KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2015/2016
pod redakcją Tomasza Kubika

KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2015/2016
pod redakcją Tomasza Kubika

Skład komputerowy, projekt okładki
Tomasz Kubik

ISBN 978-83-930823-8-4

Wydawca
Tomasz Kubik

Druk i oprawa
I-BiS sc., ul. Sztabowa 32, 50-984 Wrocław

Skład niniejszej książki wykonano w systemie LaTeX

SPIS TREŚCI

SŁOWO WSTĘPNE

Z dużą przyjemnością przedstawiam Czytelnikom kolejny zbiór prac z dziedziny komputerowego przetwarzania wiedzy zredagowany przez dra inż. Tomasza Kubika. Autorami prac zebranych w książce są studenci Wydziału Elektroniki Politechniki Wrocławskiej, studiujący w semestrze letnim roku akademickiego 2015/2016 na studiach II stopnia, na specjalności Robotyka kierunku Automatyka i robotyka. Nowa książka obejmuje 91 stron druku i składa się z ośmiu rozdziałów, których celem jest przedstawienie wybranych metod i narzędzi do przetwarzania wiedzy oraz ich zastosowań do rozwiązania konkretnych zadań praktycznych. Autorzy i tytuły rozdziałów książki są następujące:

1. Dominika Perz: *Tworzenie i zastosowanie chatbotów*
2. Małgorzata Bulanda, Michał Majstrowicz: *Analiza wydźwięku emocjonalnego tekstu*
3. Bogna Czyżewska, Tomasz Rusiński: *Komunikacja niewerbalna w implementacji cyfrowej*
4. Paweł Joniak, Bartosz Witkowski: *Automatyczna ocena ładunku emocjonalnego tekstu w systemie typu helpdesk*
5. Hanna Preiss, Piotr Zieliński: *Zarządzanie słownikiem pojęć biznesowych*
6. Damian Barański, Arkadiusz Mielczarek: *Modelowanie zachowania się stada*
7. Tomasz Baciak, Anna Sobocka: *Asystent parkowania*
8. Tomasz Wodziczko, Maciej Kuklewski: *Reguły asocjacyjne w badaniu zależności między typem gwiazdy a występowaniem towarzyszących jej egzoplanet*

Rozdziały 1, 2, 3 i 4 są zbliżone tematycznie i dotyczą szeroko rozumianej komunikacji między człowiekiem a maszyną. Szczególną uwagę poświecono w nich nadaniu komunikacji wymiaru społecznego przez wykorzystanie sygnałów społecznych, takich jak rozpoznawanie i wyrażanie emocji, a także gestykulacja. Tematyka pozostałych rozdziałów jest bardziej zróżnicowana. Korzystając tradycyjnie z metody *pars pro toto* omówimy bardziej szczegółowo zawartość rozdziałów nr 3, 6 i 7.

- *Komunikacja niewerbalna w implementacji cyfrowej*: W rozdziale przedstawiono zasady komunikacji niewerbalnej, ze szczególnym uwzględnieniem

przekazu treści emocjonalnych. Omówiono dwa wybrane modele obliczeniowe emocji. Zaproponowano własne programy komputerowe do komunikacji emocjonalnej z komputerem i z robotem (NAO). Programy te zostały zaimplementowane i poddane badaniom, a ich wyniki przedstawiono w tekście.

- *Modelowanie zachowania się stada*: Rozdział dotyczy modelowania zachowania stada zwierząt, z potencjalnym odniesieniem do zachowania grupy robotów mobilnych. Model zachowania został oparty na zestawie 6 prostych reguł zachowań stadnych podporządkowanych percepcji otoczenia. Przyjęto następujące reguły: wyrównania, spójności, rozdzielności, ucieczki, głodu i unikania przeszkód. Opracowano oprogramowanie do symulacji zachowań stada i dokonano jego implementacji.
- *Asystent parkowania*: Rozdział został poświęcony zasadom działania i konstrukcji urządzenia wspomagającego kierowcę przy manewrze parkowania samochodu. Skonstruowane urządzenie dokonuje fuzji danych z kilku czujników (sonar ultradźwiękowy, kamera wizyjna, enkoder), przedstawia tor ruchu samochodu na ekranie komputera pokładowego i ostrzega przed możliwa kolizją z przeszkodami występującymi w najbliższym otoczeniu. Urządzenie poddano testom, których wyniki opisano w tekście.

Niniejszy zbiór prac jest adresowany do Czytelników zainteresowanych praktycznymi aspektami komputerowego przetwarzania wiedzy.

Prof. Krzysztof Tchoń,
opiekun specjalności Robotyka,
Wrocław, wrzesień 2016

ROZDZIAŁ 1

TWORZENIE I ZASTOSOWANIE CHATBOTÓW

D. Perz

Zakres tematyczny niniejszego rozdziału obejmuje zagadnienia związane z implementacją oraz analizą działania wirtualnych asystentów, zwanych potocznie chatbotami. W jego ramach zamieszczono krótką definicję chatbotów, opisano ich praktyczne zastosowania oraz wymieniono podstawowe metody ich budowy. Przedstawiono platformę Playground oraz przeanalizowano jej możliwości wykorzystując w tym celu autorską implementację chatbota. Rozdział kończy podsumowanie wraz z wnioskami z wykonanych prac.

1.1. Wstęp

Chatbot – jest to program komputerowy, nazywany również botem, chatterbotem albo wirtualnym asystentem, służący do prowadzenia inteligentnej konwersacji z człowiekiem najczęściej za pomocą interfejsu tekstowego. Celem badań naukowych nad chatbotami jest osiągniecie takiego poziomu konwersacji człowiek–maszyna, na którym ludzki rozmówca nie będzie w stanie ocenić, czy komunikuje się z prawdziwym człowiekiem czy jedynie ze sztuczną inteligencją.

Chatboty doskonale sprawdziły się w obszarze rozrywki i komercji. Wirtualni asystenci są wykorzystywani jako przewodnicy wędrujących po sieci internautów. Po nawiązaniu rozmowy potrafią rozpoznać intencje internautów, a przeszukując swoje bazy wiedzy, są również w stanie odpowiadać na podstawowe pytania dotyczące działalności i usług danej firmy. Znaczącym obszarem zastosowań chatbotów stała się reklama oraz systemy rekomendacyjne (które, niestety, często są źródłem niechcianego spamu). Choć wykorzystywane tam programy są w większości mało inteligentne, to jednak dzięki udawaniu człowieka zyskują zaufanie internautów i nierzadko skutecznie namawiają ich do odwiedzenia danej strony internetowej czy kupna konkretnego towaru.

1.2. Podstawy budowy chatbotów

Głównym celem każdego chatbota jest prowadzenie inteligentnej rozmowy bez zdradzania maszynowego pochodzenia. Cel ten osiągany jest poprzez naśla-

dowanie ludzkich odpowiedzi oraz reakcji w czasie rozmowy. Aby umieć zinterpretować informację podaną przez użytkownika, a następnie wygenerować adekwatną odpowiedź, stosuje się różne podejścia. Kilka z nich omówiono poniżej.

Najprostszym i najbardziej popularnym podejściem stosowanym przy tworzeniu chatbotów jest wykorzystanie bazy wiedzy składającej się z tzw. *kategorii* – zestawów wzorcowych par pytanie-odpowiedź. Pary te określają sposób, w jaki bot powinien zareagować na zadane pytanie. Wygenerowanie właściwej odpowiedzi może okazać dość trudne. Wymaga bowiem dogłębnego przeszukania bazy wiedzy, znalezienie najtrafniejszej kategorii, a następnie wygenerowanie na jej podstawie konkretnej odpowiedź. Stąd poziom złożoności oraz trafności reakcji danego asystenta zależy od obszerności posiadanej przez niego bazy wiedzy oraz zaimplementowanych algorytmów wyszukiwania.

W większości przypadków chatboty mówią w języku angielskim. Jest to spowodowane stosunkowo prostą gramatyką tego języka, którą da się obsłużyć za pomocą nieskomplikowanych algorytmów.

Bazy wiedzy chatbotów mogą być zadeklarowane statycznie. Jednak dużo ciekawszym rozwiązaniem jest zastosowanie baz dynamicznych, które mogą z czasem ewoluować. Jednym ze sposobów na wdrożenie tej idei jest „zatrudnienie" osoby nadzorującej, która analizując prowadzone przez bota rozmowy poszerza bazę danych o brakujące kategorie. Podejście to może być zrealizowane poprzez zastosowanie specjalnego, dedykowanego języka.

1.2.1. Język AIML

AIML (ang. *Artificial Intelligence Markup Language*) – jest to XML-owy język znaczników, służący do tworzenia baz wiedzy dla chatbotów [5]. Wyróżnia się w nim cztery główne elementy strukturalne:

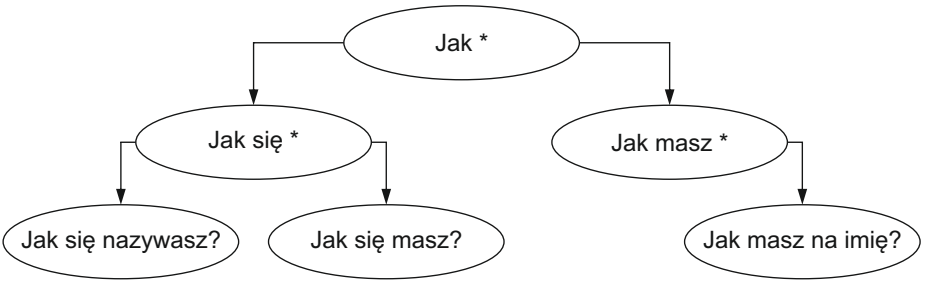
- `<aiml>` – znacznik odpowiadający za zapoczątkowanie oraz zakończenie dokumentu zawierającego kod w AIML;
- `<category>` – znacznik wyróżniający kategorię zawierającą parę wzorzec-szablon;
- `<pattern>` – znacznik nazywany wzorcem, wskazujący tekst traktowany jako pytanie;
- `<template>` – znacznik nazywany szablonem, oddzielający fragment będący odpowiedzią.

Przykład definicji kategorii

Poniżej przedstawiono przykładową definicję kategorii wyrażoną w języku AIML. Kategoria ta określa wzorzec, dzięki któremu bot zapytany o samopoczucie wygeneruje zdefiniowaną odpowiedź.

```
<category>
  <pattern>JAK SIĘ CZUJESZ</pattern>
  <template>Czuję się bardzo dobrze, a ty?</template>
</category>
```

Aby zwiększyć przejrzystość przykładu pominięto znaczniki początkowe i końcowe plików AIML: `<aiml>` oraz `</aiml>`. Zwyczajowo przyjęto, że tekst wzorca pisany jest wielkimi literami. Przy próbach programowego dopasowania zapytania do wzorca wielkości liter i znaki interpunkcyjne często są pomijane. Przykładową analizę pytania przedstawiono na rysunku 1.1.



Rys. 1.1: Schemat przeszukiwania reguł w AIML [5]

Głównym ograniczeniem zaprezentowanego podejścia jest konieczność dokładnego dopasowania pytania użytkownika do zadeklarowanego wzorca. Oznacza to, że pytanie: „`Jak się dzisiaj czujesz?`" nie zostanie dopasowane do zdefiniowanej kategorii, mimo, że semantycznie jest bardzo podobne. Istnieje jednak wiele innych znaczników, dzięki którym bazę wiedzy można rozbudować do postaci pozwalającej na prowadzenie „inteligentnych" rozmów.

Możliwości rozbudowania bazy wiedzy w AIML

W języku AIML istnieje mechanizm przekazywania informacji za pomocą zmiennych pozwalający na zwiększenie adekwatności odpowiedzi chatbota, Dzięki niemu możliwe jest zapamiętanie imienia użytkownika albo tematu rozmowy i późniejsze jego wykorzystanie. Możliwe jest również dodanie kilku szablonów do jednego wzorca. Wybór konkretnej odpowiedzi może być losowy bądź uzależniony od wartości zmiennej. Szablon może odwołać się również do innej, wcześniej zaimplementowanej kategorii, a następnie użyć zdefiniowanych tam odpowiedzi. Prowadzi to do znacznej redukcji objętości kodu.

1.2.2. Systemy oparte na wiedzy

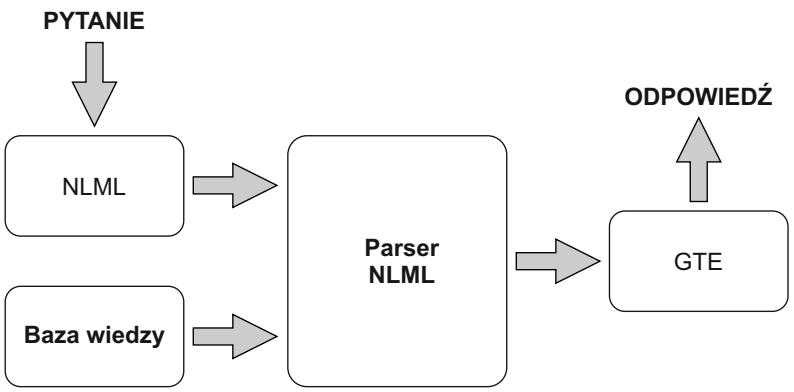
W systemach wykorzystujących język AIML przeszukiwane są listy wzorców, bez zwracania uwagi na gramatykę czy też semantykę zdań wprowadzonych przez użytkownika. Inaczej jest w przypadku systemów opartych na wiedzy. Systemy te, w zamyśle twórców, miały pełnić rolę nauczycieli konwersacji w języku angielskim dla osób poznających ten język [3].

W implementacjach systemów opartych na wiedzy mocno eksploatuje się metody przetwarzania języka naturalnego. Schemat ich działania zakłada wykonanie kilku etapów przetwarzania (zobacz rysunek 1.2). Na początku zdania wprowadzone przez użytkownika poddawane są analizie za pomocą analizatora skła-

dniowego (wspierającego przetwarzanie języka angielskiego). Uzyskane wyniki są zapisywane w języku NLML (ang. *Natural Language Markup Language* – językiem znaczników powstałym na bazie języka XML, pozwalającym tworzyć graf określający funkcje gramatyczne poszczególnych słów oraz ich zależności znaczeniowe), po czym następuje ich przekazanie do dalszego przetwarzania.

W kolejnym etapie uruchamiany jest parser NLML, który łączy świeżo utworzony graf z bazą wiedzy na temat ontologii zdaniowej oraz wyrażeniowej. Na tym etapie otrzymywane zdanie rozbijane jest na pojedynczo zapamiętywane fakty. Wszystkie zdobyte fakty przechowywane są w bazie danych NLDB (ang. *Natural Language Database*). NLDB zawiera również fakty historyczne (zdobyte wcześniej), wiedzę ogólną oraz wiedzę chatbota na swój własny temat.

W celu zwiększenia elastyczności używanego języka stosowany jest następnie mechanizm GTE (ang. *Generation of Textual Entailment*). Jest on odpowiedzialny za stwierdzanie równoważności takich wyrażeń jak: „`Jestem Bot.`”, „`Mam na imię Bot.`” oraz „`Moje imię to Bot.`”. Ostatnim etapem przetwarzania jest skonstruowanie odpowiedzi na podstawie całej wiedzy i faktów dostępnych chatbotowi.



Rys. 1.2: Schemat działania systemu opartego na wiedzy [3]

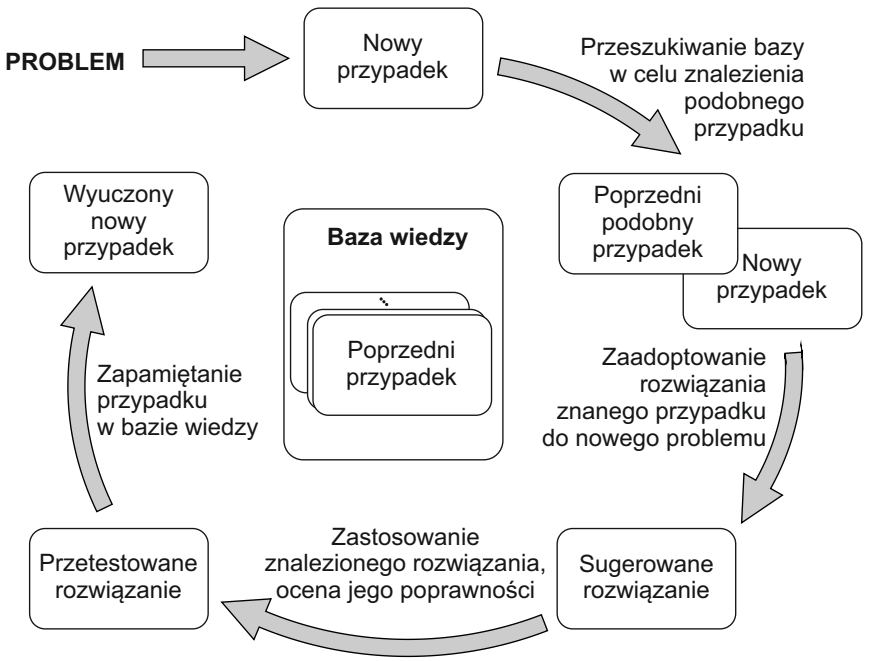
1.2.3. Wnioskowanie na podstawie przypadków (CBR)

CBR (ang. *case-based reasoning*) – termin ten jest używany do opisywania przypadków wnioskowania, w których oprócz bazy wiedzy analizowane są również wszystkie poprzednie doświadczenia („przypadek' występujący w rozwinięciu skrótu „wnioskowanie na podstawie przypadków"' oznacza problematyczną sytuację, w której znalazł się agent). W odniesieniu do chatbotów mówi się o wdrożeniu scenariusza, w którym bot przeszukuje bazę doświadczeń pod kątem przypadku najbardziej zbliżonego do zaistniałej sytuacji, a następnie, dysponując informacją o tym jakie było działanie i skutek poprzednim razem, podejmuje decyzję odnośnie nowej sytuacji. Następnie nowa sytuacja zostaje dołączona do bazy doświadczeń bota [1].

Podejście CBR wzorowane jest na sposobie, w jaki ludzie, szczególnie specjaliści, rozwiązują nowo napotkane problemy. Przykładowo, jeżeli lekarz zauważy podobieństwo choroby nowego pacjenta do choroby osoby, którą leczył wcześniej, to zestawiając zaordynowaną wcześniej terapię z zaobserwowanymi jej skutkami może zdecydować się na ponowne zastosowanie tej terapii.

Sposób działania systemu bazującego na CBR można zamknąć w czterech etapach (rys. 1.3):

1. Wyszukanie najbardziej pasującego przypadku lub zbioru przypadków do zaistniałej sytuacji.
2. Wykorzystanie znalezionej wiedzy do rozwiązania nowego problemu.
3. Ocenienie trafności i skuteczności użytego rozwiązania.
4. Zapamiętanie doświadczenia do późniejszego wykorzystania.



Rys. 1.3: Etapy w systemach opartych na CBR [1]

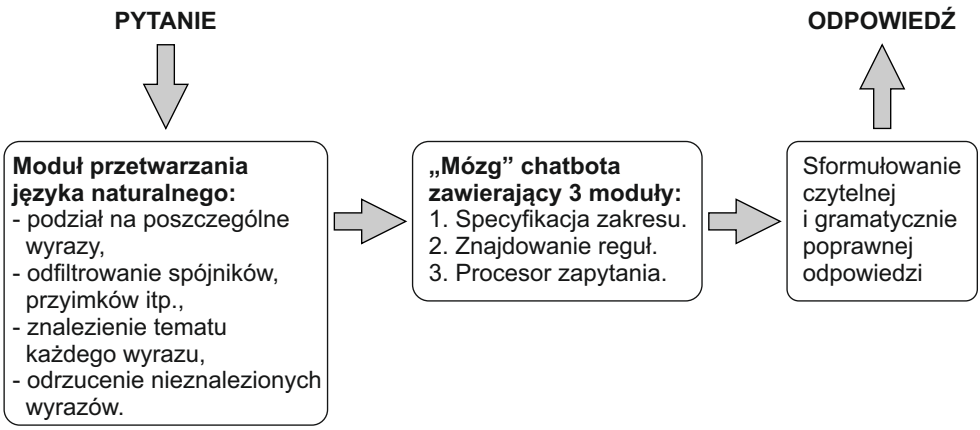
W przedstawione podejściu posiada kilka aspektów budzących zainteresowanie naukowców. Najistotniejszym z nich jest umiejętne uczenie się na podstawie doświadczeń. Metody uczenia się w CBR stanowią osobny dział badań w ramach uczenia maszynowego. Kolejnym problemem jest klasyfikacja oraz sposób przeszukiwania bazy przypadków. W niektórych rozwiązaniach wiąże się podobne przypadki w większe grupy w celu lepszego ich uporządkowania i przechowywania. Również sam sposób zapisu zdobytych doświadczeń, z jednej strony jego zwięzłość, a z drugiej zawarcie wszystkich informacji, jest problematyczny.

1.2.4. Ontologia

Ontologia – w ujęciu informatycznym jest to forma reprezentacji wiedzy za pomocą zestawów pojęć oraz relacji między nimi. Ontologia dostarcza terminologii (do budowy abstrakcyjnego modelu) oraz asercji (rzeczywistych faktów). Na takiej bazie można uruchamiać algorytmy wnioskowania, pozwalające pozyskać dodatkowych informacji z wiedzy już posiadanej [2].

Jeśli wiedza robota zostanie zapisana w postaci jakiejś ontologii, to zamiast zestawów par pytanie-odpowiedź przetwarzany będzie graf składający się z pojęć (konceptów) oraz wiążących je związków. Dzięki elastyczności użytego modelu korzystanie z bazy wiedzy można zautomatyzować oraz wyposażyć w algorytmy identyfikujące nieznane wcześniej fakty bez udziału człowieka. Nowe scenariusze rozmowy na dany temat mogą być wygenerowane na bieżąco. Kolejną zaletą jest brak potrzeby nauki dodatkowego języka typu AIML.

Schemat postępowania w systemach opierających się na ontologii przedstawiono na rysunku 1.4. Po otrzymaniu pytania wejściowego od użytkownika uruchamiany jest moduł przetwarzania języka naturalnego. Jego zadaniem jest przetworzenie otrzymanego pytania do prostszej postaci. Najpierw pytanie dzielone jest na poszczególne wyrazy, następnie odfiltrowywane są słowa nie wnoszące znaczenia jak spójniki, przyimki itp. Kolejnym posunięciem jest przywrócenie wszystkich słów do tematu bazowego, czyli części która zasadniczo nie podlega odmianie. Dodatkowo, dla wyrazów, które nie zostają rozpoznane, poszukiwane są wyrazy bliskoznaczne występujące w bazie wiedzy bota. Jeżeli któryś z otrzymanych wyrazów po raz kolejny nie zostanie znaleziony bazie, uznawany jest jako niepoprawny i jest odrzucany.



Rys. 1.4: Schemat postępowania w systemach opierających się na ontologii [2]

Po zakończeniu wstępnej odróbki zapytania użytkownika uruchamiany jest główny element architektury bota, który zwraca słowa kluczowe dla przyszłej odpowiedzi. Element ten stanowi główny „mózg" chatbota i dzieli się na trzy współpracujące moduły:

1. Moduł do specyfikacji zakresu – jego głównym zadaniem jest zrozumienie o czym mówi użytkownik oraz w zakresie jakiego tematu zadaje pytanie: czy jest ono losowe czy dotyczy konkretnego zagadnienia. Informacja wygenerowana przez ten moduł przechodzi do następnego etapu.
2. Moduł do znajdowania reguł – w tym module na podstawie obrobionego pytania oraz określonego zakresu rozmowy przeszukiwane są reguły, które generują nowy format zadanego pytania: taki, który jest zrozumiały dla kolejnego modułu.
3. Procesor zapytania – nowo wygenerowanie zapytanie zostaje porównane z bazą wiedzy bota, a znalezione słowa-klucze przesyłane są dalej.

Ostatnim etapem przetwarzania zapytania jest sformułowanie czytelnej i sensownej odpowiedzi na podstawie wygenerowanych słów-kluczy. Tutaj również sprawdzane jest czy formy rzeczowników oraz odmiany czasowników są zgodne z gramatyką języka, a także z sensem odpowiedzi.

1.2.5. Naiwny Klasyfikator Bayesowski (NBC)

Innym sposobem na przeszukiwanie bazy wiedzy bota oraz znajdowanie najbardziej adekwatnych odpowiedzi jest użycie Naiwnego Klasyfikatora Bayesowskiego [4]. Podejście to zakłada, że baza wiedzy podzielona została na kategorie, które określane są za pomocną słów kluczowych. Wewnątrz kategorii znajduje się zastaw pytań i odpowiedzi dotyczący danej tematyki.

W pierwszym etapie tego podejścia zadane przez użytkownika pytanie dzielone jest na pojedyncze słowa. Następnie dla każdej kategorii j obliczane jest prawdopodobieństwo $P_{NB}(C_j)$ zgodnie ze wzorem

$$P_{NB}(C_j) = P(C_j) * P(Matched|C_j), \tag{1.1}$$

gdzie:

- $P(Matched|C_j)$ jest ilorazem liczby słów z pytania, które pokryły się ze słowami kluczowymi kategorii j przez liczbę wszystkich słów kluczowych w kategorii j,
- $P(C_j)$ jest ilorazem liczby słów w danej kategorii przez całkowitą liczbę słów we wszystkich kategoriach.

Kategoria, która posiada największe prawdopodobieństwo zostaje wybrana do dalszego przeszukiwania.

Następny etap to odnalezienie odpowiedzi w zakresie wybranej kategorii. Tutaj również obliczane jest prawdopodobieństwo dla każdego pytania z bazy. Prawdopodobieństwo liczone jest w sposób analogiczny do sposobu zastosowanego przy wyborze kategorii:

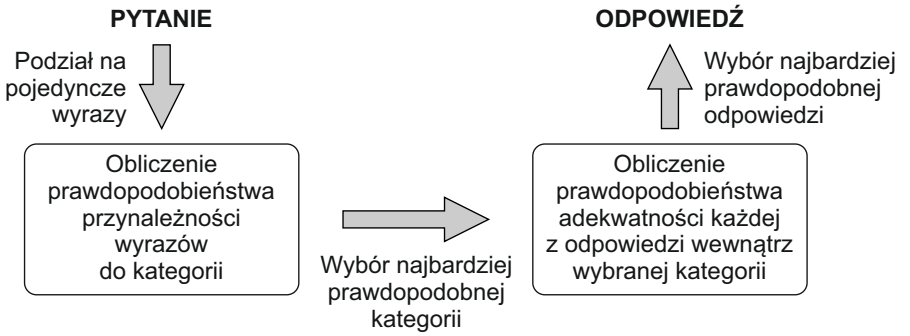
$$P_{NB}(Q_i) = P(Q_i) * P(Matched|Q_i), \tag{1.2}$$

gdzie:

- $P(Matched|Q_i)$ jest ilorazem liczby słów z zadanego pytania, które pokryły się ze słowami z pytania i znajdującego się w bazie przez liczbę wszystkich słów ze wszystkich pytań w danej kategorii,

- $P(C_j)$ jest ilorazem liczby pytań w danej kategorii przez całkowitą liczbę pytań we wszystkich kategoriach.

Odpowiedź na pytanie, którego prawdopodobieństwo jest najwyższe, zostaje użyte jako reakcja na wiadomość od użytkownika. Wszystkie etapy przedstawiono w postaci schematu na rysunku 1.5.



Rys. 1.5: Schemat działania NCB [4]

1.3. Budowa chatbota za pomocą platformy Pandorabot

The Pandorabots Playground to darmowe środowisko programistyczne zintegrowane ze stroną internetową (`https://playground.pandorabots.com`), służące do tworzenia oraz testowania własnoręcznie zaimplementowanych chatbotów, a także do prowadzenia rozmów z chatbotami stworzonymi przez innych użytkowników.

Playground jest częścią usługi internetowej o nazwie Pandorabots, która oprócz funkcjonalności zdefiniowanych dla komponentu Playground, oferuje również tworzenie spersonalizowanych chatbotów na podstawie specyfikacji klienta, prowadzenie warsztatów edukacyjnych dla pracowników firm, profesjonalną pomoc w debuggowaniu kodu oraz przetrzymywaniu danych, a także hostowanie licencjonowanych asystentów.

1.3.1. Tworzenie profilu oraz budowa nowego chatbota

Aby rozpocząć tworzenie własnego wirtualnego asystenta należy wykonać kilka następujących kroków.

Krok 1: Logowanie
Playground jest rozwiązaniem darmowym dla wszystkich użytkowników chcących zbudować bota na potrzeby osobiste i niekomercyjne. Aby się do niego zalogować należy mieć konto użytkownika założone na jednym z serwisów: Facebook, Google+, Twitter, Yahoo!. Logowanie polega bowiem na użyciu poświadczeń generowanych przez te serwisy.

Krok 2: Stworzenie chatbota
Po zalogowaniu się do profilu pojawia się lista wszystkich dostępnych chatbotów stworzonych na danym profilu. W celu stworzenia nowego asystenta należy nacisnąć przycisk „Create Bot". Platforma umożliwia wybór języka oraz nazwy asystenta, po czym ładuje edytor dla nowo stworzonego bota.

W edytorze znajdują się różne zakładki. Pierwsza zakładka oznaczona nazwą bota zawiera podstawowe informację na jego temat. Druga, o nazwie „Train", pozwala na testowanie zaimplementowanych rozwiązań w rozmowie z asystentem. Trzecia zakładka to „Files" – pozwala na przeglądanie i edytowanie plików AIML. Ostatnia z zakładek, nazwana „Logs", zawiera zapisy ze wszystkim przeprowadzonych konwersacji.

