2.2. Metody rozpoznawanie gestów

Rozpoznawanie gestów jest znanym tematem badań naukowych. Poniżej przedstawiono wybrane prace z tego obszaru, zawierające pomysły wykorzystania automatycznej segmentacji i generowania modeli potrzebnych do przeprowadzenia poprawnej klasyfikacji gestów.

W [7] zaprezentowano sposób automatycznej segmentacji i klasyfikacji ciągłych obserwacji gestów. Autorzy sprowadzili problem do tego, iż podobnie jak w mowie, ruch dłoni podzielili na „fonemy". Przestrzeń gestów była zbiorem gaussianów, gdzie każdy ruch był połączony z odpowiednim gestem. Rozłożenie gaussianów zdeterminowane było zastosowaniem standardowego algorytmu maksymalizacji wartości oczekiwanej (ang. *Expectation-Maximization*, EM), a ich ilość wyznaczana była za pomocą algorytmu minimalnej długości opisu (ang. *Minimum Description Length*, MDL). Autorzy [4] opisali metodę rozpoznawanie gestów amerykańskiego języka migowego (ang. *American Sign Language*, ASL). Rozłożyli ASL na sekwencje ruchów i stałych pozycji dłoni, które traktowali jako pojedyncze „fonemy". Ucząc system oparty na ukrytych modelach Markova (ang. *Hidden Markov Models*, HMM) starali się uzyskiwać pojedyncze fonemy zamiast całych sekwencji ruchu. W [5] opisano sposób akwizycji statystycznych modeli dla strukturalnie i semantycznie bogatych zachowań. Działania były zamodelowane jako sekwencje cząstkowych komponentów z wykorzystaniem modeli Markova o zmiennych długościach kontrolujących wysokopoziomową strukturę. Cząstkowe zachowania były widziane jako prototypowe sekwencje pomiędzy dwoma kluczowymi prototypami, które z kolei były identyfikowane jako prototypy z sekwencji, gdzie zmiany występowały poniżej wartości progowej. Metoda ta może być wykorzystana do generowania i predykcji realistycznych zachowań ludzkich, lecz nie mogły generalizować przypadków wcześniej nieznanych. W [6] zaprezentowano metodę automatycznej akwizycji danych na podsta-

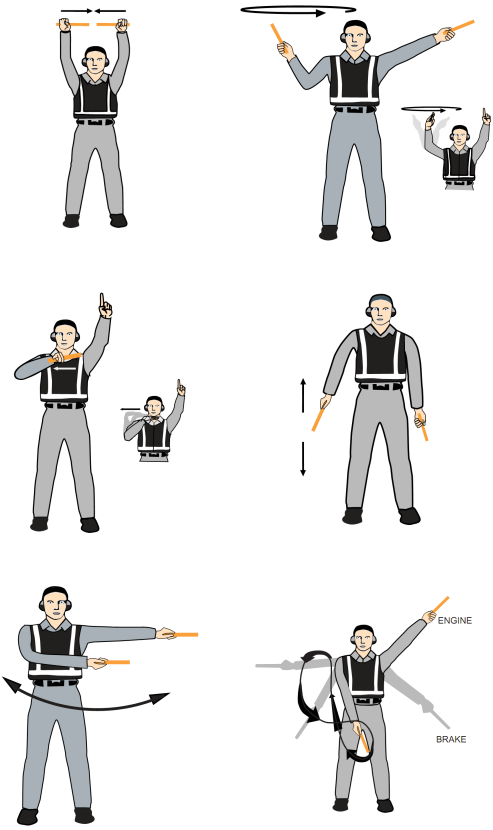


Rys. 7.1: Znaczenie gestów:*„Kołuj do następnego marszałka/ przejdź na kontrolę przez wieżę/przejdź na kontrolę przez służbę naziemną”, „Identyfikacja bramy” - marszałek pokazuje którą drogą kołowania należy się przemieszczać, „Kołuj do przodu”, „Kołuj w lewo”, „Normalne hamowanie”, „Hamowanie awaryjne”* [3, roz. 6].

wie statystycznego zachowania modelu z ciągłych obserwacji długich sekwencji obrazów. Metodę oceny i klasyfikacji zachowania oparto o statystyczny opis natury owych modeli. Autor innego podejścia [8] zaproponował wykorzystanie sieci neuronowych. Testując trzy typy sieci uzyskał on zadowalające wyniki, przy czym najlepsze dla sieci ze sprzężeniami zwrotnymi. Jej odpowiednio dobrana architektura zapewniała uzyskanie 5% poziomu błędnej klasyfikacji.

7.2.3. Klasyfikacja z wykorzystaniem sztucznych sieci neuronowych

W dziedzinie rozpoznawania przestrzennych wzorców za pomocą sztucznych sieci neuronowych (ANN) dokonała się znaczna poprawa jakości otrzymywanych wyników. W zwykłych przestrzennych ANN wzorcem zazwyczaj jest dwuwymiarowa tablica danych (tablica $2D$ zamieniana jest na wektor jednowymiarowy). Tego typu sieci dobrze sprawdziły się w rozpoznawaniu pisma ręcznego, twarzy



Rys. 7.2: Znaczenie gestów: *„Podstawki wsunięte”*, *„Włączaj wybrane silniki”*, *„Wyłącz silniki”*, *„Zwalniaj silnikiem z wskazywanej strony”*, *„Sygnał dla śmigłowca - przemieszczaj się w prawo w płaszczyźnie horyzontalnej”*, *„Pożar silnika/hamulców”*[3, roz.6].

i innych, dwuwymiarowych obiektów. Przestrzenne sieci cechuje to, że niezależnie od ilości wewnętrznych połączeń, zawsze istnieje jedna tablica o zadanym wymiarze, definiująca wejście.

Do wykrywania obiektów zmiennych w czasie potrzeba jednak sieci wrażliwych na dynamicznie zmieniające się sygnały wejściowe. Autor [8] zaproponował trzy modele ANN, które choć bazują na statycznym modelu, działają na czasowo zmiennych sygnałach wejściowych. Najlepszym uzyskanym przez niego modelem sieci, przewidującym następny krok w sekwencji, okazał się model z jedną warstwą podłączoną rekurencyjnie ze sobą. Architektura ta powoduje „stabilizowanie” się warstwy na jakimś poziomie i swoiste zapamiętywania stanów, dzięki czemu sieć jest bardziej wrażliwa na zmiany w neuronach wejściowych.

Klasyfikowanie gestów można zrealizować na podobnej zasadzie. Zadanie polega wtedy na porównywaniu wyjścia z pojedynczej sieci z następnym napływającym wektorem. Licząc błędy pomiędzy tymi wektorami i je sumując otrzymuje

się jakość klasyfikacji danego gestu. Liczba wyuczonych sieci byłaby równa liczbie gestów, które należy klasyfikować. W problemie zdefiniowanym przez autorów niniejszej pracy pozyskiwane są dane o położeniu 4 punktów na płaszczyźnie $2D$ wraz ze stemplem czasu. Dlatego, idąc za powyższą propozycją, architektura sztucznej sieci neuronowej mogłaby wyglądać następująco: $9\times9\times4\times8$ (kolejne liczby to, odpowiednio, ilość neuronów warstwy wejściowej, warstwy rekurencyjnej, warstwy ukrytej oraz warstwy wyjściowej). Jednak zastosowanie takiego rozwiązania sprawia trudności w realizacji algorytmu wstecznej propagacji błędów, odpowiedzialnego za dobre nauczenie sieci. Mozer [9] nazwał warstwę rekurencyjną warstwą kontekstową (ang. *context layer*) i zaproponował zmodyfikowany algorytm uaktualniania wag w warstwie kontekstowej. Stwierdził on, że gdy inne rekurencyjne metody przetrzymują pojedynczy stan poprzedni, warstwy kontekstowe przechowują dynamiczną reprezentację przeszłości, skutecznie skupiając propagację błędów z „przeszłości" przechowywanej w poprzednich stanach.

Alternatywnym sposobem radzenia sobie z predykcją wejścia oraz klasyfikacją gestów jest zastosowanie czasowo-opóźnionej sieci neuronowej (ang. *Time Delay Neural Network*, TDNN). TDNN, poprzez przesunięcie w czasie ramki zawierającej wektor stanu, stara się obliczyć następny przewidywany wektor. Poglądowy schemat działania TDNN pokazano na rys. 7.3.

W przypadku wystąpienia problemów z działaniem sieci neuronowych, dane napływające z sensorów należy przekształcić do innej postaci. Metodami przekształcającymi wektor danych wejściowych są m.in. redukcja ilości zmiennych, zmniejszenie wymiarowości wektorów, klasteryzacja poszczególnych punktów w przestrzeni $2D$.

7.2.4. Klasyfikacja z wykorzystaniem SVM

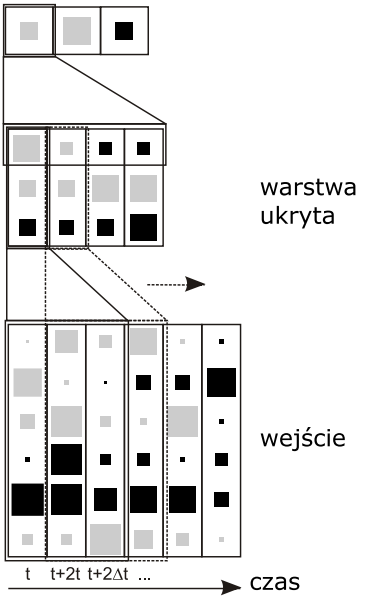
W ostatnich latach dość znacząco rozwinęły się różne algorytmy i metody klasyfikacji, które nadają się do wykorzystania podczas klasyfikacji gestów marszałka. Stosunkowo prosty w implementacji jest algorytm SVM (ang. *Support Vector Machines*) polegający na klasteryzowaniu danych uczących i oddzielaniu ich od siebie hiperpłaszczyznami, separującymi klasy z jak największym marginesem.

Podstawy formalne

W najprostszym przypadku klasyfikacja algorytmem SVM polega na znalezieniu równania hiperpłaszczyzny rozdzielającej punkty z różnych klas (należące do zbioru treningowego) z maksymalnym marginesem oraz użyciu tego równania jako funkcji decyzyjnej przy klasyfikacji kolejnych punktów (spoza zbioru treningowego). Ten prosty, opisany niżej przypadek (tylko dwie klasy) można rozszerzyć na większą ilość klas (zobacz rys. 7.4).

Formalnie zbiór danych treningowych D zawierający n wektorów należących do dwóch różnych klas można zapisać w postaci:

$$D = \left\{\left(x_i, y_i\right) | x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\right\}_{i=1}^{n},$$



Rys. 7.3: Schemat działania czasowo-opóźnionej sieci neuronowej (wg `http://en.wikipedia.org/wiki/File:DiagramTDNN_english.png`).

gdzie: x_i jest p-wymiarowym wektorem, zaś y_i określa klasę, do której ten wektor należy.

Każda hiperpłaszczyzna może być zapisana jako zbiór punktów x spełniających równanie $\mathbf{w} \cdot x - b = 0$, gdzie $\cdot$ oznacza iloczyn skalarny, zaś wektor $\mathbf{w}$ to wektorem normalnym do hiperpłaszczyny. Parametr $\frac{b}{\|\mathbf{w}\|}$ określa przesunięcie marginesu hiperpłaszczyzny wzdłuż wektora normalnego $\mathbf{w}$.

W algorytmie SVN chodzi o znalezienie takich $\mathbf{w}$ i b, aby maksymalizować margines lub odległość pomiędzy równoległymi hiperpłaszczyznami (czyli płaszczyzn, które są najdalej od siebie, a jednocześnie oddzielają dane). Te hiperpłaszczyzny można opisać za pomocą równań:

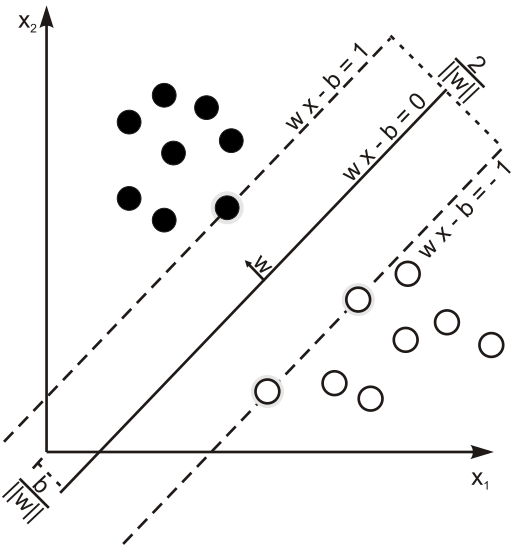
$$\mathbf{w} \cdot x - b = 1, \quad \mathbf{w} \cdot x - b = -1$$

Należy zauważyć, że jeśli dane treningowe są separowalne, to można tak wybrać równoległe hiperpłaszczyzny, aby nie było punktów pomiędzy nimi, a następnie starać się maksymalizować ich odległości. Odległość pomiędzy dwoma hiperpłaszczyznami wynosi $\frac{2}{\|\mathbf{w}\|}$, więc należy maksymalizować $\|\mathbf{w}\|$. Równocześnie należy zapobiec „wpadaniu" punktów do marginesu, dodając następujące ograniczenia

$$\mathbf{w} \cdot x_i - b \geq 1 \text{ dla } x_i \text{ z pierwszej klasy}, \quad \mathbf{w} \cdot x_i - b \leq -1 \text{ dla } x_i \text{ z drugiej}.$$

To może być zapisane jako:

$$y_i \, (\mathbf{w} \cdot x_i - b) \geq 1 \text{ dla wszystkich } 1 \leq i \leq n$$



Rys. 7.4: SVM z dwoma klasami (dane leżące na marginesie nazywają się wektorami wspomagającymi).

Można to zapisać jako problem optymalizacji w postaci:

$$\min_{\mathbf{w},b} \|\mathbf{w}\| \text{ pod warunkiem } (\forall i = 1,\ldots,n) : y_i\,(\mathbf{w}\cdot x_i - b) \geq 1.$$

Rozwiązanie problemu optymalizacji

Problem optymalizacji SVM jest trudny do rozwiązywania, gdyż zależy od $\|\mathbf{w}\|$, co wymaga pierwiastka kwadratowego. Na szczęście możliwe jest podstawienie za $\|\mathbf{w}\|$ wartości $\frac{1}{2}\|\mathbf{w}\|^2$ (czynnik $\frac{1}{2}$ jest użyty dla wygody liczenia) bez zmiany finalnego rozwiązania (minimum oryginalnego i zmodyfikowanego równania ma te same $\mathbf{w}$ i b). Rozwiązanie sprowadza się do minimalizacji $\frac{1}{2}\|\mathbf{w}\|^2$ mając na uwadze, że $y_i(\mathbf{w}\cdot x_i - b) \geq 1$ dla $\forall i = 1,\ldots,n$. Można to zapisać w postaci równania:

$$\min_{\mathbf{w},b} \max_{\alpha} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}\cdot x_i - b) - 1 \right] \right\}$$

co oznacza, że szukamy punktu siodłowego. Czyniąc w ten sposób wszystkie punkty, które mogą być wydzielone przez $y_i(\mathbf{w}\cdot x_i - b) - 1 > 0$ nie mają znaczenia, gdyż musimy ustawić odpowiadające α_i na zero. Rozwiązanie może być opisane przez liniową kombinację wektorów uczących:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Tyko nieliczne α_i będą większe od zera. Odpowiadające mu x_i są właśnie wektorami wspomagającymi (ang. *support vectors*), które leżą na marginesie i spełniają

$y_i(\mathbf{w} \cdot x_i - b) = 1$. Stąd wynika, że wektory wspomagające również spełniają

$$\mathbf{w} \cdot x_i - b = 1/y_i = y_i \Leftrightarrow b = \mathbf{w} \cdot x_i - y_i$$

co pozwala nam określić przesunięcie b.

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot x_i - y_i)$$

gdzie N_{SV} to ilość wektorów wspomagających.

Klasyfikacja nieliniowa

W roku 1992 Boser, Guyon i Vapnik, autorzy [10], zaproponowali metodę stworzenia nieliniowego klasyfikatora poprzez zastosowanie „sztuczki kernela" w celu zwiększenia marginesów hiperpłaszczyzn (pierwotnie proponowana przez Aizermana [11]). W rezultacie algorytm działa podobnie, z wyjątkiem tego, że każda operacja iloczynu skalarnego zastępowana jest przez nieliniową funkcję kernela. To pozwala algorytmowi zwiększyć margines pomiędzy hiperpłaszczyznami w transformowanej przestrzeni cech. Przykładowo, jeśli użytym kernelem jest Gaussian, odpowiadającą przestrzenią cech jest przestrzeń Hilberta. Kernel Gaussiana ma postać:

$$k(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

dla $\gamma > 0$ czasami parametryzowane przez $\gamma = \frac{1}{2\sigma^2}$

7.3. Praktyczna realizacja

Część systemu odpowiedzialna za zbieranie danych z czujników powinna gwarantować pełną obsługę błędów i wyświetlanie danych w celach debugowania. Następna część systemu odpowiedzialna za rozróżnianie gestów powinna być wyposażona w bazę wiedzy, na podstawie której będzie w stanie rozróżnić gesty. Taka baza wiedzy może powstać w procesie „nauczenia" odpowiednich gestów. W tym celu można wykorzystać sieć neuronową bądź odpowiedni klasyfikator. Podczas tworzenia bazy wiedzy należy zwrócić uwagę na różnorodne warunki w jakich system będzie musiał działać. W przypadku systemów wizyjnych może pojawić się wiele różnych utrudnień, takich jak: zmienne oświetlenie, zmienne tło, zmienna odległość od znaczników, zakłócenia od innych znaczników/źródeł światła, drgania systemu wizyjnego/kamery. W przypadku kamer podczerwonych zakłócenia są podobne, lecz poprzez zastosowanie filtracji na światło podczerwone, system może być bardziej odporny niż system wizyjny w pewnych przypadkach oświetleniowych.

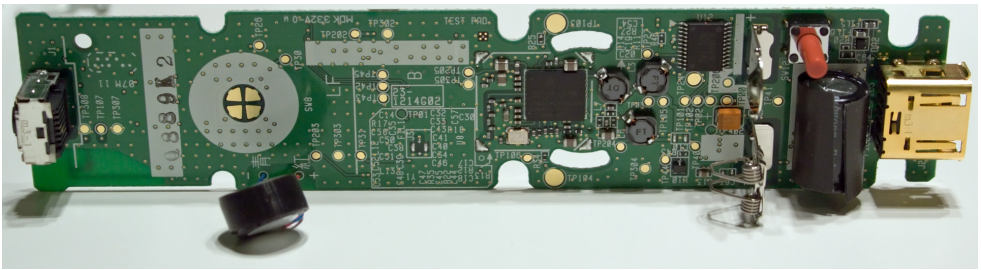
7.3.1. Dlaczego Wiimote?

Wii Remote, bo tak naprawdę nazywa się to urządzenie, jest jednym z najchętniej wykorzystywanych kontrolerów gier konsolowych na świecie (zobacz rys. 7.5). Wiimote jest nazwą nadaną przez ogromną rzeszę użytkowników konsoli Nintendo Wii - konsoli, do której Wii Remote został stworzony. Wiimote został zaprezentowany na Tokyo Game Show w październiku 2005 roku. Od tamtej pory nastąpił gwałtowny wzrost zainteresowanie możliwościami tego urządzenia. Powodem takiej sytuacji był ruch `http://wiibrew.com` [12] stosujący metody inżynierii odwrotnej w stosunku do konsol Wii oraz jej osprzętu. Opisywane urządzenie z rodziny Nintendo okazało się być bardzo przyjemne do obsługi dzięki wykorzystaniu protokołu komunikacyjnego HID (ang. *Human Interface Device*) Bluetooth, który jest standardowym protokołem dla wielu urządzeń. Dzięki temu też powstała duża ilość oprogramowania (ang. *third-party use*), która wykorzystywała m.in. Wii Remote w sposób jaki nie przewidział tego producent.

Rys. 7.5: Wii Remote firmy Nintendo.

W skład komponentów pilota Wiimote wchodzą (zobacz rys. 7.6):

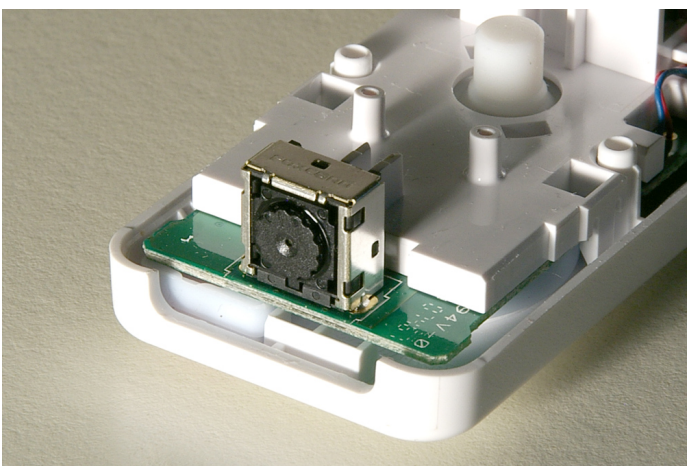
- jedenaście przycisków typu pushbutton,
- interfejs bezprzewodowy Bluetooth firmy Broadcom BCM2042,
- pamięć EEPROM 16kB,
- akcelerometr trzyosiowy ADXL330 firmy Analog Devices,
- kamera podczerwieni,
- cztery niebieskie diody led,
- silnik wibracyjny,
- głośnik,
- złącze do dodatkowych urządzeń.



Rys. 7.6: Widok płytki Wii Remote od strony układów.

Kamera podczerwieni

Kamera podczerwieni zainstalowana w Wii Remote jest najcenniejszym elementem. To właśnie za pomocą tej kamery w projekcie zbierano dane do klasyfikacji. Kamera posiada matrycę monochromatyczną o rozmiarze 128×96 oraz sprzętowe wstępne przetwarzanie obrazu polegające na wykrywaniu czterech przemieszczających się punktów podczerwieni. Kamera jest umieszczona za plastikowym filtrem podczerwonym (z testów empirycznych wynika, że najlepiej przepuszcza fale długości 940nm). Wbudowany mikroprocesor wykorzystuje ośmiokrotną subpikselizację obrazu, dzięki czemu wyjściową rozdzielczością jest 1024×768. Układ kamery jest wzbudzany 24MHz rezonatorem. Czujnik podczerwieni dostępny w Wiimote ma dość wąski zakres efektywnej pracy - jest to 33° w płaszczyźnie horyzontalnej oraz 23° w płaszczyźnie pionowej. Kamera podaje również wielkość punktu podczerwieni w trybie *extended* oraz ramę widzialności w trybie *full*.



Rys. 7.7: Kamera podczerwieni znajdującą się w Wii Remote.

Najsłynniejszy projekt z wykorzystaniem kamery IR - Wiimote Whiteboard

Jednym z projektów wykorzystujących Wii Remote, który zdobył wysokie uznanie, jest projekt interaktywnej tablicy wykonany przez Johna Lee [13]. W projekcie wykorzystano Wii Remote oraz obraz wysyłany z komputera na ekran monitora lub ekran rzutnika. Po wstępnej kalibracji czterech punktów na wyświetlanym obrazie, możliwe jest używanie źródła podczerwieni jako myszki w systemie Windows. Kamera podczerwona musi być skierowana na wyświetlany obraz. Podobne aplikacje istnieją dla systemów linuxowych. Projekt Johnna Lee był prezentowany na prestiżowej konferencji TED.

7.3.2. Biblioteki służące do obsługi urządzeń Wiimote

CWidd

Biblioteka CWiid[14], napisana w C, jest kolekcją narzędzi wspomagających tworzenie programów z wykorzystaniem kontrolerów Nintendo. Biblioteka została stworzona na platformę Linux, w szczególności istnieją jej dystrybucje ArchLinux, Debiana oraz Gentoo. O popularności omawianej biblioteki świadczy chociażby obecność w oficjalnych repozytoriach Ubuntu 10.10. Biblioteka składa się z czterech istotnych komponentów:

- `libcwiid` - pełne API,
- `wminput` - sterownik zdarzeń,
- `wmgui` - interfejs testowy,
- `wmdemo` - program demonstracyjny.

Twórcy CWiid są aktywni, wciąż zmieniają swoje dzieło i je aktualizują. Mimo tej aktywności dostępna dokumentacja jest mało przejrzysta i wdrożenie się w nią w krótkim czasie jest umiarkowanie trudne. Z powodu dokumentacji właśnie biblioteka CWiid nie została wykorzystana podczas tworzenia systemu Gestów Marszałka.

Wiimotedev Project

Trudności związane z użytkowaniem biblioteki CWiid stara się niwelować poprzez tworzenie nakładek. W efekcie powstają projekty ułatwiające użytkowanie CWiid, mające charakter „biblioteki do biblioteki". Takim projektem jest Wiimotedev Poject[15] stworzony przez Barłomieja Burdukiewicza. Omawiana biblioteka korzysta z CWiid, jest napisana w QT4 oraz bardzo dobrze kompiluje się pod systemami linuxopodobnymi. Została stworzony, aby programiści, chcący tworzyć programy z wykorzystaniem Wii Remote, mieli łatwy i pewny dostęp do tego urządzenia. Największą zaletą jest odwoływanie się do urządzenia poprzez „szynę" qdbusviewer z QT4. Biblioteka Wiimotedev składa się z pięciu części:

- `wiimotedev-daemon` - usługa systemowa działająca w tle,
- `wiimote-register` - program odpowiedzialny za połączenie z wii,
- `wiimotedev-client` - program kliencki do `wiimotedev-daemon`,
- `wiimotedev-toolkit` - program kontrolno-testowy urządzeń oraz biblioteki,

- `wiimotedev-car-example` - program demonstracyjny biblioteki.

W odróżnieniu do CWidd, projekt jest bardzo dobrze udokumentowany. Mimo to biblioteka nie została użyta w implementacji systemu Gestów Marszałka. Powodem były problemy z uruchomieniem wymaganych modułów Wiimota oraz stabilności działania. Wiimotedev Project jest obiecującym oprogramowanie, które jest rozwijane, warto skorzystać z niego w przyszłości, w połączeniu z programem pisanym w Qt4.

Wii Device Library

Ciekawą biblioteką napisaną w dość nowoczesnym („automagicznym") języku C# jest Wii Device Library [16]. Biblioteka jest międzyplatformowa, lecz wywodzi się z innego projektu napisanego w .NET (WiimoteLib [17]). Pozycja jest godna uwagi ze względu na przejrzystość działania oraz szeroką multiplatformowość oraz współpracę z różnorodnymi sterownikami interfejsu bluetooth.

wiiuse

Ostatnią biblioteką, którą rozważano do implementacji systemu Gestów Marszałka, była wiiuse [18]. Spośród wszystkich wypróbowanych bibliotek, wiiuse okazało się być najprzyjemniejsze w użytkowaniu. Biblioteka jest napisana w C, dzięki temu jest jasna i zrozumiała. Wiiuse działa zarówno pod systemami MS Windows do Visty włącznie oraz pod systemami linuxopodobnymi. Omawiany projekt posiada bardzo dobrą dokumentację, dzięki której można szybko i sprawnie uruchomić potrzebne moduły. Instalacja biblioteki sprowadza się do wydania poleceń:

```
$ make
$ sudo make install
```

Skrypt ten kompiluje i instaluje całą potrzebną bibliotekę w systemie linux. Aby móc korzystać z wszystkich możliwości biblioteki we własnym projekcie należy w kodzie dołączyć nagłówek `wiiuse.h` (i odpowiednio zlinkować kompilację).

7.3.3. Obsługa sprzętowo-programistyczna

Poniżej przedstawiono praktyczne porady dotyczące uruchomienia systemu powstałego w ramach projektu Gestów Marszałka.

Jak uruchomić Wiimota?

Aby połączyć się z urządzeniem Wii Remote pod systemem linux należy posiadać interfejs bluetooth zarówno w postaci fizycznej jaki i programowej (dla linuxa będą to biblioteki `bluetooth` oraz `bluez`, w szczególności dla Ubuntu 10.10 znajdują się one w Synapticu). Urządzenie bluetooth należy fizycznie włączyć, poczekać około jednej minuty, aż system uruchomi całą jego obsługę. Następnie należy uruchomić kontroler wiimote (potrzebne do tego są dwie baterie AA lub

zasilacz z krokodylkami dający napięcie 3V). Po dostarczeniu zasilenia najlepiej jest włączyć synchronizację Wiimota naciskając czerwony przycisk znajdujący się w komorze na baterie. Po wciśnięciu owego przycisku powinny zacząć mrugać wszystkie cztery diody urządzenia (mruganie to oznacza gotowość Wiimota na synchronizację z komputerem). Od tego momentu na połączenie się z Wii Remote pozostanie ok. 30 sekund.

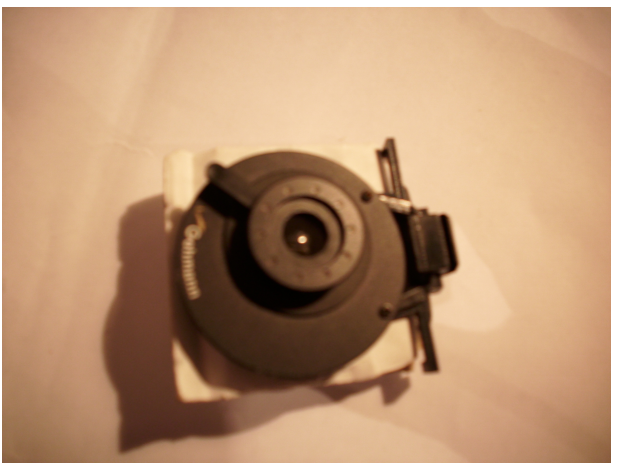
Aby dokonać testowego połączenia należy uruchomić w konsoli program `wiiuse-example`. Jeśli komputer połączy się z Wiimote, powinna zapalić się na stałe pierwsza dioda oraz na krótką chwilę włączy się alarm wibracyjny wewnątrz Wii Remote. Wewnątrz konsoli ukaże się aktualny status urządzenia. W przypadku niepowodzenia, należy powtórzyć sekwencję synchronizacji. Właściwie zawsze pierwsze uruchomienie wymaga kilkukrotnego powtórzenia sekwencji synchronizacji, w celu nawiązania stabilnego połączenia. Po uruchomieniu się przykładowego programu można włączać z pomocą przycisków znajdujących się na Wiimote urządzenia wewnętrzne, np. kamerę, akcelerometr czy alarm wibracyjny. Program przykładowy będzie raportował, jaki przycisk został wciśnięty oraz jakie dane aktualnie są dostępne.

Wielokrotne próby z przykładowym programem pozwoliły na poznanie specyfiki pracy Wii Remote. Dostrzeżono mianowicie, że kamera podczerwona nie zawsze się uruchamia, co stwarzało sporo problemów. Podczas badań nad biblioteką wiiuse odkryto, iż jednoczesne włączenie akcelerometru bądź alarmu wibracyjnego w niewiadomy sposób wspomagało drożność przesyłu danych z interesującej autorów kamery podczerwonej. Ten fakt jest uwzględniony w programie napisanym do obsługi Gestów Marszałka.

Testy kamery

Podczas empirycznych prób z kamerą podczerwoną na potrzeby testów stworzono testowe markery z źródłami światła podczerwonego. Podczas prób z różnymi sposobami na klasyfikację danych zrodził się pomysł rozróżniania diod podczerwonych na podstawie stałej odległości pomiędzy parami diod 7.8. W konsekwencji powstały dwa markery z równo-odległymi diodami. Jednocześnie zrodził się pomysł na rozróżnianie markerów od siebie za pomocą wielkości źródła światła podczerwonego. Ten pomysł spowodował powstanie kolejnych dwóch markerów z żarówkami samochodowymi. Wii Remote teoretyczne pozwala na rozróżnianie wielkości śledzonego punktu podczerwonego w zakresie od 0 do 15 jednostek. W praktyce mimo wielu prób z różnymi źródłami (poprzez silne żarówki, płomienie świeczek, wylotu hotair'a, czy grupy skupionych diod podczerwonych) udało się uzyskać maksymalnie wartość 6. Podczas testów nad rozpoznawaniem wielkości diody został stworzony testowy przyrząd złożony z żarówki samochodowej o napięciu 12V oraz blendy strzeleckiej z regulowaną przesłoną 7.9. Podczas testów z regulowaną przesłoną uzyskiwano płynne przejście od 0 do 6 jednostek pola punktu podczerwonego. Próby wykonywane były w odległości jednego metra od kamery. W przypadku za bliskiej odległości, mniejszej niż 20cm, kamera nie działała poprawnie. Maksymalny zasięg czujnika wynosił ok. 5 metrów. Podczas prób należało zwrócić uwagę na wyłączenie/ograniczenie

Rys. 7.8: Zestaw końcowy źródeł światła podczerwonego, parami rozmieszczonymi w jednakowych odległościach.



Rys. 7.9: Przyrząd do testowania rozpoznawania wielkości punktu podczerwonego.

nie wszelkich źródeł promieniowania mogących zakłócić pomiary, m.in. silnego światła słonecznego, żarówek/lampek oświetleniowych. Okazało się, że największe zakłócenia generowały świetlówki.

Niestety stabilność rozwiązania pod kątem rozróżniania wielkości punktu podczerwonego była niezadowalająca. Przyczyna leżała w malejącej rozdzielczości rozpoznawanej wielkości wraz ze wzrostem odległości (wzrost odległości sprawiał, że malała relatywna wielkość pola powierzchni punktu). Efektem było rozpoznawanie różnych źródeł jako tak samo duże punkty. Podczas wykonywania gestów duże żarówki samochodowe dawały wielkość 2, a małe diody podczer-

wone wielkość 1, przy czym zmiana kąta pomiędzy płaszczyzną czujnika, a źródeł światła zmniejszała relatywną powierzchnię, bądź powodowała zanik odczytu.

7.3.4. Uruchomienie systemu

Instalacja programu

Pierwszym krokiem podczas instalacji stworzonego oprogramowania jest odpowiednie przygotowanie systemu. Jak już wcześniej opisywano należy zacząć od instalacji paczek `bluetooth`, `bluez`, `ibus`, `libqt4`, `cmake`, `g++` dostępnych w Synapticu lub konsolowo należy wpisać:

```
$ sudo apt-get install bluetooth
$ sudo apt-get install bluez
$ sudo apt-get install ibus
$ sudo apt-get install libqt4
$ sudo apt-get install cmake
$ sudo apt-get install g++
```

Następną czynnością jest sprzętowe włączenie bluetootha. Po przetestowaniu sprawności tego urządzenia, np. poprzez połączenie się telefonem komórkowym, należy przejść do instalacji biblioteki wiiuse [18]. W celu zainstalowania biblioteki w konsoli należy przejść do folderu z biblioteką, do miejsca w którym jest `Makefile`, i wpisać komendę:

```
$ make
```

a następnie:

```
$ sudo make install
```

Następnym krokiem jest kompilacja oprogramowania do rozpoznawania gestów. Bibliotekami potrzebnymi do poprawnego działania systemu są `python`, `numpy`, `scipy` i `mlpy`. Dlatego w oknie konsoli należy wpisać:

```
$ sudo apt-get install python
$ sudo apt-get install numpy
$ sudo apt-get install scipy
$ sudo apt-get install mlpy
```

Biblioteki te są odpowiedzialne za działanie klasyfikatora SVM zaimplementowanego w pythonie. Po rozpakowaniu archiwum z programem należy w konsoli wejść do folderu `GestyMarszlka` i wpisać:

```
$ sh makefile.sh
```

Polecenie to skompiluje program `wii` odpowiadający za połączenie z Wii Remote, przesyłanie oraz formatowanie danych. Kolejnym krokiem jest wykonanie polecenia:

```
$ make
```

Polecenie to wywoła główny `Makefile` kompilujący interfejs wykonany w Qt4. Należy pamiętać, aby w systemie był uruchomiony `ibus-daemon`. Jeśli tak nie jest, to można uruchomić go z konsoli:

```
$ ibus-daemon
```

W tej chwili system rozpoznawania gestów jest już w pełni zainstalowany. Aby uruchomić program główny wystarczy przejść do folderu `/bin` i wpisać:

```
$ ./gesty
```

Obsługa programu

Po uruchomieniu aplikacji należy nauczyć program rozpoznawania gestów. W GUI należy kliknąć Plik->Uczenie. W tym momencie jest uruchamiany `klasyfikator.py`. Skrypt ten tworzy na podstawie danych klasyfikujących zawartych w folderze `/bin` bazę dla SVM. W konsoli wyświetlane są nazwy baz danych, z jakich skorzystał program, przy czym nazwa folderu jest nazwą klasyfikowanego gestu. Foldery będące bazami wiedzy dla programu muszą zawierać pliki z gestami. Pliki z gestami mogą mieć dowolne nazwy.

W pliku z danymi powinna znajdować się w każdej linii ramka danych z programu `wii`. Poniżej przedstawiono pseudokod ramki oraz przykładowy rzeczywisty ciąg danych:

```
size1 x1 y1 [size2 x2 y2] [size3 x3 y3] [size4 x4 y4] approx_X approx_Y time

3 0.069 0.544 4 -0.208 0.210 2 -0.417 0.609 -0.185 0.454 0.000
3 0.069 0.544 4 -0.208 0.210 2 -0.417 0.609 -0.185 0.454 0.010
3 0.069 0.544 4 -0.208 0.210 2 -0.417 0.609 -0.185 0.454 0.020
3 -0.769 0.272 3 -0.939 0.724 2 0.595 0.492 2 0.423 0.124 -0.173 0.403 0.020
3 -0.769 0.272 3 -0.939 0.724 2 0.595 0.492 2 0.423 0.124 -0.173 0.403 0.020
2 -0.435 0.883 2 -0.343 1.378 1 0.691 -0.129 1 0.382 -0.259 0.074 0.468 0.020
2 -0.586 0.711 2 -0.589 1.219 1 0.695 0.103 1 0.419 -0.124 -0.015 0.477 0.020
2 -0.662 0.572 2 -0.726 1.070 1 0.672 0.254 1 0.427 -0.030 -0.072 0.467 0.020
2 -0.752 0.317 2 -0.912 0.776 1 0.595 0.481 1 0.417 0.119 -0.163 0.423 0.020
2 -0.750 0.306 2 -0.916 0.768 1 0.587 0.497 1 0.415 0.129 -0.166 0.425 0.020
2 -0.752 0.304 2 -0.918 0.760 1 0.586 0.502 1 0.415 0.132 -0.167 0.424 0.020
1 -0.750 0.301 2 -0.918 0.760 1 0.582 0.507 1 0.413 0.137 -0.168 0.426 0.020
2 -0.748 0.301 2 -0.914 0.760 1 0.580 0.510 1 0.412 0.142 -0.168 0.428 0.020
2 -0.750 0.301 2 -0.914 0.760 1 0.580 0.510 1 0.412 0.142 -0.168 0.428 0.020
2 -0.748 0.301 2 -0.908 0.760 1 0.574 0.518 1 0.410 0.142 -0.168 0.430 0.020
2 -0.748 0.299 2 -0.910 0.757 1 0.574 0.520 1 0.410 0.145 -0.169 0.430 0.020
```

Niestety, czas jest podawany z dość dużym interwałem, wynoszącym dla 0,01 ok 0,43s. Aby poprawić ten parametr (nie przeszkadza to klasyfikatorowi) należałoby odwołać się do funkcji w kodzie maszynowym, które korzystałyby bezpośrednio z przerwań procesora. Po dokonanym uczeniu w folderze `/bin` tworzą się logi z których korzysta `klasyfikator.py`.

Po udanej klasyfikacji gestów można przystąpić do nagrywania gestu. Aby tego dokonać należy przełączyć Wii Remote w tryb synchronizacji przy pomocy czerwonego przycisku znajdującego się w komorze na baterie oraz kliknąć w pushbutton „Połącz”. W tym momencie w konsoli powinnien pojawić się status urządzenia Wii Remote, a na samym urządzeniu powinna zapalić się pierwsza

niebieska dioda. W przypadku przyciskania guzików na Wiimote w konsoli powinny pokazywać się raporty informujące o wciśnięciu przycisku. W celu pobudzenia czujnika podczerwieni warto wcześniej „udrożnić" komunikację z Wiimote i wcisnąć przycisk „B" znajdujący się pod spodem Wii Remote. Przycisk ten włączy silnik wibracyjny. Po chwili należy go wcisnąć powtórnie, aby wyłączyć wibracje. Ten drobny test pozwala potwierdzić poprawności działania Wii Remote. Następne, aby dokonać „nagrania" gestu należy przygotować źródła światła podczerownego. Warto wyłączyć wszystkie inne źródła podczerwieni w celu uniknięcia niepożądanych zakłóceń. Po tych czynnościach należy wcisnąć przycisk „UP". W tym momencie program `wii` zacznie przesyłać strumień danych do pliku `dane.txt`, który jest za każdym razem kasowany i nadpisywany nowymi danymi. Osobę pokazującą gest należy umieścić w polu widzenia kamery, starając się aby źródła podczerwieni przemieszczały się w środkowym sektorze - wtedy zostaną prawidłowo wykryte przez czujnik podczerwieni. Klasyfikator odczytujący plik `dane.txt` jest odporny na charakterystyczne jednostkowe szumy występujące w strumieniu danych z Wii Remote. Następnie po nagraniu dwu sekundowego gestu można przystąpić do klasyfikacji gestu poprzez naciśnięcie pushbutton'a „Klasyfikuj". W konsoli w tle ukażą się wyniki klasyfikacji. Aby ujrzeć w GUI należy kliknąć pushbutton „Pokaż wynik". Pozytywny wynik klasyfikacji jest oznaczony poprzez liczbę „1" znajdującą się po nazwie gestu z bazy, „-1" oznacza negatywny wynik klasyfikacji gestu. Kliknięcie w Plik->Koniec, rozłączy komunikację z Wii Remote, wyczyści logi z folderu `/bin` oraz zamknie aplikację.

Istnieje możliwość dokonania klasyfikacji gestu w przypadku braku urządzenia Wii Remote. Można tego dokonać poprzez utworzenie pliku o nazwie `dane.txt` w folderze `/bin` i umieszczenie tam danych z dowolnego pliku pobranego z bazy. Dopuszcza się samodzielne wygenerowanie danych, obowiązkowo ramka powinna wyglądać w następujący sposób:

```
size1 x1 y1 [size2 x2 y2] [size3 x3 y3] [size4 x4 y4] approx_X approx_Y time
```

gdzie `size1` nie może być zerowe, a `approx_X` i `approx_Y` są średnią arytmetyczną współrzędnych „iksowych" oraz „igrekowych". Czas jest nieistotny, ale musi być podany. Jeżeli wykorzystywane są dane skopiowane z bazy, warto usunąć ten plik z bazy. Wtedy przeprowadzona zostanie rzeczywista klasyfikacja gestu, który nie istnieje w identycznej postaci w bazie wiedzy. Po utworzeniu takiego pliku należy kliknąć „Uczenie", a następnie „Klasyfikacja". Program powinien sklasyfikować podany gest.

7.3.5. Dane do Klasyfikacji

Otrzymywane dane Ω w procesie akwizycji (trwającej dwie sekundy) zawierają ciągi pozycji poszczególnych diod nagrywanych w dwusekundowych sekwencjach (maksymalna długość odczytanego gestu to $M = 300$).

$$\Omega = \{\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4\},$$

gdzie Λ_i to odczyt z pojedynczej diody.

$$\Lambda_i = (\lambda_1, \lambda_2, \ldots, \lambda_k), i = \{1,2,3,4\}, k \leq M, \lambda_i = (x_i, y_i), i \leq k.$$

Położenie punktów na płaszczyźnie $2D$ określa się zmiennymi $x, y \in < -1, 1 >$. Do klasyfikacji użyto jedynie środka ciężkości z czterech diod $\Lambda_{śr}$:

$$\Lambda_{śr} = \left(\frac{\Lambda_1[1] + \Lambda_2[1] + \Lambda_3[1] + \Lambda_4[1]}{4}, \ldots, \frac{\Lambda_1[k] + \Lambda_2[k] + \Lambda_3[k] + \Lambda_4[k]}{4} \right)$$

7.3.6. Opis działania klasyfikatora SVM

W module o nazwie `klasyfikator.py`, napisanym w języku programowania Python, następuje klasteryzacja danych uczących, jak i klasyfikacja wektora cech. Ponieważ wektory cech pobieranych z urządzenia Wiimote są różnej długości, a algorytm SVM wymaga, aby były one tego samego wymiaru, dane wstępnie sprowadzane są do wektorów stuwymiarowych (50 wartości x i 50 wartości y) poprzez interpolację. Wymiar wektora został dobrany na podstawie badań. Jest on na tyle duży, że nie traci informacji o rodzaju gestu, a na tyle mały, że algorytm SVM działa bardzo szybko.

Interpolacja polega na rozciąganiu lub też zwężaniu długości wektora cech. Przykładowo, mając sinusoidę o okresie 70 próbek po interpolacji otrzymamy sinusoidę o okresie 50 próbek zachowując jej kształt. W rezultacie każdy gest zostaje sprowadzony do „długości trwania" 50 jednostek czasu. Interpolacja przebiega osobno dla wartości x jak i y, otrzymany wektor: interpolacja $(\Lambda_{śr}) = \Lambda_{int} = \{(x_1, x_2, \ldots, x_{50})^T, (y_1, y_2, \ldots, y_{50})^T\}$ zostaje następnie zapisany do postaci jednowymiarowej poprzez przeplatanie wartości x_i i y_i, dla $i = 1 \ldots 50$. W wyniku całej operacji wektor $\Lambda_{flat} = (x_1, y_1, x_2, y_2, \ldots, x_{50}, y_{50})^T$. Następnie przeprowadzany jest algorytm SVM, dla każdego wektora uczącego Λ_{int}.

Opis algorytmu SVM szerzej opisany jest w pracy [19]. Nie wchodząc w szczegóły polega on na tym, żeby rozdzielać dwa klastry danych prostą w taki sposób, żeby marginesy pomiędzy prostą a danymi były jak największe. Na podstawie powstałej prostej przeprowadzana jest później klasyfikacja nowych wektorów. Jako, iż algorytm SVM oddziela od siebie dwa klastry, niezbędne jest zastosowanie algorytmu SVM dla każdej klasy, tak aby późniejsza klasyfikacja pozwoliła określić, czy dane wejściowe należą do danego klastra, czy nie. Dlatego może się zdarzyć, że wektor wejściowy zostanie zaklasyfikowany do więcej niż jednej klasy. Podczas przeprowadzania testów sytuacja ta jednak nigdy nie miała miejsca. Wytłumaczeniem jest fakt, że gesty różniły się między sobą i w stuwymiarowej przestrzeni klastry były umiejscowione daleko od siebie tak, że SVM radził sobie z nimi wyśmienicie.

7.3.7. Testy walidacyjne dla różnych wartości kerneli

Do badania poprawności klasyfikacji wygenerowano średnio 50 sekwencji dla jednego rodzaju gestu. Spośród kilkudziesięciu gestów dostępnych w palecie ruchów Marszałka wybrano sześć. Kierowano się tym, aby niektóre były znacząco od siebie różne, jak również występowało między niektórymi pewne podobieństwo. Gestami podobnymi były „Rot 90 stopni", „Koło w lewo", zupełnie odbiegające od pozostałych były gesty: „Skos w lewo", „Gest 1".

Tab. 7.1: Błędy średnie dla różnych kerneli

Gest	linear	gaussian	tr	tversky
Skos w lewo	0,0000	0,0563	0,0000	0,0000
Koło w prawo	0,2058	0,8379	0,3313	0,2550
Koło w lewo	0,2086	0,8179	0,3246	0,2616
Rot 90 stopni	0,0266	0,1588	0,0000	0,0033
Gest 1	0,0297	0,0430	0,0066	0,0033
Gest 2	0,0330	0,1658	0,0066	0,0066

Klasyfikator napisany w Pythonie ma możliwość dobierania różnych rodzajów kerneli wykorzystywanych podczas obliczania prostych oddzielających klastry. Przeprowadzono testy czterech takich kerneli (w tabeli 7.1 przedstawiono, jak plasował się średni błąd przyporządkowań danych wejściowych). Testy przeprowadzane były dla kroswalidacji z wartością 5, co oznacza, że każdy ciąg danych dzielony był na pięć części, przy czym 4 z nich brane były do uczenia algorytmem SVM a piąta brana była jako ciąg testujący.

Zauważono, że wartości błędu dla kernela liniowego są zadowalające, a SVM działający z tym kernelem działa najszybciej (postanowiono korzystać właśnie z niego podczas właściwego działania programu, jakkolwiek w każdej chwili można zmieniać to ustawienia kernela). Patrząc na wyniki widać, że gesty z pozoru różne, takie jak „Koło w prawo" i „Koło w lewo" dla każdego kernela posiadają zbliżoną wartość błędu klasyfikacji. Zatem można przypuszczać, że to iż gesty zaczynały się i kończyły w tych samych punktach o obu przypadkach miało mocny wpływ na bliskie umiejscowienie klastrów w przestrzeni.

7.4. Zastosowanie w dydaktyce

Program rozpoznający gesty w założeniu miał służyć celom dydaktycznym. Funkcja edukacyjna miała opierać się na programie oraz algorytmie, który stworzono w ramach pracy nad oprogramowaniem rozpoznającym. Procesowi edukacji mieli zostać poddani członkowie personelu lotniczego pragnący doskonalić swoje lotnicze umiejętności. Wiedza dotycząca wykonywania oraz rozpoznawania gestów marszałka jest potrzebna:

- pilotom, którzy wykonują odpowiednie procedury, w zależności od rozpoznanego gestu wydanego przez marszałka na lotnisku kontrolowanym;
- obsłudze naziemnej lotniska, która ma uprawnienia do kierowania ruchem lotniskowym;
- pasjonatom lotnictwa, którzy mogą dzięki temu poszerzać swoją wiedzę.

Program rozpoznawania gestów został stworzony do uniwersalnego rozpoznawania dowolnego gestu pokazanego przez maksymalnie cztery źródeł światła podczerwonego. Uniwersalność ta jest zapewniona przez korzystanie z bazy danych, którą można utworzyć samemu. Uczenie mogłoby zostać zrealizowane

przez program, który określałby gest do wykonania przez użytkownika, natomiast użytkownik musiałby wykonać ten gest (narzędziem z dioda podczerwonymi). Program po nagraniu gestu dokonałby klasyfikacji, czyli stwierdziłby, do jakiej klasy zaliczyć gest. Funkcja dydaktyczna po zakończeniu testów pokazywałaby statystykę poprawności odpowiedzi. Taki test mógłby być powtarzany dla wielu różnych gestów.

System mógłby pełnić rolę również interaktywnego nauczyciela. Program dydaktyczny pokazywałby użytkownikowi nagranie gestu wykonane kamerą światła widzialnego, a użytkownik mógłby wykonywać ten gest przy użyciu zestawu diod podczerwonych. W ten sposób można by automatycznie tworzyć bazę danych dla programu, każdy nagrany gest byłby umieszczany w bazie, z której następnie po odpowiedniej filtracji korzystałby program.

7.5. Podsumowanie i dalsze badania

Przeprowadzane badania dowiodły, że akwizycja danych za pomocą kontrolera Wiimote, jak i klasyfikacja metodą SVM spełniają wymagania postawione dla systemu rozpoznającego Gesty Marszałka. Błędy klasyfikacji są na tyle małe, że zdecydowana większość gestów jest poprawnie wykrywana. Problemami, które mogą wystąpić podczas korzystania z aplikacji są m.in. mała powtarzalność w reagowaniu sprzętu na początki nagrywania sekwencji oraz odpowiednia pozycja osoby wykonującej gesty (aby pobierane dane nie traciły swoich cech przez rzutowanie). Po odpowiednim zaprogramowaniu interfejsu program może z powodzeniem służyć do nauki Gestów Marszałka.

W przyszłości warto byłoby porównać inne metody klasyfikacji gestów, zwłaszcza uprzednio założoną metodę sieci neuronowych. Jednak złożoność algorytmu obsługującego nauczanie i modelowanie odpowiedniej sieci neuronowej (ze sprzężeniami zwrotnymi wg. [8]) była znaczną przeszkodą w realizacji projektu na czas, jednak ciekawym aspektem badawczym byłaby realizacja owych sieci i porównanie jej ze „statyczną" metodą klasyfikacji jaką jest metoda SVM, pomimo tego, że skuteczność SVM w tym przypadku jest bardzo wysoka.

Literatura

[1] Marshaller Handsignals. `www.traron.org/docs/Marshaller%20Handsignals.pdf`

[2] Rulse of the Air - Annex 2 to the Convention on International Civil Aviation. `http://www2.tech.purdue.edu/at/courses/at300/Documents/Annex2.pdf`, (1990).

[3] Visual Aids Handbook - A compendium of Visual Aids intended for the guidance of Pilots and Personnel engaged in the handling of aircraft. `http://www.caa.co.uk/docs/33/CAP637.PDF`, (2007).

[4] C. V. i D Metxas: *Toward scalability in asl recognition: Breaking down signs into phonemes*, chapter Gesture Workshop Gif sur Yvette, (1999).

[5] N. J. i. D. H. A. Galata: *Learning behaviour models of human activities*, pp. 12–22 BMVC, (1999).
[6] N. Johnson: *Learning object behaviour models* PhD thesis, University of Leeds, (1998).
[7] A. P. i. S. G. M. Walter: Auto Clustering for Unsupervised Learning of Atomic Gesture Components Using Minimum Description Length., (2001).
[8] P. M. Asaro: Temporal Pattern Recognition with Neural Networks. [online], (1998).
[9] M. Mozer: *Backpropagation: Theory, Architecture, and Applications*, chapter A Focused Backpropagation Algorithm for Temporal Pattern Recognition Lawrence Erlbaum Associates, (1995).
[10] V. V. B.E. Boser, I.M. Guyon: *A training algorithm for optimal margin calssifiers*, pp. 144–152 ACM Press, (1992).
[11] M. A. L. Rozonoer, E. Braverman: *Theoretical foundations of the potential function method in pattern recognition learning*, chapter Automation and Remote Control, pp. 821–837, (1964).
[12] Wii Brew. `http://wiibrew.org/wiki/Wiimote`
[13] Johnny Lee . `http://johnnylee.net/projects/wii/`
[14] CWidd . `http://abstrakraft.org/cwidd`
[15] Wiimotedev Project. `http://gitorius.org/wiimotedev`
[16] Wii Device Library. `http://www.softwarebakery.com`
[17] WiimoteLib. `http://www.brianpeek.com/blog/pages/wiimotelib.aspx`
[18] wiiuse. `http://www.wiiuse.net`
[19] J. S.-T. . N. Cristianini: *Support Vector Machines and other kernel-based learning methods* Cambridge University Press, (2000).

ROZDZIAŁ 8

KOMPUTEROWA PSEUDOLOSOWOŚĆ

S. Chlebicki, M. Skoczylas

W niniejszym rozdziale przedstawiono takie zagadnienia związane z generowaniem liczb pseudolosowych jak: definicje i historia liczb losowych, problematyka generowania liczb pseudolosowych, sposoby tworzenia i testowania wybranych generatorów (liniowego i kwadratowego). Opisano w nim również sposób wykorzystywania liczb pseudolosowych. Ponadto omówiono przykłady praktycznych implementacji wykorzystujących generatory liczb pseudolosowych, działających na mikrokontrolerze, pozwalających na: wizualizację działania generatora liniowego i kwadratowego na wyświetlaczu LCD, oraz szyfrowanie algorytmem XOR danych przesyłanych bezprzewodowo.

8.1. Liczby losowe

Liczby losowe w nieformalnym ujęciu traktowane są jak wartości, których wartości nie można przewidzieć rozpatrując dany eksperyment losowy. Formalnie rzecz biorąc liczba losowa jest konkretną wartością przyjmowaną przez zmienną losową. Jako przykład mogli byśmy podać dowolną liczbą rzeczywistą. Trudno jest jednak rozpatrywać losowość pojedynczej liczby [1].

Zmienna losowa ξ jest z kolei funkcją, która przyporządkowuje zdarzeniom elementarnym liczby. Przykładem takiej funkcji jest wzrost albo waga człowieka przypadkowo napotkanego w danej populacji, lub wypadkowa liczba oczek podczas rzucania kością do gry. Gdy wartością funkcji będącej zmienną losową są iczby rzeczywiste, mówimy o zmiennej losowej rzeczywistej, tzn.

$$\xi : \Omega \rightarrow \mathbb{R} \tag{8.1}$$

Zdarzenie elementarne jest elementem zbioru tradycyjnie oznaczanym przez Ω. Zbiór ten reprezentuje wszystkie możliwe wyniki eksperymentu losowego. Dla przykładu przy rzutach monetą zbiorem zdarzeń elementarnych jest

$$\Omega = \{\text{Orzeł}, \text{Reszka}\} \tag{8.2}$$

Kluczem do zdefiniowania zmiennej losowej, jest określenie, czym jest „eksperyment losowy", lub też czym jest losowość. Klasyczna teoria prawdopodobieństwa,

rozwijana przez Pascala i Fermata, mówi, że wyniki losowego eksperymentu są jednakowo prawdopodobne, tzn. że gdy losujemy spośród n liczb, szansa na wylosowanie konkretnej wartości wynosi

$$P(x) = \frac{1}{n} \tag{8.3}$$

Przyczyną losowości zjawiska mogą być jego specyficzne cechy fizyczne, bądź niezmierna komplikacja niemożliwa do objęcia żadnym zdeterminowanym modelem [2].

8.2. Liczby pseudolosowe

There is no such thing as a random number,
there are only methods to produce random numbers.
John von Neumann

W odróżnieniu od liczb losowych, liczby pseudolosowe są wynikiem działania algorytmu, będącego ścisłym wzorem matematycznym. Algorytm taki nazywany jest generatorem liczb pseudolosowych. Ponieważ generatory liczb losowych są niezwykle nieliczne i rzadko wykorzystywane, potocznie mówiąc o generatorach liczb losowych ma się na myśli właśnie generatory liczb pseudolosowych [3]. Wspomniane algorytmy są implementowane w taki sposób aby liczby pseudolosowe posiadały wybrane własności liczb losowych.

