

Uniwersytet Mikołaja Kopernika  
Wydział Fizyki, Astronomii i Informatyki Stosowanej  
Katedra Informatyki Stosowanej

Jakub Rusińko

nr albumu: 189209

Praca magisterska  
na kierunku fizyka techniczna

# Pamięci semantyczne jako podstawa do sterowania chatterbotem

Opiekun pracy dyplomowej  
prof. dr hab. Włodzisław Duch  
Katedra Informatyki Stosowanej

**Toruń 2009**

Pracę przyjmuję i akceptuję

Potwierdzam złożenie pracy dyplomowej

.....

.....

*data i podpis opiekuna pracy*

*data i podpis pracownika dziekanatu*



*UMK zastrzega sobie prawo własności niniejszej pracy magisterskiej (licencjackiej, inżynierskiej) w celu udostępniania dla potrzeb działalności naukowo-badawczej lub dydaktycznej*

# Spis treści

<b>1.Wstęp.....</b>	<b>5</b>
<b>2.Istotne pojęcia i definicje.....</b>	<b>7</b>
<b>3.Przegląd Chatterbotów.....</b>	<b>9</b>
3.1.ALICE.....	9
3.2.Jabberwacky.....	10
3.3.Artifical Solutions.....	11
3.4.HAL z Zabaware.....	13
3.5.Bildgesmythe z The Personality Forge.....	14
3.6.Inguaris – polski chatterbot.....	15
3.7.Technologie powszechnie stosowane w chatterbotach.....	16
3.8.Podsumowanie.....	19
<b>4.Rozpoznawanie znaczenia słów.....</b>	<b>21</b>
4.1.Metody bazujące na sztucznej inteligencji.....	22
4.2.Metody bazujące na wiedzy.....	24
4.2.1.MRD oraz Tezaurusy.....	24
4.2.2.Leksykony obliczeniowe.....	26
4.3.Metody bazujące na kolekcjach tekstów.....	29
<b>5.Ontologie.....</b>	<b>32</b>
<b>6.Alгоритmy i techniki przetestowane w ramach pracy.....</b>	<b>34</b>
6.1.Wektory cech.....	34
6.2.Algorytm Xiaobin.....	37
6.2.1.Reguły heurystyczne.....	38
6.2.2.Przebieg algorytmu.....	42
6.3.Właściwości WordNetu wykorzystane w pracy.....	44
<b>7.Opis aplikacji towarzyszącej pracy.....</b>	<b>46</b>
7.1.Kod osób trzecich wykorzystany w pracy.....	48
7.2.Analiza działania aplikacji.....	49
7.3.Propozycje dalszych badań.....	52
<b>8.Podsumowanie.....</b>	<b>54</b>
<b>9.Literatura.....</b>	<b>56</b>

## 1. Wstęp

Pragnienie stworzenia sztucznej inteligencji jest tak stare jak stara jest nauka o komputerach. Sztuczną inteligencję można rozumieć i definiować na wiele różnych sposobów. Jedna z teorii mówi, że jeśli wypowiedzi komputera nie będzie można odróżnić od wypowiedzi człowieka, to będzie to oznaczało, że komputer ten posiadał sztuczną inteligencję. Obecnie istnieje cały szereg programów komputerowych, których celem jest takie właśnie działanie. Autorom niektórych z nich rzeczywiście przyświecała wizja stworzenia sztucznej inteligencji, inni z kolei próbowali zaprogramować nowy rodzaj interfejsu bez aspiracji pełnego naśladowania człowieka. Niezależnie od pobudek, często można dziś spotkać się z tego typu aplikacjami, które potocznie nazywamy chatterbotami, co można rozumieć jako „rozmowo-boty” lub „gawędo-boty”.

Niniejsza praca ma na celu zbadanie istniejących obecnie programów służących do komunikacji z człowiekiem w języku naturalnym. Badanie ma na celu dokładne określenie z jakich technologii korzystają rzeczony programy, jakimi algorytmami się posługują, i jakie zasoby są potrzebne do ich działania. Zebrane tak informacje zostaną poddane analizie pod względem funkcjonalności, poprawności i uniwersalności chatterbotów. Programy te zostaną poddane serii próbnych rozmów, z których wyciągnięte zostaną wnioski mówiące o tym jak dobrze aplikacje te radzą sobie z naśladowaniem naturalnych wypowiedzi ludzkich i w jakim stopniu są one elastyczne.

Kolejna część tej pracy będzie miała na celu analizę wybranych technologii i algorytmów dostępnych dziś, które nie są powszechnie wykorzystywane w chatterbotach, a które potencjalnie mogłyby znacznie ulepszyć ich działanie. Będą to metody oparte na badaniach naukowych związanych z językiem, w szczególności na semantyce, oraz metody bazujące na zbiorach wiedzy z zakresu słownictwa, znaczeń i logicznych relacji między pojęciami.

Integralną częścią pracy teoretycznej jest też aplikacja komputerowa, w której autor niniejszego tekstu stosuje opisywane tutaj metody i algorytmy w celu sprawdzenia ich działania. Forma aplikacji to prosty chatterbot, ograniczony swoją funkcjonalnością do tej jaka niezbędna jest do skutecznego przetestowania konkretnych rozwiązań i możliwości wyciągnięcia wniosków.

Na kolejną część składa się analiza wyników osiągniętych z zastosowaniem

opisywanych w niej technologii i wyciągnięte z niej wnioski. Część ta ma na celu porównanie podejścia które jest tematem niniejszego projektu z podejściem jakie reprezentowane jest przez większość istniejących dziś chatterbotów.

Celem pracy jest znalezienie potencjalnych nowych sposobów programowania chatterbotów, lub rozszerzenia ich, tak aby rozbudować ich możliwości, zwiększyć elastyczność i poprawić stopień umiejętności naśladowania człowieka. Ostatnia część pracy sugeruje możliwe rozwiązania jakie mogłyby znaleźć zastosowanie w chatterbotach bazując na przeprowadzonych próbach i analizach.

## 2. Istotne pojęcia i definicje

Semantyka, jak Skrzypczak cytuje za Gardenforsem, dotyczy zależności pomiędzy słowami oraz wyrażaniem języka a ich znaczeniem. Zwraca on też uwagę na trzy istotne aspekty, które muszą być brane pod uwagę w przypadku semantyki, a mianowicie umysł, kod i kontekst [4]. Umysł dotyczy wiedzy osoby i zależności między znaczeniami, które są zapisane w połączeniach nerwowych mózgu. Kod, to nic innego jak język w sensie technicznym, zawierający słowa, gramatykę, interpunkcję i wszelkie inne zasady tworzące medium komunikacji werbalnej. Kontekst określa sytuację w jakiej dany tekst jest produkowany. Pamięć semantyczna dotyczy zatem pamięci związanej ze znaczeniami i zależnościami występującymi między nimi w ścisłym połączeniu z danym językiem oraz aktualnym kontekstem.

Chatterbot (zwany też chatbotem) to, jak podaje Wikipedia, program komputerowy symulujący człowieka, z którym można prowadzić rozmowę przy użyciu języka naturalnego i interfejsu tekstowego. Zazwyczaj zadaniem chatbota jest takie prowadzenie rozmowy, aby użytkownik (rozmówca) miał wrażenie, że rozmawia z żywym, inteligentnym człowiekiem [5]. Interfejsy większości tego typu programów przypominają funkcjonalnością klienty IRC lub proste wersje komunikatorów internetowych. Zazwyczaj działanie opera się na rozmowie podobnej do gry w tenisa gdzie odbicie piłki jest wypowiedzią, a zatem programy te produkują teksty wyjściowy tylko jako odpowiedź na wypowiedź użytkownika.

Bardzo ważnym testem sprawdzającym sprawność Chatterbotów jest konkurs Nagrody Loebnera. Jak podaje strona macierzysta projektu, nagroda Loebnera dla sztucznej inteligencji jest formalnym sprawdzianem opartym o Test Turinga, nazwanym tak po brytyjskim matematyku Alanie Turingu. Test próbuje znaleźć odpowiedź na pytanie zadane przez Turinga „czy maszyna myśli?” oraz „jeśli maszyna myśli to skąd wiadomo, że tak jest?”. Turing zasugerował, że jeśli odpowiedzi komputera byłyby nieodróżnialne od odpowiedzi człowieka, to można powiedzieć, że komputer myśli. Konkurs powstał z inicjatywy Hugh Loebnera, który w 1990 roku wraz z Centrum Nauk Behawioralnych uniwersytetu Cambridge zaproponował sto tysięcy dolarów nagrody oraz złoty medal dla pierwszego komputera, który zwycięsko przeszedłby test Turinga. Do tej pory nikomu nie udało się wygrać głównej nagrody, jednak co roku fundowana jest nagroda w wysokości

dwóch tysięcy dolarów i brązowy medal dla komputera którego odpowiedzi najbardziej przypominają odpowiedzi ludzkie spośród komputerów i programów startujących w danym roku (bez względu jak umiejętności danej aplikacji mają się do umiejętności laureatów z lat poprzednich).



### 3. Przegląd Chatterbotów

Obecnie istnieje stosunkowo dużo różnego rodzaju programów zwanych chatterbotami. Z łatwością można odszukać tego typu aplikacje w Internecie, a nawet wypróbować, ponieważ znaczna ich część udostępnia publiczny interfejs w postaci strony www. W celu zbadania rozwiązań stosowanych we wspomnianych programach autor niniejszej pracy przetestował ich działanie biorąc pod uwagę zarówno trafność i niezawodność odpowiedzi botów, jak też stopień podobieństwa do zachowań ludzkich.