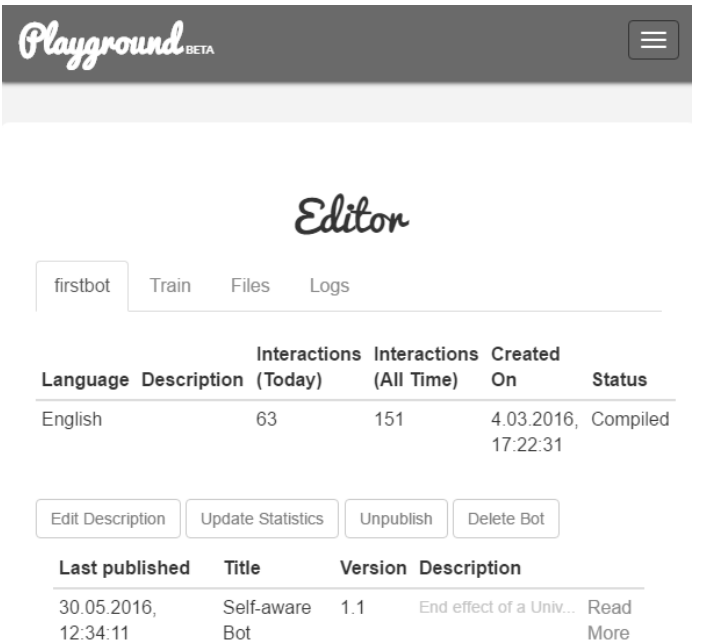
Krok 3: Tworzenie nowych oraz ładowanie istniejących plików AIML
Przechodząc do zakładki „Files" po lewej stronie pojawia się lista wszystkich plików AIML oraz trzy ikony służące kolejno do stworzenia nowego pliku, załadowania gotowych plików z dysku komputera oraz ściągnięcia plików znajdujących się na serwerze na dysk lokalny.

W sieci dostępnych jest wiele open-source'owych plików bazowy dla chatbotów, która zapewniają podstawowe możliwości konwersacyjne. Jednym z nich jest polecany przez serwer Pandorabots bot o nazwie *Rosie*. Należy pobrać jego pliki z ogólnodostępnego repozytorium, a następnie załadować do serwera. Pliki te pojawią się w zakładce „Files", gdzie można je przeglądać oraz edytować w zależności od potrzeb. Tworząc nowe pliki należy pamiętać o umieszczeniu w nich znaczników początku i końca pliku `<aiml></aiml>`.

Krok 4: Testowanie
Aby zapewnić prawidłowe funkcjonowanie programu należy przetestować użyte rozwiązania. W platformie Playground służy do tego zakładka „Train". Umożliwia ona prowadzenie rozmów ze stworzonym asystentem. W momencie wygenerowania odpowiedzi udostępniany jest również sposób jej otrzymania: wypis wszystkich kategorii, które zastały wykorzystane. Dodatkowo wyświetlany jest temat rozmowy, a także istnieje możliwość wyczyszczenia pamięci bota w celu rozpoczęcia nowej konwersacji.

Wszystkie rozmowy, które zostały przeprowadzone przez chatbota, zarówno z jego twórcą, jak i innymi użytkownikami platformy, zapisane są w zakładce „Log". Umożliwia to późniejszą analizę adekwatności odpowiedzi chatbota, która może prowadzić do wykrycia błędów oraz poprawy implementacji.

Krok 5: Publikowanie gotowego bota
Po zakończeniu procesu tworzenia i doskonalenia chatbota warto zdecydować się na udostępnienie go pozostałym użytkownikom platformy Playground. Pozwoli to zwiększyć liczbę zapisanych konwersacji, a poprzez ich analizę, może doprowadzi do ulepszania bota. Krok ten umożliwi innym użytkownikom pobranie wykonanej implementacji, jej doskonalenia i rozszerzanie. Aby upublicznić stworzonego asystenta należy przejść do pierwszej zakładki z imieniem bota oraz użyć przycisku „Publish Bot". Od tego momentu każda osoba posiadająca konto na platformie Playground będzie mogła prowadzić z nim rozmowy.



Rys. 1.6: Przykładowy widok platformy Playground Pandorabots

1.4. Chatbot *Self-aware Bot*

W ramach poszerzania wiedzy o chatbotach na platformie Playground Pandorabots (w wersji beta) stworzono nowego chatbota o nazwie *Self-aware Bot*. Miał on dysponować wiedzą dotyczącą zagadnień związanych z budową botów (przedstawioną na początku niniejszego rozdziału) jak i wiedzą o sobie. Ogólne umiejętności prowadzenia rozmowy zostały zapożyczone z open-source'owego asystenta *Rosie*, którego pliki można pobrać z repozytorium Pandorabots `https://github.com/pandorabots/rosie`. Wszystkie zmiany wprowadzono w formie jednego pliku AIML w wersji 2.0 o nazwie `chatbot.aiml`. Plik ten udostępniono publicznie na wspomnianej platformie.

1.4.1. Plik `chatbot.aiml`

Implementacja całej zebranej wiedzy zebrana została w pliku `chatbot.aiml`. Kod można podzielić na dwie główne części. Na początku zawarto wszystkie definicje oraz wyjaśnienia podstawowych zagadnień związanych z opisywanym tematem. Jako wzorce w tej części przyjęto jedynie hasła odnoszące się do wybranych pojęć, a nie pełne pytania. Drugi fragment kodu zawiera bazę pytań, które mogą być zadane przez użytkownika. Pytania te odwołują się do wypisanych wcześniej haseł. Podział ten miał na celu ułatwienie zadania implementacji: najpierw skupiono się na zwarciu zabranej wiedzy, a później starano się jak najbardziej rozszerzyć bazę pytań.

Przykładowa kategoria oparta na haśle i zawierająca definicję wygląda następująco:

```
<category>
        <pattern>CATEGORY DEFINITION</pattern>
        <template> Category is a pair of pattern (question asked
        by a user) and template (the bot's answer to that question).
        It is a basic unit in AIML.</template>
</category>
```

Wszystkie kategorie zostały napisane w języku angielskim, jako, że w tym języku utworzona jest cała implementacji bota *Rosie*. Zgodnie z wcześniejszym opisem kodu, wzorzec nie stanowi pytania, które mogłoby być zadane przez użytkownika, a jedynie określa zagadnienie, którego dotyczy dana wypowiedź. W drugiej części znajduje się kilka kategorii, które odwołują się do przytoczonego wzorca `CATEGORY DEFINITION`:

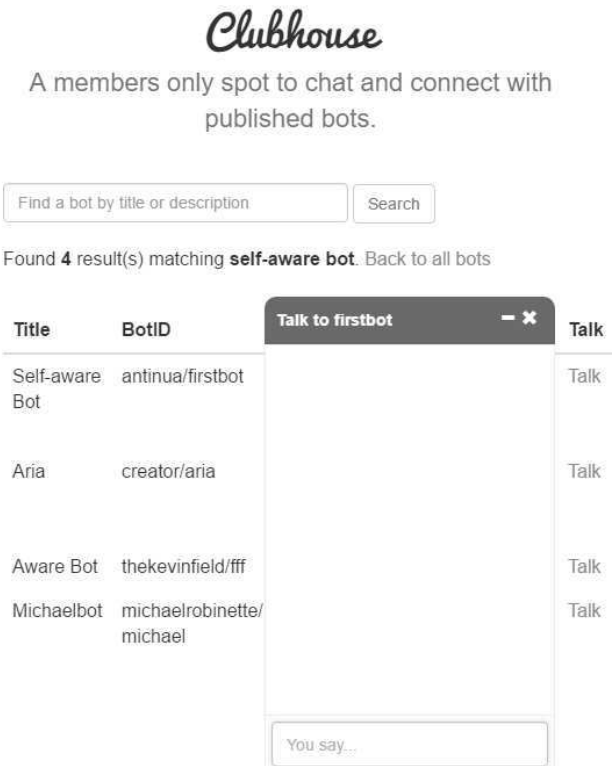
```
<category>
        <pattern>WHAT IS * CATEGORY *</pattern>
        <template><srai>CATEGORY DEFINITION</srai></template>
</category>

<category>
        <pattern>WHAT ARE CATEGORIES *</pattern>
        <template><srai>CATEGORY DEFINITION</srai></template>
</category>

<category>
        <pattern>CAN YOU DESCRIBE * CATEGORY</pattern>
        <template><srai>CATEGORY DEFINITION</srai></template>
</category>
```

Znacznik * określa dowolny wyraz lub kilka wyrazów, które mogłyby wystąpić w danym miejscu. Przykładowo użytkownik może zapytać `What is a category?` albo `What is this category?` czy też `What are categories in AIML?`, a każde z tych pytań odwoła się poprzez znacznik `<srai></srai>` do tego samego opisu pod wzorcem `CATEGORY DEFINITION`. Widok edytora platformy Playground z otwartym plikiem `chatbot.aiml` przedstawiono na rysunku 1.7.

1.4.2. Uruchomienie rozmowy z chatbotem

W celu rozpoczęcia rozmowy z zaimplementowanym chatbotem *Self-aware Botem* należy zalogować się do platformy Playground używając poświadczeń ze wspomnianych wcześniej serwisów: Facebook, Google+, Twitter czy Yahoo!. Następnie należy przejść do podstrony o nazwie *Clubhouse*, gdzie znajdują się wszystkie publicznie dostępne boty. Widok tej strony pokazano na rysunku 1.8. Chatbota można wyszukać poprzez jego nazwę *Self-aware Bot*. Po użyciu przycisku *Talk* w dolnej prawej części ekranu będzie znajdowało się okienko konwersacji. W tym miejscu można wpisywać zapytania oraz otrzymywać tekstową odpowiedź od asystenta.

Rys. 1.7: Edytor platformy Playground.



Rys. 1.8: Clubhouse - części platformy Playground, w której można rozmawiać z udostępnionymi chatbotami

1.5. Podsumowanie

Tworzenie wirtualnych asystentów zdolnych do naturalnej rozmowy z człowiekiem należy do wciąż rozwijającej się dziedziny sztucznej inteligencji. Najbardziej popularne i szeroko stosowane podejście polega na budowie bazy wiedzy składającej zawierające pary pytanie-odpowiedź z wykorzystaniem języka AIML. Strategia ta ma jednak dużo wad. Po pierwsze – jest mało elastyczna (zapytania muszą odpowiadać wzorcom przewidzianym przez twórcę bota). Po drugie – ma ona wysoki próg wejścia (związany z koniecznością nauki języka AIML oraz koniecznością zaprojektowania obszernej bazy wiedzy). Z drugiej jednak strony możliwość wykorzystania i zmodyfikowania opublikowanych kodów AIML pozwala na dość szybki start.

Rozwijane są też inne podejścia, bazujące na systemach opartych na wiedzy, wnioskowaniu na podstawie przypadków, zastosowaniu ontologii czy użyciu prawdopodobieństwa w postaci Naiwnego Klasyfikatora Bayesowskiego. Można o tym wnioskować na podstawie doniesień literaturowych. Niestety, trudno jest zaleźć dokładniejsze opisy dotyczące samej implementacji. Opublikowane materiały są niewystarczające, aby osoba zaczynająca pracę z wirtualnymi asystentami mogła zastosować w praktyce zawarte w nich wskazówki.

Użyta platforma Playground stanowi proste w nauce i obsłudze oraz darmowe środowisko. Idealnie nadaje się do tworzenia autorskich chatbotów, przy czym mogą z niej korzystać nawet osoby z małym doświadczeniem. Ponadto, dzięki innym komponentom, Pandorabots umożliwia wdrażanie aplikacji z rozszerzonym interfejsem i możliwością wizualizacji 2D. Korzystanie z tych funkcji wymaga jednak opłacenia miesięcznej składki.

Literatura

[1] A. Aamodt, E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7:39–59, 1994. `http:\www.example.org`, [dostęp dnia 7.04.2016].

[2] H. Al-Zubaide, A. Issa. Ontbot: Ontology based chatbot. *Innovation in Information Communication Technology (ISIICT), 2011 Fourth International Symposium on*, strony 7–12, List. 2011.

[3] J. Jia. CSIEC: A Computer Assisted English Learning Chatbot Based on Textual Knowledge and Reasoning. *Knowledge-Based Systems*, 22(4):249–255, Maj 2009.

[4] M. Niranjan, M. Saipreethy, T. Kumar. An intelligent question answering conversational agent using Naive Bayesian classifier. *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, strony 1–5, Sty. 2012.

[5] R. Schumaker, H. Chen. Interaction Analysis of the ALICE Chatterbot: A Two-Study Investigation of Dialog and Domain Questioning. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(1):40–51, Sty. 2010.

ROZDZIAŁ 2

ANALIZA WYDŹWIĘKU EMOCJONALNEGO TEKSTU

M. Bulanda, M. Majstrowicz

W rozdziale omówiono zagadnienia związane z analizą wydźwięku emocjonalnego tekstu. Opisano w nim także projekt autorskiej aplikacji służącej do analiz angielskich tekstów pod kątem ich emocjonalnego nacechowania oraz odpowiednio sygnalizującej użytkownikowi uzyskane wyniki.

2.1. Wstęp

Rozpoznawanie i wyrażanie emocji pełni kluczową rolę w międzyludzkiej komunikacji. Z uwagi na duże wsparcie techniczne z jakim obecnie odbywa się ta komunikacja istotnym zagadnieniem stało się opracowanie rozwiązań ułatwiających automatyczną interpretację nastroju bądź nastawienia człowieka na podstawie obserwacji jego wypowiedzi. W ostatnich latach można zanotować znaczy rozwój informatyki afektywnej (ang. *affective computing*), dziedziny nauki mającej na celu rozpoznawanie, analizę i przetwarzanie stanów emocjonalnych użytkowników komputerów. Powstające w jej ramach opracowania znajdują bardzo wiele ciekawych zastosowań.

Spektrum zastosowań aplikacji typu *wirtualny obserwator stanów wewnętrznych użytkownika* jest niezwykle szerokie. Obejmuje ono modyfikowanie na bieżąco wyglądu interfejsu komunikatorów lub przeglądarek internetowych odpowiednio do emocji rozpoznanych w czytanym bądź pisanym przez użytkownika tekście. Aplikacje tego typu mogą być stosowane w celach czysto marketingowych, odgrywając rolę sensorów klienckich opinii względem konkretnych produktów zdolnych do automatycznej analizy komentarzy czy recenzji zamieszczanych w internecie [5]. Możliwym zastosowaniem wirtualnych obserwatorów jest też współpraca np. z dziećmi dotkniętymi autyzmem czy osobami cierpiącymi na depresję. Z pewnością można byłoby wymienić jeszcze wiele innych przypadków użycia. Dzieje się tak dlatego, że emocje stanowią jedną z podstaw funkcjonowania człowieka i towarzyszą mu niemal w każdym aspekcie jego życia.

Rozpoznawanie i analiza emocji nie należą do zadań łatwych. Duże trudności powoduje chociażby wielowymiarowość struktury samej emocji, wieloznaczność słów uzależniona od kontekstu ich użycia oraz złożoność używanego języka. Osobną kwestią jest kojarzenie wyniku analizy emocji oraz reakcja programu. Nie wiadomo bowiem do końca, które z nich należy podtrzymywać, a którym przeciwdziałać oraz w jaki sposób to robić. Kłopot mogą też sprawić uwarunkowania kulturowe użytkowników, np. na odmienna percepcja kolorów (kolor biały może być kolorem radości, jak i żałoby). Ponadto niebagatelne mogą być skutki oddziaływania generowanych bodźców wizualnych na człowieka (gwałtownie zmieniającym się ekranem może mieć negatywy wpływ na osoby dotknięte epilepsją).

2.2. Analiza tekstu

Analiza tekstu jest problemem mocno skomplikowanym. Mimo, że nad jego rozwiązaniem pracowało i wciąż pracuje liczne grono naukowców, jak do tej pory nie udało się uzyskać pełnego sukcesu. Szczególnym obszarem badań jest dziedzina związana z przetwarzaniem języka naturalnego (czyli języka używanego podczas codziennej komunikacji). W jej obszarze znalazło się szerokie spektrum zagadnień – od trywialnych algorytmów służących do wyszukiwania konkretnych słów czy fraz w tekście, po zadania obejmujące tłumaczenie ich na różne języki.

2.2.1. Poziomy analizy tekstu pisanego

Przetwarzanie języka naturalnego odbywa się na siedmiu poziomach, z których każdy charakteryzuje się innym zakresem analizowanych informacji. Poniżej pokrótce opisano każdy z poziomów.

- Fonologia – dotyczy rozpoznawania i przetwarzania mowy, powiązana jest z fonetyką i bazuje na fizycznych własnościach sygnałów akustycznych towarzyszących mowie.
- Leksyka – jako wejście przyjmuje tekst w języku naturalnym. Charakteryzuje ją proces tokenizacji (identyfikacji części leksykalnych np. zdań, słów itp.) oraz znakowanie części mowy [6].
- Morfologia – dotyczy rozpoznawania sufiksów, prefiksów i fleksyjnych form słów, wyrażeń złożonych oraz przekształcanie odmienionych słów do ich bazowej formy, stosując *stemming* lub lematyzację.
- Syntaktyka – polega na identyfikacji fragmentów zdań i przypisywaniu ról do słów, analizie gramatyki danego języka i tego, w jaki sposób słowa mogą być w nim ze sobą łączone.
- Semantyka – dotyczy reprezentacji wiedzy, wyeliminowania wieloznaczności słów i rozbudowy bazy wiedzy synonimów.
- Pragmatyka – dotyczy interpolacji intencji i metafor bazując na jak największej wiedzy o świecie.
- Dyskurs – dotyczy analizy kontekstu wypowiedzi na podstawie charakteru prowadzonej narracji.

2.3. Wykorzystane słowniki

Kluczowym elementem podczas analizy tekstu jest korzystanie ze słowników definiujących znaczenie słów. Obecnie wiele instytutów zajmuje się tworzeniem baz słownikowych przydatnych w implementacji algorytmów przetwarzających tekst. Poniżej przedstawiono wybrane przykłady takich baz.

WordNet
Baza utworzona przez Instytut Nauk Poznawczych Uniwersytetu w Princeton, grupująca słowa w języku angielskim w zestawy synonimów (ang. *synsets*). Zawiera 155,237 słów kategoryzowanych według części mowy, 117,659 synsetów oraz 206,941 par słowo-znaczenie. Na całym świecie powstają odpowiedniki WordNetu mające na celu umożliwienie analizy tekstu pod różnymi kątami. Odpowiedniki te wpierają różne języki oraz rozszerzenia. Polskim odpowiednikiem jest m.in. Słowosieć tworzona na Politechnice Wrocławskiej.

WordNet-Affect
WordNet-Affect to jedno z rozszerzeń WordNetu, grupujące słowa według emocji [1] zaklasyfikowanych jako: pozytywne (podzielone dodatkowo na: entuzjazm i radość), negatywne (złość i smutek), dwuznaczne i neutralne. Baza aktywnie współpracuje z bazą synonimów WordNetu, co umożliwia łatwiejszą analizę emocjonalną wyrazów bliskoznacznych.

SentiWord
SentiWord to kolejne rozszerzenie bazy WordNetu, które proponuje odmienny system kategoryzowania słów według emocji niż WordNet-Affect. Słowa są przepisywane do następujących grup: pozytywne, negatywne, obiektywne [2]. Dodatkowo każdemu słowu nacechowanemu emocjonalnie przypisuje się polaryzację uczucia, czyli stopień, w jakim dane słowo przynależy do każdej z kategorii (wyrażane za pomocą liczby z przedziału [0,1]). Dodatkowo definiuje się objętość słowa, opisywaną wzorem: $1-(p-n)$, gdzie p to waga przynależności do kategorii pozytywne, a n – do kategorii negatywne.

2.4. Autorska aplikacja

W ramach projektu powstała aplikacja będącą podstawową wersję wirtualnego obserwatora stanu emocjonalnego użytkownika. Jej głównym zadaniem jest analiza wprowadzanego tekstu pod kątem polaryzacji uczuć. Dokładniej, ma ona sprawdzać, czy wprowadzony tekst ma wydźwięk pozytywny, negatywny czy też całkowicie neutralny.

Aplikacja została napisana w środowisku programistycznym *VisualStudio 2015* w języku programowania C# [3] i działa w trybie okienkowym. Umożliwia wprowadzenie tekstu przez użytkownika. Tekst ten zostaje poddany analizie, w wyniku której interfejs jest odpowiednio modyfikowany. Obsługiwane są tylko teksty w języku angielskim (ze względu na bardzo dużą złożoność nie zaimplementowano algorytmów analizy tekstów w języku polskim).

W architekturze aplikacji wyróżniono kilka modułów, odpowiedzialnych za realizację różnych etapów przetwarzania. Do analizy tekstu wykorzystano słownik SentiWord, natomiast do tokenizacji i POS-taggingu – bibliotekę Stanford CoreNLP. Metoda obliczania wartości wag dla całego tekstu o nazwie *'Term Score Summation'* została zaczerpnięta z pracy [2].

2.4.1. Opis działania

Aplikacja analizuje tekst wprowadzony przez użytkownika korzystając z podstawowej kategoryzacji emocji wedle słownika SentiWord: **pozytywny**, **negatywny**, **neutralny**. W pierwszej turze odbywa się tokenizacja za pomocą Stanford CoreNLP (tekst jest dzielony na pojedyncze słowa, które są klasyfikowane do odpowiednich części mowy). Wykorzystywane są też dane bazy słownika SentiWord, przypisujące zadanym słowom polaryzację uczucia. W efekcie program jest w stanie określić wydźwięk uczuciowy fragmentów wprowadzonego tekstu.

Sam proces analizy rozpoczyna się po wprowadzeniu tekstu w oknie edytora i zatwierdzeniu go przyciskiem OK. Wynik analizy jest wizualizowany zmianą koloru tła okna odpowiednio do wykrytej emocji: na kolor biały – jeśli wydźwięk tekstu jest neutralny, na kolor zielony – jeśli wydźwięk jest pozytywny, na kolor czerwony – jeśli wydźwięk jest negatywny. Program wyświetla też w konsoli dodatkowe dane, takie jak: słowa kluczowe wraz z analizą POS, wagi słownika dla wybranych słów kluczowych (kolejność: pozytywna, negatywna, obojętna), ostateczne wartości wag dla analizy całego tekstu oraz końcową ocenę wydźwięku w formie podsumowania słownego. W przypadku, gdy szukane słowo nie może zostać odnalezione w słowniku, zwracane są wagi (0,0,1) i słowo to nie jest brane pod uwagę podczas dalszej analizy. Poniżej przedstawiono części mowy z oznaczeniami wykorzystywanymi przez Stanford CoreNLP [7]:

- Adjective - przymiotnik (różne wersje JJ, JJR, JJS),
- Noun - rzeczownik (różne wersje - nieosobowe NN, NNS),
- Adverb - przysłówek (różne wersje RB, RBR, RBS),
- Verb - czasownik (różne wersje VB, VBD, VBG, VBN, VBP, VBZ).

Dodatkowo, w celu zapewnienia kompatybilności z oznaczeniami wykorzystywanymi w słowniku SentiWordNet [4], przeprowadzana jest konwersja nazw tagów do następującej formy (tag jako zmienna typu string):

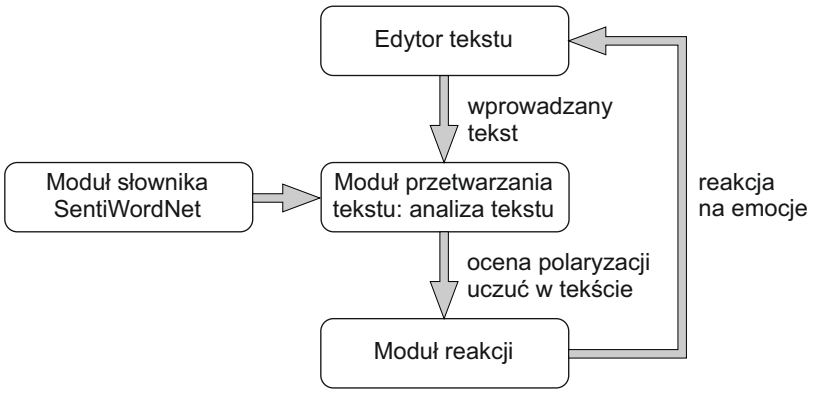
- Adjective - a,
- Noun - n,
- Adverb - r,
- Verb - v,
- w przypadku nierozpoznania żadnego z powyższych - ???

2.4.2. Moduły programowe

Aby ułatwić dalszy rozwój aplikacji oraz umożliwić jej wykorzystanie w innych projektach zaprojektowano ją używając następujących modułów (rys. 2.1):

- **Edytor tekstu** – część działająca jako pole do wpisywania tekstu i jego edycji.

- **Moduł słownika SentiWordNet** – zawiera metody pozwalające na przetwarzanie, zapis oraz wyszukiwanie słów kluczowych w słowniku SentiWordNet.
- **Moduł przetwarzania tekstu: analiza tekstu** – jego zadaniem jest wykonanie tokenizacji wprowadzonego tekstu, następnie wybór odpowiednich części mowy do analizy oraz, wykorzystując moduł słownika, obliczenie wydźwięku emocjonalnego. Ocena polaryzacji uczuć w tekście jest przekazywana dalej w formie zmiennej typu `string`, która jest dobierana na podstawie obliczonej wcześniej wartości liczbowej.
- **Moduł reakcji** – jego zadaniem jest dobór akcji modyfikującej obszar edytora tekstu w zależności od odebranej polaryzacji uczuć.



Rys. 2.1: Konstrukcja programu

Schemat działania aplikacji obejmuje następujące kroki:

1. Odbiór ustalonego fragmentu tekstu (np. zdania).
2. Analiza fragmentu.
   - tokenizacja tekstu – dzielenie tekstu na pojedyncze wyrazy,
   - *part-of-speech tagging* – zidentyfikowanie części mowy jaką reprezentuje dany wyraz,
   - ekstrakcja słów kluczowych,
   - dla każdego słowa kluczowego (przy znanym jego znaczeniu oraz tagu POS) obliczana jest (korzystając ze słownika) jego pozytywna oraz negatywna waga,
   - z uzyskanych wartości pozytywnych i negatywnych wag dla słów kluczowych liczone są wartości średnie dla całego tekstu,
   - wydźwięk emocjonalny jest analizowany dla wagi, która ma największą wartość; jeśli obie wagi, pozytywna i negatywna, są równe, wówczas polaryzacja tekstu jest neutralna i przyjmuje wartość 0,
   - na podstawie wyliczonej powyżej wartości program dobiera opis słowny wydźwięku, np. *neutralny* lub *bardzo pozytywny*.
3. Na podstawie uzyskanej oceny wydźwięku emocjonalnego tekstu program odpowiednio reaguje (obecnie zmianą tła okna edytora).

2.5. Przykładowa analiza tekstu

Program realizuje analizę emocji jedynie w oparciu o pojedyncze słowa. Nie jest to najlepsze podejście do problemu analizy wydźwięku emocjonalnego, ponieważ:

- emocjonalny wydźwięk tekstu zależy często od kontekstu, a przez wieloznaczność większości słów kontekst ten jest trudny do zidentyfikowania. Przykładowo słowo *strasznie* może sygnalizować strach, ale także inne, skrajnie odmienne odczucia, jeśli wystąpi ono w odpowiednim kontekście, jak np. *strasznie fajne*,
- brak jest możliwości względnej oceny wydźwięku emocjonalnego. Zdania takie, jak: *Ja się z niego śmiałem* oraz *On śmiał się ze mnie* będą miały taki sam wydźwięk, mimo że przedstawiają różne punkty widzenia.

Ponadto należy jeszcze brać pod uwagę błędy powstałe w trakcie analizy POS, tzn. błędne oznaczenie części mowy. Ma to ogromny wpływ na ocenę polaryzacji tekstu.

W trakcie testów zauważono, że słowa całkowicie neutralne (o wagach 0,0,1) mają duży wpływ na końcowy wynik analizy. Ze względu na specyfikę sprawdzania polaryzacji całego tekstu ich obecność rozrzedzała wagi pozytywne i negatywne (ze względu na fakt, iż nie zmieniały się ich wartości, a brane były pod uwagę przy liczeniu średniej), co było niepożądanym efektem. Z tego powodu zdecydowano się na wprowadzenie ograniczenia, które sprawia, że program nie bierze pod uwagę słów z zerowymi wagami pozytywnymi i negatywnymi.

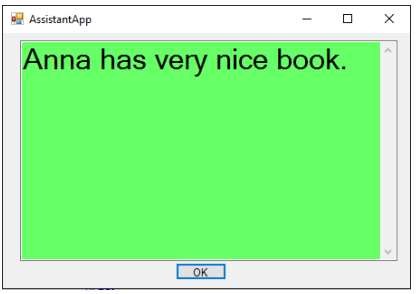
Poddając analizie przykładowe zdanie: ***Anna has very nice book (Anna ma bardzo ładną książkę)*** otrzymano następujący wynik:

Analiza:
Anna rzeczownik
has czasownik
very przysłówek
nice przymiotnik
book rzeczownik

Do określenia poziomu emocjonalnego program wybrał dwa słowa: *very* oraz *nice*. Po uzyskaniu wag dla poszczególnych słów nastąpiło wyliczenie wag średnich dla całego tekstu – była to zwykła średnia arytmetyczna. Ponieważ waga pozytywna jest większa od negatywnej, więc ostateczna ocena całego zdania będzie oparta o jej wartość. Według programu, wartość 0.5625 odpowiada ocenie *very positive*. Kolor okna edycji został zmieniony zgodnie ze zdefiniowanymi zasadami. Przykład ten został przedstawiony na rysunku 2.2a).