Liczby pseudolosowe posiadają ważną cechę, która umożliwia ich wykorzystanie tam, gdzie liczby losowe nie spełniałyby należycie swej roli. Można je stosunkowo szybko otrzymywać, tzn. nie wykonując eksperymentu losowego ani nie wykonując pomiarów. Wystarczy wyliczyć pewną wartość wedle określonej reguły.

8.3. Zastosowanie liczb pseudolosowych

Zainteresowanie liczbami pseudolosowymi wynika z praktycznych ich zastosowań. Liczby losowe potrzebne są do realizacji zadań, w których efekt przypadkowy jest lepszy niż zdeterminowany. Poniżej opisano dziedziny, w których wykorzystuje się liczby pseudolosowe.

Kryptografia Być może najważniejszą dziedziną nauki wykorzystującą generatory liczb pseudolosowych jest kryptografia. Wiele algorytmów szyfrowania wykorzystuje właściwość losowości. Jakość generatorów dla tej dziedziny ma krytyczne znaczenie, gdyż możliwość przewidzenia kolejnej liczby będącej wynikiem losowania jest wyznacznikiem bezpieczeństwa. Ponadto wysoka szybkość ich działania umożliwia szyfrowanie i deszyfrowanie w czasie rzeczywistym, co pozwala na realizacje bezpiecznych transferów danych.

Testy sprzętu i oprogramowania Liczby pseudolosowe wspomagają automatyczne testowanie sprzętu i oprogramowania. Taka metodologia jest powszechnie stosowanym sposobem nowoczesnego i szybkiego rozwoju technologicznego. Losowane mogą być zarówno same dane wejściowe, jak i sekwencje (pakiety) danych.

Symulacje komputerowe Liczby pseudolosowe są równie często stosowane do realizacji symulacji. Symulacje wspomagają prace inżynierskie i ekonomiczne. Liczby pseudolosowe mogą być wykorzystywane jak w przypadku testów do generowania danych, lub do realizacji zakłóceń, które mają charakter losowy.

Gry komputerowe Dziedziną podobną do symulacji są gry komputerowe. Tutaj zastosowanie liczb pseudolosowych ma jednak inne znaczenie. Liczby te są wykorzystywane do zwiększenia tzw. grywalności. Gra jest lepiej odbierana, gdy rozwój wydarzeń bądź scenariusz jest trudny do przewidzenia przez gracza. Rzecz może dotyczyć samego scenariusza gry, ale również renderowanej grafiki. Biblioteki programistyczne zwane „silnikami fizyki", wykorzystywane przez twórców gier, mogą stosować losowość do symulacji zakłóceń. Ponadto w grach wykorzystywane są metody sztucznej inteligencji, które również wykorzystują liczby pseudolosowe.

Sztuczna inteligencja Wiele metod sztucznej inteligencji wykorzystuje liczby pseudolosowe. Są to tak zwane metody oparte na prawdopodobieństwie.

Metody numeryczne Liczby pseudolosowe są również elementem metod numerycznych. Jak pokazują badania jakość generatorów wpływa na czas zbieżności algorytmów Monte Carlo. Można też wykorzystać je w obliczeniach przeprowadzanych zwykle metodami deterministycznymi (np. w obliczaniu całek wielowymiarowych) [2]. W algorytmach sztucznej inteligencji liczby pseudolosowe odgrywają szczególną rolę przy realizacji rzeczywistych procesów losowych, takich jak prawdopodobieństwo mutacji czy prawdopodobieństwo krzyżowania.

8.4. Generatory liczb losowych

Generatory liczb losowych to metody otrzymywania liczb prawdziwie losowych. Oznacza to, że związane są one z pewnym procesem fizycznym, który jest źródłem losowości.

Rzut monetą Jako przykład rozpatrzmy eksperyment rzucania monetą. Jest to tak zwany generator mechaniczny [3]. Generator ten pozwala na wylosowanie dowolnie dużej liczby. Jeśli wyniki ośmiu losowań potraktować jako wartości ośmiu bitów, to otrzymuje się generator liczb z przedziału od 0 do 255. Dobierając odpowiednią metodę kodowania wyniku losowań można generować liczby ze znakiem i liczby zmiennoprzecinkowe. Problemem jest jednak powolny proces

losowania, w związku z czym tego typu generator nie może być wykorzystany do realizacji zadań opisanych w punkcie 8.3.

Pomiar szumów Szum jest przykładem procesu losowego. Ze względu na własności statyczne często wykorzystywany jest szum biały. Metoda pomiaru, choć szybsza od wspomnianego eksperymentu rzucania monetą, jest nadal zbyt wolna, by tego typu generator można było wykorzystać w większości zastosowań.

Pomiar innych procesów fizycznych W literaturze [3] opisano, iż buduje się generatory liczb losowych oparte o procesy fizyczne takie jak rozpad radioaktywny czy temperatura. Z wykorzystaniem tego typu generatorów wiąże się problem stabilności: niewielkie zmiany warunków zewnętrznych mogą powodować zmiany własności probabilistycznych otrzymywanych liczb.

Urządzenie `/dev/random` Ciekawym i wartym wspomnienia rozwiązaniem realizującym generator liczb losowych w komputerze jest urządzenie `/dev/random`. Jest to plik występujący w systemach operacyjnych będących pochodnymi Unix'a, sprzężony ze znajdującym się na płycie głównej lub karcie dźwiękowej generatorem szumu. Jeśli brak takiego urządzenia, wykorzystany może być czujnik temperatury procesora. Przykład wykorzystania pliku przedstawiono na rys. 8.1.

Rys. 8.1: Wynik wypisania pliku `/dev/random` w terminalu.

Alokacja pamięci Szybką metodą wykorzystywaną przez programistów do generowania liczb losowych jest alokacja obszaru pamięci i odczytanie zawartych tam danych. Metoda ta jest naturalnie nie do zaakceptowania w kryptografii. Ponadto własności statystyczne tak generowanych liczb losowych mogą być niezadowalające.

8.5. Generatory liczb pseudolosowych

Random number generators should not be chosen at random.
Donald Knuth

Problemy związane z generowaniem liczb losowych skłaniają użytkowników do sięgnięcia po szybsze (choć mniej bezpieczne) metody, jakimi są generatory liczb pseudolosowych. Przykładami generatorów liczb losowych są:

- generator liniowy,
- generator kwadratowy von Neumanna,
- uogólniony generator Fibonacciego,
- generator SWB (substract with borrow),
- generator SR (shift register),
- generator MWC (multipy with carry),
- Eichnauer & Lehn,
- Eichnauer & Herman,
- Blum, Blum & Shub.

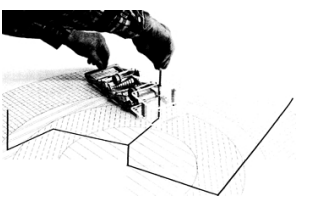
Ich nazwy są związane z nazwiskiem twórcy bądź metodą działania generatora. Różnią się one przede wszystkim szybkością działania i poziomem bezpieczeństwa (z punktu widzenia wykorzystania w kryptografii).

8.6. Historia rozwoju metod otrzymywania liczb losowych

W tym podrozdziale opisano wybrane wydarzenia historyczne mające wpływ na rozwój generatorów liczb pseudolosowych.

W 1733 roku francuski matematyk Georges-Louis Leclerc (obraz obok, źródło: `http://pl.wikipedia.org/w/index.php?title=Plik:Buffon_1707-1788.jpg&filetimestamp=20060418182514`) sformułował tzw. problem Igły Buffona. W 1777 podał on jego rozwiązanie. Problem polegał na opracowaniu eksperymentu pozwalającego na wyznaczenie liczby π za pomocą igły rzucanej na kartkę papieru podzieloną liniami.

W 1927 roku L.H. Tippet wydał pierwszą tablicę liczb losowych pod tytułem „Random Sampling Numbers". Liczby losowe miały źródło w spisie ludności Wielkiej Brytanii oraz powierzchni brytyjskich parafii. Tablice tworzył ciąg 41600 cyfr.



W latach 1930-tych Enrico Fermi zrealizował obliczenia dyfuzji neutronów w oparciu o liczby losowe. Do ich generowania wykorzystał zbudowane przez siebie urządzenie mechaniczne zwane FERMIAC (zdjęcie obok, źródło: `http://en.wikipedia.org/wiki/File:FERMIAC.jpg`). Jest to tzw. komputer analogowy.

W roku 1939 powstały dwie ważne tablice liczb losowych. Autorami pierwszej byli R.A. Fisher i F. Yates. Ich tablica składała się z 15000 cyfr losowych. Autorami drugiej byli Kendall, Babington i Smith. Ich tablica składała się ze 100000 cyfr losowych, wyprodukowanych przy użyciu tzw. elektronicznej ruletki. Był to

dysk napędzany silnikiem elektrycznym, z wyróżnionymi wycinkami odpowiadającymi cyfrom od 0 do 9.

W latach 1940-tych duży wpływ na rozwój generatorów liczb pseudolosowych miały prace nad projektem Manhattan, w których udział brali J. von Neumann, N. Metropolis i S. Ulam (rys. 8.2).


J. von Neumann


S. Ulam


N. Metropolis

Rys. 8.2: Uczestnicy projektu Manhattan (źródło: `http://commons.wikimedia.org/wiki/Category:People_associated_with_the_Manhattan_Project`).

W latach 1950 znacznie wzrosło zainteresowanie metodami Monte Carlo. Nie było jednak dostatecznie wydajnych maszyn cyfrowych by je realizować. W 1951 roku na potrzeby Głównego Urzędu statystycznego opracowano tablicę liczb losowych. W 1955 roku firma RAND Corporation opracowała tablicę 1000000 cyfr losowych przy użyciu specjalnego urządzenia elektronicznego generującego i liczącego impulsy. Dalszy rozwój dziedziny związany był z postępem w komputeryzacji.

W 1995 roku nastąpił pewnego rodzaju powrót do idei tablic, kiedy to G. Marsaglia wydał CD-ROM zawierający 650MB liczb losowych, wraz z testami Diehard.

8.7. Generator liniowy

Komputerowy generator liczb pseudolosowych [4] opisany jest zależnością:

$$x_{n+1} = f(x_0, x_1, ..., x_n) \tag{8.4}$$

Jest to deterministyczna funkcja, obliczająca $i+1$-szy element na podstawie poprzednich elementów. W przypadku generatora liniowego (8.4) przyjmuje ona postać:

$$x_n = (ax_{n-1} + c)\,\%m, \tag{8.5}$$

gdzie x_n - n-ta liczba pseudolosowa, x_{n-1} - poprzednia liczba pseudolosowa, a - mnożnik, c - przyrost, m - moduł. Znak % w równaniu (8.5) oznacza operację dzielenia modulo.

8.7.1. Typy generatorów liniowych

Wyróżnia się dwa typy generatorów liniowych [5]:

- addytywne: $x_n = (ax_{n-1} + c)\%m$,
- multiplikatywne: $x_n = ax_{n-1}\%m$.

Jedyną różnicą strukturalną tych generatorów jest brak czynnika przyrostowego w przypadku generatora multiplikatywnego. Ma to jednak istotny wpływ na różnice funkcjonalne generatorów, gdyż zmienia się zakres generowanych przez nie liczb. Dla generatora addytywnego mamy więc: $x \in \{0, 1, ..., m-1\}$, a dla generatora multiplikatywnego: $x \in \{1, 2, ..., m-1\}$.

8.7.2. Dobór współczynników

W [4] przedstawiono problematykę doboru współczynników a, c i m generatora liniowego, oraz podano twierdzenie pozwalające na budowę generatorów o zadanym okresie. Dla współczynników $a = 5$, $c = 3$ i $m = 16$ oraz $x_0 = 0$ (8.5) pozwala na utworzenie następującego ciągu:

```
0, 3, 2, 13, 4, 7, 6, 1, 8, 11, 10, 5, 12, 15, 14, 9
0, 3, 2, 13, 4, 7, 6, 1, 8, 11, 10, 5, 12, 15, 14, 9
0, ...
```

Można zauważyć, że okres wynosi 16, czyli

$$a = 5, c = 3, m = 16, x_0 = 0 \quad \Rightarrow \quad x_i \in \{0, 1, ..., 15\} \tag{8.6}$$

Generator nie zwróci liczby większej od 15 ze względu na operację dzielenia modulo przez $m = 16$. Dla innego doboru parametrów: $a = 3$, $c = 4$ i $m = 16$ oraz $x_0 = 0$ utworzony zostanie następujący ciąg:

```
0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4
0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4
0, ...
```

W tym przypadku okres wynosi 2, czyli

$$a = 5, c = 3, m = 16, x_0 = 0 \quad \Rightarrow \quad x_i \in \{0, 4\} \tag{8.7}$$

Uzyskany w ten sposób generator nie jest dobry. Nie zwraca liczb z określonego zakresu, a na podstawie jego modułu nie można ustalić okresu, dla którego liczby wynikowe miałyby jednakowe prawdopodobieństwo wylosowania.

Twierdzenie Dla generatora liniowego o parametrach a, c i m ciąg generowanych liczb losowych ma długość m wtedy i tylko wtedy, gdy

- c i m nie mają wspólnych dzielników,
- $b = a - 1$ jest wielokrotnością każdej liczby pierwszej p, która jest dzielnikiem liczby m,
- b jest wielokrotnością 4, o ile m jest te wielokrotnością 4.

W tab. 8.1 przedstawiono zestawienie współczynników a, c i m znanych generatorów liczb pseudolosowych [6].

Tab. 8.1: Współczynniki znanych generatorów liniowych liczb pseudolosowych.

Nazwa	m	a	c
Numerical Recipes	2^{32}	1664525	1013904223
Borland C/C++	2^{32}	22695477	1
GNU Compiler Collection	2^{32}	69069	5
ANSI C	2^{32}	1103515245	12345
Borland Delphi, Virtual Pascal	2^{32}	134775813	1
Microsoft Visual/Quick C/C++	2^{32}	214013	2531011
MINSTD	$2^{31}-1$	16807	0
RANDU	2^{31}	65539	0
SIMSCRIPT	$2^{31}-1$	630360016	0
BCSLIB	2^{35}	30517578125	7261067085
BCPL	2^{32}	2147001325	715136305
URN12	2^{31}	452807053	0
APPLE	2^{35}	1220703125	0
Super-Duper	2^{32}	69069	0
FISH	$2^{31}-1$	950706376	0
SIMULA	2^{35}	30517578125	0
NAG	2^{59}	302875106592253	0
DRAND 48	2^{48}	25214903917	11
CRAY	2^{48}	44485709377909	0
MAPLE	$10^{12}-11$	427619669081	0
DERIVE	2^{32}	3141592653	1
CRAND	2^{32}	663608941	0

8.8. Generator kwadratowy BBS

8.8.1. Historia powstania algorytmu BBS

Generator liniowy opisany w poprzednim rozdziale charakteryzuje się liniową zależnością kolejnych wyrazów ciągu pseudolosowego. W związku z tym nie jest możliwe jego stosowanie np. w kryptografii. Ponieważ od bezpieczeństwa szyfru zależą często ludzkie pieniądze, konieczne było zastosowanie takiego generatora liczb pseudolosowych, który zapewniłby (a przynajmniej dotychczas zapewnia) niemożliwość odkrycia algorytmu generującego kod. Jednocześnie w zastosowaniach kryptograficznych, w przeciwieństwie do symulacyjnych, szybkość działania nie ma tak istotnego znaczenia. Wszystkie te cechy spełnia generator kwadratowy, a dokładniej rzecz ujmując generator reszt kwadratowych BBS. Skrót ten wziął się od nazwisk trójki jego wynalazców: Manuela Blum'a, jego żony Lenore Blum i Michaela Shub'a. W roku 1986 zaproponowali algorytm opierający się na fakcie, iż liczba pseudolosowa jest generowana na podstawie swojej poprzed-

niczki, jako kwadrat poprzedniej liczby modulo pewna stała M. Siłą generatora BBS jest odpowiednie wybranie właśnie tej stałej M.

Manuel Blum (zdjęcie obok, źródło: (źródło: http://www.cylab.cmu.edu/images/faculty/blum_manuel.jpg) - wenezuelski informatyk, absolwent Massachusetts Institute of Technology, zajmujący się naukowo zagadnieniami kryptograficznymi. W 1995 roku otrzymał nagrodę Turinga za wkład w rozwój teorii złożoności obliczeniowej oraz jej zastosowań w kryptografii i weryfikacji formalnej. Przedstawił również ważne twierdzenie dotyczące złożoności funkcji obliczalnych. Jest również twórcą aksjomatu Bluma.

8.8.2. Charakterystyka algorytmu

Zgodnie z tym, co zostało wcześniej zaznaczone, generator kwadratowy BBS pozwala na wyliczenie kolejnej cyfry ciągu pseudolosowego na podstawie poniższego wzoru:

$$x_i = x_{i-1}^2 \ mod \ M \tag{8.8}$$

Kluczowe w algorytmie jest wybranie odpowiedniego M oraz punktu startu algorytmu. Odpowiedniego, czyli takiego, który zapewni maksymalny okres generatora. Oznacza to, iż ciąg pseudolosowy wygenerowany przez taki algorytm będzie miał maksymalną długość. W związku z tym najpierw wybierane są dwie liczby pierwsze p i q, które są kongruentne względem 3 modulo 4. Iloczyn tych liczb daje tzw. liczbę całkowitą Bluma, czyli $M = p \cdot q$. Mając ten składnik generatora, należy wybrać inną losową liczbę całkowitą x, względnie pierwszą z M. Następnie należy obliczyć:

$$x_0 = x^2 \ mod \ M \tag{8.9}$$

gdzie x_0 jest wartością początkową generatora. Pseudokod algorytmu pokazano na rys. 8.3.

```
n := pq
Generuj pseudolosową liczbę s
x := (s*s) mod n
for i = 0, ... do
  x := (x*x) mod n
  b_i := x mod 2
  Output bit b_i
end for
```

Rys. 8.3: Pseudokod algorytmu Blum Blum Shub.

Aby zrozumieć istotę i wagę założeń oraz wytłumaczyć, dlaczego algorytm właściwie nazywany jest generatorem reszt kwadratowych, warto przeanalizować

przykład przedstawiony w tab. 8.2. Na tej podstawie można wysnuć następujące wnioski:

- algorytm BBS nie generuje wszystkich liczb z założonego zakresu, dlatego do ciągu pseudolosowego wybierane są tylko jedności wygenerowanych liczb (stąd nazwa algorytmu: generator reszt kwadratowych); w założeniu do ciągu losowego brany był ostatni bit, ale oczywiście może to być, jak w powyższym przykładzie, jedna cyfra bądź więcej liczb - w zależności od wielkości wyniku,
- wyraźnie widać, że dla podanych warunków początkowych algorytm generuje dość krótki ciąg - potrzebne jest wybranie zdecydowanie większych liczb pierwszych,
- co ważne, na podstawie kolejnych wyrazów ciągu nie da się określić, na podstawie jakich parametrów został stworzony; nie ma wad algorytmu liniowego (nie występuje efekt Marsaglii), stąd jego zastosowanie w kryptografii.

Tab. 8.2: Kolejne kroki algorytm BBS (wzór 8.9) dla następujących wartości współczynników: $X_0 = 5,\ M = 121$ ($p = 11,\ q = 11$)

$X_0 =$	s^2	25	5
$X_1 =$	$25^2\ mod\ 121 =$	20	0
$X_2 =$	$20^2\ mod\ 121 =$	37	7
$X_3 =$	$37^2\ mod\ 121 =$	38	8
$X_4 =$	$38^2\ mod\ 121 =$	113	3
$X_5 =$	$113^2\ mod\ 121 =$	64	4
$X_6 =$	$64^2\ mod\ 121 =$	103	3
$X_7 =$	$103^2\ mod\ 121 =$	82	2
$X_8 =$	$82^2\ mod\ 121 =$	69	9
$X_9 =$	$69^2\ mod\ 121 =$	42	2
$X_{10} =$	$42^2\ mod\ 121 =$	70	0
$X_{11} =$	$70^2\ mod\ 121 =$	60	0
$X_{12} =$	$91^2\ mod\ 121 =$	91	1
$X_{13} =$	$25^2\ mod\ 121 =$	53	3
$X_{14} =$	$53^2\ mod\ 121 =$	26	6
$X_{15} =$	$26^2\ mod\ 121 =$	71	1
$X_{16} =$	$71^2\ mod\ 121 =$	80	0
$X_{17} =$	$80^2\ mod\ 121 =$	108	8
$X_{18} =$	$108^2\ mod\ 121 =$	48	8
$X_{19} =$	$48^2\ mod\ 121 =$	5	5
$X_{20} =$	$5^2\ mod\ 121 =$	25	5

8.8.3. Warunki bezpieczeństwa algorytmu

Po krótkim wprowadzeniu w działanie algorytmu ważne jest, aby dobrze zrozumieć działanie generatora i dowieść jego bezpieczeństwo. Należy więc wprowadzić pojęcia i definicje związane z resztami kwadratowym [7].

Definicja 1. Liczba całkowita $x \in \mathbb{Z}_n^*$ jest nazywana resztą kwadratową modulo n, jeśli istnieje y takie, że $y^2 \ mod \ n = x$. W przeciwnym wypadku, x jest resztą nie-kwadratową modulo n. Niech $Q\ R_n$ oznacza zbiór reszt kwadratowych modulo n, a zbiór reszt nie-kwadratowych modulo n zostanie oznaczone jako $Q\ N\ R_n$.

Następne twierdzenie mówi, jak $\mathbb{Z}_n^*$ może zostać podzielone na pewne ważne podgrupy.

Twierdzenie 2. Niech $n = pq$ będzie wynikiem mnożenia dwóch różnych liczb pierwszych oraz niech $\mathbb{Z}_n^*(+1)$ oznacza liczby w $\mathbb{Z}_n^*$ z symbolem Jacobiego 1 i $\mathbb{Z}_n^*(-1)$ - liczby z symbolem Jacobiego -1. Wtedy połowa elementów $\mathbb{Z}_n^*$ jest w $\mathbb{Z}_n^*(+1)$, a druga połowa w $\mathbb{Z}_n^*(-1)$. Połowa elementów z $\mathbb{Z}_n^*(+1)$ i żaden element z $\mathbb{Z}_n^*(-1)$ są resztami kwadratowymi modulo, więc $Q\ R_n \subset \mathbb{Z}_n^*(+1)$.

Teraz dla x wybranego jednolicie z $\mathbb{Z}_n^*(-1)$ prawdopodobieństwo, że $x \in QR_n$ wynosi $\frac{1}{2}$. Problem występowania kwadratowych reszt modulo polega na zdecydowaniu, czy takie x jest resztą kwadratową. Zakłada się, że jest to zadanie trudne.

Założenie 3. Każdy probabilistyczny algorytm P decydujący o reszcie kwadratowej modulo $x \in \mathbb{Z}_n^*(+1)$, gdzie $n = pq$ jest iloczynem dwóch różnych liczb pierwszych, będzie miał prawdopodobieństwo sukcesu najwyżej $\frac{1}{2} + \varepsilon$, gdzie ε jest nieistotne w porównaniu do długości bitów n.

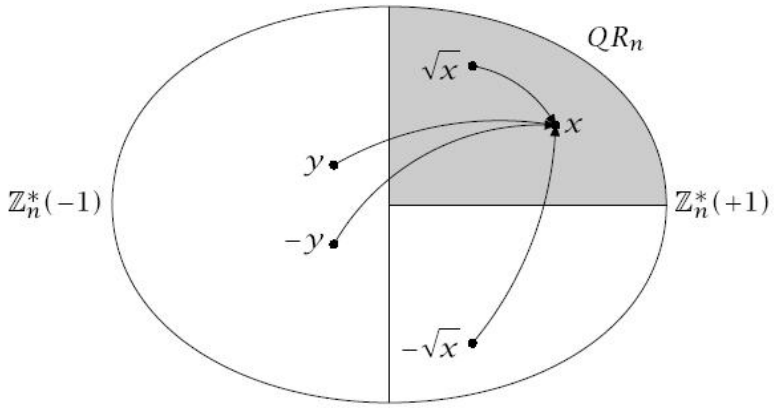
To założenie jest często nazywane Założeniem Reszt Kwadratowych Modulo (ang. *QRA*).

Definicja 4. Liczba pierwsza p jest nazywana liczbą pierwszą Bluma, jeśli $p \equiv 3\ (mod\ 4)$.

Aby udowodnić, że algorytm BBS jest bezpieczny, warto spojrzeć na kolejny problem, którego dowód nie zostanie tutaj przedstawiony.

Twierdzenie 5. Niech $n = pq$ będzie wynikiem mnożenia dwóch różnych liczb Bluma. Wtedy każda reszta kwadratowa x modulo n ma cztery różne pierwiastki kwadratowe. Dokładnie jeden z nich jest również resztą kwadratową modulo i niech $\sqrt{x}$ oznacza ten unikalny pierwiastek.

Na rys. 8.4 przedstawiono cztery pierwiastki x oraz ich lokalizację w $\mathbb{Z}_n^*$. Powyższe twierdzenie zwraca uwagę na pytanie: mając $x \in QR_n$, co jest analogią dla



Rys. 8.4: Symboliczna mapa $\mathbb{Z}_n^*$.

unikalnego pierwiastka $\sqrt{x}$. Korzystając z tego problemu można pokazać podstawową redukcję świadczącą o bezpieczeństwie BBS.

Załóżmy posiadanie algorytmu A, który otrzymując na wejściu sekwencję algorytmu BBS: $b_0, ..., b_i$, daje na wyjściu b_{-1} z pewnym prawdopodobieństwem. Innymi słowy, przewiduje sekwencję z lewej. Mając resztę kwadratową modulo x, można wygenerować sekwencję $b_0, ..., b_i$ jak w algorytmie i podać ją do A. Wtedy A znajdzie b_{-1} = parzystość($\sqrt{x}$) z takim samym prawdopodobieństwem sukcesu jak wcześniej. Niech ten nowy algorytm nazywa się A'. Teraz, korzystając z przedstawionego dalej twierdzenia 11, można zbudować algorytm B decydujący o kwadratowej parzystości modulo z takim samym prawdopodobieństwem sukcesu, oraz wiadomo, że B działa w czasie wielomianowym, jeśli A' również. Oznacza to, iż jeśli algorytm A działający w czasie wielomianowym może przewidzieć sekwencję z prawdopodobieństwem $\frac{1}{2}+\delta$, gdzie δ jest niezerowa, to istnieje algorytm B decydujący o reszcie kwadratowej modulo z prawdopodobieństwem $\frac{1}{2}+\delta$, co zaprzecza QRA. Dowodzi to, że BBS jest kryptograficznie slinym generatorem liczb pseudolosowych.

Należy teraz dowieść redukcji z problemu decydowania o parzystości $\sqrt{x}$ do problemu decydowania o resztach kwadratowych modulo. Aby to zrobić, konieczne jest wprowadzenie paru pojęć. Poniższy lemat opiera się na izomorfizmie dowiedzionym przez definicję „Chinese Remainder".

Lemat 6. Niech $n = pq$ będzie wynikiem mnożenia dwóch różnych liczb pierwszych, wtedy

$$x \in QR_n \Leftrightarrow x \; mod \; p \in QR_p \wedge x \; mod \; q \in QR_p$$

Lemat 7. Mając liczbę pierwszą p,

$$p \equiv 3 (mod \; 4) \wedge -1 \in QNR_p$$

Lemat 8. Niech $n = pq$ będzie wynikiem mnożenia dwóch różnych liczb Bluma. Wtedy x oraz $-x$ mają ten sam symbol Jacobiego (dowód pozostawiono czytelnikowi). Zostanie teraz udowodnione, że cztery pierwiastki x na symbolicznej mapie (rys. 8.4) zostały rozmieszczone właściwie.

Lemat 9. Funkcja $x \to x^2$ jest funkcją 2-1 na $\mathbb{Z}_n^*(+1)$ gdy $n = pq$ jest wynikiem mnożenia dwóch liczb Bluma.

Lemat 10. Niech $n = pq$ gdzie p i q są liczbami pierwszymi Bluma. Dla wszystkich $x \in \mathbb{Z}_n^*(+1)$, które

$$x \in QR_n \Leftrightarrow \text{parzystość}(x) = \text{parzystość}(\sqrt{x^2}).$$

Znalezienie parzystości $\sqrt{x}$ jest tak samo trudne jak decydowanie o reszcie kwadratowej modulo. Mówi o tym następujące twierdzenie:

Twierdzenie 11. Mając algorytm A znajdujący parzystość $\sqrt{x}$ można skonstruować inny algorytm B, który będzie decydował o reszcie kwadratowej modulo x. Oba algorytmy mają jednakowe pradopodobieństwo sukcesu.

Korzystając z tych twierdzeń można pokazać, że algorytm BBS jest algorytmem bezpiecznym.

8.9. Testowanie algorytmów pseudolosowych

8.9.1. Cele testowania algorytmów

Od algorytmów pseudolosowych wymaga się gwarancji, że wygenerowane wg nich liczby mają rozkład jak najbardziej zbliżony do losowego. W testowaniu generatorów dużo ważniejsze jest , aby nie zaakceptować złego generatora, niż odrzucić generator poprawny. Jest to oczywiste ze względów związanych z zastosowaniem generatorów liczb pseudolosowych. Obecnie istnieje wiele testów poprawności działania algorytmów. Podstawowe cechy, którymi powinny się cechować dobre generatory, to [1]:

1. *Jednorodność* - w każdym punkcie generowanego ciągu prawdopodobieństwo wystąpienia zera bądź jedynki jest takie samo, równe $\frac{1}{2}$.
2. *Skalowalność* -każdy podciąg ciągu, który pozytywnie przeszedł dany test, także powinien uzyskać wynik pozytywny.
3. *Zgodność* - zachowanie generatora musi dawać podobne rezultaty niezależnie od początkowej wartości.

Każdy z testów sprawdza zachowanie w pewnym środowisku, czyli testuje wystąpienie pewnej negatywnej cechy. Dlatego, aby w ogóle mówić o jakości generatora, należy wykonać testy przy użyciu pakietu lub specjalnego testu pod konkretne zastosowanie (np. obliczenia numeryczne, kryptografia, gry losowe) [1].

8.9.2. Metody analizy wyników

Poniżej zostanie przedstawionych kilka testów pochodzących z pakietu DieHarder, który jest odpowiednikiem starszego i uznanego za jeden z najlepszych pakietu DieHard. Poza testami, stosuje się również analizę statystyczną. Celem jest otrzymanie wartości mówiącej, z jakim prawdopodobieństwem mamy do czynienia z dobrym generatorem.

Test Kolmogorova-Smirnova Test ten polega na znalezieniu maksymalnej wartości odchylenia wzorcowego rozkładu od wyników uzyskanych z doświadczenia:

$$D = max|K(a_i) - F(a_i)|$$

dla każdego i, dla którego istnieje wynik próby. Wartość λ obliczana jest jako iloczyn D i pierwiastka liczebności, następnie porównywana z λ krytyczną i wynik pozwala nam stwierdzić zgodność rozkładów.

Test χ^2 W przypadku, gdy sam test wymaga badania zgodności rozkładów najczęściej stosuje się test χ^2, dany wzorem:

$$\chi^2 = \sum_{i=1}^{n} (\frac{O_i - E_i}{\sigma_i})^2$$

gdzie O_i to wartość doświadczalna, E_i to wartość wzorcowego rozkładu, σ_i to odchylenie standardowe, n to ilość pomiarów.

Birthdays test Test urodzin wykorzystuje „paradoks urodzin", czyli szansę, że w grupie osób znajdą się osoby, które mają urodziny tego samego dnia. Dla potrzeb testu grupa liczy 512 „osób", a dzień roku określa 24-bitowa liczba. Zliczane są odległości pomiędzy datami urodzin, dla dobrego generatora rozkład powinien być podobny do rozkładu Poissona.

Overlapping 5-Permutations test Test działa na milionie 32-bitowych liczb. Ciąg jest dzielony na 5-elementowe, pokrywające się podciągi, każdy z nich może być w jendym z 120 stanów wybranych pod względem kolejności liczb w ciągu, wystąpienia poszczególnych stanów są zliczane. Potem wyniki są statystycznie porównywane do danych wzorcowych otrzymanych z rozkładu normalnego.

32×32 Binary Rank test Test operuje na macierzach o wymiarach 31×31. Z 31 wylosowanych liczb 32-bitowych wybierane są pierwsze bajty i macierz zostaje zapełniona poszczególnymi bitami. Następnie liczony jest rząd macierzy i sumuje ich wystąpienia. Następnie wykonuje się test χ^2 w odniesieniu do generatora idealnego.

6×8 Binary Rank test Analogiczny do poprzedniego, główna różnica to rozmiar macierzy.

Bitstream test Test analizuje ciąg liczb losowych jako ciąg znaków o alfabecie 0 i 1, ciąg dzieli się na dwudziestoliterowe pokrywające się słowa. Test liczy ilość słów, które nie wysąpiły w ciągu składającym się z 2^{21} pokrywających się słów, przy istniejących 2^{20} możliwych kombinacjach tworzących słowa.

Overlapping Pairs Sparse Occupance test Test rozpartuje dwuliterowe słowa z alfabetu o 1024 literach, każda litera jest wyznaczana z dziesięciu bitów wybranych z 32-bitowej liczby losowej z testowanego ciągu. Obliczana jest liczba brakujących możliwych słów.

Overlapping Quadruples Sparce Occupancy test Podobnie jak poprzednio, ale zmienia się liczba liter w słowie (na 4) i liczba liter (na 32).

DNA test Test operuje na alfabecie 4-literowym: C, G, A, T (stąd analogia do DNA), litera jest ustalana na podstawie dwóch bitów z testowanego ciągu, każde słowo ma 10 liter. Podobnie jak poprzednio, obliczane sa słowa, które nie wystąpiły.

Count the 1s (stream) test Kolejny test operujący na słowach i alfabecie, ale inaczej tworzonych. Traktuje ciąg liczb jako strumień bajtów, z których każdemu przyporządkowuje literę w zależności od tego, ile dany bajt zawiera jedynek.

Jest to tylko niewielka część najbardziej znanych algorytmów testowania generatorów liczb pseudolosowych. Należy dodać, iż nigdy nie ma gwarancji, że dla algorytmu, który poprawnie przeszedł n testów, test $n+1$ również da pozytywny rezultat. Wynika stąd, że testując algorytmy celem nie jest dowieść ich poprawności, ale pokazać, że nie są niepoprawne.

8.10. Przykłady implementacji

Obecnie liczby pseudolosowe mają bardzo szerokie zastosowanie, głównie jeśli chodzi o zagadnienia związane z ochroną danych oraz szyfrowaniem. Od czasów słynnych polskich matematyków, którzy rozszyfrowali Enigmę, postęp w rozwoju kryptografii sprawił, że trwa obecnie wojna na „największą liczbę pierwszą". To właśnie sposoby generowania ciągów liczb tak, by były one jak najbardziej zbliżone do losowych są obecnie zadaniem postawionym przed kryptografami. Liczy się więc przede wszystkim moc obliczeniowa, gdyż algorytmy używane w szyfrowaniu danych mają matematyczne dowody na niedeszyfrowalność. Oczywiście nie jest to jedyne wykorzystanie liczb pseudolosowych, co zostało wytłuszczone wcześniej. W tej publikacji postarano się przytoczyć dwa przykłady zastosowań generatorów przedstawionych w poprzednich podrozdziałach. Pierwszy z nich posłuży za wizualizację działania algorytmów: liniowego oraz kwadratowego BBS.

8.10.1. Wizualizacja

Wyświetlacz graficzny Siemens S65 Do wizualizacji wykorzystano wyświetlacz graficzny Siemens S65. Wyświetlacz posiada rozdzielczość 132 na 176 piksli i 16-bitową skalę kolorów. Jego wymiary to 38,20 na 55,80 mm. Jest on wykorzystywany w telefonach komórkowych Siemens S65, M65, CX65, SK65 i Nokia 6100. Z kontrolerem sterującym komunikuje się za pośrednictwem protokołu SPI. Szczegóły dotyczące komunikacji z wyświetlaczem opisane zostały w [8].

Warto zwrócić uwagę na porównanie czasów działania algorytmów. Liniowy generator jest zdecydowanie szybszy od kwadratowego generatora BBS, co jest zgodne z teorią. Jak już wspomniano w podrozdziale dotyczącym generatora BBS, nie generuje on wszystkich liczb z zakresu, dlatego losowanie sprowadza się do wyznaczenia liczby modulo 2. W związku z tym aby wygenerować liczbę 16–bitową, potrzebną w procesie wyświetlania piksela na LCD, należy taką operację przeprowadzić 16 razy. Tym samym można powiedzieć, że generator BBS powinien być w tym przypadku około 16 razy wolniejszy od generatora liniowego.

8.10.2. Szyfrowanie XOR

Innym zastosowaniem może być użycie generatora BBS jako hasła dla prostego algorytmu szyfrującego XOR. Za platformę wizualizacyjną działania algorytmu mogą posłużyć proste moduły transmisji bezprzewodowej podłączone do komputerów PC. Przedmiotem testu jest sprawdzenie, czy wiadomość wysłana drogą radiową, uprzednio zaszyfrowana, zostanie poprawnie odszyfrowana, przy założeniu takiego samego algorytmu szyfrującego i deszyfrującego (co sprowadza się do ustalenia takich samych parametrów dla algorytmu BBS).

Algorytm XOR Ideą algorytmu jest zastosowanie operacji bitowej XOR. Aby zrozumieć jej działanie, należy prześledzić w poniższej tab. 8.3:

Tab. 8.3: Tabela XOR dla wejściowych A i B.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

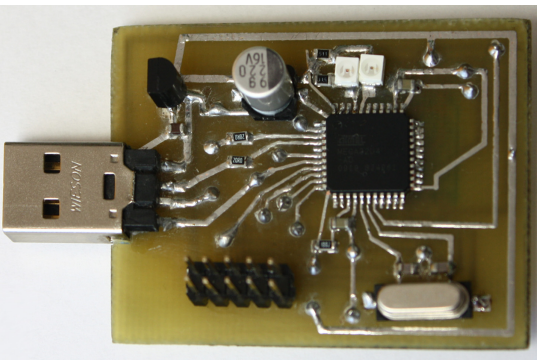
Wyraźnie widać, że funkcja XOR zwraca 1, gdy parametry wejściowe są różne, natomiast 0, gdy są takie same. Co jest najistotniejsze dla działania algorytmu, $(A\ XOR\ B)XOR\ B = A$, więc w łatwy sposób można odszyfrować tekst, znając hasło. Widać również, że aby tekst mógł zostać zaszyfrowany, musi najpierw zostać zapisany w postaci binarnej.

Aby szyfr XOR mógł być uważany za bezpieczny, należy zastosować się do trzech reguł:

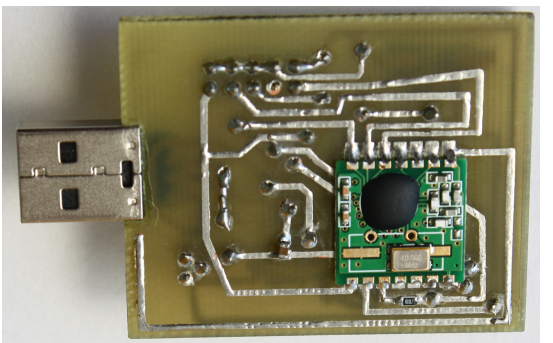
- hasło musi mieć długość co najmniej taką, jak tekst do zaszyfrowania,
- hasło musi być ciągiem pseudolosowym zer i jedynek,
- hasło musi być jednorazowe.

Moduły bezprzewodowe Przedstawiony powyżej algorytm został zaimplementowany w bezprzewodowych modułach podłączanych do USB, wykonanych przez studentów Robotyki (przedstawione na rys. 8.5).

a)



b)

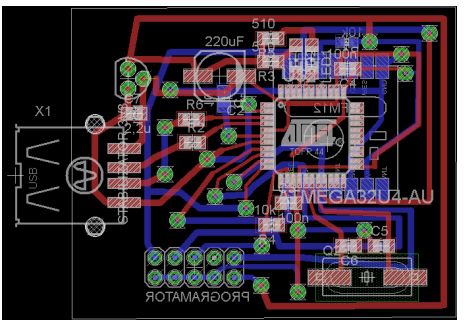


Rys. 8.5: Widok płytki z RFM12b oraz mikroprocesorem ATmega32u4: a) góra b) dół.

Mikrokontroler Jednostką centralną modułu jest mikrokontroler Atmega32u4 z rodziny AVR. Jest jeden z najnowszych mikrokontrolerów w tej rodzinie, wyróżniający się przede wszystkim posiadaniem wbudowanego sprzętowego kontrolera USB i zastosowaniem pętli PLL (mnożenia częstotliwości) dzięki czemu przy taktowaniu 8/16 MHz układ może obsługiwać magistralę USB 2.0 w trybie Full-Speed. Pełna dokumentacja mikrokontrolera jest dostępna pod adresem [9].

Moduł radiowy Zdecydowano się na zastosowanie modułu RFM12b firmy HOPE RF. Jest to tani moduł ISM ogólnego zastosowania pracujący z częstotliwością 868MHz. Pełna dokumentacja jest dostępna pod adresem [10].

Schemat montażowy Zaprojektowano i wykonano płytkę (rys. 8.6) w technologii montażu powierzchniowego, dzięki czemu uzyskano dość kompaktowe wymiary urządzenia.



Rys. 8.6: Schemat montażowy układu.

Komunikacja poprzez USB Do komunikacji poprzez USB została użyta biblioteka LUFA.

8.11. Podsumowanie

Zagadnienia związane z komputerową pseudolosowością są obecnie jednymi z prężniej rozwijających się. Ma to związek zarówno z rozwojem informatyzacji (zastosowanie liczb pseudolosowych w grach, specjalistycznych programach) jak i zabezpieczeń (generowanie szyfrów). Warto jednak zwrócić uwagę na możliwości zastosowań poszczególnych algorytmów. Wyraźnie widać to, porównując działanie algorytmu liniowego oraz BBS w dwóch różnych zastosowaniach.

Algorytm liniowy, jako bezsprzecznie szybszy od kwadratowego znajduje zastosowanie w prostych modelach korzystających z pseudolosowości, jak chociażby podświetlanie pikseli na wyświetlaczu LCD. W tym przypadku używanie algorytmu kwadratowego jest nieprzydatne ze względu na to, że wygenerowanie 16-bitowej liczby zajmuje mu 16 razy więcej czasu - związane jest to z faktem, że generuje on tylko 0 lub 1. Poza tym posiada zdecydowanie krótszy okres, co wiąże się z koniecznością zmian warunków działania algorytmu BBS po pewnym czasie.

Jeśli natomiast weźmie się pod uwagę zagadnienia szyfrowania, to w powyższym porównaniu generator kwadratowy okazuje się bezkonkurencyjny. Zaimplementowany przy jego pomocy szyfr XOR, pod warunkiem spełnienia pewnych założeń, jest nie do złamania.

Warto jednak pamiętać, że algorytm, który w tej chwili daje bezpieczeństwo, już niedługo może okazać się zupełnie bezużyteczny.

Literatura

[1] M. Pawlik: *Metody oceny jakości generatorów liczb losowych* Wydział Fizyki i Informatyki Stosowanej, Kraków
[2] Z. Kotulski: *Generatory liczb losowych: algorytmy, testowanie, zastosowania* Matematyka Stosowana 2, Warszawa, (2001).
[3] W. Płaczek: *Matematyczna ruletka, czyli jak się robi liczby (pseudo) losowe*
[4] J. Jarnicki: Komputerowe generatory liczb losowych.
[5] J. Wałaszek: Algorytmy wyszukujące - Liniowe generatory liczb pseudolosowych. http://edu.i-lo.tarnow.pl/inf/alg/001_search/0019.php, (2010).
[6] T. Lubiński: Generator LCG (Liniowy Generator Kongruentny). http://www.algorytm.org/liczby-pseudolosowe/generator-lcg-liniowy-generator-kongruentny.html, (2010).
[7] M. Geisler: *About Random Bits* Department of Computer Science - Daimi, Daimi, (2004).
[8] C. Kranz: *Using the Siemens S35 - Display* Benjamin Metz, Daimi, (2005).
[9] A. Corporation: Atmel ATmega32u4. http://www.atmel.com/dyn/resources/prod_documents/doc7766.pdf, (2010).
[10] H. RF: Universal ISM Band FSK Transciever Module - RFM12b. http://www.pliki.jm.pl/karty/RFM12B.pdf, (2006).

ROZDZIAŁ 9

NIEPEWNOŚĆ W METODACH LOKALIZACJI ROBOTA

M. Opałka, Ł. Kucharczyk

Algorytmy sterowania, systemy decyzyjne czy tzw. *problem solvers* znajdują zastosowanie w wielu dziedzinach techniki, pozwalając na automatyzację dowolnych procesów. Poprawność działania wymienionych narzędzi w dużej mierze zależy od jakości informacji dostarczonej przez użytkownika, wyników pomiarowów bądź sporządzonego modelu świata. Niestety, dane te zazwyczaj obarczone są błędem: luką w informacji, pomyłką podczas wprowadzania, niepewnością pomiarową urządzenia czy uproszczeniem praw fizyki na etapie modelowania.

Niektóre niepewności można pominąć ze względu na ich niewielki wpływ na wynik końcowy pracy systemu (tzw. algorytmy odporne), niektóre natomiast są na tyle istotne, że ich wystąpienie uniemożliwia działanie algorytmu. Konieczne jest zastosowanie metody pozwalającej na uwzględnienie wpływu niepewności, bądź zredukowanie jej rozmiaru. Propozycje rozwiązania tego problemu pojawiały się w literaturze bardzo często – głównie ze względu na praktyczny aspekt zagadnienia. W pozycji [1] przedstawiono matematyczne podstawy reprezentacji informacji niepewnej i niepełnej. W pracy [2] przedstawiono przykład zastosowania modelu sensorów (obarczonych niepewnością pomiarową) w rozwiązywaniu klasycznego problemu w reprezentacji STRIPS przy niepełnym opisie świata. Książka [3] zawiera szczegółowy opis wielu metod planowania działań przy niepewnościach pomiarowych – w tym matematyczne modele czujników wraz z analizą ich przydatności. Pozycja [4] przedstawia zastosowanie statystycznych metod w problemach robotycznych.

W niniejszym rodziale skupiono się metodach lokalizacji mobilnego robota w środowisku z niepewną informacją. W podrozdziale 9.1 przedstawiono podstawy rachunku prawdopodobieństwa, wraz z wyjaśnieniem oznaczeń stosowanych dalej. Następnie w podrozdziale 9.2 omówiono kilka popularnych metod lokalizacji robota przy niepownościach pomiarowych. W podrozdziale 9.3 przedstawiono przykładową impelmentację algorytmu filtracji Kalmana.

9.1. Podstawowe pojęcia z rachunku prawdopodobieństwa

Większość metod reprezentacji informacji niepewnej, czy niepełnej, opiera się na rachunku prawdopodobieństwa. Poniżej przedstawiono podstawowe pojęcia z tej dziedziny potrzebne do prawidłowego zrozumienia dalszej części tekstu, [4].

Zmienne losowe przyjęto oznaczać dużymi literami, np. X, zaś wartości jakie przymuje - literami małymi (np. x). Prawdopodobieństwo, że zmienna losowa X przyjmuje wartość x oznacza się przez

$$p(X = x) \tag{9.1}$$

Funkcja prawdopodobieństwa $P : X \to \mathbb{R}^1$ spełnia

$$\forall (x \in \mathbb{X}) \quad p(x) \geq 0, \tag{9.2}$$

$$p(\Omega) = 1, \tag{9.3}$$

$$\forall (x_1, x_2 \in X | x_1 \cap x_2 = \emptyset) \quad P(x_1 \cup x_2) = P(x_1) + P(x_2) \tag{9.4}$$

Prawdopodobieństwo wystąpienia jakiegokolwiek zdarzenia ze zbioru zadrzeń elementarnych Ω wynosi 1. Dla prawdopodobieństw o rozkładzie dyskretnym zachodzi:

$$\sum_{x \in \Omega} p(X = x) = 1, \tag{9.5}$$

zaś dla rozkładów ciągłych

$$\int_{x \in \Omega} p(x) dx = 1. \tag{9.6}$$

Prawdopodobieństwo łączne zajścia kilku zdarzeń oznaczane jest przez:

$$p(x, y) = p(X = x \text{ i } Y = y), \tag{9.7}$$

w przypadku gdy zdarzenia X oraz Y są niezależne zachodzi

$$p(x, y) = p(x) \cdot p(y). \tag{9.8}$$

Prawdopodobieństwo, że zmienna X przyjmie wartość x gdy wiemy, że zmienna Y ma wartość y, określa prawdopodobieństwo warunkowe

$$p(x|y) = p(X = x|Y = y) \tag{9.9}$$

wyrażane wzorem

$$p(x|y) = \frac{p(x, y)}{p(y)} \qquad \text{dla} \quad p(y) > 0. \tag{9.10}$$

Gdy zdarzenia X oraz Y są niezależne, prawdopodobieństwo to można zapisać zgodnie z (9.8)

$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x). \tag{9.11}$$

Prawdopodobieństwo warunkowe jest związane z prawopodobieństwo całkowitym, co wyrażane jest wzorem:

$$p(x) = \sum_y p(x|y)p(y) \tag{9.12}$$

$$p(x) = \int p(x|y)p(y)dy. \tag{9.13}$$

Podstawą wnioskowania oraz większości metod opisywanych w dalszej części rozdziału jest reguła Bayes'a, pozwalająca wyrazić prawdopodobieństwo warunkowe w następujący sposób:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{9.14}$$

Zależność ta może być wykorzystywana do aktualizowania wartości wierzeń (np. aktualnie estymowanej wartości stanu) po pojawieniu się np. nowej informacji z czujników.

Zwykle, w prezentowanych metodach przyjmuje się rozkład normalny zmiennej losowej

$$p(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{9.15}$$

a dla rozkładów wielowymiarowych

$$p(x) = \det(2\pi\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{9.16}$$

gdzie Σ oznacza dodatnio półokreśloną macierz kowariancji.

Wartość oczekiwana zmiennej losowej X jest wyznaczana wg wzoru:

$$E[X] = \int xp(x)dx = \mu. \tag{9.17}$$

Momentem n-tego rzędu dla zmiennej losowej X nazywa się wartość

$$\mu_n = E[X^n]. \tag{9.18}$$

Wariancją zmiennej losowej X jest zmienna losowa

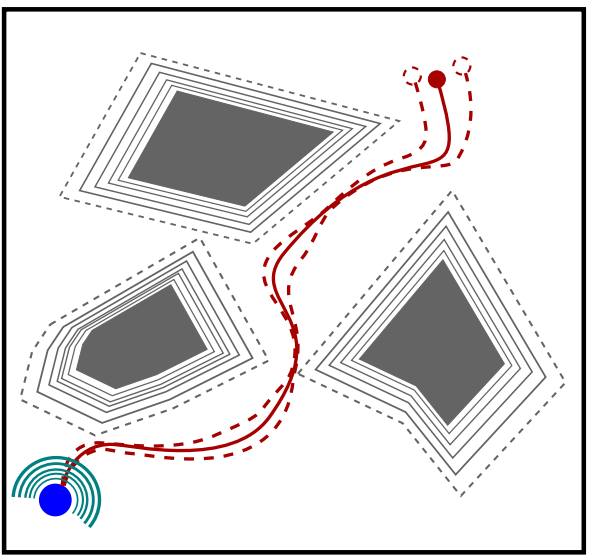
$$\sigma^2(X) = E[(X-\mu)^2] = E[X^2] - E[X]^2. \tag{9.19}$$

Natomiast kowariancją dwóch zmiennych losowych X i Y nazywamy zmienną losową

$$cov(X,Y) = E\left[(X - E[X]) \cdot (Y - E[Y])\right] = E[X \cdot Y] - E[X] \cdot E[Y]. \tag{9.20}$$

9.2. Metody lokalizacji robota

Jednym z podstawowych problemów robotyki jest lokalizacja. W problemie tym robot (mobilny), wyposażony w mapę terenu (pomieszczenia), musi przy pomocy dostępnych czujników określić swoje położenie, orientację. Odczyty z czujników robota obarczone są błędem, co prowadzi do niepewności w estymacji położenia i złej oceny otoczenia. Na przykład przy stosowaniu czujników ultradźwiękowych pojawiają się odbicia fali – prowadzi to do powstawania sztucznych przeszkód (odbitych obrazów przeszkód rzeczywistych).



Rys. 9.1: Przykładowy obraz sceny na podstawie informacji z czujników.

W literaturze wyróżnia się wiele wariantów problemu lokalizacji. Poniżej została przedstawiona zbiorcza klasyfikacja. Lokalizacja robota może być zarówno aktywna jak i pasywna [3].

Lokalizacja pasywna pozwala robotowi na estymację położenia i orientacji na podstawie informacji z czujników oraz na podstawie przewidywania efektów podjętych działań.

Lokalizacja aktywna wymaga od robota dokonywania działań mających na celu zmniejszenie niepewności lokalizacji – np. eksploracja terenu w celu wykrycia charakterystycznych obiektów.

Problem lokalizacji można także klasyfikować ze względu na zakres niepewności [4].

Lokalizacja lokalna jest najprostszą odmianą zagadnienia lokalizacji - śledzeniem pozycji. W chwili początkowej położenie i orientacja robota są znane

– informacja o aktualnym położeniu ustalana jest na podstawie filtrowania zakłóceń nałożonych na odczyty czujników mierzących realizację sterowań.

Lokalizacja globalna zakłada, że początkowe położenie robota nie jest znane. Zwykle w skład zadania lokalizacji globalnej wchodzi element śledzenia pozycji robota, czyli zadanie lokalizacji lokalnej.

Dodatkowo dla lokalizacji globalnej wyróżnia się problem *porwanego robota* [3], w którym zakłada się możliwość przeniesienia (teleportacji) robota w dowolne miejsce mapy w trakcie trwania zadania. Zagadnienie to jest najtrudniejsze spośród wszystkich rodzajów lokalizacji.

Dodatkowe utrudnienia w zadaniu lokalizacji mogą wynikać z doboru środowiska. Można wyróżnić dwa podstawowe rodzaje [4]: środowisko statyczne – czyli takie w którym jedynym poruszającym się obiektem jest robot; środowisko dynamiczne – w którym położenie obiektów na mapie zależy od czasu.

Dla środowiska dynamicznego zmiany mogą być deterministyczne, probablistyczne określone pewnym rozkładem, bądź całkowicie nieprzewidywalne.

9.2.1. Rodzaje czujników

W celu lokalizacji, a zatem estymacji stanu, robot musi zostać wyposażony w odpowiednie czujniki. Układ robota wraz z czujnikami określony jest przez równanie stanu oraz równanie wyjścia

$$\begin{cases} \dot{q} = f(q,t) + g(q,u), & q \in \mathbb{Q}, \quad u \in \mathbb{U} \\ y = h(q) & y \in \mathbb{Y} \end{cases} \tag{9.21}$$

gdzie $\mathbb{Q}$ jest przestrzenią stanu odpowiadającą możliwym konfiguracją robota, $\mathbb{U}$ jest przestrzenią sterowań przenoszących układ z jednego stanu do następnego, a $\mathbb{Y}$ odpowiada za przestrzeń wyjściową, w której obserwowane jest zachowanie systemu.

Równanie wyjścia odpowiada informacji uzyskanej z czujnika. Zadaniem lokalizacji jest zbudowanie funkcji obserwatora

$$\hat{q} = k(y,q), \tag{9.22}$$

która pozwoli estymować stan na podstawie informacji z czujników. W pozycji [3] wyróżniono trzy typy czujników:

1. Czujniki stanu – realizowane jako funkcja $h : \mathbb{Q} \to \mathbb{Y}$. Wynik działania jest całkowicie deterministyczny.
2. Czujniki właściwości stanu – zawierają zakłócenia $\psi(q)$, tak więc obserwacja wyjścia wyraża się zależnością $y = h(q,\psi)$.
3. Czujniki oparte na historii – ich działanie oparte jest na informacji zależnej od pewnej ilości stanów poprzednich i działań w nich podjętych, $y_k = h_k(q_1,\ldots,q_k)$.

Poniżej przedstawiono kilka przykładów czujników [3]:

Czujnik zerowy niezależnie od stanu zwraca wartość 0, tak więc jst określony przez zależność $y = h(q) = 0$. Czujnik ten nie niesie żadnej informacji o stanie – może służyć do modelowania nieaktywnych czujników, bądź czujników uszkodzonych.

Czujnik znaku pozwala poznać znak zmiennej stanu q, opisany jest wzorem $y = h(q) = \text{sgn}(q)$. Możliwe jest jedynie uzyskanie informacji na temat położenia względem $q = 0$. Czujnik tego typu używany jest między innymi w algorytmach zmodyfikowanego jakobianu dla przejścia przez osobliwości: $y = \text{sgn}(\det[J(q)])$, gdzie $J(q)$ jest jakobianem manipulatora.

Czujnik wybiórczy dostarcza informacji jedynie o pewnej części zmiennych stanu. Przykładowo, gdyby stan robota był reprezentowany jako współrzędne $q = (i, j)$ pól na prostokątnej planszy, możliwa byłaby do uzyskania jedynie informacja o jednej ze współrzędnych; $y = h(i, j) = i$. Sytuacja taka może być spowodowane istnieniem nieobserwowalnej części w przestrzeni stanu – której zmiany nie powodują zmian na wyjściu z systemu.

Czujnik z zakłóceniem pozwala jedynie na estymację stanu z dokładnością do zakłócenia ψ; $y = h(q, \psi) = q + \psi$. Zakłócenie może być zmienną losową o znanych parametrach bądź całkowicie niedeterministyczne.

Czujnik znaku z zakłóceniem analogicznie do czujnika znaku wyraża się funkcją $y = h(q, \psi) = \text{sgn}(x + \psi)$. Podobnie jak w przypadku czujnika z zakłóceniem ψ jest zmienną losową.