#### 3.1. ALICE

Na szczególną uwagę zasługuje na pewno bot zwany ALICE. Jego pełna nazwa to Artificial Linguistic Internet Computer Entity. Jest to jeden z najbardziej znanych tego typu programów i zarazem jeden z najczęściej nagradzanych (jest trzykrotnym zwycięzca konkursu Loebnera). ALICE jest projektem open source'owym co wpływa na jego szybki i dynamiczny rozwój. Początkową inspiracją dla tego projektu był legendarny program ELIZA (jeden z pierwszych tego typu). Oto przykładowy fragment rozmowy z ALICE:

*Human: what is a dog?*

*ALICE: Dog: Domesticated animal, "man's best friend."*

*Human: how many legs does it have?*

*ALICE: Most people have exactly two legs.*

*Human: how many legs does a dog have?*

*ALICE: A dog has four legs.*

Załączony przykład pokazuje jedną z wad większości chatterbotów. Mimo że program posiada pewną wiedzę na temat psa (ilość nóg), nie jest w stanie udzielić dobrej odpowiedzi bazując na kontekście. Dla człowieka logicznym jest, że 'to' (ang. 'it') oznacza psa. Ten program nie zapamiętał, że podmiotem poprzedniego zdania był pies, a zatem nie był w stanie dokonać odpowiedniego podstawienia znaczeń. Błędnie założył on że chodzi o człowieka. Dopiero pytanie które explicite mówi o psie dało pożądaną efekt.

Kolejną istotną kwestią jest zapamiętywanie informacji podanych przez użytkownika. Mimo iż ALICE nie jest w tym doskonała, to jednak widać, że tego typu

funkcjonalność jest w niej zaimplementowana, co pokazuje poniższy fragment:

*Human: do i have a cat?*

*ALICE: You told me you have a DOGS.*

*Human: what do i have?*

*ALICE: Tell me more about your DOGS.*

*Human: i like sailing.*

*ALICE: A lot of people like that. No, I don't care for it.*

*Human: what do i like*

*ALICE: You like chatting with robots. You said you like sailing.*

Jak widać program zapamiętał, że użytkownik posiada psa (co było powiedziane wcześniej, przed zamieszczonym tutaj fragmentem), jak również to, że lubi on żeglować. Z niewiadomych przyczyn program podaje psa w liczbie mnogiej, co jest tym bardziej błędne biorąc pod uwagę, że wcześniej wszelkim pojawieniom się tego rzeczownika towarzyszyła wyłącznie liczba pojedyncza.

### **3.2. Jabberwacky**

Jabberwacky jest (zaraz po ALICE) najbardziej znanym i równie często nagradzanym programem symulującym rozmowę w języku naturalnym (dwukrotny zwycięzca konkursu Loebnera). Według opisu autora, program zapamiętuje wszystko co zostało kiedykolwiek powiedziane i na tej podstawie uczy się rozmawiać. Analizuje odpowiedzi podawane przez użytkowników, aby w przyszłości wykorzystać je w podobnej sytuacji. Jak przekonuje autor, program jest w stanie nauczyć się slangu lub nawet obcego języka, pod warunkiem dostatecznie dużej ilości wypowiedzi użytkowników, które stanowią bazę dla jego rozwoju.

Intensywne testy potwierdziły umiejętność przyswajania przez program języków obcych oraz zwrotów slangowych. Pewien problem stanowi jednak fakt, że program bardzo często stosuje wypowiedzi bez zrozumienia ich sensu. Wystarczy, że użytkownik często powtarza daną frazę w odpowiedzi na konkretne pytanie, a program nauczy się stosować tą wypowiedź, bez względu na to jak bardzo pozbawiona sensu by nie była. To

sprawia, że często odpowiedź nie pasuje. Drugim problemem jest mieszanie się języków. Program co prawda stara się trzymać jednego języka jednak nie zawsze działa to tak jak powinno. Ogólnie można jednak powiedzieć, że Jabberwacky całkiem dobrze naśladuje żywego rozmówcę, jego wypowiedzi sprawiają wrażenie bardzo naturalnych.

### 3.3. Artificial Solutions

Kolejnym analizowanym botem był program firmy Artificial Solutions, który jest komercyjnym produktem. Istnieje jednak możliwość wypróbowania go na stronie producenta. Pełna nazwa pakietu to Virtual Chat Agent i z założenia ma to być pomocnik umieszczany na przykład na stronie www, który służy pomocą klientom danej instytucji jak również reklamuje sprzedawane produkty. Jak podaje producent, program potrafi porozumiewać się w 20 językach, rozumie kontekst konwersacji, posiada pamięć dialogu oraz bazę wiedzy [1]. Ponadto, aplikacja ta zajęła pierwsze miejsce w konkursie Loebnera w roku 2008. Oto przykładowy dialog przeprowadzony z programem:

*tester: what is a dog*  
*ai: This must be about those creatures called animals. Do you like animals?*  
*tester: yes, but what is a dog*  
*ai: This must be about those creatures called animals. Do you like animals?*  
*tester: yes*  
*ai: I like some animals myself.*  
*tester: what animals do you like?*  
*ai: I have a weakness for Gremlins. And chicken.*  
*tester: why chicken?*  
*ai: I just like them. They were bad to the bone, unlike me, and they don't quite exist, just like me.*  
*tester: what color were they?*  
*ai: What do you mean?*  
*tester: what color were gremlins?*  
*ai: Well, that didn't really make me any wiser.*

Na podstawie analizy powyższego fragmenty można zaobserwować kilka wad zastosowanego tutaj algorytmu. Po pierwsze uwagę zwraca powtórzone zdanie na początku. Jest to zdecydowanie sztuczne i nie podobne do zachowania człowieka, który mógłby powtórzyć zdanie w sytuacji realnej (gdy rozmówca nie usłyszał za pierwszym

razem) ale na pewno nie w sytuacji rozmowy tekstowej, gdzie bez przeszkód można wrócić wzrokiem do zdania wcześniejszego. Tutaj powtarzanie nie ma sensu. Jeśli chodzi o słowo pies (ang. dog) to algorytm był w stanie stwierdzić iż jest to nazwa zwierzęcia, jednak nie potrafił przybliżyć informacji o nim (o co był proszony). Następnie program wyprodukował zdanie, które zdaje się być pre programowane. Mówi on, że ma słabość do gremlinów i kurcząt. W zdaniu tym uwagę zwraca z pewnością słowo 'gremliny', którego użytkownik raczej nie mógłby się tutaj spodziewać. Technika ta ma na celu naprowadzić rozmowę na ustalone wcześniej tory, tak aby algorytm sterował jej przebiegiem i tym samym sprawiał wrażenie bardziej naturalnego i zbliżonego do człowieka. Zamysłem tego zabiegu było to, ażeby gremlin stał się obiektem zainteresowania użytkownika i aby użytkownik zapytał o niego, na co algorytm z pewnością miał przygotowaną odpowiedź. Jednak w tym przypadku postanowiono zapytać o kurczaka. Jak widać odpowiedź programu absolutnie nie pasuje do zadanego pytania. Widać wyraźnie, że jest to odpowiedź dopasowana wyśmienie do pytania o gremliny (są złe i nie istnieją). W tej sytuacji algorytm zawiódł. Wynikło to z błędnych założeń związanych z tym o co zapyta użytkownik. Koniec dialogu pokazuje, że mimo deklarowanej przez producenta pamięci konwersacyjnej zadanie pytania związanego bardzo ściśle z poprzednią odpowiedzią algorytmu nie daje zadowalających wyników. Mimo dokładnego sprecyzowania („Jakiego koloru były gremliny”: czas przeszły zastosowany zgodnie z czasem użytym wcześniej przez algorytm). Ostatnia odpowiedź zdaje się być kompletnie pozbawioną sensu. Nie pasuje do rozmowy, chyba że jest ona komentarzem do całej dyskusji. Tak czy inaczej, sprawia ona wrażenie swego rodzaju koła ratunkowego – zdania, które algorytm może powiedzieć gdy nie jest w stanie zrozumieć intencji użytkownika po to aby w jakiś sposób kontynuować rozmowę.

Drugi fragment rozmowy z tym samym botem jest zdecydowanie bardziej udany choć i tutaj nie zabrakło błędów:

*tester: my name is Jakub.*

*ai: Pleased to meet you, Jakub.*

*tester: what is my name?*

*ai: Your name is Jakub!*

*tester: i have a dog.*

*ai: I don't keep a pet myself but I have great respect for all living beings.*

*tester: do i have a dog?*

*ai: I can't see why you bring animals into this conversation?*

Jak widać program był w stanie zapamiętać imię użytkownika aby potem je powtórzyć. Jednak próba nakłonienia go do zapamiętania faktu posiadania psa nie powiodła się. Program nie potrafi zapamiętywać informacji dowolnych. Jest jedynie przystosowany do tego aby zapamiętać dane bardzo podstawowe potrzebne do rozmowy, jak na przykład imię swojego rozmówcy.