Podczas analizy innego zdania: ***This dress looks terrible (Ta sukienka wygląda okropnie)*** do określenia poziomu emocjonalnego program wybrał tylko jedno słowo: *terrible*. Pozostałe słowa nie zostały znalezione w słowniku (bądź też wartości ich wag pozytywnych i negatywnych były równe 0). Analiza wyniku działania programu w tym przypadku jest prosta: waga negatywna jest większa

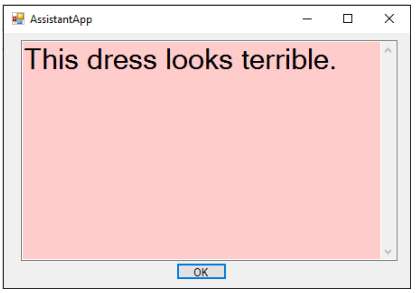
a)




```
Anna - NNP ???
has - VBZ v
very - RB r
Term very
0,25 0 0,75
nice - JJ a
Term nice
0,675 0 0,125
book - NN n
```

```
Analiza tekstu
0,5625 0 0,4375
very positive
```

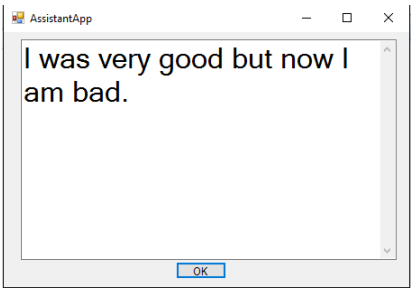
b)




```
This DT ???
dress - NN n
looks - VBZ v
terrible - JJ a
Term terrible
0,125 0,25 0,625
```

```
Analiza tekstu
0,125 0,25 0,625
slightly
negative
```

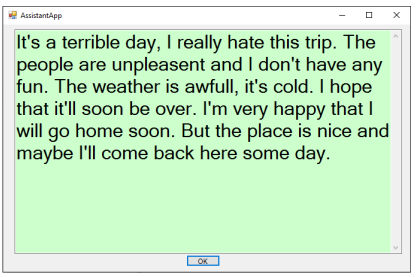
c)




```
very - RB r
Term very
0,25 0 0,75
good - JJ a
Term good
0,5 0 0,5
bad - JJ a
Term bad
0 0,75 0,25
```

```
Analiza tekstu
0,25 0,25 0,5
neutral
```

d)




```
Analiza tekstu
0,229 0,125
0,646
slightly
positive
```

Rys. 2.2: Przykłady wyników analizy prostych tekstów: (a, b, c) oraz tekstu bardziej złożonego (d). Kolor tła w okienkach edycji odpowiada rozpoznanym emocjom, zaś widoczny po prawej stronie tekst to wartości wyliczonych parametrów wypisywane na konsoli

od pozytywnej i to ona jest brana pod uwagę przy ocenie polaryzacji zdania, którą program ocenił na *trochę negatywną* (zobacz rysunek 2.2b).

Dla kolejnego zdania ***I was very good but now I am bad. (Byłem bardzo dobry, ale teraz jestem kiepski/zły)*** program wziął pod uwagę trzy słowa: *very*, *good* i *bad*. W oknie konsoli można zauważyć, że po obliczeniu średnich wag dla całego tekstu wagi pozytywna i negatywna uzyskały tą samą wartość. W tym przypadku program ocenił fragment jako *neutralny* (zobacz rysunek 2.2c) (informacje ograniczone są tylko do termów).

Analizie poddano także dłuższy tekst, zawierający zróżnicowany zestaw słów nacechowanych pozytywnie jak i negatywnie. Zgodnie z oczekiwaniem program ocenił tekst jako *trochę pozytywny*. Przykład ***analizy tekstu bardziej złożonego*** przedstawiono na rysunku 2.2d).

2.6. Podsumowanie

Ze uwagi na potencjał do opracowania funkcjonalnych rozwiązań rozpoznawanie emocji jest ciekawym zagadnieniem. Jednak algorytmy, jakie należy przy tym zaimplementować, są skomplikowane i podatne na generowanie pomyłek. Sam proces wyrażania emocji za pomocą słów jest już mocno złożony. Cechuje go przede wszystkim wieloznaczność znaczeniowa uzależniona od kontekstu wypowiedzi danej osoby oraz jej środowiska.

Przedstawiona aplikacja stanowi jedynie bardzo uproszczoną wersję wirtualnego obserwatora. Istnieje jednak możliwość jej rozbudowy np. w stronę bardziej interaktywnych reakcji programu na wpisywany tekst (generowania reakcja na emocje w czasie rzeczywistym, co zdanie lub określony odcinek czasu) lub w stronę nowych metod analizy, wykorzystujących inne słowniki oraz złożone modele wykrywanych emocji. Można również pójść w kierunku implementacji algorytmów ujednoznacznienia sensu słów znajdujących się w słownikach albo spróbować obsłużyć opisane wcześniej przypadki użycia.

Warto też zauważyć, że implementacja aplikacji w wersji podstawowej wcale nie była łatwa. Problemy napotkano już na etapie opracowywania samej koncepcji, a potem, w fazie planowania, pojawiły się nowe, nieprzewidziane. Wraz ze wzrostem zaawansowania budowanego rozwiązania liczba pojawiających się problemów może tylko rosnąć. Zwłaszcza, że język naturalny jest obiektem bardzo trudnym do jednoznacznej analizy. Niemniej możliwości i potencjalne korzyści jakie niesie ze sobą inteligentne rozpoznawanie emocji są zbyt duże, aby zrezygnować z prób.

Literatura

[1] A. Agrawal, A. An. Unsupervised emotion detection from text using semantic and syntactic relations. *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, wolumen 1, strony 346–353. IEEE, 2012.

[2] A. Hamouda, M. Rohaim. Reviews classification using SentiWordNet lexicon. *World Congress on Computer Science and Information Technology*, 2011.

[3] M. Lis. *C#: praktyczny kurs.* Helion, 2012.

[4] SentiWordNet, strona domowa projektu. `http://sentiwordnet.isti.cnr.it/`. [dostęp dnia 29.05.2016].

[5] N. Shelke. Approaches of emotion detection from text. *International Journal of Computer Science and Technology*, 2(2), 2014.

[6] P. Sołdacki. *Zastosowania metod płytkiej analizy tekstu do przetwarzania dokumentów w języku polskim.* Rozprawa doktorska, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska, Warszawa, 2006. `https://repo.pw.edu.pl/docstore/download.seam?fileId=WEiTI-b192c072-00cc-41df-9bba-a2b0a211e9bc` [dostęp dnia 16.06.2016].

[7] Stanford Core NLP, strona domowa projektu. `http://stanfordnlp.github.io/CoreNLP/`. [dostęp dnia 01.06.2016].

ROZDZIAŁ 3

KOMUNIKACJA NIEWERBALNA W IMPLEMENTACJI CYFROWEJ

B. Czyżewska, T. Rusiński

Niniejszy rozdział poświęcono zagadnieniom komunikacji niewerbalnej. Korzystając z opublikowanych wyników badań zaproponowano dwa sposoby wysyłania komunikatów niewerbalnych: na łączu program komputerowy-człowiek oraz na łączu humanoid-człowiek.

Na początku rozdziału zamieszczono krótki wstęp o naturze komunikacji niewerbalnej oraz jej technicznych zastosowaniach. W dalszej części przedstawiono autorskie implementacje dwóch programów wysyłających komunikaty niewerbalne za pomocą mowy gestów połączonej z symboliką kolorów oraz mimiką. Działanie obu programów przetestowano z udziałem kilkuosobowych grupach respondentów, przed którymi postawione zostało zadanie odczytania wygenerowanych komunikatów.

3.1. Wstęp

Komunikacja niewerbalna wydaje się być pojęciem tak oczywistym, że aż trudnym do zdefiniowania. Może dlatego trudno jest znaleźć jej hasłowe opracowanie zarówno w literaturze popularnonaukowej, jak i w podręcznikach akademickich. Zazwyczaj ludzie posługują się tym terminem bez wnikania w jego istotę. Jednak komunikacja niewerbalna daje się ująć w formalne ramy. W najszerszym ujęciu kojarzy się z nią wszystkie sposoby przekazywania informacji bez użycia mowy. W kontekście międzyludzkich interakcji zazwyczaj wiąże się ją z mową ciała – zestawem gestów, postawy i mimiki. Oczywiście nie wyczerpuje to listy komunikatów, jakie człowiek jest w stanie odebrać. Potrafimy rozpoznawać informacje przekazywane niewerbalnie przez zwierzęta, zarówno w zakresie zachowań wpojonych przez człowieka (np. wskazanie tropu przez gończego psa), jak i naturalnych reakcji niewystępujących u ludzi (ruchy ogona, nastroszenie futra). Istotą prac opisanych w tym rozdziale było znalezienie zestawu komunikatów nie tyle nowych, ile jednoznacznych w swej treści i dostosowanych do ogra-

niczeń wytyczonych przez charakter zastosowanej platformy: komputerów oraz humanoidalnych robotów pozbawionych mimiki czy wyświetlacza.

Do przekazywania informacji bez użycia mowy wykorzystuje się najróżniejsze systemy. Wystarczy wymienić tu kody kolorów (np. sygnalizacja drogowa, oznaczenia jaskrawymi barwami produktów objętych promocjami), język figur geometrycznych (np. oznaczenia publicznych toalet kółkiem lub trójkątem), mnemotechniczne wskaźniki (migotania lampek kontrolnych, wychylenia wskazówek), czy też szeroką gamę piktogramów (wykorzystywanych w różnego rodzaju systemach oznaczeń – choćby w znakach drogowych). Jednak nie wszystkie komunikaty niewerbalne są tak łatwe do zidentyfikowania i zrozumienia, jak te z przytoczonych przykładach. Część komunikatów jest przekazywana i odbierana zupełnie nieświadomie, inne zaś są starannie zaplanowane tak, aby wywrzeć na odbiorcę określony wpływ bez uświadamiania mu istnienia przekazu. Doktor Mary R. Power z Bond University [5] zalicza do komunikacji niewerbalnej również umeblowanie i aranżację biur i mieszkań prywatnych czy styl ubioru.

Komunikacja niewerbalna odbywa się niemal w sposób ciągły, łamiąc granice wytyczone przez ramy ustnych wypowiedzi. Widać to między innymi podczas używania programów komputerowych. Mało komu trzeba tłumaczyć, do czego służy znak krzyżyka w rogu okienka, zapełniający się pasek postępu czy migający kursor. W dobrym programie komputerowym niemal cała informacja przekazywana jest w formie niezwerbalizowanej i bez wykorzystania pisma. Użytkownicy bowiem sięgają po pisemne oznaczenia dopiero wtedy, kiedy program staje się nieczytelny. Zazwyczaj do zapewnienia interakcji wystarcza migotanie kursora, szary kolor obszarów nieaktywnych, czerwone gwiazdki pól obowiązkowych i zestaw prostych obrazków: krzyżyk, kółko zębate, dyskietka. Oznaczenia te tworzą swoisty język, który, co ciekawe, zaczął oddziaływać na język pisany. Tak jest w przypadku emotikonów – znaków, które zaczęto stosować do transkrypcji mimiki na znaki diakrytyczne, a obecnie ich interpretacja odbywa się na osobnych zasadach (co widać na interfejsach portali społecznościowych i komunikatorów internetowych, gdzie pojawiają się coraz to nowe zestawy emotikonów i „naklejek", częstokroć tak abstrakcyjnych, że aż trudnych do słownego opisania).

Podobny do ludzkiego kształt robota NAO wymusza podobieństwo komunikatów do tych przekazywanych przez ludzi, a więc postawy ciała i gestów. Dzięki zastosowaniu ledów RGB wokół jego „oczu" rozszerzyło tę listę o możliwość przekazywania sygnałów za pomocą kolorów. Opracowywanie zestawu gestów wykonywanych przez robota i zrozumiałych dla ludzi niesie ze sobą liczne wyzwania (stopnie swobody robota nie są równe ludzkim, przez co ludzkie ruchy nie mogą być w pełni odwzorowane) oraz niebezpieczeństwo zapadnięcia w Dolinę Niesamowitości. Termin ten ukuł japoński robotyk Masahiro Mori na określenie nieprzyjemnych odczuć, jakie stają się udziałem człowieka postawionego w obliczu czegoś, co przypomina człowieka, ale nim nie jest, na przykład androida o doskonale ludzkiej powłoce, lecz zachowującego się w sposób maszynowy (sztywny, pozbawiony emocji). Z pomocą przychodzi tu wzornictwo robotów NAO - nawiązanie do kształtów zabawki oraz brak mimiki nie pozwalający uznać NAO za coś nazbyt ludzkiego.

3.2. Implementacje cyfrowe komunikacji niewerbalnej

Od wielu lat w świecie nauki i techniki podejmowane są próby stworzenia inteligentnych agentów w postaci robotów [1] lub programów komputerowych, współpracujących z ludźmi w otaczających ich środowisku [3, 6, 7]. Tacy agenci powininni być w stanie komunikować się z użytkownikami, a nawet wchodzić z nimi w fizyczne interakcje – o ile pozwalają im na to uwarunkowania techniczne oraz wyznaczone cele. Zagadnienie to okazało się wystarczająco rozległe i trudne, aby stać się inspiracją do podejmowania wielu naukowych prac i badań [4].

O różnych aspektach komunikacji niewerbalnej wspomniano już we wcześniejszej części tego rozdziału. Pojawia się jednak pytanie: w jaki sposób jest to powiązane z tworzeniem inteligentnego agenta? Skupiając się nad tym zagadnieniem można zastanawiać się nad treścią przesyłanych komunikatów, jak również nad formą, w jakiej są one przekazywane. W przypadku robotów czy programów komputerowych dąży się to tego, aby przesyłane komunikaty formułowane były w języku zrozumiałym dla odbiorcy, czy to za pośrednictwem tekstu wyświetlanego na ekranie albo syntezowanych komunikatów głosowych. Praktyka pokazała, że w wielu przypadkach użytkownicy wolą komunikację słowną. Jest to dość zrozumiałe, zważywszy że jest to najbardziej powszechny sposób komunikowania się ludzi między sobą. Jednak w przypadku maszyn złożoność mowy stanowi niemały problem. Samo generowanie głosowych komunikatów nie wystarcza, muszą one być przekazane z odpowiednią intonacją i szybkością, by jak najbardziej przypominały ludzką mowę. Aby wzbogacić ekspresję często rozważa się możliwość dołączenia do komunikatów mowy ciała i gestykulacji.

Do komunikacji niewerbalnej zalicza się: gesty, wyraz twarzy, postawa, sposób mówienia. Przyjęto więc, że inteligentny agent powinien móc wyrażać swój stan z odpowiednią do sytuacji ekspresją oraz zależnie od istoty przekazywanej informacji, przy czym nie może odbywać się w sposób losowy ani rutynowy (z wykorzystaniem zestawu przygotowanych wcześniej scenariuszy). W pierwszym przypadku prowadziłoby to do braku zrozumienia, w drugim – do utraty naturalności (w rozmieniu człowieka).

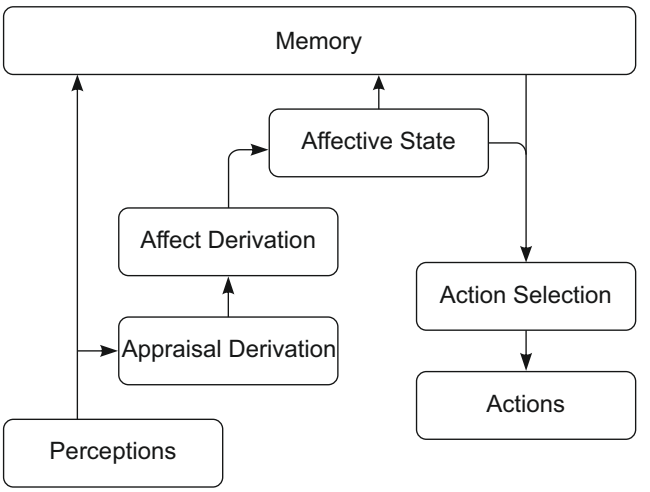
Agent również powinien umieć przystosować się do sytuacji, w której się znalazł. W tym celu opracowywane są wszelkiego rodzaju obliczeniowe modele emocji, które w zależności od zadanych przesłanek mogą w sztuczny sposób wygenerować coś na kształt emocji wpływających na zachowanie agenta oraz sposób jego komunikacji z człowiekiem. W dalszej części rozdziału przeanalizowano kilka znanych z literatury modeli, skupiających się na generowaniu emocji i zachowań inteligentnych agentów.

3.2.1. Analiza dostępnych modeli i architektur

FAtiMA

FAtiMA (ang. *Fearnot AffecTive Mind Architecture*) [2] jest architekturą, która stara się wykorzystywać sztuczne emocje inteligentnego agenta oraz tworzyć sztuczną osobowość w celu wpływania na zachowania agenta. Jest jedną z bardzo rozwiniętych tego typu architektur i oferuje bardzo wiele funkcji, np. uwzględ-

nianie zachowań wynikających z kultury oraz motywacji. Ponieważ implementacja wszystkich funkcji jest bardzo trudna i niekoniecznie przydatna w każdym projekcie zakładającym istnienie inteligentnego agenta, w architekturze tej wyróżniono specyficzne moduły. Jednym z nich jest moduł zajmujący się emocjami agenta. Podział na moduły w znacznym stopniu uprościł wykorzystanie całego rozwiązania w rzeczywistych implementacjach obsługujących prostsze przypadki użycia oraz zapewnił łatwą jego rozbudowę.



Rys. 3.1: Architektura jądra FAtiMA [2]

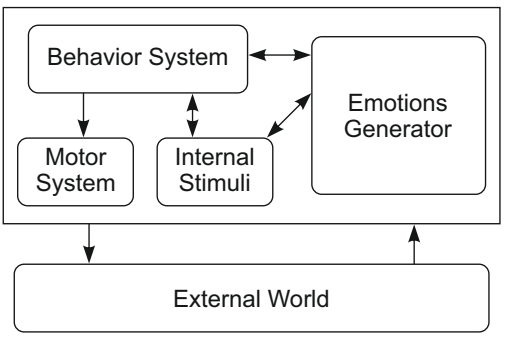
Głównym elementem architektury FAtiMA jest jej jądro, do którego dokłada się kolejne komponenty poszerzające dostępne możliwości (rys. 3.1). W zarysie działanie rozwiązań zbudowanych w tej architekturze jest następujące. Agent dostaje informacje pochodzące ze świata zewnętrznego. Są one wykorzystywane do aktualizowania pamięci agenta oraz stanu, w jakim się znajduje. Ich dostarczenie rozpoczyna proces oceny sytuacji oraz generowania potencjalnych akcji do podjęcia (ang. *appraisal process*). Zatem to, co agent postanowi zrobić oraz jak odniesie się do napotkanej sytuacji zależy w głównej mierze od zaimplementowanego systemu oceny.

Architektura FAtiMA jest bardzo rozbudowana, a jej efektywne wykorzystaniem wymaga ogromnej wiedzy i czasu. Z powodu wysokiego progu wejścia nie została wybrana do wykonania autorskich implementacji.

Cathexis

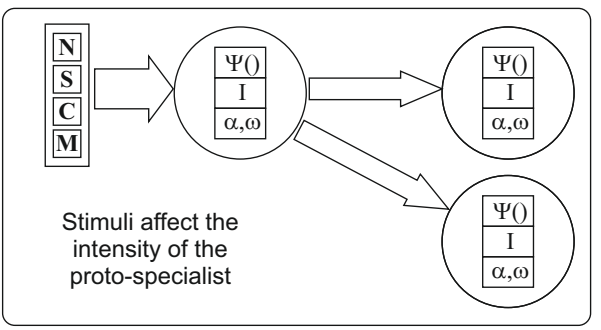
W latach 90-tych na uniwersytecie MIT (ang. *Massachusetts Institute of Technology*) zaproponowano obliczeniowy model emocji *Cathexis*. Model ten został zainspirowany przez prace dokonane w dziedzinie psychologi, etologi i neurobiologi. Opis modelu, jego składowe części oraz samą implementację przedstawiono szczegółowo w [9].

Model *Cathexis* jest częścią obiektowego frameworku dostarczającego narzędzi do symulowania emocji w implementacjach sztucznych agentów. Tacy agenci mogą być częścią aplikacji wykorzystywanych w rozrywce (interaktywne sztuki teatralne, gry komputerowe itd.), edukacji (nauczyciele, trenerzy itp.) oraz mogą być elementami interfejsu służącego do komunikacji człowiek – komputer. Agent otrzymuje informacje ze świata zewnętrznego oraz podejmie określone akcje na podstawie wygenerowanych emocji i wbudowanego systemu zachowań (rys. 3.2).



Rys. 3.2: Model Cathexis [9]

Model *Cathexis* opiera się na prostym założeniu – emocje tworzą sieć, a każdy węzeł tej sieci reprezentuje inną emocję opisywaną przez kilka zmiennych i funkcji. Połączenia pomiędzy węzłami oznaczają wpływ, jaki dana emocja wywiera na inną (pobudza ją lub wypiera). Dodatkowo wprowadzona została funkcja, która służy do obliczania intensywności emocji każdego z węzłów na podstawie informacji o aktualnym stanie agenta oraz intensywności pozostałych emocji. Takie węzły nazywano proto-specjalistami.



Rys. 3.3: Schemat Proto-Specialisty [9]

Każdy proto-specjalista reprezentuje rodzinę emocji i jest połączony z pozostałymi proto-specjalistami (rys. 3.3). Z każdym proto-specjalistą stowarzyszone są cztery różne sensory: nerwowy, sensoryczno-motoryczny, motywacyjny

i kognitywny. Sensory te odpowiadają za informacje dostarczane ze środowiska zewnętrznego i wewnętrznego agenta. Ponadto każdy proto-specjalista opisywany jest przez cztery wartości: α – oznaczającą poziom aktywacji emocji oraz ω – oznaczającą poziom nasycenia emocji, funkcję $\Psi(\cdot)$ – odpowiadającą za czas trwania emocji w czasie oraz I - intensywność emocji wyliczaną ze wzoru:

$$I_{et} = \chi\left(\Psi(I_{et-1}) + \sum_{k} L_{ke} + \sum_{l} G_{le} \cdot I_{lt} - \sum_{m} H_{me} \cdot I_{mt}\right)$$

gdzie: I_{et} – wartość intensywności emocji, $\Psi(\cdot)$ – funkcja rozkładu emocji w czasie, L_{ke} – wartość wzbudzenia emocji, G_{le} – wartość wzbudzenia l przez emocję e, I_{lt} – intensywność emocji l w czasie t, H_{me} – wartość obniżenia emocji m przez emocję e, I_{mt} – wartość emocji m w czasie t, $\chi(\cdot)$ – funkcja, która zapewnia że wartość intensywności emocji będzie w przedziale od 0 do wartości nasycenia danej emocji.

Po przeanalizowaniu modelu Cathexis zdecydowano się na wykorzystanie jego uproszczonej wersji w implementacji autorskiej aplikacji.

3.3. Propozycja implementacji komunikacji niewerbalnej w programie komputerowym

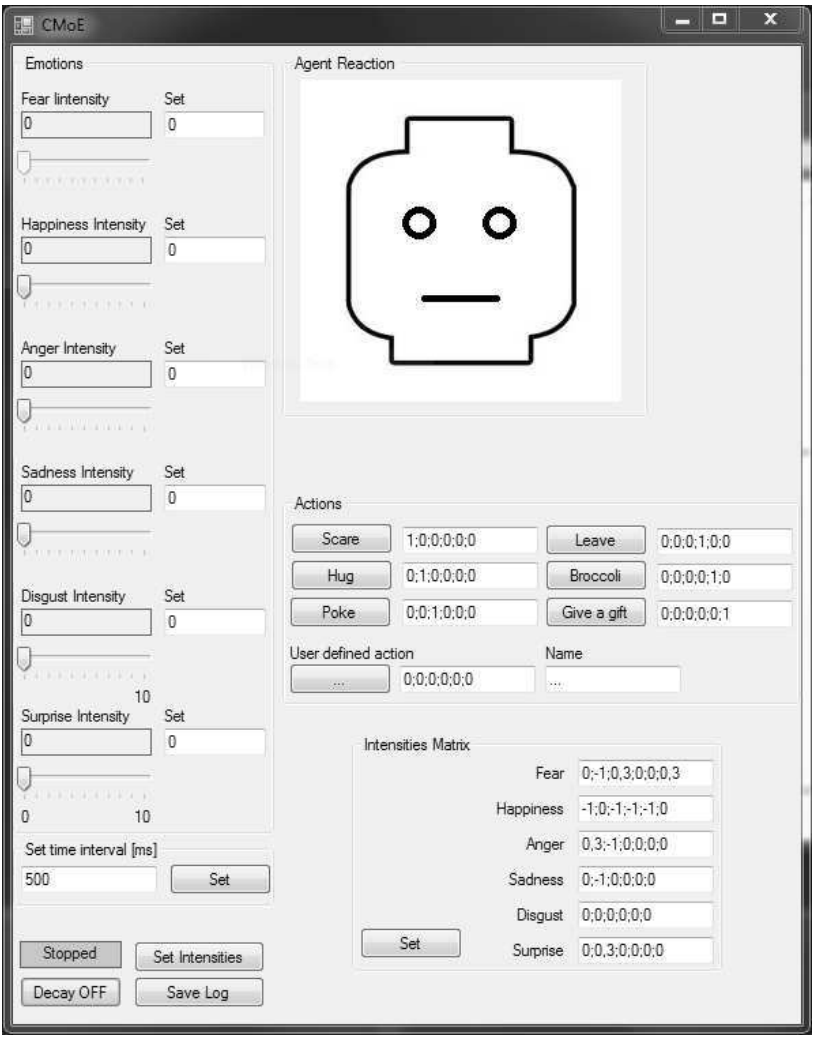
Wzorując się na modelu Cathexis zaproponowano i zaimplementowano własny obliczeniowy model emocji w formie aplikacji komputerowej z graficznym interfejsem użytkownika. Model ten nazwano CMoE (ang. *Computational Model of Emotions*). Jest on w dużym stopniu uproszczonym modelem Cathexis. W oryginalnym modelu wykorzystuje się bardzo szczegółowe informacje dotyczące stanu w jakim aktualnie znajduje się agent, np. ciśnienie krwi w mózgu, temperaturę otoczenia itd. co odzwierciedlać ma wyniki wieloletnich badań z dziedziny psychologi, biologi i neurobiologi. W modelu CMoE zaś ograniczono się jedynie do symulacji wpływu emocji na siebie. Dodatkowo zaimplementowano zestaw narzędzi pozwalających na przeprowadzanie badań modelu oraz wizualizację aktualnego stanu agenta.

3.3.1. Opis aplikacji

Główne okno aplikacji

Aplikacja została napisana przy wykorzystaniu platformy programistycznej .NET, języka C# oraz biblioteki WindowsForms. Oferowany przez nią graficzny interfejs użytkownika składa się z głównego okna (rys. 3.4), którego elementy zostaną opisane w kolejnych podrozdziałach.

W oknie głównym, obok kontrolek pozwalających na zmianę parametrów określających poziom emocji (rys. 3.5a) umieszczono panel, na którym wizualizowany jest aktualny stan emocjonalny agenta. Wizualizacja ta przyjmuje postać grafik (rys. 3.5b) zmieniających się odpowiednio do wyliczonej wartości intensywności emocji.

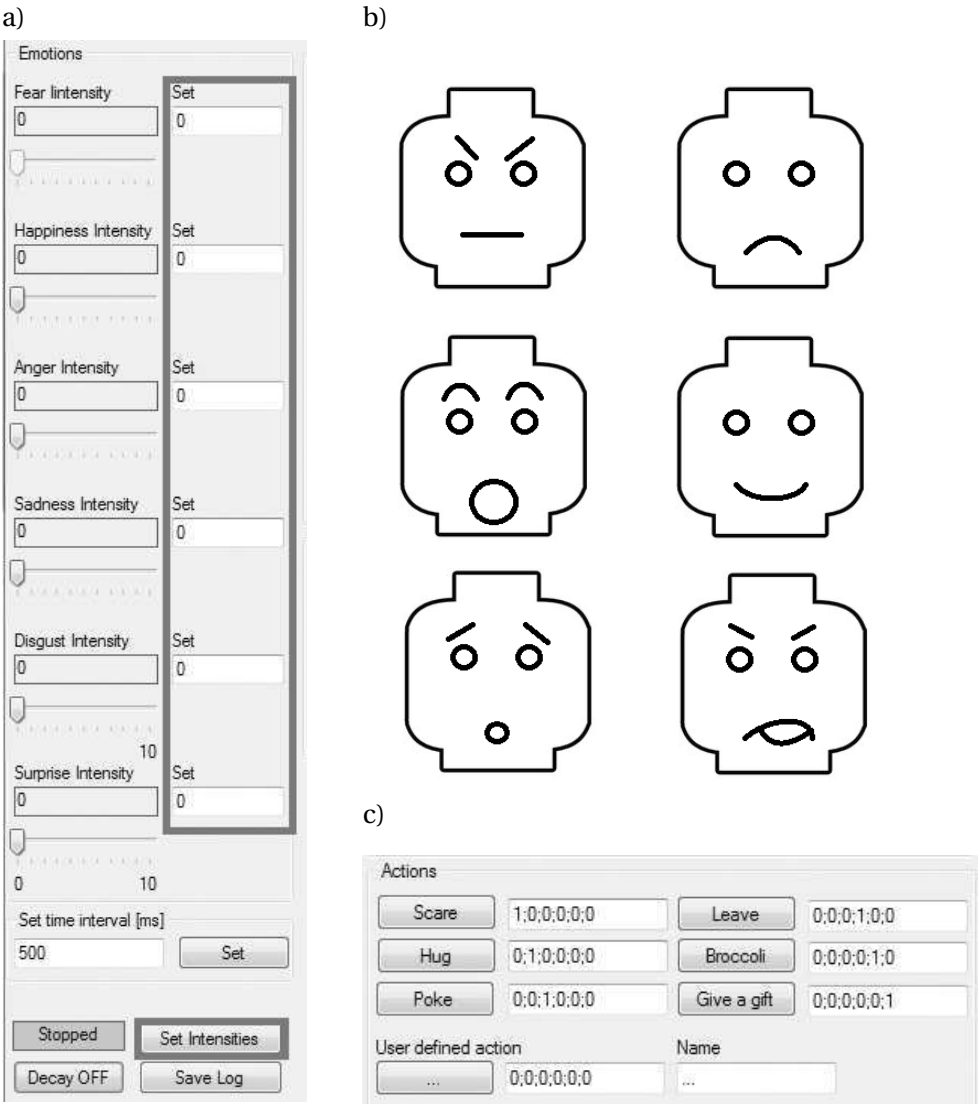


Rys. 3.4: Główne okno aplikacji

Poniżej wizualizacji stanu agenta znajduje się panel zawierający wszystkie zdefiniowane akcje pobudzające agenta oraz miejsce na wprowadzenie własnych akcji przez użytkownika i nadawanie im nazw poprzez wypełnienie pola Name i zatwierdzenie przyciskiem Enter (rys. 3.5c).

Wpływ danej akcja na agenta określa wektor znajdujący się obok przycisku reprezentującego akcję. Wektor ten składa się z szeregu liczb $[0;0;0;0;0;0]$ oznaczających wagi z jakimi dane zdarzenie będzie miało wpływ na kolejne emocje: Strach, Szczęście, Złość, Smutek, Obrzydzenie i Zaskoczenie. Wartości wag mogą być modyfikowane przez użytkownika (rys. 3.6).