Czujnik z opóźnieniem pozwala poznać informację tylko przeszłych wartościach zmiennych stanu. Dla przykładu, jeśli stany będą mierzone w dyskretnych chwilach czasu: $y_i = q_{i-k}$, opóźnienie k może być stałe bądź zmienne. Czujnik może także podawać niepełne informacje o przeszłym stanie.

Czujnik parzystości pozwala uzyskać informacje o parzystości aktualnych wartości zmiennych stanu. Może być bardzo użyteczny np. w przypadku lokalizacji lokalnej robota, przy obecności zakłóceń kierunku ruchu, na planszy złożonej z pól o współrzędnych określonych jako $q = (i, j)$. Znając stan początkowy robot jest w stanie na podstawie odczytów z czujnika całkowicie wyeliminować niepewność konfiguracji.

Niektóre z wymienionych powyżej czujników pozwalają rozwiązać problem lokalizacji dość sprawnie. Niektóre natomiast cechują się niepewnością, bądź niepełnością, dostarczanych danych. W celu minimalizacji tych wad konieczna jest filtracja informacji uzyskanych z czujników oraz ich fuzja.

W celu minimalizacji błędu estymacji konfiguracji można także odpowiednio zaplanować ruch. Wykorzystując np. metody nawigacji przybrzeżnej zmniejsza się błąd wynikający z niedokładności sterowań, czy niepewność generowaną przez czujniki odometryczne.

Możliwe jest też estymowanie stanu metodą predykcji efektu sterowania. Przy znanym sterowaniu u_t i estymowanym poprzedniej konfiguracji q_{t-1} należy wyznaczyć q_t. Zadanie to można rozszerzyć o uwzględnianie niepewności samego sterowanie (np. w przypadku systemów z dryfem o nieznanej, zmiennej charakterystyce). Czujnik predykcyjny opisujemy jako

$$\hat{q} = w(q, u, \psi). \tag{9.23}$$

9.2.2. Filtr Bayes'a

Podstawowym narzędziem służącym do wnioskowania w obliczu niepewności są Filtry Bayes'a, [4]. Pozwalają one uzyskać bardziej precyzyjną informację biorąc pod uwagę nowo otrzymane informacje dodatkowe.

W problemie decyzyjnym, czy podczas lokalizacji robota, brany jest pod uwagę rozkład prawdopodobieństwa reprezentujący za tzw. *wierzenia*. Rozkład ten określa jakie jest prawdopodobieństwo, że robot jest w danej konfiguracji, przy uwzględnieniu dotychczas otrzymanych informacji sensorycznych oraz ostatnio podjętych działań.

Rozkład prawdopodobieństwa wierzeń w chwili czasu t będzie oznaczany przez $bel(q_t)$, co można wyrazić jako prawdopodobieństwo warunkowe zależne od danych sensorycznych oraz ostatnich sterowań

$$bel(q_t) = p(q_t|y_1, y_2, \ldots, y_t, u_1, u_2, \ldots, u_t) \tag{9.24}$$

W zadaniu lokalizacji w każdej chwili czasu dostępny jest rozkład wierzeń. Tak więc można przewidywać obecną konfigurację robota jeszcze przed wykonaniem pomiaru

$$\overline{bel}(q_t) = p(q_t|y_1, y_2, \ldots, y_{t-1}, u_1, u_2, \ldots, u_t), \tag{9.25}$$

Wyliczenie $bel(q_t)$ na podstawie poprzedniego rozkładu wierzeń, $\overline{bel}(q_t)$, nazywa się naniesieniem poprawki bądź uaktualnieniem informacji z czujników [4].

Filtr Bayes'a pozwala wyliczyć rozkład prawdopodobieństwa wierzeń na podstawie wyników pomiarów i wybranych sterowań. Niepewność pomiarów uwzględniana jest przy wyliczaniu wartości $bel(q_t)$ na podstawie $\overline{bel}(q_t)$.

W pierwszym kroku algorytmu wyliczane są prawdopodobieństwa osiągnięcia wszystkich konfiguracji na podstawie predykcji efektu sterowania – wyliczenie uwzględnia rozkład wierzeń z poprzedniej chwili czasu.

Drugi krok algorytmu prowadzi do zmniejszenia niepewności estymacji obecnej konfiguracji poprzez dodanie informacji pochodzącej z czujników – przejście z $\overline{bel}(q_t)$ do $bel(q_t)$.

Algorytm zapisany w pseudokodzie [4] przedstawiony został na rys. 9.2. Do poprawnego działania algorytmu konieczne jest podanie rozkładu prawdopodobieństwa wierzeń w chwili $t = 0$. Kolejne wartości wyliczane są rekurencyjne zgodnie z zasadą działania algorytmu. Matematyczny dowód poprawności działania algorytmu został przedstawiony w pozycji [4].

Bayes_filter$(bel(q_{t-1}),u_t,y_t)$

1. *for all* q_t *do*
2. $\quad \overline{bel}(q_t) = \int p(q_t|u_t,q_{t-1})bel(q_{t-1})dq_{t-1}$
3. $\quad bel(q_t) = p(y_t|q_t)\overline{bel}(q_t)$
4. *endfor*
5. *return* $bel(q_t)$

Rys. 9.2: Filtr Bayes'a.

9.2.3. Filtr Kalmana

Filtr Kalmana został szczegółowo opisany w [5] oraz w [4]. Jest jednym z tak zwanych filtrów Gaussowskich. Do poprawnego działania wymaga by wszystkie parametry losowe posiadały rozkład normalny

$$p(x) = \det(2\pi\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{9.26}$$

Aby jednoznacznie określić rozkład normalny wystarczy podać jego dwa pierwsze momenty centralne: macierz kowariancji Σ oraz średnią μ. Dodatkowo system musi należeć do klasy liniowych systemów Gaussowskich, a co za tym idzie muszą być spełnione następujące własności [4]:

1. Prawdopodobieństwo zmiany stanu $p(q_t|u_t,q_{t-1})$ musi być funkcją liniową z dodanym szumem Gaussowskim ϵ

   $$q_t = A_t q_{t-1} + B_t u_t + \epsilon_t. \tag{9.27}$$

   Dokładniejsze informacje na temat opisu liniowych systemów w przestrzeni stanu zawiera pozycja [6]. Przestrzeń stanu $\mathbb{Q}$ jest n-wymiarowa, przestrzeń sterowań $\mathbb{U}$ jest wymiaru m. Szum Gaussowski może np. odpowiadać za niepewność zmiany stanu spowodowaną poślizgiem kół robota.
   Wartość średnia szumu ϵ wynosi 0, a kowariancja R_t. Dla takich parametrów rozkład prawdopodobieństwa zmiany stanu cechuje się średnią $A_t q_{t-1} + B_t u_t$ i kowariancją R_t

   $$p(q_t|u_t,q_{t-1}) = \det(2\pi R_t)^{-1/2}\cdot \exp\left(-\frac{1}{2}(q_t - A_t q_{t-1} - B_t u_t)^T R_t^{-1}((q_t - A_t q_{t-1} - B_t u_t))\right). \tag{9.28}$$

Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, y_t$)

1. $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3. $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
4. $\mu_t = \overline{\mu}_t + K_t (y_t - C_t \overline{\mu}_t)$
5. $\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$
6. *return* μ_t, Σ_t

Rys. 9.3: Filtr Kalmana.

2. Rozkład prawdopodobieństwa czujników także musi być funkcją liniową z Gaussowskim szumem δ

$$y_t = C_t q_t + \delta_t. \tag{9.29}$$

Rozkład zmiennej δ ma średnią równą 0 i kowariancję Q_t co prowadzi do następującego rozkładu prawdopodobieństwa czujników

$$p(y_t|q_t) = \det(2\pi Q_t)^{-1/2} \cdot \exp\left(-\frac{1}{2}(y_t - C_t q_t)^T Q_t^{-1}((y_t - C_t q_t))\right). \tag{9.30}$$

3. Rozkład prawdopodobieństwa wierzeń $bel(q_0)$ w chwili początkowej także musi być rokładem normalnym

$$bel(q_0) = \det(2\pi\Sigma_0)^{-1/2} \exp\left(-\frac{1}{2}(q_0 - \mu_0)^T \Sigma_0^{-1}(q_0 - \mu_0)\right). \tag{9.31}$$

Działanie filtru Kalmana składa się z dwóch faz podstawowych:

1. Faza uaktualnienia rozkładu wierzeń przewidując wynik sterowania.
2. Faza uaktualnienia rozkładu wierzeń poprzez uwzględnienie wyniku pomiaru stanu.

Przyjęte założenia pozwalają zapewnić, że rozkład wierzeń wynikowych $bel(q_t)$ zawsze będzie Gaussowski, dla każdego t. Dowód matematyczny został przedstawiony w pozycji [4]. Ponieważ rozkłady normalne można jednoznacznie określić poprzez podanie wariancji Σ oraz wartości średniej μ, można operować tylko na tych wielkościach podczas wyliczania nowego rozkładu wierzeń w chwili następnej. Działanie algorytmu zostało przedstawione w pseudokodzie na rys. 9.3. W algorytmie dokonywana jest swego rodzaju fuzja dwóch informacji: obarczo-

nej niepewnością predykcji konfiguracji na podstawie sterowań w chwili t oraz wyniku pomiaru stanu za pomocą czujników (także cechujących się niepewnością wyniku). Na skutek działania filtru Kalmana uzyskujemy rozkład wierzeń o mniejszej wariancji, pozwalający dokładniej estymować bieżącą konfigurację q_t.

Przykład

Niech model robota poruszającego się wzdłuż jednej osi będzie opisany funkcją

$$q_t = A \cdot q_{t-1} + B \cdot u_t + \epsilon_t, \tag{9.32}$$

gdzie $A = 1$, $B = 1$, oraz normalny rozkład zmiennej ϵ_t o wartości średniej 0 i wariancji $R_t = 2$. Funkcję pomiaru czujnika można zamodelować jako

$$y_t = 1 \cdot q_t + \delta_t, \tag{9.33}$$

gdzie zmienna δ_t posiada rozkład normalny o średniej 0 i wariancji $Q_t = 1$.

Zbiór sterowań wyrazić można za pomocą $\mathbb{U} = \{-4,-3,-2,-1,0,1,2,3,4\}$, a przestrzeń konfiguracyjną $\mathbb{Q} = \mathbb{R}$. W oczywisty sposób wyniki pomiarów należą do przestrzeni $\mathbb{Y} = \mathbb{R}$.

Ponieważ zarówno wyniki pomiarów, jak i predykcja położenia po aplikacji znanego sterowania, obarczone są niepewnościami, należy estymować, możliwie jak najdokładniej, końcowe położenie robota.

Niech rozkład wierzeń co do stanu robota w chwili początkowej $t = 0$ wyrażony będzie funkcją rozkładu normalnego o średniej $\mu_o = 5$ oraz wariancji $\Sigma_0 = 3$

$$bel(q_0) = (3\sqrt{2\pi})^{-1} e^{-\frac{1}{18}(-5+q_0)^2}. \tag{9.34}$$

Niepewność położenia robota można przedstawić graficznie (rys. 9.4). Z powodu dużej wartości wariancji określenie dokładnego położenia początkowego robota jest niemożliwe. Konieczne jest wykonywananie akcji pozwalających na zmniejszenie niepewności – co prowadzi do lokalizacji aktywnej.

Dokonując pomiaru położenia (9.33) otrzymuje się rozkład wierzeń o dużo mniejszej wariancji, przedstawiony na rys. 9.4.

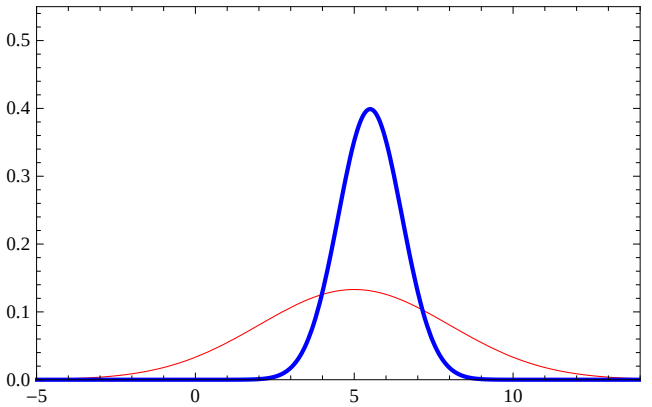
Wykonując ruch w prawo o cztery jednostki ($u_1 = +4$), przy stanie początkowym (9.34) oraz przy założonych wcześniej niepewnościach pomiarowych i predykcyjnych otrzymuje się rozkład wierzeń z predykcji

$$\overline{bel}(q_1) = (2\sqrt{2\pi})^{-1} e^{-\frac{1}{8}(-9+q_1)^2}. \tag{9.35}$$

Wykonując pomiar w stanie q_1 uzyskuje się wartość $y_1 = 8$, co prowadzi do rozkładu prawdopodobieństwa

$$bel(y_1) = (\sqrt{2\pi})^{-1} e^{-\frac{1}{2}(-8+y_1)^2}. \tag{9.36}$$

Aby dokonać połączenia obu nowo uzyskanych informacji, przy uwzględnieniu rozkładu wierzeń $bel(q_0)$ z poprzedniej chwili czasu, należy zastosować algorytm

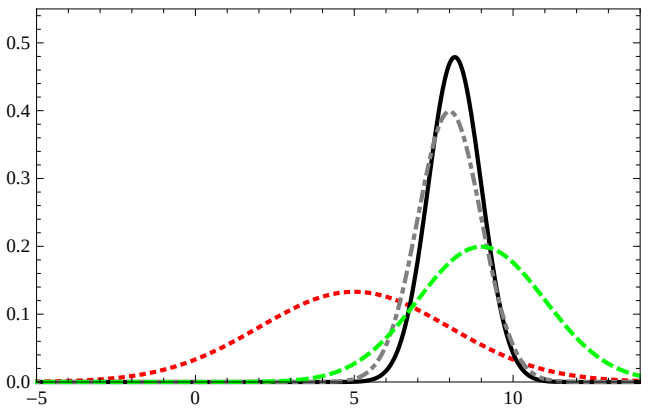


Rys. 9.4: Rozkład prawdopodobieństwa po wykonanym pomiarze położenia – linia pogrubiona; rozkład początkowy – linia cienka.

filtru Kalmana. W wyniku jego działania uzyskuje się rozkład wierzeń o wartości średniej $\mu_1 = 8.166$ i wariancji $\Sigma_1 = 0.833$

$$bel(q_1) = 0.478e^{-0.720(-8.166+q_1)^2}. \tag{9.37}$$

Kolejne rozkłady prawdopodobieństwa, wraz z końcowym rozkładem wierzeń zostały przedstawione na rys. 9.5.



Rys. 9.5: Zastosowanie filtru Kalmana: linia kropkowana – stan wierzeń początkowych; linia kreskowana – predykcja położenia; linia kropkowano-kreskowana – wykonany pomiar; linia ciągła – wynik filtracji Kalmana.

Poprawa w dokładności szacowania położenia robota jest znacząca. Wykorzystanie dodatkowych informacji z czujników oraz predykcja ruchu, po dokonanej fuzji informacji, pozwala osiągnąc zadawalające wyniki. Dokładność szacowania

Markov_localization$(bel(q_{t-1}),u_t,y_t,m)$

1. *for all* q_t *do*
2. $\quad \overline{bel}(q_t) = \int p(q_t|u_t,q_{t-1},m)bel(q_{t-1})dq_{t-1}$
3. $\quad bel(q_t) = p(y_t|q_t,m)\overline{bel}(q_t)$
4. *endfor*
5. *return* $bel(q_t)$

Rys. 9.6: Lokalizacja Markova.

zależy w oczywisty sposób od paramatrów czujników i dokładności ruchu robota – np. gdy występują jedynie niewielkie poślizgi.

9.2.4. Lokalizacja Markova

Algorytm lokalizacji Markova [4] działa w oparciu o filtr Bayes'a (rys. 9.2) przy dodatkowym uwzględnieniu mapy otoczenia m. W kolejnych krokach algorytmu uwzględniane są możliwe położenia w zależności od wykonanego ruchu oraz wyniki pomiarów z czujników. Schemat działania algorytmu przedstawia rys. 9.6.

Rozkład wierzeń $bel(q_0)$ w chwili czasu $t = 0$ zależy od doboru zadania lokalizacji. Jeśli położenie robota nie jest znane można przyjąć rozkład jednostajny – każdy punkt mapy jest równie prawdopodobny jako stan początkowy. Gdy dostępne są dodatkowe informacje, rozkład prawdopodobieństwa wierzeń należy odpowiednio zmodyfikować. W przypadku gdy punkt startowy $\bar{q}_0$ jest znany, przyjmuje się

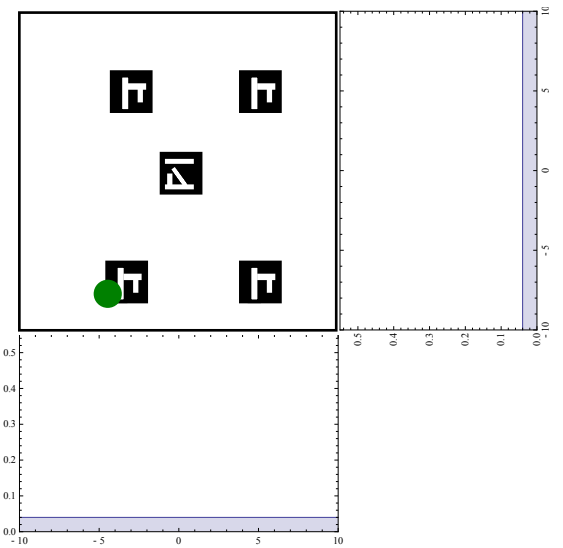
$$bel(q_0) = \begin{cases} 1 & q_0 = \bar{q}_0 \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{9.38}$$

W rzeczywistości położenie początkowe znane jest z pewną dokładnością. Należy więc przyjąć np. rozkład normalny, którego wariancja będzie odpowiadać niepewności pomiarowej czujników służących do wyznaczenia konfiguracji w chwili $t = 0$.

Przykład

Rozważmy holonomicznego robota mobilnego poruszającego się w dwuwymiarowym środowisku. Niech robot zostanie wyposażony w system wizyjny rozpoznający dwa typy znaczników pomagających w lokalizacji. Początkowa pozycja robota nie jest znana – zagadnienie lokalizacji globalnej. Znana jest natomiast mapa otoczenia uwzględniająca rozmieszczenie znaczników lokalizujących.

W chwili początkowej robot z równym prawdopodobieństwem może znajdować się w każdym miejscu pomieszczenia. Sytuacja ta, oraz mapa pomieszczenia zostały przedstawione na rys. reffig:mark1. Położenie robota w chwili początkowej zostało zaznaczone kołem. Po bokach mapy otoczenia zaznaczono rozkłady prawdopodobieństwa odpowiadające położeniom robota odpowiednio w osiach X i Y.

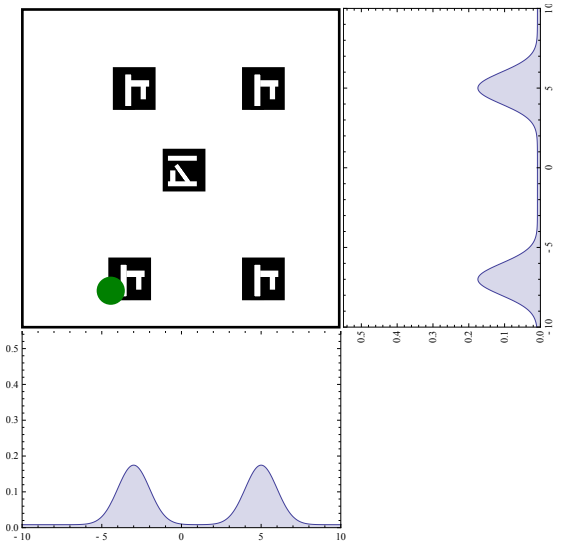


Rys. 9.7: Lokalizacja Markowa – początkowy stan wierzeń $bel(q_0)$.

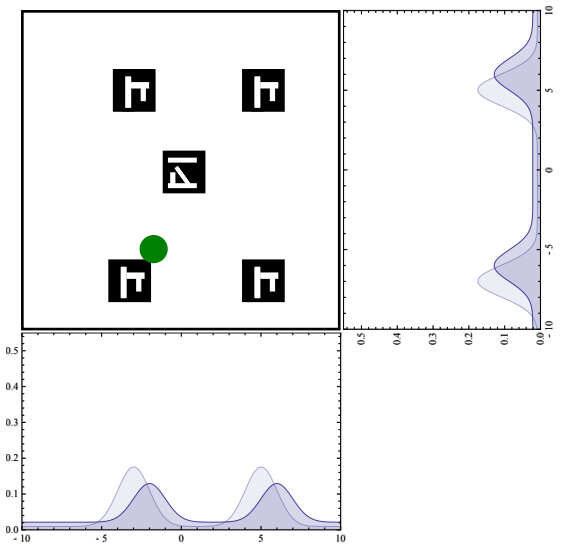
W kolejnej chwili czasu robot dokonuje odczytów z czujników – system wizyjny rozpoznaje znacznik znajdujący się w pobliżu robota. Sytuacja ta, wraz z odpowiednimi rozkładami prawdopodobieństwa przedstawiona została na rys. reffig:mark2.

Wykryty znacznik, zgodnie z mapą otoczenia m, może oznaczać cztery różne miejsca planszy. Powoduje to zwiększenie prawdopodobieństw punktów plansz znajdujących się w pobliży odpowienich znaczników. Położenie robota w dalszym ciągu nie jest znane.

Podczas ruchu robota odowiednie maksima rozkładu poruszają się wraz z nim (rys. 9.9). Następuje niewielkie spłaszczenie rozkładów w wyniku niedokładności przewidywania efektu starowań. Gdy robot na skutek ruchu dotrze do miejsca w którym zostanie wykryty kolejny znacznik, pojawia się nowa informacja przydatna do lokalizacji. Robot uwzględniają wynik pomiaru i dzięki niemu jest w stanie dość dokładnie określić swoją aktualną pozycję (rys. reffig:mark4). Ponieważ istnieje prawdopodobieństwo, że znaleziony znacznik nie został poprawnie rozpoznany, oraz poprzedni model rozkładu prawdopodobieństwa mógł być błędny, uzyskany rozkład nie jest unimodalny.



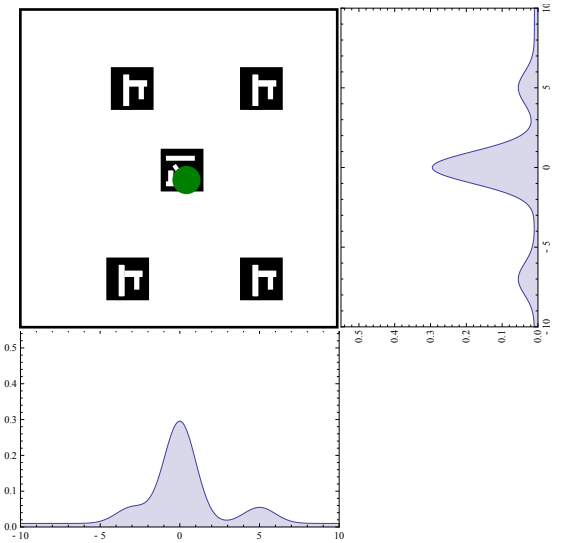
Rys. 9.8: Lokalizacja Markowa – uwzględnienie wyników pomiaru.



Rys. 9.9: Lokalizacja Markowa – przedykcja efektu sterowania.

9.3. Oprogramowanie

Aby sprawdzić działanie metody wykorzystującej filtr Kalmana (rys. ref-fig:kalman) przeprowadzono testy za pomocą specjalnie do tego celu zaimplementowanej aplikacji. Aplikacja symuluje zachowanie obiektu opisanego równa-



Rys. 9.10: Lokalizacja Markowa – uwzględnienie wyniku pomiaru.

niami (9.27, 9.29), z dwoma zmiennymi stanu. Obiekt, określany w dalej również jako robot lub pojazd, porusza się w dwuwymiarowym środowisku zawierającym przeszkody będące wielokątami. Kolizje pojazdu z przeszkodami nie są wykrywane ani symulowane, gdyż postawione zadanie tego nie uwzględnia. Przeszkody stanowią jedynie graficzne urozmaicenie sceny i ułatwiają użytkownikowi interpretację obserwowanych położeń.

Danymi wejściowymi programu są macierze definiujące model (A, B i C), macierze opisujące czujniki (R i Q), początkowa wartość niepewności położenia Σ_0, rozmiar świata, ścieżka po której ma poruszać się robot, oraz informacje o przeszkodach. Rozmiar świata (szerokość i wysokość), dane dotyczące ścieżki i przeszkód wczytywane są z pliku XML. Macierze definiujące parametry modelu i czujników wprowadzane są przez użytkownika w czasie działania programu z wykorzystaniem interfejsu graficznego. Użytkownik ma możliwość włączenia funkcji zapisywania informacji z przebiegu symulacji i podania nazwy pliku do którego te dane mają być zapisywane. W programie zaimplementowana została również prosta funkcja wygładzająca wczytaną z pliku scieżke. Przed włączeniem symulacji, użytkownik ma możliwość doboru parametrów tego wygładzania. Zmieniony może być także sposób generowania sterowania u, a sam algorytm Kalmana może być wyłączony.

Program napisany został w języku C++, z wykorzystanie biblioteki *Qt4.7* oraz niewielkiego fragmentu biblioteki *boost*.

9.3.1. Postać pliku XML

Poniżej przedstawione zostały rodzaje informacji, jakie wczytywane są przez program z pliku XML. Bloki zawierające te informacje powinny znajdować się między parą znaczników o dowolnej nazwie, np. *<map>* i *</map>*.

- **Rozmiar świata**, podany jako parametry *width* i *height* znacznika *level_size*. Jeśli w pliku podanych zostanie więcej niż jeden znacznik określający rozmiar świata, wczytane zostaną dane z pierwszego wystąpienia, oraz wypisane zostanie ostrzeżenie na standardowe wyjście błędu.
- **Przeszkody**, podane w bloku *obstacle*. Blok ten może mieć dodatkowe parametry *x* i *y* definiujące położenie przeszkody na mapie. Przeszkód może być dowolnie wiele. Blok przeszkody zawiera znaczniki określające wierzchołki przeszkody *point*, zawierające atrybuty *x* i *y*. Jesli przeszkoda składa się z mniej niż 3 wierzchołków, nie jest dodawana do mapy, a na standardowe wyjście błędu zostaje wysłana stosowna informacja.
- **Ścieżka**, podana w bloku *path*. Podobnie jak przeszkoda, składa się z elementów *point*. Jeśli w pliku znajduje się więcej niż jeden blok *path*, pod uwagę brany jest tylko pierwszy. Jeśli ścieżka złożona jest z mniej niż 2 punktów, uznawana jest za niepoprawną i uruchamianie symulacji dla tej mapy jest przerywane.

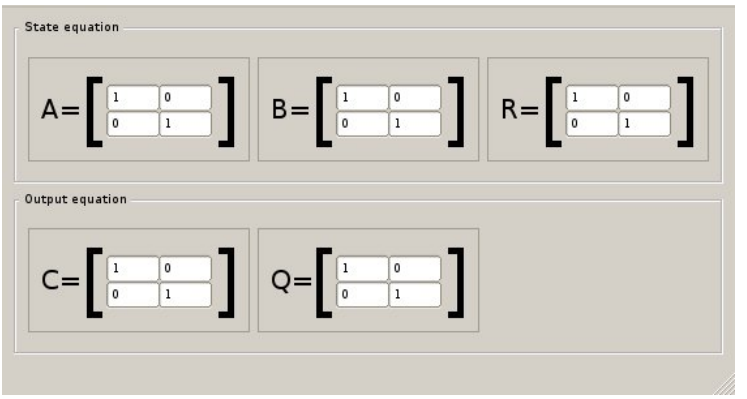
Poniżej zamieszczono zawartość przykładowego pliku wejściowego programu.

```
<map>
  <level_size width="800" height="600"/>
  <obstacle x="50" y="140">
    <point x="10" y="10"/>
    <point x="60" y="30"/>
    <point x="30" y="60"/>
    <point x="30" y="60"/>
    <point x="5" y="23"/>
  </obstacle>
  <obstacle x="900" y="905">
    <point x="10" y="10"/>
    <point x="30" y="10"/>
    <point x="30" y="30"/>
    <point x="10" y="30"/>
    <point x="5" y="23"/>
  </obstacle>
  <path>
    <point x="0" y="500"/>
    <point x="200" y="300"/>
    <point x="400" y="320"/>
    <point x="550" y="90"/>
```

```
    <point x="570" y="100"/>
    <point x="520" y="20"/>
    <point x="300" y="200"/>
    <point x="0" y="0"/>
  </path>
</map>
```

9.3.2. Interfejs

Graficzny interfejs użytkownika składa się z trzech okien. Okno służące do wprowadzania parametrów modelu i czujników przedstawione jest na rys. 9.11. Wywoływane jest z menu *File→Model parameters*. Okno nie posiada żadnych



Rys. 9.11: Widok okna do ustawiania parametrów modelu.

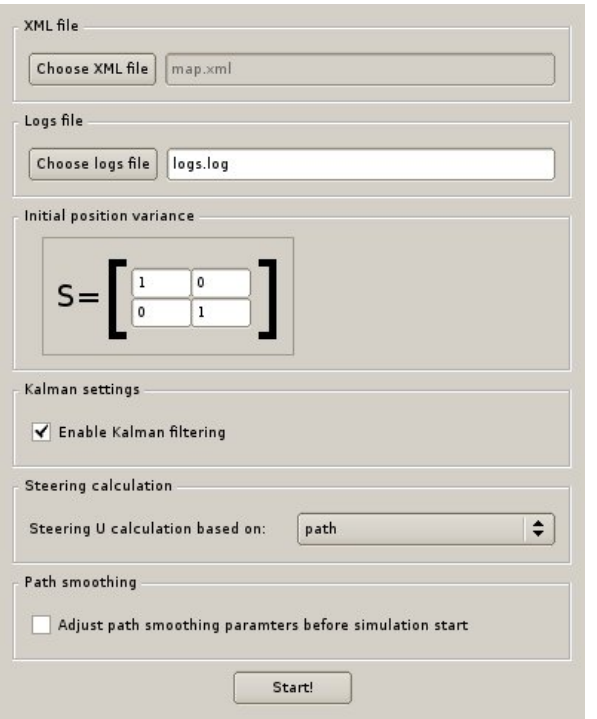
przycisków, dane pobierane są z niego, a ich poprawność jest sprawdzana w momencie, gdy użytkownik wyda polecenie uruchomienia nowej symulacji z okna konfiguracji symulacji, które przedstawione jest na rys. 9.12.

W tym oknie (wywoływanym z *File→New simulation*), użytkownik musi wybrać plik XML z danymi dotyczącymi mapy i zadanej ścieżki (domyślna nazwa *map.xml*). Istnieje możliwość podania nazwy pliku, do którego zapisywany ma być przebieg symulacji (domyślnie *logs.log*). Używkownik może nie wybrać żadnego pliku (zostawić puste pole z nazwą) – wtedy przebieg symulacji nie będzie zapisywany.

Poniżej przedstawiony został przykładowy fragment pliku z zapisem przebiegu symulacji.

```
Starting simulation at Tue Jan 04 00:35:01 2011

Name of XML file: map.xml
Name of logs file: logs.log
Kalman filter enabled: yes
Steering based on: path
```



Rys. 9.12: Widok okna do ustawiania parametrów symulacji.

```
A=[ 1 0 ; 0 1 ]
B=[ 1 0 ; 0 1 ]
R=[ 1 0 ; 0 1 ]
C=[ 1 0 ; 0 1 ]
Q=[ 1 0 ; 0 1 ]
S=[ 1 0 ; 0 1 ]

step 1
Q=[ 12.5382 ; 491.047 ]
Y=[ 13.1554 ; 490.936 ]
U=[ 4.94975 ; -4.94975 ]
M=[ 12.0701 ; 490.658 ]
S=[ 0.666667 0 ; 0 0.666667 ]

step 2
Q=[ 16.9056 ; 486.36 ]
Y=[ 17.3779 ; 486.902 ]
U=[ 4.94975 ; -4.94975 ]
M=[ 17.2437 ; 486.454 ]
```

```
S=[ 0.625 0 ; 0 0.625 ]
```

Na samym początku pliku znajduje się informacja, kiedy rozpoczęta została symulacja. Następnie podana jest konfiguracja programu, przy czym wartość początkowa niepewności położenia pojazdu Σ_0 oznaczona jest jako S. W dalszej części pliku znajdują się informacji z kolejnych kroków symulacji, gdzie Q oznacza prawdziwe położenie robota, Y, oznacza położenie robota odczytane z czujników, U to wartość sterowania, M położenie obliczone z wykorzystaniem filtru Kalmana, a S to macierz określająca niepewność tego położenia. Macierze i wektory zapisywane są w nawiasach kwadratowych, spacje odzielają poszczególne elementy w wierszu, średniki oddzielają poszczególne kolumny.

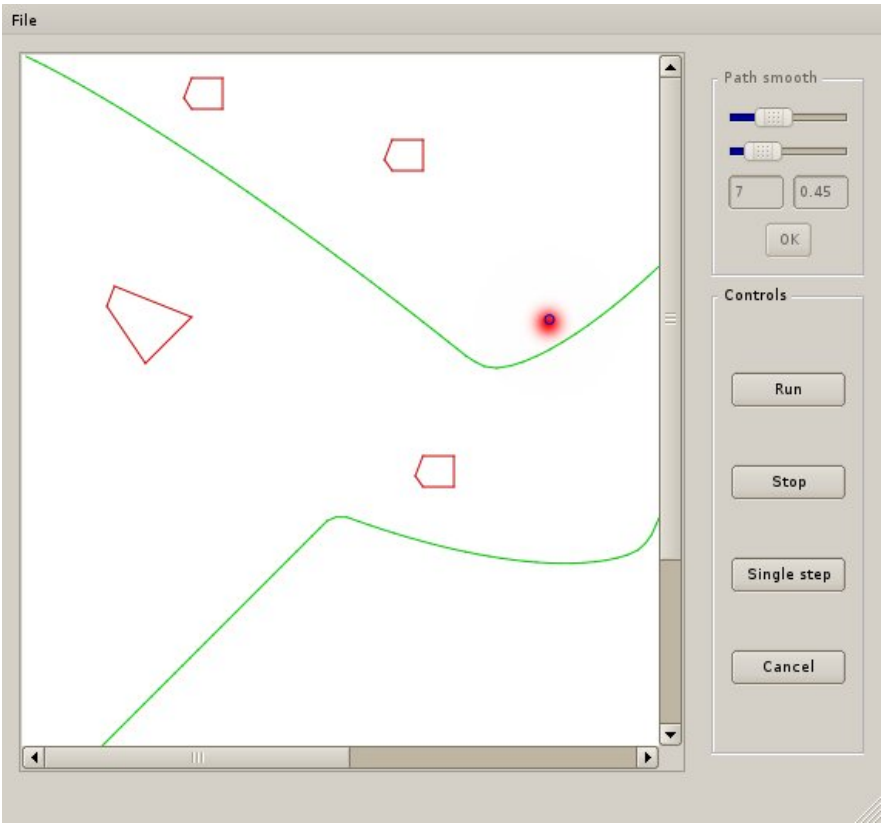
Kolejnym parametrem wprowadzanym z okna ustawień symulacji jest początkowa niepewność położenia (podobnie jak w pliku logów, oznaczona jest przez S). Następnie, użytkownik ma możliwość włączenia/wyłączenia filtru Kalmana. Kiedy filtr Kalmana jest wyłączony, nowa wartość położenia i jego niepewność przyjmowane są na podstawie predykcji (dwa pierwsze kroki algorytmu przedstawionego na rys. 9.3).

Przedostatnim ustawieniem jest wybór sposobu generowania sterowań. Istnieją 4 możliwości:

- *path* – sterowanie generowane jest bez sprzężenia zwrotnego, jako różnica punktu na ścieżce odpowiadającemu następnemu krokowi i punktu na ścieżce odpowiadającemu aktulnemu krokowi,
- *previous output* – sterowanie generowane jest na podstawie punktu na ścieżce odpowiadającemu następnemu krokowi algorytmu i położeniu odczytanym z czujników (Y),
- *previous position* – sterowanie generowane jest na podstawie punktu na ścieżce odpowiadającemu następnemu krokowi algorytmu i położeniu rzeczywistym robota (Q) – jest to przypadek nierzeczywisty, prawdziwe położenie robota nigdy nie jest dokładnie znane,
- *previous mean* – sterowanie generowane jest na podstawie punktu na ścieżce odpowiadającemu następnemu krokowi algorytmu i położeniu robota obliczanym przy wykorzystaniu filtru Kalmana.

Na samym końcu użytkownik wybiera, czy przed uruchomieniem symulacji chce mieć możliwość korekcji parametrów wygładzania ścieżki. Naciśnięcie przycisku *Start!* powoduje sprawdzenie wprowadzonych parametrów – czy istnieje podany plik xml oraz czy wszystkie wprowadzone macierze (parametry modelu z odrębnego okna, opisanego wcześniej, oraz wartość początkowa niepewności położenia) mają liczbową zawartość na wszystkich pozycjach. Błędy sygnalizowane są wyskakującym oknem. Kiedy wszystkie parametry są poprawne i uda się sparsować podany plik XML, oba okna edycji parametrów zostają zamrożone, odblokowane natomiast zostają funkcje głównego okna, które przedstawione jest na rys. 9.13. Jednocześnie zostają w nim narysowane wczytane przeszkody i wstępnie wygładzona ścieżka.

Jeśli użytkownik wybrał opcję dopasowania parametrów wygładzania ścieżki, w oknie głównym aktywuje się pole *Path smooth*, umożliwiające edycje dwóch



Rys. 9.13: Widok okna głównego aplikacji.

parametrów. Algorytm wygładzania ścieżki oparty jest o prostą heurezę: kierunek tworzenia ścieżki zmieniany jest w zależności od tego, który fragment odcinka między dwoma punktami ścieżki jest wygładzany. Im bliżej któregoś z punktów, tym większy wpływ ma on na kierunek tworzenia ścieżki. Algorytm regulowany jest dwoma parametrami. Pierwszy z nich, ustawiany za pomocą górnego suwaka i którego wartość wyświetla się w lewym polu, odpowiada za rozdzielczość z jaką aproksyowana jest ścieżka. Im większa wartość parametru, z tym mniejszej ilości punktów będzie składać się wygładzona ścieżka. Drugi parametr jest współczynnikiem z jakim zmieniają się wagi punktów na podstawie których wybierany jest kierunek ścieżki. Im większy parametr, tym bardziej kanciasta ścieżka, im mniejszy, tym bardziej obła, jednocześnie jednak nie trafia ona dokładnie w zadane punkty.

Gdy wygląd ścieżki odpowiada oczekiwaniom użytkownika, powinien on wcisnąć przycisk *OK*. Spowoduje to wejście w tryb symulacji. Program wchodzi w ten tryb również po wciśnięciu przycisku *Start!* w oknie konfiguracji symulacji, jeśli nie zostanie zaznaczona opcja korekcji parametrów wygładzania ścieżki. W tym trybie aktywują się przycisku w polu *Controls* głównego okna, w polu wizualiza-

cji zostaje dodany robot, oznaczony niebieskim okręgiem, oraz na czerowno narysowany zostaje rozkład prawdopodobieństwa jego pozycji. Kolor rysowanego rozkładu jest normalizowany, tak aby maksymalna wartość obliczonego rozkładu odpowiadała maksymalnej jasności koloru. Wraz ze wzrostem pola powierzchni rysowanego rozkładu (wzrostem wariancji), spada wartość prawdopodobieństwa jaka odpowiada poszczególnym jasnościom koloru. Przycisk *Run* powoduje uruchomienie płynnej symulacji, która może być zatrzymana przyciskiem *Stop*. Pojedynczy krok symulacji wykonywany jest po naciśnięciu przycisku *Single step*. Przycisk *Cancel* powoduje przerwanie symulacji, wyczyszczenie wizualizacji oraz odblokowanie okien edycji parametrów modelu i symulacji.

Uwagi dotyczące działania Filtru Kalmana

Symulacje przeprowadzone z wykorzystaniem dedykowanej aplikacji pozwoliły zweryfikować działanie algorytmu filtracji Kalmana w praktyce. W przypadku wykorzystania jedynie części predykcyjnej algorytmu uzyskane wyniki mają podobny sens do wyników pochodzących z czujników odometrycznych.

$$\begin{aligned} \overline{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\ \overline{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t \end{aligned} \tag{9.39}$$

Zgodnie z równaniem (9.39) błąd estymacji położenia rośnie z każdym krokiem, gdyż kolejna estymacja wymusza dodanie niepewności pochodzącej z modelu (R_t).

Filtracja Kalmana pozwala na ustawienie wartości granicznej dla wariancji położenia. Podczas wykonywania fuzji wyniku pomiaru z wynikiem predykcji wartość pomiaru zawsze zostaje poprawiona (w najgorszym przypadku pozostaje bez zmian). Ponieważ w rozpatrywanym przypadku czujniki posiadały stałą dokładność pomiar zawsze cechował się stałą wariancją. Uzyskanie dodatkowej informacji pochodzącej z modelu pozwala nanieść poprawkę. Kolejne kroki procesu lokalizacji wykonywane są na podstawie poprzednich rozkładów wierzeń, tak więc błąd nie może wzrosnąć.

Wykorzystanie tylko jednego rodzaju czujników nawet o dużej dokładności może okazać się mniej opłacalne niż wykorzystanie filtru Kalmana dla dwóch czujników o dużej wariancji.

9.4. Podsumowanie

W rozdziale przedstawiono najważniejsze metody lokalizacji robota, uwzględniające występowanie niepewności w procesie pomiarowym. Przedstawione zostały algorytmy lokalizacji Markowa, filtracji Kalmana oraz filtracji Bayesa. Podstawy działania algortymów zostały omówione na przykładach.

W rozdziale omówione zostały aspekty implementacyjne filtru Kalmana w symulatorze prostego robota mobilnego wyposażonego w różne rodzaje czujników.

Więcej informacji na temat metod reprezentacji danych niepełnych i niepewneych można znaleźć w [4] oraz [1]. Metody i algorytmy lokalizacji robota zostały szczegółowo omówione w [3].

Literatura

[1] L. Bolc, W. Borodziewicz, and M. Wójcik: *Podstawy przetwarzania informacji niepewnej i niepełnej* PWN, Warszawa, (1991).
[2] A. Gałuszka: Block World Planning with Uncertainty and Sensing Action as Integer Linear Programming Problem. *Challenging Problems of Science*, (2006).
[3] S. M. LaValle: *Planning Algorithms* Cambridge University Press, Cambridge, U.K., (2006).
[4] S. Thrun, W. Burgard, and D. Fox: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)* The MIT Press, London, (2005).
[5] R. E. Kalman: A New Approach to Linear Filtering and Prediction Problems. *Trans. ASME, Journal of Basic Engineering*, (82):35–45, (1960).
[6] T. Kaczorek: *Teoria sterowania: Układy liniowe ciągłe i dyskretne*, volume 1 PWN, Warszawa, (1977).

ROZDZIAŁ 10

WYKORZYSTANIE ŁAŃCUCHÓW MARKOVA DO GENEROWANIA TEKSTÓW

B. Foryś, J. Łubiński

Niniejszy rozdział poświęcono przybliżeniu pojęć związanych z pojęciem „łańcuchów Markova". W kolejnych podrozdziałach opisano przykładowe praktyczne zastosowanie takich łańcuchów do generacji krótkich tekstów na wybrany temat.

10.1. Wstęp teoretyczny

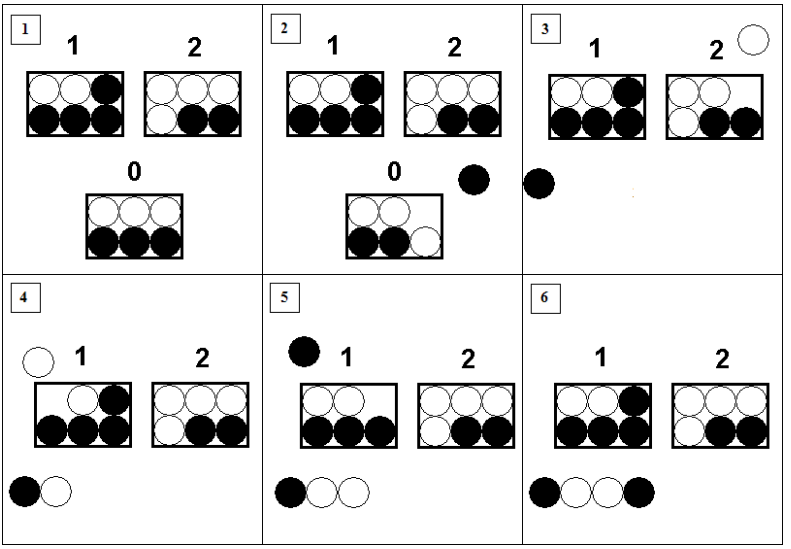
Aby ułatwić zapoznanie się z koncepcją łańcuchów Markova w niniejszym podrozdziale pokazano przykład praktycznego ich wykorzystania. Łańcuchy Markova zaprezentowane zostały w dwóch ujęciach - probabilistycznym oraz przy użyciu teorii automatów.

10.1.1. Przykład - losowanie kul

Na rys. 10.1 pokazano przykład problemu, który zamodelować można za pomocą łańcuchów Markova. Problem ten można opisać następująco:

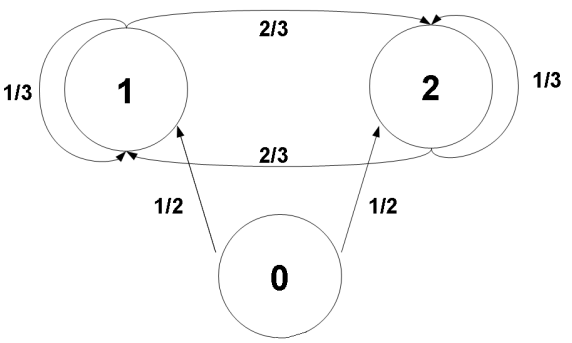
- Niech w trzech urnach o numerach 0, 1, 2 będą umieszczone piłeczki o dwóch kolorach. Wiadomo, że: w urnie 0 znajduje się tyle samo czarnych i białych piłeczek, w urnie 1 jest dwa razy więcej czarnych piłeczek niż białych piłeczek, a w urnie 2 jest dwa razy więcej białych piłeczek niż piłeczek czarnych.
- Z urn mają być losowane piłeczki. Algorytm losowania rozpoczyna się od wylosowania piłeczki z urny 0. Wyciągnięcie czarnej piłeczki oznacza, że następne losowanie odbędzie się z urny 2, a białej piłeczki, że następne losowanie odbędzie się z urny 1. Po dokonaniu losowania, piłeczka wraca do urny, z której została wylosowana. Losowanie z urny 0 inicjalizuje algorytm, po czym urna 0 jest chowana.

Przykładowy przebieg losowania może być następujący. Z urny 0 wylosowano piłeczkę czarną, po czy schowano tę urnę. Ponieważ wylosowano czarną piłeczkę,



Rys. 10.1: Trzy urny użyte do losowaniami.

zatem następne losowanie odbywało się z urny 2. Wynikiem tego losowania okazała się biała piłeczka, a więc następne losowanie miało odbyć się z urny 1. Ponieważ z urny tej wylosowano białą piłeczkę, kolejne losowanie ponownie musiało odbyć się z urny 1. Wynikiem tego losowania okazała się czarna piłeczka. Wynik ten oznacza, że następne losowanie musiało odbyć się z urny 2, ale na tym zakończono eksperyment.



Rys. 10.2: Graf stochastyczny eksperymentu z urnami

Jak widać, prawdopodobieństwo wyboru piłeczki określonego koloru zależy wyłącznie od tego, z której urny odbywa się losowanie. Z uwagi na równoliczność białych i czarnych piłeczek w urnie 0 prawdopodobieństwo przejścia do urny 1 lub 2 w kolejnym losowaniu wynosi $1/2$ dla obu przypadków. Prawdopodobieństwa przejścia pomiędzy urnami 1 i 2 jest równe $2/3$, a pozostania w urnach jest

równe 1/3. Przejścia te wraz prawdopodobieństwami można przedstawić za pomocą grafu jak na rys. 10.2. Taki graf nazywany jest grafem stochastycznym, ponieważ suma wag łuków wychodzących z danego węzła jest równa 1. Graf ten można również przedstawić w postaci macierzy przejścia.

$$\mathbf{P} = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 0 & 1/3 & 2/3 \\ 0 & 2/3 & 1/3 \end{pmatrix}$$

Jeśli ponumerować wiersze i kolumny od zera, to dla danego elementu macierzy numer wiersza oznacza obecną urnę, a numer kolumny zaś oznacza urnę, do której ma nastąpić przejście. Wartość elementu w macierzy równa jest prawdopodobieństwu zajścia danego zdarzenia. Jak widać, z żadnego pudełka nie można przejść do urny 0. Prawdopodobieństwo takiej sytuacji dla każdego z przypadków jest równe 0. Macierz ta jest stochastyczna, ponieważ suma elementów w wierszach jest równa 1.

10.1.2. Łańcuch Markova w ujęciu probabilistycznym

Korzystając z przykładu losowania czarno-białych piłeczek podstawowe definicje można przedstawić w następujący sposób.

Proces stochastyczny - rodzina zmiennych losowych określonych na pewnej przestrzeni probabilistycznej (w przykładzie są to kolejne losowania), o wartościach w pewnej przestrzeni mierzalnej (kolory piłeczek).

Łańcuch - proces stochastyczny określony na dyskretnej przestrzeni (dyskretną przestrzeń stanowią kolejne losowania piłeczek z urn).

Proces Markova (posiadający własność Markova) - to taki proces stochastyczny, w którym warunkowe rozkłady prawdopodobieństwa przyszłych stanów procesu są zdeterminowane wyłącznie przez jego bieżący stan, bez względu na przeszłość. (prawdopodobieństwo wylosowania danego koloru piłeczki zależy bezpośrednio od tego z jakiej urny odbywa się losowanie).

Łańcuch Markova - to proces Markova, który zdefiniowany jest na dyskretnej przestrzeni stanów.

10.1.3. Łańcuch Markova w ujęciu teorii automatów

Łańcuch Markova, zwany także obserwowalnym modelem Markova, jest rozwinięciem automatu skończonego. Automat skończony jest zdefiniowany przez możliwe stany oraz przejścia między nimi. W przypadku obciążonego (ang. *weighted*) automatu dodatkowo każde przejście ma wyznaczone prawdopodobieństwo tego, że zostanie wybrane. Suma prawdopodobieństw wszystkich przejść z jednego stanu musi wynosić 1. Łańcuch Markova jest takim obciążonym skończonym automatem, gdzie sekwencja wejść wyznacza przez jakie stany przejdzie automat. Formalnie można to zapisać w następujący sposób.

- $Q = q_1 q_2 ... q_N$ - N stanów,
- $A = a_{01} a_{02} a_{03} ... a_{n1} ... a_{nn}$ - macierz prawdopodobieństwa przejść między stanami (od i do j),
- q_0, q_F - specjalne stany: początkowy i końcowy.

Łańcuchy Markova wykorzystają własność Markova, która mówi, że warunkowe rozkłady prawdopodobieństwa przyszłych stanów procesu są zdeterminowane wyłącznie przez jego bieżący stan, bez względu na przeszłość. Inaczej rzecz ujmując - przyszłe stany procesu są warunkowo niezależne od stanów przeszłych.

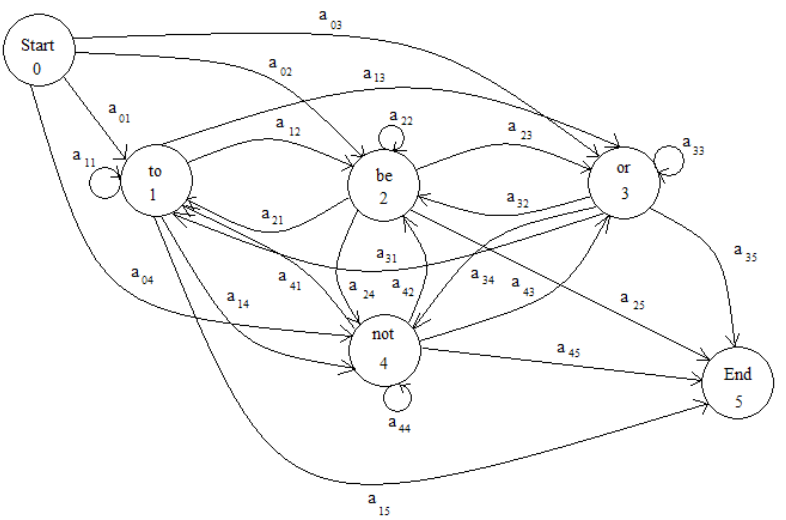
$$P(q_i | q_1 ... q_{i-1}) = P(q_i | q_{i-1})$$

Należy zauważyć, że zgodnie z prawami prawdopodobieństwa, suma prawdopodobieństwa wyjść z danego stanu musi wynosić 1:

$$\sum_{j=1}^{n} a_{ij} = 1$$

10.2. Przykładowe zastosowanie

Łańcuchy Markova można wykorzystać do automatycznego generowania tekstów. Na rys. 10.3 pokazano graf przejść obrazujący prawdobodobieństwo wystąpienia czterech wybranych słów (*to, be, or, not*) w sekwencjach reprezentujących zdania. Na bazie tego grafu można generować proste zdania. W dalszej części tego podrozdziału przedstawiono założenia oraz wynik implementacji przykładowej aplikacji do generowania tekstów w języku angielskim.



Rys. 10.3: Łańcuch Markova dla sekwencji słów (*to, be, or, not*).

10.2.1. Założenia

Tworzone narzędzie powinno umożliwiać generowanie krótkich utworów literackich na wybrany przez użytkownika temat. Idea działania tego narzędzia powinna polegać na wykorzystaniu modeli Markova stworzonych na podstawie analizy istniejących utworów literackich. Analiza tekstów powinna pozwolić na wyznaczeniem prawdopodobieństwo z jakim jedno słowo pada po drugim, a więc wyznaczeniem modelu Markova. Model ten wykorzystany powinien zostać wykorzystany w procesie generowania tekstu. Generowanie tekstu zaczynać się ma od wyboru węzła startowego odpowiedniego grafu. Następnie, w wyniku losowania zgodnie z założonym modelem, wybrane powinno być kolejne słowo oraz powinno dokonać się przejście do kolejnego węzła grafu. Proces ten ma trwać aż do momentu, gdy generator trafi na kropkę lub przekroczy limit słów w zdaniu wybrany wcześniej przez użytkownika. Po wygenerowaniu określonej parametrem ilości zdań program prezentuje uzyskany wynik użytkownikowi.

Istotnym warunkiem mającym wpływ na prawidłowe nauczania programu jest dostęp do różnorodnych, objętościowo dużych tekstów. Spełnienie tego warunku powinno zabezpieczyć program przed generowaniem trywialnych tekstów lub też powtarzaniem tekstów, na podstawie których odbywało się uczenie.

Zakładając, że program został już nauczony przy użyciu odpowiedniej ilości materiału źródłowego, można przystąpić do kolejnego kroku - generacji zdań. Nim jednak to tego dojdzie, należy określić temat generowanego utworu. Aby było to możliwe, dane uczące podzielono wcześniej na pięć kategorii i zbudowano dla każdej z nich osobny model. Wybór tematu uaktywnia odpowiadający mu model. Możliwe tematy utworów to: komedia, dramat, powieść historyczna (jej bohaterowie są prawdziwymi postaciami historycznymi), powieść obyczajowa (jej bohaterowie to zwykli ludzie z problemami rodzinnymi), powieść fantastyczna (w powieści występują elementy magiczne).

Przyjęto, że bazę do stworzenia modeli Markova będą dzieła autorstwa Williama Sheakespeara. Stworzony narzędzie powinnno więc generować utwory literackie w stylu podobnym do stylu tego autora.

10.2.2. Implementacja

Stworzony program do generowania tekstu nazwano Bill 2.0. Został on zaimplementowany w język programowania C++. Na wejście programu podawany jest plik uczący (plik .txt zawierający tekst literacki), wyjściem z programu jest plik wynikowy (plik .txt z wygenerowanym tekstem), o nazwie zadanej przez użytkownika programu.

W kodzie programu wyróżnić można następujące klasy:

- `Bill` - klasa odpowiedzialna głównie za interakcje z użytkownikiem, pobiera nazwy plików. Obsługuje tryb nauki lub generowanie tekstu poprzez wywoływanie metod odpowiednich podklas.
- `Baza` - klasa pełniąca rolę magazynu tematów. Zajmuje się wczytywaniem plików w trakcie nauki do odpowiednich podklas - typu `Temat`, obsługuje również

odczytywanie tych podklas podczas generacji tekstów oraz zapisywaniu wygenerowanych zdań do plików wskazanych przez klasę `Bill`.

- `Klasa Temat` - klasa pełniąca rolę magazynu wyrazów. Znajdują się w niej jedynie te wyrazy, które zostały znalezione w danym temacie, np. komedii. Generuje teksty zgodne z wybranym tematem. Wyrazy zawarte w danym temacie są uporządkowne alfabetycznie, oprócz wyrazów, które rozpoczynały zdania. One mają uprzywilejowane miejsce w wektorze przechowującym wyrazy, a mianowicie mają one indeks 0.
- `Wyraz` - klasa pełniąca rolę magazynu następników danego wyrazu. Odpowiada za dodawanie i losowanie następników wybranego wyrazu. Następniki są uporządkowane alfabetycznie.
- `Następnik` - klasa będąca podstawową cegiełką w programie. Przechowuje takie informacje, jak: nazwa, ilość wystąpień, prawdopodobieństwo danego następnika.