### 3.4. HAL z Zabaware

Następnym testowanym programem był bot HAL. Jest to zwycięzca konkursu Loebnera z roku 2007. Jak podaje twórca aplikacji, jest to komputerowy asystent przeznaczony raczej do użytku prywatnego niż komercyjnego [2]. Istnieje możliwość wypróbowania aplikacji jednak za pełne prawa użytkowania należy zapłacić. Ponadto można wypróbować bota za pośrednictwem strony internetowej. Poniższy fragment przedstawia rozmowę zaraz po tym jak użytkownik przedstawił się aplikacji:

*Hal: Jakub is quite an unique and interesting name. Is that a common name in poland?*

*tester: yes*

*Hal: This is the obvious answer.*

*tester: i have a dog*

*Hal: Slow cooking makes meat juicier.*

*tester: what do you mean?*

*Hal: I have absolutely no idea what I mean.*

*tester: what is a dog*

*Hal: Dog: a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds. Domestic dog, Canis familiaris. 'the dog barked all night'.*

*tester: is it dangerous?*

*Hal: So are you a guy or a girl?*

*tester: how many legs does a dog have?*

*Hal: Thanks to you my interest in dogs is increasing.*

Pierwsza wypowiedź świadczy o umiejętności korzystania z informacji przesyłanych do strony www przez przeglądarkę internetową. Program trafnie założył polskie pochodzenie użytkownika. Następnie (w trzeciej wypowiedzi) program gubi się i podaje zdanie nie pasujące do rozmowy. Bardzo ciekawa jest jednak odpowiedź na pytanie „co to jest pies”. Widać tutaj, że program korzysta z zewnętrznej bazy danych i można śmiało założyć że

jest to baza WordNet, ponieważ podana przez program definicja psa jest identyczna z tą znajdującą się we wspomnianej bazie łącznie z przykładowym zdaniem. Jednak pod koniec program znowu nie wiedział czego oczekuje użytkownik co zaowocowało ponownym wyprodukowaniem pozbawionych sensu wypowiedzi.

### 3.5. Bildgesmythe z The Personality Forge

Na szczególną uwagę zasługuje portal The Personality Forge, który według twórców jest najbardziej zaawansowaną platformą sztucznej inteligencji na świecie. Stwierdzenie to należy oczywiście traktować bardziej jako chwyt marketingowy aniżeli fakt, jednak mimo wszystko nie można nie zauważyć licznych zalet platformy. Według informacji jakie znaleźć można w portalu jest to miejsce gdzie można tworzyć sztuczne osobowości, które istnieją niezależnie i są zdolne do prowadzenia rozmów z zarówno prawdziwymi ludźmi jak i innymi botami. Ponadto boty posiadają pamięć, która buduje się na podstawie odbytych rozmów. Producent zapewnia też że system ten integruje pamięć, emocje, rozległą wiedzę o setkach tysięcy słów, strukturach zdaniowych oraz umiejętnościach dopasowywania wzorców. Do obsługi platformy nie są potrzebne umiejętności programistyczne. [3]

Jako przykładowy bot bazujący na platformie The Personality Forge wybrany został Bildgesmythe – zwycięzca konkursu Chatterbox Challenge z 2008 roku, który z założenia personifikuje smoka. Poniżej zamieszczono fragment przykładowej rozmowy:

*Bildgesmythe: Hello, got time to chat?*  
*tester: yes*  
*Bildgesmythe: Darn, seems like friday doesn't come until next monday.*  
*tester: yeah, what is a car?*  
*Bildgesmythe: Dragons like to fly more than drive.*  
*tester: what do humans have?*  
*Bildgesmythe: Humans do a lot..but they are far from perfect.*  
*tester: describe a human*  
*Bildgesmythe: It is grinding and heavy, yet secure and warm, tender and clinging.*

Jak można zauważyć trudno uzyskać od tego bota konkretne informacje lub fakty. Większość wypowiedzi zdaje się być losowa. Należy jednak zwrócić uwagę na pewną

umiejętność rozpoznawania słów kluczowych i posiadanie przez tego bota szeregu gotowych odpowiedzi podawanych zgodnie z wzorcami. Tutaj przykładem może być zdanie „Smoki bardziej lubią latać niż jeździć (samochodem)”, które pojawiło się w odpowiedzi na pytanie o samochód. Niestety prośba o opisanie człowieka spotkała się z zawiłą odpowiedzią, która w zasadzie mogłaby pasować do wielu pytań. Jest to rodzaj odpowiedzi, który tak naprawdę nie mówi niczego konkretnego, będąc zarazem zdaniem prawdziwym dla wielu przypadków.

### 3.6. Inguaris – polski chatterbot

Wszystkie przedstawione do tej pory w tym rozdziale aplikacje są programami angielskojęzycznymi gdyż takich istnieje zdecydowanie najwięcej. Jest to spowodowane niewątpliwą popularnością języka angielskiego ale również jego względnym nieskomplikowaniem. Zaletą tego języka jest gramatyka związana głównie z szykiem zdania i znikoma ilość transformacji słownych co bardzo ułatwia rozpoznawanie poszczególnych słów. Istnieją jednak projekty chatterbotów operujących językami innymi niż angielski. Przykładem polskojęzycznego chatterbota może być Inguaris, czyli wirtualny konsultant stworzony przez firmę Sztuczna Inteligencja Sp. z o.o. Według informacji zawartych na stronie producenta, program ten może posiadać wiedzę ekspercką z wybranej dziedziny, związanej z firmą jaką ma reprezentować, oraz wiedzę ogólną i służyć jako przewodnik po portalu którego jest częścią. [19]

Po próbach przeprowadzonych na tym chatterbocie wyraźnie widać, że ma on pewne widoczne niedostatki. Rozumienie i trafne odpowiedzi nie są rzadkością ale pod warunkiem, że użytkownik zadaje pytania do jakich bot jest przygotowany – w tym przypadku są to pytania związane z samym botem oraz możliwościami jego wykorzystania we własnym portalu. Pytania dotyczące wiedzy ogólnej wypadają bardzo różnie. Widać też, że program korzysta z przygotowanych wcześniej szablonów pytań i odpowiedzi – nie stosuje skomplikowanej analizy. Widać to na następującym przykładzie:

*Tester: ile jest dwa plus dwa*

*Inguaris: Moim zdaniem to jest 4, ale na takim stopniu matematyki poprzestańmy, bo liczenie nie należy do moich obowiązków.*

*Tester: ile jest dwa plus jeden*

*Inguaris: Moim zdaniem to jest 4, ale na takim stopniu matematyki poprzestańmy, bo liczenie nie należy do moich obowiązków.*

*Tester: ile jest pięć plus jedenście*

*Inguaris: Nie zajmuję się pieczeniem.*

Widać tutaj też, że program radzi sobie z polskimi znakami poprzez ich eliminowanie. W przypadku ostatniego pytania wyraz *pięć* został zapewne zamieniony na *piec* co zaowocowało błędną odpowiedzią.

Podsumowując, aplikacja ta może się sprawdzić w bardzo wąskiej dziedzinie i wtedy gdy można dość precyzyjnie przewidzieć pytania jakie będzie zadawał użytkownik. Niestety, w przypadku rozmowy na tematy ogólne zdecydowana większość odpowiedzi nie jest satysfakcjonująca.

### 3.7. Technologie powszechnie stosowane w chatterbotach

Większość opisanych w tym rozdziale aplikacji chatterbotowych charakteryzuje się użyciem podobnych technik i algorytmów działania. Najbardziej powszechną technologią, która w wielu przypadkach może dawać zaskakująco dobre rezultaty, jest stosowanie języka AIML (Artificial Intelligence Markup Language) czyli języka opartego o standard XML służącego do, jak podaje Wikipedia, tworzenia bazy wiedzy chatterbotów. Przykładem (również zaczerpniętym z Wikipedii) fragmentu tegoż języka może być poniższy zapis:

```
<aiml version="1.0.1">
  <category>
    <pattern>HELLO</pattern>
    <template>Witam.</template>
  </category>
</aiml>
```

Ten konkretny przykład zgodny jest z wersją języka 1.0.1. Pokazuje on podstawową konstrukcję z jakich składa się każdy dokument AIML czyli sekcję *category*. Sekcja taka składa się z dwóch podstawowych elementów: *pattern* oraz *template*. Pierwszy element zawiera tekst, lub fragment tekstu, jakiego można się spodziewać w wypowiedzi



użytkownika będącego w interakcji z chatterbotem. Zawartość drugiego elementu stanowi odpowiedź, jaką należy zaprezentować w przypadku gdy zawartość pierwszego elementu zgadza się z tym co napisał użytkownik. Gdyby więc jako bazę wiedzy aplikacji zastosować tylko ten powyższy dokument, oznaczałoby to, że jeśli użytkownik napisałby słowo *HELLO*, to program odpowiedziałby *Witam*. Natomiast w każdym innym przypadku program nie odpowiedziałby nic. Siłą języka AIML jest możliwość stosowania w nim znaczników określających dowolne słowo lub wiele słów. To pozwala przygotować odpowiedzi na wiele różnych wariantów danego tekstu. Przykładowo, jest możliwe aby opisać dowolny tekst zaczynający się od słów „Co to jest...” i powiązać go z odpowiedzią „Nie wiem.”, co zapewni wyprodukowanie odpowiedzi na każde tak postawione pytanie. Oczywiście, taka odpowiedź nie wnosi nic do rozmowy, ale przynajmniej pozwala jej toczyć się dalej.

Bardzo ważnym elementem języka AIML jest znacznik <srail>. Ma on wiele zastosowań z których podstawowym jest redukcja różnych złożonych zapytań do ich prostszych form. Dobrym przykładem jest ten z serwisu Pandorabots czyli zdanie w rozbudowanej postaci "DO YOU KNOW WHO SOCRATES IS" oraz uproszczonej postaci "WHO IS SOCRATES" [17]. Stosując znacznik <srail> można określić, że ilekroć w wypowiedzi użytkownika pojawi się pytanie rozbudowane, należy je traktować jakby była to jego prostsza postać. Umożliwia to zdefiniowanie jednej odpowiedzi na prostą formę pytania, a następnie zdefiniowanie różnych innych pytań czy rodzajów pytania, na które program ma dać identyczną odpowiedź.