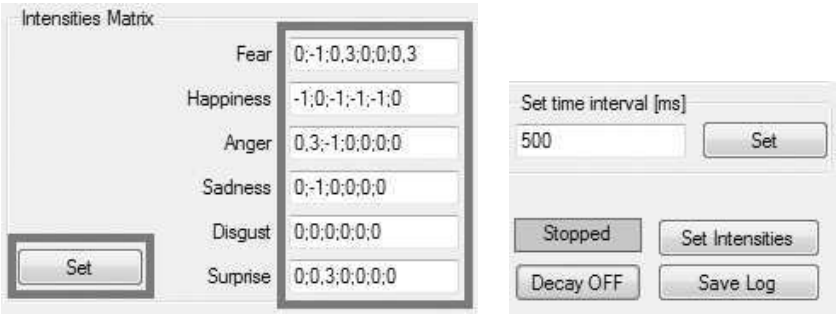
Poniżej panelu z dostępnymi akcjami znajduje się główna macierz, która definiuje oddziaływanie pomiędzy emocjami. Użytkownik może w powolny spo-



Rys. 3.5: Elementy graficznego interfejsu użytkownika a) parametry intensywności emocji, b) graficzna wizualizacja emocji, c) dostępne akcje

sób edytować oddziaływania podobnie jak w przypadku akcji poprzez modyfikowanie wartości poszczególnych wierszy macierzy i potwierdzanie zmiany przyciskiem Set.

Ostatnim elementem głównego okna aplikacji są przyciski umożliwiające rozpoczęcie symulacji oraz zapisanie pliku z informacjami dotyczącymi intensywności emocji rozłożonych w czasie przeprowadzania symulacji. Powyżej znajduje się panel umożliwiający zmianę interwału czasowego prowadzonej symulacji (domyślnie jest to pół sekundy).



Rys. 3.6: Macierz intensywności i sterowanie symulacją

Działanie aplikacji

Po rozpoczęciu symulacji na bieżąco wyliczany jest aktualny stan agenta (intensywności emocji) oraz wyświetlana jest odpowiadająca mu grafika. Przy braku jakichkolwiek pobudzeń stan agenta stabilizuje się na wszystkich wartościach intensywności emocji równych 0. Przy pobudzeniu intensywności te zmieniają się w zależności od wprowadzonych ustawień.

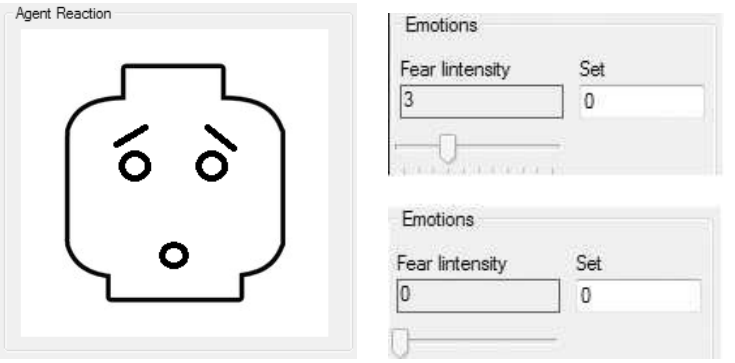
3.3.2. Badania

W celu przetestowania zaimplementowanego modelu przeprowadzono kilka symulacji. Na początku agent znajdował się w stanie podstawowym, wszystkie intensywności emocji na poziomie 0. Następnie agent został pobudzony bodźcem zwiększającym intensywność strachu o 3 (rys. 3.7). Wzrost intensywności wpłynął na aktualny stan agenta (rys. 3.8). Po pewnym czasie intensywność emocji znów spadła do zera.

W kolejnej, podobnej symulacji, bodziec pobudzający strach zwiększał jego intensywność o 7. Po pobudzeniu nie tylko intensywność strachu zwiększyła się, ale również intensywność złości wzrosła. Stan agenta wszedł w punkt stabilności z niezerowymi wartościami intensywności emocji strachu i złości. Aby zmienić ten stan pobudzono agenta bodźcem zwiększającym szczęście o 3 (rys. 3.9 i 3.10). Po tym pobudzeniu intensywności strachu i złości zaczęły spadać do zera.

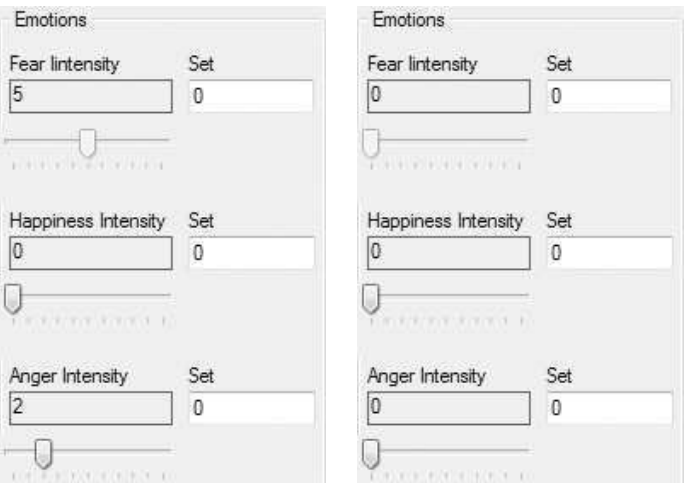
Przeprowadzone badania pokazały podobieństwo pomiędzy sztucznym agentem a prawdziwym człowiekiem. Reakcje agenta w bardzo dużym stopniu odzwierciedlały reakcje człowieka na podawane bodźce. Pomimo uproszczenia modelu można z jego pomocą symulować proste scenariusze zachowań.

Rys. 3.7: Panel intensywności akcji



Rys. 3.8: Reakcja agenta na wzrost intensywności strachu

Rys. 3.9: Intensywność akcji wpływających na agenta w kolejnych krokach symulacji



Rys. 3.10: Poziomy emocji w kolejnych krokach symulacji

3.3.3. Podsumowanie

Zaimplementowana aplikacja daje możliwości przeprowadzania badań nad obliczeniowym modelem emocji – istotnym zagadnieniem dla rozwoju robotów oraz systemów, które współdziałać mają z człowiekiem w jego naturalnym otoczeniu. Na jej przykładzie pokazano, że stworzenie środowiska do symulowania rzeczywistych emocji nie jest aż tak bardzo skomplikowane. Wszystko zależy od przyjętych założeń oraz od wiedzy posiadanej przez zespół osób odpowiedzialnych za implementacje. Oczywiście bez doświadczenia z zakresu neurobiologi oraz psychologi trudno jest stworzyć wiarygodny model, który bezbłędnie symulowałby rzeczywiste zachowania ludzkie. Zwłaszcza, że zdobycie takiego doświadczenia jest bardzo trudne i trwa wiele lat. Jednak już samo stworzenie prostego modelu daje ogólne rozeznanie co do charakteru symulowanych emocji i może stać się bazą wyjściową do tworzenia bardziej skomplikowanych modeli.

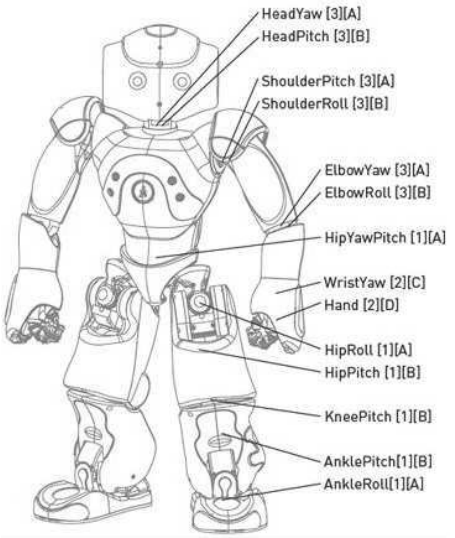
3.4. Propozycja implementacji komunikacji niewerbalnej w programie dla robota NAO

Drugim z opisywanych autorskich programów powstał na platformie robota NAO, a dokładniej – w środowisku umożliwiającym wirtualne testowanie algorytmów sterowania dla tego robota dostarczanym wraz z nim.

Implementacja elementów komunikacji niewerbalnej w zachowaniu robota NAO nie może być jedynie prostym odzwierciedleniem ludzkiej mowy ciała ze względu na brak mimiki oraz odmienną od ludzkiej budowę. Zaproponowano więc zrównoważenie owych braków poprzez wykorzystanie w komunikacji oczu robota. Należy tu podkreślić, iż to, co w fizjonomii NAO gra rolę oczu, w rzeczywistości jest nieruchomą parą czujników światła podczerwonego, z których każdy otoczony jest ośmioma programowalnymi LEDami RGB. Zapalaniem i gaszeniem diod można symulować mruganie oczu, a dodatkowo zmianą koloru można sygnalizować nastrój. Wyświetlanym kolorom przypisano powszechnie obowiązujące znaczenia: kolor czerwony interpretowany jest jako przeczenie lub brak zgody, a zielony – jako potwierdzenie lub zgoda. Ponadto posłużono się zaczerpniętą z języka literackiego koncepcją powiązania zmian w kolorze oczu z emocjami: „pojaśniałe z radości", „pociemniałe z gniewu" czy też „pełne psotnych iskierek". Choć jest to koncepcja czysto literacka, okazała się bardzo przydatna do wzbogacenia ekspresji emocji robota. Dodatkowym ułatwieniem jest podobieństwo NAO do zabawki – zazwyczaj odbiorca czy też współuczestnik relacji nie oczekuje od takiego robota czysto ludzkich zachowań, jest za to gotów zaakceptować pewne skróty myślowe i zachowania symboliczne.

3.4.1. Platforma robotyczna

NAO jest robotem mobilnym dwunożnym, ale ze względów bezpieczeństwa na potrzeby programu ustawiono go w stabilnej pozycji siedzącej. Robot jest wyposażony w wiele czujników, umożliwiających rozpoznawanie otoczenia: czujniki podczerwieni, mikrofony, sonary, czujniki dotyku i kamery. W projekcie wy-



Skonstruowany przez firmę Aldebaran komercyjno-edukacyjny humanoidalny robot NAO H25 w wersji akademickiej mierzy 58 cm wzrostu i waży 5,4 kg. Posiada 25 stopni swobody, z czego po pięć przypada na każdą kończynę. Liczba i rozmieszczenie napędów pozwala z grubsza naśladować ludzkie ruchy, jednak przy pewnych ograniczeniach narzuconych przez rozwiązania motoryczne. W szczególności dotyczy to ruchów dłoni, które wyposażono tylko w trzy palce, z czego jeden przeciwstawny. Palce te mają wspólne sterowanie, pozwalające jedynie na ich zaciśnięcie lub rozwarcie, co uniemożliwia wykonywanie bardziej skomplikowanych gestów. Za to możliwy jest dość szeroki zakres ruchów w stawie barkowym i łokciowym, dzięki czemu robot może gestykulować ruchami całych rąk.

Rys. 3.11: NAO z zaznaczonymi przegubami

korzystuje się jedną kamerę oraz czujniki dotyku umieszczone na dłoniach i barkach robota. Przetwarzanie danych odbywa się lokalnie - płyta główna NAO wyposażona jest w procesor ATOM Z530 1,6 GHz, 1GB RAM i 2GB pamięci flash. Dostęp do danych z czujników zapewniony jest przez lokalny system operacyjny NAOqi, zarządzający niskopoziomowym sterowaniem robota i udostępniający użytkownikowi obrobione dane w strukturach czytelnych dla człowieka.

3.4.2. Opis programu

Program napisano w języku Python 2.7 z wykorzystaniem frameworku graficznego Choregraphe i dostarczanych z nim gotowych metod API. Zaprojektowano go z myślą o zastosowaniu w ramach pomocy edukacyjnej dla dzieci w wieku przedszkolnym (roboty NAO są bowiem z powodzeniem wykorzystywane jako pomoce edukacyjne, temu zagadnieniu poświęcony jest projekt ASK NAO, prowadzony przez twórców robota). Dokładniej, robot miał pełnić rolę pomocnika w rozpoznawaniu stron. Choć zagadnienie wydaje się trywialne, rozróżnienie prawej od lewej jest niełatwym zadaniem dla wielu dzieci ze względu na zmienny układ odniesienia. Robot ma pomóc w zrozumieniu, że strony rozróżnia się w zależności od orientacji punktu odniesienia - zwykle samego siebie lub innej osoby. W tym celu zaprogramowano prostą grę.

Robot zaczyna od przywitania się. Słownemu „Cześć” towarzyszy przy tym gest machania ręką. Oczy robota mają w tym czasie kolor biały, uznany za najmniej nacechowany znaczeniowo. Pojedyncza czerwona dioda zapala się jako sygnał rozpoznania twarzy – jest to element wbudowany w metodę rozpoznawania twarzy i niemożliwy do zlikwidowania z poziomu API.

Następnie robot pokazuje strony w swojej orientacji - wysuwa przy tym odpowiednią rękę i zmienia kolor danego oka na zielony. Prosi gracza o dotknięcie czujników na prawej, a następnie lewej ręce. Jeżeli polecenie wykonano prawidłowo, robot potwierdza głosowo („Tak, to jest moja lewa/prawa ręka!"), wyciąga daną rękę do góry (zarówno jako potwierdzenie, jak i w geście zwycięstwa) a LEDy wokół oczu zmieniają kolor na zielony. Jeżeli strony zostały pomylone, robot wydaje buczący dźwięk, a kolor LEDów zmienia się na czerwony. Następuje powtórzenie prośby poparte prezentacją odpowiedniej strony. W razie dotknięcia innego czujnika lub braku jakiejkolwiek reakcji prośba zostaje powtórzona po 15 sekundach.

W kolejnym etapie uczestnik dostaje do ręki obiekt-znacznik łatwy do śledzenia przez robota, a NAO prosi o wyciągnięcie go w lewej lub prawej ręce. Kiedy robot jednoznacznie określi po której stronie znalazł się obiekt, następuje potwierdzenie lub zaprzeczenie analogiczne do pierwszej części.

Po pomyślnym zakończeniu gry robot sygnalizuje zadowolenie hasłem „Brawo!" i gestem klaskania. W razie niepowodzenia, NAO schyla głowę w geście smutku, a LEDy wokół oczu zmieniają kolor na niebieski.

3.4.3. Szczegóły implementacji

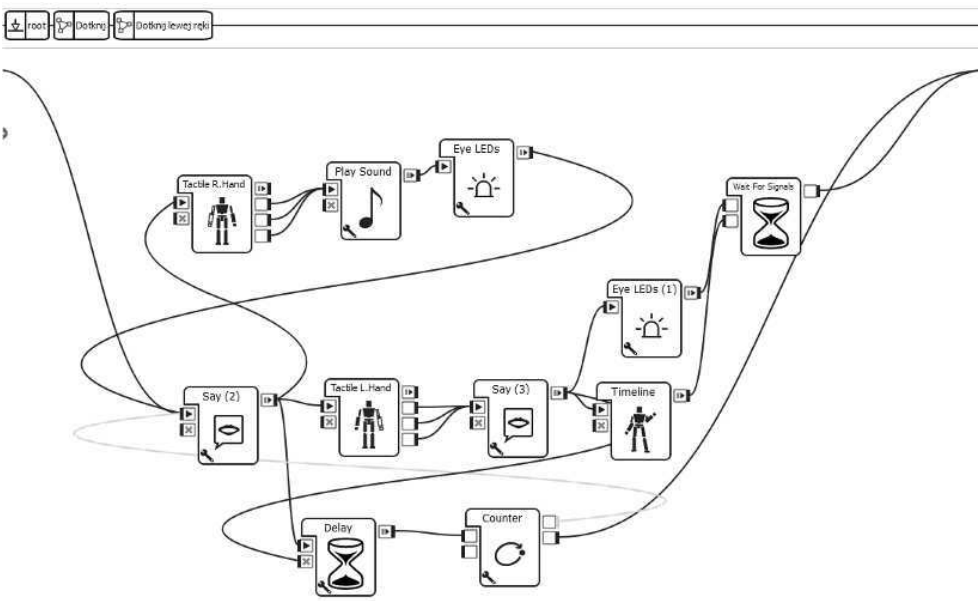
Pierwszym etapem tworzenia programu było zbadanie funkcjonowania dostępnych metod rozpoznawania elementów obrazu w celu ustalenia najlepszego sposobu określenia pozycji gracza. NAO jest wyposażony w funkcję automatycznego rozpoznawania obrazów zapisanych w bazie, tak więc najprostszym rozwiązaniem wydawało się zapamiętanie zestawu kilku do kilkunastu zdjęć osób prezentujących lewą lub prawą rękę. Niestety, w trakcie badań okazało się, że system NAOqi 2.1 zawiera błąd, wskutek którego baza obrazów jest zapisywana w niewłaściwej lokacji. Istnieje możliwość ręcznego przekopiowania bazy, lecz okazało się to niewskazane z dwóch względów: po pierwsze, prace prowadzono na robocie będącym własnością Politechniki Wrocławskiej i wszelkie ingerencje w jego wewnętrzne systemy wymagałyby zgody i nadzoru opiekuna robota; po drugie zaś, takie rozwiązanie utrudniłoby prace nad dalszym rozwojem programu oraz wykorzystanie go przez osoby postronne, będące jedynie użytkownikami, a nie programistami robota. Z tego względu przetestowano dwa inne rozwiązania: wykrywanie znaczników NAOmark oraz wykrywanie jednolitej plamy koloru, dostępnej jako moduł rozpoznawania czerwonej piłeczki.

NAOmark przypominają znak zagrożenia promieniowaniem, ale liczba i rozłożenie zapełnionych wycinków koła różnią się pomiędzy znacznikami, pozwalając na zdefiniowanie około czterdziestu rozpoznawalnych symboli. Moduł wykrywania NAOmark (ALLandmarkDetection) jest stosunkowo łatwy w użyciu ze względu na dobre udokumentowanie. Znaczniki są rozpoznawane poprawnie w zakresie od 14 do 160 pikseli średnicy (na obrazie QVGA), przy nachyleniu do 60^o od kamery i w niemal każdych warunkach oświetleniowych (pod warunkiem zachowania dobrego kontrastu obrazu). Jest więc możliwe rozpoznawanie postawy poprzez danie graczowi sztywnej flagi z nadrukowanym znacznikiem i rozpoznawanie jego położenia.

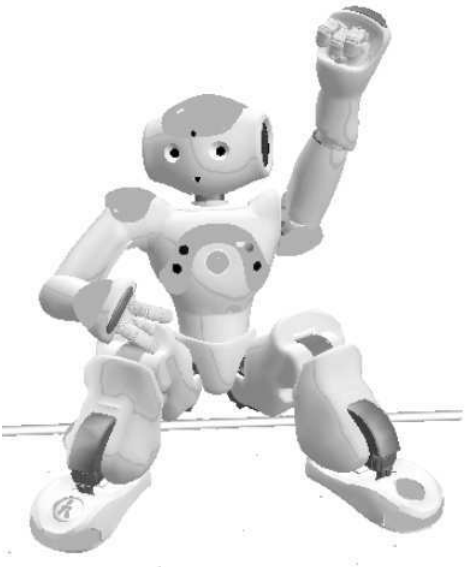
Moduł wykrywania piłki (ALRedBallDetection) również działa poprawnie, pozwalając na śledzenie obiektu nawet częściowo przysłoniętego dłonią w słabym oświetleniu, lecz ze względu na braki w dokumentacji jest znacznie cięższy w implementacji. Głównym problemem jest odczytanie danych o położeniu wykrytego obiektu. Pomimo trudności w implementacji posłużenie się czerwoną piłką wydaje się uzasadnione, ponieważ jest to zabawka znajoma dla niemal każdego dziecka, niegroźna (można wykorzystać miękką, piankową piłeczkę) i łatwa do zastąpienia w przypadku zgubienia lub zniszczenia. Dodatkową zaletą jest fakt, że można ją trzymać w dowolny sposób, podczas gdy NAOmark musi być obrócony przynajmniej częściowo przodem do robota.

Na podstawie przeprowadzonych testów zdecydowano się na wykorzystanie modułu rozpoznawania piłki.

Następnie przystąpiono do zaprojektowania przebiegu programu, odtwarzając w Choregraphe kolejne etapy: prezentację stron, rozpoznawanie stron w orientacji robota, rozpoznawanie stron w orientacji gracza (rys. 3.12). Po zaprojektowaniu przebiegu i obróbki sygnałów przystąpiono do głównego zadania, to jest zamodelowania komunikacji niewerbalnej. Wszystkie gesty robota zostały zaprojektowane metodą animacji – ręcznie ustawiano robota w żądanej pozycji, po czym zapamiętywano ją w programie (zobacz rysunek 3.13). Odtworzenie jednego gestu wymaga kilku do kilkunastu zapamiętanych klatek. Dokładna liczba zależy od długości gestu, jego skomplikowania i ryzyka kolizji z inną częścią ciała (trajektorie pomiędzy położeniami są wyznaczane automatycznie przez NAOqi z uwzględnieniem stanu początkowego robota i opisywane za pomocą krzywych Beziera). Przy projektowaniu ruchu należy zachować szczególną ostrożność ze



Rys. 3.12: Fragment programu w środowisku Choregraphe



Rys. 3.13: Wirtualny NAO z ręką wyciągniętą w geście zwycięstwa

względu na dynamikę, zarówno w celu uzyskania możliwie płynnego i naturalnego ruchu, jak i ze względu na stabilność robota. Dla bezpieczeństwa przez cały czas trwania programu robot siedzi w stabilnej pozycji, ale nagły ruch ręką – jak podczas gestu zwycięstwa – może przeważyć robota i w rezultacie przewrócić go.

Ruch robota został przetestowany na wirtualnym NAO przed uruchomieniem programu na rzeczywistym robocie; pozostałe części programu musiały być tworzone i testowane bezpośrednio na rzeczywistym robocie, ponieważ wymagały wprowadzania zewnętrznych danych (dotyk, znaczniki wizualne).

3.4.4. Badania

Ze względu na charakter projektu do testów zaproszono osoby dorosłe obojga płci: trzech neurotypowych mężczyzn w wieku 20-25 lat, jeden dwudziestoletni mężczyzna z zaburzeniami komunikacji niewerbalnej wywołanymi Zespołem Aspergera oraz czterdziestopięcioletnia kobieta z przygotowaniem z zakresu pedagogiki specjalnej. Wszyscy respondenci zostali poproszeni o przeprowadzenie gry z robotem, popełniając celowo co najmniej jeden błąd w celu poznania wszystkich zaprogramowanych reakcji. Następnie zostali poproszeni o krótką ocenę zachowania robota, swoją interpretację poszczególnych sekwencji oraz sugestie poprawy i dalszego rozwoju programu.

Intencje każdego z gestów zostały odczytane poprawnie, choć z drobnymi różnicami w interpretacji (jedna osoba uznała gest podniesienia ręki do góry za powtórzenie informacji, nie widząc w nim radości ani zadowolenia z otrzymanej odpowiedzi). Zmiana koloru oczu została odczytana poprawnie w sekwencjach zaprzeczenia i potwierdzenia, ale w dwóch przypadkach respondent nie zwrócił żadnej uwagi na wykorzystanie zmiany koloru pojedynczego oka na etapie pre-

zentacji stron. Uwagi krytyczne odnosiły się głównie do płynności gestów, które różnią się czasem trwania (np. machanie lewą ręką trwa dłużej niż prawą).

Planowane było przetestowanie programu z udziałem drugiej grupy badawczej, złożonej z dzieci w wieku wczesnoszkolnym. Ze względu na konieczność współpracy ze szkołą i zakłócenia zajęć lekcyjnych ten etap zaplanowano na końcówkę czerwca 2016 roku.

3.5. Podsumowanie i wnioski

Komunikacja niewerbalna nie tylko jest możliwa do przedstawienia w implementacji cyfrowej, ale wręcz stanowi niezbędne narzędzie i logiczny krok w tworzeniu zaawansowanych interfejsów użytkownika oraz, naturalnie, w oprogramowaniu robotów społecznych. Półświadome rozpoznawanie sygnałów komunikacji niewerbalnej stwarza niezwykłe możliwości błyskawicznej komunikacji, ograniczając przy tym do minimum konieczność skupienia się li tylko na przekazie, jaka występuje w przypadku komunikatów pisemnych.

Dodatkowym atutem stosowania komunikacji niewerbalnej w programach jest możliwość nawiązania swoistej więzi z użytkownikiem. Jest to cecha niewątpliwie pożądana w przypadku robotów społecznych, których zadaniem – jak w przypadku NAO – jest docelowo stać się towarzyszem człowieka, spędzającym z nim długi czas. Taki robot musi umieć przekazywać i rozumieć informacje przekazywane przez ludzi naturalnie, mimochodem – a więc właśnie komunikację niewerbalną. Dobra implementacja komunikacji niewerbalnej sprawia, że robot wydaje się „żywy", przez co użytkownikowi łatwiej nawiązać z nim swobodny kontakt (tego typu zagadnienia z wykorzystaniem NAO są omawiane np. pod kątem wykorzystania robotów w opiece nad osobami starszymi [8]).

W interfejsach programów komputerowych można stosować różne podejścia do komunikacji niewerbalnej, zależnie od występowania lub braku wirtualnego agenta, poprzez którego można przedstawiać mimikę i gesty. Implementacja komunikacji opartej na wzorcach międzyludzkich jest obecnie na etapie badań [3], lecz jej proste wersje z powodzeniem funkcjonują np. w formie aplikacji wirtualnych towarzyszy, dostępnych na smartfony i komputery osobiste. W każdym programie graficznym stosuje się natomiast przekazy oparte o symbolikę ruchu i koloru.

Cyfrowa implementacja rozumienia i wysyłania komunikatów niewerbalnych jest bardzo obszernym tematem, zasługującym na duże zainteresowanie i dogłębne badania ze względu na szeroki wachlarz potencjalnych zastosowań.

Literatura

[1] A. G. Brooks, R. C. Arkin. Behavioral overlays for non-verbal communication expression on a humanoid robot. *Autonomous Robots*, 2007.

[2] J. Dias, S. Mascarenhas, A. Paiva. *FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework*, strony 44–56. Springer International Publishing, Cham, 2014.

[3] B. Hartmann, M. Mancini, C. Pelachaud. Implementing expressive gesture synthesis for embodied conversational agents. *Lecture Notes in Computer Science*, 3881:188–199, 2005.

[4] M. Pantic, L. Rothkrantz, H. Koppelaar. Automation of non-verbal communication of facial expressions. *Euromedia'98*, 1998.

[5] M. R. Power. *Working through communication*, rozdzia/l 11, Non-Verbal Communication. Bond University, 1998.

[6] B. Salem, N. Earle. Designing a non-verbal language for expressive avatars. *Proceedings of the third international conference on Collaborative virtual environments*, strony 93–101, 2000.

[7] T. Shibata, M. Youshida, J. Yamato. Artificial emotional creature for human-machine interaction. *IEEE International Conference on Systems, Man and Cybernetics*, strony 2269 – 2274. IEEE, October 1997.

[8] E. Torta, F. Werner, D. Johnson, J. Juola, R. Cuijpers, M. Bazzani, J. Oberzaucher, J. Lemberger, H. Lewy, J. Bregman. Evaluation of a small socially-assistive humanoid robot in intelligent homes for the care of the elderly. *Journal of Intelligent And Robotic Systems*, 76:55–71, 2014.

[9] J. D. Velásquez. Modeling Emotions and Other Motivations in Synthetic Agents. *IN: PROCEEDINGS OF AAAI97*, 1997.

ROZDZIAŁ 4

AUTOMATYCZNA OCENA ŁADUNKU EMOCJONALNEGO TEKSTU W SYSTEMIE TYPU HELPDESK

P. Joniak, B. Witkowski

W niniejszym rozdziale skupiono się na problemie rozpoznawania emocji zawartych w tekście pisanym oraz wykorzystaniu tak pozyskanych informacji w systemie typu helpdesk.

4.1. Wstęp

W dobie powszechnego Internet natychmiastowy dostępem do olbrzymiej ilości informacji stał się swego rodzaju standardem. Jednak mnogość prezentowanych treści rodzi często problemy z szybką oceną ich jakości. Stąd też duże zapotrzebowanie na metody filtracji danych według wybranych parametrów. Parametrami tymi mogą być między innymi emocje. Rozpoznanie emocji pełni bowiem ważną rolę w komunikacji człowiek – komputer. Stanowi też szeroki dział sztucznej inteligencji w obrębie przetwarzania języka naturalnego, gdzie w ostatnich latach włożono wiele wysiłku w rozpoznawanie emocji na podstawie wyglądu twarzy, tonu wypowiedzi czy zapisanego tekstu.

Zastosowanie emocji do filtracji informacji wymaga wydzielenia emocji podstawowych. W. Gerrod Parrott zaproponował następujący ich zbiór: miłość, radość, gniew, smutek, strach, zaskoczenie. Jednak zdecydowanie częściej w literaturze występuje model Ekmana [6], w którego składa wchodzą: gniew, wstręt, strach, radość, smutek, zaskoczenie. W niniejszej pracy wykorzystywano właśnie ten model.

Ekstrakcji emocji z tekstu może pozwolić, przykładowo, na:

- **wartościowania prac naukowych** – dobra praca naukowa powinna być napisana językiem stosunkowo neutralnym. Niepożądanym jest, by osobiste emocje autora wpływały prezentowaną treść;
- **wykrywanie zaburzeń emocjonalnych** – autonomiczny bot mógłby przeglądać fora internetowa w poszukiwaniu potencjalnych samobójców;

- **wspieranie operatorów systemów typu heldesk** – analiza treści otrzymywanej od klienta może pozwolić na ustalenia priorytetu obsługi zgłoszenia lub też wybranie odpowiedniej strategii postępowania. Dzięki rozpoznaniu emocji zadanie obsługi trudnego klienta może zostać automatycznie przekazane do osoby z odpowiednimi umiejętnościami i doświadczeniem. Co wiecej, system może sam właściwie zareagować na wiadomości zbyt wulgarne lub nacechowane innymi negatywnymi emocjami.

W dalszej części pracy przedstawiono prototypową implementację rozwiązania pozwalającego obsłużyć ostatni z wymienionych wyżej przypadków.