10.2.3. Działanie programu

W przypadku, gdy program podczas uczenia miałby do dyspozycji tylko jedno zdanie: *To be or not to be*, prawdopodobieństwa występowania po sobie kolejnych słów wynosiłyby: $P(to|START) = 1$, $P(be|to) = 1$, $P(or|be) = 0.5$, $P(not|or) = 1$, $P(to|not) = 1$, $P(KONIEC|be) = 0.5$

W przypadku obszerniejszego materiału uczącego wartości prawdopodobieństw przy poszczególnych słowach miałyby dużo mniejsze wartości.

Podczas implementacji przyjęto, że losowanie kolejnych wyrazów dokonuje się w zbiorach odpowiednich następników. Szansa na wylosowanie danego następnika jest równa jego prawdopodobieństwu występowania w tekstach użytych do budowy bazy. Wyjątkiem od tej reguły jest Pierwszy wyraz w zdaniu, który jest losowany zarówno z pierwszych wyrazów zdań tekstów użytych do nauki, jak i z pozostałych wyrazów. Prawdopodobieństwo wylosowania dowolnego wyrazu jako początku zdania jest równe 0.5. Wyraz ten nie może być kropką, ani przecinkiem.

Poniżej zaprezentowany został zapis przykładowej sesji działania programu.

```
Wybierz:
1 Nauka
2 Generacja tekstow
3 Zakoncz program
2
Czy chcesz uzyc bazy innej niz domyslna (BazaBilla.txt): [T/N]
N
Bedzie wczytany domyslny plik bazy
Podaj nazwe pliku wyjsciowego, do prezentacji wynikow:
wyniki.txt
Tematy:
1 komedia
```

```
2 dramat
3 historyczny
4 obyczajowy
5 fantastyka
Podaj nr tematu:
2
Podaj ilosc zdan:
15
Podaj maksymalna dlugosc zdania:
8
```

Wynik działania programu, choć niepoprawny gramatycznie, zawierał ciekawe sentencje. Wyprodukowany przez program tekst zamieszczono poniżej.

```
So please.
What replication should with me with him.
Why thy brain.
Thou character.
How his army over your hands apt.
Entertainment to use.
And a sleep, which are oft breaking.
The king and by our age for i.
A whole kingdom give me my heart of.
Will watch over the main motive, sirrah?.
Do not, thrift, as my true.
The duke's name; his ass- pol.
Of words! heaven's face too much prov'd most.
Enter king your own honour.
'I know you.
```

10.2.4. Wnioski

Program działał zgodnie z oczekiwaniami. Z oczywistych względów zaimlementowane rozwiązanie nie dostarcza gramatycznie i stylistycznie poprawnych tekstów. Niemniej już nawet w tak prostym przypadki widać, ile ukrytego potencjału mają w sobie łańcuchy Markova. Pomijając jednak krytyczną ocenę samej implementacji w ostateczności można uznać, że wygenerowane zdania posiadają ukrytą głębię i znacznie (zwłaszcza ostatnie zdanie).

ROZDZIAŁ 11

ODKRYWANIE REGUŁ ASOCJACYJNYCH

M. Homa, M. Strycharski

Wraz ze wzrostem wykorzystania komputerowych baz danych można zaobserwować wzrost zainteresowania analizą i znajdowaniem wiedzy nie wynikającej bezpośrednio ze struktury lub zawartości tych baz. Analiza zawartości baz danych pod tym kątem nazywana jest eksploracją bądź drążeniem danych (ang. *Data Mining*). Jednym z obszarów takiej analizy jest poszukiwanie reguł asocjacyjnych. Niniejszy rozdział poświęcony jest właśnie takim zagadnieniom.

Mówiąc o regułach asocjacyjnych nie sposób nie przytoczyć przykładu „pieluszek i piwa" (ang. *diaper and beer*), szeroko cytowanego w literaturze przy okazji opisu tego zagadnienia. Wziął się on z analizy sprzedaży sklepów *Osco Drug Stores* w 1992 roku. W jej wyniku stwierdzono, że w godzinach między 17 a 19 jeżeli w koszyku pojawiły się pieluszki, to było w nim też piwo. Wytłumaczeniem miał być fakt, że młodzi ojcowie w tym czasie robią zakupy, więc poza zakupem pieluszek, prawdopodobnie kupią piwo.

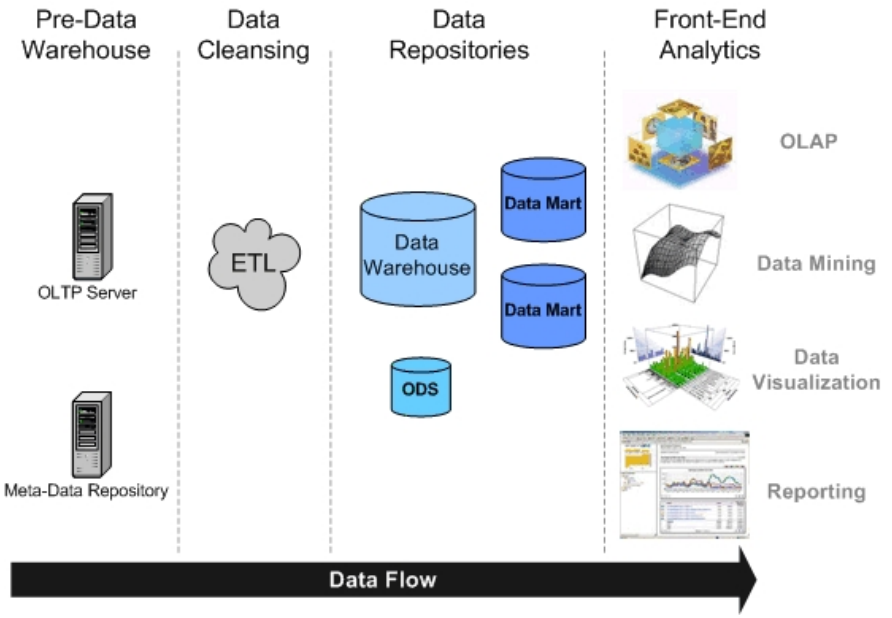
Podobnych spostrzeżeń, nie wynikających bezpośrednio ze struktury bazy danych, można dokonywać więcej. Pozyskiwana wiedza może znaleźć zastosowanie w wielu dziedzinach: począwszy od marketingu i sprzedaży (ułożenie towarów na półkach, akcje promocyjne), przez personalizację usług (sugerowanie podobnych utworów/zdjęć w serwisach internetowych), diagnostykę medyczną, prognozowanie pogody, kończąc na wielu innych.

11.1. Hurtownie danych

Zagadnienie odkrywania reguł asocjacyjnych jest jednym z obszarów drążenia danych (ang. *Data Mining*). Bez względu na stosowaną metodę drążenia danych dużo problemów sprawiają same dane, które zazwyczaj są przechowywane w heterogenicznych i rozproszonych systemach. Dostęp do danych pochodzących z wielu rozproszonych źródeł i mających różne modele jest zazwyczaj bardzo nieefektywny. Rozwiązaniem tego problemu są hurtownie danych.

Czym są hurtownie danych? Ogólnie można powiedzieć, że to bardzo duże bazy danych, z których dane są tylko odczytywane. Hurtownia danych jest zawsze zorientowana tematycznie (np. hurtownia danych na temat sprzedaży pojazdów)

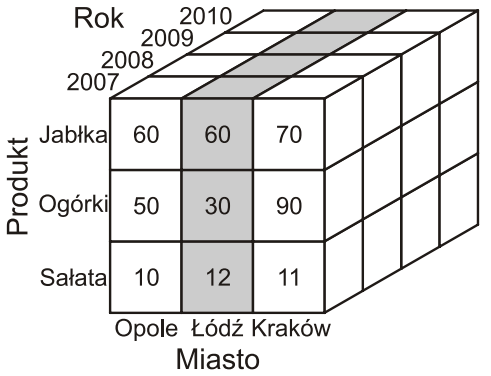
i zawiera dane historyczne i bieżące. Dane w hurtowni są zagregowane na wielu poziomach szczegółowości. Na rys. 11.1 pokazano poglądową architektura typowej hurtowni danych. Dane z wielu heterogenicznych i rozproszonych źródeł trafiają do oprogramowania ETL (ang. *extraction-translation-loading*), które jest odpowiedzialne za odczyt danych, oraz wczytanie do hurtowni wszystkich danych, które zostały wcześniej przetransformowane do wspólnego modelu. Dane z hurtowni mogą być następnie poddawane analizie za pomocą specjalnego oprogramowania.



Rys. 11.1: Poglądowa architektura typowej hurtowni danych (`http://www.isgdatabasedevelopment.com/images/data-warehouse-overview.gif`).

Wszystkie dane w hurtowniach są zorganizowane w postaci wielowymiarowego modelu danych (ang. *Multidimensional Data Model*), który jest też określany jako kostka danych. Model ten bazuje na odpowiednich relacjach między danymi na podstawie których są one w ułożone w wielowymiarowe macierze. W modelu tym istnieje zbiór miar numerycznych, które są przedmiotem analizy. Przykładami takich miar są sprzedaż, budżet, przychody, itp. Każda z miar jest uzależniona od zbioru wymiarów, które tworzą kontekst miary lub, mówiąc inaczej, nadaje znaczenie miarom. Na rys. 11.2 pokazano kostkę danych, która organizuje dane ze sprzedaży produktu według lat i miast, w których miała miejsce sprzedaż. Każda komórka może zawierać dane dla określonego produktu, określonego roku i określonego miasta.

Dane, które znajdują się w hurtowniach, podlegają eksploracji. Eksploracja to proces automatycznego odkrywania nietrywialnych, dotychczas nieznanych, potencjalnie użytecznych reguł, zależności, wzorców schematów, podobieństw lub



Rys. 11.2: Przykład wielowymiarowego modelu danych (kostka).

trendów w dużych repozytoriach danych. Metody eksploracji danych, ze względu na cel eksploracji i typy wzorców odkrywanych w procesie eksploracji danych, można podzielić, bardzo ogólnie, na kilka zasadniczych klas:

- klasyfikacja/regresja,
- grupowanie,
- odkrywanie sekwencji,
- odkrywanie charakterystyk,
- analiza przebiegów czasowych,
- **odkrywanie asocjacji**,
- wykrywanie zmian i odchyleń,
- eksploracja WWW,
- eksploracja tekstów.

Odkrywanie reguł asocjacyjnych jest jedną z wielu metod eksploracji danych. Zatem można stwierdzić, że niniejszy rozdział dotyczy ogólniejszej dziedziny jaką jest eksploracji danych.

11.2. Opis problemu

W poniższych podrozdziałach zostaną opisane podstawowe pojęcia oraz definicje wykorzystywane w algorytmach odkrywania reguł asocjacyjnych oraz ogólny algorytm ich poszukiwania.

11.2.1. Podstawowe pojęcia

Należy zacząć od zdefiniowania przestrzeni i organizacji danych, które będą dostępne w trakcie odkrywania reguł asocjacyjnych. Zakładamy, że istnieje pewien zbiór atrybutów

$$I = \{i_1, i_2, \ldots, i_m\}, \tag{11.1}$$

opisujący dostępne towary w koszyku. Ponadto, niech

$$T = \{t_1, t_2, \ldots, t_n\}, \tag{11.2}$$

będzie zbiorem transakcji (obserwacji). Pojedyncza transakcja zawiera informacje o wystąpieniu danego atrybutu w koszyku (nie musi natomiast określać ilości jego wystąpień). Przykład zbioru danych przedstawiono w tab. 11.1.

Tab. 11.1: Przykładowy zbiór danych

Transakcja	i_1 (pieluszki)	i_2 (piwo)	i_3 (czipsy)	i_4 (chleb)
t_1	1	1	1	0
t_2	0	1	1	0
t_3	1	1	0	1
t_4	1	1	0	0
t_5	1	0	0	1

Reguły będą zapisywane w postaci

$$X \quad \Rightarrow \quad Y, \tag{11.3}$$

gdzie X będzie nazywane przesłanką, natomiast Y wnioskiem. Przyjmują one wartości ze zbioru I

$$X, Y \subseteq I. \tag{11.4}$$

Ważną własnością przesłanek i wniosków jest to, że są one rozłączne (w ramach jednej reguły)

$$X \cap Y = \emptyset. \tag{11.5}$$

Dla przykładu, wykorzystując dane z tab. 11.1 można zdefiniować następujące reguły asocjacyjne:

- {pieluszki} ⇒ {piwo}
- {piwo} ⇒ {czipsy}
- {pieluszki, czipsy} ⇒ {chleb}

Określając reguły asocjacyjne mówi się o zdarzeniach, które występują z pewnym prawdopodobieństwem określonym przez analizę zbioru danych. Miarą prawdopodobieństwa wystąpienia X i Y w zbiorze transakcji T jest *wsparcie*

$$sup(X \cup Y) = \frac{|X \cap Y|}{|T|}. \tag{11.6}$$

Należy jeszcze zdefiniować prawdopodobieństwo wystąpienia zbioru Y pod warunkiem wystąpienia X. Na potrzeby problemu odkrywania reguł asocjacyjnych nazywa się je *zaufaniem*

$$conf(X \cup Y) = \frac{|X \cap Y|}{|X|}. \tag{11.7}$$

Opisowo, wsparcie określa ilość wystąpień zbiorów zawierających X i Y w stosunku do liczności zbioru transakcji T, natomiast zaufanie opisuje ilość wystąpień zbiorów zawierających X i Y w stosunku do ilości wystąpień zbiorów zawierających X.

Dla przykładu, bazując na danych z tab. 11.1, parametry wsparcia i zaufania dla reguły {pieluszki} ⇒ {piwo} mają następujące wartości:

$$sup(\{pieluszki\} \Rightarrow \{piwo\}) = \frac{3}{5} = 0.6$$

$$conf(\{pieluszki\} \Rightarrow \{piwo\}) = \frac{3}{4} = 0.75$$

11.2.2. Zbiory częste

Ważnym pojęciem z punktu widzenia algorytmów odkrywania reguł asocjacyjnych są zbiory częste. Ponieważ szukając reguł podaje się minimalne wsparcie, jakie mają one spełniać (jest to jeden z parametrów poszukiwania reguł), można powiedzieć, że zbiór częsty jest to taki zbiór X, dla którego zachodzi

$$sup(X) \geq minsup. \tag{11.8}$$

W szczególności, zbiorem częstym może być zbiór $X \cup Y$ spełniający nierówność (11.8). Zbiory częste wykazują własność monotoniczności. Oznacza to, że wszystkie podzbiory zbioru częstego są również zbiorami częstymi. Nietrudno stwierdzić również, że jeżeli zbiór B nie jest częsty, to żaden nadzbiór A zbioru B nie jest częsty. Własność ta wykorzystywana jest w niektórych algorytmach (np. algorytmie Apriori).

11.2.3. Ogólny algorytm

W celu odkrycia reguł asocjacyjnych należy zdefiniować dwa parametry, które będą musiały spełniać wygenerowane reguły $X \Rightarrow Y$. Są to minimalne wsparcie *minsup* oraz minimalne zaufanie *minconf*. Znalezione reguły spełniają zatem nierówności

$$\begin{aligned} &sup(X \Rightarrow Y) \geq minsup, \\ &conf(X \Rightarrow Y) \geq minconf. \end{aligned} \tag{11.9}$$

Łatwo zauważyć, że otrzymane reguły zależą od bazy danych, a nie od użytego algorytmu. Swego rodzaju wyjątkiem może być algorytm MSApriori, w którym minimalne wsparcie *minsup* może być definiowane dla różnych atrybutów niezależnie (pozwala to na odkrycie reguł dla atrybutów pojawiających się rzadko w koszyku).

Ogólny algorytm odkrywania reguł asocjacyjnych zawiera następujące operacje:

- Określenie danych - zbioru atrybutów i transakcji, minimalnego wsparcia *minsup* oraz minimalnego zaufania *minconf*.
- Znalezienie zbiorów częstych dla określonego *minsup*.
- Znalezienie reguł asocjacyjnych o zaufaniu nie mniejszym niż *minconf* dla każdego zbioru częstego.

11.3. Algorytm Apriori-T

W niniejszym podrozdziale zostanie przestawiona idea algorytmu *Apriori* [1], a następnie jego niewielka modyfikacja, polegająca na przechowywaniu zbiorów częstych w strukturze *T-drzewa*.

Algorytm *Apriori* został po raz pierwszy zaproponowany w [1]. Jego niewątpliwymi zaletami są prostota oraz stosunkowa łatwa implementacja. Niestety, ma on znaczną złożoność obliczeniową oraz wymagania względem pamięci. Warto również podkreślić, że algorytm ten wymaga wielokrotnego skanowania bazy danych, co znacznie wydłuża jego działanie w praktycznych zastosowaniach (operacja wykonywane na dysku są znacznie dłuższe niż te wykonywane w pamięci).

Podstawe pojęcie w algorytmie Apriori wiąże się z pojęciem zbiorów częstych.

Definicja 11.1 (Monotoniczność zbiorów częstych - własność Apriori)
Monotoniczność zbiorów częstych oznacza, że dowolny podzbiór zbioru częstego jest również zbiorem częstym.

Na podstawie definicji 11.1 można pokazać, że jeżeli zbiór B nie jest częsty, to żaden nadzbiór A zbioru B nie jest częsty. Powyższe spostrzeżenia leżą u podstaw omawianego algorytmu. Ogólny jego przebieg jest zgodny z opisem przedstawionym w podrozdziale 11.2.3.

11.3.1. Algorytm Apriori

W opisie algorytmu będą używane następujące oznaczenia:

- Z_k - kolekcja k-elementowych zbiorów częstych,
- F_k - k-elementowy zbiór częsty,
- C_k - kolekcja k-elementowych zbiorów kandydujących (utworzony na podstawie F_{k-1}).

Ważnym założeniem algorytmu jest posortowanie elementów w ustalonym, niezmiennym porządku leksykalnym. Algorytm, w ogólnym zarysie, przedstawia się następująco:

1. Utworzenie kolekcji jednoelementowych zbiorów częstych Z_1.
2. Przypisanie $k = 2$.
3. Utworzenie C_k - kolekcji k-elementowych zbiorów kandydujących zbudowanych przy wykorzystaniu F_{k-1}.
4. Utworzenie Z_k poprzez odrzucenie z C_k tych zbiorów, które nie są częste.
5. Jeżeli k jest mniejsze od maksymalnej liczności zbioru: zwiększenie k o 1 i powrót do kroku 3; w przeciwnym przypadku koniec algorytmu.

Bazując na kolekcji zbiorów częstych o rosnących licznościach można wygenerować reguły asocjacyjne. Szczegółowy opis algorytmu w postaci pseudokodu przedstawiono na listingach 11.1 oraz 11.2. W celu ilustracji procesu generowania zbiorów częstych załóżmy, że dane są zbiory częste o liczności 3:

$$F_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{1,3,5\},\{2,3,4\}\}$$

W wyniku łączenia otrzymuje się:

$$C_4 = \{\{1,2,3,4\},\{1,3,4,5\}\}$$

Natomiast po eliminacji:

$$C_4 = \{\{1,2,3,4\}\}$$

Listing 11.1: Algorytm Apriori(T).

```
Apriori(T)
C_1 ← zainicjuj(T);
F_1 ← {f|   f ∈ C_1 f.count/n ≥ minsup }; // n − liczba transakcji w T
for (k = 2; F_{k−1} ≠ ∅; k++) do
        C_k ← generuj_kandydatów(F_{k−1});
        for each t ∈ T do
                for each c ∈ C_k do
                        if c zawiera się w t then
                        c.count++;
                end
        end
        F_k ← {c ∈ C_k| c.count/n ≥ minsup }
end
return F ← ⋃_k F_k;
```

Listing 11.2: Generowanie zbiorów kandydujących.

```
Function generacja_kandydatów(F_{k−1})
C_k ← ∅;
forall f_1, f_2 ∈ F_{k−1}
                with f_1 = {i_1, ..., i_{k−2}, i_{k−1}}
                and f_1 = {i_1, ..., i_{k−2}, i*_{k−1}}
                and i_{k−1} < i*_{k−1} do
        c ← {i_1, ..., i_{k−1}, i*_{k−1}}; // łączenie f_1 i f_2
        C_k ← ∪C_k{c};
        for each (k−1)−subset s of c do
                if (s ∉ F_{k−1}) then
                        delete c from C_k // usuwanie zbiorów nieczęstych
        end
end
return C_k
```

11.3.2. Struktura T-drzewa

Przy próbie praktycznej implementacji algorytmu *Apriori* szybko okazuje się, że dużym problemem jest przechowywanie wygenerowanych zbiorów częstych. W najprostszym podejściu można je przechowywać w postaci tablicy z elementami zawierającym kolejne zbiory częste. Należy przy tym pamiętać, że rzeczywiste problemy potrafią generować ogromne ilości danych (bazy danych wykorzystywane w analizie koszykowej mogą mieć tysiące atrybutów i setki tysięcy rekordów). Alternatywnym podejściem jest przechowywanie zbiorów kandydujących w efektywniejszej strukturze, np. w *T-drzewie*. W niniejszym podrozdziale zostanie przedstawiona struktura T-drzewa wraz z przykładem jego generacji.

Przed przystąpieniem do omawiania przykładu należy podkreślić własności T-drzewa:

- Kolejne poziomy budowane są w oparciu o tablice.
- Drzewo budowane jest niejako „od tyłu", tzn. dodając element $\{1,3\}$ rozwijany jest zbiór $\{3\}$ o zbiór $\{1\}$
- W celu zapamiętania zbioru o pewniej liczności dodaje się 2-składnikową strukturę, posiadającą ilość wystąpień oraz wskaźnik na potomka. Wskazuje na nią element o indeksie równym jego numerowi.
- Dodanie nowego poziomu drzewa wymaga dodanie tablicy o wielkości o 1 mniejszej niż rodzic.

Niech dane zamieszczane w strukturze drzewa będą jak w tab. 11.2.

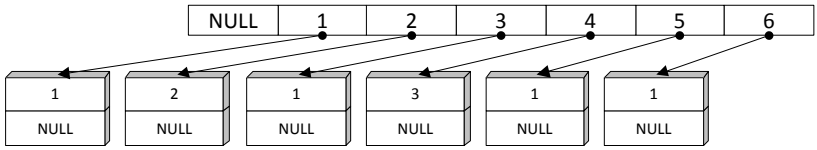
zbiór częsty (liczność)	zbiór częsty (liczność)	zbiór częsty (liczność)
1 (1)	1 3 (1)	4 5 (1)
2 (2)	1 4 (1)	4 6 (1)
3 (1)	2 4 (2)	1 3 4 (1)
4 (3)	2 5 (1)	2 4 5 (1)
5 (1)	2 6 (1)	2 4 6 (1)
6 (1)	3 4 (1)	

Tab. 11.2: Przykładowe zbiory częste do umieszczenie na T-drzewie.

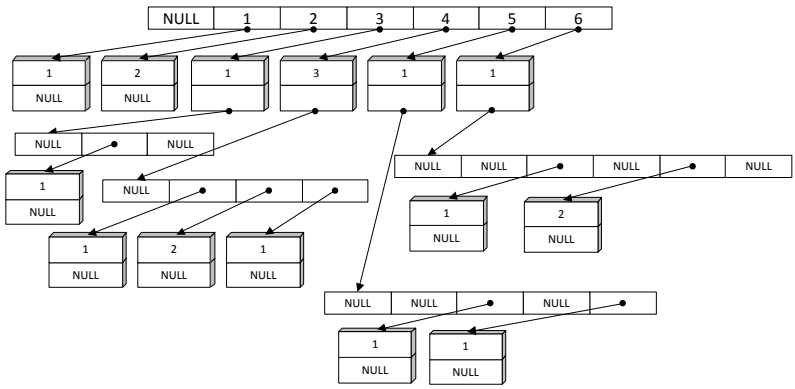
Zbiory te będą dodawane do drzewa według rosnącej liczności, zaczynając od zbiorów 1-elementowych. T-drzewo po dodaniu tych zbiorów przedstawiono na rys. 11.3. Następnie powinny być dodane zbiory o liczności równej 2. Należy pamiętać, że dodawane są one w odwrotnej kolejności. Nowo powstałe gałęzie posiadają $k-1$ elementów (poza elementem NULL), gdzie k to wielkość rodzica. Na rys. 11.4 przedstawiono T-drzewo po dodaniu zbiorów 2-elementowych.

Podobnie postępuje się dla kolejnych zbiorów, wraz z ich rosnącą licznością. Na rys. 11.5 przedstawiono pełne T-drzewo po dodaniu wszystkich zbiorów z przykładu.

Podsumowując, zasada działania algorytmu *Apriori-T* jest identyczna jak algorytmu *Apriori*. Jedyną różnicą jest sposób przechowywania wygenerowanych



Rys. 11.3: T-drzewo po umieszczeniu w nim zbiorów 1-elementowych.

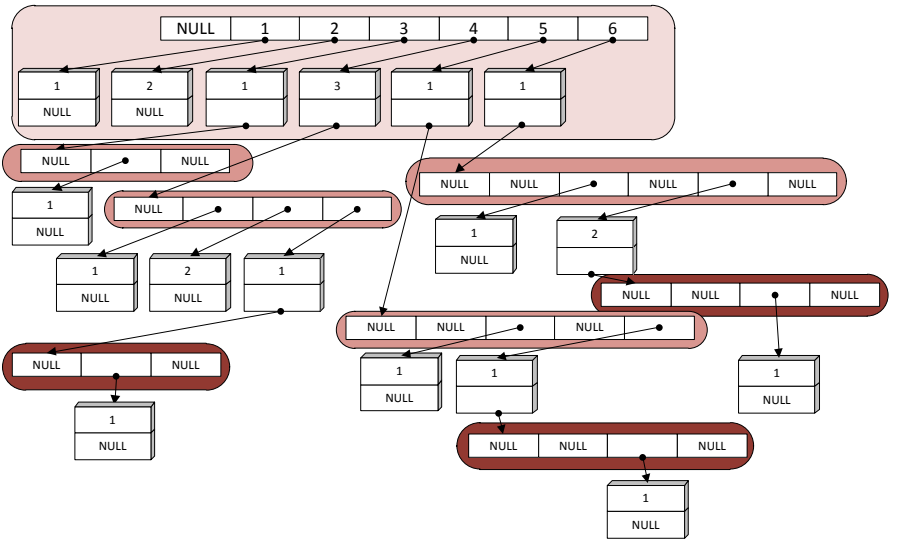


Rys. 11.4: T-drzewo po umieszczeniu w nim zbiorów 2-elementowych.

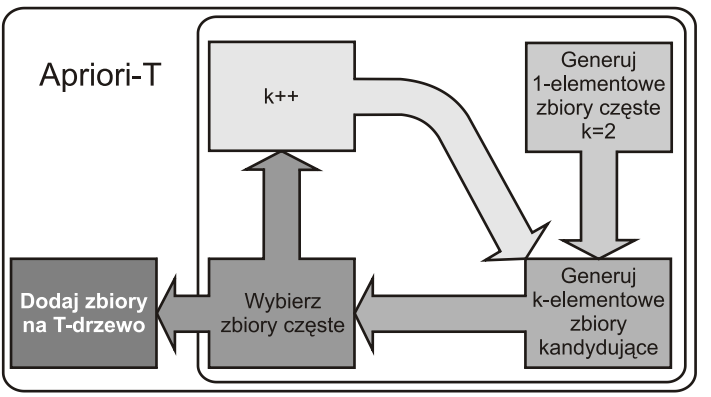
zbiorów częstych. Dodanie nowego zbioru częstego wymaga zajęcia przynajmniej $12B$ pamięci, a dokładniej: $4B$ - na wskaźnik od rodzica, $4B$ - na wskaźnik do potomka, $4B$ - na ilość wystąpień. Niestety, należy doliczyć też zmienną wielkość drzewa i puste pola (dla przypomnienia - pól w potomku jest o 1 mniej niż w rodzicu). Trudno jednak oszacować te wartości, gdyż zależą one od samych danych. Na rys. 11.6 przedstawiono poglądowo przebieg algorytmu *Apriori-T* (nie zawarto na nim m.in. warunku stopu) .

11.4. Algorytm FP-Growth

Najbardziej znanym algorytmem generowania reguł asocjacyjnych jest algorytm *Apriori*. Skoro istnieje już pewien algorytm, który pozwala rozwiązać stawiany problem, po co szukać innych algorytmów? Odpowiedź na to zasadne pytanie jest stosunkowo prosta. Algorytm Apriori ma kilka niewątpliwych i niezaprzeczalnych zalet: jego idea jest prosta do zrozumienia oraz doskonale sprawdza się w przypadku stosunkowo małych zbiorów danych. Nad tymi zaletami górują jednak znacznie poważniejsze wady. Algorytm ten nie pozwala uniknąć dość kosztownego generowania tzw. *zbiorów kandydujących* oraz, co jest szczególnie istotne, wymaga wielokrotnego skanowania bazy danych. Wady te spowodowały, że szukano nowego algorytmu rozwiązującego problem odkrywania reguł asocja-



Rys. 11.5: T-drzewo po umieszczeniu w nim zbiorów 3-elementowych.



Rys. 11.6: Idea działania algorytmu Apriori-T.

cyjnych, który byłby wolny od tych wad. Jednym z takich algorytmów jest algorytm *FP-Growth* (*Frequent-Pattern Growth*).

11.4.1. Idea algorytmu

W algorytmie *FP-Growth* proces odkrywania zbiorów częstych jest realizowany w dwóch krokach:

- Pierwszy krok polega na kompresji bazy danych transakcji. Tak skompresowana baza danych jest następnie przekształcana do specjalnej struktury danych, zwanej *FP-drzewem* (11.4.3).
- Drugi krok polega na eksploracji powstałego w kroku pierwszym FP-drzewa. Eksploracja polega na analizowaniu powstałego FP-drzewa w celu znalezienia zbiorów częstych.

Dzięki zastosowaniu oszczędnej struktury danych, jaką jest FP-drzewo, baza danych jest skanowana tylko dwa razy. Pierwszy raz w celu wyznaczenia częstości wystąpienia każdego z elementów i kompresji bazy danych. Drugi raz w celu konstrukcji FP-drzewa. Od tego momentu wszystkie operacje związane z wyznaczanie zbiorów częstych są realizowana w odniesieniu do tego FP-drzewa. Algorytm *FP-Growth* jest o rząd wielkości szybszy niż algorytm *Apriori*.

11.4.2. Kompresja bazy danych

Proces kompresji bazy danych polega na usunięciu z wszystkich transakcji niektórych elementów. Proces ten opiera się na wyznaczeniu wszystkich jednoelementowych zbiorów częstych. Po wyznaczeniu wszystkich takich jednoelementowych zbiorów częstych, wszystkie transakcje T_i są transformowane do postaci skompresowanej T_{r_i}. Kompresja w tym kroku polega na usunięciu z każdej transakcji wszystkich elementów, które nie są jednoelementowymi zbiorami częstymi. Ostatni krok kompresji bazy danych polega na posortowaniu elementów wewnątrz każdej transakcji według malejącej wartości wsparcia.

Kompresję bazy danych zilustrować można następującym przykładem. Niech $min\_sup = 3$ oraz dostępna jest poniższa baza transakcji:

TID	Elementy
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

Algorytm wymaga, aby wyznaczyć wsparcie każdego elementu jaki pojawia się w transakcjach (pierwsze skanowanie bazy danych transakcji). W wyniku otrzymuje się wartości wsparcie jak w tabeli poniżej:

f	c	a	b	m	p	l	o	d	e	g	h	i	j	k
4	4	3	3	3	3	2	2	1	1	1	1	1	1	1

Kolejny krok polega na usunięciu z każdej transakcji przedmiotów o wsparciu mniejszym niż założony próg (min_sup). W wyniku otrzymuje się posortowaną listę częstych przedmiotów (tab. 11.3).

Tab. 11.3: Posortowana lista częstych przedmiotów.

f	c	a	b	m	p
4	4	3	3	3	3

Ostatni krok kompresji bazy danych polega na usunięciu z każdej transakcji elementów, które nie znajdują się w tabeli 11.3 i posortowaniu elementów wewnątrz transakcji według malejącej wartości ich wsparcia. Wynikiem tej operacji jest lista elementów przedstawiona w tab. 11.4.

Tab. 11.4: Baza danych transakcji po kompresji.

TID	Elementy
1	f, c, a, m, p
2	f, c, a, b, p
3	f, b
4	c, b, p
5	f, c, a, m, p

11.4.3. FP-drzewo

FP-drzewo jest ukorzenionym, etykietowanym w wierzchołkach grafem acyklicznym. Korzeń owego grafu posiada etykietę ***null***. Pozostałe wierzchołki grafu reprezentują jednoelementowe zbiory częste. Z każdym wierzchołkiem (z wyłączeniem korzenia) związana jest etykieta oraz ***licznik transakcji***, który reprezentuje liczbę transakcji wspierających dany zbiór.

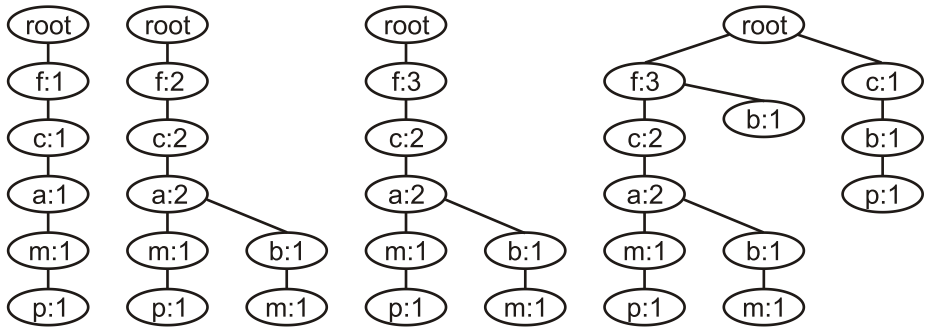
Proces transformacji do FP-drzewa polega na tworzeniu w FP-drzewie kolejnych ścieżek, które odpowiadają kolejnym transakcją. Kolejność występowania elementów w posortowanej transakcji, odpowiada kolejności wierzchołków w ścieżce FP-drzewa reprezentującej daną transakcję. Dla każdego nowego wierzchołka, który został dodany do ścieżki, wartość ***licznik transakcji*** jest początkowo równa 1.

Jeśli do FP-drzewa jest transformowana nowa transakcja (T_{r_j}), która posiada wspólną listę elementów (tzw. prefiks) z transakcją już przetransformowaną do FP-drzewa (T_{r_i}), to transakcja T_{r_j} po transformacji współdzieli ścieżkę reprezentującą wspólny prefiks z transakcją T_{r_i}. Pozostałe elementy transakcji T_{r_j}, które nie należą do wspólnego prefiksu, tworzą nowe wierzchołki połączone łukami.

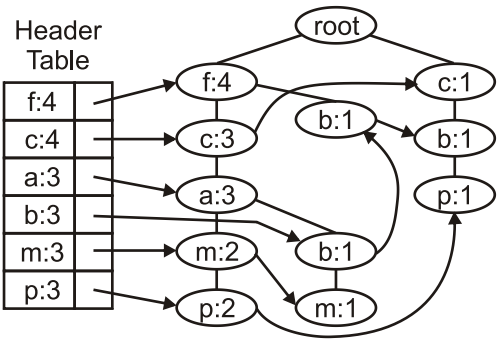
Zatem pojedyncza ścieżka w FP-drzewie, która rozpoczyna się w korzeniu, reprezentuje zbiór transakcji zawierających identyczne elementy. Licznik transakcji ostatniego wierzchołka danej ścieżki zawiera informacje o liczbie transakcji wspierających zbiór elementów reprezentowanych przez wierzchołki grafu należące do tej ścieżki.

W trakcie transformacji kolejnych transakcji tworzona jest tzw. ***tablica nagłówkowa elementów***, która jest strukturą danych pełniącą rolę katalogu – dla każdego elementu wskazuje ona jego lokalizację w FP-drzewie.

Na rys. 11.7 pokazano przykład transformacji zbioru transakcji opisanych w tabeli 11.4 do FP-drzewa. Natomiast na rys. 11.8 pokazano FP-drzewo po transformacji ostatniej transakcji ze zbioru transakcji wraz z tablicą nagłówkową elementów.



Rys. 11.7: Transformacja do FP-drzewa kolejnych transakcji z tabeli 11.4, [4].



Rys. 11.8: FP-drzewo wraz z tablicą nagłówkową elementów dla transakcji z tabeli 11.4, [4].

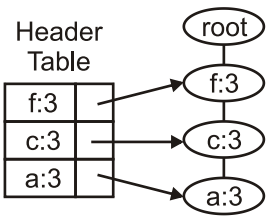
11.4.4. Eksploracja FP-drzewa

Kolejny etap odnajdowania zbiorów częstych za pomocą algorytmu *FP-Growth* polega na eksploracji powstałego w poprzednim kroku FP-drzewa. Proces ten polega na przeszukiwaniu tablicy nagłówkowej elementów, w kierunku *od dołu do góry*. Dla każdego elementu x z tablicy nagłówkowej jest znajdowana tzw. *warunkowa baza wzorca*, czyli zbiór wszystkich ścieżek prefiksowych wierzchołka x. Ścieżka prefiksowa jest to ścieżka prowadząca od korzenia do wierzchołka x.

Kontynuacja eksploracja FP-drzewa polega na konstrukcji tzw. *warunkowego FP-drzewa*. Proces ten polega na potraktowaniu warunkowej bazy wzorca x jako małej bazy danych transakcji $D(x)$. A zatem dla $D(x)$ możemy skonstruować nowe FP-drzewo, które nosi nazwę warunkowego FP-drzewa. Jeżeli powstałe warunkowe FP-drzewo posiada tylko jedną ścieżkę algorytm zatrzymuje się i wypisuje wszystkie zbiory częste (sposób generowania zbiorów został przedstawiony w podrozdziale 11.2.2). W przeciwnym razie cały opisany algorytm powtarza się dla powstałego warunkowego FP-drzewa. Algorytm ten polega zatem na wykonywaniu rekurencyjnego przeszukiwania po strukturze drzewa. Eksporację FP-drzewa można zilustrować następującym przykładem.

Niech w FP-drzewie (11.8) istnieją dwie ścieżki prefiksowe dla elementu p. Są to $[f:2, c:2, a:2, m:2]$ oraz $[c:1, b:1]$. Dla tych dwóch transakcji tylko c jest elementem częstym (bo wystąpiło 3 razy). Zatem warunkowe FP-drzewo ma jeden wierzchołek $[c:3]$. W tym wypadku FP-drzewo ma tylko jedną ścieżkę i wygenerowany zbiór częsty to $\{p, c\}$.

Dla elementu m, podobnie jak dla elementu p, istnieją dwie ścieżki prefiksowe. Są to $[f:2, c:2, a:2]$ oraz $[f:1, c:1, a:1, b:1]$. Elementami częstymi dla tych dwóch transakcji są f, c, a (każdy z nich występuje 3 razy). Warunkowe FP-drzewo znajduje się na rysunku (11.9).



Rys. 11.9: Warunkowe FP-drzewo dla warunkowej bazy danych elementu m, [4].

Korzystając z warunkowego FP-drzewa (11.9) można odczytać następujące zbiory częste $\{f, m\}, \{c, m\}, \{a, m\}, \{f, c, m\}, \{f, a, m\}, \{c, a, m\}, \{f, c, a, m\}$.

Pseudokod algorytmu FP-Growth wraz z wyjaśnieniem zasady generowania zbiorów częstych z warunkowego FP-drzewa opisano w podrozdziale (11.4.5).

11.4.5. Procedura FP-Growth

Listing 11.3: Procedura FP-Growth.

```
1  FP−Growth(Tree, α)
2  if (Tree zawiera pojedynczą ścieżkę P)
3  then
4     for each kombinacji β wierzchołków ścieżki P do
5        generuj zbiór α ∪ β o wsparciu równym minimalnemu wsparciu
6        elementów należących do β.
7     end do
```

```
8         else
9            for each α_i należącego do tablicy nagłówkowej elementów Tree do
10              generuj zbiór β = α_i ∪ α o wsparciu = wsparcie (α_i)
11              utwórz warunkową bazę wzorca β
12              utwórz warunkowe FP−drzewo wzorca β − Tree−β
13              if Tree − β ≠ ∅ then
14                 FP−Growth(Tree − β, β)
15           end do
16        end
```

11.5. Generowanie reguł na podstawie zbiorów częstych

Jak opisano w podrozdziale 11.2.3, reguły asocjacyjne generowane są na podstawie zbiorów częstych. Należy podkreślić, że w przypadku dużej ilość zbiorów częstych, a także ich dużej liczności - ilość reguł może być ogromna (do tego stopnia, że wyciągnięcie z nich wniosków może być kłopotliwe). Algorytm generowania reguł asocjacyjnych można zapisać tak, jak na listingu 11.4

Listing 11.4: Generowanie reguł asocjacyjnych na podstawie zbiorów częstych.

```
1         Function Generuj_reguły(X)
2         for each (zbiór częsty X)
3                 for each (Niepusty zbiór A będący podzbiorem X)
4                         B ← X − A
5                         sup(A → B) ← sup(A⋃B) = sup(X)
6                         conf(A → B) ← sup(A⋃B)/sup(A)
7                         if conf(A → B)≥ minconf
8                                 A → B jest reguła asocjacyjną
9                         end
10                end
11        end
```

11.6. Implementacja

Implementacja algorytmów przedstawionych w podrozdziałach 11.3 oraz 11.4 okazała się niespecjalnie trudnym zadaniem. Do jego realizacji wykorzystano i zmodyfikowano klasy dostępne w [2]. Jako efekt końcowy powstał program w języku *Java* oferujący interfejs użytkownika pozwalający na wygodne wybranie plików z danymi oraz generowanie zbiorów częstych (do pliku, bądź na ekran).

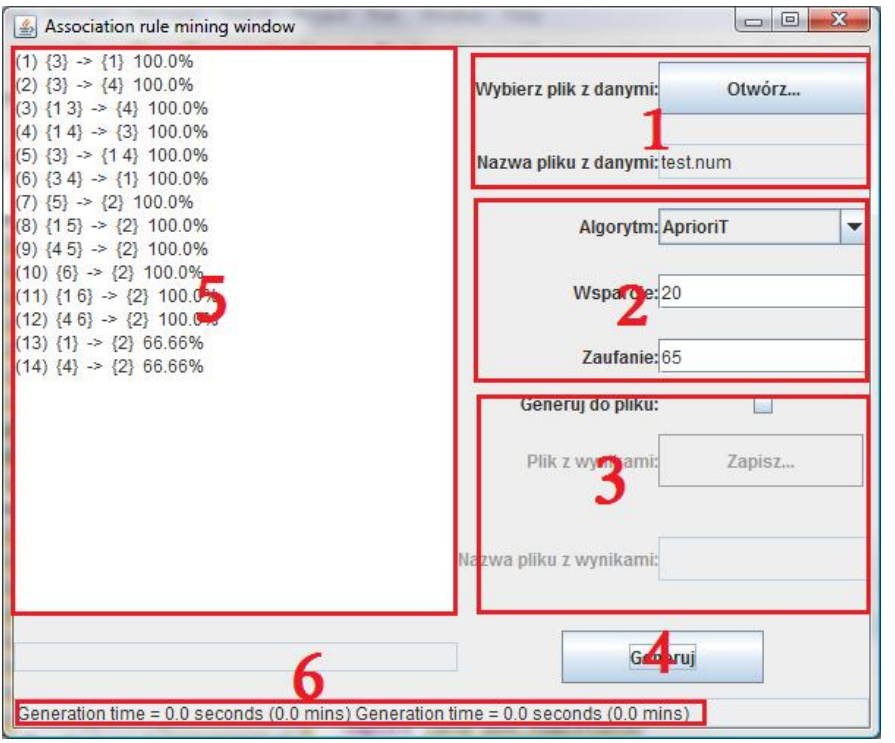
11.6.1. Obsługa programu

Program umożliwia wybranie pliku z danymi. Dane muszą mieć format całkowitoliczbowy, gdzie każdy rekord zapisany jest w osobnej linii. Pojawienie się w rekordzie konkretnej wartości oznacza dla programu wystąpienie atrybutu

przyporządkowanego do tego numeru w konkretnej transakcji. Dla przykładu, plik wejściowy w postaci:

```
1 3 4
2 4 5
2 4 6
1 2 5
1 2 6
```

oznacza bazę danych z transakcjami $\{\{1,3,4\},\{2,4,5\},\{2,4,6\},\{1,2,5\},\{1,2,6\}\}$.
Na rys. 11.10 przedstawiono główne okno programu. Znaczenie poszczególnych pól jest następujące:



Rys. 11.10: Okno programu.

1. Wybór pliku z danymi.
2. Wybór algorytmu oraz ustawienie jego parametrów: wsparcia i zaufania.
3. Włączenie generowania pliku wynikowego (należy podać jego nazwę). W przypadku, gdy ta opcja jest wyłączona, wyniki (tj. wygenerowane reguły) zostaną wyświetlone w oknie programu.
4. Uruchomienie algorytmu.
5. Wygenerowane reguły w formacie (numer) przesłanka -> wniosek zaufanie[%].

6. Czasy generowania odpowiednio zbiorów częstych i reguł asocjacyjnych na podstawie zbiorów częstych.

11.6.2. Wykonane testy

Do testowania efektywności działania algorytmu zostały wykorzystane dane z [3]. Zawierają one opisy sytuacji wypadków (rodzaj terenu, zakres prędkości itp.). Szczegółowy opis znajduje się pod adresem `http://fimi.cs.helsinki.fi/data/accidents.pdf`. Zbiór atrybutów ma liczność równą 572, natomiast zmiennym parametrem w testach jest ilość rekordów (największy zbiór ma 150 tysięcy rekordów).

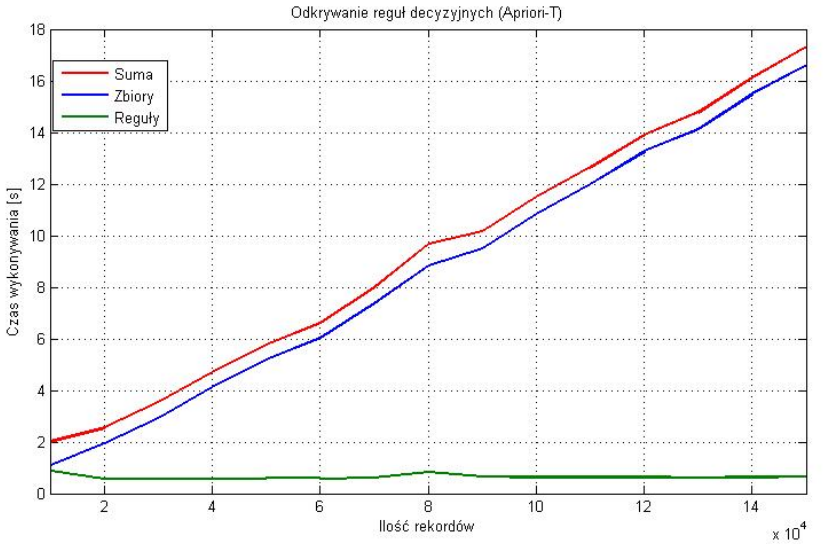
Podczas pracy programu mierzono osobno czas generowania zbiorów częstych i oraz czas generowania reguł asocjacyjnych na podstawie zbiorów częstych. Przyjęto wsparcie *sup=60%* i zaufanie *conf=90%*. Testy przeprowadzono na komputerze *Intel Q6600@2.75Ghz, 6GB RAM, Windows Vista*. Wyniki symulacji dla podanych zbiorów danych przedstawiono w tab. 11.5 oraz na rys. 11.11 i rys. 11.12.

Tab. 11.5: Czasy działania algorytmów dla zbioru testowego, wsparcie = 60%, zaufanie = 90%.

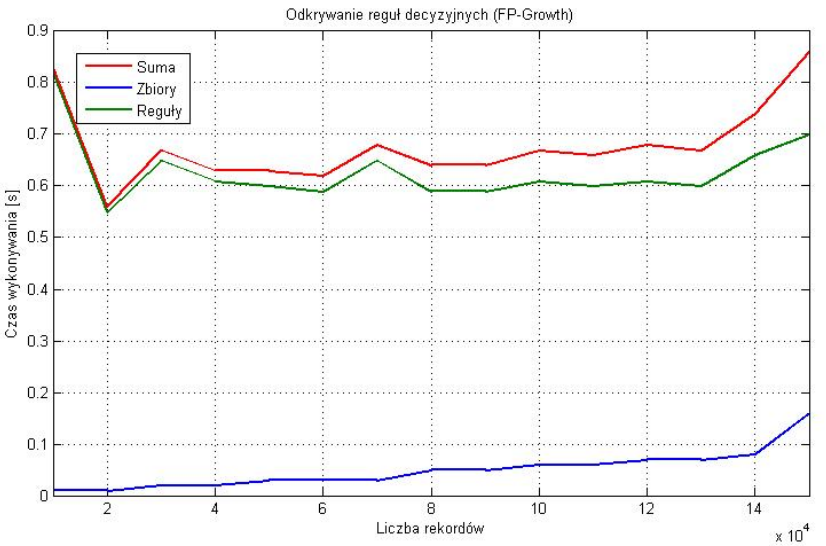
rekordy	Apriori-T			FP-growth		
	zbiory [s]	reguły [s]	suma [s]	zbiory [s]	reguły [s]	suma [s]
150000	16,64	0,68	17,32	0,16	0,7	0,86
140000	15,5	0,65	16,15	0,08	0,66	0,74
130000	14,15	0,64	14,79	0,07	0,6	0,67
120000	13,27	0,66	13,93	0,07	0,61	0,68
110000	12	0,65	12,65	0,06	0,6	0,66
100000	10,86	0,66	11,52	0,06	0,61	0,67
90000	9,52	0,68	10,2	0,05	0,59	0,64
80000	8,85	0,85	9,7	0,05	0,59	0,64
70000	7,38	0,64	8,02	0,03	0,65	0,68
60000	6,04	0,6	6,64	0,03	0,59	0,62
50000	5,2	0,6	5,8	0,03	0,6	0,63
40000	4,12	0,58	4,7	0,02	0,61	0,63
30000	2,95	0,59	3,54	0,02	0,65	0,67
20000	1,97	0,6	2,57	0,01	0,55	0,56
10000	1,12	0,91	2,03	0,01	0,82	0,83

11.6.3. Podsumowanie

Okazuje się, że podstawowym problemem (zarówno jeżeli chodzi o złożoność obliczeniową, jak i złożoność pamięciową) w odkrywaniu reguł asocjacyjnych jest



Rys. 11.11: Czasy działania algorytmu Apriori-T.



Rys. 11.12: Czasy działania algorytmu FP-growth.

generowanie zbiorów częstych. Widać to szczególnie wyraźnie w przypadku algorytmu *Apriori*, którego czasy generowania zbiorów częstych są wielokrotnie większe niż generowania reguł na ich podstawie. Wiąże się to m.in. z wielokrotnym odczytywaniem bazy danych (przetwarzanie informacji zapisanej na dysku jest dużo wolniejsze niż przetwarzanie informacji w pamięci). Należy jednak pamię-

tać, że przechowanie informacji o zbiorach danych i zbiorach częstych w pamięci operacyjnej komputera wymaga wykorzystania złożonych i oszczędnych struktur (jak na przykład *T-drzewo*).

Warto też zauważyć, że czasy generowania reguł asocjacyjnych na podstawie zbiorów częstych są bardzo zbliżone dla *FP-growth* i *Apriori-T* (które używają tego samego algorytmu do ich uzyskiwania). Niewielkie różnice mogą brać się ze sposobu dostępu do nieco innych struktur danych służących do przechowywania zbiorów częstych.

Literatura

[1] T. I. R. Agrawal and A. Swami: Mining association rules between sets of items in large databases. *In Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 207–216, (1993).

[2] F. Coenen: The LUCS-KDD Apriori-T Association Rule Mining Algorithm. epartment of Computer Science, The University of Liverpool, UK., (2004).

[3] T. K. K.Geurts, G.Wets: Profiling High Frequency Accident Locations Using Association Rules. In *Proceedings of the 82nd Annual Transportation Research Board*, page 18, Washington DC. (USA), (2003).

[4] N. H. Son: Reguły asocjacyjne. Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

ROZDZIAŁ 12

INTERFEJSY PROGRAMOWE SPOŁECZNOŚCIOWYCH PORTALI

M. Dubrawski, J. Jewłoszewicz

Portale społecznościowe są ogromnymi zasobami wiedzy. Gromadzą informacje z takich dziedzin, jak np. muzyka czy film, ale również informacje na temat ludzi, ich wzajemnych relacji, zainteresowań czy poglądów. W tym rozdziale skupiono się na przykładowym wykorzystaniu interfejsów programowych popularnych serwisów w celu komputerowego przetworzenia zawartych w nich treści.

12.1. Portale społecznościowe

Portale społecznościowe są serwisami internetowymi skupiającymi i współtworzonymi przez osoby o podobnych zainteresowaniach lub poglądach [1, 2]. Podstawową funkcją takich portali jest tworzenie własnego profilu osobowości, w którym zamieszczane są informacje o użytkowniku, jego zainteresowaniach, wyglądzie oraz gustach. Dzięki takiej budowie systemu możliwe jest odnajdywanie profili innych użytkowników, znajomych bądź osób o podobnych zainteresowaniach. Serwisy zapewniają pewien stopień prywatności, ukrywając część bardziej poufnych informacji przed obcymi. Dostarczają one również użytkownikom wielu możliwości komunikowania się za pośrednictwem takich narzędzi jak czaty, komunikatory, fora dyskusyjne czy prywatne lub publiczne wiadomości. Można wyróżnić dwa charaktery portali społecznościowych [2]:

- otwarte (ang. *external social networking*, ESN)- każda osoba może się do niego zapisać,
- zamknięte (ang. *internal social networking*, ISN) - do serwisu mogą dołączyć osoby, które otrzymały zaproszenia od innych użytkowników serwisu lub legitymują się np. adresem pocztowym w danym serwisie.

12.1.1. Podejście socjologiczne

Portale społecznościowe wywarły nieodwracalny wpływ na kulturę, tradycję, sposób kontaktów międzyludzkich i życie milionów ludzi na całym świecie [2].

Z tego względu stały się ciekawym zagadnieniem dla badań socjologicznych. Najpoważniejsze problemy wiążące się z powstaniem tych serwisów dotyczą:

- prywatności i tożsamości - użytkownicy często ujawniają zbyt wiele iformacji na swój temat, umożliwiając np. kradzież tożsamości, napady rabunkowe lub stwarzając motywy do zwolnienia z pracy,
- rozwoju całego Internetu - w przypadku Polski portal `nasza-klasa.pl` spowodował napływ rzeszy nowych internautów, którym dotąd Internet nie był przydatny,
- przeniesienia relacji z świata realnego do świata wirtualnego - wielu użytkowników serwisów przekłada kontakt za pośrednictwem portalu społecznościowy nad kontakt rzeczywisty ze względu na jego prostotę, szybkość i oszczędność energii.

12.1.2. Historia

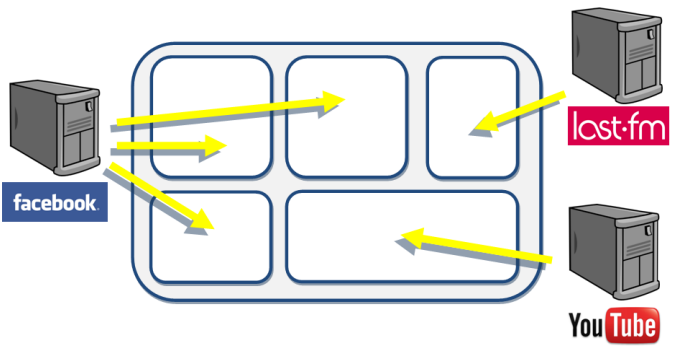
Początek rozwoju serwisów społecznościowych przypada na połowę lat 90 [3]. Przykładami pierwszych portali były Theglobe.com (1994), Geocities (1994) oraz Tripod (1995). Skupiały się one na pozyskiwaniu użytkowników przez udostępnienie im możliwości wzajemnej interakcji, wymiany informacji na swój temat i własnych pomysłów za pomocą prywatnych stron, oraz wsparcie narzędziami pozwalającymi na łatwe publikowanie powyższych treści w internecie. Pod koniec lat 90 wprowadzono funkcję listy znajomych i możliwość szukania użytkowników o podobnych zainteresowaniach. Kolejne serwisy wprowadzały coraz bardziej rozbudowane funkcje i wkrótce stały się jednym z dominujących trendów wykorzystania Internetu. Potwierdzeniem bardzo szybkiego wzrostu popularności portali społecznościowych było odnotowanie większej liczby odwiedzin portalu MySpace niż Google. Facebook uruchomiony w 2004 roku jest obecnie największym serwisem tego typu. Szacuje się, że na świecie istnieje ponad 200 aktywnych portali społecznościowych.

12.2. Przykładowe wykorzystanie interfejsów programowych portali społecznościowych

W tym podrozdziale opisano przykład wykorzystania ogólnodostępnych interfejsów programowych aplikacji określonych portali społecznościowych. Interfejs programowy aplikacji (ang. *Aplication Programming Interface*, API) to interfejs do programowania aplikacji (w postaci biblioteki procedur lub innej formy oprogramowania), umożliwiający realizację określonego zakresu zadań (dostęp do bazy danych, systemu operacyjnego, interfejsu graficznego itp. z pewnego języka programowania) [4].

Pomysłem autorów było wykorzystanie istniejących interfejsów programowych aplikacji do stworzenia klienta sieciowego agregującego informacje z różnych portali społecznościowych o konfigurowalnym interfejsie użytkownika. Strona webowa aplikacji zbudowana byłaby z segmentów, z których każdy powiązany byłby z wybranym przez użytkownika portalem. W zamierzeniu seg-

menty byłyby tworami dynamicznymi, dodawanymi, usuwanymi oraz minimalizowanymi bez konieczności odświeżania strony. Dodatkową funkcją byłaby możliwość dowolnego rozmieszczania segmentów przez użytkownika metodą *drag'n'drop* (podobnie jak ma to miejsce z elementami w iGoogle, zobacz rys. 12.1).