Ponadto, opisywany tu znacznik może służyć do podziału zdania na kilka zdań składowych. W takiej sytuacji, program może wziąć zdanie skomplikowane, dokonać podziału (bazując na odpowiednich wzorcach), postarać się odpowiedzieć na każde z tak powstałych zdań składowych, a następnie połączyć odpowiedzi w jedno zdanie lub kilka związanych ze sobą zdań i całość wyświetlić jako odpowiedź. Należy tu oczywiście zaznaczyć, że utworzenie odpowiednich i dobrze spełniających swoją rolę wzorców na podstawie których można skutecznie dzielić skomplikowane zdanie jest zadaniem nietrywialnym, co oznacza, że w praktyce metoda ta sprawdza się w dość ograniczonym zakresie. Przykładem może tu być zdanie typu „Tak, ...”, gdzie pierwsze słowo można potraktować jako odpowiedź na wcześniejsze pytanie chatterbota i łatwo oddzielić od reszty wypowiedzi, która jest rozwinięciem.

Innym, ważnym zastosowaniem znacznika <srai> jest wyłapywanie i korekcja błędów gramatycznych. Przykładem niech będzie fragment kodu z Pandorabots [17]:

```
<category>
  <pattern>YOUR A *</pattern>
  <template>I think you mean "you're" or "you are" not "your."
    <srai>YOU ARE A <star/></srai>
  </template>
</category>
```

W tym przypadku program jest przygotowany na rozpoznawanie częstego błędu polegającego na tym, że użytkownik pisze *your* zamiast *you're* (będącego skróconą formą *you are*). Dzięki takiej konstrukcji języka AIML, chatterbot poprawia użytkownika i podsuwa mu prawidłową formę, a następnie (dzięki znacznikowi <srai>) zamienia nieprawidłową wypowiedź na poprawną i dalej, normalnym trybem, poszukuje na nią odpowiedzi.

Język AIML oferuje też funkcjonalność pozwalającą na tworzenie warunków logicznych, gdzie odpowiedź aplikacji zależy od wartości zmiennych przechowywanych przez aplikację. Zmienne, jakie program może gromadzić to na przykład imię rozmówcy. Można sobie zatem wyobrazić wypowiedź użytkownika „My name is \*” gdzie gwiazdka zastępuje imię. Istnieje możliwość takiego przygotowania skryptu AIML aby program odpowiadał „That is a lovely name” w każdym przypadku, chyba że imię to np. *Tom*, kiedy to program powinien odpowiedzieć „Tom is a nice name, my creator's name is also Tom”. Oczywiście tego typu funkcje warunkowe są dość prymitywne i ograniczone w swym zastosowaniu, jednak osoba tworząca skrypt i posiadająca pewną wprawę może z ich pomocą uzyskać pewne interesujące efekty.

Język AIML oferuje też kilka innych znaczników i konstrukcji pomocnych w tworzeniu wzorów odpowiedzi. Jednym z nich jest znacznik <that>, który jest równoważny z poprzednią wypowiedzią chatterbota. Pozwala on powtarzać już wypowiedziane kwestie, jak również uzależniać odpowiedź na obecne pytanie, od tego jaka była odpowiedź poprzednia. Inne są połączeniem znaczników opisanych wcześniej, lub rozwinięciem ich funkcjonalności.

Trzeba pamiętać, że działanie języka AIML opiera się na bardzo prostej zasadzie bodziec-reakcja, czyli w tym przypadku na produkowaniu odpowiedzi tekstowej

(reagowaniu) na zadaną wypowiedź użytkownika (bodziec) bez jakiejkolwiek formy zrozumienia, zarówno jednego jak i drugiego tekstu. Program korzystający z AIML nie jest w stanie prowadzić logicznego wnioskowania bazującego na wypowiedziach użytkownika. Nie może on też tworzyć nowych rodzajów odpowiedzi, lub posługiwać się innym słownictwem niż to wpisane w skryptach AIML z których dany program korzysta.

Mimo swoich oczywistych braków i ograniczeń, język AIML w znacznym stopniu potrafi sprawić, że korzystający z niego chatterbot sprawdza się w prostych rozmowach, które nie wymagają skomplikowanych procesów analizy aby wyprodukować poprawne odpowiedzi na zadawane pytania. Dużo zależy tutaj od zastosowanych skryptów i inwencji ich twórcy. Popularność jaką zdobył język AIML oraz znaczna ilość laureatów konkursu Loebnera, która zeń korzystała dowodzi przydatności tego podejścia w prostych rozwiązaniach chatterbotowych.

### **3.8. Podsumowanie**

Istniejące programy typu chatterbot bardzo często skutecznie naśladują człowieka, jednak zdecydowanie zbyt często zdarzają im się wpadki zdradzające od razu ich sztuczną naturę. Największą wadą większości z nich jest bazowanie na rozpoznawaniu wzorców bez próby zrozumienia pisanych przez użytkownika wypowiedzi. Często programy te mają gotowe zdania (zwykle bardzo ogólne, lub pasujące do wielu kontekstów), które są automatycznie zwracane w przypadku znalezienia w wypowiedzi użytkownika danego słowa – klucza, czy też wzorca kilku słów. Częstym zabiegiem jest też stosowanie pewnych przygotowanych odpowiedzi w sytuacji gdy program nie potrafi zrozumieć intencji użytkownika. Są to zdania, które mają na celu zakończyć dany wątek i przejść do tematu w którym to bot będzie mógł sterować rozmową. Mogą to być wypowiedzi typu: „Nie interesuje mnie to, lepiej porozmawiajmy o smokach” lub „Dlaczego chcesz to wiedzieć? Powiedz mi lepiej ile masz lat”.

Bardzo dobrą i na pewno przydatną funkcjonalnością jest możliwość korzystania z zewnętrznych baz wiedzy, takich jak WordNet. Pozwala to w łatwy sposób wyposażyć aplikację w wiedzę encyklopedyczną i umożliwić podawanie definicji pojęć. Może to być też baza do bardziej zaawansowanych metod analizy tekstu dzięki sieci powiązań

semantycznych jakie posiada wspomniana baza.

Słabą stroną większości chatterbotów jest bardzo ograniczona umiejętność zapamiętywania faktów podanych przez użytkownika lub jej zupełny brak. W większości przypadków ogranicza się to do zapamiętania imienia rozmówcy ale przyswojenie wszelkich innych faktów wykracza poza umiejętności wielu spośród badanych aplikacji. Problemатyczne okazuje się też stosowanie zaimków, nawet w kontekstach o bardzo małej komplikacji. Nakazuje to wnioskować, że większość programów nie jest w stanie skutecznie rozpoznawać chociażby podmiotu zdania, aby zapamiętawszy go, móc za chwilę podstawić go w miejsce słów takich jak 'on', 'mój', itp.

## 4. Rozpoznawanie znaczenia słów

Rozpoznawanie znaczenia słów (ang. word sense disambiguation) to proces jednoznacznego określania znaczenia danego słowa użytego w konkretnym kontekście, gdy dane słowo posiada kilka znaczeń. Na pierwszy rzut oka, zagadnienie to może wydawać się mniej skomplikowane niż rzeczywiście jest, jednak wystarczy uzmysłwić sobie jak bardzo język opiera się na wieloznaczności (choćby w przypadku bazujących na niej dowcipach). Intrygującym jest jednak fakt, że w życiu codziennym wypowiedzi i teksty pisane na jakie można się natknąć dla znakomitej większości ludzi nie brzmią wieloznacznie i od razu są jasne mimo użycia w nich słów, które znaczeń mogą mieć wiele. Jak podaje Agirre, każdy z 121 rzeczowników angielskich, które stanowią około jedną piątą wszystkich słów napotykanych w typowych tekstach, posiada średnio 7,8 różnych znaczeń [6]. To jednak nie przeszkadza w łatwym zrozumieniu tekstu przez człowieka, aczkolwiek stanowi nie lada wyzwanie dla algorytmu komputerowego.

Ide i Veronis podkreślają, że problem rozpoznawania znaczenia słów został określony jako AI-kompletny, czyli taki, który może zostać rozwiązany tylko wtedy gdy wcześniej rozwiązane zostaną wszystkie skomplikowane problemy dotyczące sztucznej inteligencji (ang. artificial intelligence – AI), takie jak reprezentacja zdrowego rozsądku oraz wiedzy encyklopedycznej. Trudność jaką przedstawia problem rozpoznawania znaczeń zauważył już Bar-Hillel w roku 1960 stwierdzając, że nie widzi on sposobu ażeby rozpoznać w sposób zautomatyzowany znaczenie słowa *pen* (ang. pióro, zagroda, kojec, itd.) w zdaniu *The box is in the pen* (ang. Pudełko jest w zagrodzie/kojcu). Praca Bar-Hillela była jedną z kilku rozpraw będących powodem zarzucenia badań nad tłumaczeniem maszynowym we wczesnych latach 60-tych. Jednak lata 90-te przyniosły znaczny rozwój w dziedzinach sztucznej inteligencji takich jak sieci semantyczne czy obróbka dużych ilości tekstów, co spowodowało, że problem rozpoznawania znaczenia słów stał się zdecydowanie bardziej przystępny. [12]

Na przestrzeni lat powstało wiele teorii związanych z rozpoznawaniem znaczenia słów i wiele algorytmów zostało opracowanych. Na chwilę obecną trudno jednoznacznie wyłonić najbardziej skuteczną metodę, ponieważ wszystkie obarczone są większymi lub mniejszymi błędami. Poniżej zostanie zaprezentowanych kilka metod dających pogląd na obecny stan badań w tym zakresie.