4.1.1. Metody rozpoznawania emocji

Tworzenie systemów rozpoznawania emocji może odbywać się na wiele sposobów i z wykorzystaniem wielu technik. Poniżej przedstawiono kilka z nich.

- **Wykrywanie słów kluczowych ze słownika** – algorytmy tego typu operują na poziomie pojedynczych słów [5]. Tak prosty model nie radzi sobie jednak z tekstami, w których emocje są wyrażane w sposób niebezpośredni.
- **Podejście bazujące na regułach językowych** – algorytmu tego typu oprócz pojedynczych słów biorą pod uwagę różne zależności.
  - **Podejście oparte na regułach z wpływem słowników** – przykładem są systemy ESNA oraz UPAR7 służące do klasyfikowania emocji zawartych w nagłówkach prasowych. W UPAR7 wykorzystuje się ręcznie wprowadzane słowa do sieci słów powiązanych z emocjami. System identyfikuje główny temat rozpatrywanego tekstu i wzmacnia ocenę emocji bazując na grafach zależności. Efekty działania systemów analizowano z wykorzystaniem WordNetu czy General Inquirera.
  - **Podejście oparte na regułach bez wpływu słowników** – charakteryzuje je próba zrozumienia semantyki leżącej u podstaw języka lub rozpoznawanie informacji zawartych nie wprost. Stosowane tu metody zapewniają elastyczność dla dowolnego zestawu emocji, zaś stosowane zasady są specyficzne względem reprezentacji źródła danych.
- **Zastosowanie uczenia maszynowego** – wykorzystywane tu algorytmy pozwalają ominąć ograniczenia metod wymienionych wcześniej.
  - **Nadzorowane maszynowe uczenie z wpływem słowników** – jedną z pierwszych prób wykorzystania tego podejścia podjęto na Uniwersytecie Illinois w Urbana – Champaign. Użyto tam hierarchiczno-sekwencyjnego modelu połączonego z bazą SentiWordNet. Treści pobrane z internetowych blogów sklasyfikowano wykorzystując maszynę wektorów nośnych (ang. *Support Vector Machine*, SVM). Pomimo wielu pozytywnych cech wadą tego rozwiązania jest konieczność dostarczenia dużej ilości danych uczących oraz ograniczenie do wyuczonej dziedziny.
  - **Nadzorowane maszynowe uczenie bez wpływu słowników** – do tej grupy zalicza się rozwiązania wykorzystujące algorytmy działające w oparciu o maszynę wektorów nośnych.

- **Nienadzorowane maszynowe uczenie z wpływem słowników** – przykładami tego typu rozwiązań są rozwiązania przetestowane na Uniwersytecie w Sydney. Próbowano tam użyć WordNet oraz wykorzystać model wektorowy (ang. *Vector Space Model*) i metody redukcji wymiarowości. Klasyfikowano nagłówki prasowe używając prostej heurystyki oraz bardziej złożonych algorytmów (np. podobieństw w ukrytej przestrzeni semantycznej).
- **Nienadzorowane maszynowe uczenie bez wpływu słowników** – do bardziej interesujących algorytmów z tej grupy należą algorytmy oparte o analizę ukrytych grup semantycznych.

4.2. System wspierający operatorów helpdesku

Analiza zawartości emocjonalnej tekstu pisanego może pozwolić na częściowe zautomatyzowanie procesu dystrybucji zadań wśród pracowników działu obsługi klienta (helpdesku). W przypadku wykrycia zgłoszenia silnie nacechowanych emocjami powinno nastąpić automatyczne lub półautomatyczne przekierowanie jego obsługi do pracowników posiadających odpowiednie kompetencje.

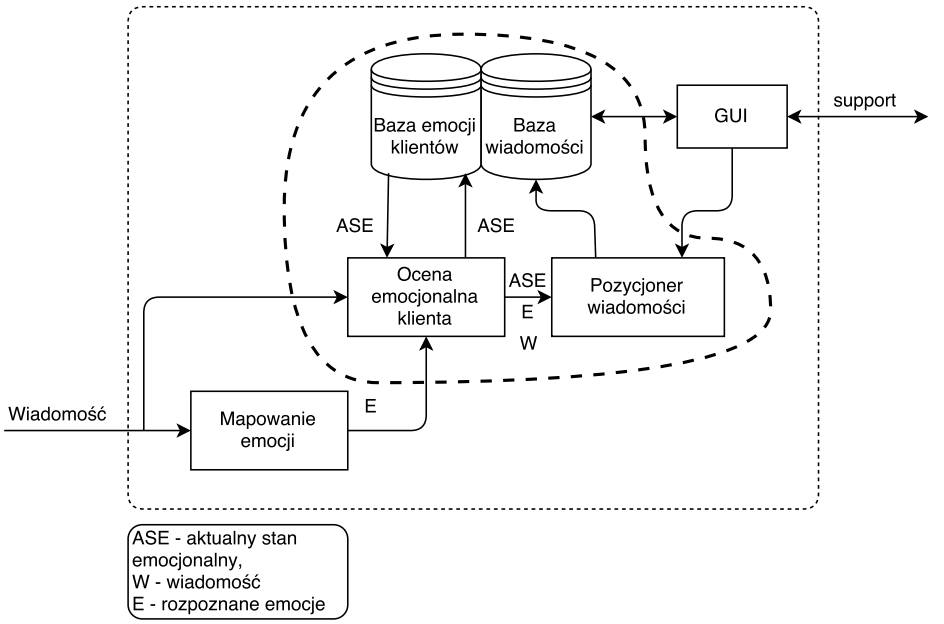
Przykładowo, zgłoszenie nacechowane pozytywie mogłoby zostać przekazane początkującemu pracownikowi, o zbyt małym doświadczeniu by poradzić sobie z niezadowolonym klientem. Zgłoszenia o negatywnym ładunku mogłyby zostać przekazane do pracownika z większym stażem. Tego typu system mógłby znacznie zmniejszyć czas potrzebny na obsługę zgłoszeń. Mógłby także podnieść jakość pracy w całym dziale. Dodatkowo zapewniałby początkującym pracownikom łagodniejsze wdrożenie się na wyznaczonym stanowisku.

4.2.1. Architektura systemu

W systemie wydzielić można trzy podsystemy:

- **Podsystem odpowiedzialny za rozpoznawanie emocji**. Wejściem do tego podsystemu ma być treść wiadomości, wyjściem natomiast wektor rozpoznanego stanu emocjonalnego składający się z sześciu wartości (odpowiadających podstawowym emocjom wymienionym we wstępie). Na rysunku 4.1 temu podsystemowi odpowiada blok „Mapowanie emocji". Implementacja algorytmów wykorzystywanych w tym podsystemem oraz ich testowanie stała się głównym tematem opisywanych prac.
- **Podsystem odpowiedzialny za ocenę stanu emocjonalnego klienta oraz pozycjonowanie wiadomości w bazie danych**. Na rysunku 4.1 podsystem ten został otoczony linią przerywaną. Korzysta on z bieżących danych przekazanych z pierwszego podsystemu, a także odwołuje się do bazy danych gromadzącej historię ocen stanów emocjonalnych klientów. Korzystając z dostępnych danych dokonuje ewaluacji nastawienia klienta i na tej podstawie pozycjonuje wiadomość w bazie danych.
- **Graficzny interfejs użytkownika**. Ten podsystem jest odpowiedzialny za dostarczenie użytkownikowi (pracownikowi działu obsługi klienta) narzędzi pozwalających na komfortowe korzystanie z systemu. Ma umożliwiać odpowia-

danie na zgłoszenia oraz przeglądanie historii konwersacji z oceną stanów emocjonalnych przypisanych do konkretnego klienta.



Rys. 4.1: Zarys architektury systemu

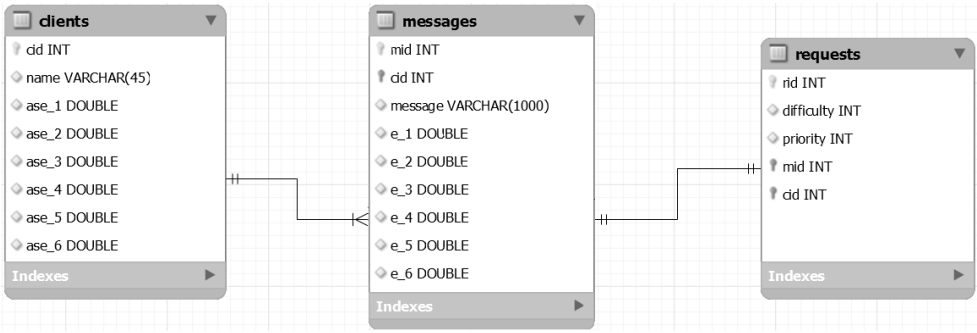
4.2.2. Implementacja prototypu systemu

Prototyp systemu został napisany w języku Python. Podczas implementacji wykorzystano bibliotekę flask (`http://flask.pocoo.org/`). Z jej pomocą stworzono serwer oraz interfejs webowy. Zgłoszenia (wiadomości) są redagowane przez użytkowników na stronie internetowej, skąd wysyłane są do obsługi helpdesku. Obsługa helpdesk-u usyskuje dostęp do nadesłanych wiadomości również poprzez stronę internetową.

W skład systemu wchodzą także moduły zawierające zaimplementowane algorytmy oraz służące do komunikacji z bazą danych (odpowiednio do modułów przedstawionych z zarysie architektury). Jako system zarządzania bazą danych wykorzystano MySQL (`https://www.mysql.com/`). Schemat samej bazy danych przedstawiono na rysunku 4.2. W schemacie tym wyróżnić można następujące tabele:

- `messages` – zawiera informacje o wiadomościach oraz wyliczone dla nich informacje o emocjach,
- `clients` – zawiera informacje o klientach i ich stanie emocjonalnym,
- `request` – zawiera nieprzeczytane wiadomości oraz ich trudność i priorytet.

Do szybkiego pisania i testowania algorytmów użyto dystrybucji Anaconda i notatników Jupyter. Najważniejszą biblioteką wykorzystaną w algorytmach ana-



Rys. 4.2: Schemat bazy danych

lizy tekstu jest `nltk` (ang. *Natural Language Toolkit*, `http://www.nltk.org/`) [3]. Udostępnia ona wiele przydatnych funkcji, w tym: tokenizację, identyfikacja części zdania, klasyfikatory (naiwny Bayes). Nltk umożliwia także łatwy dostęp do bazy WordNet. Jest to jedna z największych i najbardziej rozwiniętych leksykalnych baz danych języka angielskiego. Daje, między innymi, możliwość rozpoznawania części mowy oraz zawiera wiele gotowych korpusów. Z tego powodu jest używana w wielu algorytmach rozpoznawania emocji [4].

4.3. Zaimplementowany algorytm rozpoznawania emocji

W pracy [1] przedstawiono algorytm nienadzorowanej detekcji emocji. Poniżej zostanie pokrótce omówiona zasada działania algorytmu.

Niech $w = \{w_1, w_2, ..., w_n\}$ będzie zbiorem słów NAVA (noun, adjective, verb, adverb) w zdaniu s, gdzie $w \subset s$. Niech $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ będzie zbiorem c słów znaczących w s. Niech $\beta = \{\beta_1, \beta_2, ..., \beta_d\}$ będzie zestawem d zależnych słów w zdaniu s. Jak łatwo zauważyć $\alpha \subset w$ i $\beta \subset w$. Niech $e = \{e_1, e_2, ..., e_m\}$ będzie wektorem m stanów emocjonalnych, według których można sklasyfikować zdanie. W tym wypadku wykorzystywany jest model emocji według Ekmana, zatem $e =$ { radość, smutek, złość, strach, zaskoczenie, wstręt}.

Stosunkowo prostym sposobem dostarczenia wektora emocji jest wyznaczenie dla słów NAVA współczynnika korelacji (ang. *Pointwise Mutual Information*, PMI) między każdym z nich i słowem reprezentującym koncepcję emocji. PMI jest współczynnikiem semantycznego powiązania i podobieństwa pomiędzy dwoma wyrażeniami wyznaczanym na podstawie prawdopodobieństwa ich wspólnego wystąpienia:

$$PMI(x, y) = \frac{wspólne\ wystąpienie(x, y)}{wystąpienia(x) \cdot wystąpienia(y)}$$

gdzie *wspólne wystąpienie*(x, y) to prawdopodobieństwo wystąpienia słów x i y wewnątrz zdefiniowanego okna (w tym wypadku długości ośmiu słów) w korpusie (wzorcowym tekście), a *wystąpienie*(x) to liczba wystąpień słowa x w korpusie (i odpowiednie - wystąpień słowa y). W celu wyszukiwania nie tylko identycznych słów, ale też słów silnie powiązanych (np. liczby pojedynczej i mnogiej, czy

odmiany) można tutaj wykorzystać bazę WordNet. Jednakże emocje mogą być wyrażane poprzez wiele różnych słów. Dlatego jako wzorzec nie jest wykorzystywane pojedyncze słowo, a zbiór wyrazów wyrażających daną emocję. Słowa w zbiorach wzorcowych zostały wybrane na podstawie najczęściej używanych synonimów wybranych z tezaurusa.

Podstawą działania algorytmu jest założenie, iż słowo wyrażające pewną emocję będzie silnie powiązane z większością słów wzorcowych wyrażających tę emocję. Zredukuje to możliwe błędy wynikające z mylnego powiązania z pojedynczym słowem wzorcowym, które mogłoby skutkować nieprawidłowym identyfikowaniem emocji.

PMI dostarcza powiązania danego słowa w_i z każdym ze wzorcowych słów, co jest wykorzystywane do wyznaczenia powiązania w_i z grupą emocji. Niech K_j będzie zestawem r reprezentatywnych słów dla emocji e_j. Wynik semantycznego powiązania pomiędzy słowem w_i i kategorii emocji jest obliczany następująco:

$$PMI(w_i, e) = \sqrt[r]{\prod_{g=1}^{r} PMI(w_i, K_j^g)}$$

gdzie K_j^g jest g-tym słowem w K_j. Wyjściowym wektorem emocji ma następującą postać:

$$\sigma_{w_i} = < PMI(w_i, e_1), PMI(w_i, e_2), ..., PMI(w_i, e_m) >$$

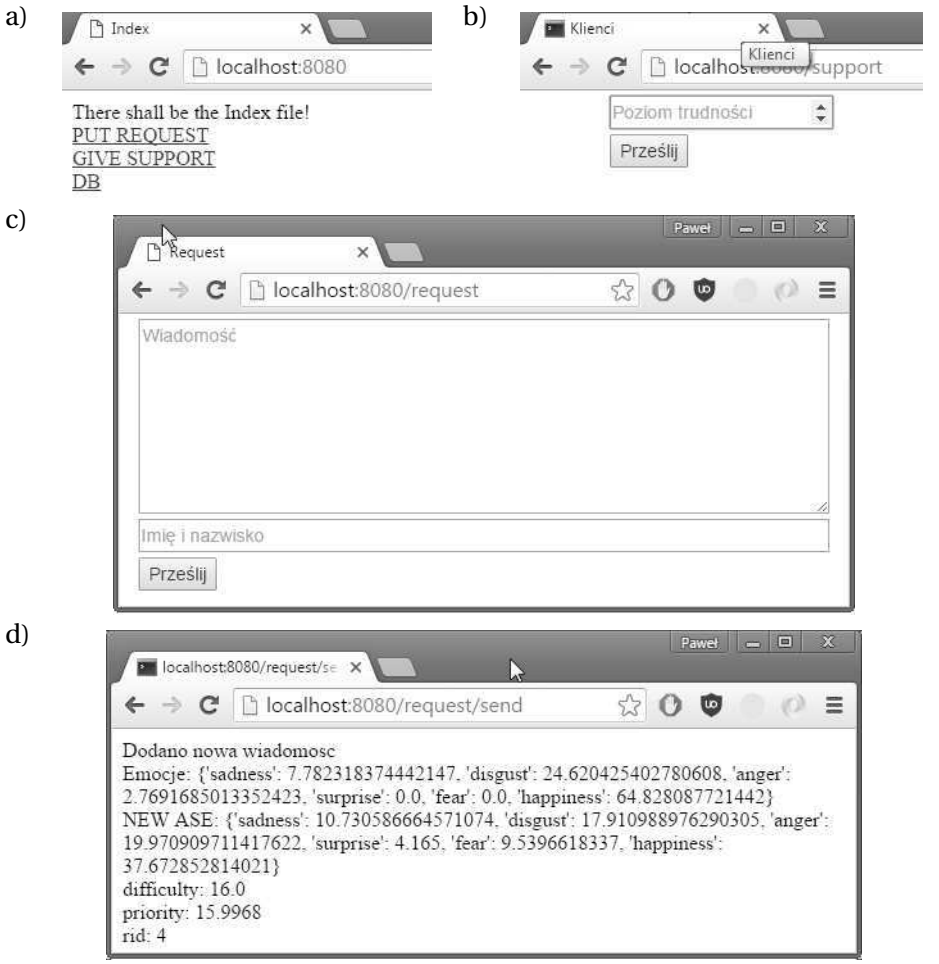
Jakość działania przedstawionego powyżej algorytmu w dużej mierze zależy od wyboru odpowiedniego korpusu. Musi być on reprezentatywną próbką danego języka. Wskazane jest, aby znalazły się w nim teksty nacechowane emocjonalnie. Ważne jest także, by nie był on zbyt duży, gdyż odbija się to negatywnie na czasie działania algorytmu. Zaletą jest niewątpliwie fakt, iż złożoność obliczeniowa algorytmu jest liniowa ze względu na długość korpusu. W opisywnych badaniach wykorzystano angielski korpus OANC[2] (ang. *The Open American National Corpus*).

4.4. Działanie prototypu

Zaimplementowany system dostarcza podstawowych narzędzia do obsługi systemu helpdesk (rys. 4.3). Za jego pomocą można stworzyć nowe zgłoszenie oraz wyszukać zgłoszenia na podstawie zadanego poziomu trudności. Możliwa jest też rozbudowa systemu np. o formularze ułatwiające przeglądanie bazy wiadomości. Obecnie jednak zaimplementowano jedynie minimalną funkcjonalność pozwalającą zaprezentować koncepcję całego systemu oraz ocenić jakość zaimplementowanego algorytmu rozpoznawania emocji i oceny poziomu trudności odpowiedzi na wiadomość.

System przetestowano na kilku przykładowych wiadomościach. Pierwszą wiadomością był fragment romansu „This Man" autorstwa Jodi Ellen Malpas:

„I stand there in utter shock. How dare he come here and start shouting the odds at me. I feel my unease disappear and my earlier irritation convert



Rys. 4.3: Elementy interefejsu użytkownika stworzonego prototypu: a) główna strona systemu, b) interfejs dla pracownika pozwalający na wybranie wiadomości o zadanym poziomie trudności, c) formularz wprowadzania nowej wiadomości, d) przykład danych zwracanych przez program

.

into boiling rage. The urgent need to defend myself, to put him straight, has my jaw clenched to aching point. His opinion of me is very low if he thinks I'll just jump into bed with any man I meet. But then, I don't have to answer to him. The fact that he has a girlfriend is immaterial at this point. He thinks he can just take what he wants or throw a wobbly if he meets some resistance."

Wynik działania algorytmu był następujący:

```
sadness 19.9484197871
disgust 40.185617504
```

```
anger 12.4346349032
surprise 0.995502095502
fear 8.96403035343
happiness 17.4717953567
```

Następnie algorytm przetestowano na fragmencie tekstu piosenki "What is love":

„What is love?
Baby don't hurt me,
Don't hurt me, no more
Baby don't hurt me,
Don't hurt me, no more
What is love?
yeeeahh
Oh I don't know
Why you're not there
I give you my love, but you don't care
So what is right,
And what is wrong
Gimme a sign!"

W wyniku analizy otrzymano:

```
sadness 53.4567147795
disgust 14.0233218704
anger 5.26969930572
surprise 0.636479956598
fear 4.14660415011
happiness 22.4671799377
```

Wyniki tych testów pokazały, że program poprawnie rozpoznaje natężenie emocjonalne przekazywanych mu wiadomości.

Literatura

[1] A. Agrawal, A. An. Unsupervised emotion detection from text using semantic and syntactic relations. *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '12, strony 346–353, Washington, DC, USA, 2012. IEEE Computer Society.

[2] American National Corpus Project. The Open American National Corpus. `http://www.anc.org/` [dostęp dnia 20 czerwca 2016].

[3] S. Bird, E. Klein, E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., wydanie 1st, 2009.

[4] R. Rajesh, K. Ganesh, S. C. L. Koh, R. Ezhilarasi, R. Minu. International conference on modelling optimization and computing automatic emotion recognition and classification. *Procedia Engineering*, 38:21 – 26, 2012.

[5] S. N. Shivhare, S. Khethawat. Emotion detection from text. *Data Mining and Knowledge Management Process (DKMP 2012)*, New Delhi, Maj 2012.

[6] C. Strapparava, R. Mihalcea. Learning to identify emotions in text. *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, strony 1556–1560, New York, NY, USA, 2008. ACM.

ROZDZIAŁ 5

ZARZĄDZANIE SŁOWNIKIEM POJĘĆ BIZNESOWYCH

H. Preiss, P. Zieliński

W rozdziale omówiono zagadnienia związane z zarządzaniem wiedzą oraz słownikami pojęć biznesowych. Opisano także implementację aplikacji webowej umożliwiającej zarządzanie słownikiem pojęć biznesowych i jego przeglądanie.

5.1. Wstęp

Zarządzanie wiedzą (ang. *Knowledge management*) kojarzone jest z multidyscyplinarnym podejściem do problemu gromadzenia i przetwarzania wiedzy formalnej i ukrytej, mającym na celu jak najefektywniejsze i najpełniejsze jej wykorzystanie [4]. Cel ten jest osiągany poprzez zastosowanie odpowiednich metod zdobywania, przechowywania oraz współdzielenia wiedzy. Dużą w nim rolę odgrywa również zastosowanie odpowiednich zwyczajów i rozwiązań technicznych.

Wiedza, dane, informacja i mądrość – to pojęcia, które wydają się mieć dość intuicyjną interpretację. Jednak zdefiniowanie różnicujących je cech oraz określenie ich zasięgu znaczeniowego bywa problematyczne. Zwykle pojęcia te przedstawia się w postaci piramidy obrazującej ich nadrzędność czy też podrzędność (rys. 5.1). Mimo prostej graficznej reprezentacji schemat ten nie rozwiewa wszystkich wątpliwości. Wyjaśniając to na rzeczywistym przykładzie: kierowca otrzymuje z zewnętrznego systemu dane o kolorze światła na skrzyżowaniu oraz lokalizację tego sygnalizatora. Jego tor przetwarzania dane → informacja → wiedza → mądrość może wyglądać następująco:

- Dane - `Red, 192.234.235.245.678, v2.0` - nie zawierają kontekstu.
- Informacja - *W kierunku zachodnim na skrzyżowaniu Prusa i Wyszyńskiego zapaliło się czerwone światło* - nadaje znaczenie danym.
- Wiedza - *Jadę w stronę czerwonego światła* - specyfikuje wnioski wynikające z umiejętnego połączenia wielu informacji.
- Mądrość - *Powinienem zatrzymać samochód!* - wynik zastosowania wiedzy by osiągnąć cel lub uniknąć porażki.

Rys. 5.1: Porównanie znaczeń mądrości, wiedzy, informacji oraz danych (wg `http://www.allthingy.com/data-information-knowledge-wisdom/`)

Mądrość często utożsamiana jest z wiedzą, rozumianą jako zdolność do wykorzystania informacji w sposób trudny do jednoznacznego określenia. Wiedzę zaś dzieli się na dwie kategorie: wiedzę formalną (ang. *explicit knowledge*) oraz wiedzę ukrytą (ang. *tacit knowledge*). **Wiedza formalna** zawarta jest między innymi w dokumentach, bazach danych, instrukcjach obsługi. Do tej grupy zaliczają się na przykład zalecenia odnośnie resuscytacji krążeniowo-oddechowej („Wykonaj 30 uciśnięć klatki piersiowej na głębokość 5 centrymetrów w tempie 100 ucisków na minutę"). Ten typ wiedzy jest stosunkowo łatwy do przechowywania i współdzielenia. Znacznie trudniejsza do uchwycenia jest **wiedza ukryta**, intuicyjna, przeważnie wynikająca z doświadczenia. To ona pozwala ratownikowi medycznemu określić poprawność wykonywanej resuscytacji, jakie szanse przeżycia ma pacjenta oraz jak zmaksymalizować te szanse w danej sytuacji. Właśnie wiedza ukryta jest najistotniejsza i na jej utrwalaniu oraz współdzieleniu najbardziej powinno zależeć osobom odpowiedzialnym za zarządzanie wiedzą.

Zarządzanie wiedzą obejmuje między innymi takie obszary, jak (rys. 5.2, [2]):

- Strategia zarządzania wiedzą - baza wiedzy dopasowanej do danego zespołu.
- Organizacja kultury - kontakty międzyludzkie.
- Organizacja procesów - procesy przetwarzania wiedzy.
- Zarządzanie - liderzy nadzorujący wymianę wiedzy.
- Technologia - zaimplementowane narzędzia.
- Polityka korporacji - długoterminowy plan.

Wystąpienie jakichś zawirować na którymś z nich może pociągnąć za sobą porażkę przy próbach osiągnięciu postawionego celu. Zły zakres bazy wiedzy, oporność na naukę, zawodne narzędzia – wszystko to może zaprzepaścić starania całej grupy ludzi. Również brak dalekosiężnego planu może być tragiczny w skutkach. Gdy na emerytuję odejdzie pracownik z wieloletnim stażem, który od początku swego zatrudnienia rozwiązywał problemy w firmie i zawsze można było na niego

Rys. 5.2: Aspekty zarządzania wiedzą

liczyć, firma może pozostać bez jedynej osoby zdolnej do nadzorowania/wpierania przebiegu procesów krytycznych dla jej działania.

Często zarządzanie wiedzą nie przynosi bezpośrednio widocznego zysku pomimo wysokich kosztów oraz długiego czasu inwestowania. Jednakże, po pewnym okresie lub w sytuacjach awaryjnych może się okazać strategią ratującą firmę przed upadkiem.

5.2. Słownik pojęć biznesowych

Skuteczne zarządzanie wiedzą jest ściśle zależne od prawidłowej komunikacji między ludźmi [3]. Popularnym jest stwierdzenie, iż słowa stanowią zaledwie kilka procent informacji przekazywanych pomiędzy uczestnikami konwersacji – reszta oparta jest o mimikę twarzy oraz gestykulację. Dlatego do efektywnego przekazywania informacji w formie pisanej niezbędne jest posługiwanie się zbiorem odpowiednio dobranych słów oraz jednakowe rozumienie pojęć pojawiających się tekście.

Częstym problemem w biznesie i współpracy międzyludzkiej są nieporozumienia wynikłe z rozbieżnego rozumienia np. założeń projektowych lub oczekiwań klienta, co przekłada się pośrednio na uruchamianie niewłaściwych ścieżek opracowania wynikowego produktu oraz nieusatysfakcjonowanie klienta. Nawet w tak prozaicznej sytuacji jak wizyta u fryzjera rozbieżne rozumienie określenia „na krótko” może sprawić, że w przeciągu kilka minut może dojść do zaprzepaszczenia kilkutygodniowych przygotowań, i co więcej, przez kilka kolejnych tygodni

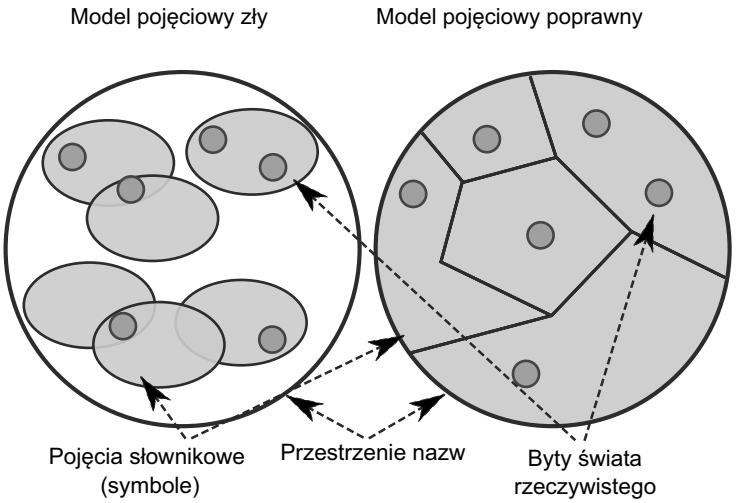
Rys. 5.3: Skutki różnego rozumienia pewnych pojęć (źródło: `http://www.pleated-jeans.com/2012/04/05/just-a-little-off-the-top/`)

klient może być niezadowolony ze swojego wyglądu. Ilustrację tego przypadku przedstawiono na rysunku 5.3. Zdarza się, że równie drobne nieporozumienia powodują w biznesie nawet wielomilionowe straty.

Sposobem na uniknięcie tego typu porażek jest stworzenie słownika stosowanych pojęć należących do odpowiedniej przestrzeni nazw (ang. *name space*) już na etapie planowania projektu. Przykład nieprawidłowego oraz prawidłowego modelu pojęciowego pokazano na rysunku 5.4 [1]. Z lewej strony nie cała przestrzeń nazw jest pokryta przez wyrażenia słownikowe, część przestrzeni określana jest przez więcej niż jedno wyrażenie i występują byty rzeczywiste, które nie są opisane w ogóle. Model z prawej strony nie zawiera żadnej z powyższych wad i jest zrealizowany wzorowo.

Do wymagań stawianych słownikom pojęć biznesowych oraz aplikacjom umożliwiającym ich przeglądanie zalicza się:

- Uwzględnienie istotnych pojęć (w większości przypadków wystarczy zdefiniować kilkadziesiąt pojęć).



Rys. 5.4: Porównanie niepoprawnej i prawidłowej przestrzeni nazw

- Obsługę powiązań z przyjętymi definicjami i innymi słownikami (tworzone referencje powinny dotyczyć np. konkretnego wydania słownika języka polskiego).
- Możliwość odpowiedniego nadzoru (edytorami pojęć, ich czytelnikami czy też użytkownikami mogącymi przeglądać historie zmian powinny być osoby z odpowiednimi uprawnieniami).