Rys. 12.1: Schemat ideowy.

Główną cechą systemu, a zarazem zasadniczą różnicą w stosunku do interfejsów użytkownika proponowanych przez portale społecznościowe, byłoby profilowane wyświetlane treści pochodzących z różnych serwisów. Polegałoby to na możliwości wyboru przez użytkownika informacji najbardziej dla niego atrakcyjnych. Każdy element witryny byłby przypisany do konkretnego serwisu oraz zawierałby rozłączne, w stosunku do innych segmentów, treści. Dzięki takiemu rozwiązaniu użytkownik określonych portali społecznościowych miałby możliwość jednoczesnego przeglądania interesujących go informacji związanych z profilami na tych portalach oraz usuwania segmentów z treścią, która jest dla niego mniej istotna. Przykładowa aplikacja zakładała wykorzystanie interfejsu programistycznego udostępnionego przez najpopularniejszy obecnie portal społecznościowy Facebook oraz integrację z serwisami Last.fm i YouTube.

12.2.1. Przykładowe funkcje segmentów

Wykorzystanie API udostępnianych przez portale społecznościowe ma służyć filtrowaniu informacji związanych z zainteresowaniami użytkownika.

Facebook jest portalem agregującym przede wszystkim informacje o zainteresowaniach, gustach naszych znajomych oraz polecanych przez nich ciekawych linkach, filmach czy utworach muzycznych. Dlatego autorzy zaproponowali następujące typy segmentów współdziałających z Facebook'iem:

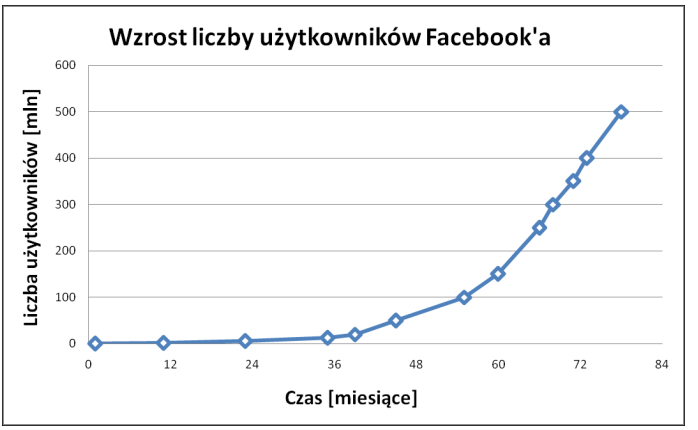
- filtrowanie postów związanych z rodziną użytkownika,
- filtrowanie postów związanych z partnerem użytkownika,
- wyświetlanie linków do zdjęć dodanych do portalu przez znajomych,
- rekomendacje artykułów, linków lub klipów wideo najbardziej lubianych przez znajomych.

W przypadku skorzystania z interfejsu programowego aplikacji portalu Last.fm, który jest największym katalogiem muzycznym online na świecie, przygotowano następujące założenia dotyczące treści ewentualnych segmentów podłączonych do bazy danych serwisu Last.fm:

- utworach aktualnie odsłuchiwanych przez znajomych,
- utworach najczęściej odsłuchiwanych w tygodniu przez użytkownika,
- najpopularniejszych utworach w portalu w oparciu o gusta muzyczne użytkownika (wykorzystanie tagów),
- najpopularniejszych artystach w portalu w oparciu o gusta muzyczne użytkownika (wykorzystanie tagów).

12.3. Facebook

Facebook jest obecnie najpopularniejszym społecznościowym serwisem internetowym na świecie. Liczba aktywnych użytkowników przekracza 500 mln (zobacz rys. 12.2). Swoją popularność Facebook osiągnął dzięki prostocie obsługi oraz wielu funkcjom udostępnionym użytkownikom.



Rys. 12.2: Wykres wzrostu liczby użytkowników serwisu Facebook.

12.3.1. Funkcjonalności Facebook'a

Jak w każdym serwisie społecznościowym Facebook umożliwia tworzenie własnych profili użytkownika. Głównym elementem strony ilustrującej profil jest ściana (ang. *wall*), która jest jednym z możliwych sposobów komunikacji między użytkownikami. Na ścianie pojawiają się również automatycznie generowane wiadomości na temat zainteresowań naszych znajomych oraz ich działalności na portalu.

Na utworzonym przez siebie profilu użytkownik może umieścić swoje zdjęcia pogrupowane w albumy tematyczne. Każde zdjęcie może skomentować oraz zaznaczyć znajomych z portalu, którzy również się na nim znaleźli.

Facebook umożliwia wiele sposobów komunikowania się między użytkownikami. Oprócz wspomnianej już ściany, istnieje również czat, który wyświetla dodatkowo listę zalogowanych aktualnie znajomych, oraz możliwość przesłania prywatnej wiadomości w formie listu. Ponadto Facebook pozwala na tworzenie i uczestniczenie w grupach tematycznych oraz wymianę informacji wewnątrz tych grup.

Jedną z ważniejszych funkcji są tzw. *like pages*. Użytkownik ma możliwość łatwego dzielenia się ze znajomymi swoimi zainteresowaniami, ciekawymi linkami, filmikami, muzyką lub artykułami prasowymi. Mechanizm umożliwiający działanie tej funkcji został opisany w podrozdziale 12.3.3.

Rozbudowany system rekomendacji pozwala na deklarowanie swojej obecności na różnego rodzaju wydarzeniach oraz polecanie swoim znajomym do przyłączenia się do różnego typu akcji.

Duży wpływ na zachęcenie użytkowników do regularnego korzystania z serwisu oraz powiększania liczby znajomych wciąż mają gry udostępnianie przez Facebook'a. Ponieważ sukces oraz postępy w grze wymagają częstego odwiedzania serwisu, liczby aktywnych użytkowników (logujących się co najmniej raz w miesiącu) bardzo wzrosła. Rywalizacja pomiędzy znajomymi, grającymi w tę samą grę, zmusza do aktywnego uczestniczenia w portalu. Gry takie jak `FarmVille` czy `MafiaWars` gromadzą dziesiątki milionów użytkowników.

12.3.2. Historia Facebook'a

Mark Zuckerberg w październiku 2003 roku napisał poprzednika Facebook'a o nazwie `Facemash`. Był to portal typu „Hot or not", na stronie pojawiały się zdjęcia dwóch studentek Harvardu, a użytkownik miał wybrać tę z bardziej odpowiadającym mu wizerunkiem. Z uwagi na nielegalne ściągniecie zdjęć wykorzystywanych w serwisie, Facemash działał jedynie przez 4 godziny. Udało mu się jednak w tym krótkim czasie zachęcić ponad 450 użytkowników, którzy przejrzeli ponad 22 000 zdjęć.

Zachęcony sukcesem student drugiego roku postanowił napisać elitarny portal społecznościowy. Elitarny ze względu na wymaganie posiadania konta pocztowego na serwerze uczelni Harvard. Ogromne powodzenie przedsięwzięcia spowodowało ekspansję TheFacebook'a (tak się pierwotnie nazywał serwis) na inne uczelnie amerykańskie.

W 2005 roku szybko rosnąca popularność serwisu skłoniła założycieli do powiększenia grona użytkowników o uczniów liceum (wymagane było zaproszenie od użytkownika portalu) oraz kilku firm, w tym Apple i Microsoft.

Dwa lata później Microsoft kupił 1,6% akcji Facebook'a za kwotę 240 mln dolarów. W 2008 roku Facebook umożliwił korzystanie z serwisu Europejczykom. Obecnie wartość rynkowa Facebook'a wyceniana jest na 25 mld dolarów. W Polsce jest ponad 8 mln aktywnych użytkowników.

12.3.3. Współdziałanie z Facebook'iem

Dane z portalu Facebook są przechowywane w formie grafu społecznego (ang. *social graph*) [5]. Graf składa się z dwóch typów elementów: obiektów (użytkowników, zdjęć, wydarzeń itp.) oraz tzw. połączeń (ang. *connection*). Platforma Facebook'a udostępnia trzy narzędzia umożliwiające integracje grafu społecznego z zewnętrznymi stronami internetowymi, aplikacjami oraz urządzeniami.

12.3.4. Graph API

Graph API zapewnia możliwość czytania danych z bazy Facebook'a oraz ich zmianę. Udostępnia prostą i jednolitą reprezentację obiektów grafu i połączeń pomiędzy nimi. Każdy obiekt w grafie społecznościowym posiada unikalny identyfikator. Aby dowiedzieć się o właściwościach danego obiektu należy posłużyć się adresem url o następującej strukturze `https://graph.facebook.com/IDENTYFIKATOR`

Natomiast w celu zbadania połączenia dotyczącego danego obiektu `https://graph.facebook.com/IDENTYFIKATOR/TYP_POLACZENIA`.

Typy obiektów występujących w grafie społeczności:

- użytkownik
- strona
- wydarzenie
- grupa
- aplikacja
- wiadomość statusowa
- zdjęcie
- album
- zdjęcie profilowe
- video
- notatka

Typy połączeń:

- znajomi
- wiadomości
- ściana (wall)
- zainteresowania
- filmy
- muzyka
- książki
- notatki
- tagi zdjęć
- albumy
- tagi video
- wydarzenia
- grupy

Wszystkie odpowiedzi bazy Facebook'a są zwracane w formacie `JSON` (patrz (12.6.3)). Przykładowy wynik zapytania o obiekt typu użytkownik wygląda następująco:

```
{
   "id": "671798440",
   "name": "Maciej Dubrawski",
   "first_name": "Maciej",
   "last_name": "Dubrawski",
   "link": "http://www.facebook.com/people/Maciej-Dubrawski/",
   "gender": "male",
   "locale": "en_US"
}
```

Natomiast odpowiedź na zapytanie o połączenie typu film


```
{
   "data": [
      {
         "name": "The Rock (1996)",
         "category": "Movie",
         "id": "114324145263104",
         "created_time": "2010-11-01T10:15:55+0000"
      }
   ]
}
```


12.3.5. Social plugins

Social plugins pozwalają na sprawdzenie, co znajomi lubią lub co skomentowali na różnych stronach internetowych. Wszystkie wtyczki są rozszerzeniami Facebook'a specjalnie zaprojektowanymi tak, że żadna z informacji o użytkowniku nie jest udostępniana stronie, na której się pojawia. Istnieją dwa sposoby umieszczenia wtyczek na stronie: XFBML (rekomendowany) oraz Iframe. Najważniejsze wtyczki:

- przycisk logowania
- rekomendacje
- przycisk 'Lubię to!'
- komentarze
- Facepile

Wtyczki są bardzo przydatnymi narzędziami, gdy celem jest połączenie zawartości strony internetowej z zasobami Facebook'a. Dzięki przyciskowi logowania, użytkownik może zalogować się na swoje konto na Facebook bez potrzeby wchodzenia na orginalną stronę logowania. W dużym stopni ułatwia mu to korzystanie z różnego rodzaju stron posiadających sprzężenie z Facebook'iem.

Rekomendacje dotyczące danej strony, pojawiają się w nawiązaniu do zainteresowań użytkownika. Dzięki temu mechanizmowi rekomendacje są możliwie najlepiej dopasowane do użytkownika i zawierają potencjalnie najbardziej pożądane informacje.

Przycisk 'Lubię to!' (ang *Like button*) jest jedną z najpopularniejszych i najczęściej spotykanych wtyczek udostępnianych przez Facebook'a. Dzięki niej do bazy danych Facebook'a przesyłana jest informacja o tym czym użytkownik się interesuje i co lubi oraz stosowana notatka jest natychmiastowo generowana i publikowana na ścianie profilu. Wtyczka ta wykorzystuje Open Graph protocol (patrz (12.3.6)).

12.3.6. Open Graph protocol

Open Graph protocol jest narzędziem współpracującym z wtyczką 'Lubię to!'. Pozwala on na integrację stron internetowych z grafem społeczności. Dzięki temu strony zyskują funkcjonalności innych obiektów grafu.

Open Graph protocol wykorzystuje metadane zawarte na stronie. Przy pomocy tych informacji kategoryzuje ją jako obiekt jednej z klas. Przykładowe wykorzystania Open Graph protocol w odniesieniu do portalu z katalogiem filmów prezentuje poniższy fragment kodu:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:og="http://opengraphprotocol.org/schema/"
      xmlns:fb="http://www.facebook.com/2008/fbml">
  <head>
    <title>The Rock (1996)</title>
    <meta property="og:title" content="The Rock"/>
    <meta property="og:type" content="movie"/>
    <meta property="og:url" content="http://www.imdb.com/..."/>
    <meta property="og:image" content="http://.../rock.jpg"/>
    <meta property="og:site_name" content="IMDb"/>
    <meta property="fb:admins" content="USER_ID"/>
    <meta property="og:description"
          content="A group of U.S. Marines, under command of
                   a renegade general, take over Alcatraz and
                   threaten San Francisco Bay with biological
                   weapons."/>
    ...
  </head>
  ...
</html>
```

W celu zaznaczenia swojego zainteresowania daną witryną użytkownik (zalogowany do Facebook'a) musi jedynie nacisnąć widoczny na stronie przycisk 'Lubię to!' oraz ikonkę publikowania informacji na ścianie. Po tych operacjach wygenerowana zostanie automatycznie notatka na stronie profilu użytkownika (analogiczna do tej zaprezentowanej na rys. 12.3) oraz zostanie utworzone połączenie pomiędzy obiektem reprezentującym użytkownika oraz konkretną stroną.

Rys. 12.3: Efekt działania 'Like button' i Open Graph protocol.

Wymagane pozycje dla Open Graph protocol zawarte w metadanych:

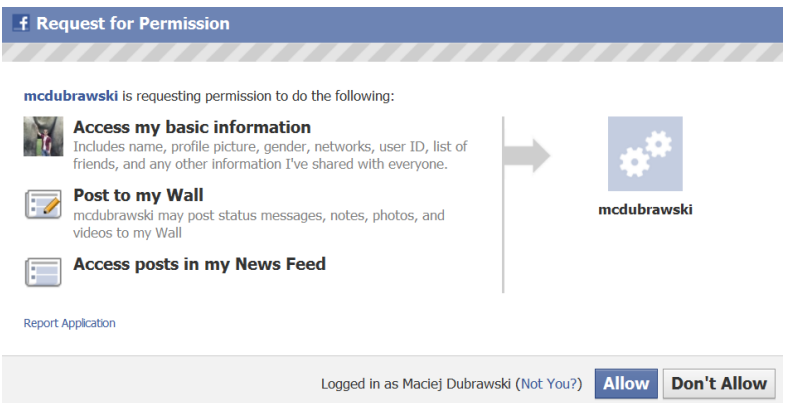
- `og:title` - tytuł obiektu
- `og:type` - typ obiektu
- `og:image` - adres url obrazka reprezentującego obiekt (min. 50×50)
- `og:url` - adres url obiektu, będący niepowtarzalnym ID obiektu

Dostępne kategorie:

- aktywności, sporty
- biznes (firmy, hotele, restauracje)
- grupy (ligi sportowe, drużyny sportowe)
- organizacje (zespoły muzyczne, szkoły)
- ludzie (aktorzy, muzycy, sportowcy)
- miejsca (państwa, miasta, prowincje)
- produkty i rozrywka (książki, gry, jedzenie, filmy)
- strony internetowe (blogi, artykuły)

12.3.7. Autoryzacja

Autoryzacja jest jednym z istotniejszych aspektów korzystania z interfejsów programowych aplikacji serwisów społecznościowych. Dzięki autoryzacji, użytkownicy mają większa gwarancję, że ich dane osobowe umieszczane na stronie nie są ogólnodostępne dla wszystkich aplikacji korzystających z API portali.



Rys. 12.4: Okno dialogowe autoryzacji.

W przypadku Facebook'a, każda aplikacja musi zostać zarejestrowana na stronie. Do rejestracji wymagane są jedynie nazwa aplikacji oraz adres url strony zawierającej aplikację (dla aplikacji webowych). Po zakończeniu rejestracji, administrator dostaje dwa klucze: application ID oraz application secret.

Wszystkie zarejestrowane aplikacje mają dostęp jedynie do podstawowych informacji dotyczących użytkowników portalu Facebook takich jak nazwisko użytkownika, zdjęcie profilowe, płeć oraz lista znajomych. Jeżeli aplikacja chce uzyskać dostęp do bardziej szczegółowych danych, musi poprosić o rozszerzony dostęp do informacji konkretnego użytkownika (przykładowe okno dialogowe pokazano na rysunku 12.4). Po akceptacji przez użytkownika, aplikacja ma wgląd do

wszystkich treści objętych rozszerzonym dostępem. Rozszerzony dostęp nie musi wymagać od użytkownika portalu udostępnienia wszystkich informacji, może poprosić np. tylko o wgląd do zdjęć. Użytkownik ma prawo w dowolnym momencie anulować rozszerzony dostęp wybranej aplikacji i tym samym zablokować jej definitywnie dostęp do swoich danych.

12.4. Last.fm

Portal Last.fm łączy w sobie internetową radiostację oraz system muzycznych rekomendacji. Tworzone są szczegółowe profile gustu muzycznego każdego z użytkowników, na podstawie których na spersonalizowanych stronach prezentowane są jego ulubieni artyści i utwory. Uwzględniane są przy tym utwory odtworzone z filtrowanych kanałów radiowych udostępnianych przez serwis oraz dodatkowo zarejestrowane przez aplikację Last.fm zainstalowaną na komputerze użytkownika i pobierającą informację z odtwarzacza multimedialnego.

12.4.1. Gromadzenie danych przez portal Last.fm

Użytkownik Last.fm może rozbudowywać swój profil muzyczny na dwa sposoby:

- odsłuchując posiadane utwory na odtwarzaczu multimedialnym i pozwalając oprogramowaniu Last.fm (tzw. `scrobblerowi`) rejestrować informację o nich,
- słuchając radia internetowego korzystając z oprogramowania Last.fm.

Odtworzone utwory dodawane są do logów, dzięki którym powstają spersonalizowane tabele ulubionych wykonawców i/lub utworów i generowane dzięki nim muzyczne rekomendacje. Proces automatycznego logowania danych utworu został nazwany *scrobblingiem*.

12.4.2. Główne funkcjonalności portalu Last.fm

Rekomendacje są tworzone przy użyciu algorytmu spersonalizowanego filtrowania. Dlatego użytkownik otrzymuje listę nieznanych mu dotychczas artystów, którzy najprawdopodobniej odpowiadają gustowi muzycznemu danego użytkownika. Last.fm pozwala także na dowolne polecanie wybranych wykonawców, utworów lub albumów muzycznych innym użytkownikom (jeśli tylko rekomendowany element znajduje się w bazie Last.fm).

Prawdopodobnie najbardziej wykorzystywaną opcją z zakresu wirtualnych społeczności, jest możliwość stworzenia grup użytkowników, którzy mają ze sobą coś wspólnego. Last.fm wygeneruje grupę, która upodobniona jest do profilu użytkownika, z tą różnicą, że tworzone są listy zawierające sumę upodobań wszystkich członków danej grupy.

Last.fm daje użytkownikom możliwość tagowania (tj. przypisywania pewnych słów kluczowych) artystów, albumów i utworów, w celu stworzenia rozległej folksonomii muzyki. Użytkownicy mogą segregować muzykę według tagów, jednak większą zaletą tego istnienia tagów jest możliwość korzystania z „radia tagów”,

które pozwala użytkownikom słuchać muzyki, jaka została oznaczona w dany sposób.

Wraz z aktualizacją w październiku 2006 roku dodano do portalu zbiór funkcjonalności związanych z wydarzeniami muzycznymi. Pozwala on użytkownikowi określić dowolną lokalizację i pewną odległość od niej, które będą wykorzystywane do tworzenia propozycji koncertów, którymi dany użytkownik może być zainteresowany.

12.4.3. Korzystanie z API

Pierwszym krokiem do korzystania z API portalu Last.fm (prócz oczywiście samej rejestracji i zalogowania się w serwisie) jest pozyskanie odpowiedniego klucza. Należy skonfigurować konto developera, tj. określić formę tworzonej aplikacji (zwykle jest to opcja aplikacji do użytku niekomercyjnego), a także jej nazwę oraz krótki opis. Po wysłaniu formularza od razu użytkownik natychmiast otrzymuje dwa klucze opisane jako `API_Key` oraz `secret`. W celu tworzenia aplikacji desktopowej wymagane mogą być też biblioteki. W odpowiednim dziale na stronie internetowej znajdują się interfejsy do większości popularnych języków programowania [6].

12.4.4. Żądania REST

Udostępniane przez Last.fm API można wykorzystać za pomocą usług webowych w oparciu o architekturę REST. Polega to na wysyłaniu przez klienta żądania do serwera, które następnie jest przetwarzane. Serwer posiadający dostęp do zasobów zwraca tzw. reprezentację zasobów (ang. `representation of resource`), która zwykle przyjmuje postać dokumentu XML. Zasoby muszą mieć pewien jednoznaczny identyfikator URI. Serwer nie przechowuje informacji o stanie – to klient odpowiedzialny jest za przejścia pomiędzy różnymi stanami.

Korzystanie z żądań REST polega na wysyłaniu żądania typu HTTP POST lub HTTP GET pod odpowiedni adres URL. W przypadku portalu Last.fm jest to adres `ws.audioscrobbler.com/2.0/`. W żądaniu wysyłanym do serwisu występuje nazwa danej metody w postaci wyrazenia `pakiet.metoda`, wraz z odpowiednimi argumentami [7]. Przykładowo, w celu otrzymania informacji związanych z biografią, zdjęciami, odtworzeniami, czy tagami danego artysty należałoby skorzystać z metody `artist.getInfo`. Posiada ona szereg parametrów, z których jedynie 2 są wymagane (potrzebny do uwierzytelniania klucz `API_KEY` oraz nazwa artysty). Żądanie HTTP GET w tym przypadku przyjęłoby postać:

```
http://ws.audioscrobbler.com/2.0/?method=artist.getinfo
&artist=Oasis&api_key=xxx...
```

W odpowiedzi na to żądanie usługa webowa serwisu Last.fm wysyła dokument XML, w którym w postaci drzewa zawarte są wymagane informacje (w celu skrócenia listingu usunięto zawartość niektórych węzłów drzewa XML):

```
<artist>
  <name>Oasis</name>
```

```
  <url>http://www.last.fm/music/Oasis</url>
  <image size="small">http://...</image>
  <stats>
    <listeners>2227380</listeners>
    <playcount>79431244</playcount>
  </stats>
  <similar>
    <artist>
      <name>The Verve</name>
    </artist>
    ...
  </similar>
  <tags>
    <tag>
      <name>pop</name>
      <url>http://www.last.fm/tag/pop</url>
    </tag>
  </tags>
  <bio>
    <published>Thu, 7 July 2010 20:08:16 +0000</published>
    <summary>...</summary>
    <content>...</content>
  </bio>
</artist>
```

12.4.5. Żądania XML-RPC

Tworząc zapytania do serwisu można też korzystać z architektury XML-RPC. Wysyłając żądania HTTP POST na podany adres podaje się wtedy pewną strukturę zawierającą nazwę metody i jej parametry w postaci dokumentu XML. W poniższym przypadku drzewo dokumentu XML reprezentuje żądanie listy najpopularniejszych utworów dla użytkownika `sample_user`.

```
<methodCall>
 <methodName>user.gettoptracks</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>user</name>
      <value>
       <string>sample_user</string>
      </value>
```

```
      </member>
      ...
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Odpowiedź także w tym przypadku przyjmuje postać drzewa XML, podobnie jak dla żądań REST.

12.4.6. Wykorzystanie Last.fm API za pomocą języka Python

Istnieje możliwość wykorzystania języków skryptowych (takich jak Python czy JavaScript) do pobrania danych z Last.fm. Serwis udziela wsparcia Pythonowi poprzez udostępnienie modułu `pylast` (dostępnego na witrynie portalu). Służy on do szybkiego, intuicyjnego i wygodnego wykonywania połączeń do tego serwisu. Przed wykonaniem jakiejkolwiek metody dostępu do danych wymagane jest uwierzytelnienie w serwisie za pomocą klucza `API_KEY`. Początek każdego połączenia będzie identyczny – należy stworzyć obiekt `network`, który służy do pobierania odpowiednich obiektów zawierających informacje o artystach, albumach czy utworach, a także do wysyłania poleceń do serwisu [8]:

```
>>> import pylast
>>> API_KEY = 'Twój API KEY'
>>> API_SECRET = 'Twoje hasło'
>>> username = 'nazwa Twojego użytkownika'
>>> password_hash = pylast.md5('hasło Twojego użytkownika')
>>> network = pylast.get_lastfm_network(api_key = API_KEY,
... api_secret = API_SECRET,
... username = username,
... password_hash = password_hash)
>>> artist = network.get_artist('Massive Attack')
>>> tracks=artist.get_top_tracks()
```

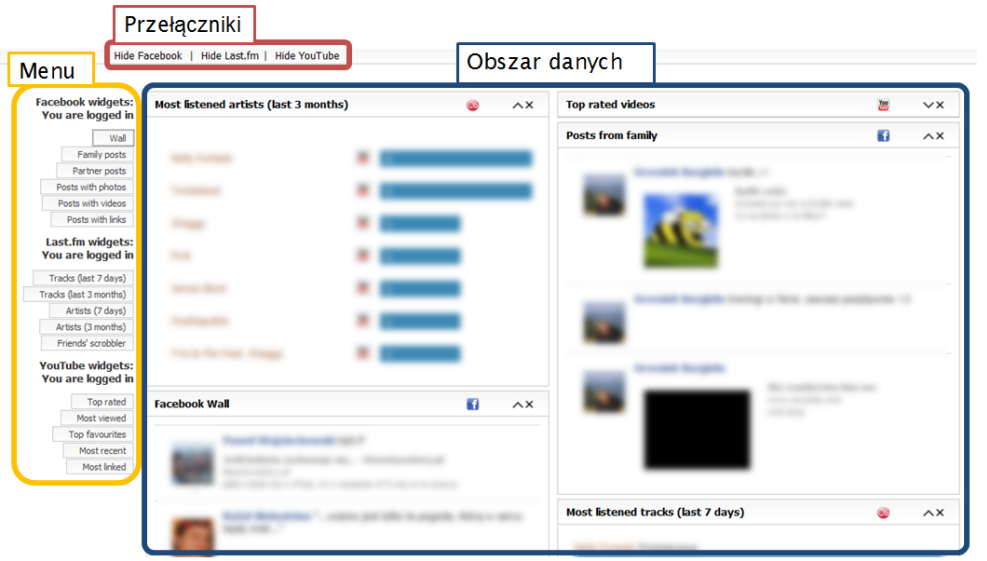
12.5. Implementacja

Podążając za wytycznymi przedstawionymi w podrozdziale 12.2 utworzono aplikację webową, agregującą dane z kilku najpopularniejszych serwisów społecznościowych opisanych powyżej oraz serwisu YouTube.

Interfejs użytkownika składa się z trzech elementów:

- menu, umożliwiającego zalogowanie się do portali oraz wyświetlanie widżetów,
- belki z przełącznikami, umożliwiającymi chwilowe chowanie widżetów skojarzonych z danym serwisem,

- obszaru wyświetlania danych, w którym znajdują się aktywne widżety.



Rys. 12.5: Wygląd okna aplikacji.

12.5.1. Menu

W menu można wydzielić trzy sektory, odpowiadające trzem serwisom społecznościowym. Główną funkcjonalnością menu jest możliwość wyboru danych wyświetlanych w obszarze widżetów. Kliknięcie na jedną z możliwości powoduje pojawienie się widżetu z wybranymi danymi (jeśli widżet jeszcze nie znajdował się w obszarze wyświetlania danych), bądź aktualizację danych w odpowiednim widżecie (jeśli już wcześniej znajdował się w obszarze wyświetlania danych). Menu umożliwia również zalogowanie się do każdego z portali, co jest niezbędne do przejścia fazy autoryzacji i poprawnego pobrania informacji z serwisów.

12.5.2. Widżet

Widżet jest podstawową jednostką, z której składa się interfejs użytkownika (zobacz rys. 12.6). Każdy z widżetów reprezentuje dane pobrane z jednego portalu. Dzięki segmentacji interfejsu na widżety użytkownik ma pełną dowolność w wyborze informacji wyświetlanych w oknie przeglądarki. Segmenty można dowolnie przemieszczać w obrębie obszaru wyświetlania danych.

Widżet składa się z:

- nagłówka opisującego informacje w nim zawarte,
- ikony świadczącej o źródle wyświetlanych danych,
- ikony minimalizacji,
- ikony zamknięcia,

Rys. 12.6: Wygląd pojedynczego widżetu.

- obszaru zawierającego dane pobrane z portalu.

12.5.3. Belka przełączników

Wykorzystując fakt, iż każdy z widżetów należy do jednego z serwisów, wprowadzono przełączniki umożliwiające ukrywanie wszystkich widżetów powiązanych z jednym portalem. Dzięki temu, zapewniono użytkownikowi łatwy przegląd wyświetlanych informacji, minimalizując potrzebę przewijania okna i szukania wiadomości z konkretnego portalu, gdy wybrano wiele widżetów.

12.5.4. Typy widżetów

Dane pobierane z Facebook'a pochodzą z tablicy i są odpowiednio selekcjonowane na pięć kategorii:

- posty rodziny
- posty partnera
- posty ze zdjęciami
- posty z wideo
- posty z linkami

Last.fm dzięki aplikacji Scrobbler agreguje informacje na temat utworów odsłuchiwanych przez jego użytkowników. Jednocześnie portal jest największym zbiorem online informacji na temat artystów i ich utworów. Wykorzystując te dane utworzono następujące widżety:

- najczęściej odsłuchiwani artyści w ciągu ostatnich 7 dni oraz 3 miesięcy,
- najczęściej odsłuchiwane utwory w ciągu ostatnich 7 dni oraz 3 miesięcy,
- lista utworów słuchanych ostatnio przez znajomych użytkownika.

Serwis YouTube jest największym i najpopularniejszym portalem zawierającym krótkie pliki wideo. Korzystając z tego faktu zaimplementowano widżety wyświetlające filmiki:

- najczęściej oglądane,
- najwyżej oceniane,
- najczęściej oznaczane jako ulubione przez użytkowników serwisu,
- najczęściej linkowane,
- ostatnio dodane do serwisu.

Udało się również połączyć dane z Facebook'a i Last.fm z portalem YouTube. Kliknięcie na posta związanego z plikiem wideo umieszczonym w serwisie YouTube automatycznie otwiera dodatkowy widżet **Player**, który odtwarza filmik.

Istnieje również możliwość wyszukania na portalu YouTube plików wideo, które są skojarzone z najczęściej odsłuchiwanymi artystami. Kliknięcie na ikonkę YouTube przy nazwisku artysty automatycznie otwiera nowy widżet zawierający pięć najlepszych dopasowań dla danego artysty.

12.6. Wykorzystane narzędzia

Wykonanie aplikacji webowej ukazującej przykładowy wykorzystanie interfejsów programowych serwisów społecznościowych opierało się na wykorzystaniu szeregu środowisk i bibliotek wykorzystywanych typowo do tworzenia aplikacji webowych.

12.6.1. Programowanie po stronie serwera

Ze względu na prostotę składni oraz ogromną wszechstronność zdecydowano się na zastosowanie języka **Python** do realizacji podstawowego zadania projektu, tj. przetwarzania informacji uzyskanych w wyniku użycie API portali społecznościowych. W celu stworzenia skalowalnej aplikacji, zgodnej z ogólnie przyjętymi wzorcami programistycznymi zdecydowano się na zaprojektowanie aplikacji wykorzystując do tego celu framework webowy o nazwie **Pylons**. Jest to framework do tworzenia dynamicznych serwisów internetowych, który stawia na elastyczność rozwiązań oraz łatwość tworzenia kodu. Wiele jego komponentów może zostać zmieniona lub dopasowana, co sprawia, że jest to idealne narzędzie do budowania wielomodułowej aplikacji webowej, będącej celem projektu.

Python jest interpretowanym, interaktywnym język programowania stworzony przez Guido van Rossuma w 1990. Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią, jest zatem podobny do takich języków, jak Tcl, Perl, Scheme czy Ruby. Python rozwijany jest jako projekt Open Source, zarządzany przez Python Software Foundation, będącą organizacją non-profit.

Głównym powodem dla którego autorzy skłonili się do korzystanie z tego języka programowania jest dostępność modułów napisanych w Python'ie zarówno dla Facebook'a jak i Last.fm wspomagających korzystanie z interfejsów programowych aplikacji.

12.6.2. Programowanie po stronie klienta

Utworzenie modułowej witryny internetowej wymaga użycia języka Javascript dla spersonalizowania wyświetlanych treści i umożliwienia użytkownikowi wyboru rozmieszczenia elementów na stronie internetowej. Zaplanowano wykorzystanie nowoczesnej biblioteki **jQuery** pozwalającej w znacząco łatwiejszy sposób korzystać z możliwości języka JavaScript m.in. do implementacji wywołań metod w technologii AJAX (ang. *Asynchronous JavaScript And XML*). Asynchroniczność wysyłanych żądań i obsługi zdarzeń pozwalają użytkownikowi witryny na nieza-

kłóconą pracę podczas wymiany danych z serwisami społecznościowymi. Dzięki temu wyświetlenie kolejnych widgetów nie wymaga odświeżenia całości strony internetowej. Biblioteka **jQuery** pozwala tworzyć zaawansowane funkcjonalności w bardzo prosty sposób, przy zachowaniu dużej szybkości działania, mniejszej objętości kodu w porównaniu z pozostałymi bibliotekami oraz, co równie istotne, zgodności z wieloma popularnymi przeglądarkami.

12.6.3. JSON

Dane pochodzące z bazy danych Facebook'a oraz Last.fm są pobierane w formacie **JSON**. JSON (*JavaScript Object Notation*) jest lekkim, tekstowym formatem wymiany danych komputerowych, będącym podzbiorem JavaScript. Jego cechami jest prostota w czytaniu i pisaniu dla człowieka a zarazem łatwość w parsowaniu i generowaniu dla komputera.

Dane otrzymywane z portalu YouTube wymagały jednak przetwarzania plików w powszechnie stosowanym formacie XML. Dostęp do danych w formacie JSON uważany jest za łatwiejszy i szybszy z poziomu języka JavaScript. Analiza składniowa takich danych jest również wygodniejsza dla programisty. Panuje opinia, iż format JSON jest bardziej naturalny niż XML, natomiast inni uważają, że jego skąpa notacja jest myląca.

Dane w formacie JSON są zbudowane na dwóch strukturach: kolekcji par nazwa/wartość, uporządkowanej liście wartości. Struktury typów w JSON:

- obiekt - pary nazwa/wartość w nieokreślonej z góry kolejności,
- tablica - uszeregowana kolekcja wartości,
- string - łańcuch znaków ograniczony podwójnymi nawiasami,
- liczba - podobnie jak w języku C,
- wartość - może przyjmować typy string, liczbę, obiekt, tablice, `true`, `false` oraz `null`.

12.7. Podsumowanie

Podczas korzystania z interfejsów programowych portali społecznościowy, najtrudniejszym etapem w procesie tworzenia oprogramowania korzystającego z danych serwisów jest proces autoryzacji. Po pokonaniu przeszkód z nim związanych i pomyślnym pobraniu pierwszych informacji z bazy wiedzy portalu, tworzenie kolejnych zapytań jest stosunkowo proste, gdyż jest to czynione w sposób analogiczny. To sprawia, że dalszy rozwój takiej aplikacji poprzez dodawanie kolejnych funkcjonalności i przetwarzanie gromadzonej wiedzy następuje dość szybko.

Literatura

[1] A. interaktywna Efnetica: Serwisy społecznościowe. `http://efnetica.pl/index.php?page=82&subpage=217&mod=pdf`

[2] A. Leniek: Co to jest Serwis społecznościowy lub portal społecznościowy?. http://www.i-slownik.pl/1,2458,serwis,spolecznosciowy,lub,portal,spolecznosciowy.html, (2010).
[3] Social network service. http://en.wikipedia.org/wiki/Social_network_service
[4] Co to jest API?. http://top1-1.com/co-to-jest-api/
[5] Facebook Developers – Documentation. http://developers.facebook.com/docs/
[6] Last.fm. http://en.wikipedia.org/wiki/Last.fm
[7] Last.fm API. http://www.lastfm.com/api
[8] Wykorzystanie Last.fm API w Pythonie. http://www.openserver.eu/index.php/2010/08/24/37programowanie-z-uzyciem-last-fm-ap/

ROZDZIAŁ 13

TECHNOLOGIE AGENTOWE W WYSZUKIWANIU INFORMACJI

W. Para, J. Lubiński

13.1. Wstęp

Technologie agentowe są zagadnieniem stosunkowo młodym. Powstały jako wyższy od obiektowego paradygmat programowania w związku z potrzebą rozwiązywania różnorakich zagadnień w sposób rozproszony. Dzięki nim możliwym stało się podejmowanie kolektywnych działań, a więc rozwiązywanie problemów w sposób rozproszony, z czym nie mogły poradzić sobie programy działające na jednym komputerze. Mobilność i współpraca wielu agentów oraz inne cechy systemów agentowych powodują, że generowanie rozwiązań problemów wykazuje większą odpornością na błędy w stosunku do podejścia obiektowego. Dzięki zastosowaniu agentów stało się możliwe wykonywanie działań w strukturach o dużej niepewności co do zgodności i dokładności danych. Wreszcie, umożliwiło to przeprowadzanie z niespotykaną dotąd realnością symulacji układów, w których działa wiele niezależnych elementów, jak np. przy przewidywaniu mechanizmów giełdowych, animacjach komputerowych, tworzeniu filmowowych efektów specjalnych (symulacja tłumu, animowanie tkanin).

Podejście agentowe jest nie tyle lepsze od podejścia obiektowego, co stwarza możliwości leżące do tej pory poza zasięgiem programistów. W kolejnych podrozdziałach przybliżono cechy technologii agentowych oraz omówiono potencjalne przypadki ich wykorzystania, ukazano narzędzia i metody, którymi można posługiwać się przy programowaniu agentowym, podpierając to przykładami.

13.2. Podstawowy teoretyczne systemów agentowych

13.2.1. Defincja agenta, cechy agenta, typy agentów

Z powodu braku ogólnie obowiązujących standardów nie istnieje formalna defincja agenta. Nieformalnie jest to jednostka działająca w pewnym środowisku, zdolna do komunikowania się, monitorowania swego otoczenia i podejmowania

autonomicznych decyzji, aby osiągnąć cele określone podczas jej projektowania lub działania. Do podstawowe cech agenta zalicza się:

- autonomiczność (podejmowanie samodzielnych decyzji),
- komunikatywność (komunikacja z innymi agentami i użytkownikiem),
- percepcja (postrzeganie i reagowanie na zmiany środowiska),
- zdolność do wykorzystywania wiedzy,
- tolerancyjność na błędy, złe wejścia,
- zdolność do używania symboli i abstrakcji,
- zdolność do adaptacji w celu osiągnięcia danego celu,
- zdolność do uczenia się,
- umiejętność do przeprowadzania operacji w czasie rzeczywistym,
- zdolność do komunikacji w języku naturalnym.

Działanie agenta określane jako zachowanie jest planowane i przeprowadzane w oparciu o umiejętności w jakie został wyposażony na etapie projektowania i implementacji. Prowadzi ono do osiągnięcia celu, do którego agent dąży. W związku z tym istnieje wiele typów agentów. Wyróżnić można m.in. agentów:

- poszukujących informacji,
- zarządzających siecią,
- wspomagających zarządzanie przedsiębiorstwem,
- działających w ramach systemów ekspertowych,
- symulujących zachowania socjalne,
- oraz agentów w rozumieniu sztucznej inteligencji, nie będących jednostkami programowymi.

Agenci działają często jako centralne ośrodki przetwarzania wiedzy. Coraz częściej pojedynczy agent jest częścia większej całosci, elementem składowym tzw. **systemu wieloagentowego**.

13.2.2. Systemy wieloagentowe

Systemy wieloagentowe (ang. *Multiagent Systems*, MAS) stanowią alternatywę dla scentralizowanego modelu przetwarzania informacji. Wywodzą się z obszaru badań nad rozproszoną sztuczną inteligencją (ang. *Distributed Artificial Intelligence*, DAI).

System wieloagentowy składa się z wielu agentów komunikujących się ze sobą przy jednoczesnym zachowaniu autonomii działania i podejmowania decyzji przez poszczególnych agentów. Agenci wykonują akcje w oparciu o wiedzę jaką dysponują: początkową oraz zgromadzoną w trakcie działania. Nową jakością wprowadzoną przez MAS jest wzajemna interakcja między jednostkami programowymi, modyfikującymi swoją wiedzę i działania w oparciu o zachowania przejawiane przez innych agentów i informacje jakie otrzymują w procesie komunikacji z innymi jednostkami.

Racjonalna jednostka wyposażona jest w strategię prowadzącą ją do osiągnięcia założonego celu. Jako część systemu wieloagentowego może ona realizować strategię współpracy z innymi jednostkami, jeśli przybliża to ją do osiągnię-

cia celu lub nawet być częścią grupy agentów realizujących określony cel. Systemy MAS umożliwiają lepsze i bardziej naturalne modelowanie wielu dziedzin charakteryzujących się: rozproszeniem, komunikacją i interakcjami elementów, współpracą, rywalizacją.

Inną ważną cechą systemów wieloagentowych jest odporność na błędy i awarie. Systemy MAS stanowią jedną z najszybciej rozwijających się gałęzi badań nad sztuczną inteligencją, jednocześnie są wykorzystywane w konkretnych zastosowaniach przemysłowych i biznesowych:

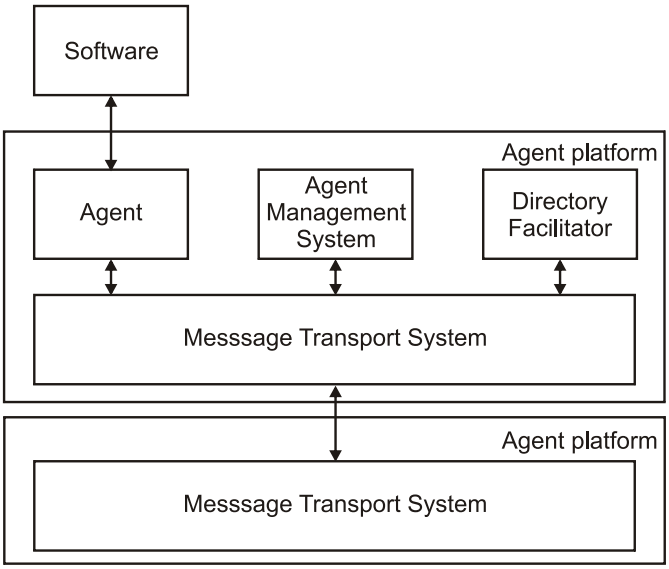
- rozwiązywanie problemów, których natura ma charakter rozproszony – podejście agentowe ułatwia modelowanie systemów działających w środowisku rozproszonym, mierzących się z problemami, których złożoność przekracza możliwości jednostki,
- symulacje rzeczywistości – zwłaszcza grup składających się z autonomicznych jednostek, które znajdują zastosowanie w naukach biologicznych i społecznych,
- zarządzanie wiedzą – wyszukiwanie, gromadzenie, analiza informacji znajdujących się w internecie,
- eksploracja sieci – agenci mobilni, przemieszczający się w sieci i wykonujący zadania na rzecz użytkowników, agent może stanowić interfejs pomiędzy użytkownikiem a maszyną,
- zagadnienia sterowania w robotyce – agenci reaktywni mogący wpływać na otoczenie w oparciu o impulsy z niego odbierane,
- modelowanie oprogramowania – technologie agentowe wpływają na koncepcje modelowania, dostarczając wysokopoziomowej abstrakcji jaką jest agent.

13.3. FIPA jako standard w systemach agentowych.

Powstała w 1996 r. organizacja **FIPA** (ang. *Foundation for Intelligent Physical Agents*) postawiła sobie za zadanie stworzenie standardów dla programowania agentowego [1]. Współpracując z firmami oraz uczelniami starała się wypracować ogólnie obowiązujące zasady działania agentów i systemów agentowych, co zaowocowało powstaniem szeregu specyfikacji. Najpopularniejszymi z nich są *Agent Managment* oraz *Agent Communication Language* (*FIPA-ACL*).

Struktura systemu agentowego FIPA

W systemach agentowych FIPA agenty działają na tzw. platformie (ang. *Agent Platform*, AP). Zawiera ona (zobacz rys. 13.1) system zarządzający agentem (ang. *Agent Managment System*), system transportujący wiadomości w ACL wewnątrz platformy i pomiędzy platformami (ang. *Message transport System*), a także (opcjonalnie) system rejestracji usług (*Directory Facilitator*) możliwych do wykonania przez poszczególne agenty (*Yellow Pages*). Usługa *Agent Managment System* odpowiada za tworzenie i usuwanie agentów, decyduje o wstrzymywaniu ich pracy, przenoszeniu na inne platformy oraz przechowuje nazwy agentów i odpowiadające im identyfikatory (*White Pages*).



Rys. 13.1: Struktura systemu agentowego FIPA, za [1].

Dodatkowe usługi FIPA

Standard FIPA zapewnia możliwość korzystania z dodatkowych, opcjonalnych usług. Są to m.in.:

- integracja systemów agentowych,
- interfejsy użytkownika,
- zarządzanie ontologiami,
- protokoły interakcji – wzorce podstawowych lub typowych działań,
- usługi oparte na języku komunikacji ACL.

Język ACL

Język ACL (ang. *Agent Communication Language*) jest wprowadzoną przez FIPA propozycją standardowego języka dla komunikacji agentów. Jest on oparty na teorii tzw. *aktu mowy*' jako wypowiedzi języka naturalnego. Definiuje on sposoby przekazu informacji agentów o wspólnej *ontologii* (część wiedzy agenta, opisująca nad jakimi zagadnieniami może on pracować oraz jakie zależności pomiędzy nimi występują).

ACL opisuje zagadnienie struktury wiadomości jakie mogą wysyłać i odbierać agenty. Składają się one z opisanego niżej zestawu parametrów:

- `performative` – jedyny wymagany parametr wiadomości ACL. Określa typ *aktu mowy*, jakim jest dana wiadomość [2],
- `sender` – nadawca; parametr standardowy choć nieobowiązkowy,
- `receiver` – adresat w postaci niepustego zbioru nazw agentów; parametr standardowy choć nieobowiązkowy,

- `reply-to` – adresat odpowiedzi na daną wiadomość,
- `content` – treść wiadomości, do zinterpretowania przez agenta-adresata; parametr standardowy choć nieobowiązkowy,
- `language` – język, w którym zapisana jest treść wiadomości; parametr może zostać pominięty jeśli agent-adresat zna język treści danego systemu agentowego lub potrafi go zinterpretować,
- `encoding` – precyzuje kodowanie treści wiadomości,
- `ontology` – precyzuje ontologię do interpretacji treści wiadomości,
- `protocol` – precyzuje protokół interakcji, w którym generowana jest wiadomość ACL; parametr nieobowiązkowy choć zalecany. Wiadomość z niepustym parametrem protocol uważana jest za część *konwersacji* i musi mieć niepusty parametr *conversation-id*, jednakowy dla wszystkich wiadomości w danej konwersacji oraz niepusty parametr `reply-by`,
- `conversation-id` – zawiera identyfikator, przypisujący wiadomość do konwersacji,
- `reply-with` – parametr skierowany do agenta-adresata, który powinien zostać zamieszczony w polu `in-reply-to` w odpowiedzi na daną wiadomość,
- `in-reply-to` – parametr *w odpowiedzi na*, zawierający wartość z pola `reply-with` z wiadomości, na którą bieżaca wiadomość stanowi odpowiedź,
- `reply-by` – precyzuje czas, w którym agent wysyłający daną wiadomość życzy sobie otrzymać na nią odpowiedź.

Poniżej zamieszczono przykładową wiadomość ACL:

```
(inform
      :sender agent1
      :receiver hpl-auction-server
      :content (price (bid good02) 150)
      :in-reply-to round-4
      :reply-with bid04
      :language sl
      :ontology hpl-auction
      )
```

Jak widać na przykładzie, parametr obowiązkowy `performative` jest określany na samym początku wiadomości (tu jego wartość to `inform`).

Protokoły interakcji

Współpraca i rywalizacja pomiędzy agentami jest kluczową cechą systemów wieloagentowych. Strategie interakcji między agentami formułowane są jako protokoły interakcji. Protokoły interakcji określają scenariusze wymiany komunikatów, budowę oraz wartości konkretnych pól komunikatu. Mogą one być implementowane przez programistę, można także wykorzystać implementacje dostępne w specjalistycznych systemach, służących do tworzenia systemów wieloagentowych. Do najważniejszych protokołów interakcji należą:

- `FIPA-Query`, `FIPA-Request` – proste protokoły pozwalające jednemu agentowi zgłaszać żądanie wykonania jakiejś akcji do innego agenta. Adresat żądania może wykonać akcję bądź odmówić jej wykonania, w każdym wypadku informując agenta inicjatora o rezultacie swojej decyzji, wynikach wykonania żądanej akcji. `FIPA-Request` umożliwia zlecenie wykonania akcji dopiero gdy zadane warunki wykonania będą spełnione,
- `FIPA-Propose` – zgodnie z tym protokołem agent składa ofertę wykonania jakiejś akcji innemu agentowi. Adresat propozycji może zrezygnować z oferty lub ją przyjąć, w takim wypadku następuje zazwyczaj dalsza komunikacja mająca na celu wykonanie proponowanej akcji,
- `FIPA-Subscribe` – protokół umożliwiający agentowi wysłanie żądania wykonywania akcji do innego agenta gdy stan określonego obiektu ulega zmianom. Agent adresat może odmówić subskrybowania usługi lub zgodzić się na nią, informując o swojej decyzji nadawcę komunikatu, natomiast agent wykonujący usługę informuje o wynikach wykonania i stanie referowanego obiektu agenta, który ją sybskrybował,
- `FIPA-Contract-Net` – bardziej złożony protokół, zazwyczaj angażujący w interakcję większą ilość agentów. Jeden agent inicjator, chcąc aby jakieś zadanie zostało wykonane, wysyła do jednego lub wielu agentów zapytanie o ofertę wykonania zadania wraz z jego charakterystyką (*call for Proposal*). Poszczególni agenci składają swoje oferty przesyłając odpowiedź do inicjatora, który następnie wybiera najlepszą (lub kilka najlepszych) i zleca odpowiednim agentom wykonanie zadania.

13.4. Środowiska programowania agentowego

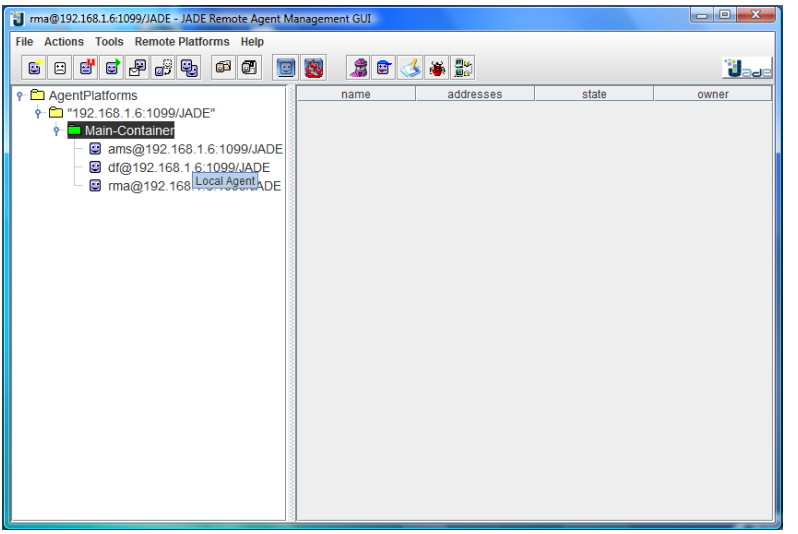
W ciągu ostatnich lat stworzonych zostało dużo narzędzi oraz środowisk wspomagających projektowanie i tworzenie systemów agentowych. Wykazują one szereg cech, takich jak np. zgodność ze standardami FIPA, rodzaj licencji, na której oprogramowanie jest rozprowadzane, język programowania używany do tworzenia systemów agentowych i wiele innych. Dla przeprowadzenia testów opisanych w dalszej części rozdziału wybrano środowisko *JADE* [3].

13.4.1. JADE

JADE (ang. *Java Agent DEvelopment Framework*) jest systemem oprogramowania, ułatwiającym tworzenie systemów agentowych i wieloagentowych (zobacz rys. 13.2). Oparty jest na licencji *LGPL v.2*. Umożliwia programowanie w języku Java i stąd działa w jakimkolwiek środowisku wspierającym tę platformę. Jest zgodny ze standardami *FIPA*. Składa się z trzech podstawowych części:

- systemu czasu wykonania – środowiska, w którym „żyją" programy agentów (`jade.Boot`),
- biblioteki klas, które są wykorzystywane do oprogramowannia podstawowych „funkcji życiowych" agentów,

- zestawu narzędzi (programów) umożliwiających monitorowanie i administrowanie wszytkimi elementami infrastruktury JADE.



Rys. 13.2: Interfejs JADE.

Rodzaje agentów w JADE

Jako elementy systemu czasu wykonania wyróżnić można **platformy** i **kontenery**. Kontener jest działającą instancją środowiska, w którym działają agenci. Platforma jest natomiast zbiorem aktywnych instancji (uruchomionych na jednym komputerze lub na kilku komputrach połączonych siecią). Każda platforma zawiera przynajmniej jeden kontener, pierwszy z uruchomionych określany jest jako główny kontener (ang. *main container*). Posiada on zawsze dwóch specjalnych agentów wynikających ze struktury systemu agentowego FIPA (rys. 13.1):

- AMS (ang. *Agent Management System*) – realizuje usługi zarządzania regułami współżycia agentów na platformie, np. unikalnością nazw agentów,
- DF (ang. *Directory Facilitator*) – dostarcza usługi katalogowe, agenci mogą rejestrować swoje usługi u agenta DF aby udostępnić je innym agentom.

JADE zawiera także innych specjalnych agentów ułatwiających tworzenie i debugowanie systemów agentowych. Są to:

- RMA (ang. *Remote Management Agent Gui*) – graficzna konsola do zarządzania platformą i kontrolą cyklu życia agentów. Umożliwia m.in. tworzenie i likwidację obiektów agentów. Na jednej platformie może być wielu agentów RMA ale w danym kontenerze tylko jeden,
- *DummyAgent* – narzędzie do monitorowania i debugowania. Możliwe jest tworzenie i wysyłanie wiadomości do pozostałych agentów,

- *The Sniffer* – agent mający możliwość przechwycenia wiadomości jakie wymieniają między sobą agenci i wyświetlenia jej użytkownikowi w formie podobnej do diagramu UML. Jest użyteczny przy odpluskwianiu społeczności agentów poprzez możliwość obserwacji komunikatów jakie wymieniają między sobą,
- *Introspector* – agent umożliwiający monitorowanie cyklu życia agenta, jego wymienianych wiadomości i wykonywanych zadań,
- *SocketProxyAgent* – prosty agent służący jako dwukierunkowa brama między platformą JADE a łączem TCP/IP.

Zachowania, klasy agentów w JADE

Tworzenie agentów w środowisku JADE polega na rozszerzeniu specjalnej klasy `jade.core.Agent` i zaimplementowaniu metody `setup()` (wywoływanej po utworzeniu obiektu agenta) i `takeDown()` (wywoływanej po likwidacji obiektu agenta). Każdy agent jest osobnym wątkiem, na platformie identyfikowanym przez nazwę będącą ciągiem znaków, natomiast globalnie poprzez odpowiedni identyfikator: `<nazwa_agenta>@<nazwa_platformy>`.

Zadania, jakie ma wykonywać agent, implementowane są w ramach jego **zachowań**. Każde z nich przypisywane jest za pomocą metody `addBehaviour()` (powinna ona być wywoływana w metodzie `setup()`). Biblioteki systemu JADE dostarczają klas modelujących charakterystyczne zachowania. Są to:

- `SimpleBehaviour` – modeluje pojedyncze zachowanie; dziedziczy po klasie Behaviour,
- `OneShotBehaviour` – modeluje zachowanie, które może być wykonane tylko raz i nie może być blokowane,
- `CyclicBehaviour` – modeluje pojedyncze zdarzenie, które musi być wykonywane zawsze,
- `CompositeBehaviour` – modeluje zachowanie, które składa się z podzachowań definiujących aktualnie wykonywane operacje; jest to klasa bazowa dla klas modelujących złożone zachowania,
- `SequentialBehaviour` – podklasa klasy CompositeBehaviour, która wykonuje swoje podzachowania sekwencyjnie, i kończy działanie, kiedy zakończyły się wszystkie podzachowania; ta klasa wykorzystywana jest, wtedy gdy złożone zadanie może być wyrażone sekwencją pojedynczych kroków,
- `WakerBehaviour` – klasa implementująca jednorazowo wykonywane zadanie po upłynięciu określonego czasu,
- `TickerBehaviour` – klasa implementuje cykliczne zachowanie wykonywane okresowo,
- `FSMBehaviour` – klasa wykonująca podzachowania zgodnie z przyjętym modelem skończonej maszyny stanów, zdefiniowanej przez użytkownika; podzachowania reprezentują stany, użytkownik definiuje przejścia między nimi.

Niektóre zachowania mogą być zarejestrowane jako stany końcowe. Zachowanie jako całość kończy się po osiągnięciu jednego z takich stanów.

Ważną cechą platformy JADE jest możliwość porozumiwania się agentów między sobą. Agenci mogą wymieniać komunikaty zgodnie ze standardem ACL,

w sposób asynchroniczny. Każdy agent ma swoją kolejkę wiadomości, do której trafiają informacje do niego zaadresowane. Gdy przychodzi wiadomość, agent jest powiadamiany o tym fakcie. Standard ACL określa parametry opisujące wiadomość: nadawcę, odbiorcę, intencję komunikacyjną (`performative`), język w jakim wyrażona jest wiadomość, zawartość i szereg innych. Komunikat reprezentowany jest jako obiekt klasy `jade.lang.acl.ACLMessage`.