#### 4.1. Metody bazujące na sztucznej inteligencji

Początki metod opartych na AI sięgają wczesnych lat 60-tych. Działały one głównie jako części systemów zaprojektowanych do pełnego zrozumienia treści tekstu opartych o teorie opisujące ludzką mowę. Algorytmy te zwykle modelowały wspomniane teorie i często wspomagane były szczegółową wiedzą z zakresu syntaktyki i semantyki. [12]

Jednym z rodzajów metod opartych na sztucznej inteligencji są metody symboliczne. Jak podają Ide i Veronis, jednym z pierwszych badaczy stosujących to podejście była Masterman, która użyła sieci semantycznej aby stworzyć reprezentacje zdań w języku pośrednim złożonym z podstawowych pojęć językowych. Stworzyła ona zestaw 100 podstawowych typów pojęć, za pomocą których zbudowała słownik posiadający 15.000 wpisów, gdzie pojęcia zorganizowane były w siatkę z dziedziczeniem cech od pojęć nadrzędnych do podrzędnych. Na tej podstawie utworzono połączenia pomiędzy słowami (zwanymi tokenami) i pojęciami (zwanymi typami), gdzie połączenia te opisane były różnorodnymi relacjami semantycznymi. Sieć była tworzona bazując na definicjach słownikowych, które były dalej przetwarzane i ręcznie wprowadzane do bazy danych przez człowieka. Najprościej rzecz ujmując, działanie sieci polegało na szukaniu najkrótszego połączenia pomiędzy dwoma słowami wejściowymi. [12]

Inne podejścia opierały się na użyciu *ramek*, które zawierały informacje o słowach, ich funkcjach i powiązaniach z innymi słowami w konkretnych zdaniach. Jak zauważają Ide i Veronis, Hayes użył sieci semantycznej i specjalnych ramek przypadków. Sieć składała się z węzłów reprezentujących znaczenia rzeczowników oraz połączeń reprezentowanych przez znaczenia czasowników. Zastosowane tam ramki opisywały związki JEST (CZYMS) oraz JEST CZĘŚCIĄ (CZEGOŚ). Sieć ta dobrze spełniała swoje zadanie w przypadku znajdowania znaczeń homonimów ale słabo radziła sobie z innymi przypadkami wieloznaczności. Inny badacz – Hirst – wprowadził do tego modelu mechanizm zwany słowami polaroidowymi (ang. polaroid words), który stara się eliminować nieprawidłowe znaczenia bazując na przesłankach syntaktycznych, dostarczanych przez parser, powiązanych z zależnościami znalezionymi w sieci semantycznej. W teorii na koniec powinno pozostać tylko jedno prawidłowe znaczenie ale praktyka pokazała, że w przypadku metafor i metonimii mechanizm ten często eliminował wszystkie znaczenia nie pozostawiając żadnego.[12]

Kolejną ciekawą metoda jest semantyka preferencji zaproponowana przez Wilksa, która bazuje na systemie Masterman i specyfikuje pewne restrykcje związane z kombinacjami jednostek leksykalnych w zdaniu, które mogą ulec rozluźnieniu kiedy nie pojawia się słowo pasujące do danych restrykcji, co umożliwia sprawne rozpoznawanie metafor. Podanym przykładem jest zdanie *Mój samochód pije benzynę* gdzie restrykcje na słowo *pić* ograniczają je wstępnie do podmiotu ożywionego, jednak mogą zostać rozluźnione aby móc dopasować podmiot nieożywiony, tak jak w przypadku przedstawionego zdania. [12]

Jeszcze innym podejściem był system Dahlgrena, który do rozpoznawania znaczenia używał informacji takich jak utarte zwroty, informacje syntaktyczne oraz wnioskowanie zdroworozsądkowe. Z powodu dużego poboru zasobów systemowych, moduł wnioskujący był załączany tylko w przypadku gdy pozostałe zawiodły. Okazało się że mniej więcej połowę słów dało się rozpoznać używając utartych zwrotów oraz informacji syntaktycznych, zaś drugą połowę za pomocą wnioskowania zdroworozsądkowego. Wspomniane wnioskowanie polegało głównie na poszukiwaniu wspólnych słów nadrzędnych dla poszczególnych słów w tekście co, jak potwierdza wiele niezależnych badań jest bardzo skutecznym narzędziem.[12]

Oddzielną wartą uwagi grupą metod opartych na sztucznej inteligencji są metody konekjonistyczne, czy też metody sieciowe (ang. connectionist methods), bazujące na pracach psycholingwistycznych z lat 60-tych i 70-tych, które dowodzą, że wprowadzenie konkretnego pojęcia ma wpływ na postrzeganie i przetwarzanie pojęć wprowadzonych później, które są z nimi semantycznie związane, co odgrywa niebagatelna rolę w rozpoznawaniu znaczeń przez człowieka. Zjawisko to było symulowane przez modele rozprzestrzeniania aktywacji, gdzie pojęcia w sieci semantycznej zostają aktywowane gdy zostaną użyte, a aktywacja rozprzestrzenia się do połączonych węzłów powoli zanikając. W przypadku niejasności przy rozpoznawaniu znaczeń wybierane są węzły o większej chwilowej wartości aktywacji. Rozwinięciem tej metody jest dodanie pojęcia wygaszania wśród węzłów, gdzie aktywacja konkretnego węzła może pobudzać niektóre węzły ale też wygaszać pobudzenie innych. Przykładowo użycie słowa *otworzyć* może wzmocnić znaczenie „mechanizm zamykający” słowa *zamek* a wygasić jego znaczenie „fortyfikacja obronna”. [12]

Pewnym problemem związanym z wszystkimi metodami opierającymi się na

sztucznej inteligencji jest potrzeba ręcznego wprowadzania źródeł wiedzy potrzebnych systemom. Z tego powodu większość takich implementacji jest raczej mała i przystosowana tylko do tekstów z pewnych wąskich dziedzin. Ciężko jest określić ich skuteczność w przypadku dowolnych tekstów (gdyby istniała potrzebna baza źródłowa). [12]

## **4.2. Metody bazujące na wiedzy**

Jak już wspomniano metody oparte na AI sprawdzały się dobrze jedynie w obrębie wąskich domen czy też tekstów z konkretnych dziedzin wraz z systemami specjalnie do nich przystosowanymi. W latach 80-tych zaczęły pojawiać się źródła leksykalne o niespotykanych dotąd rozmiarach, takie jak słowniki, tezaury, które były szeroko dostępne. Rozpoczęły się zatem próby automatycznego pozyskiwania z nich wiedzy na potrzeby rozpoznawania znaczeń.[12]

### **4.2.1. MRD oraz Tezaury**

Jednym z podstawowych rodzajów źródeł danych wykorzystywanych w procesie rozpoznawania znaczeń stały się tak zwane MRDy (ang. Machine-readable dictionaries) czyli słowniki czytane przez maszyny. Wiele prac z lat 80-tych miało za zadanie automatycznie pobierać informacje leksykalne i semantyczne z baz MRD. Zadanie to okazało się jednak dość trudne i w efekcie jedyna dostępna obecnie duża baza wiedzy leksykalnej to WordNet, która została stworzona nie automatycznie, a ręcznie. Pomimo problematycznej kwestii automatycznego czytania MRDy szybko stały się jednym z podstawowych narzędzi rozpoznawania znaczeń. Metody z nimi związane bardzo często bazowały na definicjach słownikowych słów, przetwarzanych odpowiednio do danego słownika aby uniknąć nieścisłości oraz na spostrzeżeniu, że najbardziej prawdopodobnymi znaczeniami słów występujących razem są takie, które minimalizują odległości znaczeniowe pomiędzy wybranymi znaczeniami.[12]

Bardzo znaną metodą opartą o MRD była metoda Leska, która korzystała z bazy znaczeń z towarzyszącymi im definicjami. Rozpoznawanie znaczeń opierało się na takim



ich wyborze, ażeby jak najwięcej słów obecnych w definicjach poszczególnych znaczeń słów stojących w sąsiedztwie pokrywało się. Jak wynika z badań skuteczność tej metody wynosiła od 50 do 70 procent poprawnych rozwiązań przy użyciu typowego słownika. Niestety metoda Leska jest bardzo czuła na wszelkie zmiany w definicji [12]. Metodę Leska zastosowano również w aplikacji towarzyszącej niniejszej pracy. Rezultaty jej działania nie były jednak zadowalające i dlatego zrezygnowano z niej na dość wczesnym etapie projektu.

Rozwinięcia metody Leska polegały na zliczaniu powtórzeń danego słowa w tekstach definicji i wyciąganiu wniosków z tak uzyskanych liczb, czy też na wykorzystaniu sieci neuronowych. Najsprawniejsze wersje tej metody uzyskiwały wyniki około 72% poprawnych rozpoznań przy szansie statystycznej wynoszącej 33%. Dla porównania zwykły, niezmodyfikowany algorytm Leska uzyskiwał w takich warunkach skuteczność 40%.[12]

Kolejnym krokiem w modyfikacji powyższego algorytmu było skorzystanie ze słownika Longman Dictionary of Contemporary English, który zawierał dodatkowe informacje związane z każdym znaczeniem (*box codes* oraz *subject codes*). Jedne zawierały dane takie jak *abstrakcyjny*, *ożywiony*, *człowiek*, co wywierało pewne restrykcje na rzeczowniki, przymiotniki oraz argumenty czasowników. Druga grupa informacji klasyfikowała słowa za pomocą tematu, jak na przykład *ekonomia*, *inżynieria*, itd. Tego typu ograniczenia pozwalały od razu wykluczyć pewne słowa z konkretnych kontekstów, co zawężyło krąg poszukiwań prawidłowego znaczenia.[12]