5.2.1. Modele słowników pojęć biznesowych

Jednym z bardziej znanych modeli informacyjnych słowników pojęć biznesowych (oraz innych systemów organizacji wiedzy) jest model opisany w specyfikacji SKO (ang. *Simple Knowledge Organization System*). Specyfikacja ta została stworzona przez konsorcjum W3C z myślą o budowie oraz integracji różnych słowników dających się przetwarzać komputerowo oraz narzędzi do zarządzania wiedzą. Zastosowanie tej specyfikacji przy tworzeniu metadanych pozwala na realizację ideę semantycznego Internetu (ang. *Semantic Web*), automatyczne powiązanie danych ze sobą oraz ułatwienia publikowanie i konsumowanie treści dzięki wprowadzonej standaryzacji. Konsorcjum W3C jest odpowiedzialne również za specyfikację RDF (ang. *Resource Description Framework*), czyli język opisu zasobów internetowych przy użyciu odpowiednich metadanych.

Słownik utworzony w standardzie SKOS posiada następujące cechy:

- Głównymi modelowanymi elementami są koncepty (ang. *concept*) - abstrakcje opisujące elementy rzeczywistości i niezależne od terminów, które je określają.
- Dany koncept jest powiązany z językiem poprzez etykiety (ang. *labels*) – zakłada się, że istnieje jedna preferowana etykieta służąca jako najprecyzyjniejsze określenie w danym języku oddające sens konceptu, jak również może istnieć wiele alternatywnych etykiet (wyrazów bliskoznacznych) powiązanych (często

ukrywanych na interfejsie użytkownika, służących do wyszukiwania, mogących reprezentować słowa z celowo wprowadzoną błędną pisownią).
- W modelu można definiować mapowanie – dzięki mapowaniu daje się powiązać dany koncept z innymi konceptami (jako wyrażenia powiązane, szersze lub węższe itd.). Umożliwia to przypisanie kategorii lub innych pojęć stosowanych z danym konceptem, w tym również pojęć istniejących w innych słownikach.

Przykładem słownika, który został zbudowany w oparciu o model informacyjny SKOS jest *EuroVoc* – interdyscyplinarny, ogólnodostępny tezaurus w ponad 20 językach Unii Europejskiej. Na rysunku 5.5 zaprezentowano widok przykładowej strony słownika EuroVoc, na której zebrane są pojęcia odpowiadające etykiecie „fryzjerstwo i kosmetyka". Zamieszczono tu: kategorię oraz podkategorie wraz z ich identyfikatorami, URI (ang. *Uniform Resource Identifier*), referencje do powiązanego wyrażenia w słowniku ECLAS oraz tłumaczenia pojęcia na wszystkie pozostałe języki obsługiwane przez EuroVoc.

Rys. 5.5: Przykładowa strona słownika EuroVoc dla hasła „fryzjerstwo i kosmetyka"

5.3. Stworzona aplikacja

W wielu praktycznych zastosowaniach słowniki powstają na bazie bardzo prostego modelu, w którym dane słowo otrzymuje swój własny identyfikator a do obsługi procesów biznesowych tworzy się w bazie danych schemat uwzględniający różne uprawnienia i role. Właśnie takie praktyczne podejście zostało zastosowane w implementacji opisanej dalej webowej aplikacji.

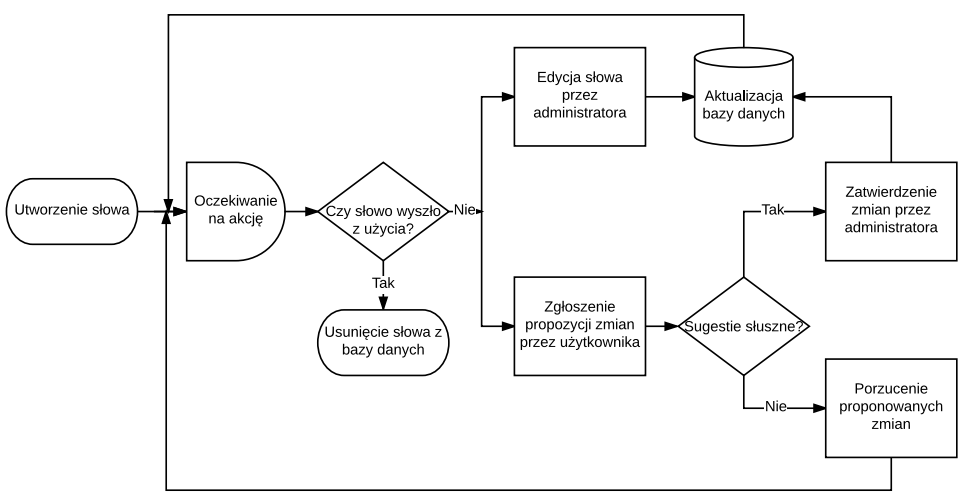
Do stworzenia aplikacji wykorzystano framework *JHipster* co w znacznym stopniu ułatwiło budowę interface'u użytkownika. Oprócz tego frameworka stos wykorzystanych technologii składał się z: platformy *Java*, języka *JavaScript*, języka *HTML* oraz bazy danych *SQL*.

5.3.1. Obsługiwane funkcje

Dostęp do funkcji aplikacji odbywa się z uprawnieniami przypisanymi do trzech ról: gościa (nie wymaga logowania), użytkownika oraz administratora.

- Uprawnienia gościa:
  - przeglądanie słowników.
- Uprawnienia użytkownika:
  - przeglądanie słowników,
  - zgłaszanie propozycji zmiany atrybutów słowa.
- Uprawnienia administratora:
  - przeglądanie słowników,
  - przeglądanie propozycji zmian i ich akceptacja bądź odrzucenie,
  - dodawanie kolejnych słów,
  - bezpośrednia edycja atrybutów słowa.

Na rysunku 5.6 przedstawiono wdrożony *cykl życia* pojedynczego słowa (hasła) w słowniku. Można rozważyć dodanie do tego schematu również aspektu przypisywania danego hasła do określonych kategorii oraz wiązania go z wyrazami bliskoznacznymi (lub przeciwnie - dodanie drugiego znaczenia do omawianego słowa). Te aspekty nie zostały jednak uwzględnione w wykonanej implementacji.



Rys. 5.6: Wdrożony cykl życia pojedynczego hasła w słowniku

5.3.2. Model informacyjny słownika

Aby obsłużyć zaplanowany cykl życia słów w słowniku zaprojektowano dwa zbiory do ich przechowywania. W zbiorze *Zmiany* znajdują się słowa niezatwierdzone jeszcze przez administratora, natomiast w zbiorze *Słownik* są już wyłącznie słowa po przyjęciu zmian, nadające się do oficjalnej publikacji. Po zatwierdzeniu modyfikacji, słowo o odpowiadającym ID zostaje zaktualizowane. W schemacie bazy danych znalazło to odzwierciedlenie w definicji następujących atrybutów: `Termin`, `Definicja słowa`, `ID słowa`, `ID słownika` (do którego przynależy dane słowo), `Login użytkownika` (który dokonał zmian jako ostatni).

5.3.3. Przykład użycia aplikacji

Działanie aplikacji przetestowano na przykładzie dziedziny pojęć biznesowych stosowanych w gabinecie fryzjerskim. Wybór dziedziny był podyktowany jej popularnością i powszechnym zrozumieniem.

Załóżmy sytuację, w której w pewnym zakładzie fryzjerskim klientki nie zawsze były zadowolone z wykonanej usługi ze względu na odmienną interpretację składanych przez nie zamówień przez fryzjera. Aby temu zaradzić szef zakładu, z wykształcenia również będący fryzjerem, udostępnia na stronie internetowej zakładu słownik pojęć biznesowych w postaci aplikacji webowej. Przy okazji wypełnia go początkową treścią oraz zakłada i udostępnia w nim konta dla swoich pracowników, przypisując im uprawnieniami właściwe dla roli użytkownika.

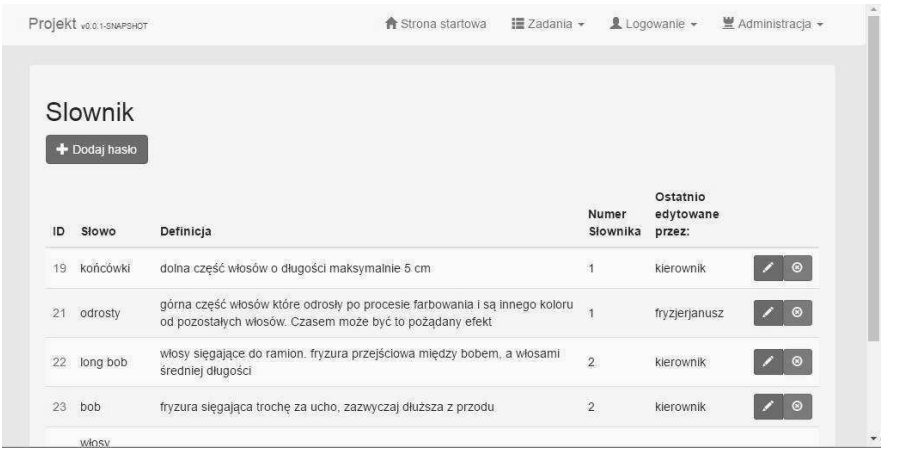
Gdy słownik zacznie być wykorzystywany produkcyjnie może zdarzyć się, że pomimo wieloletniego doświadczenia kierownikowi zakładu umknie gdzieś wiedza o najnowszych trendach oraz metodach układania włosów. Dlatego powstaje konieczność, by słownik był uzupełniany przez innych (bardziej zainteresowanych nowinkami) fryzjerów. W związku z tym muszą oni uzyskać możliwość proponowania zmian i wprowadzania nowych pojęć pojawiających się wraz z upływem czasu. Muszą więc uzyskać uprawnienia odpowiadające roli edytora.

Pomimo zaufania do swoich pracowników kierownik chciałby mieć możliwość zatwierdzania w słowniku ich propozycji. Takie „recenzowanie" pozwoliłoby mu zapobiec błędom, które mogłyby się pojawić w propozycjach niedoświadczonych pracowników bądź praktykantów.

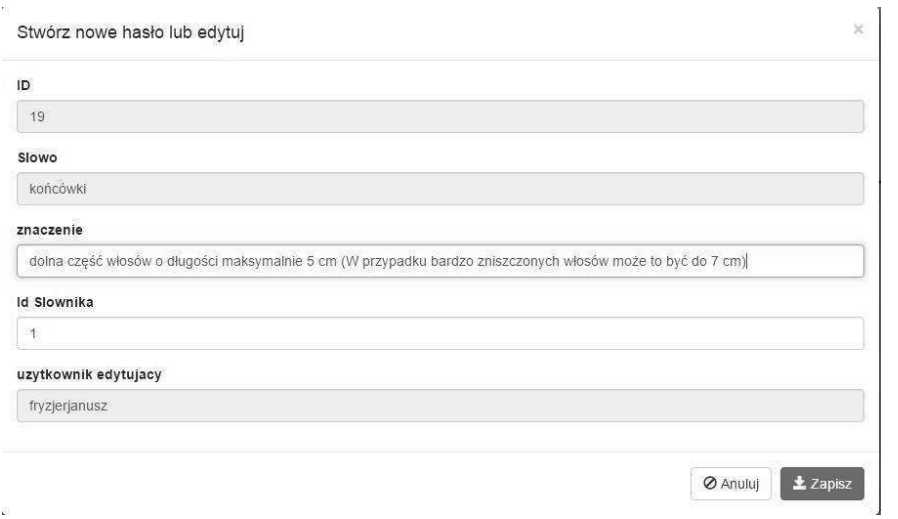
Uzupełnieniem wiedzy zawartej w słownikowych definicjach publikowanych na stronie internetowej zakładu fryzjerskiego mogłyby być definicje z innych słowników, do których prowadziłyby odpowiednie linki. Dzięki temu klientki odwiedzające zakład mogłyby poznać zasób terminów obowiązujący w zakładzie jeszcze przed jego odwiedzeniem, zyskując gwarancję wzajemnego zrozumienie podczas wizyty.

Opisany przypadek użycia może być w całości obsłużony za pomocą zaimplementowanej aplikacji. Na rysunku 5.7 pokazano fragmenty jej graficznego interfejsu użytkownika: panel słownika z możliwością edycji i usuwania słów (przy uprawnieniach administratora) oraz okno wprowadzania nowego słowa.

a)



b)



Rys. 5.7: Fragmenty graficznego interfejs użytkownika aplikacji a) panel słownika dla użytkownika z uprawnieniami administratora, b) okna wprowadzania słowa

5.4. Podsumowanie

Stosowanie metod przetwarzania wiedzy oraz tworzenie słowników pojęć biznesowych stanowią narzędzia istotnie podnoszące szanse powodzenia rozmaitych projektów lub pozwalające ograniczyć koszty (tak pieniężne, jak i czasowe) wprowadzania poprawek. Stworzona aplikacja dostarcza jedynie zestaw podstawowych funkcji. Jej zastosowanie w szerszym kontekście (jako elementu większej, dziedzinowej bazy wiedzy) wymagałoby zaimplementowania dodatkowych usprawnień i rozszerzeń. Modyfikacją, którą przede wszystkim należałoby wdrożyć, to zastosowanie modelu danych przeznaczonego dla tego typu słowników (np. omawianego wcześniej formatu *SKOS*). Pozwoliłoby to na integrację z innymi

narzędziami do zarządzania wiedzą. Listę potencjalnych modyfikacji można byłoby rozwinąć o kolejne pozycje:

- zwiększenie liczby atrybutów słów (np. czy słowo jest przestarzałe, do jakiej kategorii należy, jakie ma synonimy, przykład zastosowania itd.),
- rejestrowanie historii zmian w obrębie słowa (kto edytował, kiedy, z jakiego powodu),
- możliwość eksport słownika do formatu *.pdf* w celu ułatwienia wydruku,
- wsparcie szybkiego wyszukiwania słowa,
- zbudowanie mechanizmu rozszerzeń pozwalającego na szerszą współpracę z innymi programowymi narzędziami.

Literatura

[1] J. Żeliński. *Słownik pojęć biznesowych czyli po co nam przestrzeń nazw.* IT-Consulting. Modelowanie i analizy biznesowe, 2013. http://it-consulting.pl/autoinstalator/wordpress/2013/08/08/slownik-pojec-biznesowych-czyli-po-co-nam-przestrzen-nazw/ [dostęp dnia 26.05.2016].

[2] A. Frost. *Synthesis of Knowledge Management Failure Factors.* 2011. http://www.knowledge-management-tools.net/A%20Synthesis%20of%20Knowledge%20Management%20Failure%20Factors.pdf [dostęp dnia 25.05.2016].

[3] G. W. Mineau. Sharing knowledge: Starting with the integration of vocabularies. *Conceptual Structures: Theory and Implementation*, 754:34–45, 2005.

[4] U. of North Carolina at Chapel Hill. *Introduction to Knowledge Management.* 2007. https://web.archive.org/web/20070319233812/http://www.unc.edu/~sunnyliu/inls258/Introduction_to_Knowledge_Management.html [dostęp 21.05.2016].

ROZDZIAŁ 6

MODELOWANIE ZACHOWANIA SIĘ STADA

D. Barański, A. Mielczarek

W rozdziale omówiono podstawowe zagadnienia związane z modelowaniem zachowania się stada. Przedstawiono w nim również autorską aplikację do symulacji takiego zachowania, implementację której wykonano na bazie własnej propozycji modelu.

6.1. Wstęp

Zagadnienia związane z poruszaniem się zwierząt w stadzie wzbudzały zainteresowanie ludzi już od bardzo dawna. Patrząc na klucze przelatujących ptaków, ławice przepływających ryb czy też stada galopujących antylop można odnieść wrażenie, że poruszającymi się w grupie zwierzętami kieruje jakaś wspólna świadomość. Że samo stado, ławica czy klucz jest samodzielnym podmiotem, mogącym wpływać na ruch poszczególnych zwierząt oraz odpowiedzialnym za jego synchronizację. Wrażenie to jest źródłem pewnego dysonansu poznawczego, ponieważ nie daje się zlokalizować miejsca występowania tej świadomości w samej strukturze stada. Widać jedynie zbiór pojedynczych jednostek. Okazuje się jednak, że początkowe przeświadczenie o scentralizownym, skomplikowanym systemie sterowanie jest błędne, a zasady rządzące zachowaniem zwierząt w grupie można opisać prostymi regułami. Zostało to wykazane m.in. w pracach [1, 2, 4]. Natomiast w pracy [3] pokazano możliwości, jakie niesie ze sobą implementacja takich reguł w robotach mobilnych.

6.2. Model zachowania zwierząt w stadzie

W przełomowej dla poruszanego tematu pracy [4] zaproponowano model zbudowany na trzech prostych regułach: *wyrównania* (ang. *alignment*), *spójności* (ang. *cohesion*) i *rozdzielności* (ang. *separation*). Ten minimalny zestaw pozwala modelować zachowanie się stada zwierząt jednego typu, w nieograniczonej, pozbawionej przeszkód przestrzeni. We wspomnianej pracy zasugerowano również użycie reguły *unikania przeszkód* (ang. *avoiding obstacles*), zaś w pracy [2] przedstawiono regułę *ucieczki* (ang. *escape)* opartą na strachu przed drapieżnikami.

Każda z reguł stosowana jest indywidualnie do każdego osobnika, przy czym jego finalne zachowanie się wynika z kombinacji liniowej wszystkich użytych reguł.

W wyniku podjętych prac zestaw wymienionych reguł poszerzono o regułę *głodu* (dualną do reguły ucieczki). Odpowiada ona za ruch drapieżnika w kierunku zlokalizowanej w sąsiedztwie ofiary. Pozwoliło to na utworzenie modelu środowiska z przeszkodami oraz kilkoma gatunkami stadnych zwierząt, oddziałującymi na siebie. Ostatecznie opracowano model opisany przez 6 reguł:

wyrównanie – osobnik oddziałuje z osobnikami tego samego gatunku,
spójność – osobnik oddziałuje z osobnikami tego samego gatunku,
rozdzielność – osobnik oddziałuje z osobnikami ze wszystkich gatunków poza zwierzętami określonymi dla jego gatunku jako ofiary lub drapieżniki,
ucieczka – osobnik oddziałuje z osobnikami określonymi dla jego gatunku jako drapieżniki,
głód – osobnik oddziałuje z osobnikami określonymi dla jego gatunku jako ofiary,
unikami przeszkód – osobnik oddziałuje z przeszkodami.

Reguły te zostaną przedstawione bliżej w poniższych podrozdziałach. Poprzedzi je prezentacja modelu percepcji otoczenia zaimplementowanego w aplikacji.

6.2.1. Percepcja osobnika

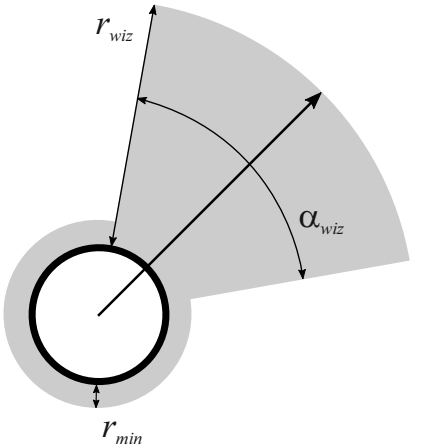
Dwuwymiarowy model percepcji otoczenia najczęściej jest budowany na bazie wycinka koła o pewnym promieniu r_{wiz} oraz kącie α_{wiz}. Dwusieczna kąta α_{wiz} pokrywa się z wektorem prędkości osobnika określającym jego przód. Jest to bardzo uproszczony model, mający imitować zmysł wzroku. Można go rozszerzać wprowadzając np. bardziej złożoną definicję geometrii otoczenia. Ta myśl stała się punktem wyjścia do implementacji własnego modelu percepcji.

W zaproponowanym rozwiązaniu powiększono obszar analizowanego otoczenia dołączając do niego powierzchnię mniejszego koła znajdujące się w bezpośrednim sąsiedztwie osobnika (tzn. uwzględniono dodatkowo powierzchnię koła o promieniu równym minimalnej odległości działania modułu separacji r_{min}, rys. 6.1). Pozwoliło to rozbudować scenariusz użycia modułu separacji oraz uzyskać dane wykorzystywane przez moduł ucieczki w przypadku, gdy drapieżnik zachodził ofiarę od tyłu.

Aby sprawdzić, czy jakieś zwierzę znajduje się w obszarze percepcji danego osobnika, należy policzyć odległość l pomiędzy położeniem tegoż osobnika a położeniem każdego ze zwierząt. W dwuwymiarowej przestrzeni euklidesowej odległość l dana jest wzorem:

$$l = \sqrt{(x_{osobnika} - x_{sąsiada})^2 + (y_{osobnika} - y_{sąsiada})^2} \tag{6.1}$$

Wyliczone odległości należy porównać z wartościami r_{min} i r_{wiz}. Jeśli $l < r_{min}$ to sąsiad jest wykrywany, jeśli zaś $l > r_{wiz}$ to sąsiad nie jest wykrywany. W trzecim przypadku należy jeszcze sprawdzić, czy dany sąsiad znajduje się w kącie widzenia α_{wiz}. Należy zatem policzyć kąt θ (pod którym porusza się osobnik) oraz kąt β (pomiędzy osobnikiem a sąsiadem):



Rys. 6.1: Model percepcji otoczenia przez osobnika w stadzie

$$\theta = \arctan\left(\frac{\dot{y}_{osobnika}}{\dot{x}_{osobnika}}\right)$$
$$\beta = \arctan\left(\frac{y_{sąsiada} - y_{osobnika}}{x_{sąsiada} - x_{osobnika}}\right)$$

Po tych krokach do uzyskania końcowego wyniku (percepcji otoczenia) należy jeszcze sprawdzić nierówność $|\theta - \beta| < \alpha_{wiz}$, czyli sprawdzić, czy wartość bezwzględna różnicy między kątami jest mniejsza od kąta widzenia. Jeżeli tak, to sąsiad jest wykrywany, w przeciwnym wypadku nie jest.

Mając wyjaśnioną kwestię percepcji otoczenia można przejść do omówienia działania samego modelu zachowania się zwierząt w stadzie zbudowanego na bazie reguł.

6.2.2. Reguła wyrównania

Reguła *wyrównania* odpowiada za synchronizację prędkości pomiędzy osobnikami. Jest to najważniejsza reguła. Jak pokazały badania symulacyjne jest ona wystarczająca do zamodelowania zgrubnego zachowania się stada. Reguła ta ma bardzo prostą implementację. Wystarczy policzyć średnią prędkość wszystkich widocznych sąsiadów i następnie zmodyfikować prędkość danego osobnika biorąc pod uwagę tę wyliczoną prędkość. Wzory opisujące regułę są następujące:

$$\dot{x}_{średnia} = \frac{1}{N}\sum_{i=1}^{N}\dot{x}_i \tag{6.2}$$

$$\dot{y}_{średnia} = \frac{1}{N}\sum_{i=1}^{N}\dot{y}_i \tag{6.3}$$

$$\dot{x}_{korekta} = w_{wyrównanie}(\dot{x}_{średnia} - \dot{x}_{osobnika}) \quad (6.4)$$

$$\dot{y}_{korekta} = w_{wyrównanie}(\dot{y}_{średnia} - \dot{y}_{osobnika}) \quad (6.5)$$

gdzie $\dot{x}_i$ i $\dot{y}_i$ – to współrzędne prędkości i-tego sąsiada, $w_{wyrównanie}$ – jest wagą wyrównania (opisującą, jak bardzo ta reguła ma wpływ na nową prędkość osobnika), N – to liczba sąsiadów branych pod uwagę.

6.2.3. Reguła spójności

Reguła *spójności* wyraża chęć osobników będących w stadzie do pozostania w nim jak najdłużej. W pracy [2] zasugerowano, że biologiczną podstawą do istnienia tej reguły jest uczucie strachu. Osobniki będące otoczone przez zwierzęta swojego gatunku czują się bezpieczniej. Reguła spójności mówi o tym, że każdy osobnik stada pragnie znaleźć się w jego środku. Na potrzeby implementacji za środek stada jest przyjmowany środek masy widocznego układu punktów (sąsiadów). Sposób liczenia tej reguły nie polega jednak na prostym wyliczeniu średniego położenia wszystkich sąsiadów. Okazuje się, że lepszym sposobem jest wykorzystanie twierdzenia o podobieństwie trójkątów. Liczymy wiec odległości od każdego z sąsiadów l_i na podstawie wzoru 6.1, oraz średnią odległość do wszystkich sąsiadów $l_{średnia}$ analogicznie do liczenia średniej prędkości (wzory 6.2, 6.3). Następnie, za pomocą poniższych wzorów liczymy wartość korekcji prędkości dla obu współrzędnych:

$$\dot{x}_{korekta} = w_{spójność} \sum_{i=1}^{N} \frac{(x_{sąsiada} - x_{osobnika})(l_i - l_{średnia})}{l_i}$$

$$\dot{y}_{korekta} = w_{spójność} \sum_{i=1}^{N} \frac{(y_{sąsiada} - y_{osobnika})(l_i - l_{średnia})}{l_i}$$

6.2.4. Reguła rozdzielności

Regułą przeciwdziałającą do poprzedniej reguły jest reguła *rozdzielności*. Reguła ta odpowiada za realistyczne ułożenie osobników wewnątrz stada. Gdyby wyłączyć tę regułę z modelu, stosowanie reguły spójności doprowadziłoby do skoncentrowania wszystkich osobników stada w jednym miejscu – stado stałoby się punktowe. Ponadto poprzez wielkość parametru r_{min} używanego w tej regule można sterować odległościami pomiędzy poszczególnymi osobnikami. Co więcej, ta reguła bierze pod uwagę także zwierzęta innego gatunku niż dany osobnik. W implementacji tej reguły ponownie zostały zastosowane twierdzenie o podobieństwie trójkątów oraz wzór na odległość 6.1. Wzory opisujące wpływ tej reguły na wynikową prędkość są następujące:

$$\dot{x}_{korekta} = -w_{rozdzielność} \sum_{i=1}^{N} \left(\frac{(x_i - x_{osobnika}) r_{min}}{l_i} - (x_i - x_{osobnika}) \right)$$

$$\dot{y}_{korekta} = -w_{rozdzielność} \sum_{i=1}^{N} \left(\frac{(y_i - y_{osobnika}) r_{min}}{l_i} - (y_i - y_{osobnika}) \right)$$

gdzie x_i i y_i są współrzędnymi położenia i-tego sąsiada.

6.2.5. Reguła unikania przeszkód

Reguła *unikania przeszkód* jest jedyną regułą, która odnosi się do otoczenia i nie bierze pod uwagę innych zwierząt. Reguła ta pozwala na wyłączenie z przestrzeni poruszania się zwierząt pewnych obszarów nazywanych przeszkodami. Przed omówieniem samej implementacji reguły warto wspomnieć o przyjętym sposobie modelowania przeszkód oraz sposobie ich percepcji.

Przeszkody w przestrzeni poruszania się zwierząt przyjmują postać wielokątów. Dzięki temu można je zapamiętywać jako zbiór punktów przestrzeni – wierzchołków przeszkody. Aby określić, czy dany punkt należy do przeszkody (znajduje się w przeszkodzie) wystarczy poprowadzić z niego półprostą w dowolnym kierunku, a następnie sprawdzić liczbę jej przecięć z odcinkami łączącymi sąsiednie wierzchołki przeszkody. Jeśli liczba przecięć jest nieparzysta to znaczy, że punkt należy do przeszkody, jeśli parzysta – nie należy.

Percepcja przeszkody opiera się na skwantowaniu części przestrzeni odpowiadającej za percepcję osobnika, a następnie sprawdzeniu, czy te skwantowane punkty należą do jednej z przeszkód. Kwantowanie podprzestrzeni percepcji ma zredukować złożoność obliczeniową tej części modelu.

Implementacja reguły unikania przeszkód jest bardzo podobna do implementacji reguły rozdzielności Wykorzystuje się w niej twierdzenie o podobieństwie trójkątów oraz wzór na odległość 6.1. Pierwszym krokiem jest policzenie tymczasowej korekcji prędkości $\dot{x}^{tmp}_{korekta}$ i $\dot{y}^{tmp}_{korekta}$.

$$\begin{aligned}
\dot{x}^{tmp}_{korekta} &= w_{przeszkody} \sum_{i=1}^{N} \left(\frac{(x_i - x_{osobnika}) r_{min}}{l_i} - (x_i - x_{osobnika}) \right) \\
\dot{y}^{tmp}_{korekta} &= w_{przeszkody} \sum_{i=1}^{N} \left(\frac{(y_i - y_{osobnika}) r_{min}}{l_i} - (y_i - y_{osobnika}) \right)
\end{aligned}$$

gdzie x_i i y_i są współrzędnymi punktów należących do przeszkód w podprzestrzeni percepcji, a N liczbą tych punktów.

W przypadku przeszkody ważne jest również, czy osobnik ma ją na wprost siebie, czy mija ją bokiem. Jeżeli przeszkoda cały czas znajduje się pod dużym kątem od kierunku ruchu, to nawet położona dość blisko nie powinna wywierać dużego wpływu. Dlatego też stosuje się drugi krok liczenia wpływu przeszkód, wykorzystując przy tym wiedzę o kącie względem położenia przeszkody γ.

$$\begin{aligned}
\gamma &= \arctan\left(\frac{\dot{y}^{tmp}_{korekta}}{\dot{x}^{tmp}_{korekta}} \right) \\
\dot{x}_{korekta} &= -\cos(\gamma) \dot{x}^{tmp}_{korekta} \\
\dot{y}_{korekta} &= -\cos(\gamma) \dot{y}^{tmp}_{korekta}
\end{aligned}$$

Funkcja $\cos(\theta)$ użyta w powyższych wzorach powoduje wyzerowanie się korekty prędkości przy kącie θ równym $\pm\pi$, propaguje pełną korektę dla $\theta = 0$ i odpowiednio zachowuje się dla wartości pośrednich.