13.5. Przykład zastosowania MAS

W celu zbadania możliwości zastosowania systemu agentowego w praktyce stworzono projekt systemu agentowego do wyszukiwania aukcji samochodów osobowych o zadanych przez użytkownika kryteriach na kilku portalach aukcyjnych. Poniżej przedstawiono założenia oraz sposób realizacji tego projektu.

13.5.1. Założenia

System służyć ma wyszukiwaniu pojazdu według zadanych kryteriów. Wyszukiwanie realizowane ma być przez agentów, zaś rezultaty ich pracy powinny być zapisane do odpowiednich plików tekstowych.

W systemie wyróżniono: agenta-asystenta oraz agentów-szperaczy, przeszukujących serwisy aukcyjne. Zadaniem agenta-asystenta jest komunikacja z użytkownikiem za pomocą graficznego interfejsu użytkownika. Interfejs umożliwia zdefiniowanie takich parametrów wyszukiwania takich jak: serwis aukcyjny, który ma zostać przeszukany, marka samochodu, jego cena minimalna oraz maksymalna. Po zdefiniowaniu kryteriów i zatwierdzeniu ich odpowiednim przyciskiem agenci-szperacze rozpoczynają przeszukiwać serwisy aukcyjne na bazie przekazanych parametrów. Agent przeszukujący po zakończeniu pracy wysyła wiadomość do agenta-asystenta z informacją o tym fakcie. Po otrzymaniu takiej wiadomości agent-asystent daje znać użytkownikowi o zakończeniu pracy agenta poprzez wyświetlenie odpowiedniego okienka informacyjnego.

System obsługuje trzy serwisy aukcyjne: Allegro.pl, Aukro.cz oraz Aukro.sk. Do każdego z tych systemów zostaje przydzielony jeden agent przeszukujący, stąd cały system tworzy czterech agentów. Użytkownik może,wybrać, które serwisy są przeszukiwane. W celu uniknięcia wystąpienia sytuacji, w której czas oczekiwania na efekty pracy agentów przeciągnie się w nieskończoność, zostały nałożone ograniczenia na czas odpowiedzi ze strony agentów przeszukujących oraz na ilość uzyskanych wyników. Po przekroczeniu jednej z wartości maksymalnych agent-asystent wysyła do pozostałych agentów komunikat nakazujący przerwanie pracy i zwrócenie dotychczas uzyskanych wyników. Należy przez to rozumieć wyodrębnienie z pozyskanych informacji tych, które spełniają wszystkie ograniczenia i ich zapis do odpowiednich plików tekstowych.

13.5.2. Narzędzia programistyczne

W celu realizacji projektu skorzystano z systemu JADE, który ułatwia tworzenie systemów agentowych. System ten ma otwarty charakter (do użytku nieko-

mercyjnego) i bogatą dokumentację z dużą liczbą przykładowych kodów źródłowych. Dzięki tym cechom łatwo jest poznać samo środowiskiem oraz nauczyć się tworzyć własne rozwiązania.

Ponieważ JADE jest systemem napisanym w języku Java, w trakcie pracy nad projektem skorzystano ze środowiska Netbeans. O jego wyborze zadecydowały przede wszystkim popularność, jaką się cieszy, oraz otwarty charakter. Istotnym czynnikiem był też szeroki dostęp do różnych tutoriali, pokazujących jakie możliwości ma to środowisko i jak z nich skorzystać. Nie bez znaczenia był też fakt istnienia tutoriali, pokazujących jak zintegrować Netbeans z platformą JADE oraz jak stworzyć aplikację do komunikacji z serwerami portali aukcyjnych – `WebServiceClient`. Docelowa aplikacja została stworzona w środowisku Windows.

13.5.3. Implementacja

Metoda przeszukiwania serwisów

Serwisy aukcyjne Aukro.cz i Aukro.sk, wraz z wieloma innymi serwisami zagranicznymi, są członkami Grupy Allegro. Serwisy te można przeszukiwać przy użyciu `Allegro WebAPI` – usługi sieciowej o znanym interfejsie, która umożliwia wymianę informacji między zasobami serwisów internetowych Grupy Allegro a oprogramowaniem zewnętrznym [4]. Interfejs usługi jest opisany w języku definiowania usług sieciowych WSDL (ang. *Web Services Description Language*), komunikacja zaś odbywa się z pośrednictwem protokołu SOAP (ang. *Simple Object Access Protocol*) .

Na początku działania aplikacji przy pomocy metody `doLogin` odbywa się logowanie, przy czym uzyskiwany jest klucz sesji ważny 3 godziny [5]. Warto nadmienić, że dla uniknięcia niepotrzebnego logowania przy wciąż ważnym kluczu sesji tworzony jest plik `logowanie.txt`, który przechowuje klucz sesji z poprzedniego użycia programu oraz czas serwera, pozwalający sprawdzić, czy upłynęły od tej chwili trzy godziny.

Następnie wywoływana jest funkcja `doSearch`, do której parametrów należą klucz sesji oraz odpowiednio skonstruowane zapytanie, zawierające szukany ciąg znaków, numer kategorii (odpowiadający marce samochodu), cenę minimalną i maksymalną, odpowiedni numer serwisu aukcyjnego oraz parametry, takie jak np. „`tylko Kup Teraz`".

Realizacja systemu agentowego

W ramach systemu agentowego służącego do wyszukiwania aukcji samochodów osobowych wyróżniono dwa typy agentów: agent-asystent i agent-szperacz.

Agent-asystent tworzony jest jako obiekt klasy `AgentAsystent`. Jego zadaniem jest komunikacja z użytkownikiem poprzez graficzny interfejs użytkownika, tworzenie agentów przeszukujących serwis aukcyjny Allegro oraz komunikacja z nimi.

Agenci-szperacze są instancjami klasy `AgentSzperacz`. Oprócz poszukiwania danej marki samochodu, wyświetlają parametry przeszukiwania zdefinio-

wane przez użytkownika oraz inicjują komunikację z agentem-asystentem. Ten, w zależności od dokonanego wyboru tworzy od 1 do 3 agentów przeszukujących. Każdy agent-szperacz po swoim stworzeniu wyświetla parametry wyszukiwania i zaczyna przeszukiwać serwis. Po zakończeniu przeszukiwania agent-szperacz szuka na platformie agenta-asystenta. Po jego odnalezieniu wysyła mu wiadomość o treści „Koniec szukania". Agent-asystent odpowiada wiadomością o treści „Koniec pracy", po ktorej odebraniu agent-szperacz wywołuje metodę likwidującą obiekt agenta.

Rolą agenta-asystenta po wysłaniu poleceń wyszukiwania agentom przeszukującym jest cykliczne sprawdzanie, czy nie nadeszła wiadomość od jakiegoś agenta przeszukującego o treści „Koniec szukania". Jeśli taka sytuacja miała miejsce, to wysyłana jest wiadomość o treści „Koniec pracy". Po jej otrzymaniu agent-szperacz wywołuje metodę, które likwiduje obiekt agenta.

Po zlikwidowaniu wszystkich agentów-szperaczy można ponownie je uruchomić naciskając przycisk „Zaczynaj". Zamknięcie głównego okna aplikacji pociągnie za sobą likwidację agenta-asytenta. Agent wyrejestruje się, zamyka główne okno aplikacji, po czym kończy swój żywot.

13.6. Podsumowanie

Systemy agentowe dają niespotykane dotąd możliwości programistyczne. Są elastyczne, przenośne i w dużym stopniu odporne na błędy. Posiadają jednak także wady. Ponieważ powstały stosunkowo niedawno, nie istnieją pełne i ujednolicone standardy programowania, a istniejące platformy programistyczne we wszystkich najważniejszych językach programowania nie pozwalają na pełną integrację systemów agentowych z innymi usługami (co znacznie może utrudnić realizację projektu MAS). Szybki rozwój tej dziedziny w połączeniu ze znacznymi możliwościami, jakie daje ona programistom, rokuje jednak nadzieje na możliwość szybkiego, wygodnego i efektywnego programowania agentowego.

Literatura

[1] F. for Intelligent Physical Agents: Specyfikacje FIPA. `http://www.fipa.org/specifications/index.html`, (2005).

[2] J. Searle: *Speech Acts: An Essay in the Philosophy of Language* Cambridge University Press, (1969).

[3] J. A. D. F. Team: Dokumentacja JADE. `http://jade.tilab.com/doc/index.html`, (2010).

[4] G. Allegro: Baza wiedzy Allegro WebAPI. `http://allegro.pl/webapi/general.php`, (2010).

[5] G. Allegro: Dokumentacja pakietów Allegro WebAPI. `http://webapi.allegro.pl/uploader.php`, (2010).

ROZDZIAŁ 14

METODY OCENY RYZYKA

S.Gospodarek, J.Urbanowicz

14.1. Wstęp

Termin ryzyko (risk) wywodzi się z języka włoskiego (wł. *risico*), w którym oznacza przede wszystkim przedsięwzięcie, którego wynik jest nieznany albo niepewny lub oznacza prawdopodobieństwo tego, że coś się uda lub nie [1, s. 9]. Ryzykiem można nazwać również stan, w którym rezultat osiągnięty w przyszłości jest nieznany, ale można zidentyfikować jego przyszłe alternatywy, przy założeniu, że szanse wystąpienia możliwych alternatyw są znane. Precyzyjna definicja ryzyka została podana w [2, s. 4].

Ryzyko – szansa jakiegoś wydarzenia, które będzie miało wpływ na realizację zamierzonego celu.

Ocena ryzyka występuje w bardzo wielu dziedzinach życia. Przede wszystkim stosuje się ją w instytucjach finansowych takich jak banki, ubezpieczalnie oraz domy maklerskie, gdzie aplikacja ocenia ryzyko inwestycji, pomaga podjąć decyzję o manipulacji kapitałem bądź też umożliwia rozumienie efektywności danego przedsięwzięcia uwarunkowanego probabilistyczne. Często ryzyko zależy od bardzo wielu czynników i nie ma możliwości obserwacji ich wszystkich, co gorsza, za każdym razem badane czynniki mogą mieć inny wpływ lub też sama ilość zjawisk składających się na ryzyko może się zmieniać.

Mimo licznych problemów zarządza się ryzykiem. Celem zarządzania ryzykiem jest jego minimalizacja oraz zabezpieczenie przed następstwami badanego zjawiska. Chcąc sprawnie zarządzać ryzykiem i np. minimalizować prawdopodobieństwo poniesienia strat, potrzebny jest aparat matematyczny do jego oceny.

Model zarządzania ryzykiem podany w [3, s. 4] składa się z następujących etapów: identyfikacji, oceny, planowania, monitorowania, sterowania, komunikacji.

Na etapie identyfikacji znajduje się rodzaje ryzyka jakie mogą wystąpić w rozważanej sytuacji. Na etapie oceny określa się poziom ryzyka korzystając z różnych metod. Dzięki temu etapowi można dokonać gradacji czynników wpływających na rozważany problem, zwracając uwagę na czynniki bardziej istotne. Sterowanie to etap, w którym dokonuje się działań mających na celu ograniczenie

ryzyka do dopuszczalnych rozmiarów lub ochronę przed następstwem rozważanej sytuacji. Monitorowanie ma na celu badanie efektywności działań podejmowanych na etapie sterowania.

14.2. Polskie normy

Ryzyko jest ogólnym pojęciem i w zależności od dziedziny, branży czy też sytuacji ma inny wydźwięk. Z tego też powodu w większości przypadków nie da się wyznaczyć dokładnych norm, metod lub reguł określających poziom, zakres i inne cechy ryzyka. Często sposoby określania ryzyka są informacją zastrzeżoną i o dużej wartości, jak ma to miejsce przy ocenie ryzyka inwestycji w bankach czy też biurach maklerskich. W polskich normach [4, 5, 6] można spotkać wyłącznie zalecenia dotyczące oceny ryzyka zawodowego i sposobów jego zmniejszania. Jednak i tu, ze względu na mnogość różnych przypadków, norma jest raczej ogólna oraz prezentuje sposób podejścia do problemu, a nie dokładny przepis.

14.2.1. Ocena ryzyka

Polskie prawo nakazuje ocenę ryzyka zawodowego. Pracodawca bądź też pracownik BHP jest odpowiedzialny za prawidłową ocenę ryzyka jak i ewentualną pracę nad jego zmniejszaniem. Ocena powinna być przeprowadzona w możliwie najprostszy sposób oraz wskazane jest, aby identyfikacja zagrożeń była oparta na zasadzie zdrowego rozsądku. Po dokonanej analizie ocena pozwoli określić czy ryzyko pracy jest dopuszczalne czy niedopuszczalne. Można przyjąć, że ryzyko jest dopuszczalne tylko jeśli zastosowane środki chroniące pracownika przed oddziaływaniem czynników niekorzystnych są wystarczające, w przeciwnym wypadku ryzyko jest niedopuszczalne i nie można prowadzić pracy.

14.2.2. Metody oceny ryzyka

Polskie normy podają kilka metod, które pomagają w ocenie ryzyka. Metody na podstawie prawdopodobieństwa oraz skutków wystąpienia wypadku (określenie negatywnych następstw zdarzenia – w podejściu humanitarnym: urazu czy też śmierci, w podejściu materialnym: strat finansowych; określenie może dotyczyć pracownika, pracowników bądź społeczeństwa) wyznaczają poziom ryzyka. Głównie są to tabele, w kolumnach i wierszach których przyrostowo wypisano prawdopodobieństwo oraz zakres czy też skutki wypadku.

Metoda oceny trójstopniowej i pięciostopniowej

Metoda ta jest raczej przedstawieniem działania kolejnych bardziej szczegółowych metod opisanych w dalszej części. Określa się w niej prawdopodobieństwo i wagę następstw wypadków i na ich podstawie odczytuje się wartość ryzyka z tab. 14.1. Jeśli znany jest poziom ryzyka, można odczytać niezbędne działania z tab. 14.2 oraz to czy ryzyko jest dopuszczalne czy też nie. Wszystko opiera się zatem na zdrowym rozsądku i wymagane jest doświadczenie oraz dokładna zna-

jomość badanego problemu, w innym przypadku ocena może być bezużyteczna lub co gorsze może wpłynąć na bezpieczeństwo.

Tab. 14.1: Poziomy ryzyka dla skali trzystopniowej i pięciostopniowej.

Prawdopodobieństwo wystąpienia możliwych zagrożeń	Ciężkość następstw zagrożeń		
	MAŁA	ŚREDNIA	DUŻA
MAŁE	MAŁE	MAŁE	ŚREDNIE
ŚRENIE	MAŁE	ŚREDNIE	DUŻE
DUŻE	ŚREDNIE	DUŻE	DUŻE

Prawdopodobieństwo wystąpienia możliwych zagrożeń	Ciężkość następstw zagrożeń		
	MAŁA	ŚREDNIA	DUŻA
MAŁE	BARDZO MAŁE	MAŁE	ŚREDNIE
ŚRENIE	MAŁE	ŚREDNIE	DUŻE
DUŻE	ŚREDNIE	DUŻE	BARDZO DUŻE

Dla skali trójstopniowej występują następujące poziomy ryzyka: małe, średnie i duże. Skala pięciostopniowa jest dodatkowo rozszerzona o poziomy bardzo małego i bardzo dużego ryzyka, pozwala to na wyznaczenie dodatkowych poziomów i szerszy opis. Trzeba jednak pamiętać, że szersza skala wymaga lepszej znajomości problemu. Przykład pozwoli lepiej zrozumieć opisane zagadnienie:

Stanowisko pracy:
- kierowca samochodu dostawczego, dla zagrożenia wypadkiem komunikacyjnym.
Ciężkość następstw:
- duża (ewentualność ciężkich dolegliwości, a nawet śmierci)
Prawdopodobieństwo:
- małe (firma nie odnotowała wypadków komunikacyjnych przez ponad 20 lat, obsługuje głównie sklepy osiedlowe w mieście do 20 tys. mieszkańców)
Oznacza to ryzyko zawodowe na poziomie średnim, czyli dopuszczalne lecz wymagające planowania i realizacji działań ukierunkowanych na jego zmniejszenie.

Metoda wstępnej analizy zagrożeń

Metoda wstępnej analizy zagrożeń pozwala na jakościowe oszacowanie ryzyka. Posiadając dane statystyczne oraz wykorzystując własne spostrzeżenia można łatwo wartościować ryzyko. Tak jak w przypadku poprzedniej ogólnej metody ocena zależy od dwóch parametrów:

- S - wielkości (stopnia) ewentualnej szkody,

Tab. 14.2: Wykaz działań w zależności od ryzyka.

Poziom ryzyka	Wartościowanie ryzyka	Niezbędne działania
DUŻY	NIEDOPUSZCZALNE	Jeżeli ryzyko zawodowe związane jest z pracą już wykonywaną, działania zmniejszające ryzyko trzeba podjąć natychmiast. Planowana praca nie może być rozpoczęta do czasu zmniejszenia ryzyka.
ŚREDNI MAŁY	DOPUSZCZALNE	Zalecane jest kontrolowanie i dbanie o to by wskaźnik bezpieczeństwa stale się polepszał.

- P - prawdopodobieństwa powstania takiej szkody.

Parametr wielkość szkody (S) przyjmuje wartości 1 - 6 według następującego zestawienia:

1. niewielka, znikome urazy, szkody nieznaczne,
2. lekkie obrażenia, szkody wymierne,
3. ciężkie obrażenia, szkody znaczne,
4. wypadek śmiertelny jednej osoby, szkody ciężkie,
5. wypadek śmiertelny zbiorowy, bardzo ciężkie szkody na terenie przedsiębiorstwa,
6. wypadek śmiertelny zbiorowy, bardzo ciężkie szkody poza terenem przedsiębiorstwa.

Prawdopodobieństwo powstania szkody (P) przyjmuje wartości 1 - 6 według następującego zestawienia:

1. nieprawdopodobne,
2. mało prawdopodobne, szkoda powstaje raz na 10 lat,
3. szkoda może się wydarzyć raz w roku,
4. dosyć częste, szkoda może się wydarzyć raz w miesiącu,
5. częste, szkoda może się wydarzyć raz na tydzień,
6. bardzo prawdopodobne.

Po oszacowaniu parametrów S i P ryzyko jest określane (wartościowane) na trzech poziomach według tab. 14.3:

od 1 do 3 – akceptowalne,
od 4 do 9 – dopuszczalna akceptowalność ryzyka po przeprowadzeniu oceny,
powyżej 10 – ryzyko niedopuszczalne.

W zależności od wartości powinno się podjąć odpowiednie działania. Widać też ile trzeba zmienić by osiągnięty został lepszy poziom bezpieczeństwa. Zobrazuje to poniższy przykład:

Tab. 14.3: Wartościowanie ryzyka w metodzie wstępnej analizy zagrożeń.

		Prawdopodobieństwo powstania szkody (P)					
		1	2	3	4	5	6
Wielkość szkody (S)	1	1	2	3	4	5	6
	2	2	4	6	8	10	12
	3	3	6	9	12	15	18
	4	4	8	12	16	20	24
	5	5	10	15	20	25	30
	6	6	12	18	24	30	36

Stanowisko pracy: bibliotekarz, dla zagrożenia uderzeniem przez spadające przedmioty:
- wielkość szkody S = 2 (lekkie obrażenia)
- prawdopodobieństwo szkody P = 3 (może zdarzyć się raz w roku, w bibliotece panuje porządek)
Oznacza to ryzyko na poziomie 6, czyli dopuszczalna jest akceptacja ryzyka po przeprowadzeniu oceny.

Metoda wskaźnika ryzyka

Metoda ta jest najbardziej rozbudowana z wszystkich przedstawionych, bierze się w niej pod uwagę trzy parametry:

S - możliwych skutków (następstw) zagrożenia,
E - ekspozycji (narażenia) na zagrożenie,
P - prawdopodobieństwa wystąpienia zdarzenia.

Zgodnie z tą metodą poziom ryzyka określa się jako iloczyn parametrów. Tak jak wcześniej istnieją w normach tabele, które pokazują jakie wartości mają poszczególne parametry oraz jak zinterpretować wynik. Przykład użycia metody:

Stanowisko pracy: murarz-tynkarz, dla zagrożenia upadkiem na niższy poziom (np. podczas prac na rusztowaniach lub podnośnikach):
- możliwe skutki zagrożenia S = 15 (bardzo duże, jedna ofiara śmiertelna)
- ekspozycja (narażenie) na zagrożenie E = 6 (częsta - codzienna)
- prawdopodobieństwo wystąpienia zdarzenia P = 0,5 (zdarzenie sporadycznie możliwe, na budowie przestrzega się przepisów)
Oznacza to ryzyko na poziomie 45 ($15 \cdot 6 \cdot 0{,}5$), czyli małe, przy czym potrzebne jest kontrolowanie tego zagrożenia.

14.3. Bezpieczeństwo na drodze

Jednym z głównych celów producentów samochodów jest zapewnienie jak największej ochrony podczas wypadku, czyli nic innego jak zmniejszanie ryzyka utraty zdrowia czy też życia. Jest to jednak operowanie wskaźnikiem wielkości szkody. Chcąc jeszcze bardziej podnieść poziom bezpieczeństwa można zmniej-

szyć prawdopodobieństwo wystąpienia wypadku. Problem tego typu można rozwiązać poprzez umożliwienie sprawdzenia przez kierowcę, która z dróg pomiędzy miejscem startu, a celem cechuje się najmniejszą ilością wypadków – najmniejszym prawdopodobieństwem wystąpienia wypadku.

14.3.1. Identyfikacja problemu

Z danych statystycznych [7] oraz raportów policyjnych można wyznaczyć kilka podstawowych czynników wpływających na prawdopodobieństwo wystąpienia wypadku drogowego. Dodatkowo można stwierdzić, który z tych czynników ma największy wpływ. Kolejno są to:

czynnik ludzki – właśnie głównie z winy człowieka dochodzi do większości wypadków, czynnik ten nie jest jednak brany pod uwagę ponieważ niestety w wypadku zazwyczaj bierze udział jeden „spokojny" kierowca i jeden „brawurowy",

stan dróg – jest to kolejny z czynników, jazda po autostradzie czy drodze z wydzielonymi pasami mijania znacznie zmniejsza ryzyko wypadku, natomiast ruchliwe nieprzystosowane drogi o zniszczonej nawierzchni sprzyjają wypadkom,

pogoda – ostre słońce, deszcz, śnieg i inne czynniki pogodowe mają duży wpływ na bezpieczeństwo,

trasa – długość trasy wpływa na prawdopodobieństwo wystąpienia wypadku, często dłuższa trasa po lepszych drogach będzie bezpieczniejsza,

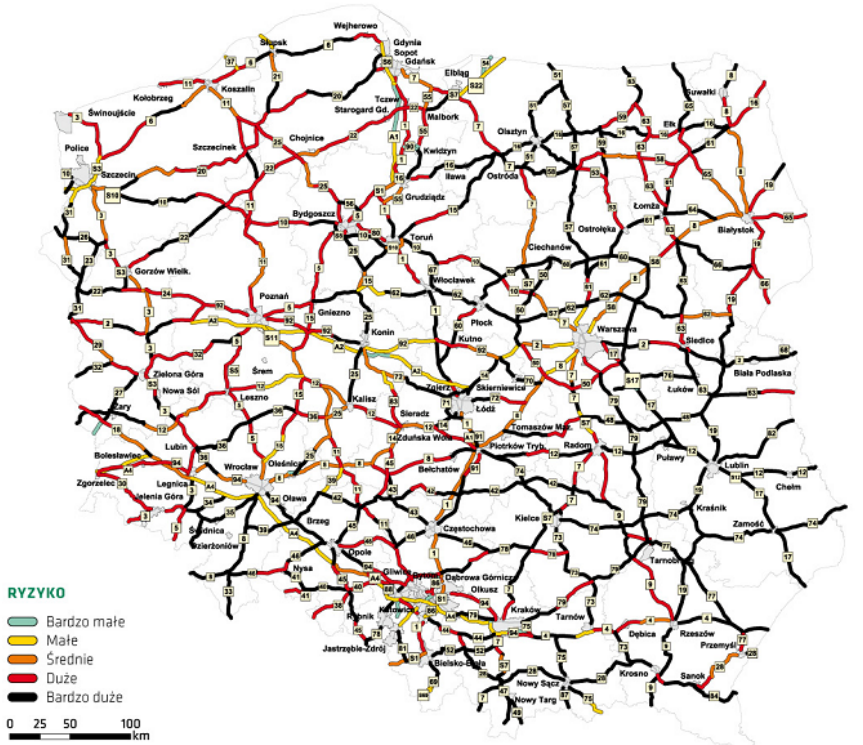
biomet – ogólny stan pogody może wpływać na nasze samopoczucie i koncentrację czyli pośrednio na bezpieczeństwo.

Dane o bezpieczeństwie dróg

Dane dotyczące bezpieczeństwa dróg i ilości wypadków oraz ich rodzaju są dostępne w Atlasie ryzyka na drogach w Polsce [8]. Przykładowa mapa, obrazująca bezpieczeństwo na drogach pod względem wszystkich możliwych wypadków i ich skutków, wygląda bardzo źle (rys. 14.1) – ilość „czarnych" odcinków jest znaczna. Na mapie uwzględniono, między innymi, wypadki takie jak: zderzenia czołowe, zderzenia boczne, wypadnięcie z drogi i najechanie pieszego.

Dane pogodowe

Dane pogodowe są dostępne w wielu miejscach w Internecie. Przy wyznaczaniu bezpieczeństwa brane są pod uwagę takie elementy jak opady deszczu i śniegu czy też mgła na trasie. Oczywiście zależnie od trasy warunki pogodowe mogą być inne i inaczej mogą wpływać na bezpieczeństwo. Wymienione elementy pokazano na przykładowych mapach pogodowych (rys. 14.2).



Mapa 14.1: Ogólna mapa bezpieczeństwa na drogach w Polsce, dane z lat 2007-2009.

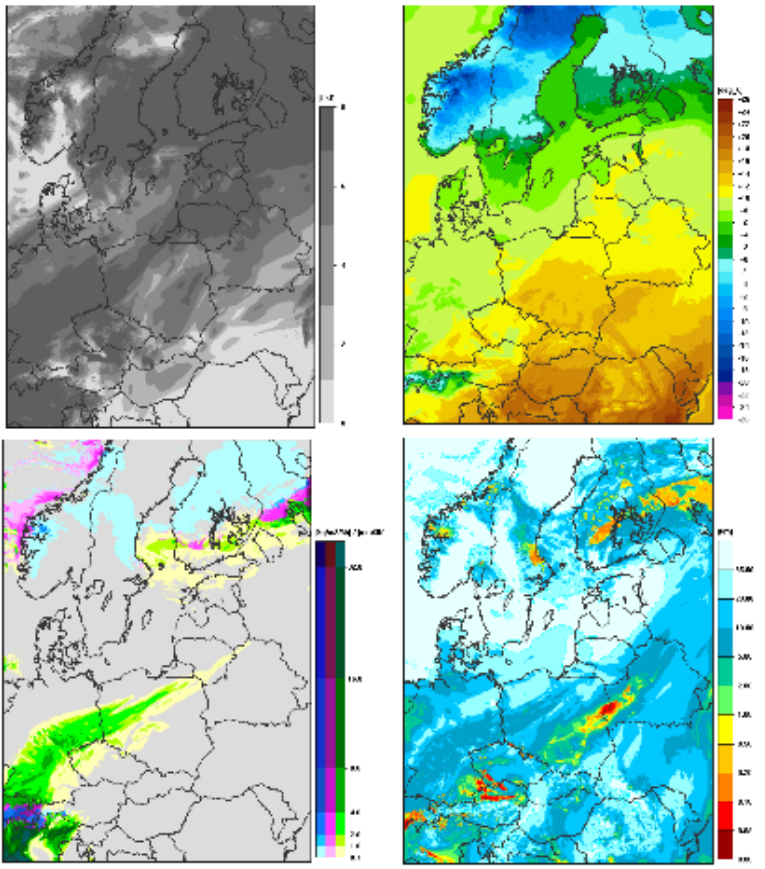
Dane o trasie

Wpływ na bezpieczeństwo ma również długość trasy. Można jechać dłuższą trasą, ale mimo to będzie ona bezpieczniejsza. Oczywiście czasem jazda dłuższą i nieco bezpieczniejszą może wcale nie być lepsza ze względu na ilość dodatkowych kilometrów do pokonania.

Do wyznaczania trasy pomiędzy punktami można wykorzystać internetową Mapę Google. Jest to wygodne narzędzie, które posiada wiele ciekawych funkcji. Istnieje możliwość pisania aplikacji i umieszczania ich na dowolnym serwerze (rys. 14.3).

Aplikacja Google Maps posiada bardzo wiele możliwości, m.in.:

- powiększenie/pomniejszenie interesującego fragmentu mapy,
- zmianę widoku mapa/satelita/hybrydowy/teren,
- dodanie mapy statycznej lub dynamicznej (z możliwością zmiany widoku),
- dodawanie linii i znaczników,
- wyświetlanie informacji na mapie,
- możliwość geokodowania adresów,
- wyznaczanie tras przejazdów,
- możliwość podglądu wybranych miejsc z poziomu ulicy (Street View),

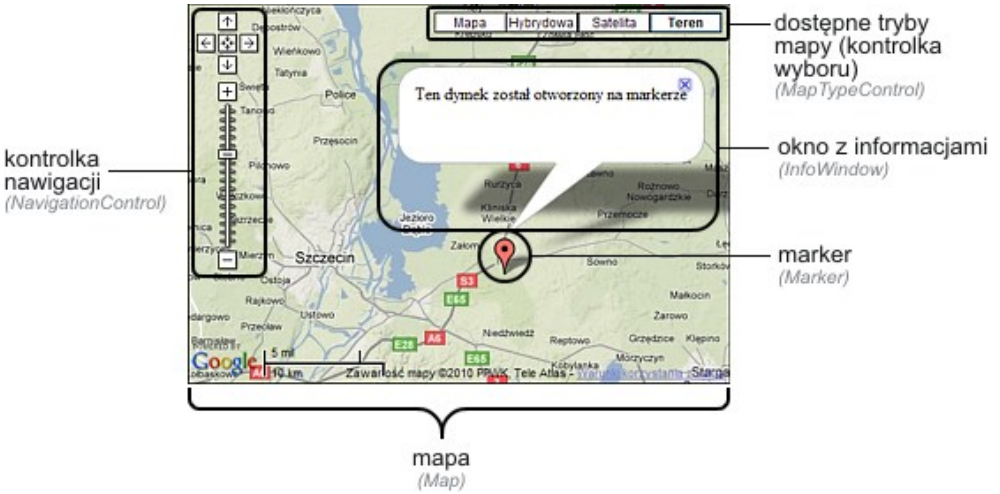


Mapa 14.2: Przykładowe mapy pogodowe – serwis instytutu ICM.

- możliwość wyszukania żądanego połączenia komunikacji miejskiej w wybranych miastach (Google Transit),
- optymalizację pod względem działania na urządzeniach przenośnych.

14.3.2. Ocena bezpieczeństwa na drodze

W podrozdziale 14.2.1 wspomniano, że normy pozwalają ogólnie określić poziom ryzyka i to raczej w sposób opisowy niż matematyczny. Dodatkowo normy pokazują, że są dwa podstawowe kryteria przy ocenie ryzyka: ciężkość wypadku oraz prawdopodobieństwo, z jakim ten wypadek może wystąpić (podrozdział 14.2.2). Problem oceny ryzyka na drodze nie polega na dokładnym określeniu poziomu bezpieczeństwa na danej trasie, ale raczej stwierdzeniu, która z tras jest najbezpieczniejsza. Obierając odpowiednie współczynniki - ich ilość i wartość - można, podobnie jak w normach, analizować ryzyko.



Rys. 14.3: Wygląd okna w usłudze Google Maps.

Bezpieczeństwo na wybranej przez użytkownika trasie może być wyznaczane na podstawie aktualnych, historycznych lub też statystycznych danych. Przy określeniu ryzyka brane pod uwagę powinny być wcześniej wymienione czynniki wpływające na bezpieczeństwo (dane pogodowe) z odpowiednią wagą. Wyznaczony poziom ryzyka na najkrótszej trasie może być porównywany z innymi dłuższymi trasami (dane o trasie). W ten sposób może zostać wyznaczona najbezpieczniejsza trasa. Takie rozwiązanie pozwala nie podawać wartości ryzyka, która nie odniesiona do niczego, nie stanowi żadnej informacji. W końcu wybiera się najbezpieczniejszą trasę, a nie zastanawia nad tym tym czy w ogóle jechać.

14.4. Strona internetowa – ocena bezpieczeństwa trasy

Jednym ze sposobów na udostępnienie kierowcom możliwości sprawdzenia bezpieczeństwa tras byłoby stworzenie dedykowanej do tego zadania aplikacji webowej. Zadaniem tej aplikacji, oprócz wyznaczania trasy pomiędzy punktami, byłoby sprawdzenie, która z zaproponowanych tras jest obarczona najmniejszym ryzykiem wystąpienia wypadku. W dalszej części rozdziału opisano testową implementację takiej aplikacji.

14.4.1. Obliczanie bezpieczeństwa trasy

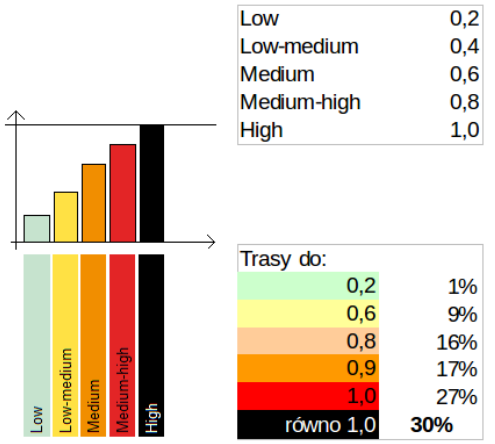
Do oceny poziomu bezpieczeństwa zastosowano funkcję wagową pozwalającą porównywać trays między sobą i najlepiej odzwierciedlają sytuację na polskich drogach.

$$B_i(x_{1i}, x_{2i}, x_{3i}, x_{4i}, x_{5i}, x_i) = \frac{1}{x_i}(x_{1i}b_1 + x_{2i}b_2 + x_{3i}b_3 + x_{4i}b_4 + x_{5i}b_5) \qquad (14.1)$$

gdzie: B_i to bezpieczeństwo analizowanej trasy, $x_{1i}...x_{5i}$ to długość trasy o poziomie bezpieczeństwa od 1 - bezpieczna do 5 - niebezpieczna, x_i to długość całej trasy, $b_1...b_5$ to współczynnik bezpieczeństwa z przedziału 0-1, i to numer badanej trasy. Najbezpieczniejsza trasa to ta, dla której wartość wyznaczona ze wzoru 14.1 jest najmniejsza.

Wyznaczanie wartości współczynników bezpieczeństwa

Współczynniki bezpieczeństwa $[b_1...b_5]$ we wzorze 14.1 trzeba dobrać zdroworozsądkowo i doświadczalnie. Jest to trudne ze względu na fakt, iż powinna być zachowana równowaga pomiędzy długością trasy, a jej bezpieczeństwem. Na rys. 14.4 przedstawiono sytuację, w której współczynniki zostały dobrane liniowo. Czyli kolejne poziomy bezpieczeństwa są liniowo zwiększane o tę samą wartość. Widać jednak, iż tylko niewielki procent tras jest wyróżnionych jako bezpieczne, a podczas testów praktycznie zawsze wybierana była trasa krótsza.



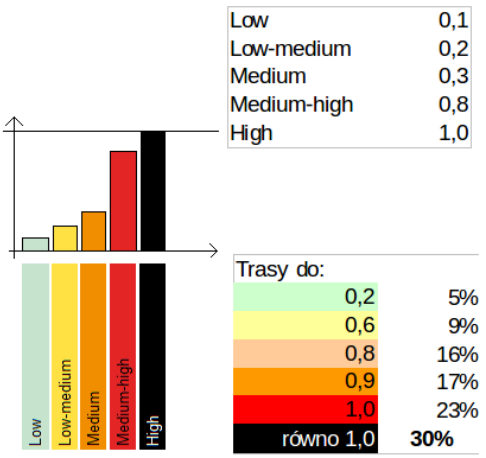
Rys. 14.4: Dobór parametrów - liniowy

Kolejny z podziałów został przedstawiony na rys. 14.5. Jednak i w tym przypadku nie osiągnieto zadowalających wyników - trasy bezpieczniejsze nadal nie odgrywały zbyt dużej roli. Jak widać, poziom dróg o najmniejszym bezpieczeństwie jest stale na poziomie 30%.

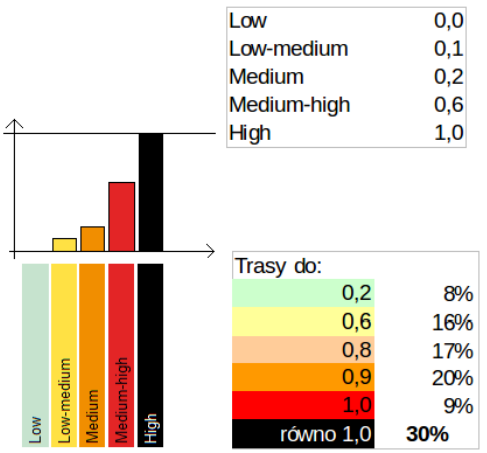
Dopiero po wielu próbach udało się ustalić odpowiednie współczynniki funkcji wagowej (rys. 14.6).

14.4.2. Budowa strony internetowej

Podstawowym zadaniem strony internetowej jest zaproponowanie użytkownikowi kilku wariantów trasy, wizualizacja proponowanych tras na mapie, podanie informacji o długości trasy oraz najważniejsze – ocena poziomu bezpieczeństwa każdej z tras. Ocena ta jest wyliczana na podstawie takich czynników jak:



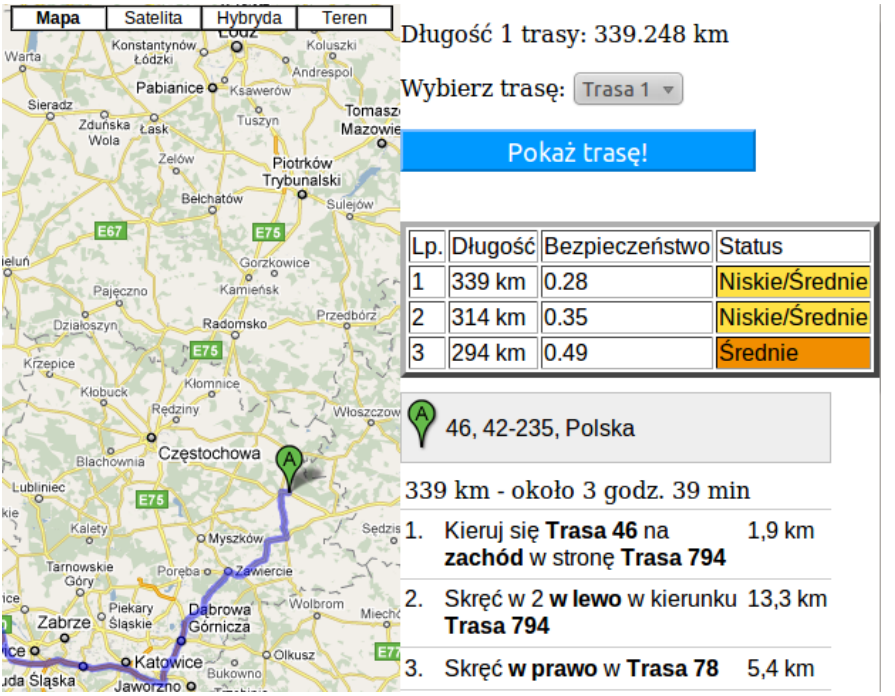
Rys. 14.5: Dobór parametrów - zmniejszone wagi tras bezpiecznych.



Rys. 14.6: Dobór parametrów - wybrane parametry.

- długość trasy – niebezpieczeństwo związane ze zmęczeniem kierującego pojazdem i ilość kilometrów do pokonania,
- stopień bezpieczeństwa drogi wyznaczony na podstawie statystyk – niebezpieczeństwo związane z przebiegiem danej drogi.

Użytkownik na podstawie informacji pochodzących ze strony może zwiększyć bezpieczeństwo swoje oraz pasażerów wybierając odpowiednią trasę. Zaproponowana aplikacja z założenia ma być narzędziem pomagającym kierowcy dokonać najlepszego wyboru, a nie podjąć decyzję za niego.



Rys. 14.7: Widok strony.

Dodanie mapy na stronę internetową

Dodanie mapy na własną stronę internetową jest prostym zadaniem. Wystarczającą umiejętnością jest podstawowa znajomość języka JavaScript oraz HTML. Chcąc umieścić mapę na własnej stronie należy wykonać trzy kroki

- dołączyć skrypt API w nagłówku strony

```
<script src="http://maps.google.com/maps/api/js?sensor=false"\
type="text/javascript"></script>
```

- wstawić obiekt w którym będzie wyświetlana mapa w ciele strony

```
<div id="mapka" style="width: 700px; height: 500px;\
border: 1px solid black; background: gray;">
<!-- tu bedzie mapa -->
</div>
```

- zainicjować mapę w skrypcie JavaScript

```
<script type='text/javascript'>
function mapaStart()
{
    var wspolrzedne = new google.maps.LatLng(10.541366,19.560615);
    var opcjeMapy = {
      zoom: 15,
      center: wspolrzedne,
```

```
        mapTypeId: google.maps.MapTypeId.SATELITTE
      };
      var mapa = new google.maps.Map(document.getElementById("mapka"),\
   opcjeMapy);
  }
  </script>
```

W powyższym skrypcie zdefiniowano współrzędne geograficzne na jakich mapa będzie wyśrodkowana, zbliżenie określonego punktu oraz typ mapy. W końcowej części skryptu pożądana mapa została zainicjowana.

Jak widać dodanie prostej mapy nie jest skomplikowanym zadaniem. Mapy Google dają ogromne możliwości programiście, szczegółowa dokumentacja dostarczanych przez nie opcji jest opublikowana na stronie projektu [9].

Jedną z opcji jest wyznaczanie tras pomiędzy dwoma miejscami, które zostaną wprowadzone przez użytkownika. Taką możliwość daje klasa `google.maps.DirectionsService` metodą `route()`.

```
route(request:DirectionsRequest,
          callback:function(DirectionsResult, DirectionsStatus)))
```

Metoda `route()` pozwala znaleźć interesującą trasę. Przyjmuje ona jako jeden z argumentów obiekt typu `DirectionsRequest` w którym podaje się parametry wyszukiwanej trasy, m.in.:

- punkt startu,
- punkt docelowy,
- punkty pośrednie (współrzędne geograficzne lub adres miejsca, maks. 8),
- podanie alternatywnych tras,
- unikanie autostrad,
- unikanie płatnych odcinków dróg,
- typ poruszania (samochodem, rowerem, pieszo),

Jeśli z pewnego względu trasa nie zostanie znaleziona, metoda zwraca powód braku rozwiązania, który można przedstawić użytkownikowi. Jeżeli ustawiona zostanie opcja wyszukiwania alternatywnych tras, trasy te zostaną zwrócone w tablicy obiektów typu `DirectionsRoute`. W przeciwnym wypadku tablica ta będzie miała tylko jeden obiekt z informacją dotyczącą danej trasy, a w tym m.in.:

- współrzędne prostokąta ograniczającego trasę,
- długość,
- czas trwania,
- kroki pośrednie na trasie.

Po uzyskaniu alternatywy tras kolejnym etapem jest wyznaczenie poziomu bezpieczeństwa każdej trasy. Numery dróg wraz z dystansem do pokonania na każdej drodze można uzyskać także dzięki usłudze Google Maps API. Udostępnia ona informacje o każdej zmianie kierunku jazdy na pokonywanej trasie. Trasa uzyskana z metody `route()` jest podzielona na pojedyncze odcinki dróg, zwane

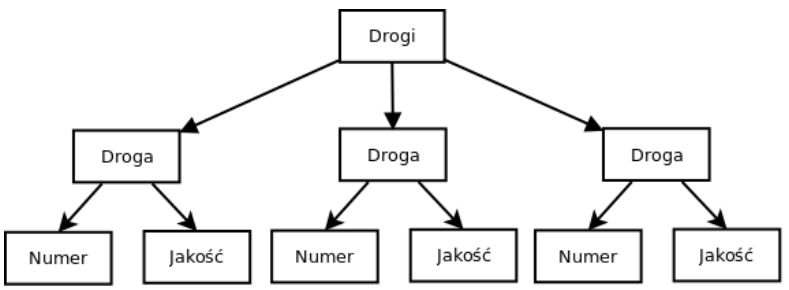
krokami. Każdy krok to dystans pomiędzy kolejnymi zmianami kierunku jazdy lub zmianami nazwy drogi. Do każdego kroku dołączona jest informacja podobna do tych używanych w systemach nawigacyjnych, bazujących na odbiornikach GPS, np.:

- Skręć w lewo w Pilczycka.
- Kontynuuj wzdłuż Główna.
- Na rondzie zjazd nr 2 w Wojska Polskiego.
- Kontynuuj wzdłuż Trasa 336.
- Kontynuuj wzdłuż S11.

Jeżeli trasa wiedzie np. drogą krajową lub autostradą, jej numer jest podawany w polu informacyjnym dotyczącym danego kroku. Parsując powyższą informację można ustalić numer drogi, aby następnie odnaleźć ją w bazie danych zawierającej numery dróg wraz z informacją o ich bezpieczeństwie. Jeżeli dany odcinek nie wiedzie przez drogę znajdującą się w bazie, przyjmuje się dla niego domyślny poziom bezpieczeństwa. Obliczony wynik dla każdej z zaproponowanych tras jest wizualizowany za pomocą jej koloru odpowiedniego dla danego poziomu bezpieczeństwa. Ponadto wynik liczbowy dla każdej z tras jest prezentowany w przeznaczonym dla tego polu tekstowym.

14.4.3. Format danych

Znalezione dane dotyczące bezpieczeństwa na drogach zamieszczane są w pliku XML. Jego strukturę przedstawiono na rysunku 14.8. Dane te są wczy-



Rys. 14.8: Struktura pliku xml zawierającego poziomy bezpieczeństwa dróg.

tywane na stronie z wykorzystaniem języka JavaScript i obiektów typu DOM XML. To narzędzie daje możliwość obiektowego traktowania wczytanych danych. Samo drzewo utworzone w pliku xml oraz każdy z liści tego drzewa można traktować jako jako osobny obiekt. Mechanizm ten bardzo ułatwia pracę z danymi zapisanymi w formacie xml, co więcej, istnieje możliwość łatwego przeszukiwania wczytanej struktury danych.

14.5. Podsumowanie

Można zauważyć, że ryzyko jest trudne do badania i opisu, często nie można powiedzieć nawet czy wynik, który poda algorytm jest poprawny. Nieosiągalne jest oczywiście stwierdzenie, że algorytm daje optymalne wyniki.

Stworzona strona internetowa pozwala sprawdzić, która trasa z wyznaczonych jest najbezpieczniejsza. Może się to przydać każdemu kierowcy, zwłaszcza w Polsce, gdzie problem wypadków drogowych jest ogromny.

Literatura

[1] *Hedging i nowoczesne usługi finansowe* Wydawnictwo Akademii Ekonomicznej w Poznaniu, (2001).

[2] Risk managment - AS/NZS 4360:2004. Technical report, Joint Technical Committee OB-007, (2004).

[3] S. L. K. I. M. F. C. U. C. F. W. Marvin J. Carr: Taxonomy-Based Risk Identification. Technical report, Carnegie Mellon University, (1993).

[4] *PN-N-18001:2004,Systemy zarządzania bezpieczeństwem i higieną pracy. Wymagania.*

[5] *PN-N-18002:2000,Systemy zarządzania bezpieczeństwem i higieną pracy. Ogólne wytyczne do oceny ryzyka zawodowego.*

[6] *PN-N-18004:2001,Systemy zarządzania bezpieczeństwem i higieną pracy. Wytyczne.*

[7] F. rozwoju inżynierii lądowej: *TRANSPORT – WYNIKI DZIAŁALNOŚCI W 2009 R.* Gdańsk, (2010).

[8] GUS: *Atlas ryzyka na drogach w Polsce 2007-2009* Zakład Wydawnictw Statystycznych, Warszawa, (2010).

[9] Google Maps API. `http://code.google.com/intl/pl/apis/maps/`

ROZDZIAŁ 15

KOMPUTEROWE WSPARCIE KONTROLI JAKOŚCI

P. Bolek, D. Paszuk

Systemy ekspertowej są systemami informatycznymi, które na podstawie posiadanej bazy wiedzy są w stanie rozwiązać postawiony przed nimi problem decyzyjny. Systemy ekspertowe znajdują szerokie zastosowanie nie tylko w przemyśle w działach kontroli jakości, ale także w medycynie (przy diagnozie chorób), ekonomii (do różnego rodzaju analizy), przy diagnozowaniu problemów technicznych oraz wielu innych. Systemy te cechuje szeroka dostępność, w przeciwieństwie do wykwalifikowanych pracowników. Mogą one pracować 24h/dobę, nie nużąc się monotonnym zajęciem, w związku z czym nie popełniają przypadkowych błędów. Nie są też subiektywne, ocenę podejmują zawsze według tych samych reguł zapisanych w bazie wiedzy.

Jednym ze sposobów budowania baz wiedzy wykorzystywanych w systemach ekspertowych jest posłużenie się metodą drzew decyzyjnych. Metoda ta został wykorzystana do budowy systemu ekspertowego dla sortowni owoców, co zostało opisane w niniejszym rozdziale

15.1. Zagadnienia teoretyczne

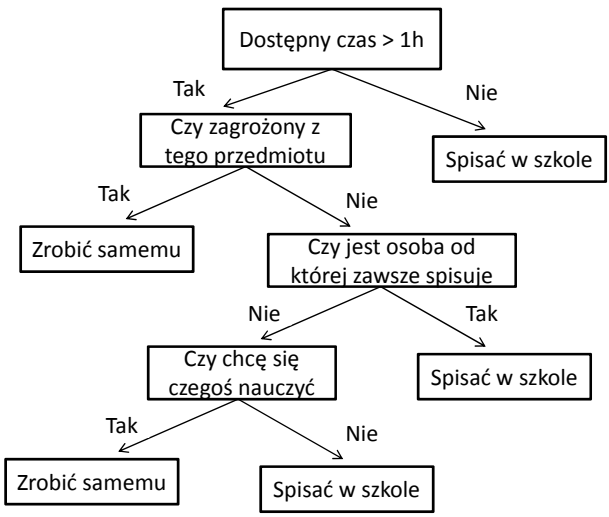
15.1.1. Drzewa decyzyjne

Analizując pewien problem należy zdać sobie sprawę z różnych dróg jego rozwiązania. Przykładem może tutaj być wybór „Czy rozwiązać zadanie domowe samemu, czy je przepisać w szkole". Jak wiadomo każda droga prowadzi do jakiegoś określonego celu i posiada jakieś plusy i minusy. Przedstawioną formę analizy problemu można w uproszczeniu nazwać drzewem decyzyjnym.

Obecnie drzewa decyzyjne głównie są wykorzystywane do indukcyjnego uczenia się maszyn. Spowodowane jest to faktem prostej programowej implementacji (instrukcje warunkowe typu *if-then*), dużej efektywności oraz łatwej interpretacji (łatwo stworzyć reguły kwalifikacyjne). Budowa drzewa decyzyjnego opiera się na analizie szeregu przykładów, z których każdy musi być opisany zestawem atry-

butów. Każdy atrybut może przyjmować różne wartości. Drzewa decyzyjne pozwalają, aby wartości były dyskretne oraz ciągłe, jednak w tym drugim przypadku należy dokonać dyskretyzacji na kilka przedziałów. Co ciekawe, ciąg przykładów może zawierać błędy, a nawet braki w wartościowaniu atrybutów.

Drzewo decyzyjne, formalnie rzecz biorąc, jest grafem, a dokładniej drzewem. Korzeniem tego drzewa jest atrybut bądź cecha wybrana na podstawie odpowiedniego kryterium, natomiast poszczególne gałęzie reprezentują wartości tego atrybutu. Węzłom drzewa na kolejnych poziomach przyporządkowane są kolejne atrybuty, natomiast węzły na najniższym poziomie reprezentują poszczególne klasy, czyli decyzje. Budowa drzewa decyzyjnego oparta jest o kryterium stopu, kryterium wyboru atrybutów oraz o przycinanie drzewa. W zależności od wybranego algorytmu budowy drzewa te trzy elementy mogą się różnić. Przykładowe drzewo decyzyjne do problemu „Czy rozwiązać zadanie domowe samemu, czy je przepisać w szkole" może mieć postać jak na rys. 15.1.



Rys. 15.1: Przykładowe drzewo decyzyjne.

Zalety i wady drzew decyzyjnych są następujące:

- Zalety drzew decyzyjnych:
  - mogą być budowane przy wykorzystaniu algorytmicznych technik „dziel i rządź", znanych ze swej szybkości i wykorzystywanych niemal przez wszystkie algorytmy budowy drzew
  - mogą reprezentować dowolnie złożone pojęcia pojedyncze lub wielokrotne, jeśli tylko ich definicje da się wyrazić w zależności od atrybutów,
  - ich forma reprezentacji jest względnie czytelna dla człowieka,
  - doskonale bronią się przed szumem danych,
  - pozwalają na łatwe do reprezentacji regułowej

– mogą być wykorzystywane do selekcji jak i ekstrakcji cech.

- Wady drzew decyzyjnych:
  – im więcej klas oraz im bardziej się one nakładają, tym większe drzewo decyzyjne,
  – trudno zapewnić jednocześnie wysoką jakość klasyfikacji i małe rozmiary drzewa,
  – przy budowie drzewa testuje się wartość jednego atrybutu na raz, co powoduje niepotrzebny rozrost drzewa dla danych, w których istnieją zależności między atrybutami,
  – tylko pewne klasy problemów daje się rozwiązywać za pomocą drzew decyzyjnych.

15.1.2. C4.5

Algorytm C4.5 jest jednym z popularniejszych algorytmów do tworzenia drzew decyzyjnych, i to głównie ze względu na jego prostotę oraz skuteczność. Został zaproponowany przez dr Ross'a Quinlan około roku 1996. Budowa drzewa jest realizowana począwszy od korzenia, a jako atrybut testowy wybiera się taki, który maksymalizuje przyrost informacji (ang. *information gain*) w węźle. Przyrost informacji jest wzbogacony o czynnik normalizujący, nazwany współczynnikiem przyrostu informacji (ang. *gain ratio*). Czynnik normalizujący znajdujący się w mianowniku (15.4) oznacza wartość informacyjną atrybutu a. Dla zbioru danych D, który jest podzbiorem wszystkich danych treningowych badanym w węźle, określamy najpierw przyrost informacji IG dla atrybutu a.

$$IG_a(D) = I(D) - E_a(D) \tag{15.1}$$

Powyższe równanie wykorzystuje entropię $E_a(D)$ jaka by wystąpiła gdyby do podziału został wybrany argument a. Przy ustalonych wartościach argumentu v, wartości decyzji $c \in C$ dla danej ze zbioru danych D otrzymujemy wyrażenie:

$$E_a(D) = \sum v \in R_a \frac{|D_{a=v}|}{|D|} \sum_{c \in C} -\frac{|D^c_{a=v}|}{|D|} log_2 \frac{|D^c_{a=v}|}{|D|} \tag{15.2}$$

$I(D)$ oznacza informację jaką posiada zbiór danych D i oblicza się go przez obliczenie entropii w zależności tylko od końcowej decyzji dla elementu ze zbioru treningowego $c \in C$

$$I(D) = \sum_{c \in C} -\frac{|D^c|}{|D|} log_2 \frac{|D^c|}{|D|} \tag{15.3}$$

Ostatecznie wybór atrybutu jest dokonywany przez testowanie wszystkich dostępnych atrybutów za pomocą równania (15.4)

$$GR_a(D) = \frac{IG_a(D)}{\sum v \in R_a - \frac{|D_{a=v}|}{|D|} \log \frac{|D_{a=v}|}{|D|}} \tag{15.4}$$

Proces budowy drzewa decyzyjnego można podzielić na dwa etapy:

- zbudowanie drzewa maksymalnego,
- przycinanie drzewa.

Zbudowanie drzewa maksymalnego można przedstawić (na bazie algorytmu ID3) z pomocą poniższej pseudo-funkcji, która zwraca drzewo dla następujących oznaczeń: R - zbiór nie kategoryzujących atrybutów (jeszcze nie były testowane), C - kategoryzujące atrybuty, S - zbiór danych w tym węźle.

Funkcja ID3 dla R,C,S

- Jeśli S jest pusty → zwróć liść z wartością Błąd
- Jeśli S zawiera atrybuty z taką samą wartością → zwróć liść z tą wartością
- Jeśli R jest pusty → zwróć liść z wartością najczęściej powtarzających się wartości w S
- Znajdź D atrybut z największym Gain_Ratio w R
- Niech dj| j=1,2, .., m będą wartościami atrybutu D
- Niech Sj| j=1,2, .., m będą podzbiorami S dla odpowiednich wartości atrybutu D
- Zwróć → drzewo z korzeniem w D i łukami d1, d2, .., dm
- wywołaj: ID3(R-D, C+D, S1), ID3(R-D, C+D, S2), .., ID3(R-D, C+D, Sm)

Funkcję tę wywołuje się dopóki drzewo nie zostanie utworzone w całości, to znaczy każdy węzeł wystawi odpowiednie liście.