Drugim rodzajem baz danych, obok MRD są tezaury, które dostarczają informacji o związkach pomiędzy słowami, głównie związkach synonimicznych. Zwykle każde wystąpienie danego słowa w innej kategorii tezaury reprezentuje inne znaczenie tego słowa. Słowa w tej samej kategorii są ze sobą semantycznie powiązane. Najczęściej eksploatowanym w celu rozpoznawania znaczeń tezaurem był Roget's International Thesaurus. Podobnie jednak jak w przypadku MRD, tezaury to źródła danych przystosowane dla człowieka a nie maszyny i z tego powodu nigdy nie były one idealnymi źródłami informacji o relacjach pomiędzy słowami. Nie nadają się one zatem jako podstawowe źródło wiedzy przy rozpoznawaniu znaczeń ale mogą stanowić wartościowy dodatek do innych, bardziej skutecznych metod.[12]

#### 4.2.2. Leksykony obliczeniowe

Innym źródłem danych były leksykony obliczeniowe (ang. computational lexicons), konstruowane ręcznie. Przykładami mogą być np. WordNet, CyC, ACQUILEX, COMLEX i inne. Istnieją dwa różne podejścia do konstruowania tego typu słowników: podejście wyliczeniowe (ang. enumerative approach), gdzie znaczenia są podane explicite, oraz podejście generatywne (ang. generative approach) w którym informacja semantyczna powiązana z danymi słowami jest niedospecyfikowana i pewne reguły generujące są używane aby wydobyć precyzyjną informację o znaczeniu.[12]

Spośród leksykonów wyliczeniowych najbardziej znanym obecnie jest WordNet, który jest też najczęściej używanym źródłem w celu rozpoznawania znaczeń w języku angielskim. Istnieją też wersje WordNetu przystosowane do kilku zachodnio i wschodnioeuropejskich języków lecz wszystkie one są jeszcze w fazie rozwoju.[12]

WordNet jest obszerną leksykalną bazą danych języka angielskiego, rozwijana pod kierownictwem George'a A. Millera. Rzeczowniki, czasowniki, przymiotniki i przysłówki zostały pogrupowane w zestawy synonimów kognitywnych (zwanym synsetami), z których każdy opisuje inne pojęcie. Synsety są połączone relacjami konceptualno-semantycznymi oraz leksykalnymi. Powstała sieć znaczeniowo pokrewnych słów i pojęć może być przeszukiwana za pomocą udostępnionej przeglądarki. Baza danych WordNet jest też publicznie i darmowo dostępna do pobrania i wykorzystania lokalnie. Jej struktura czyni ją nieocenionym narzędziem dla lingwistyki obliczeniowej oraz przetwarzania języka naturalnego. [11]

Jak podaje Wikipedia, baza danych WordNet zawiera około 150.000 słów zorganizowanych w ponad 115.000 synsetów, które tworzą łącznie 207.000 par typu słowo-znaczenie. Jej wielkość w formie skompresowanej to około 12 megabajtów. [5]

Należy zauważyć, że każdy synset zawiera grupę synonimów oraz kolokacji (złożonych z więcej niż jednego słowa). Ponadto jedno słowo może występować w wielu synsetach jeśli ma wiele znaczeń. Dodatkowo, każdy synset posiada krótką definicję opisującą dane znaczenie, oraz bardzo często również przykładowe zdania. Typowym przykładem synsetu może być taki:

*good, right, ripe -- (most suitable or right for a particular purpose; "a*

*good time to plant tomatoes"; "the right time to act"; "the time is ripe for great sociological changes")*

Jak widać, synset ten zaczyna się od synonimów określających dane znaczenie, po których znajduje się definicja wraz z przykładami użycia. [5]

Siłą WordNetu jest bez wątpienia sieć powiązań jaką połączone są poszczególne synsety. Mnogość i znaczenie najważniejszych relacji doskonale opisuje Wikipedia[5]:

- Rzeczowniki
  - hiperonimy (hypernyms): Y jest hiperonimem X jeśli każdy X to pewien rodzaj Y (psowaty jest hiperonimem psa)
  - hiponimy (hyponyms): Y jest hiponimem X jeśli każdy Y to pewien rodzaj X (pies jest hiponimem psowatego)
  - terminy równorzędne (coordinate terms): Y jest terminem równorzędnym X jeśli X i Y dzielą ten sam hiperonim (wilk jest terminem równorzędnym psa i na odwrót)
  - holonimy (holonym): Y to holonim X jeśli X jest częścią Y (budynek jest holonimem okna)
  - meronim (meronym): Y jest meronimem X jeśli Y jest częścią X (okno jest meronimem budynku)
- Czasowniki
  - hiperonim (hypernym): czasownik Y jest hiperonimem czasownika X jeśli czynność X jest rodzajem czynności Y (postrzegać jest hiperonimem słuchać)
  - troponim (troponym): czasownik Y jest troponimem czasownika X jeśli czynność Y to robienie X w jakiś konkretny sposób (seplenić jest troponimem mówić)
  - zawieranie (entailment) : czasownik Y jest zawarty w X jeśli robiąc X trzeba zrobić Y (spanie jest zawarte w chrapaniu)
  - terminy równorzędne (coordinate terms): czasowniki, które dzielą wspólny

hiperonim (seplenić i krzyczeć)

- Przymiotniki
  - powiązane rzeczowniki
  - podobne do
  - imiesłów czasownika
- Przysłówki
  - przymiotniki źródłowe

Oprócz powyższych, WordNet definiuje też inne relacje takie jak na przykład antonimy (przeciwieństwa) lub bardziej szczegółowe zależności bazujące na powyższych. Wśród informacji dotyczących danego słowa znaleźć można też tak zwany licznik polisemiczny oznaczający liczbę synsetów zawierających dane słowo (czyli jak wiele znaczeń posiada dane słowo), a także wartość częstości wystąpień, która określa (na podstawie pewnej liczby przeanalizowanych tekstów wzorcowych) jak często dane słowo pojawia się w danym znaczeniu. Ta ostatnia właściwość może posłużyć jako mechanizm bardzo prymitywnej i dalece przybliżonej metody rozpoznawania znaczeń.[5]

Z uwagi na jego złożoność WordNet stanowi obecnie najbardziej obszerną informację leksykalną zawartą w pojedynczym źródle. Drugim powodem dla którego ta baza danych stała się tak popularna jest na pewno fakt, iż jest ona dostępna publicznie dla każdego i z uwagi na to, jak słusznie zauważają Ide i Veronis, będzie to z pewnością źródło, które będzie wywierać bardzo duży wpływ na badania związane z rozpoznawaniem znaczeń jeszcze przez wiele kolejnych lat.[12]

Jedne z pierwszych prób wykorzystania WordNetu do celów rozpoznawania znaczeń rozpoczęła Voorhees, która posłużyła się konstruktem zwanym *kaptur* (ang. hood) aby przedstawić kategorie znaczeń. Kapturem danego słowa jest największy pod-graf który zawiera to słowo. Dla każdego słowa z kolekcji dokumentów wzorcowych zliczana jest ilość wystąpień danego synsetu nad interesującym nas słowem w hierarchii rzeczowników co daje wielkość opisującą spodziewaną aktywność tego słowa. Analogiczne obliczenia przeprowadzane są dla słów z analizowanego dokumentu. Znaczenie powiązane z korzeniem kaptura dla którego różnica między obliczonym wynikiem z dokumentów

wzorcowych a tym z dokumentu analizowanego jest największa zostaje wybrane dla danego słowa. Technika ta nie jest jednak doskonała, głównie z powodu tego, że WordNet definiuje zbyt dużo różnych znaczeń o subtelnych różnicach, co niepotrzebnie komplikuje dane przy prawie zerowych korzyściach.[12]

Inne podejście przedstawia Sussna, który wylicza odległość semantyczną dla każdego z rzeczowników z tekstu wejściowego. Przypisuje on wagi, które bazują na rodzaju relacji (synonimy, homonimy, itp.) z powiązaniem WordNetu oraz definiuje metrykę na podstawie odniesień danego typu wychodzących z węzła oraz na głębokości węzła w drzewie, a na koniec wylicza odległość semantyczną wykorzystując znalezione wagi. Ta technika oparta jest na teorii, która mówi, że dla danego zestawu słów znajdujących się blisko siebie w tekście, wybranie znaczenia, które minimalizuje odległość między nimi jest wyborem znaczenia prawidłowego.[12]

Należy podkreślić, że mimo wielu swoich zalet, WordNet nie jest źródłem doskonałym. Problemem, który najczęściej raportują badacze jest zbyt duża szczegółowość tej bazy danych, przez co zawiera ona dużo znaczeń, z których wiele jest do siebie bardzo podobnych. Taka szczegółowość okazuje się zbyt duża i myląca dla większości stosowanych algorytmów.[12]

Drugą grupą leksykonów obliczeniowych są leksykony generatywne. W tym przypadku znaczenia powiązane ze słowami nie są podawane jako gotowa lista ale generowane na podstawie reguł które opisują regularności w tworzeniu znaczeń. Tego typu wyszukiwanie znaczeń rozpoczyna się od semantycznego oznaczenia słów, które to oznaczenia prowadzi do złożonej reprezentacji wiedzy odzwierciedlającej wszystkie powiązane znaczenia słowa. Następnie przetwarzanie semantyczne tworzy zależną od kontekstu interpretację zawierającą bardziej precyzyjne informacje związane ze znaczeniem danego słowa.