6.2.6. Reguła ucieczki

W pracy [2] zaproponowano rozszerzenie modelu o interakcje oparte na zagrożeniu i strachu. Następstwem tego rozszerzenia jest reguła *ucieczki*. Opisuje ona wpływ pojawienia się zagrożenia (drapieżnika) na zachowania osobników w stadzie. Implementacja tej reguły też jest podobna do implementacji reguły rozdzielności. Wykorzystuje się w niej twierdzenie o podobieństwie trójkątów oraz wzór na odległość 6.1. Występują tu jednak dwie różnice. Pierwsza polega na braniu pod uwagę jedynie najbliższego zagrożenia (drapieżnika), natomiast druga – na używaniu we wzorach promienia obserwacji r_{wiz} a nie minimalnej dopuszczalnej odległości pomiędzy zwierzętami r_{min}.

$$l_m = \max_{i \in N} l_i \tag{6.6}$$

$$x_m = x_i, \text{ dla } i \text{ maksymalizującego } l_i \tag{6.7}$$

$$y_m = y_i, \text{ dla } i \text{ maksymalizującego } l_i \tag{6.8}$$

$$\dot{x}_{korekta} = -w_{ucieczka}\left(\frac{(x_m - x_{osobnika})r_{wiz}}{l_m} - (x_m - x_{osobnika})\right) \tag{6.9}$$

$$\dot{y}_{korekta} = -w_{ucieczka}\left(\frac{(y_m - y_{osobnika})r_{wiz}}{l_m} - (y_m - y_{osobnika})\right) \tag{6.10}$$

6.2.7. Reguła głodu

Dualną regułą do reguły ucieczki jest reguła głodu. Odpowiada ona za wpływ pojawienia się pożywienia na zachowanie osobników w stadzie. Implementacja tej reguły jest prawie identyczna do implementacji reguły ucieczki. Różni się jedynie znakiem w równaniach 6.9 i 6.10. Dla formalności zamieszczono poniżej równania odpowiadające tej regule.

$$l_m = \max_{i \in N} l_i$$

$$x_m = x_i, \text{ dla } i \text{ maksymalizującego } l_i$$

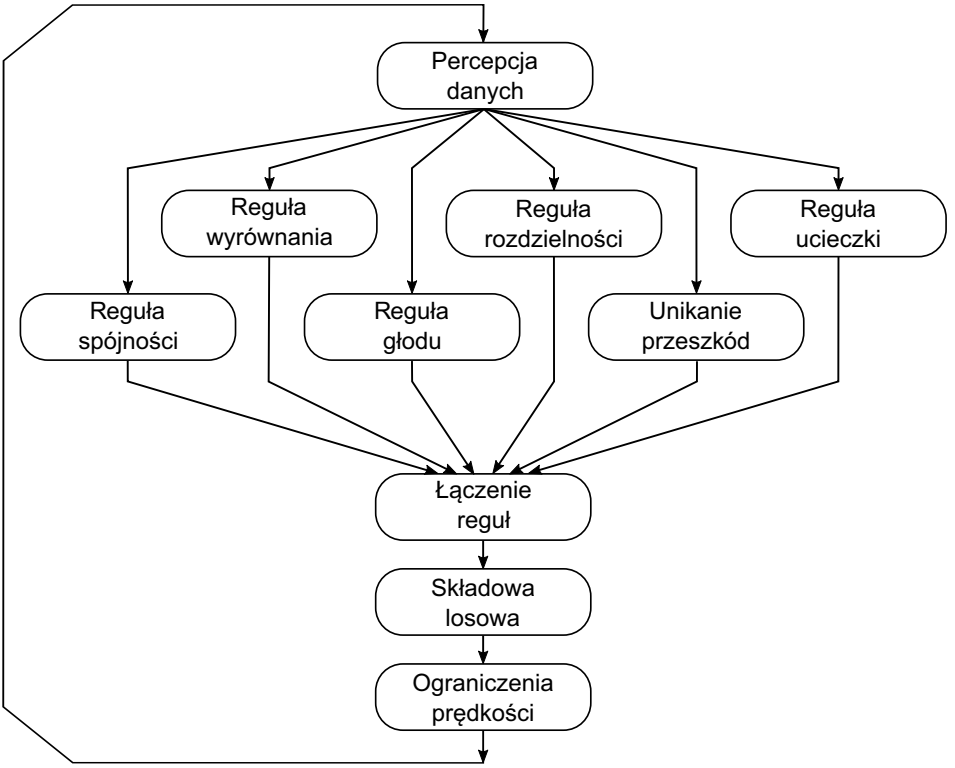
$$y_m = y_i, \text{ dla } i \text{ maksymalizującego } l_i$$

$$\dot{x}_{korekta} = w_{głód}\left(\frac{(x_m - x_{osobnika})r_{wiz}}{l_m} - (x_m - x_{osobnika})\right)$$

$$\dot{y}_{korekta} = w_{głód}\left(\frac{(y_m - y_{osobnika})r_{wiz}}{l_m} - (y_m - y_{osobnika})\right)$$

6.2.8. Złożenie reguł

Wyliczanie reguł i ich składanie obywa się iteracyjnie według schematu przedstawionego na rysunku 6.2. Każda z reguł odpalana jest równolegle z innymi i niezależnie od nich. Po wyliczeniu każdej z korekt prędkości ($\dot{x}_{korekta}$ i $\dot{y}_{korekta}$) są one sumowane i dodawane do poprzednich prędkości.



Rys. 6.2: Procedura obliczania reguł modelu

$$\dot{x} = \dot{x} + \sum_{i=1}^{6} \dot{x}^i_{korekta} \tag{6.11}$$

$$\dot{y} = \dot{y} + \sum_{i=1}^{6} \dot{y}^i_{korekta} \tag{6.12}$$

Trzeba podkreślić, że górny prawy indeks we wzorach 6.11 i 6.12 jest indeksem wyników kolejnych metod, a nie potęgą.

Po zsumowaniu powstaje nowy wektor prędkości $v = [\dot{x}\ \dot{y}]^T$. Aby uniknąć szybkiego nasycenia się wartości prędkości $|v|$ do wartości v_{max} zastosowano przekształcenie powodujące stopniowe ustalanie się $|v|$ na wartości $0.5|v_{max}|$ gdy żadna z reguł nie działa. Pozwala to obserwować przyspieszenia w momencie zastosowania którejś z reguł. Przekształcenie, o którym mowa, dane jest wzorem:

$$\begin{aligned} \dot{x} &= 0.45\dot{x} + 0.5\cos(\theta)\, v_{max} \\ \dot{y} &= 0.45\dot{y} + 0.5\sin(\theta)\, v_{max} \end{aligned}$$

Obie składowe prędkości zaszumiano szumem białym o wariancji σ.

$$\begin{aligned} \dot{x}_{szum} &\sim \mathcal{N}(0,\sigma) \\ \dot{y}_{szum} &\sim \mathcal{N}(0,\sigma) \\ \dot{x} &= 0.05\dot{x}\dot{x}_{szum} \\ \dot{y} &= 0.05\dot{y}\dot{y}_{szum} \end{aligned}$$

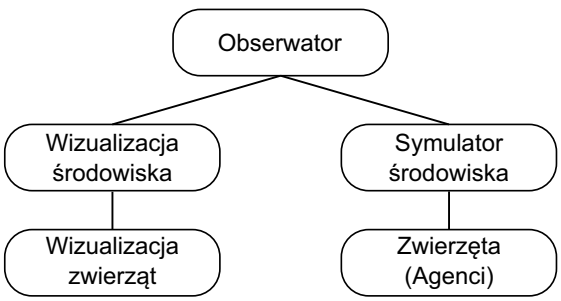
Ostatnim krokiem procedury jest zmniejszanie długości wektora prędkości v tak, aby nie przekraczał on maksymalnego przyspieszenia a_{max} oraz maksymalnej prędkości v_{max}. Uzyskuje się to następującą formułą:

$$\begin{aligned} \dot{x} &= \max(\dot{x}, \cos(\theta)\min(|v_{pop}| + a_{max}, v_{max})) \\ \dot{y} &= \max(\dot{y}, \sin(\theta)\min(|v_{pop}| + a_{max}, v_{max})) \end{aligned}$$

gdzie v_{pop} jest wektorem prędkości przed zastosowaniem reguł.

6.3. Aplikacja do symulacji zachowań modelu

W wyniku przeprowadzonych prac powstała aplikacja pozwalającą na badanie i wizualizację wpływu poszczególnych reguł na zachowanie się stada. Aplikacja ta udostępnia interfejs użytkownika, w którym można edytować parametry stada każdego z dostępnych gatunków. Do jej implementacji wykorzystano język `C++14`, kompilator `GNU gcc 4.9.2`, biblioteki `Qt 5.4.1` oraz `Qt Creator IDE 3.1.1` Na rysunku 6.3 przedstawiono uproszczony schemat struktury zaimplementowanego rozwiązania.

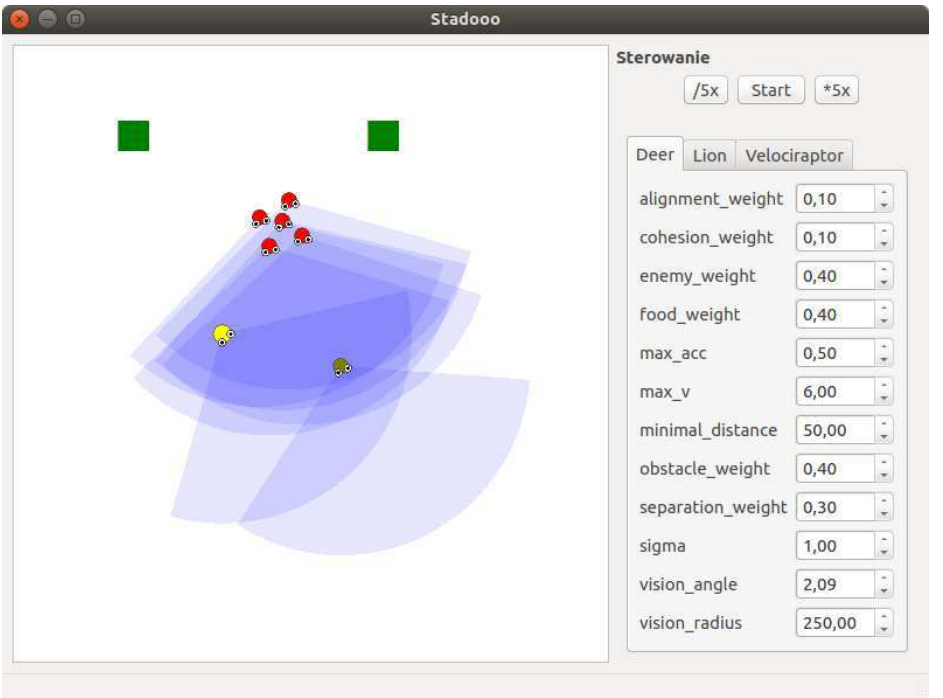


Rys. 6.3: Uproszczony schemat struktury zaimplementowanego rozwiązania

Choć główny algorytm wymusza synchroniczny ruch agentów w trybie iteracyjnym, to przy krótkim interwale czasowym symulacje wydają się płynne. Co iterację, symulator środowiska wysyła agentom reprezentującym zwierzęta informacje o tym, co znajduje się w ich podprzestrzeni percepcji. Następnie agenci obliczają swój wektor prędkości (na podstawie reguł modelu z parametrami odpowiadającymi ich gatunkowi) i przesyłają tę informację do symulatora środowiska. Symulator liczy ich nową pozycję, usuwa agentów, którzy zostali „zjedzeni" przez drapieżniki i poprzez obserwatora przekazuje informacje o zakończonej iteracji do modułu wizualizacji środowiska wraz z nowymi danymi. Moduł

ten wizualizuje obecny stan agentów przy pomocy informacji zgodnie z ustawieniami dotyczącymi sposobu wizualizowania poszczególnych gatunków zwierząt. Następnie rozpoczyna się kolejna iteracja. Dodatkowo moduł wizualizacji może w każdej chwili przyjąć dane od użytkownika (tj. przyjąć zmianę parametrów opisujących poszczególne gatunki) i za pomocą obserwatora przekazać je do części symulacyjnej.

Warto w tym miejscu wyjaśnić pojęcie „zjedzenia". Otóż „zjedzenie" (usunięcie agenta reprezentującego ofiarę) następuje w momencie, gdy w bardzo bliskiej odległości znajdą się dwaj agenci reprezentujący gatunki zwierząt będące w zależności drapieżnik–ofiara. Zrzut z ekranu zawierający widok działającej aplikacji pokazano na rysunku 6.4.



Rys. 6.4: Widok działającej aplikacji

6.4. Badanie wpływu współczynników modelu

Zaimplementowanie aplikacji umożliwiło przeprowadzenie badań nad zaproponowanym modelem. W ich trakcie skupiono się na określeniu wpływu współczynników występujących przy poszczególnych regułach na sposób zachowania się stada: jego kształt, zdolność do zachowania spójności, zmianę odległości pomiędzy poszczególnymi osobnikami, reakcję na pojawianie się drapieżników i pożywienia oraz zdolność przeżycia.

Kształt stada zależy głównie od współczynnika występującego przy regule rozdzielności oraz od kąta widzenia, przy czym współczynnik występujący w regule rozdzielności okazał się istotniejszy. Dla małych wartości współczynnika przy tej regule stado ma podobną długość oraz szerokość i kształtem przypomina koło. Dla dużych wartość współczynnika poszczególne osobniki poruszają się gęsiego (dla małej wartości kąta widzenia) lub stado kształtem przypomina rozciągniętą łzę ze stosunkową dużą szerokością na początku i cienkim końcem.

Zachowanie spójności zależy od współczynników występujących przy regule spójności, rozdzielności oraz, co było początkowo zaskoczeniem, bardzo mocno od kąta widzenia. Kąt widzenia odgrywa w tym przypadku zasadniczą rolę. W przypadku, gdy jest on większy niż π, nie sposób wręcz nadać takich wartości współczynnikom spójności i rozdzielności, które „zepsułyby" spójność stada. Jednak w przypadkuk, gdy ten kąt staje się bardzo mały (powiedzmy, mniejszy niż $\frac{\pi}{4}$), wpływ współczynników spójności i rozdzielności bardzo się uwydatnia.

Jeśli rozdzielność jest większa niż spójność, wtedy wielkie stada łatwo rozpadają się na małe, przy czym małe stada nie rozpadają się już dalej. Ciekawą obserwacją jest fakt, że dzieje się tak zarówno dla małych jak i dla dużych wartości współczynnika spójności w porównaniu do rozdzielności (ale wciąż mniejszych). Dla małych wartości rozdzielności i spójności stada nie rozpadają się łatwo a ich szerokość jest duża w porównaniu z długością. Kolejną ciekawą obserwacją jest fakt, że dla dużych wartości współczynnika spójności i małych rozdzielności stada łatwo się rozpadają – co nie jest zgodne z początkową intuicją. Oczywistym wydaje się, że dla bardzo dużej odległości minimalnej w porównaniu z promieniem widzenia stada rozpadają się prawie natychmiastowo, natomiast bardzo mała odległość minimalna sprzyja zachowaniu spójności.

Odległość pomiędzy osobnikami zależy przede wszystkim od odległości minimalnej. Oprócz niej jedynym parametrem wpływającym na odległość pomiędzy osobnikami jest współczynnik przy rozdzielności. Jest to jednak wpływ marginalny, wykrywalny tylko dla bardzo małych wartości tego współczynnika. Wpływ odległości minimalnej jest bliski liniowemu, tzn. odległość pomiędzy osobnikami rośnie proporcjonalnie do wzrostu odległości minimalnej. Dzieje się jednak tak tylko do momentu, gdy wartość odległości minimalnej zaczyna zbliżać się do wartości promienia widzenia – co powoduje utratę spójności stada.

Reakcja na drapieżniki i pożywienie oraz zdolność przeżycia zależą od innych współczynników niż poprzednie cechy. Są nimi współczynniki ucieczki i głodu oraz maksymalne przyspieszenie i prędkość. Podstawową i banalną zależnością jest ta, że przy dużych wartościach przyspieszeń maksymalnych i izolowanych układach drapieżnik–ofiara jedyne znaczenie dla problemu „czy ofiara będzie złapana czy nie" ma prędkość maksymalna. W przypadku, gdy maksymalne przyspieszenie jest u ofiary bardzo niskie, zdarza się, że jest ona złapana nawet jeśli ma wyższą prędkość maksymalną. W odwrotnym przypadku (tzn. kiedy drapież-

nik ma małe przyspieszenie ale wyższą prędkość maksymalną) drapieżnik prawie zawsze był górą.

Ciekawym przypadkiem testowym było odejście od izolowanych układów stado drapieżników – stado ofiar na rzecz kilkustopniowych i skupienie się na zachowaniu osobników gatunku, który był jednocześnie ofiarą dla jednego i drapieżcą dla innego. Wyniki jednak nie są zbyt zaskakujące. W przypadku, gdy współczynnik ucieczki był zauważalnie większy, osobniki tego gatunku nie ryzykowały polowania i uciekały z sąsiedztwa dwóch pozostałych gatunków. Natomiast w przypadku gdy współczynniki te były podobnej wartości, osobniki często podejmowały ryzyko i równie często ginęły chwilę po zabiciu swojej ofiary. Stawało się to bardziej widoczne wraz ze wzrostem różnicy pomiędzy współczynnikami, aż do momentu, gdy częściej zdarzało się ginąc nawet przed upolowaniem swojej zdobyczy.

Współczynnik wyrównania okazał się najważniejszym współczynnikiem dla modelu stada, jednak nie pozwalał on na przeprowadzenie ciekawych analiz. Okazało się, że małe wartości tego współczynnika w porównaniu do innych powodują, że osobniki przestają zachowywać się stadnie.

6.5. Podsumowanie

Okazuje się, że matematyczne modelowanie zachowań stada na podstawia kilku prostych reguł odnoszących się do każdego osobnika pozwala na dobre przybliżenie zachowań stadnych. Przybliżenie to ma jednak pewne ograniczenia związane przede wszystkim z nieuwzględnieniem w modelu wpływu zmęczenia oraz innych zmysłów niż wzrok. Powoduje to szybkie nasycenie się prędkości osobnika oraz czasem nierealistyczne zachowanie osobników przy dużych zmianach kierunku ruchu. Choć wprowadzone przez autorów rozwiązania tych problemów dodawały modelowi realizmu, okazały się one jednak jeszcze niewystarczające do pełnego odzwierciedlenia rzeczywistego zachowania się stada. Niemniej wyznaczyły one ścieżkę do budowy dokładniejszych modeli.

Literatura

[1] A. Banks, J. Vincent, C. Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6(4):467–484, 2007.

[2] C. Delgado-Mata, J. I. Martinez, S. Bee, R. Ruiz-Rodarte, R. Aylett. On the use of virtual animals with artificial fear in virtual environments. *New Generation Computing*, 25(2):145–169, 2007.

[3] H. Min, Z. Wang. Design and analysis of group escape behavior for distributed autonomous mobile robots. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, strony 6128–6135. IEEE, 2011.

[4] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, wolumen 21, strony 25–34. ACM, 1987.

ROZDZIAŁ 7

ASYSTENT PARKOWANIA

T. Baciak, A. Sobocka

W niniejszym rozdziale opisano projekt systemu, którego zadaniem jest wspomaganie kierowcy podczas manewru parkowania. System dokonuje pomiarów danych z czujników w celu nałożenia odpowiedniej ścieżki na obraz z kamery.

7.1. Wstęp

Miejski tłok i ścisk nie sprzyjają wygodnemu i szybkiemu parkowaniu. Do wykonania tego manewru zwykle jest na tyle mało miejsca, że nie tylko początkujący, ale również zaawansowani kierowcy mogą mieć z nim problemy. Szczególne znaczenie zaczynają mieć wtedy inteligentne systemy parkowania, które z roku na rok zdobywają coraz to większą popularność. Rozwój technologii sprawił, że asystenci parkowania dołączani są obecnie do wyposażenia wielu nowych modeli samochodów. Oferują one użytkownikom wsparcie podczas manewrów parkowania wykrywając przeszkody i wyświetlając potencjalną trajektorię ruchu pojazdu. Bardziej zaawansowane wersje posiadają specjalne algorytmy potrafiące wykonać tę czynność w pełni automatycznie [4].

Pierwszy asystent parkowania [1] powstał już w latach 50 w Kalifornii (rys. 7.1). O „inteligencji" tego systemu nie da się zbyt wiele powiedzieć: był to układ hydrauliczny z dodatkowym kołem, które wysuwając się podnosiło tylne koła pojazdu i umożliwiało skręt po mniejszym promieniu.

Obecne systemy parkowania [5, 2] potrafią znaleźć miejsce parkingowe oraz zmierzyć odległość od przeszkody. Niektóre jedynie wspomagają manewr parkowania, ale są i takie, które mogą wyręczyć w tym całkowicie użytkownika pojazdu. Wiele z nich posiada ekran, na którym można obserwować obraz otoczenia rejestrowany za pomocą kamery. Na obraz ten nakładane są linie pokazujące potencjalną trajektorię pojazdu w danym położeniu oraz odległości od przeszkód. Nie wszystkie systemy wykorzystują jednak wyświetlacz. Najnowsze mogą być również sterowane głosowo.

Typowy inteligentny asystent parkowania działa podobnie jak system instalowany w Volkswagenie Passat B8 [3]. Po wybraniu przez użytkownika rodzaju parkowania (równoległe lub prostopadłe) system przystępuje do szukania miejsce

Rys. 7.1: Pierwszy asystent parkowania [1]

parkingowego podczas dalszej jazdy kierowcy (z odpowiednio małą prędkością). Po dokonaniu wyboru wyświetla komunikat z prośbą o wybranie odpowiedniego biegu i po tym zabiegu kontrola nad kierownicą zostaje przyjęta przez asystenta. Zadaniem kierowcy jest monitorowanie położenia samochodu (za pomocą czujników i kamery) oraz operowanie hamulcem i gazem zgodnie z instrukcjami systemu. Asystent pomaga również w opuszczeniu miejsca parkingowego ostrzegając przed ruchomymi pojazdami znajdującymi się w bliskim otoczeniu. Jednak nie wszystkie systemy przejmują kontrolę nad kierownicą. Mogą również informować użytkownika o jej skręcie w sposób głosowy lub wyświetlając komunikaty na ekranie.

7.2. Projekt prototypu asystenta parkowania

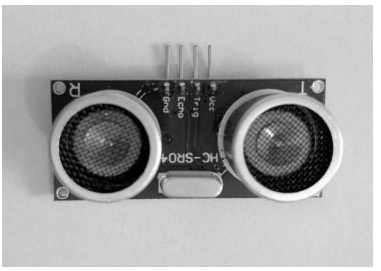
7.2.1. Założenia

Celem omawianego projektu było stworzenie sprzętowego modułu integrującego dane z wielu czujników, przeprowadzającego ich analizę oraz prezentującego uzyskane wyniki w postaci toru nakładanego na obraz kamery wyświetlany na monitorze. Moduł ten miał pełnić rolę prostego asystenta (prototypu) wspierającego kierowcę podczas manewru parkowania (bez możliwości sterowania samochodem i uruchamiania złożonych algorytmów). Dokładniej mówiąc: zadaniem modułu miało być wyświetlanie toru ruchu odpowiadającego aktualnemu położeniu pojazdu oraz informowanie użytkownika o przeszkodach znajdujących się w najbliższym otoczeniu.

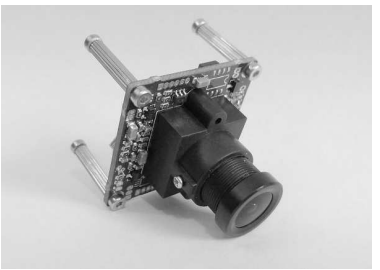
Do wykonania prototypu miał być użyty jeden czujnik ultradźwiękowy, buzzer (do informowania o otoczeniu) oraz enkoder symulujący ruch kierownicy. Istotnym elementem miała być również kamera przesyłająca obraz na ekran monitora (tablet) oraz system OSD, dzięki któremu potencjalny tor ruchu pojazdu mógł się pojawić na ekranie. Komunikacja całego modułu z tabletem służącym do wyświetlania obrazu miała odbywać się w sposób bezprzewodowy.

7.2.2. Opis rozwiązania

Do wykonania prototypu postanowiono wykorzystać elementy przedstawione na rysunkach 7.2 i 7.3.

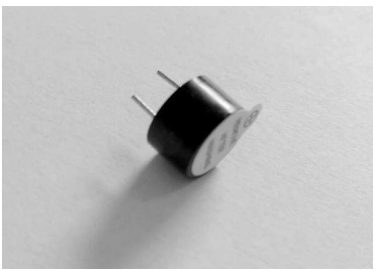


Czujnik ultradźwiękowy HC-SR04
Dane techniczne:
zasilanie: 5V,
pobór prądu: < 2 mA,
kąt pomiary: $< 15^o$,
praca w odległości: 2 - 500 cm,
dokładność: 0,3 cm,
wymiary: 20×40 mm.



Kamera szerokokątna SONY 600TVL
Dane techniczne:
soczewka: 2,8 mm,
wyjście video: 1.0 V_{pp} NTSC/PAL,
zasilanie: 12 V ± 10% DC,
pobór prądu: 100 mA,
kąt widzenia: 100^o.

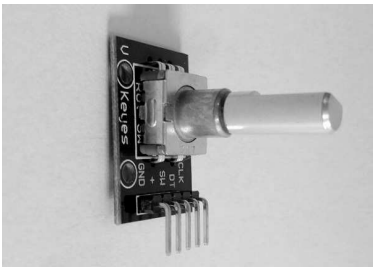
Videograbber Media-Tech MT4169
Dane techniczne:
Wejście wideo: 1.0 V_{pp} NTSC/PAL,
zasilanie: 5 V DC,
kompatybilny z USB: 2.0/1.1,
złącza Video: 1x Composite, 1x S-Video,
złącza Audio: Stereo Audio (Cinch),
DVD+/-R/RW, DVD+/-VR i DVD-Video.



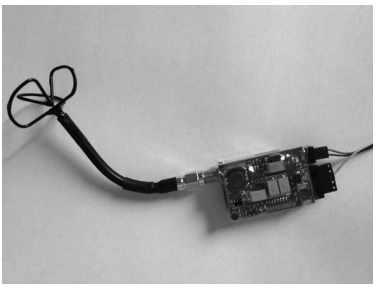
Buzzer
Dane techniczne:
napięcie zasilania: 5 V,
pobór prądu: maks. 20 mA,
częstotliwość: 2,5 kHz ± 500 Hz,
głośność: 75 dB.

Rys. 7.2: Elementy systemu

Głównym elementem systemu jest mikrokontroler STM32F103C8T6 taktowany zegarem 72 MHz, obsługujący wszystkie funkcjonalności modułu oraz komunikację z użytkownikiem. Na schemacie blokowym przedstawiającym architekturę całego rozwiązania zajmuje on miejsce centralne (rys. 7.4).



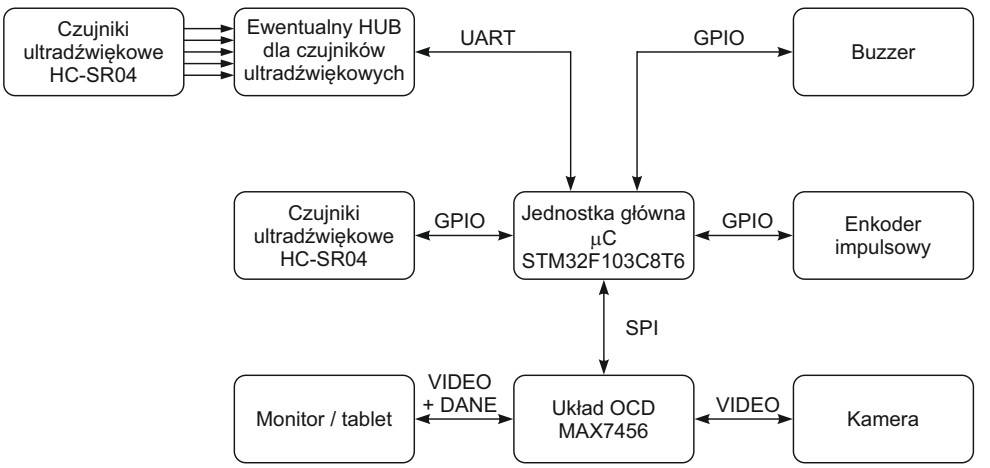
Enkoder
Dane techniczne:
24 Impulsy/obrót,
enkoder z przyciskiem,
żywotność: 20000 cykli,
sygnały wyjściowe: 2 sygnały A i B,
średnica osi: 6 mm,
oś: plastikowa ścięta, długość: 14.,5 mm.



Nadajnik Boscam TS351
Dane techniczne:
moc: 200 mW,
pasmo: 5,8 GHz,
ilość kanałów: 8,
zasilanie 6,5-12 V DC,
pobór prądu: 200 mA.

Odbiornik Boscam RC805
Dane techniczne:
czułość: -90 dBm,
pasmo: 5,8 GHz,
ilość kanałów: 8,
napięcie pracy: 5-12 V DC,
pobór prądu: 220 mA.

Rys. 7.3: Elementy systemu (cd.)



Rys. 7.4: Schemat architektury wykonanego prototypu asystenta parkowania

Za pomiar odległości odpowiada czujnik ultradźwiękowy HC-SR04. Jego obsługa odbywa się za pomocą pinów GPIO połączonych z wewnętrznym timerem mikrokontrolera. Generuje on impuls wyzwalający o długości $10\mu s$ na pinie TRIG czujnika oraz bada długość stanu wysokiego na linii ECHO czujnika z dokładnością do $1\mu s$. Na podstawie znajomości wartości prędkości dźwięku w powietrzu oraz pomiaru sygnału ECHO wyliczana jest odległość do przeszkody.

Do testów wykorzystano jeden czujnik bezpośrednio podłączony do mikrokontrolera. Istnieje jednak możliwość zastosowania całego systemu czujników, składającego się nawet z 8 sztuk rozlokowanych w różnych miejscach pojazdu. Do ich obsługi przewidziano układ zewnętrzny, tzw. HUB, wysyłający dane pomiarowe do jednostki głównej przez magistrale UART.