Przycinanie drzewa nie tylko pozwala na utworzenie drzewa mniejszego, a co za tym idzie szybszego w działaniu, ale również zapobiega zbytniemu dopasowaniu do danych. Przycinanie powoduje (w stopniu regulowanym przez współczynniki np. m, najczęściej $m = 1$) rozmycie decyzji i zwiększenie błędu klasyfikacji na zbiorze trenującym, co pozwala na zmniejszenie tego błędu dla nieznanych danych. Metodę tę stosuje się przeglądając drzewo od dołu i testując poddrzewa wg nierówności (15.5). To kryterium bazuje na ocenie prawdopodobieństwa zmniejszenia błędu klasyfikacji liścia, który miałby zastąpić istniejący fragment drzewa, dla statystycznie sprawdzonych przykładów ze zbioru testowego. Węzeł zastępuje się odpowiednim liściem jeśli błąd liścia e_l jest mniejszy niż pewna ustalona wartość progu. gdzie e_n oznacza błąd klasyfikacji dla zastępowanego węzła, a $|D_n|$ liczbę przykładów znajdujących się w węźle n.

$$e_l \leq e_n + m\sqrt{\frac{e_n(1-e_n)}{|D_n|}} \tag{15.5}$$

Algorytm C4.5 pozwala także na podawanie danych ciągłych, jak i danych z brakującymi wartościami argumentów. Jeśli danych jest dużo, do budowy algorytmu wykorzystuje się $\frac{2}{3}$ danych, a do przycinania $\frac{1}{3}$. Jeśli dane są ciągłe, przedział tak dzieli się na określoną ilość klas (zazwyczaj dwie) korzystając z kryterium entropii, aby zysk z danego atrybutu był jak największy. Jeśli chodzi o reakcję na brakujące wartości, jest kilka sposobów:

- Pomijanie wszystkich przykładów niezdefiniowanych kompletnie, oraz dołożenie wagi zmniejszającej znaczenie danego algorytmu $\frac{pelne\,dane}{wszystkie\,dane}$
- Uzupełnianie brakujące informacji przez wstawienie:
  - najczęściej występującą wartości,
  - najczęściej występującej wartości w zbiorze przykładów tej samej kategorii,
  - wartości ustalonej na podstawie znanych wartości innych atrybutów.

15.1.3. CART

Algorytm CART (ang. *Classification and Regression Trees*) to jeden z najpopularniejszych i najskuteczniejszych algorytmów do indukcji drzew decyzyjnych. Narzędzie to powstało na początku lat 80 ubiegłego stulecia [1], jednak w przeciągu lat doczekało się kilku nieznacznych modyfikacji [2] stanowiąc oddzielną grupę algorytmów do budowy drzew. Jak sama nazwa wskazuję algorytm ten może rozwiązywać problemy klasyfikacyjne, jak i regresyjne. Jego działanie opiera się na tworzeniu drzew binarnych (z jednego węzła mogą wychodzić co najwyżej dwie gałęzie) wykorzystując cechy ciągłe albo dyskretne. Dla cech ciągłych rozpatrywane są wszystkie możliwe podziały na dwa zbiory $(-\infty, a]$ oraz (a, ∞), a dla cech dyskretnych wszystkie możliwe podziały zbioru wartości cech na dwa rozłączne i uzupełniające się podzbiory. Każdemu węzłowi przypisuje się etykietę atrybutu dominującego w węźle, bądź wynikająco z oceny kosztów pomyłek. Algorytm CART jako kryterium jakości podziałów wykorzystuje przyrost czystości węzłów. Miarą nieczystości jest tzw. Indeks Giniego (ang. *Gini Index*):

$$Gini(C) = 1 - \sum_{j=1}^{k} p_j^2 \tag{15.6}$$

gdzie: p_j- jest prawdopodobieństwem $j - tej$ klasy w węźle C.

Algorytm CART pozwala także skorzystać z miary entropii obliczanej w identyczny sposób jak dla C4.5 oraz z tzw. *twoing*. W tym algorytmie sugeruje się budowę maksymalnego drzewa i ocenę jego zdolności do uogólnienia. w wewnętrznym teście wyznacza się optymalną wartość λ minimalizującą:

$$Err(D) + \lambda l_D \tag{15.7}$$

gdzie: l_D jest liczbą liści drzewa.

Algorytm CART stosuje dwa warianty tej metody. W pierwszym z nich zostaje wybrane drzewo o minimalnej wartości powyższego wyrażenia, a w drugim drzewo zostaje maksymalnie przycięte.

Zaczynając od węzła ze wszytkimi danymi, ogólne działanie algorytmu można zobrazować następująco:

- Powtórz:
  - Rozważ wszystkie możliwe wartości atrybutów
  - Wybierz atrybut $(X = t_1)$ najlepiej dzielący cały zbiór, korzystając z jednego z kryteriów podziału

– Jeśli $X < t_1$ wyślij dane w „lewo", w przeciwnym wypadku wyślij dane w „prawo"

dla otrzymanych węzłów, aż kryterium stopu nie każe zatrzymać procesu budowy drzewa

- Jeśli dla jakichś danych brakuje wartości wykorzystywanej w danym węźle → analizuj tzw. podziały zastępcze (ang. *surrogate splits*)
- Ostatecznie przytnij drzewo korzystając z walidacji krzyżowej (ang. *cross-validation*)

Ważną zaletą algorytmu jest jednoczesne zestawienie kosztu resubstytucji (współczynnika błędu obliczonego ze zbioru uczącego) ze współczynnikiem błędu obliczonym na zbiorze testowym (ta druga wartość może być wynikiem prostej walidacji, walidacji krzyżowej, wielokrotnej walidacji krzyżowej czy metod bootstrapowych). Przycinanie drzewa dokonywane jest z uwzględnieniem obu współczynników.

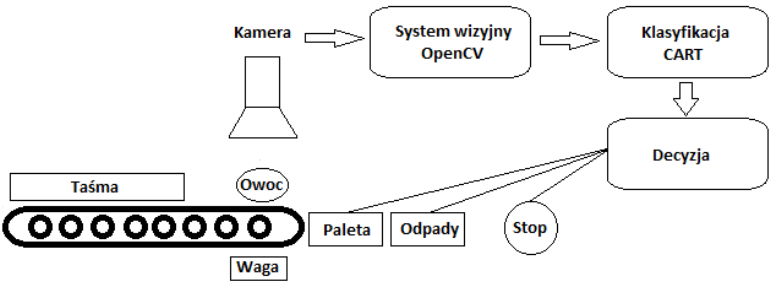
Algorytm CART nie ma jedynej, prostej i słusznej implementacji. Budowanie drzewa za pomocą tego narzędzia może przebiegać na wiele sposobów, które jednak są do siebie dość zbliżone. Przykładem może być wybranie jednego z trzech kryteriów podziału, wybór kryterium stopu czy różne wykorzystanie walidacji krzyżowej w procesie odpowiedniego przycięcia drzewa. Proces budowy drzewa także jest inny dla problemu regresyjnego jak i klasyfikującego, co dodatkowo komplikuje wybór tej „słuszne" metody. Sami twórcy udokumentowali szereg procedur mających ułatwić budowanie drzewa wykorzystując CART, jednak są to jedynie propozycje, które, jak sami zaznaczają, nie są niezawodne i czasem warto spróbować innego rozwiązania, gdyż może ono przynieść lepsze rezultaty.

15.1.4. Sortownia owoców

Aby zobrazować działanie drzew decyzyjnych w praktycznej implementacji systemu ekspertowege wykonano projekt systemu zarządzającego linią w sortowni owoców. Sam problem jest dość abstrakcyjny, ponieważ przeważnie nie miesza się różnych typów owoców na jednej linii. Pozwala jednak ukazać w przystępny sposób szczegóły procesu budowy takiego systemu, biorąc pod uwagę nie tylko klasyfikację samego owocu, ale także inne aspekty ekonomiczne.

W projekcie przyjęto kilka uproszczeń. Założono, że na jednej taśmie będą pojawiały się pojedyncze owoce i trzeba będzie ja zakwalifikować albo do odpowiedniego typu owocu albo do odpadów. Nad taśmą znajduje się system wizyjny, za pomocą którego system ekspertowy będzie dokonywał odczytu atrybutów. Dodatkowo pod taśmą znajduje się waga, dzięki której system zna wagę owocu. Całość systemu przedstawiona jest na rys. 15.2. Wynikiem projektu miał być prosty program do symulacji komputerowych, dlatego pominięto w nim zagadnienia buforu linii, transportu, magazynowania zasobów itp. Klasyfikacji owoców na taśmie miała zostać wykonana z uwzględnieniem celów ekonomicznych firmy. To znaczy, że brane miały być pod uwagę różne priorytety firmy. Problem ten jest na tyle ogólny, że może mieć zastosowanie przy rozwiązywaniu innych problemów klasyfikacyjnych.

Dużą zaletą projektowanego rozwiązania miała być możliwość budowania wielu drzew za pośrednictwem panelu programu oraz przy wykorzystaniu podanej przez użytkownika bazy danych symulacyjnych. Zdecydowano, że zaimplementowanym algorytmem do budowy drzew będzie algorytm CART. Skutkiem tego, użytkownik nie będzie musiał posiadać specjalistycznej wiedzy z zakresu ekonomii, ale powinna wystarczyć mu intuicja w kwestii od czego może zależeć decyzja. Założono, że system sam rozpoznaje różne owoce, co miało ułatwić ekstrahowanie cech i pozwalać skupić się na skuteczności realizacji postawionego zadania.



Rys. 15.2: Poglądowy schemat systemu.

Klasyfikacja obiektu

Na podstawie przeprowadzonej klasyfikacji system może podjąć jedną z sześciu decyzji.

- jabłko typu A (zwykłe czerwone jabłko),
- jabłko typu B (jabłko zielone),
- pomarańcza,
- banan,
- odpady (zepsute owoce, nierozpoznane, z powodów ekonomicznych),
- wstrzymanie linii.

Każda z tych decyzji zależy zarówno od ekonomicznego aspektu, jak i oceny owocu na podstawie analizy obrazu.

Założenia dotyczące analizy obrazu

Głównym źródłem informacji na temat owocu jest jego obraz przekazywany do systemu. Dodatkowym parametrem jest jego waga. Początkowo waga miała być losowana dla danego owocu, jednak w trakcie realizacji projektu zdecydowano, że bliższe prawdy będzie ocenianie jej również na podstawie obrazu, korzystając z wielkości owocu.

Kamera znajdująca się nad systemem została zasymulowana przez zdjęcia owoców robionych zawsze w takich samych warunkach, to znaczy z takiej sa-

mej wysokości (około 30cm), na takim samym podłożu (czarny brystol), przy takim samym oświetleniu. Zdjęcia zostały wykonane aparatem o rozdzielczością 2Mpx oraz przekonwertowane do rozmiaru 800x600 punktów. Głównym powodem użycia takiego aparatu i konwersji było pokazanie, że system wizyjny nie musi być bardzo dobrej jakości, a wystarczające rozwiązanie nie musi być drogie.

Do ekstrahowanie cech wykorzystano bibliotekę OpenCV. Pozwala ona na przetwarzanie obrazów w dość łatwy sposób. Zawiera między innymi funkcje pozwalające na wyszukiwanie krawędzi, konturu, oraz wykonanie wielu innych obliczeń prowadzących do parametryzacji, np. na wyliczenie momentów geometrycznych czy momentów Hu, a nawet odnajdywanie szablonów. Ponadto biblioteka OpenCV pozwala na podstawową obróbkę zdjęć za poprzez wyrównywanie histogramu, sumowanie czy mnożenia obrazów albo stosowania określonych filtrów.

Po znalezieniu konturu, korzystając z innych funkcji biblioteki, łatwo można określić dość dużą ilość parametrów obrazu, takich jak: wielkości obiektu (pole powierzchni - moment geometryczny m_{00}), wielkość łuku oraz kulistość, położenie środka obiektu, jego kąt nachylenia do osi głównej i inne. Ponieważ wektor danych zawiera też atrybuty do analizy pod kątem ekonomicznym, postanowiono zawęzić ich ilość i z analizy obrazu wybierać tylko najbardziej znaczące cechy:

- kolor (wartość z przedziału 0-255 osobno dla składowej czerwonej, zielonej i niebieskiej),
- waga(oceniania na podstawie pola powierzchni),
- krągłość (wartość liczbowa),
- pomarszczenie owocu.

Na podstawie tych cech miały być wyznaczane wartości atrybutów, od których zależeć miał algorytm budowy drzewa oraz późniejsza klasyfikacja.

Najtrudniejszym i najważniejszym etapem przetwarzania obrazu jest znalezienie odpowiedniego konturu. Istnieje na to kilka metod. W systemie postanowiono skorzystać z funkcji `cvFindContour` biblioteki OpenCV, która zapisuje wszystkie znalezione na obrazie kontury we sekwencji `cvSeq`. Funkcja ta wymagała jednak odpowiedniego przygotowania obrazu. `CvFindContour` jako obraz wejściowy przyjmuje obraz pseudobinarny, to znaczy do analizy wykorzystuje tylko dwie wartości koloru pikseli: białe i czarne (jeśli obraz zawiera szare piksele, wszystkie nie czarne piksele traktowane są jako białe tzn. o wartości 1).

Aspekt ekonomiczny

Skupiając się na tym aby jedynym zadaniem systemu nie była czysta klasyfikacja, której jedynym wynikiem jest stwierdzenie czy dany owoc jest jabłkiem czy nim nie jest, należało dodać atrybuty odzwierciedlające rzeczywisty stan firmy oraz jej założenia. Jednak aby lepiej dobrać odpowiednie cechy należy zrozumieć główne aspekty, którymi kieruje się właściciel sortowni. Oczywiście ogólnie sprawa wydaję się dość prosta, gdyż głównym kołem, które napędza cały sort owoców jest zysk, czyli pieniądz. To właśnie na to należy zwrócić największą

uwagę projektując system ekspertowy wspierający realizację procesu produkcyjnego. Tak więc celem systemu jest nie tylko poprawna klasyfikacja, ale także zoptymalizowanie zysku przez sortownię.

Opierając się na doświadczeniu autorów zdobytym podczas praktyk w firmach produkcyjnych zdecydowano dobrać takie cechy, które związane są z bezawaryjnością linii oraz wymogami stawianymi przez klienta, co rzutuje nie tylko na przychód, ale także na markę sortowni:

- Liczba odpadów w danej jednostce czasu t - cecha, która ma wpływ na zatrzymanie linii, czy to z powodu usterki systemu wizyjnego, czy zepsutej partii owoców. Może przyjmujmować wartości: mała, średnia, duża.
- Miejsce w palecie - cecha odpowiada za wolne miejsca w palecie. Wymiana palety w zależności od jej rodzaju wymaga określonej ilości czasu co może nie być pożądane jeśli zależy nam na szybkości przesortowania jakiegoś owocu, np. w skutek braków w zamówieniu. Cecha ta przyjmuje wartość TAK lub NIE.
- Braki w zamówieniu - cecha określa aktualny stan sortowni. W skutek wystąpienia braków system ekspertowy podejmie odpowiednie kroki. Cecha określana jest w trzystopniowej skali (1- nie ma braków, 2 - braki, 3- duże braki).
- Priorytet sortu - cecha określa, jaki owoc jest obecnie najważniejszy dla systemu. Może się to wiązać z brakami w zamówieniu, zbliżającym się terminem dostawy, czy specjalnymi życzeniami klienta (np. „Dzień banana" w sieci supermarketów). Dostępne wartości odpowiadają dostępnym owocom (banan, pomarańcza, jabłko A, jabłko B) oraz brak priorytetu.
- Usterka - cecha, która określa prawdopodobieństwo wystąpienia problemu mechanicznego na linii sortującej owoce. Ma ona związek z czasem pracy linii, datą ostatniego wykonanego przeglądu oraz zastosowania się do matrycy TPM. Przyjmuję wartości: małe (30%), średnie (60%) oraz duże (90%).

Powyższe cechy pozwalają na uwzględnienie czynników, które wpływają na przychód przedsiębiorstwa. Załóżmy, że firma ma określone zamówienia (kontrakty) i ich daty realizacji i na tej podstawie wskazuje priorytet sortu oraz braki w zamówieniu. Te obydwa atrybuty także wpływają na kształtowanie wizerunku sortowni oraz są brane pod uwagę przy analizie prawdopodobieństwa wystąpienia usterki. Z jednej strony nie wywiązanie się z przyjętego kontraktu ma wpływ na markę firmy oraz wiąże się ze sporymi karami finansowymi, natomiast z drugiej strony ciągłe sortowanie owoców bez względu na duże prawdopodobieństwo wystąpienia usterki prowadzi do zatrzymania linii, co powoduje utratę czasu i pieniędzy.

Wszystkie te atrybuty są wybierane w interfejsie użytkownika tworzonego programu . Ma to odzwierciedlić realizację zamówień i terminów. Nie są one obliczane bezpośrednio przez program, ponieważ samo zagadnienie dotyczące zarządzania i księgowości firmy powodowałoby odsunięcie się od meritum problemu i zagłębienie się w problemy nie związane z przedstawianym tu tematem.

15.2. Opis implementacji

Cały system ekspertowy został zrealizowany w środowisku Visual Studio C++ 2008 Express Edition. W programie można wydzielić trzy części, które opisano poniżej.

- Moduł analizujący obraz,
- Moduł budujący drzewa,
- Interfejs programu wykorzystujący pozostałe moduły.

15.2.1. Moduł analizujący obraz

Implementacja tego modułu znajduje się on w osobnym pliku `ParameterFinder.cpp`. Zawiera on klasę o nazwie `ParameterCounter`, której obiekt należy utworzyć w programie głównym. Klasa ta pozwala na przetworzenie pliku podanego jako parametr metody `Find`, która dla danego obrazu wywołuje szereg innych metod, a w wyniku ich działania zostają wypełnione poszczególne pola metody z parametrami obrazu.

Analiza obrazu rozpoczyna się od znalezienia konturu owocu. Wykorzystaną do tego metodą jest metoda znajdująca się w bibliotece OpenCv o nazwie `cvFindContour`. Do przygotowanie obrazu dla funkcji `cvFindContour` zapisującej kontur do użytecznej formy wskaźnika na sekwencję `cvSeq`. Do dalszej obróbki rozważano kilka funkcji również zaimplementowanych w bibliotece OpenCV:

- `cvThreshold` - funkcja, która dla pewnego progu dzieli obraz w zależności od parametru `ThresholdType` na obraz czarno-biały, albo biało-czarny, zeruje wartości poniżej, lub powyżej progu, albo zmniejsza liczbę użytych wartości. Wadą tej funkcji jest trudność w ustawieniu parametrów tak aby działała ona poprawnie dla różnych typów owoców.
- `cvCanny` - funkcja, dla której obrazem wyjściowym są znalezione krawędzie za pomocą filtra Sobel 3×4, 5×5 lub $7t\times7$. Zanim zostaje użyty filtr obraz podlega transformacji za pomocą funkcji `threshold` dla dwóch rożnych wartości podawanych w parametrach wywołania. Wadą tej funkcji jest fakt, że efektem jej działania były tylko linie, które zaznaczały największe gradienty zmiany koloru. Linie te były w zależności od parametrów albo nieciągłe, albo było ich bardzo wiele i były poszarpane w związku z tym nie dało się wyznaczyć odpowiedniego konturu.
- `cvInRange` - do tej funkcji podawane są dwa porównywane obrazy. Funkcja zwraca obraz, pozostawiając na nim piksele obrazu wejściowego o wartościach mieszczących się w zadanym przedziale, oraz piksele o wartościach zmienionych, jeśli nie mieściły się w tym przedziale.
- `FloodFill` - fukcja ta dla zadanego punktu startowego (`floodSeed`) obrazu wejściowego, sprawdza, czy sąsiednie piksele są ciemniejsze lub jaśniejsze od tego punktu, z uwzględnieniem zadanego w parametrach marginesu. Jeśli tak, to są one zamalowywane na zadany kolor. Funkcja sprawdza w ten sposób cały obraz.

Ostatecznie zdecydowano się na funkcję `cvFloodFill`, ponieważ ustawienie parametrów dla pozostałych funkcji sprawdzało się tylko w części przypadków, a funkcja `cvFloodFill` miała największą skuteczność dla użytych danych testowych.

Kolejne etapy przetwarzania obrazu rozpoczęto od zastosowania wybranej funkcji na kolorowym obrazie. Następnie obraz był konwertowany do skali szarości. Taki obraz został podawany do funkcji znajdującej kontur. Kolejno należało spośród odnalezionych konturów w sekwencji wybrać ten największy. Jako kryterium oceny wielkości został przyjęty moment geometryczny m_{00}, obliczany za pomocą funkcji `CvMoments`. Niestety, dla bardzo ciemno czerwonych jabłek metoda okazała się nieskuteczna, ponieważ owoc zlewał się z tłem. Dlatego wprowadzono dodatkowy test - jeśli długość łuku największego konturu była mniejsza niż ustalona wartość minimalna, procedura znalezienia konturu następowała od nowa, ale za pomocą innej funkcji napisanej przez autorów. Funkcja ta działała dużo wolniej (chociaż czas jej wykonania mieścił się w dopuszczalnym czasie równym około 1s). Podobnie jak `cvFloodFill` funkcja ta polega na wypełnieniu pikseli tła czarnym kolorem. W tym celu przeglądany i testowany jest cały obraz pod względem koloru piksela. Warunki w przypadku których piksel jest klasyfikowany jako tło zostały dobrane doświadczalnie. Następnie znowu obraz jest poddawany transformowaniu do skali szarości, znajdowaniu konturu, oraz wyszukaniu największego spośród nich. Założono, że po użyciu którejś z tych dwóch funkcji kontur jest poprawny.

Ostatecznie obliczano następujące cechy obiektu na obrazie:

- krągłość (ang. *shape*) - parametr zdefiniowany równaniem:

$$\frac{ArcLength^2}{4\pi m_{00}}$$

  gdzie $ArcLength$ jest długością łuku, m_{00} oznacza powierzchnię bryły otoczonej konturem
- pomarszczenie (ang. *shrivel*) - parametr zdefiniowany jako współczynnik ilości (określanej przez ilość białych pikseli) krawędzi w obrębie konturu owocu do jego powierzchni. Są efektem wrysowania na obraz linii powstałych przez zastosowanie filtru Canny dla obrazu. Ponieważ wykrywa on gwałtowne zmiany kolorów, jest w stanie obrysować „zmarszczki" owocu ze względu na inny kolor zepsutych miejsc.
- kolor owocu - jest odnajdywany metodą przeglądania obrazu oraz obliczenia średniej wartości pikseli znajdujących się w konturze osobno dla kanału czerwonego, zielonego i niebieskiego.

15.2.2. Moduł budowy drzewa

Biblioteka OpenCV posiada metody pozwalające na zaawansowane przetwarzanie danych i uczenie maszynowe, znajdują się tu również metody na budowanie drzew decyzyjnych. Cały moduł do budowy drzew można znaleźć w bibliotece OpenCV. Korzysta on z algorytmu CART. Tworzenie drzewa odbywa się przez

wywołanie funkcji trenującej drzewo `train` na utworzonym wcześniej obiekcie klasy `CvDTree`. Funkcja ta sama wywołuje inne funkcje oraz korzysta z innych obiektów bez potrzeby udziału programisty w tym procesie. Jako parametry wejściowe funkcji należy podać dane uczące w postaci macierzy dwuwymiarowej oraz wektor rezultatów. Obydwa parametry muszą być zapisane w formacie `CvMat`. Dodatkowo można określić, w których miejscach brakuje danych (np. wskutek problemu z ekstrakcją koloru z owocu). Do tego celu należy utworzyć zero-jedynkową macierz o dwóch wymiarach formatu `CvMat`. Brak danej określa się jedynką w odpowiednim miejscu macierzy. W strukturze `CvDTreeParams` również podawanej jako parametr, można określić parametry budowy drzewa takie jak:

- `max_categories`- określa maksymalną liczbę atrybutów wziętych pod uwagę przy budowie drzewa. Pozwala to na zmniejszenie wielkości drzewa, albo poprawienie jego jakości w przypadku kiedy jest dość dużo atrybutów i nie wiadomo które z nich są istotne, a które nie mają znaczenia.
- `max_depth`- określa maksymalną głębokość drzewa. Tym parametrem również można wpływać na wielkość drzewa, co wiąże się z bardziej skomplikowanymi regułami oraz dokładniejszą predykcją.
- `min_sample_count`- nie wydziela nowego poddrzewa jeśli w węźle jest mniej danych niż ustawiona wartość.
- `cv_folds`- określa ilość wymaganych krzyżowych walidacji. Parametr ten wykorzystuje się do przycinania drzewa i określenia jego optymalnej głębokości.
- `use_surrogates`- parametr pozwala na wykorzystywanie przy brakujących danych dodatkowych podziałów, kiedy systemowi nie udało ustalić się jakiejś cechy (np. użytkownik jej nie podał, albo w pliku danych wejściowych był błąd) wtedy ta cecha jest przewidywana.
- `use_1se_rule` -określa wykorzystywanie metody *Harsher pruning* do przycinania drzewa, co ma wpływ na wielkość drzewa.
- `truncate_pruned_tree`- ustalone na wartość prawdy powoduje że wycięte gałęzie drzewa nie są pamiętane.
- `regression_accuracy`- jedno z kryterium stopu dla dzielenia kolejnych węzłów.
- `priors`- pozwala na ustalenie wag przewidywanych kategorii. Łatwo zobrazować wykorzystanie tego pojęcia tak jak pokazano to w przykładowym programie biblioteki openCV na przykładzie klasyfikacji grzybów jadalnych. Większy priorytet przy budowie drzewa został ustalony tak, aby błędnie klasyfikowany był jadalny grzyb do trujących niż odwrotnie. W efekcie nigdy nie zdarzy się sytuacja zakwalifikowania trującego grzyba jako jadalny. W tym drzewie również istnieje możliwość dodania takiego priorytetu.

Po utworzeniu drzewa istnieje możliwość jego zapisania do formatu "xmlźa pomocą funkcji `save`. Kolejną przydatną funkcją jest `get_var_importance`. Pozwala ona na pobranie informacji o ważności danego atrybutu w danym drzewie. Te informacje są przekazywane do interfejsu programu.

Aby zakwalifikować owoc używając utworzonego drzewa należy użyć funkcji `predict` podając jej wektor cech kwalifikowanego obiektu. Funkcja zwraca dostęp do końcowego węzła z którego można odczytać rezultat predykcji (poprzez pole `value`). OpenCV dostarcza możliwość własnej implementacji procesu przewidywania na podstawie utworzonych w drzewie reguł. W tym przypadku należy skorzystać m.in. z takich obiektów i funkcji jak `CvDTreeSplit`, `CvDTreeNode`, `get_pruned_tree_idx`, `split`, `get_root`, `get_node` i innych opisanych w dokumentacji dotyczącej OpenCV.

15.2.3. Interfejs i obsługa programu

Interfejs dla systemu ekspertowego sortowni owoców musi posiadać opcje pozwalające na ustawienia priorytetów ekonomicznych firmy, możliwości zmiany istniejącego drzewa, wytrenowania go na nowych danych, a także możliwość uzyskania informacji o aktualnie klasyfikowanym obiekcie (monitorowanie poprawności) lub o drzewie (istotność parametrów). Wszystkie te opcje zostały podzielone między 4 zakładki:

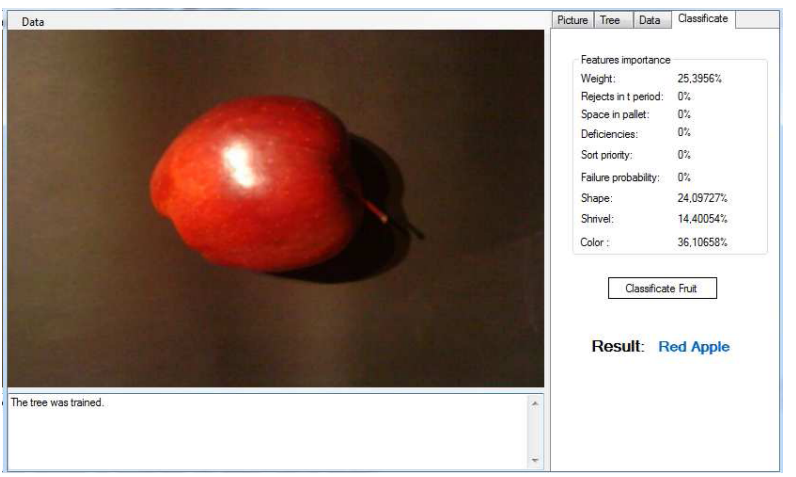
- Picture - jest to jedyna w tym programie zakładka symulacyjna, pozwala użytkownikowi na wybranie danego owocu do klasyfikacji. W normalnym systemie zamiast przycisku po prostu pojawiałby się owoc, a jego wykrywanie mogłoby zostać zrealizowane za pomocą np. czujników przerwania strumienia światła.
- Tree - posiada dwa panele. Pierwszy pozwala na ustalenie pliku danych za pomocą przycisku „Load Data", na podstawie których budowane jest drzewo. Można również wykorzystać aktualnie utworzone dane wykorzystując tryb „Data Generation Mode" opisany niżej. Drugi panel zawiera pola w których można zmienić parametry dla budowanego drzewa opisane w podrozdziale „Moduł budowy drzewa", następnie wystarczy wybrać przycisk „Build Tree"
- Data - dotyczy ona danych dla obiektu do klasyfikacji (lub danych testowych). Posiada dwa panele. „Economic data" w którym należy ustawić aktualne priorytety firmy (który owoc jest priorytetowy- „Sort priority"), wielkości braków w zamówieniach („deficiences"), a także parametry które w rzeczywistym systemie powinny być automatycznie odczytywane, a u nas zostały zasymulowane „Failure probability" czyli prawdopodobieństwo usterki maszyny (np. przy braku zatrzymania linii w celu przeglądu), miejsce w palecie na owoce „Space in pallet", oraz ilość owoców zakwalifikowanych do odpadów w określonej jednostce czasu „Reject in t period". Panel „Visualisation data" jest nieaktywny do zmiany przez użytkownika, wyświetla parametry zidentyfikowane dla danego obiektu przez moduł do analizy obrazu ParameterFinder.
- Classificate - po wczytaniu zdjęcia oraz wybraniu odpowiednich parametrów w celu uzyskania decyzji systemu ekspertowego należy nacisnąć znajdujący się przycisk „Classificate fruit". Poniżej znajduje się pole, w którym pojawi się decyzja. W tej zakładce znajduje się również panel opisujący znaczenie poszczególnych atrybutów dla decyzji podejmującej przez zbudowane drzewo.

W trybie „Data Generation Mode" (wybierany z paska menu) dla kolejno wczytywanych owoców ustawia się nie tylko parametry ekonomiczne, ale także odpo-

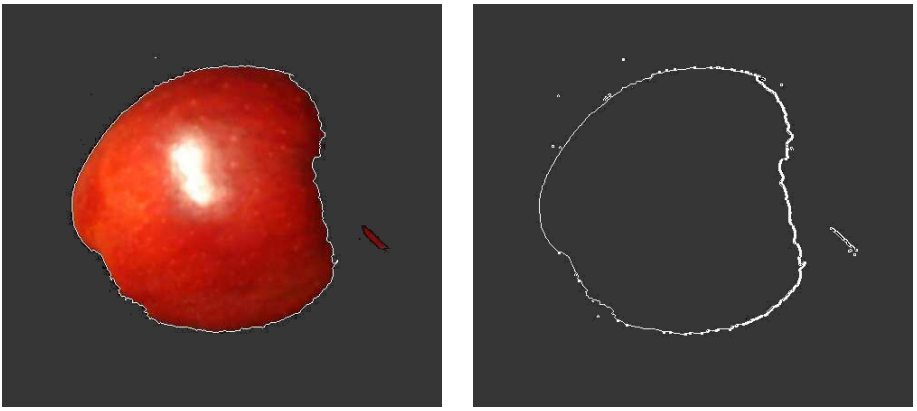
wiednią decyzję, na podstawie której system będzie potem trenował drzewo. Gotowy zbiór danych można zapisać za pomocą przycisku „Save Data" w zakładce Tree. Jest to jednak dość czasochłonna metoda, dlatego można również wczytać odpowiedni zbiór danych uczących korzystając z przycisku „Load Data" w tej samej zakładce.

15.2.4. Badania

Głównym problemem, który ujawnił się podczas tworzenia tego systemu ekspertowego był brak poprawnych danych uczących. Biblioteka OpenCV wymaga przynajmniej 100 wektorów danych, aby utworzyć minimalne drzewo, a te jednak jest nie wystarczające do poprawnej predykcji.
Początkowo podjęto próbę wygenerowania takich danych, które nauczą system samej klasyfikacji owoców bez względu na atrybuty ekonomiczne, aby sprawdzić poprawność ekstrahowania cech ze zdjęć. Ten etap był dość prosty i zakończył się powodzeniem. Głównymi atrybutem podczas budowania reguł był jak łatwo przewidzieć kolor, którego ważność oceniona została na 36%. Kolejnymi atrybutami zostali kształt i waga (około 25%), co także było zgodne z oczekiwaniami. Ostatnią pozycję zajął atrybut marskość (około 15%) co głównie związane jest z dużą liczbą poczernionych owoców dostarczonych do tworzenia danych. W tym etapie przeanalizowano ponad 200 zdjęć owoców takich jak: dojrzałe banany, zepsute banany, czerwone jabłka, niedojrzałe czerwone jabłka, zielone jabłka, żółto-czerwone jabłka, pomarańcze i niedojrzałe pomarańcze. Na rys. 15.3 pokazano wartości ważności poszczególnych atrybutów i wynik predykcji czerwonego jabłka. Na kolejnym rysunku, 15.4 pokazano wynik ekstrahowania cech ze zdjęcia.



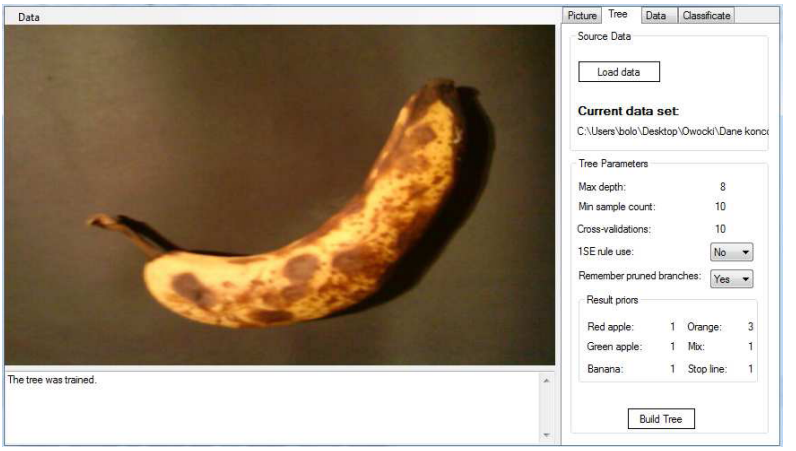
Rys. 15.3: Wartości poszczególnych atrybutów i wynik predykcji dla danych nie uwzględniających aspektu ekonomicznego.



Rys. 15.4: Ekstrakcja cech a) obrysowany kontur jabłka, z którego obliczany jest współczynnik określającego kształt, b) pomarszczenie jabłka.

Kolejnym krokiem było wygenerowanie danych, dla których oprócz wymienionych już atrybutów (zobacz rys. 15.5) uwzględniono również atrybuty odpowiedzialne za aspekt ekonomiczny (zobacz rys. 15.6). Bez wątpienia była to najtrudniejsza część zadania. Do analizy znów użyto ponad 200 zdjęć tych samych owoców co poprzednio. Podczas klasyfikacji dokonywano odpowiedniej zmiany atrybutów ekonomicznych . Zaniedbanie jednego z nich prowadziło do błędnej klasyfikacji, i nie było to tylko błędne zatrzymanie linii, ale np. zakwalifikowanie banana jako pomarańczy, co wynikało z reguł utworzonych przez nowo dodane atrybuty (zobacz rys. 15.7 i rys. 15.8). Powstało wiele różnych wersji danych, gdzie każde zdjęcie owocu musiało być kwalifikowane przynajmniej trzy razy (dla kwalifikacji samego owocu, dla kwalifikacji do kosza odpadów oraz dla kwalifikacji czego skutkiem było zatrzymanie linii). Co ciekawe, pewne usystematyzowanie ich tworzenia (np. poprzez wprowadzanie tych samych zmian dla podobnych owoców) prowadziło do jeszcze gorszych wyników niż dane tworzone w sposób bardziej losowy. Ostatecznie zbudowane drzewo działało w sposób w miarę satysfakcjonujący, jednak nie nieomylny. Jak widać wagi atrybutów rozkładają się w sposób zbliżony do siebie, co świadczy o wielu wytworzonych regułach i obrazuje większą złożoność problemu przy uwzględnieniu ekonomicznych przesłanek przedsiębiorstwa.

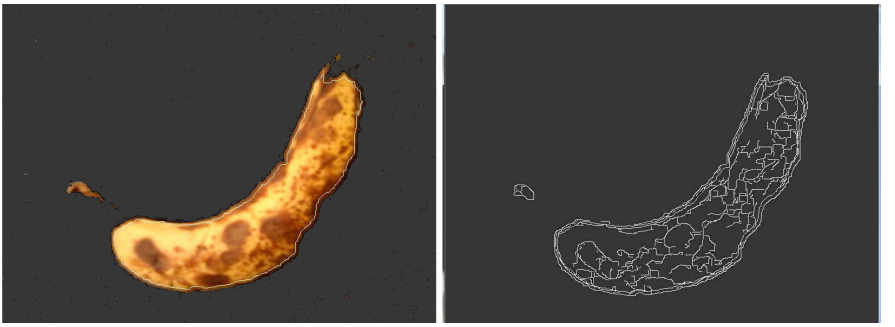
Na jakość drzewa można wpływać przez dobór parametrów przed jego budową. W zależności od doboru wartości parametrów i przesłanek ekonomicznych nawet dla tych samych danych powstają różne drzewa decyzyjne. Różnice te dobrze widać jeśli np. w budowie drzewa wykorzystana zostanie metody przycinania. W przeprowadzonych eksperymentach na skutek przycinania trudno było wygenerować drzewo z głębokością większą niż 3. Przy ustawieniach domyślnych zdarzało się, że niedojrzałe pomarańcze były klasyfikowane jako zielone jabłka. Można to było oczywiście skorygować ustawiając dla pomarańczy większą wartość „Result priority”. W tym wypadku drzewo okazało się bardziej rozbudowane. Na rys. 15.10 można zaobserwować, jak zmienia się znaczenie niektórych cech.



Rys. 15.5: Wybrane parametry tworzenia drzewa. Ustawienie parametru "1Se rule" na "No" pozwala na zbudowanie większego drzewa, które jest potrzebne przy takiej liczbie atrybutów.

Rys. 15.6: Wybrane wartości atrybutów ekonomicznych oraz wyekstrahowane cechy ze zdjęcia.

W przykładzie tym wyraźnie widać, że kolor bardziej decyduje o wynikach klasyfikacji. Łatwo też wyróżnić główne atrybuty dzielące dane. Widać, że ich istotność niektórych z nich jest większa od pozostałych, przy czym znaczenie sporej grupy atrybutów było zaledwie kilkuprocentowe. Kiedy przycinanie drzewa (ang. *harsher prunning*) nie jest wykorzystywana, rozkład znaczenia atrybutów jest bardziej równomierny, a kolor nie jest już taki ważny. Przełożyło się to na problemy z rozpoznawaniem czerwonych jabłek przez system, co mogło być spowodowane dominacją ustawień aspektów ekonomicznych. To również dało się skorygować, jak widać na rys. 15.9, znaczenie koloru zostało poprawione, jak również inny parametr opisujący sam owoc (jego pomarszczenie) uzyskał większe znaczenie.

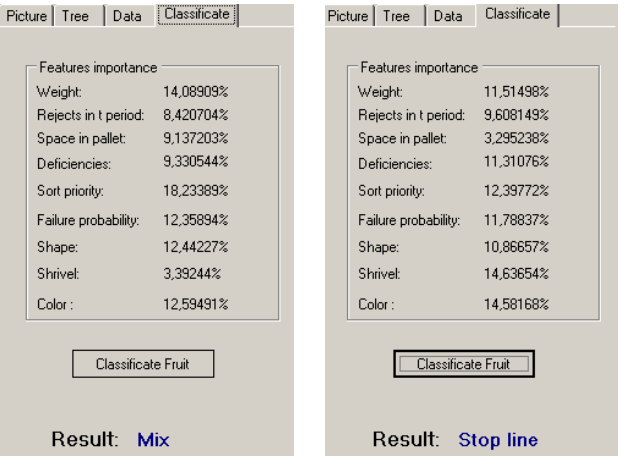


Rys. 15.7: Zaznaczony kontur oraz zmarszczki na owocu.

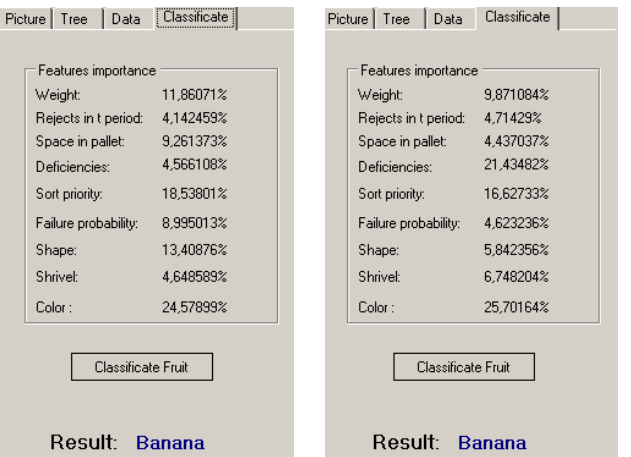
Rys. 15.8: Rezultat predykcji oraz ważność poszczególnych atrybutów.

15.2.5. Podsumowanie

Drzewa decyzyjne doskonale sprawdzają się jako narzędzie do budowania bazy wiedzy dla systemu ekspertowego. Ich zaletą jest przejrzystość generowanych reguł, bardzo dobra klasyfikacja oraz w miarę prosta implementacja. W powyższym rozdziale pokazano jak za pomocą systemu wizyjnego oraz drzew decyzyjnych wspomóc działanie przedsiębiorstwa. Praca 24h na dobę, minimalizacja pomyłek oraz szybka reakcja na spodziewane wydarzenia, to tylko niektóre z zalet zautomatyzowanej sortowni. Co prawda przedstawiony tutaj problem jest czysto fikcyjny (żadna sortownia nie sortuje na jednej taśmie czterech różnych typów owoców), ale można o nim mówić jak o pewnej skali problemów, w których drzewa decyzyjne plasują się na samej górze najlepszych dotychczas wymyślonych rozwiązań.



Rys. 15.9: Metoda przycinania niewykorzystana: a) brak priorytetów rezultatów, b) z priorytetami rezultatów.



Rys. 15.10: Metoda przycinania wykorzystana: a) brak priorytetów rezultatów, b) z priorytetami rezultatów.

Literatura

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone: *Classification and Regression Trees* Wadsworth and Brooks, Monterey, CA, (1984).

[2] B. D. Ripley: *Pattern Recognition and Neural Networks* Cambridge University Press, Cambridge, (1996).

ROZDZIAŁ 16

ANALIZA RYZYKA KREDYTOWEGO Z WYKORZYSTANIEM REGRESJI LOGISTYCZNEJ

K. Kurpanik, A. Trznadel

Nauka o wydobywaniu informacji z ogromnych zbiorów danych nazywana jest eksploracją danych. Jest to dziedzina na pograniczu statystyki, sztucznej inteligencji, zarządzania danymi i wielu innych. Niniejszy rozdział przedstawia umówienie jednej z metod drążenia danych jaką jest regresja logistyczna.

16.1. Metody regresji w drążeniu danych

Cytując za [1]: *Eksploracja danych jest analizą (często ogromnych) zbiorów danych obserwowacyjnych, w celu znalezienia nieoczekiwanych związków i podsumowania danych w oryginalny sposób, tak aby były zarówno zrozumiałe, jak i przydatne dla ich właściciela.*

Eksploracja wiedzy powinna być postrzegana jako całkowity proces – od zrozumienia uwarunkowań biznesowych, przez zebranie i zarządzanie danymi, przygotowanie danych, modelowanie oraz ewaluację modelu aż do wdrożenia modelu. Istnieje wiele różnych metod drążenia danych:

- Regresja liniowa,
- Regresja wielokrotna,
- Regresja logistyczna,
- Naiwna estymacja bayesowska,
- Algorytmy genetyczne.

16.1.1. Regresja logistyczna, a regresja liniowa

Regresja liniowa jest wykorzystywana w celu estymowania wartości oczekiwanej zmiennej celu przy znanych wartościach zmiennych objaśniających. Jednak w przypadku, gdy zmienna celu jest zmienną jakościową stosuje się analogiczny model regresji logistycznej (ang. *logistic regression*). Inaczej mówiąc,

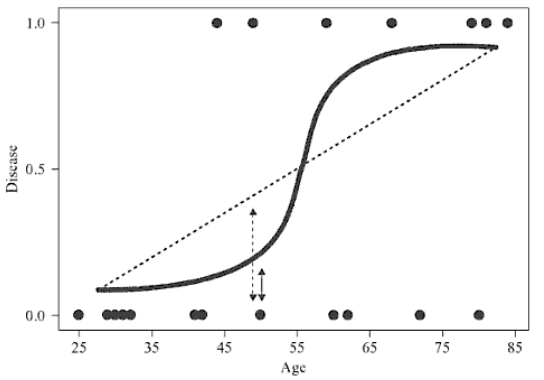
regresja logistyczna jest metodą predykcji zmiennych binarnych (przyjmujących tylko dwie wartości). Więcej informacji na temat zastosowania regresji logistycznej dla zmiennych celu z większa liczbą kategorii można znaleźć w książce Hosmera i Lemeshowa [2].

16.1.2. Przykład regresji logistycznej

Rozważmy medyczny przykład, w którym lekarze są zainteresowani sprawdzeniem zależności między wiekiem pacjenta (zmienna *wiek*) i obecnością (1) lub nieobecnością (0) choroby (zmienna *choroba*) [3]. Dane z obserwacji 20 pacjentów przedstawiono w tab. 16.1 oraz na rys. 16.1.

Numer pacjenta	Wiek	Choroba	Numer pacjenta	Wiek	Choroba
1	25	0	11	50	0
2	29	0	12	59	1
3	30	0	13	60	0
4	31	0	14	62	0
5	32	0	15	68	1
6	41	0	16	72	0
7	41	0	17	79	1
8	42	0	18	80	0
9	44	1	19	81	1
10	49	1	20	84	1

Tab. 16.1: Wiek 20 pacjentów wraz z odpowiedzią czy cierpią na daną chorobę.



Rys. 16.1: Wykres zmiennej choroba wzlędem zmiennej wiek z linią regresji liniowej i regresji logistycznej.

Na wykresie widoczny jest model regresji liniowej, zaznaczony linią przerywaną, oraz model regresji logistycznej, zaznaczony linią ciągłą. Jak widać, regre-

sja logistyczna zakłada, że zależność pomiędzy zmienną objaśniającą a zmienną celu jest nieliniowa. Na wykresie widoczne jest, że regresja liniowa gorzej przewiduje obecność choroby dla większości pacjentów.

16.1.3. Linia regresji logistycznej

Rozważmy za [3] warunkową wartość oczekiwaną zmiennej celu Y, przy zadanej wartości zmiennej objaśniającej $X = x$, czyli $E(Y|x)$. W regresji liniowej zmienna celu jest definiowana jako $Y = \beta_0 + \beta_1 + \epsilon$, skąd $E(Y|x) = \beta_0 + \beta_1 x$. Niech $\pi(x)$ oznacza wartość oczekiwaną $E(Y|x)$ dla regresji logistycznej. Wtedy warunkowa wartość oczekiwana dana jest wzorem

$$\pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}. \tag{16.1}$$

Oczywiście powyższy model jest prawdziwy tylko w przypadku, gdy istnieje jedna zmienną objaśniająca. Aby uprościć rozważania przyjęto na razie takie założenie. Krzywa dana równaniem (16.1) jest nazywana krzywą sigmoidalną. Twórcy metody zdecydowali się na taki rozkład, gdyż jest on bardzo łatwy w modelowaniu i interpretacji. Można zauważyć, że $\pi(x)$ w granicy w $+\infty$ przyjmuje wartość 1, natomiast w granicy w $-\infty$ przyjmuje wartość 0, zatem funkcję (16.1) można interpretować jako prawdopodobieństwo, że wynik jest pozytywny. Analogicznie $1 - \pi(x)$ oznacza prawdopodobieństwo, że wynik jest negatywny.

Model regresji liniowej zakłada, że $Y = \beta_0 + \beta_1 + \epsilon$, gdzie błąd ϵ ma rozkład normalny ze średnią równą zeru i stałą wariancją. W przypadku regresji logistycznej obowiązuje nieco inne założenia. Ponieważ zmienna celu jest binarna, błąd może przyjąć tylko dwie wartości:

- Jeżeli $Y = 1$, co zachodzi z prawdopodobieństwem $\pi(x)$, to $\epsilon = 1 - \pi(x)$,
- Jeżeli $Y = 0$, co ma miejsce z prawdopodobieństwem $1 - \pi(x)$, to $\epsilon = 0 - \pi(x) = -\pi(x)$.

Zatem wariancja ϵ dana jest przez $\pi(x)[1 - \pi(x)]$. Jest to wariancją rozkładu dwumianowego. Zakłada się więc, że zmienna celu w modelu regresji logistycznej $Y = \pi(x) + \epsilon$ ma rozkład dwumianowy z prawdopodobieństwem sukceu $\pi(x)$.

16.1.4. Funkcja logitowa

Przydatnym przekształceniem w przypadku regresji logistycznej jest transformacja logitowa, zdefiniowana następująco:

$$g(x) = \ln \frac{\pi(x)}{1 - pi(x)} = \beta_0 + \beta_1 x. \tag{16.2}$$

Jak widać, funkcja ta ma kilka atrakcyjnych własności regresji liniowej, takich jak liniowość i ciągłość.

16.1.5. Estymacja największej wiarygodności

Jedną z własności regresji liniowej jest to, że można analitycznie wyliczyć optymalne wartości współczynników regresji metodą najmniejszych kwadratów. W przypadku regresji logistycznej, niestety, nie istnieje taka analityczna postać rozwiązania. W tym przypadku stosuje się metodę estymacji największej wiarygodności. Aby wyznaczyć szukane współczynniki definiuje się tzw. funkcję wiarygodności $l(\beta|x)$ zależną od parametrów β, która określa prawdopodobieństwo uzyskania obserwowanych danych x. Zadanie polega na znalezieniu takich wartości β, które maksymalizują $l(\beta|x)$. Jeśli założyć, że wszystkie posiadane obserwacje są niezależne, funkcję wiarygodności można przedstawić w postaci

$$l(\beta|x) = \prod_{i=1}^{n} [\pi(x_i)]^{y_i} [1 - \pi(x_i)]^{1-y_i}. \tag{16.3}$$

Dla uproszczenia obliczeń definiuje się logarytm wiarygodności $L(\beta|x) = \ln(l(\beta|x))$. Estymatory największej wiarygodności znajdujemy różniczkując $L(\beta|x)$ względem β i przyrównując wyniki do zera. Niestety w odróżnieniu od regresji liniowej, nie istnieją analityczne wzory na obliczenie tych różniczek.

16.1.6. Interpretacja wyników regresji logistycznej

Dla przykładu omówionego w podrozdziale 16.1.2 estymatory największej wiarygodności nieznanych parametrów β_0 i β_1 są odpowiednio równe $b_0 = -4,372$ i $b_1 = 0,06696$. Zatem wartość funkcji (16.1) została oszacowana jako

$$\hat{\pi}(x) = \frac{e^{\hat{g}(x)}}{1 + e^{\hat{g}(x)}}, \tag{16.4}$$

z oszacowaną funkcją logitową

$$\hat{g}(x) = -4,372 + 0,06696 \cdot wiek. \tag{16.5}$$

Powyższe równanie można wykorzystać do oszacowania prawdopobieństwa, że pacjent cierpi na daną chorobę, jeśli tylko znany jest jego wiek. Dla przykładu, dla 50-letniego pacjenta $\hat{g}(x) = -4,372 + 0,06696 \cdot 50 = -1,024$ oraz $\hat{\pi}(x) = 0,26$. Zatem oszacowane prawdopodobieństwo, że pacjent cierpi na daną chorobę jest równe 26%, a prawdopodobieństwo, że nie ma tej choroby to 74%.

16.1.7. Interpretacja modelu regresji logistycznej

W regresji liniowej współczynnik nachylanie β_1 jest interpretowany jako zmiana wartości zmiennej celu, gdy wartość zmiennej objaśniającej wzrośnie o jedną jednostkę. W regresji logistycznej współczynnik ten można interpretować analogicznie, przy czym jest on współczynnikiem nachylenia funkcji logitowej. Innymi słowy $\beta_1 = g(x+1) - g(x)$. Aby uprościć interpretację współczynnika β_1 wprowadza się pojęcie *szansy*. Jest ona zdefiniowana jako iloraz prawdopodobieństwa, że wydarzenie nastąpi, przez prawdopodobieństwo, że wydarzenie nie nastąpi. Przykładowo oszacowane prawdopodobieństwo dla 50-letniego

pacjenta, że jest bądź nie jest chory wynoszą odpowiednio 26% i 74%, a zatem *szansa* jest równa $0,26/0,74 = 0,35$. Łatwo zauważyć, że jeżeli bardziej prawdopodobne jest nastąpienie zdarzenia, to wartość szansy jest większa od 1 (w przypadku przeciwnym wartość szansy jest mniejsza od 1). Zatem szansa określa, w jakim stopniu jest bardziej prawodopodobne, że wydarzenie nastąpi w porównaniu z tym, że nie nastąpi. Dla regresji logistycznej z binarną zmienną objaśniającą szansę, że wydarzenie nastąpi dla rekordów z wartością $x = 1$ można obliczyć jako

$$\frac{\pi(1)}{1-\pi(1)} = e^{\beta_0+\beta_1}, \tag{16.6}$$

odpowiednio szansa, że wydarzenie nastąpi dla rekordów z wartością $x = 0$ można obliczyć jako

$$\frac{\pi(0)}{1-\pi(0)} = e^{\beta_0}. \tag{16.7}$$

Dodatkowo można wprowadzić pojęcie ilorazu szans, zdefiniowane jako iloraz szansy, że wydarzenie nastąpi dla rekordów z wartością $x = 1$ przez szansę, że wydarzenie nastąpi dla rekordów z wartością $x = 0$. Zatem

$$OR = \frac{e^{\beta_0+\beta_1}}{e^{\beta_0}}. \tag{16.8}$$

Na przykład, jeżeli przeprowadzono próby kliniczne i otrzymano iloraz szansy wystąpienia raka śluzówki macicy u kobiet, które kiedykolwiek stosowały zastępczą terapię estrogenową, przez szansą wystąpienie tej choroby u kobiet, które nigdy tej terapii nie stosowały wyniosł 5, oznacza to, że jest pięć razy bardziej prawdopodobe, iż kobiety, które stosowały terapię zastępczą zachorują na tę odmianę raka, niż kobiety, które tej terapii nie stosowały.

16.2. Modelowanie ryzyka kredytowego z wykorzystaniem regresji logistycznej

Model do analizy ryzyka kredytowego można wyznaczyć na różne sposoby. Jednym z możliwym rozwiązań jest zastosowanie pakietów do analizy statystycznej. Na rynku istnieje wiele narzędzi umożliwiających wyliczanie modelu regresji logistycznej. W niniejszym rozdziale zostanie przedstawiony sposób wyliczenia modelu w aplikacji WEKA (ang. *Waikato Environment for Knowledge Analysis*).

16.2.1. Zbiór danych

W celu ilustracji ścieżki postępowania prowadzącej od postawienia problemu aż do otrzymania gotowego modelu, wykorzystany zostanie zbiór danych `GERMAN` (`http://www.dataminingconsultant.com/data/german.dat`). Jest to zbiór zawierający 1000 danych kredytowych z niemieckiego banku podzielone na 2 klasy: kredyty przyznane oraz kredyty odrzucone. Każdy rekord opisany jest 20 atrybutami:

1. status istniejącego konta bankowego,
2. okres posiadanego konta podawany w miesiącach,
3. historia konta bankowego,
4. cel udzielenia kredytu,
5. wysokość kredytu,
6. oszczędności bankowe / obligacje,
7. okres zatrudnienia,
8. wielkość procentowa raty w stosunku do dochodu netto,
9. status personalny i płeć,
10. inni dłużnicy i żyranci,
11. okres obecnego miejsca zamieszkania,
12. posiadane mienie/nieruchomości,
13. wiek kredytobiorcy,
14. inne systemy ratalne,
15. rodzaj zakwaterowania,
16. liczba obecnych kredytów w tym banku,
17. typ zatrudnienia,
18. liczba poręczycieli,
19. telefon,
20. czy pracownik jest obcokrajowcem.

Aby skorzystać z dostarczonego zbioru danych, należy na początku przekształcić go do jednej z postaci obsługiwanych przez program WEKA. Najprościej jest to zrobić, zapisując plik `german.dat` w formacie csv, wykorzystując w tym celu np. program OpenOffice.org Calc.