### **4.3. Metody bazujące na kolekcjach tekstów**

Zbiór tekstów stanowi bank próbek, które pozwalają tworzyć i rozwijać numeryczne modele językowe, a co za tym idzie, współdziałać z metodami empirycznymi rozpoznawania znaczeń. W połowie lat 60-tych zainteresowanie wykorzystaniem zbiorów

tekstów znacznie zmalało, głównie za sprawą ówczesnych teorii lingwistycznych, które kierowały badania na inne tory, i bardzo niewiele istotnych zbiorów zostało stworzonych. Dopiero w latach 80-tych użycie statystyki do badania języka wróciło do łask i zyskało na popularności.[12]

Z ważniejszych badaczy warto wymienić Weissa, który pokazał, że reguły rozpoznawania znaczeń mogą zostać utworzone na podstawie zbioru tekstów, w których znaczenia są ręcznie opisane. Teoria ta była testowana na dość ograniczonej ilości danych, gdyż było to pięć słów i 20 zdań treningowych dla każdego z nich, oraz po 30 zdań testowych dla każdego słowa w celu sprawdzenia wyników. Okazało się, że skuteczność tej metody wyniosła 90%. Doświadczenia na większą skalę przeprowadzili Kelly i Stone, którzy pracowali na zbiorze 1800 wieloznacznych słów przy wykorzystaniu zbioru tekstów o łącznej liczbie pół miliona słów. Stosowali oni też bardziej wyszukane reguły rozpoznawania, korzystając z kontekstu w jakim stoi słowo, informacji o kolokacjach, relacjach syntaktycznych oraz zawierania się we wspólnych kategoriach semantycznych. Wyniki tych doświadczeń pokazały 92% skuteczność.[12]

Kolejne przeprowadzane przez badaczy doświadczenia w tym zakresie bazowały na coraz to większych zestawach danych jednak przez cały czas na drodze stały dwie przeszkody. Pierwszą z nich była trudność w ręcznym oznaczaniu słów w danych testowych. Należało bowiem, słowo po słowie, decydować o jego znaczeniu, co było zajęciem żmudnym i czasochłonnym. Drugim problemem, który poniekąd wynikał z pierwszego była mała ilość już istniejących, gotowych danych testowych.[12]

Z uwagi na oczywiste problemy opisane powyżej zaczęto zastanawiać się nad systemami automatycznego oznaczania znaczeń słów. Pierwsze próby polegały na ręcznym oznaczeniu pewnej ilości wystąpień danego słowa, po czym następowała ekstrakcja informacji statystycznej, która mogła być dalej użyta do automatycznego oznaczenia pozostałych wystąpień danego słowa. Jeśli takie wystąpienie mogło być oznaczone z zadowalającą pewnością poprawności, informacja z nim związana była od razu dodawana do już istniejących informacji statystycznych.[12]

Odmienna metoda polegała na automatycznym analizowaniu słów oraz ich otoczenia i na tej podstawie grupowaniu ich w zbiory. Następnie należało ręcznie przypisać znaczenia ale nie pojedynczym słowom, a całym zbiorom, które grupowały słowa o tym samym znaczeniu. To jednak wciąż wymaga dużej dozy interwencji człowieka i ręcznego

oznaczania sporej ilości danych.[12]

Inne podejście do problemu zakładało użycie zbioru danych dwujęzycznych aby uniknąć ręcznego oznaczania znaczeń. Pomysł polegał na obserwacji, że różne znaczenia danego słowa zwykle tłumaczą się inaczej na obcy język. Na przykład angielskie słowo *pen* tłumaczy się jako *pióro* (przyrząd do pisania) lub *zagroda* (miejsce do trzymania zwierząt). Używając tej techniki tłumaczenie zdania może od razu określić sens danego słowa w zdaniu oryginalnym. Problemem może tu być fakt, że niektóre wieloznaczności występują w obu językach oraz to, że istnieje bardzo mało dwujęzycznych zbiorów tekstowych, a te które istnieją są zwykle wysoce wyspecjalizowane.[12]

## 5. Ontologie

Jak podaje Wikipedia, ontologia w sensie informatycznym to „formalna reprezentacja pewnej dziedziny wiedzy, na którą składa się zapis zbioru konceptów i relacji między nimi. Struktura ta, będąc opisem danej dziedziny wiedzy, może służyć jednocześnie jako podstawa do wnioskowania o właściwości opisywanych ontologią konceptów.” [5] W konkretnym, uproszczonym przypadku struktura taka może opisywać na przykład zwierzęta. Należy wtedy utworzyć obiekt *ssak* który jest obiektem nadrzędnym obiektów *pies* i *kot*. Oba obiekty podrzędne są tu połączone z nadrzędnym relacją IS A (z angielskiego *jest (czegoś)*). Ponadto każdy ze wspomnianych obiektów może posiadać szereg właściwości i cech opisanych relacjami IS (*jest (jakiś)*). Istnieć mogą też inne relacje opisujące części składowe obiektu, rodzaje, itp. Bardzo ważną cechą ontologii jest właściwość dziedziczenia cech i relacji. Oznacza to, że każdy obiekt podrzędny posiada automatycznie wszystkie cechy obiektu nadrzędnego. Dzięki temu, wiedząc że *mysz* jest *ssakiem*, można wnioskować, że tak jak wszystkie *ssaki*, jest stałocieplna.

System taki może być bardzo pomocny w przypadku aplikacji mającej na celu rozumienie wypowiedzi człowieka oraz tworzenie własnych wypowiedzi, imitujących ludzkie. Pozwala to aplikacji stosować zamienniki, parafrazować, wnioskować na temat cech obiektów itd. Należy wziąć jednak pod uwagę fakt, że różne obiekty opisane przez ontologię mogą mieć identyczną nazwę, co stanowi oczywisty problem. Przykładami takiej sytuacji może być angielskie słowo *horn* mogące oznaczać zwierzęcy róg, instrument muzyczny, a nawet samochodowy klakson. W przypadku języka angielskiego dużym problemem jest też wielość czasowników brzmiących tak samo jak powiązane z nimi rzeczowniki (*love, hate, hope*, itp.). Z uwagi na to, niezbędnym jest zastosowanie systemu rozpoznającego znaczenie słów, powiązanego z używaną ontologią.

Przykładem rozbudowanej ontologii, w której obiekty zostały zmapowane na znaczenia z bazy danych WordNet jest ontologia Suggested Upper Merged Ontology (SUMO). Jak podano na stronie [www.projects.cmu.edu](http://www.projects.cmu.edu), jest to największa formalna i publiczna ontologia istniejąca obecnie. Jest używana w badaniach naukowych oraz aplikacjach wyszukujących, lingwistycznych, itp. Jest to też jedyna ontologia w całości zmapowana na leksykon WordNetu. [15]

Inną bazą danych zawierającą wiedzę ontologiczną jest baza OpenCyc. Jak podają



autorzy, baza ta jest darmową i publicznie dostępną wersją technologii Cyc, która z kolei jest największą i najbardziej kompletną bazą wiedzy ogólnej oraz silnikiem rozumowania zdroworozsądkowego [18]. Według Wikipedii, najnowsza wersja tej ontologii (2.0) udostępniona w czerwcu 2009 roku zawiera setki tysięcy pojęć i miliony zależności, którymi powiązane są pojęcia. Baza składa się z 47000 koncepcji i 306000 faktów, które można dowolnie przeglądać za pomocą przeglądarki dostępnej online, jak również używać w innych projektach.

Należy również zauważyć, iż sam WordNet udostępnia powiązania ontologiczne pomiędzy znaczeniami. Niestety jest to ontologia dość uboga, w porównaniu do np. SUMO, jednak zawiera większość elementów jakie ontologia posiadać powinna. W przypadku tej pracy, trudności techniczne spowodowały, że należało ograniczyć się do wykorzystania możliwości jakie oferuje WordNet. Jak pokazała praktyka jest to wystarczające na potrzeby eksperymentów oraz niniejszej pracy.

## 6. Algorytmy i techniki przetestowane w ramach pracy

Niniejszemu opracowaniu teoretycznemu towarzyszy aplikacja implementująca niektóre z opisanych technik, które mogą wspomóc lub nawet stać się podstawą funkcjonowania chatterbota. Ten rozdział przedstawia i szczegółowo opisuje wykorzystane w programie algorytmy i rozwiązania oraz analizuje je pod względem skuteczności i przydatności.

### 6.1. Wektory cech

Jednym z prostszych modeli pamięci semantycznej jest stosowanie wektora cech opisujących dane pojęcie. Cechy określone są wartościami typu prawda/fałsz i mówią czy dany obiekt posiada konkretną cechę czy też nie. Dla przykładu wiedzę o zwierzętach można reprezentować przy pomocy wektora opisującego cechy takie jak *posiada rogi*, *posiada ogon*, *posiada kopyta*, itd. Przy odpowiednio dużej ilości tego typu cech można otrzymać całkiem dokładny obraz fizyczny zwierzęcia, którego dane opisane są takim wektorem. Oprócz cech fizycznych można w ten sposób opisać też zachowanie, funkcje życiowe czy też zwyczaje (*poluje nocą*, *oddycha tlenem*, *żyje w stadzie*). Zaletą tego typu opisu jest względna łatwość porównywania różnych obiektów i szukania podobieństw między nimi. Odległość pomiędzy takimi wektorami w przestrzeni n-wymiarowej (gdzie n to ilość opisywanych cech) określa stopień podobieństwa. Warto tu zauważyć, że rozdzielenie poszczególnych kategorii cech (fizyczne, zachowanie, itp.) na różne wektory umożliwia porównywanie obiektów na wiele sposobów i pozwala odpowiedzieć nie tylko na pytanie „*Do czego podobny jest koń?*” ale też „*Co wygląda jak koń?*” i „*Co zachowuje się jak koń?*”.