Czujnikiem odpowiadającym za odczyt położenia kierownicy jest enkoder impulsowy, także bezpośrednio połączony do mikrokontrolera. Obsługa enkodera odbywa się za pomocą dwóch pinów GPIO wywołujących przerwania w przypadku zmiany stanu na którymś z nich. Znajomość sekwencji zmian pozwala jednoznacznie wyznaczyć położenie enkodera. Na tej podstawie jest wyznaczana i wizualizowana potencjalna trajektoria poruszania się samochodu.

Do wizualizacji trajektorii służy układ MAX7456, który nakłada odpowiednie symbole wizualizujące trajektorię oraz dane z czujników ultradźwiękowych na obraz z kamery (w tym przypadku odległość w centymetrach). Połączony jest on do mikrokontrolera za pomocą magistrali SPI skonfigurowanej według informacji zawartej w instrukcji obsługi układu. Wejście sygnału video układu MAX7456 podłączone jest do kamery za pomocą przewodu Analog Video, zaś wyjście sygnału video połączone jest do nadajnika systemu bezprzewodowej transmisji wizji także za pomocą przewodu AV. Sygnał trafia do odbiornika, korzystając z pasma 5,8 GHz.

Odbiornik jest połączony do video-grabbera za pomocą przewodu AV. Videograbber jest odpowiedzialnego za dyskretyzację sygnału do postaci cyfrowej. W tej bowiem postaci jest on wysyłany do komputera przez magistralę USB. Dodatkowo system wyposażony jest w sygnalizator dźwiękowy z wbudowanym generatorem, który sterowany jest z portu GPIO połączonego do timera mikrokontrolera. Połączenie nie jest bezpośrednie. Z racji dużego obciążenia prądowego zastosowano tranzystor. Działanie sygnalizatora dźwiękowego polega na zwiększaniu częstotliwości występowania dźwięku wraz ze zmniejszaniem się odległości mierzonej przez czujnik ultradźwiękowy – w przypadku zastosowania kilku czujników, częstotliwość zależałaby od najbardziej krytycznego odczytu, czyli najmniejszej odległości. Nie przewidziano rozróżniania dźwiękowego każdego z czujników z powodu niskiej ergonomii tego rozwiązania.

7.2.3. Wyniki pracy

Na rysunkach 7.5 i 7.6 przedstawiono efekt końcowy projektu – wizualizację danych na obrazie z kamery. Na rysunku 7.5 pokazano widok z kamery przy wyprostowanej kierownicy, natomiast na rysunku 7.6 – wyznaczoną przez asystenta ścieżkę przy skręcie samochodu. W górnych rogach ekranu wyświetlają się od-

czyty z czujników: ultradźwiękowego (z lewej strony) i enkodera symulującego skręt kierownicy (z prawej).

Rys. 7.5: Wizualizacja danych przy wyprostowanej kierownicy

Rys. 7.6: Wizualizacja danych przy skręconej kierownicy

7.3. Podsumowanie

Stworzony w ramach projektu system spełnił wszystkie założenia. Uruchomiony czujnik ultradźwiękowy jest w stanie bez problemu lokalizować przeszkody w odległości do 2 metrów z częstotliwością do prawie 100 Hz (częstotliwość ta zależy od odległości do przeszkody, w najgorszym wypadku dla zasięgu 2 metrów i dla prędkości dźwięku w powietrzu można wyliczyć, że czas pomiaru to 11,765 ms). Stosownie do odległości odczytanej przez ten czujnik generowany jest sygnał sterujący dla buzzera. Powoduje on zwiększanie się częstotliwości wraz ze zmniejszaniem się odległości do przeszkody. Generowane są krótkie impulsy o stałej długości, zmienną jest częstotliwości ich występowania. Algorytm w bardzo dobry sposób imituje system zastosowany w samochodach Skoda, na którym wzorowano się w projekcie. Enkoder impulsowy został uruchomiony i w pełni poprawnie symuluje zachowanie się kierownicy, która wpływa na kształt generowanej trajektorii pojazdu. Układ OSD również został poprawnie uruchomiony i pozwala na płynne wyświetlanie obrazu oraz nakładanie danych z częstotliwością 10 Hz.

Zastosowana kamera pozwala na wyświetlanie obrazu w jakości NTSC, czyli w rozdzielczości 720x480 pikseli, zaś układ scalony MAX7456 pozwala jedynie na operowanie na przestrzeni 30 znaków w każdej z 16 linii (gdzie każdy znak składa się z 18x12 pikseli, co daje rozdzielczość 360x288 pikseli). Powoduje to znaczne pogorszenie jakości obrazu. Dodatkową wadą tego układu jest możliwość wyświetlania jedynie monochromatycznych znaków. Zaletą tego rozwiązania jest łatwa implementacja sprzętowa oraz niska cena. W przypadku podjęcia zamiaru dalszego rozwijania projektu należy wziąć pod uwagę kamery w rozdzielczości HD, oraz cyfrową obróbkę obrazu, co pozwoli uzyskać lepszą jakość obrazu oraz efekty wizualne.

Literatura

[1] Pierwszy na świecie asystent parkowania, Mar. 2011. `http://www.geekweek.pl` [dostęp dnia 10 czerwca 2016].

[2] Systemy pomocne podczas parkowania, 2014. `http://autokult.pl/` [dostęp dnia 10 czerwca 2016].

[3] Asystent parkowania w samochodzie Volkswagen Passat, który sam znajdzie miejsce i (prawie) sam zaparkuje, Sier. 2015. `http://technowinki.onet.pl/` [dostęp dnia 10 czerwca 2016].

[4] B. Dafflon, J. M. Contet, F. Gechter, P. Gruer. Toward a reactive agent based parking assistance system. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, wolumen 1, strony 500–507, List. 2012.

[5] P. Krężołek. Robot mobilny z systemem automatycznego parkowania. Praca dyplomowa inżynierska. Politechnika Warszawska, 2014.

ROZDZIAŁ 8

REGUŁY ASOCJACYJNE W BADANIU ZALEŻNOŚCI MIĘDZY TYPEM GWIAZDY A WYSTĘPOWANIEM TOWARZYSZĄCYCH JEJ EGZOPLANET

T. Wodziczko, M. Kuklewski

W niniejszym rozdziale opisano sposób wykorzystania metody odkrywania reguł asocjacyjnych do badania zależności pomiędzy występowaniem egzoplanet a parametrami ich gwiazdy macierzystej. Sprawdzono, czy istnieją reguły pomagające określić prawdopodobieństwo wystąpienia planety przy gwieździe o danych parametrach. Badanie przeprowadzono na ogólnodostępnej bazie danych systemów gwiezdnych obserwowanych Kosmicznym Teleskopem Keplera [2].

8.1. Wstęp

Metody odkrywania reguł asocjacyjnych są wykorzystywane do badania związków w zbiorach danych. Można je stosować do badania zjawisk opisanych rekordami obserwacji. Ich opracowanie zostało zainspirowane badaniem zawartości koszyka zakupów. Celem tego badania było zidentyfikowanie zestawów produktów pojawiających się razem w koszykach klientów oraz zoptymalizowanie rozlokowania towarów w sklepie. W związku z tym o wykorzystaniu reguł asocjacyjnych najczęściej mówi się w kontekście koszyka zakupów.

W każdej analizie koszyka zakupów bada się częstość występowania pewnych produktów w zarejestrowanych transakcjach. Formalizując to zagadnienie: zbiór transakcji oznaczany jest przez $T = \{t_1, t_2, ..., t_N\}$, zaś zbiór wszystkich produktów przez $I = \{i_1, i_2, ..., i_M\}$. Zależność pomiędzy T oraz I jest taka, że każda transakcja $t_n = \{i_{p_1}, ..., i_{p_{l_n}}\}$ jest zbiorem produktów (krotką) o długości l_n , przy czym długości poszczególnych transakcji mogą być różne.

W najprostszym podejściu w transakcjach nie zamieszcza się informacji o ilości zakupionych produktów danego typu. Wiadomo jedynie, że zostały one zakupione. Pojawiające się oznaczenia N oraz M, to, odpowiednio, liczba transakcji

zarejestrowanych w bazie danych oraz liczba dostępnych (typów) produktów. Ich wartości przyjmuje się jako stałe podczas analizy. W przypadku ich zmiany wyszukiwanie reguł asocjacyjnych powinno rozpocząć się na nowo.

8.1.1. Odkrywanie reguł asocjacyjnych

Reguły asocjacyjne definiowane są jako implikacje wiążące dwa podzbiory produktów, które wystąpiły razem w transakcjach wystarczająco często, aby dało się stwierdzić z dużym prawdopodobieństwem, że zakup produktów z jednego podzbioru (X) implikuje zakup produktów z drugiego podzbioru Y, czyli $X \rightarrow Y$. Przykładowo: reguła $\{i_1, i_2\} \rightarrow \{i_3\}$ oznacza, że przy zakupie produktów i_1 oraz i_2 zachodzi duże prawdopodobieństwo dokonania zakupu dodatkowego produktu i_3 w tej samej transakcji. Prawdopodobieństwo to określa się za pomocą parametru ufności *conf* wyliczanego ze stosunku liczby transakcji zawierających produkty z obu podzbiorów do liczby transakcji zawierających produkty z podzbioru pierwszego [4]:

$$conf(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{8.1}$$

gdzie $\sigma(A)$ oznacza liczbę transakcji zawierających produkty ze zbioru A.

Drugim istotnym parametrem branym pod uwagę podczas odkrywania reguł asocjacyjnych jest wsparcie *supp*. Mówi ono jak często w bazie danych występuje reguła $X \rightarrow Y$. Wsparcie wyznacza się w następujący sposób [4]:

$$supp(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{8.2}$$

Przystępując do odkrywania reguł asocjacyjnych przyjmuje się pewne minimalne poziomy ufności *confmin* i wsparcia *suppmin*. Pozwalają one postawić próg, przekroczenie którego pozwoli potraktować okrytą zależność jako regułę. Dzięki takiemu zabiegowi daje się wyeliminować transakcje, w których zbiór produktów był przypadkowy, lub transakcje rzadkie. W efekcie można otrzymać zbiór reguł asocjacyjnych o zmniejszonej liczności, spełniający warunki: $conf(X \rightarrow Y) \geqslant confmin$ i $supp(X \rightarrow Y) \geqslant suppmin$. Ponadto należy pamiętać, że przy odkrywaniu reguł asocjacyjnych obowiązuje warunek rozłączności podzbiorów X i Y (czyli $X \cap Y = \emptyset$) [4].

Załóżmy zatem, że mamy następujący zbiór transakcji, w którym $N = M = 5$:

$t_1 = \{i_1, i_2, i_4\}$
$t_2 = \{i_1, i_3\}$
$t_3 = \{i_2, i_3, i_4\}$
$t_4 = \{i_1, i_2, i_3, i_4\}$
$t_5 = \{i_2, i_4, i_5\}$

a wartości minimalne wsparcia i ufności ustalono na $suppmin = 0.6$ i $confmin = 0.8$. Dla przykładowej reguły $\{i_2\} \rightarrow \{i_4\}$ wartości wsparcia i ufności wynoszą, odpowiednio, $supp = \frac{4}{5} = 0.8$ i $conf = \frac{4}{4} = 1$. Oba podzbiory występujące w regule są rozłączne. Można zatem uznać, że $\{i_2\} \rightarrow \{i_4\}$ jest regułą asocjacyjną (z interesującą nas istotnością). W innym przykładzie $\{i_1, i_2\} \rightarrow \{i_4\}$ nie będzie re-

gułą asocjacyjną, ponieważ $supp = \frac{2}{5} = 0.4$, a stąd nie zostaje spełniony warunek $supp(X \rightarrow Y) \geqslant suppmin$.

Generowanie reguł zasadniczo nie jest trudnym zadaniem. Można go rozwiązać stosując algorytm siłowy, tj. generując wszystkie możliwe podzbiory produktów we wszystkich transakcjach i sprawdzając, czy przekraczają one założone progi na wartości ufności i wsparcia. Jest to jednak mało wydajna i obliczeniowo kosztowne, szczególnie w przypadku analiz dokonywanych na bazach danych o duży rozmiarze. Dostępną liczbę kombinacji podzbiorów dla M różnych produktów wyraża równanie [4]:

$$R = 3^M - 2^{M+1} + 1 \tag{8.3}$$

Jak widać, jest to liczba rosnąca wykładniczo z liczbą analizowanych produktów. Zamiast więc stosować algorytm siłowy należało znaleźć wydajniejsze rozwiązanie. Okazał się nim algorytm Apriori.

8.2. Algorytm Apriori

Algorytm Apriori jest jednym z najlepszych algorytmów wyszukiwania reguł asocjacyjnych. Bazuje on na podziale zadania na dwie części: fazę wyznaczania zbiorów częstych i fazę generowania reguł. Zbiory częste Z_i są w tym przypadku zbiorami które spełniają warunek $supp(Z_i) \geqslant suppmin$.

Faza wyznaczania zbiorów częstych opiera się w głównej mierze na fakcie, iż dla zbioru częstego każdy jego podzbiór również jest zbiorem częstym. Przystępując do generowania zbiorów częstych można więc całą procedurę zacząć od wygenerowania zbiorów częstych o liczności 1, a następnie na ich bazie generować zbiory o liczności 2 i tak dalej, aż do wygenerowania zbiorów o liczności M. W każdej iteracji należy sprawdzać, czy wygenerowane zbiory spełniają narzucone warunki na ufność. Tak więc w fazie generowania zbiorów częstych algorytm Apriori ogranicza się do przeszukiwania wszerz dostępnej przestrzeni generowanej ze zbioru I z odcinaniem gałęzi nieistotnych.

Zbiory częste służą do odkrywania reguł asocjacyjnych. Jeśli Z_i oznacza zbiór częsty, to wygenerowana na jego postawie zależność $X \rightarrow Y$, gdzie $X, Y \subset Z_i$, oraz $X \cap Y = \emptyset$ będzie regułą asocjacyjną, gdy spełniony zostanie warunek $conf(X \rightarrow Y) \geqslant confmin$. W celu redukcji ilości obliczeń można wykorzystać obliczone wcześniej wartości wsparcia:

$$conf(X \rightarrow Y) = \frac{supp(X \rightarrow Y)}{supp(X)} = \frac{supp(Z_i)}{supp(X)} \tag{8.4}$$

Istotne jest tutaj również twierdzenie [4] mówiące, iż jeżeli warunek dotyczący ufności *conf* nie jest zachowany dla reguły $X \rightarrow Y$ to nie może on być również zachowany dla $X' \rightarrow Y$, gdzie $X' \subset X$. W praktyce oznacza to, że dla zbioru częstego Z_i należy zacząć poszukiwanie reguł od podzbiorów X o największej liczności.

8.3. Badania na zbiorze danych z teleskopu Kepler

Kosmiczny Teleskop Keplera jest instrumentem naukowym służącym głównie do obserwacji planet pozasłonecznych. Jest on wykorzystywano do [1]:

1. Określania częstości występowania planet typu ziemskiego i większych w pobliżu ekosfery różnych gwiazd.
2. Określania parametrów orbit tych planet.
3. Szacowania ile planet może znajdować się w układach gwiazd wielokrotnych (układ więcej niż jednej gwiazdy).
4. Określanie parametrów planet typu Gorący Jowisz (gazowe olbrzymy, z orbitą o niewielkiej średnicy).
5. Identyfikacji innych obiektów należących do układu.
6. Określania parametrów gwiazd, przy których znajdują się planety.

Teleskop Keplera wykrywa planety poprzez wykrywanie ich tranzycji na tarczy gwiazdy macierzystej. Jest to osiągane za pomocą urządzenia optycznego, które dokonuje pomiaru jasności gwiazdy, jak również tworzonych na tej podstawie serii danych reprezentujących jasność gwiazdy w dziedzinie czasu. Na tak przygotowanych danych można przeprowadzać analizę fluktuacji jasności gwiazdy. Okresowe przyciemnienia o jednakowej amplitudzie oznaczają, że wokół gwiazdy może krążyć planeta [1].

O ile wykrywanie planet nie jest zadaniem związanym z regułami asocjacyjnymi, to można go jednak zastosować do określania parametrów gwiazd, przy których znajdują się planety. Poniżej opisano dokładniej ten przypadek.

8.3.1. Wykorzystany zbiór danych

Do badań wykorzystano zbiór danych zawierający parametry gwiazd powstały na bazie obserwacji wykonanych Kosmicznym Teleskopem Keplera [2]. Kryterium doboru gwiazd do zbioru danych był liczba planet w systemie gwiezdnym $N_{planet} \geqslant 0$. Tak dobrane kryterium daje informacje o wszystkich badanych przez teleskop gwiazdach. Skutkuje to bazą danych zawierającą 793148 rekordów.

Do testów wybrano następujące parametry gwiazdy:

1. Temperatura - jest to temperatura gwiazdy podana w Kelwinach.
2. Metaliczność - jest stosunek masy pierwiastków cięższych od helu do masy helu i wodoru w gwieździe.
3. Promień - jest to promień gwiazdy podany w średnicach słońca.
4. Masa - jest to masa gwiazdy podana w masach słońca.
5. Gęstość - jest to gęstość gwiazdy podana w $\frac{g}{cm^3}$.
6. Odległość - jest to odległość od słońca mierzona w parsekach.
7. Planety - jest to liczba planet.

Aby dane nadawały się do wykorzystania w algorytmie Apriori należało je jeszcze pogrupować, stawiając przedziały na wartości parametrów. Bez tego zabiegu każda nowa wartość tworzyłaby inny przypadek danego parametru, a przez to nawet niewielka wartość *suppmin* nie byłaby osiągana, czyli nie powstawałyby zbiory częste.

Przyjęto, że każdy z rekordów bazy danych reprezentuje transakcję T, natomiast parametry gwiazdy odpowiadają produktom. W odróżnieniu do koszyka zakupów, gdzie obecność lub nieobecność produktów w koszyku reprezentowano odpowiednio za pomocą wartości binarnych (0 lub 1) w badanym przypadku produkt może przyjąć wartości od 0 do N_{dys_i}, przy czym N_{dys_i} jest liczbą przedziałów dyskretyzacji danego parametru. Można to interpretować jako sprawdzenie liczby zakupionych produktów danego typu.

8.3.2. Środowisko testowe

W celu przeprowadzenia testów stworzono program realizujący algorytm Apriori. Do napisania programu wykorzystano środowisko Matlab. Program wyposażono w funkcję dyskretyzującą (do grupowania wartości atrybutów w zadane przedziały), oraz funkcję zapisującą odkryte reguły do plików w formacie XML.

Dyskretyzacja polegała na przyporządkowaniu wszystkich parametrów, poza ilością planet, do 10 przedziałów (z możliwością modyfikacji) według logarytmów ich wartości. Takie przyporządkowanie dobrze obrazowało naturę zjawisk. Liczby planet nie przyporządkowywano do przedziałów, gdyż parametr ten miał charakter dyskretny. Zaimplementowana funkcja zwraca tablicę zdyskretyzowanych wartości, oraz informację na temat granicy dolnej i górnej przedziałów, z wyjątkiem ostatniego przedziału, dla którego zwracana jest granica dolna.

Jako wynik działania programu generowane są dwa pliki xml. Jeden z plików zawiera definicję parametrów i wartości graniczne każdego z przedziałów, natomiast drugi zawiera odkryte reguły asocjacyjne. Format danych pliku z regułami jest zgodny ze standardem zapisu reguł RuleML. Natomiast definicja parametrów były zapisane w formie jak w poniższym przykładzie (warto zauważyć, jak zdefiniowano atrybuty zdyskretyzowane (grupy) i numeryczne).

```
<attributes>
  <attribute unit="pc" numerical="no" name="Odleglosc">
    <group max_value="0.000848" min_value="0" id="1"/>
    <group max_value="0.007195" min_value="0.000848" id="2"/>
    <group max_value="0.061029" min_value="0.007195" id="3"/>
    <group max_value="0.517659" min_value="0.061029" id="4"/>
    <group min_value="0.517659" id="5"/>
  </attribute>
  <attribute numerical="yes" name="Planety"/>
</attributes>
```

Poniżej przedstawiono przykład użycia zaimplementowanych funkcji (szersze opisy zamieszczono w prezentowanym kodzie w komentarzach):

```
% Wczytanie tablicy danych danych
load dane3;
% Definicja nazwy plików wyjściowych
outfile1='rules.ruleml.xml';
outfile2='parametrs.xml';
% Definicja nazw parametrów i ich jednostek
datacolumns=['Temperatura ';'Metalicznosc';'Promien     ';
    'Masa        ';'Gestosc     ';'Odleglosc   ';'Planety     '];
```

```
datacolumns = cellstr(datacolumns);
units=['K          ';'dex        ';'Solar_radii';
    'Solar_Mass ';'g/m^3      ';'pc         ';'           '];
units = cellstr(units);
% Definicja liczby przedziałów dyskretyzacji każdego z parametrów
nprzedzial=[10,10,10,10,10,10,0];
% Ustawienie minsupp i minconf
minsupp=0.20;
minconf=0.90;
% Dyskretyzacja
[ tablica_dyskre, zakresy ] = dyskretyzuj( tablica, nprzedzial );
% Wyznaczanie zbiorów częstych
[grupy,wsparcia_grup,indexy]=apriori(nprzedzial,tablica_dyskre,
 minsupp);
% Wyznaczanie reguł asocjacyjnych
[reguly, LP,WW]=generuj_reguly(grupy,wsparcia_grup,indexy,minconf);
% Zapisywanie reguł do plików
if reguly(1,1)~=0
  wypisz3(outfile1,reguly,LP,WW,nprzedzial,datacolumns,
        zakresy,units);
  wypisz3_grupy(outfile2,reguly,LP,WW,nprzedzial,datacolumns,
        zakresy,units);
end
```

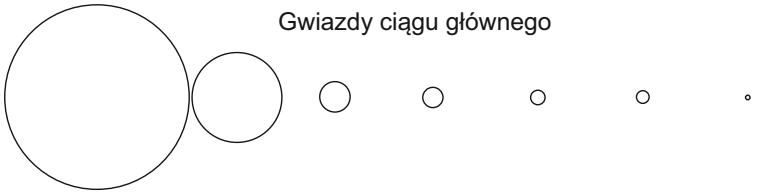
8.3.3. Wyniki testów

Podczas generowania reguł asocjacyjnych na bazie zadanego zbioru danych kluczowe jest dobranie wartości parametrów *suppmin* i *confmin* odpowiednio do natury zjawiska oraz sposobu implementacji.

W opisanym przypadku liczba potwierdzonych planet w zbiorze danych wyniosła 2329, przy czym nie oznacza to, że planety wystąpią w takiej liczbie rekordów. Gwiazdy zawierają od 1 do 7 potwierdzonych planet, więc liczba rekordów w bazie zawierająca planety jest jeszcze niższa. Poza tym sposób implementacji (podział każdego z M parametr na N_{dys} grup) powoduje, że przy obliczaniu wsparcia zbiorów częstych efektywna liczba parametrów wynosi $M \cdot N_{dys}$.

Przeprowadzenie testów przy $suppmin < 0.2; confmin < 0.9$ zakończyło się wygenerowaniem 11 reguł. Były to reguły zawierające jedynie związki pomiędzy masą, gęstością, promieniem, metalicznością i temperaturą. Przykładem takiej wygenerowanej reguły jest: $(Promień = 3, Masa = 7) \rightarrow (Temperatura = 4)$ z $supp = 0.32$ i $conf = 0.98$. Czyli jeżeli promień gwiazdy mieści się w przedziale od 0.51 do 1.14 promieni słońca oraz masa należy do przedziału od 0.83 do 1.21 mas słońca, to temperatura kształtować się będzie pomiędzy 5145K a 6545K. Gwiazdy o takich parametrach są dość często występującymi gwiazdami ciągu głównego, podobnymi do Słońca. Są to gwiazdy typu widmowego G, oraz K0 [3]. Pozostałe 10 reguł dotyczy tego samego typu gwiazd.

W celu poszukiwania zależności między parametrami samej gwiazdy a liczbą planet wykonano testy dla mniejszych wartości *suppmin* i *confmin*. Pożądanym efektem są reguły w formie (*parametry gwiazd*)→(*liczba planet*).



Typ widmowy:	O	B	A	F	G	K	M
Temperatura:	40 000K	20 000K	8 500K	6 500K	5 700K	4 500K	3 200K
Średnica (Słońce = 1):	10	5	1.7	1.3	1.0	0.8	0.3
Masa (Słońce = 1):	50	10	2.0	1.5	1.0	0.7	0.2
Jasność (Słońce = 1):	100 000	1000	20	4	1.0	0.2	0.01
Czas życia (10^6 lat):	10	100	1000	3000	10 000	50 000	200 000
Częstość występowania:	0.00001%	0.1%	0.7%	2%	3.5%	8%	80%

Rys. 8.1: Porównanie parametrów gwiazd ciągu głównego [3]

Pierwsze reguły wiążące w jakikolwiek sposób parametr *Planety* z pozostałymi parametrami uzyskano dla $suppmin < 0.001; confmin < 0.8$ (wygenerowano 1431 reguł). Przykładem takiej reguły jest: $(Promień = 4, Planety = 0) \rightarrow (Temperatura = 4)$ z $supp = 0.001035$ i $conf = 0.85$ co oznacza, że jeżeli *Promień* jest z zakresu od 1.13 do 2.52 promieni słońca i gwiazda nie posiada planet, to jej temperatura zawiera się w przedziale od 5145 do 6545 Kelwinów. Otrzymana reguła daje informację podobną do poprzednio opisywanej, z tą jednak różnicą, że tutaj gwiazda kwalifikowała by się raczej do typu F5 [3]. Pozostałe reguły wygenerowane w tym eksperymencie (zawierające planety) wiążą brak planet z różnymi parametrami gwiazdy w sposób identyczny do opisanego.

Eksperyment powtórzono dla $suppmin < 0.0001; confmin < 0.8$. Wygenerowano 4678 reguł asocjacyjnych. Ponownie nie znaleziono żadnych reguł w zakładanej formie. Wystąpiły natomiast reguły analogiczne do opisanych w poprzednim akapicie, jednak z parametrem *Planety* przyjmującym wartości od 0 do 3.

8.4. Podsumowanie

W rozdziale opisano przykład zastosowania autorskiej implementacji algorytmu Apriori do odkrywania reguł asocjacyjnych wiążących parametry gwiazd z występowaniem egzoplanet. Sama aplikacja pozwala na odkrywanie reguł asocjacyjnych w dowolnych zbiorach danych, jeśli tylko dane wejściowe zostaną odpowiednio przygotowane. Przygotowanie danych zgodnie z opisem zamieszczonym w podrozdziale 8.3.1 pozwoliło uruchomić aplikację na przykładach opisanych w podrozdziale 8.3.2. W przypadku danych, dla których dyskretyzacja logarytmiczna nie jest wskazana należy zmodyfikować funkcję dyskretyzującą.

Przeprowadzone badania pozwalają wysnuć wniosek, że algorytm Apriori w wersji standardowej nie pozwala znaleźć reguł asocjacyjnych wiążących parametry gwiazdy z występowaniem egzoplanet. Powodem jest znacznie niższe występowanie parametru *Planety* w stosunku do pozostałych parametrów. Wynika to z niewielkiego stosunku potwierdzonych egzoplanet do przebadanych gwiazd. Lepszym rozwiązaniem problemu byłoby zmodyfikowanie algorytmi Apriori po-

przez wprowadzenie różnych wartości *suppmin* dla różnych parametrów. Dzięki takiemu zabiegowi możliwe stałoby się wyznaczanie zbiorów częstych zawierające rzadko występujące parametry, bez generowania nadmiernej liczby pozostałych zbiorów. Warto wspomnieć, że metody badania obecności egzoplanet nie są doskonałe i prawdopodobnie ich występowanie jest znacznie częstsze.

W badaniach udało się wyznaczyć reguły potwierdzające wiedzę na temat typów widmowych gwiazd [3]. Najczęściej znajdowano gwiazdy z ciągu głównego. Największą częścią gwiazd w bazie danych teleskopu Keplera były gwiazdy typu G (według wygenerowanych reguł). Stanowią one około 3.5% gwiazd ciągu głównego, podczas gdy gwiazdy typu M (mniejsze, chłodniejsze i ciemniejsze) stanowią około 80% ciągu. Można byłoby wysunąć tezę, że wygenerowane wyniki nie pokrywają się z teoretycznym założeniem co do obecności gwiazd. Należy jednak pamiętać, że misją teleskopu Keplera jest odnajdywanie egzoplanet, których szukano przede wszystkim przy gwiazdach podobnych do Słońca. Poza tym jaśniejsze gwiazdy są łatwiejsze do detekcji i obserwacji.

Literatura

[1] Ames Research Center. Kepler. A Search for Habitable Palets. About the Mission. `http://kepler.nasa.gov/Mission/QuickGuide/`[dostęp dnia 10 czerwca 2016].

[2] NASA Exoplanet Science Institute. NASA Exoplanet Archive. `http://exoplanetarchive.ipac.caltech.edu/index.html`[dostęp dnia 10 czerwca 2016].

[3] R. Powell. The classification of stars. `http://www.atlasoftheuniverse.com/startype.html`[dostęp dnia 10 czerwca 2016].

[4] P.-N. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining*. Pearson, 2006.

Od redaktora i wydawcy

W niniejszej książce zebrano opracowania studentów II roku studiów magisterskich Wydziału Elektroniki Politechniki Wrocławskiej, kierunku Automatyka i robotyka, specjalności Robotyka, wykonane w semestrze letnim roku akademickiego 2015/2016 podczas realizacji prowadzonego przeze mnie kursu *Komputerowe przetwarzanie wiedzy*. Zawierają one opisy rożnych naukowych i inżynierskich problemów oraz metod i narzędzi stosowanych przy ich rozwiązywaniu. Opisom tym towarzyszą relacje z wykonanych projektów. Zakres poruszanych tematów można hasłowo podsumować w następujący sposób:

- systemy do prowadzenia inteligentnej konwersacji z człowiekiem,
- rozpoznawanie emocji w tekście,
- komunikacja człowieka z maszyną bez użycia słów, za pomocą gestów i obrazów,
- budowa słownika pojęć biznesowych,
- modelowanie stadnych zachowań,
- budowa prostego systemu parkowania,
- wykorzystanie reguł asocjacyjnych dla danych astronomicznych.

Mam nadzieję, że lektura tych opracowań okaże się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, październik 2016

ISBN 978-83-930823-8-4