16.2.2. Wykorzystanie aplikacji WEKA

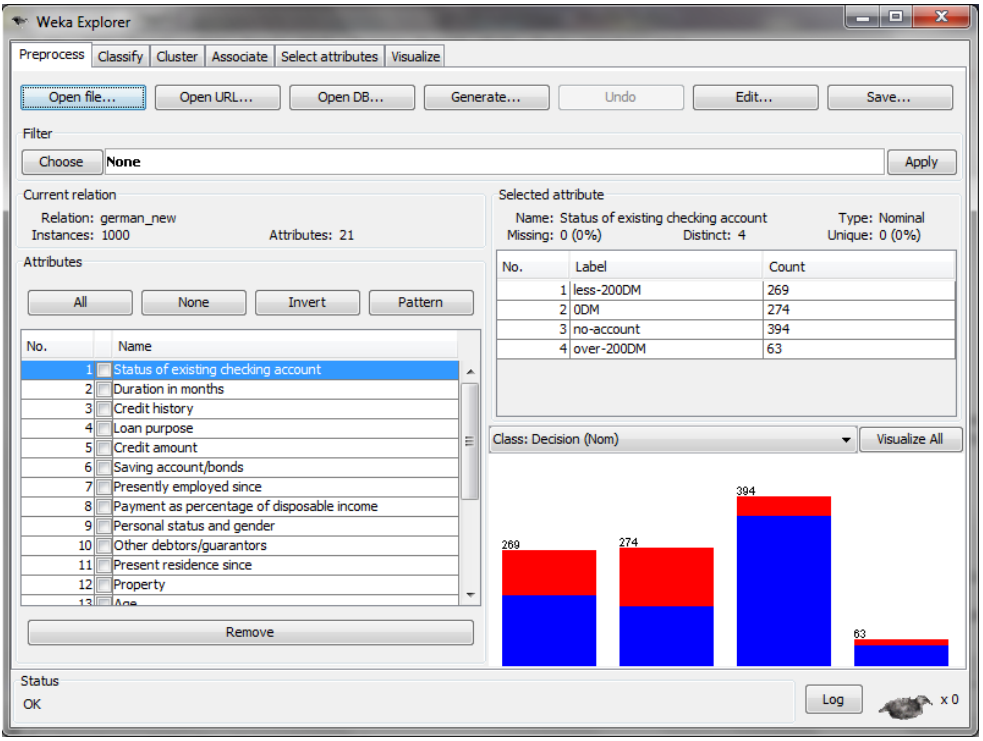
W celu zbudowania modelu należy na początek załadować plik zawierający zbiór uczący. Zrobić to można w następującej sekwencji czynności:

1. Otwórz okno `Explorer` programu WEKA.
2. Na zakładce `Preprocess` kliknij `Open file` i określ ścieżkę do pliku wejściowego `germat.csv`.

Okno `Explorer` programu WEKA pokazuje podstawowe informacje na temat pliku uczącego (jak pokazano na rys. 16.2). Dwadzieścia zmiennych objaśniających i zmienną klasy pokazano w ramce `Attributes`. Ramka `Status` na dole okna informuje, że plik został poprawnie wczytany przez program WEKA. Aby zdefiniować, jaki model ma zostać zbudowany należy wykonać następującą sekwencję czynności:

1. Wybierz zakładkę `Classify`.
2. Wewnątrz ramki `Classifier` naciśnij przycisk `Choose`.
3. Wybierz `Classifiers` -> `Functions` -> `Logistic`.
4. Ponieważ w doświadczeniu jeden zbiór zostanie wykorzystany zarówno jako zbiór uczący i testujący, zatem wewnątrz ramki `Test options` należy wybrać `Percentage split`, i ustalić ile procent całego zbioru ma zawierać zbiór testujący.
5. Naciśnij `Start`, aby zbudować model.

WEKA buduje model regresji logistycznej i pokazuje wyniki w oknie `Classifier output`. Znajduje się tam kompletny model regresji logistycznej, wraz z wynikami testów. Otrzymany wynik można wykorzystać do przewidywania ryzyka kredytowego.



Rys. 16.2: Okno eksploracji WEKA - zakładka wstępnego przetworzenia.

16.2.3. Model otrzymany z aplikacji WEKA dla zbioru uczącego `german.dat`

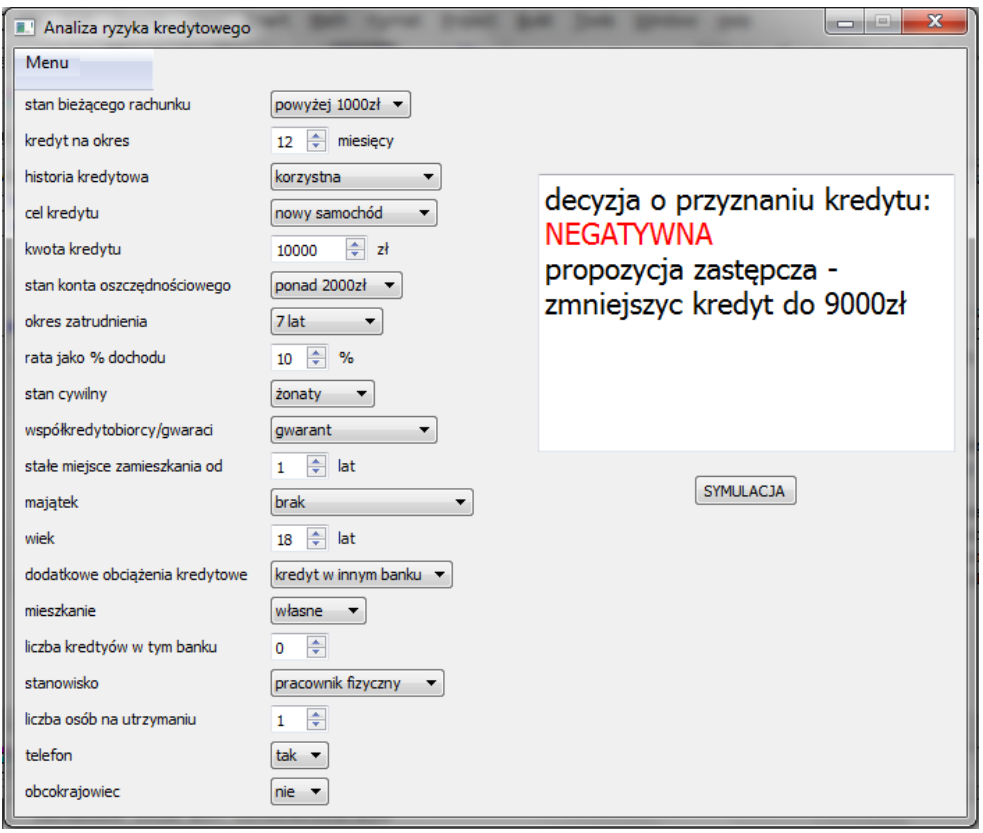
Model przedstawiony w tym rozdziale został wygenerowany w aplikacji WEKA. Jest to model regresji logistycznej, którego parametry przedstawiono poniżej.

$\beta_0 =$ 3.0507
$\beta_1 =$ -0.4006 $\rightarrow x_1$ – Status of existing checking account = less-200DM
$\beta_2 =$ -0.7802 $\rightarrow x_2$ – Status of existing checking account = 0DM
$\beta_3 =$ 0.9337 $\rightarrow x_3$ – Status of existing checking account = no-account
$\beta_4 =$ 0.1867 $\rightarrow x_4$ – Status of existing checking account = over-200DM
$\beta_5 =$ -0.0275 $\rightarrow x_5$ – Duration in months
$\beta_6 =$ -0.2323 $\rightarrow x_6$ – Credit history = duly-till-now
$\beta_7 =$ 0.6253 $\rightarrow x_7$ – Credit history = critical
$\beta_8 =$ 0.0345 $\rightarrow x_8$ – Credit history = delay
$\beta_9 =$ -0.9689 $\rightarrow x_9$ – Credit history = bank-paid-duly
$\beta_{10} =$ -0.7624 $\rightarrow x_{10}$ – Credit history = all-paid-duly
$\beta_{11} =$ 0.0646 $\rightarrow x_{11}$ – Loan purpose = business

β_{12} = 0.1015 → x_{12} – Loan purpose = furniture
β_{13} = -0.6949 → x_{13} – Loan purpose = new-car
β_{14} = 0.1983 → x_{14} – Loan purpose = radio-tv
β_{15} = -0.7332 → x_{15} – Loan purpose = education
β_{16} = -0.1667 → x_{16} – Loan purpose = domestic-app
β_{17} = -0.485 → x_{17} – Loan purpose = repairs
β_{18} = 0.9689 → x_{18} – Loan purpose = used-car
β_{19} = 0.7972 → x_{19} – Loan purpose = others
β_{20} = 1.3619 → x_{20} – Loan purpose = retraining
β_{21} = -0.0001 → x_{21} – Credit amount
β_{22} = 0.9012 → x_{22} – Saving account/bonds = over1000DM
β_{23} = -0.4439 → x_{23} – Saving account/bonds = less100DM
β_{24} = -0.0677 → x_{24} – Saving account/bonds = less500DM
β_{25} = 0.5031 → x_{25} – Saving account/bonds = unknown
β_{26} = -0.0657 → x_{26} – Saving account/bonds = less1000DM
β_{27} = -0.2288 → x_{27} – Presently employed since = one-year
β_{28} = -0.1144 → x_{28} – Presently employed since = four-years
β_{29} = -0.0216 → x_{29} – Presently employed since = over-seven
β_{30} = 0.5523 → x_{30} – Presently employed since = seven-years
β_{31} = -0.2933 → x_{31} – Presently employed since = unemployed
β_{32} = -0.3281 → x_{32} – Payment as percentage of disposable income
β_{33} = -0.1241 → x_{33} – Personal status and gender = married-male
β_{34} = 0.3227 → x_{34} – Personal status and gender = single-male
β_{35} = -0.216 → x_{35} – Personal status and gender = female-divorced
β_{36} = -0.4917 → x_{36} – Personal status and gender = male-divorced
β_{37} = -0.1804 → x_{37} – Other debtors/guarantors = none
β_{38} = -0.6126 → x_{38} – Other debtors/guarantors = co-applicant
β_{39} = 0.7973 → x_{39} – Other debtors/guarantors = guarantor
β_{40} = -0.0031 → x_{40} – Present residence since
β_{41} = 0.2568 → x_{41} – Property = real-estate
β_{42} = 0.0617 → x_{42} – Property = car
β_{43} = -0.4701 → x_{43} – Property = none
β_{44} = -0.0249 → x_{44} – Property = building-society
β_{45} = 0.0144 → x_{45} – Age
β_{46} = -0.3314 → x_{46} – Other installment plans = bank
β_{47} = -0.2031 → x_{47} – Other installment plans = stores
β_{48} = 0.322 → x_{48} – Other installment plans = none
β_{49} = 0.0895 → x_{49} – Housing = own
β_{50} = -0.3549 → x_{50} – Housing = rent

$\beta_{51} =$ 0.3512 $\rightarrow x_{51}$ – Housing = free
$\beta_{52} =$ -0.2811 $\rightarrow x_{52}$ – Number of existing credits in this bank
$\beta_{53} =$ 0.3882 $\rightarrow x_{53}$ – Job = skilled
$\beta_{54} =$ 0.422 $\rightarrow x_{54}$ – Job = unskilled-resident
$\beta_{55} =$ 0.4667 $\rightarrow x_{55}$ – Job = management
$\beta_{56} =$ 0.9459 $\rightarrow x_{56}$ – Job = unemployed-non-resident
$\beta_{57} =$ -237.20 $\rightarrow x_{57}$ – Job = Unskilled-resident
$\beta_{58} =$ -0.2489 $\rightarrow x_{58}$ – Number of people being liable to provide maintenance for
$\beta_{59} =$ -0.2959 $\rightarrow x_{59}$ – Telephone
$\beta_{60} =$ 1.3908 $\rightarrow x_{60}$ – Foreign worker

16.2.4. Wykorzystanie aplikacji do analizy ryzyka kredytowego



Rys. 16.3: Interfejs aplikacji do analizy ryzyka kredytowego.

Model przedstawiony w podrozdziale 16.2.3 został wykorzystany do budowy aplikacji pozwalającej na szybkie podjęcie decyzji kredytowej. Umożliwia on do-

konanie klasyfikacji klientów banku pod względem ryzyka kredytowego. Interfejs programu przedstawiono na rys. 16.3. Program względnia wszystkie parametry opisane w podrozdziale 16.2.1 i oprócz decyzji kredytowej wylicza również alternatywną propozycję kredytową, czyli do jakiej kwoty należy zmiejszyć kredyt, aby klient mógł go otrzymać.

16.2.5. Podsumowanie

W niniejszym rozdziale pokazano w jaki sposób można wykorzystać regresję logistyczną do modelowania ryzyka kredytowego. Testy przeprowadzone na otrzymanym modelu pokazały, że pozwala on na podjęcie prawidłowej decyzji kredytowej w ok. 80% przypadków, co jest bardzo dobry wynikiem. Dla porównania model zbudowany za pomocą metody naiwnej klasyfikacji bayesowskiej, dla tego samego zbioru danych, miał skuteczność tylko 70%. Można zatem stwierdzić, iż regresja logistyczna z całą pewnością nadaje się do modelowania zjawisk, dla których zmienna celu jest zmienną binarną. Przeprowadzone badania i otrzymana aplikacja utwierdziły autorów w przekonaniu, że wiele metody analizy statystycznej pozwalają na modelowanie rzeczywistych zjawisk z nieciągłą zmienną celu.

Literatura

[1] D. Hand, H. Mannila, and P. Smyth: *Eksploracja danych* Wydawnictwa Naukowo-Techniczne, (2005).
[2] D. Hosmer and S. Lemeshow: *Applied Logistic Regression* Wiley, (2000).
[3] D. Larose: *Metody i modele eksploracji danych* Informatyka - Zastosowania Wydawnictwo Naukowe PWN, (2008).

ROZDZIAŁ 17

METODY WNIOSKOWANIA W SYSTEMACH EKSPERTOWYCH (PROGRESYWNE I REGRESYWNE)

S. Wikaliński, P. Zych

Celem niniejszego rozdziału jest przedstawienie podstawowych metod wnioskowania, stosowanych w systemach ekspertowych. Zostały w nim omówione podstawowe zadania, rodzaje i zastosowania systemów ekspertowych, ze szczególnym uwzględnieniem systemów regułowych. Na przykładzie zrealizowanego projektu zademonstrowano sposoby tworzenia prostych systemów ekspertowych oraz implementacji metod wnioskowania progresywnego i regresywnego w środowisku SWI-Prolog.

17.1. Zagadnienia teoretyczne

17.1.1. Kim jest ekspert?

Według definicji prof. Antoniego Niederlińskiego [1]:

> Ekspert to człowiek posiadający specjalistyczną wiedzę w pewnej dziedzinie i umiejętność stosowania jej dla rozwiązywania problemów tej dziedziny, nabyte w wyniku studiów i praktyki.

Wysokiej klasy specjalista, potrafiący rozwiązać bardzo skomplikowane problemy dotyczące np. układów sterowania silników odrzutowych, poświęcając swój czas na zgłębianie jednej tylko wąskiej specjalności, w innych kwestiach pozostaje kompletnym laikiem. W cywilizacji opartej na wiedzy konieczność tak skrajnej specjalizacji stała się faktem. To dzięki niej, żyjąc w XXI wieku, możemy korzystać z najnowszych zdobyczy techniki oraz kultury, jednocześnie nie zaprzątając sobie głowy olbrzymią ilością zbędnych informacji [2].

17.1.2. Systemy ekspertowe

Wiedza i umiejętności ekspertów, często niedoceniane przez zarządzających kadrami, posiadają dziś niejednokrotnie większą wartość niż cały majątek trwały przedsiębiorstw. Są one jednak zgromadzona w ulotnej ludzkiej pamięci, co czyni je bardzo trudnymi do zachowania. Ponadto, nawet najlepsi eksperci są „tylko" ludźmi, w związku z czym popełniają typowo ludzkie błędy, wynikające ze zmęczenia bądź chwili nieuwagi. Kolejnym problemem jest długi czas, jaki potrzebuje człowiek na podjęcie decyzji w przypadku rosnącej ilości danych. Dlatego też tak istotnym wyzwaniem jest opracowanie skutecznych metod gromadzenia wiedzy eksperckiej w nieulotnej pamięci komputerów oraz zastąpienie ludzkich procesów myślowych szybkimi i bezbłędnymi obliczeniami maszynowymi. Jedynym znanym sposobem, aby tego dokonać, jest tworzenie komputerowych programów, bazujących na osiągnięciach sztucznej inteligencji, zwanych systemami ekspertowymi, lub też po prostu „systemami z bazą wiedzy"[1].

Podsumowując: System ekspertowy jest programem (bądź też zbiorem programów), w którym zawarta jest specjalistyczna, ekspercka wiedza. Rozwiązuje on problemy zlecane ekspertom w taki sposób, że skorzystać z niego może nawet osoba nie znające się na danej dziedzinie [1].

Systemy tego typu stosuje się obecnie prawie wszędzie: przy diagnozowaniu chorób, poszukiwaniu złóż minerałów, udzielaniu porad prawnych. Powszechnie stosowane są w diagnostyce uszkodzeń (czasem także przy ich naprawie), sterowaniu procesami, a także na etapie projektowania np. układów scalonych. Znalazły one również szerokie zastosowanie w analizie danych oraz prognozowaniu (między innymi pogody). Jednym z przykładów częstego zastosowania systemów ekspertowych jest wstępna ocena wniosków kredytowych.

Systemy ekspertowe dzielimy ze względu na zastosowanie, na trzy ogólne grupy:

- systemy doradcze - najbardziej popularne systemy ekspertowe, których zadaniem jest szybkie dostarczenie użytkownikowi rozwiązania problemu, które może być przez niego zaakceptowane, odrzucone bądź zmienione,
- systemy krytykujące - których zadaniem jest sprawdzenie poprawności rozwiązania dostarczonego przez człowieka i wydanie odpowiedniej opinii,
- systemy podejmujące decyzję bez kontroli człowieka - całkowicie autonomicznie, bez konieczności akceptowania wyników przez użytkownika. Pracują najczęściej tam, gdzie wywiązują się z zadania znacznie lepiej od człowieka, bądź też udział człowieka jest niemożliwy.

Nie wszystkie programy komputerowe, rozwiązujące specjalistyczne problemy, możemy od razu nazywać systemami ekspertowymi. Najczęściej postuluje się wobec nich pewną zasadniczą własność: baza wiedzy takiego programu musi być oddzielona od systemu wnioskującego. Mówiąc inaczej, program wykonywalny nie może zawierać gotowego algorytmu-recepty na rozwiązanie problemu. Jest on tylko systemem wnioskującym, rodzajem „silnika" wykonującego operacje na bazie wiedzy oraz danych wejściowych i w efekcie generującego rozwiązanie. Taka struktura systemu daje nam możliwość łatwej modyfikacji bazy

wiedzy - dodawania do niej nowych reguł postępowania bądź informacji. System taki powinien zostać wyposażony także w edytor bazy wiedzy oraz interfejs użytkownika.

17.1.3. Metody reprezentacji wiedzy w systemach ekspertowych

Spotykane są różne metody reprezentacji wiedzy, takie jak reguły, wektory wiedzy, sieci semantyczne, sztuczne sieci neuronowe, algorytmy genetyczne lub zbiory rozmyte [3]. Każdy z wymienionych sposobów umożliwia zgromadzenie pewnej ilości wiedzy eksperckiej, chociaż jej reprezentacja (oraz modyfikacja) może przebiegać w różny sposób. Regułowe bazy wiedzy, zbudowane na pewnych zasadach logiki, jawnie obrazują tok i zasady podejmowania decyzji w sposób bardziej przejrzysty i zrozumiały dla człowieka. Pozwalają też na stosunkowo łatwą edycję poszczególnych reguł, czego nie można już powiedzieć na przykład o sieciach neuronowych. Sieć neuronowa, wytrenowana na pewnych danych wejściowych, często bywa „czarną skrzynką" - ciężko jest prześledzić jej działanie, nie ma też możliwości poprawienia jej wyników w inny sposób, niż przez wytrenowanie jej od nowa. Metody wnioskowania omówione w tym rozdziale w sposób naturalny dotyczą więc głównie systemów regułowych, bądź też regułowo-modelowych (RMSE).

17.1.4. Klasyfikacja regułowych baz wiedzy

Bazy wiedzy systemów regułowych można podzielić na dwa sposoby. Pierwszy sposób klasyfikacji opiera się na różnicach w rodzaju stosowanej w systemie logiki. Wyróżnia się w nim:

- bazy wiedzy dokładne - oparte na tradycyjnej, arystotelesowskiej logice dwuwartościowej (prawda/fałsz),
- bazy wiedzy niepewne - oparte na zmodyfikowanej stanfordzkiej algebrze współczynników pewności. Prawdopodobieństwu wniosków każdej reguły przyporządkowuje się liczbą z przedziału [-1,1], nazywaną współczynnikiem pewności.

Drugim sposobem podziału baz wiedzy jest podział ze względu na tryb zagnieżdżania reguł oraz przyjęte założenia wobec świata. Mówiąc najogólniej, można przyjąć dwa założenia odnośnie świata, w którym nasz system działa: może on być otwarty lub zamknięty. Nie jest to jednak kwestia filozoficzna - ma ona bowiem decydujący wpływ na zachowanie naszego systemu podczas wnioskowania. Dlaczego? Otóż przyjmując, że świat jest „otwarty", przyznajemy jednocześnie, że mogą w nim znajdować się fakty, których nie znamy. Jeżeli nie mamy przesłanek potwierdzających dany fakt, nie można być pewnym, że jest on nieprawdziwy - co najwyżej nieokreślony (nieukonkretniony)! Nie wolno więc używać zanegowanych wniosków jednych reguł w innych regułach. Z kolei przy założeniu zamkniętego świata przyjmuje się, że posiadana wiedza jest kompletna i jeśli nie można czegoś dowieść z dostępnych faktów - bez wątpienia jest to nieprawdziwe. Stąd przy założeniu zamkniętego świata można śmiało korzystać

z negacji przy zagnieżdżaniu reguł. Z tego powodu rozróżniane są dwa rodzaje bez wiedzy:

- bazy wiedzy elementarne - przyjmują założenie otwartego świata, w związku z czym regułom zagnieżdżonym w innych regułach nie może towarzyszyć negacja,
- bazy wiedzy rozwinięte - przyjmują założenie zamkniętego świata, zagnieżdżaniu reguł może towarzyszyć negacja.

Różnicę między implikacją logiki a implikacją regułową korzystającą z założenia otwartego świata przedstawiają tabele 17.2 i 17.3. Ogólny podział baz wiedzy przedstawiono w tab. 17.1.

Tab. 17.1: Klasyfikacja baz wiedzy.

elementarne dokładne	rozwinięte dokładne
elementarne niepewne	rozwinięte niepewne

Tab. 17.2: Implikacja logiki

q	p	$q \Rightarrow p$
prawda	prawda	prawda
fałsz	prawda	prawda
fałsz	fałsz	prawda

Tab. 17.3: Implikacja regułową korzystającą z założenia otwartego świata.

q	p	$q \rightarrow p$
prawda	prawda	prawda
fałsz	nieukonkretnione	prawda

17.1.5. Wnioskowanie

Zdolność do wyciągania wniosków wydaje się być nieodłączną cechą ludzkiego rozumowania. Aby omówić metody wnioskowania systemów ekspertowych, należy wpierw zdefiniować samo wnioskowanie. Co jest charakterystyczne dla procesów wnioskowania? Po pierwsze, wniosków nie można wyciągać z niczego. Do przeprowadzenia procesu wnioskowania potrzebne są jakieś informacje, dane, fakty. Efektem procesu wnioskowania są natomiast bez wątpienia nowe, wcześniej nieznane fakty. Aby je wygenerować należy posiadać szerszą wiedzę, należy znać wzajemne relacje faktów posiadanych z faktami, których próbuje się dowieść - na przykład ich wzajemne wykluczanie się lub wynikanie. Zależności takie nazywane są regułami - tworzą one bazę wiedzy systemów regułowych.

Wyróżniane są dwa podstawowe rodzaje wnioskowania: wnioskowanie progresywne (w przód) oraz regresywne (wstecz). Dalsza klasyfikacja zależy od rodzaju bazy wiedzy. Rozróżnia się więc wnioskowanie elementarne dokładne, rozwinięte dokładne, elementarne niepewne oraz rozwinięte niepewne [1].

Wnioskowanie progresywne

Wnioskowanie progresywne, zwane także wnioskowaniem w przód (ang. *forward chaining, data driven, event driven*) jest wnioskowaniem od warunków do wniosku. Proces ten zaczyna się od pewnych znanych warunków początkowych, aby za pomocą reguł wyznaczać kolejno nowe fakty, będące wnioskami poprzednich. Celem tego typu wnioskowania jest wyznaczenie wszystkich wniosków prawdziwych i wszystkich wyników możliwych do wygenerowania ze zbioru faktów początkowych. Dokonuje się tego testując wielokrotnie wszystkie nietestowane dotąd reguły, zaczynając od pierwszej nietestowanej reguły. Jeżeli testowanie zostanie przeprowadzone dla wszystkich reguł, nazywane jest to cyklem testowania. W czasie cyklu testowania zostają wyznaczone nowe fakty, a następnie wnioskowanie przeprowadza się od początku dla rozszerzonej bazy faktów. Cykle testowania kończą się stanem ustalonym, czyli takim stanem, gdzie w poprzednim cyklu nie został wygenerowany żaden nowy wniosek ani żaden nowy wynik [3]. Testowana reguła może zwrócić różne wyniki :

- Fakty potrzebne do spełnienia reguły są prawdziwe, a więc wniosek jest też prawdziwy.
- Fakty potrzebne do spełnienia reguły nie są prawdziwe, reguła nigdy nie będzie spełniona.
- Fakty są nieokreślone i reguła jest pomijana, natomiast w kolejnym testowaniu może się okazać spełniona lub niespełniona.
- Jeżeli kilka reguł ma ten sam wniosek, to jeżeli chociaż jedna będzie spełniona to wniosek jest prawdziwy.
- Jeżeli jakiś reguła zwróciła prawdziwy wniosek to można ją pominąć ponieważ nie ma możliwości aby wniosek uznany za prawdę przestał nią być w wyniku testowania innych reguł.

Przykład wnioskowania w przód Zakładamy, że A, B, D są faktami.

$$\begin{gathered} F \rightarrow G \\ A\ B \rightarrow C \\ D\ E \rightarrow F \\ A\ C \rightarrow E \\ J \rightarrow K \end{gathered}$$

Podczas pierwszego cyklu testujemy regułę $F \rightarrow G$ i pomijamy ją, gdyż nic nie wiemy o F. Z drugiej reguły $A\ B \rightarrow C$ możemy wywnioskować, że C jest prawdziwe, a więc do zbioru faktów dochodzi jeszcze wniosek C. Następną regułę opuszczamy, bo nic nie wiemy o E, a z przedostatniej reguły $A\ C \rightarrow E$ wiemy, że A i C

jest prawdziwe, więc wnioskujemy, że E jest prawdziwe. Ostatnią regułę pomijamy.

Podczas drugiego cyklu testowania pierwszą regułę w dalszym ciągu omijamy, druga reguła była spełniona i nie możemy z niej już wygenerować nowych faktów, więc ją pomijamy. Z trzeciej reguły D E → F możemy wywnioskować, że F jest prawdziwe. Reguła A C → E jest już spełniona, a ostatnią regułę zgodnie z zasadami pomijamy.

Podczas trzeciego cyklu możemy z pierwszej reguły wywnioskować, że G jest prawdziwe, bo F jest prawdziwe, a następnie pominąć następne reguły. Podczas czwartego cyklu wszystkie reguły spełnione pomijamy, a więc zostaje tylko jedna reguła nieokreślona J →K i jest to zatem stan ustalony.

Podsumowywując powyższy przykład można zauważyć, że niektóre reguły (np. F → G) są nieokreślone w kolejnych cyklach aż do wygenerowania faktów, z których można wyciągnąć wnioski. Niektóre reguły tak jak J → K pozostają do końca nieokreślone. Idea wnioskowania w przód polega na testowaniu wszystkich reguł, zaczynając od pierwszej nietestowanej reguły do momentu, gdy już nie można nic wywnioskować. W omówionym przykładzie po trzecim cyklu testowania 4 reguły są spełnione, a jedna pozostaje nieokreślona, a więc jest to stan ustalony.

Wnioskowanie regresywne (wstecz)

Wnioskowanie regresywne, zwane także wnioskowaniem wstecz (ang. *backward chaining, goal driven, expectation driven*) przyjmuje się odwrotny kierunek wnioskowania. Zaczyna się bowiem od postawienia pewnej hipotezy głównej, będącej wnioskiem jednej z reguł. Następnie weryfikuje się jej warunki. Jeśli są one faktami początkowymi, wtedy znana jest ich wartość logiczna. Mogą one też być są wnioskami innych reguł - nazywane są wtedy hipotezami pomocniczymi, które należy sprawdzić w podobny sposób. Można powiedzieć, że proces wnioskowania regresywnego przebiega w sposób rekurencyjny w kierunku od hipotezy do faktów początkowych. Celem tego typu wnioskowania jest sprawdzenie prawdziwości jednej, konkretnej hipotezy bez potrzeby generowania dużej ilości niepotrzebnych wniosków, co ma przeważnie miejsce podczas wnioskowania w przód. Wnioskowanie w przód generuje dużą ilość nowych faktów. Co zrobić, gdy interesuje nas tylko jeden z faktów? Przykładowo interesuje nas czy G jest prawdziwe. W tym przypadku G nazywane jest hipotezą i podczas wnioskowania hipoteza ta jest weryfikowana albo odrzucana. Aby hipotezę można było zweryfikować musi ona być wnioskiem jednej z reguł. W przypadku wnioskowania wstecz sprawdzana jest najpierw reguła, której wnioskiem jest badana hipoteza. Reguła ta posiada warunki niedopytytywalne, które stają się hipotezami pomocniczymi pierwszego rzędu.

Przykład wnioskowa wstecz z wykorzystaniem tego samego co wcześniej przykładu Zakładamy A, B, D są faktami. Sprawdzana jest hipoteza : G fakt ?

$$F \rightarrow G$$

A B → C
D E → F
A C → E
J → K

Przebieg weryfikacji hipotezy głównej zaczyna się na pierwszej regule F → G, hipoteza G jest wnioskiem faktu F. W tym przypadku F staje się hipotezą pomocniczą pierwszego rzędu. Należy sprawdzić z czego wywnioskować fakt F. Fakt F jest wnioskiem reguły D E → F, D jest faktem, natomiast E jest hipotezą pomocniczą drugiego rzędu. E jest wnioskiem reguły czwartej A C → E, gdzie A i C są faktami, a z tego wynika, że G jest faktem.

17.2. Opis projektu

17.2.1. Cel

Celem projektu było stworzenie prostego programu przedstawiającego podstawowe własności systemów ekspertowych oraz stosowane w nich metody wnioskowania progresywnego oraz regresywnego. Jednym z założeń było oddzielenie bazy wiedzy od silnika wnioskującego oraz stworzenie możliwości dodawania nowych faktów i reguł. Istotne wydawało się także stworzenie prostego i czytelnego interfejsu graficznego, umożliwiającego łatwą modyfikację bazy wiedzy.

Powyższe cele zrealizowane zostały z wykorzystaniem otwartej implementacji języka Prolog (SWI-Prolog). W celu stworzenia interfejsu graficznego wykorzystana została nakładka obiektowa XPCE, dostępna standardowo wraz kompilatorem SWI-Prologa.

17.2.2. Opis działnia programu

Zadaniem stworzonego programu jest diagnozowanie chorób zakaźnych na podstawie występujących u pacjenta objawów. System rozpoznaje 6 różnych chorób na podstawie około 30 różnych objawów. Każdej z chorób przydzielona zostaje lista możliwych objawów, a także lista symptomów, która z daną chorobą nigdy nie występuje. W trakcie pracy z programem istnieje możliwość wyboru sposobu wnioskowania. We wnioskowaniu progresywnym na podstawie podanych informacji Prolog dokonuje kolejnych podstawień. Sprawdzając wszystkie możliwości, wyświetla tylko te choroby, których objawy zgadzają się z faktami zawartymi w bazie wiedzy. W przypadku wnioskowania regresywnego w wyświetlonym okienku należy wybrać chorobę, którą podejrzewamy u pacjenta, a system sprawdzi tylko postawioną hipotezę.

17.2.3. Instalacja, kompilacja i obsługa programu

Program dostarczony jest w wersji skompilowanej dla systemu Windows. Aby uruchomić program w tym środowisku wystarczy rozpakować plik `Diagnozowanie chorob.zip` do jednego katalogu, a następnie uruchomić plik wykonywalny `run.exe`. Wraz z plikiem wykonywalnym dostępny jest także kod

źródłowy, znajdujący się w pliku `start.pl`. Pracując pod dowolnym systemem operacyjnym można go zaimportować do SWI-Prologa za pomocą polecenia:

```
?- consult(start).
```

Następnie w celu uruchomienia programu wystarczy wpisać:

```
?- main.
```

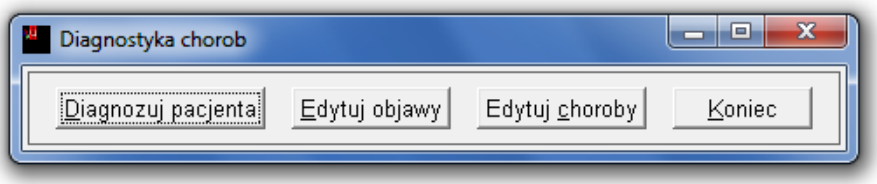
Możliwa jest też kompilacja programu do postaci wykonywalnej. W tym celu po wczytaniu pliku `start.pl` z poziomu SWI-prologa należy użyć polecenia:

```
?- kompilacja('NazwaPliku').
```

gdzie `NazwaPliku` to nazwa wynikowego pliku wykonywalnego, który chcemy otrzymać w wyniku kompilacji. Niezwykle ważne jest, aby skompilowany plik przenosić wraz z resztą plików znajdujących się w katalogu - są one bowiem wykorzystywane przez program i ich usunięcie powoduje brak możliwości uruchomienia systemu. Jedynym wyjątkiem jest plik `start.pl`.

17.2.4. Opis interfejsu użytkownika

Stworzony system wyposażony został w graficzny interfejs użytkownika. Po uruchomieniu programu pojawia się okno dialogowe z czterema kolejnymi przyciskami (rys. 17.1).

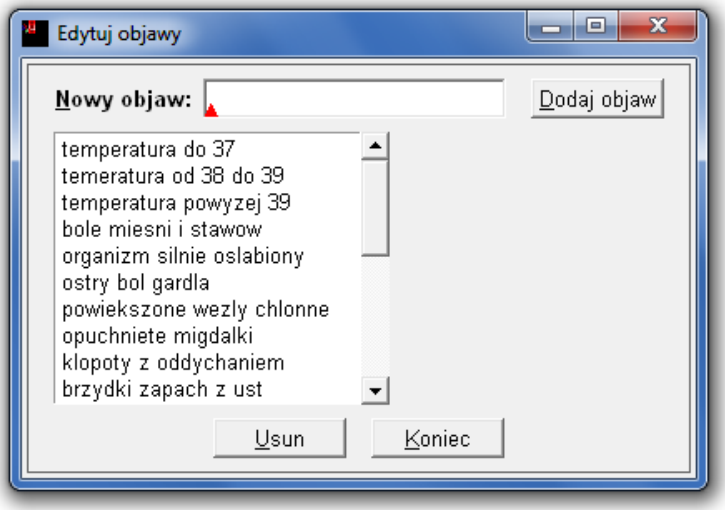


Rys. 17.1: Okno główne programu.

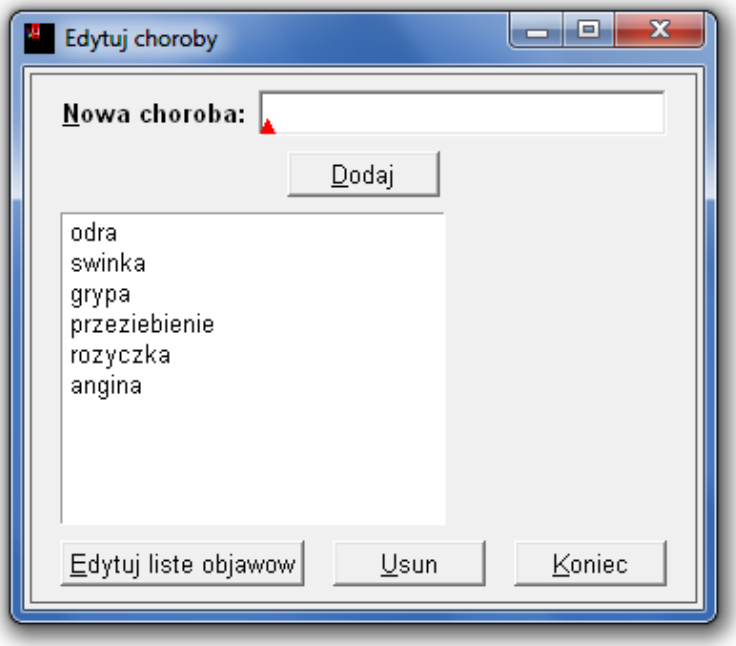
Po kliknięciu przycisku *Edytuj objawy* otwiera się nowe okienko (rys. 17.2) wraz z listą aktualnie dostępnych symtomów. Obok listy wyboru pojawia pole tekstowe oraz przycisk *Dodaj objaw*. Po wpisaniu nowego objawu i kliknięciu przycisku *Dodaj objaw* lista zostaje zaktualizowana.

Trzeci przycisk, przedstawiony na rys. 17.3 umożliwia dodanie nowej choroby do bazy danych. Po jego kliknięciu pojawia się okno dialogowe z listą chorób znajdujących się w bazie danych.

Każdą chorobę można zaznaczyć, a następnie klikając *Edytuj objawy* można zmienić objawy do niej przyporządkowane. W przypadku edytowania symptomów choroby pojawia się nowe okienko wraz z 3 listami do wyboru (rys. 17.4). W środkowej liście znajdują się elementy nieprzyporządkowane do żadnej z pozostałych. Używając przycisków <== i ==> można przenosić je między listami. Podczas edycji objawów choroby niemożliwe jest przyporządkowanie tego sa-

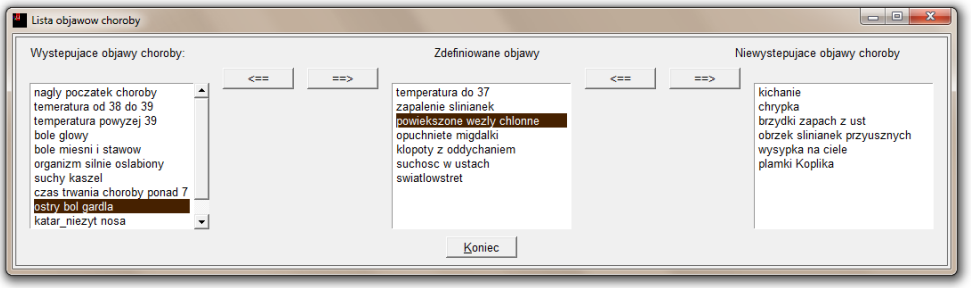


Rys. 17.2: Okno dodawania objawów.



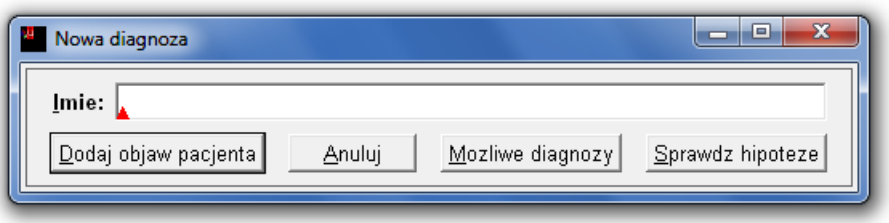
Rys. 17.3: Okno edycji chorób.

mego elementu do obu grup, co rozwiązuje problem potencjalnego braku spójności bazy wiedzy.



Rys. 17.4: Okno edycji objawów choroby.

Po kliknięciu na przycisku *Diagnozuj pacjenta* pojawia się okno (rys. 17.5) z polem tekstowym, gdzie wpisać należy imię pacjenta. Po naciśnięciu przycisku *Dodaj*, pojawia się nowe okno dialogowe z pustą listą objawów pacjenta i listą wszystkich możliwych objawów. Przycisk <== służy do przenoszenia objawów z jednej listy na drugą. Następnie możliwe jest wykorzystanie jednego z dwóch silników wnioskowania. Wciśnięcie przycisku *Możliwe diagnozy* powoduje wyświetlenie wszystkich możliwych diagnoz dotyczących choroby. Kolejnym przyciskiem jest przycisk *Sprawdź hipotezę*. Po jego naciśnięciu pojawia się okno dialogowe z chorobami zdefiniowanymi w bazie wiedzy. Następnie należy wybrać jedną z chorób jako hipotezę odnośnie diagnozy, a system będzie starał się ją udowodnić. Wybierając z listy dowolną chorobę i wnioskując w tył otrzymamy odpowiedzi: „Hipoteza potwierdzona” w przypadku udowodnienia postawionej diagnozy lub „Hipoteza odrzucona” w przypadku braku możliwości jej udowodnienia.



Rys. 17.5: Okno nowej diagnozy.

Kliknięcie na przycisku *Koniec* zawsze kończy pracę aktualnego okienka, w którym został wywołany.

17.2.5. Opis implementacji

W poniższej części dokumentu przedstawiony został sposób implementacji poszczególnych fragmentów stworzonego programu wraz z przykładami kodu źródłowego.

Język Prolog

Całość projektu utworzona została w środowisku SWI-Prolog. Prolog jest językiem programowania logicznego. Stąd też pochodzi jego nazwa (ang. *Programming in Logic*, ProLog - programowanie w logice). Idea jego działania polega na wnioskowaniu i sprawdzaniu wartości logicznych (testowaniu) kolejnych predykatów zamiast wykonywania kolejnych linii kodu i podprogramów. W wielu przypadkach można przyjąć pewne uproszczenie, porównując testowanie predykatów do wykonywania funkcji. Dla przykładu: predykat `write` powoduje wypisanie informacji w oknie konsoli. Trzeba jednak pamiętać, że wszystkie predykaty (w przeciwieństwie do standardowego algorytmu wykonywanego krok po kroku) nie muszą koniecznie zostać wykonane - w zależności od użytych spójników logicznych oraz zwracanych wartości. Ponieważ stworzono już wiele świetnych kursów języka Prolog, poniższy opis dotyczy jedynie przykładów zaczerpniętych z kodu źródłowego programu wraz z krótkimi opisami ich działania [4].

XPCE

Biblioteki XPCE stanowią obiektową nakładkę na język Prolog. Jest ona standardowo zintegrowana ze SWI-Prologiem, rozszerzając jego możliwości przez dodanie mechanizmów obiektowości. Za pomocą szeregu nowych predykatów, takich jak `new, send` i `get` możliwe jest tworzenie okien dialogowych, elementów graficznych, przycisków, list wyboru i wielu innych elementów graficznego interfejsu użytkownika GUI (ang. *Graphical User Interface*), zapisanych bezpośrednio w plikach Prologa. Biblioteka zawiera ponad 140 wstępnie zdefiniowanych klas i ponad 2000 metod [5].

Przykłady implementacji

Pliki Prologa są w istocie plikami tekstowymi, w których znajdują się zapisane kolejne fakty i reguły. W celu załadowania bazy wiedzy takiego pliku wykorzystuje się polecenie `consult`.

```
?- consult('NazwaPliku').
```

Uruchomienie programu związane jest z uruchomieniem predykatu `main`:

```
main :- consult('objawy'), consult('reguly'), consult('pacjent'),
consult('choroby'), consult('objawy_choroby'),
consult('nie_objawy_choroby'),
diagnostyka.
```

Powyższy predykat spełnia rolę funkcji wczytującej bazę wiedzy z plików Prologa. Kolejno wczytywane są pliki:

- `objawy.pl` - baza wstępnie zdefiniowanych objawów chorób,
- `reguly.pl` - baza reguł wnioskowania systemu wraz z predykatem `diagnoza/2`
- `pacjent.pl` - baza pacjentów wraz z ich objawami,
- `choroby.pl` - baza wstępnie zdefiniowanych chorób,

- objawy_choroby.pl - baza faktów przyporządkowujących objawy występujące w poszczególnych chorobach,
- nie_objawy_choroby.pl - baza objawów, które nie występują przy poszczególnych chorobach.

Ponieważ w przypadku koniunkcji (znak przecinka między predykatami) testowane są wartości logiczne kolejnych predykatów do momentu, aż jeden z nich zwróci wartość false, powyższe instrukcje wykonywane są w kolejności ich wywołania. Po zakończeniu ładowania plików bazy wiedzy następuje uruchomienie predykatu/funkcji diagnostyka.

Aby skompilowany program mógł dodawać i usuwać fakty z bazy wiedzy, na początku pliku następuje zdefiniowanie predykatów jako dynamic w następujący sposób:

```
:-dynamic(zdiagnozowano/2).
:-dynamic(pacjent/2).
:-dynamic(diagnoza/2).
:-dynamic(mozliwa_choroba/2).
:-dynamic(niemozliwa_choroba/2).
...
```

Przykładem predykatu-procedury wykorzystującego mechanizm dynamicznego dodawania faktów jest predykat diagnozuj, odpowiadający za mechanizm wnioskowania w przód:

```
%%%%%% EFEKT PRZYCISKU DIAGNOZUJ %%%%%%%%
diagnozuj(X) :- diagnoza(X,Y),% testowanie reguły diagnoza,
% podstawienie pod Y nazwy choroby
new(@DIA, dialog('Diagnoza dla pacjenta')), %utworzenie okienka
send(@DIA, append(label(choroby, Y))),   %dodanie napisu
send(@DIA,open),    %wyswietlenie okienka
assert(zdiagnozowano(X,Y)),fail. %dodanie nowego faktu
```

Powyższy fragment kodu odpowiada za postawienie diagnozy podczas wnioskowania w przód. Efektem jego działania jest wyświetlenie nowego okna dialogowego z nazwą choroby dla każdej postawionej diagnozy. Predykat new tworzy nowe obiekty, w tym przypadku okno dialogowe, następnie za pomocą predykatu send oraz append dodawane są nowe elementy interfejsu. Operacja send(@DIA,open) powoduje wyświetlenie okna dialogowego o nazwie @DIA. Na końcu do bazy dodawany jest nowy fakt.

Wnioskowanie regresywne realizowane jest przez testowanie predykatu sprawdz_hipoteze, gdzie X to imię pacjenta, a Y to nazwa choroby.

```
sprawdz_hipoteze(X,Y) :- diagnoza(X,Y),
new(@XXX, dialog('Wynik testu hipotezy')),
send(@XXX, append(label(choroby, 'Hipoteza potwierdzona'))),
send(@XXX, open).
```

```
%
sprawdz_hipoteze(X,Y) :- \+ diagnoza(X,Y),
new(@NNN, dialog('Wynik testu hipotezy')),
send(@NNN, append(label(choroby, 'Hipoteza odrzucona'))),
send(@NNN, open).
```

Zdefiniowanie dwóch predykatów o tej samej nazwie zastępuje w Prologu alternatywę logiczną. Za każdym razem testowane są więc oba predykaty. W przypadku gdy `diagnoza(X,Y)` zwraca wartość true, drugi predykat wykonuje się tylko do pierwszej linii, a pierwszy wykonuje się w całości, wyświetlajac stosowne okno dialogowe. W przypadku, gdy `diagnoza(X,Y)` zwróci wartość `false`, wyświetlane jest drugie okienko.

```
mozliwa_choroba(Pacjent,Choroba) :-
ma_objaw(Pacjent,Objaw), choroba_objaw(Choroba,Objaw).
niemozliwa_choroba(Pacjent,Choroba) :-
ma_objaw(Pacjent,Objaw), choroba_nie_objaw(Choroba,Objaw).
diagnoza(Pacjent,Choroba) :-
mozliwa_choroba(Pacjent, Choroba),
\+ niemozliwa_choroba(Pacjent, Choroba).
```

Głowny mechanizm stawiania diagnozy znajduje się w osobnym pliku `reguly.pl` i jest wczytywany do programu podczas jego uruchomienia. Powyższy zbiór prostych reguł odpowiada za stawianie diagnozy na podstawie samych symptomów. Przez edycję pliku możliwe jest rozszerzenie bazy o nowe reguły (np. dotyczące analizy wyników badań) bądź całkowita zmiana zasad wnioskowania.

Głowny kod interfejsu programu znajduje się w pliku `start.pl` Większość okien dialogowych zdefiniowana jest w nim według następującego schematu:

```
%%%%%%LISTA OBJAWOW PACJENTA %%%%%%%
%% POBJ - lista objawow pacjenta,
%% XOBJ - lista wszystkich pozostalych objawow
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lista_objawow_pacjenta_dialog(Pacjent) :-
new(@LOP, dialog('Lista objawow pacjenta')),
new(POBJ, browser),
new(XOBJ, browser),
  %% dodanie zdefiniowanych objawow do list wyboru
objaw_pacjenta_do_listy(Pacjent,POBJ),
objaw_nie_przypisany_do_pacjenta(Pacjent,XOBJ),
send(@LOP, append(label(objawyp,'Objawy pacjenta:
                                        Zdefiniowane objawy:'))),

send(@LOP, append(POBJ)),     %dodanie listy wyboru
```

```
%% Tworzenie przyciskow:
send(@LOP, append(button('<==', message(@prolog,
     dodaj_objaw_pacjenta, Pacjent,
                 XOBJ?selection?key, POBJ, XOBJ)),right)),
send(@LOP, append(button('==>', message(@prolog,
     usun_objaw_pacjenta, Pacjent,
                 POBJ?selection?key, POBJ, XOBJ)),right)),
send(@LOP, append(XOBJ, right)),
    send(@LOP, append(button(koniec, message(@LOP, destroy)))),
send(@LOP,open).
```

Jest to przykład okna dialogowego, umożliwiającego przypisanie objawów pacjenta na podstawie globalnej listy objawów wcześniej zdefiniowanych, reprezentowanych w bazie wiedzy przez fakty `ma_objaw(pacjent,objaw)`. Predykaty `objaw_pacjenta_do_listy` odpowiada za wypełnienie listy wyboru `POBJ` objawami, które występują u pacjenta. Poniżej znajdują się także przyciski <== oraz ==>. Na powyższym przykładzie widać rónież, że silnik Prologa traktowany jest w XPCE jako osobny obiekt `@prolog`. Umożliwia to względnie łatwą obsługę zdarzeń przycisków:

```
send(@LOP, append(button('<==', message(@prolog,
       dodaj_objaw_pacjenta, Pacjent, XOBJ?selection?key,
                                              POBJ, XOBJ)),right)),
```

Działanie powyższej linii kodu jest następujące: Do okna o nazwie `@LOP` dodany zostaje przycisk o etykiecie <==, z którym skojarzona jest komenda wysłana do silnika Prologa. Ten z kolei uruchomia dwuargumentowy predykat `dodaj_objaw_pacjenta`. Pierwszym argumentem testowanego predykatu jest imię pacjenta. Drugi z nich to `XOBJ?selection?key`. Jak nietrudno odgadnąć, operator `?` jest równoznaczny operacji `get` i służy do wyłuskania nazwy zaznaczonego pola listy wyboru wszystkich objawów. `POBJ` i `XOBJ` to dwie listy wyboru, których nazwy też są przekazywane w argumentach predykatu.

Rozwiązania zapewniające spójność bazy wiedzy

Aby uchronić system przed skutkami niespójności bazy wiedzy, w programie zastosowano szereg zabezpieczeń. Pierwszym z nich jest stosowanie list wyboru. Aby przypisać objawy do choroby, muszą one być wcześniej zdefiniowane w bazie wiedzy. Przypisywanie objawów do chorób, widoczne na rys. 17.4, odbywa się na zasadzie „przerzucania" elementów między listami wyboru. Za każdym razem objaw może znajdować się tylko na jednej z list. Nie może więc zajść sytuacja, w której użytkownik popełni błąd i wprowadzi sprzeczność w bazie wiedzy, sabotując tym samym działanie mechanizmu wnioskującego.

```
dodaj_objaw_choroby(Choroba,Objaw,ListaX,ListaAll) :-
    choroba_objaw(Choroba,Objaw);
    assert(choroba_objaw(Choroba,Objaw)),
```

```
send(ListaAll, delete, Objaw),
send(ListaX,append,Objaw).
```

Powyższy fragment kodu to funkcja odpowiadająca za dodawanie objawu choroby. Zmienne `Choroba`, `Objaw`, `ListaX` oraz `ListaAll` pełnią w niej rolę argumentów. Na początku sprawdzana jest wartość logiczna predykatu `choroba_objaw`. Jeśli jest ona prawdziwa, oznacza to, że taki fakt już istnieje w bazie wiedzy. Średnik w trzeciej linii kodu oznacza alternatywę logiczną - w przypadku, gdy fakty znajdujące się przed nim są prawdziwe, wartości dalszych predykatów nie są już testowane i wykonywane, dzięki czemu system nie pozwala na dodawanie faktów już istniejących. W przeciwnym wypadku nowy fakt należy dodać do bazy wiedzy za pomocą polecenia `assert`. Ostatnią czynnością jest przerzucenie odpowiedniego elementu między listami `ListaX` oraz `ListaAll` w oknie dialogowym, aby użytkownik natychmiast zauważył efekt swojego działania.

17.2.6. Podsumowanie

Ze względu na skomplikowany proces diagnozowania oraz konieczność posiadania bardzo dużej wiedzy eksperckiej z dziedziny medycyny, realizowany projekt został ograniczony do systemu przykładowego, którego głownym zadaniem była demonstracja mechanizmów wnioskowania języka Prolog. Realizacja praktyczna projektu ukazała typowe problemy, z jakimi spotkać się można podczas budowy systemu ekspertowego. Pierwszym z nich jest rozmyty charakter wiedzy eksperckiej i oddanie toku myślenia eksperta, który niejednokrotnie sam nie jest całkowicie świadom reguł swojego postępowania. Kluczowy jest tu wybór odpowiedniego narzędzia w zależności od problemu - systemy regułowe dokładne, oprate na wnioskowaniu logicznym, nie zawsze okazują się najlepszym wyborem. Programowanie w Prologu jest jednak fascynującym doświadczeniem, gdyż wymaga wypracowania zupełnie innego sposobu myślenia i zastosowania odmiennych schematów postępowania, niż w przypadku tradycyjnych języków programowania. Podobnie istotny jest właściwy wybór dystrybucji Prologa oraz sposobu realizacji interfejsu. Możliwości XPCE odbiegają tu co prawda od powszechnego wyobrażenia o nowoczesnych, wizualnych edytorach GUI, lecz żadna z pozostałych dystrybucji Prologa również nie imponuje pod tym względem. Z drugiej strony, połączenie Prologa i XPCE ma bardzo wiele zalet. Po pierwsze, jest to rozwiązanie darmowe i całkowicie zintegrowane, a SWI-Prolog jest obecnie jedną z najbardziej popularnych, nowoczesnych dystrybucji tego języka. Po drugie - po bliższym poznaniu XPCE okazuje się być zupełnie wystarczającym narzędziem do budowy kompletnych interfejsów systemów ekspertowych. Stworzone dzięki niemu oprogramowanie jest też niezwykle przenośne - jak zostało wielokrotnie sprawdzone, kod programu kompiluje się, działa i wygląda dokładnie tak samo, niezależnie od systemu operacyjnego. Jako podsumowanie projektu można więc śmiało polecić korzystanie ze wspomnianych narzędzi zarówno podczas zajęć edukacyjnych, jak i w trakcie realizacji podobnych, nawet komercyjnych projektów.

Literatura

[1] A. Niederliński: *Regułowo-modelowe systemy ekspertowe rmse* Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, (2006).
[2] J. Mulawka: *Systemy ekspertowe* Wydawnictwo Naukowo Techniczne, Warszawa, (1997).
[3] U. Sowa: *Podstawowe systemy wnioskowania sztucznej inteliencji*, (2006).
[4] OMG Laboratorium z języka Prolog. `https://ai.ia.agh.edu.pl/wiki/pl:prolog:prolog_lab`, (2002).
[5] OMG Programming in XPCE/Prolog. `http://www.swi-prolog.org/download/xpce/doc/userguide/userguide.pdf`, (2002).

Od redaktora i wydawcy

Czym jest wiedza?

Nad tym pytaniem zastanawiali się już starożytni, głowili się filozofowie, łamali głowę psycholodzy, próbowali na nie odpowiadać praktycy. Choć wiedza powszechnie kojarzona jest z informacją, nauką, doświadczeniem, zbiorem faktów, nie posiada ona jednej, uniwersalnej definicji. W zależności od kontekstu odpowiedź na to pytanie może przybrać różną postać i formę.

Czym jest przetwarzanie wiedzy?

To kolejna niewiadoma, sięgająca w swej materii do sposobów reprezentacji wiedzy, jej interpretacji i wykorzystania, włączając w to metody wnioskowania i podejmowanie decyzji. Podobnie jak w pytaniu o wiedzę, mnogość możliwych odpowiedzi może tu być ogromna.

Czym jest komputerowe przetwarzanie wiedzy?

Odpowiedź na to pytanie jest projekcją sumy odpowiedzi na powyższe dwa pytania na płaszczyznę zdefiniowaną przymiotnikiem „komputerowe". Mówiąc prościej jest to dziedzina, w której wykorzystuje się komputery do rozwiązywania złożonych problemów zdefiniowanych na różnych poziomach abstrakcji. Wykracza ona poza samą implementację algorytmów ekstrahujących wartości parametrów opisujących otaczający nas świat. Istnieje na pograniczu sztucznej inteligencji i inteligencji istot żywych, tworząc pomost pomiędzy czymś, co jesteśmy w stanie sami przeanalizować, a czymś, co umyka naszym zmysłom z powodu wielkiej złożoności albo szczegółowości.

W niniejszej książce zebrano opracowania wykonane przez studentów V roku Automatyki i robotyki w ramach kursu Komputerowe przetwarzanie wiedzy prowadzonego przeze mnie na Politechnice Wrocławskiej w semestrze zimowym 2010/2011. Zgodnie z zarysowanym kształtem odpowiedzi na ostatnie z powyższych pytań, zadanie studentów polegało nie tyle na zaimplementowaniu jakiegoś opublikowanego bądź autorskiego algorytmu, co na posłużeniu się zdobywaną dzięki temu wiedzą do rozwiązania jakiegoś problemu na wyższym poziomie abstrakcji.

Zakres tematyczny opracowań można zawrzeć w następującym zestawieniu:

- Przetwarzania dokumentów
- Percepcja i ekspresja
- Losowość jej zastosowanie
- Bazy danych
- Systemy ekspertowe
- Portale społecznościowe
- Systemy agentowe
- Analiza ryzyka
- Kontrola jakości

Mam nadzieję, że lektura tych opracowań okażą się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, wrzesień 2011

ISBN 978-83-930823-2-2
9 788393 082322