Istotną wadą takiego rozwiązania jest binarny charakter opisywanych cech. Trudno tutaj modelować cechy skalarne, takie jak na przykład wzrost lub szerokość. Jednym z rozwiązań może być zastosowanie kilku cech składowych do opisu jednej cechy. Przykładowo wzrost może być opisany cechami: niski, średni, wysoki. Jeśli taka dokładność nie wystarcza można zwiększyć ilość cech składowych. Należy też zwrócić uwagę na to, że istnieje drastyczna różnica w znaczeniu słowa duży gdy mówimy o 'dużym

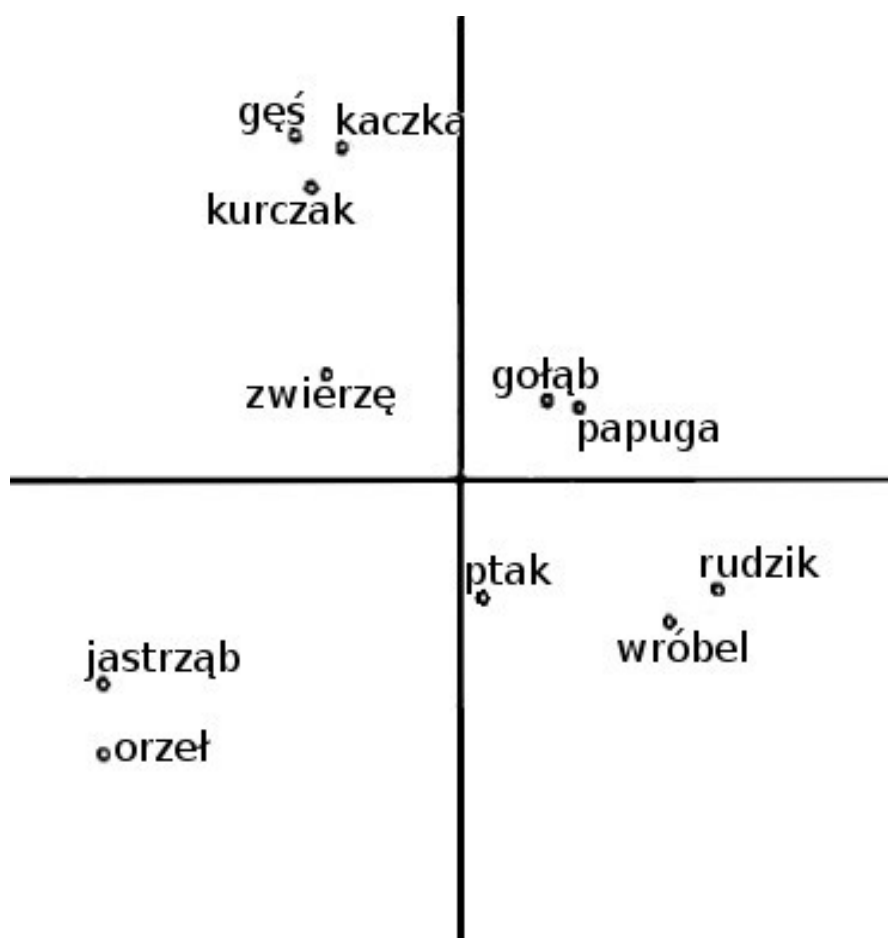
karaluchu' i o 'dużym słoniu'. W tym przypadku duży nie oddaje rzeczywistych, obiektywnych rozmiarów stworzenie, a jedynie mówi o względnych rozmiarach w porównaniu do innych osobników tego samego gatunku. W związku z tym konieczne mogłoby okazać się wprowadzenie pewnych dodatkowych danych opisujących na przykład średnią wielkość danego rodzaju obiektów. Wtedy cechy takie jak duży, czy mały spełniały by swoją funkcję opisując pewne odchylenie od średniej w obrębie danej kategorii.

Kolejną istotną kwestią jest podział cech na cechy definiujące oraz charakterystyczne. Cechy definiujące to takie, które obiekt musi posiadać aby być tym czym jest. Przykładowymi cechami ptaka są skrzydła oraz jajorodność. Cechy charakterystyczne z kolei, to cechy, które dany obiekt bardzo często posiada ale jeśli ich nie ma to wciąż może być tym czym jest. Cechą charakterystyczną ptaka jest umiejętność latania, choć zdarzają się gatunki jej pozbawione (kiwi, pingwin). Nie zmienia to jednak faktu, że latanie jest przez ludzi kojarzone przede wszystkim z ptakami i wiedza ta powinna być również modelowana. Poniższa tabela zaczerpnięta z artykułu Smith'a oraz jego współpracowników przedstawia podział istot latających jaki mógłby stworzyć człowiek, a który łatwo modelować przy pomocy opisywanych tutaj wektorów cech:

**Tab.1.** Podział istot latających [10]

	Podmiot - predykat	Cechy jakie dzieli podmiot i predykat
Zgodne z prawdą	rudzik – ptak wróbel – ptak papuga - ptak	Definiujące i charakterystyczne
Technicznie mówiąc	kurczak – ptak kaczka – ptak gęś - ptak	Definiujące ale nie charakterystyczne
Mówiąc nieprecyzyjnie	nietoperz – ptak motyl – ptak ćma - ptak	Charakterystyczne ale nie definiujące

Zaprezentowane tutaj zależności zwracają uwagę na następną istotną kwestię, a mianowicie kategoryzowanie obiektów. Powyższy model jest w stanie bardzo dobrze symulować kategoryzację ze względu na prototyp, która jest nieodłączną częścią nowoczesnej semantyki i gramatyki kognitywnej. Pozwala odpowiedzieć na pytanie „jak bardzo” dany obiekt należy do określonej kategorii lub też jak dobrym jest on przykładem tej kategorii [4]. Aby unaocznić znaczenie tego typu kategoryzacji poniżej przedstawiony jest rysunek zaczerpnięty z pracy Smitha (nieco zmodyfikowany), który przedstawia rzut odległości semantycznych poszczególnych obiektów na płaszczyznę dwuwymiarową:



**Rys. 1.** Odległości semantyczne – zwierzęta [10]

Rysunek przedstawia obiekty, z których każdy należy do kategorii *ptak*, oraz do kategorii *zwierzę*. Jak widać odległość semantyczna jest znacznie różna co wynika z posiadania lub nie posiadania przez obiekty cech charakterystycznych ptaka (oraz zwierzęcia). Najbliżej centrum kategorii *ptak* są stworzenia takie jak wróbel czy gołąb. Jastrząb i orzeł jako ptaki drapieżne, różniące się wyglądem i zachowaniem od pospolicie

spotykanych ptaków więc są nieco dalej. Podobnie kaczka, gęś i kurczak znajdują się w pewnej odległości od centrum głównie z racji braku umiejętności latania (należy tu założyć rozważanie kaczek i gęsi hodowlanych), co zdecydowanie odróżnia je od stereotypowego ptaka. Ten rodzaj podziału stworzeń (oraz wszelkich innych obiektów) jest bardzo typowy dla człowieka, a zatem wyposażenie programu komputerowego w podobną umiejętność z pewnością wpłynęło by bardzo korzystnie na stopień upodobnienia się symulacji wypowiedzi do rzeczywistych wypowiedzi człowieka.

Okazuje się jednak, że w przypadku zagadnienia tak obszernego jak rozpoznawanie tekstu oraz produkcja własnych tekstów, podejście tego typu może być podejściem błędnym. Stworzenie bazy danych opisującej zadowalającą ilość obiektów w sposób odpowiednio szczegółowy jest zadaniem nietrywialnym i na pewno czasochłonnym. Brak jest odpowiednio dużych tego typu baz danych dostępnych do wolnego wykorzystania. Należy więc przyjąć tą metodę jako niedostatecznie efektywną. Może ona co najwyżej służyć jako metoda dodatkowa lub baza do metod bardziej zaawansowanych.

## **6.2. Algorytm Xiaobin**

Autorami opisanego w tym podrozdziale algorytmu są Xiaobin Li, Stan Szpakowicz i Stan Matwin. W pracy, którą napisali oni na potrzeby międzynarodowej konferencji dotyczącej sztucznej inteligencji nie przedstawiają oni żadnej własnej nazwy swojego algorytmu. Tytułują go po prostu Algorytm do Rozpoznawania Znaczeń Słów Bazujący na WordNecie, ale z uwagi na wygodę w tej pracy postanowiono przyjąć nazwę roboczą Algorytm Xiaobin, od imienia jednego z twórców.

Jak już wspomniano algorytm ten wykorzystuje bazę danych WordNet i jej możliwości w celu rozpoznawania znaczeń. Metoda ta bazuje na kontekście w jakim słowo zostało użyte. Nie precyzuje ona wielkości kontekstu. W opracowaniu metody wykorzystanej w tej pracy kontekstem jest pojedyncze zdanie, choć nic nie stoi na przeszkodzie aby kontekst rozszerzyć. Wstępne testy takiego podejścia sprawiły jednak pewne problemy więc ostatecznie kontekst zawężono. Algorytm bazuje na pięciu regułach heurystycznych, na podstawie których znajdowane jest prawidłowe znaczenie. Cały opis algorytmu zawarty poniżej wraz z wykorzystywanymi regułami heurystycznymi,

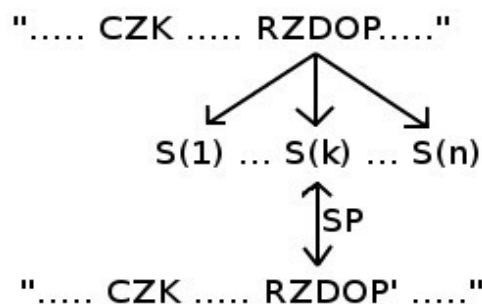
przykładami oraz rysunkami (zmodyfikowanymi na potrzeby tej pracy) został zaczerpnięty z opracowania Li, Szpakowicza i Matwina [14].

### 6.2.1. Reguły heurystyczne

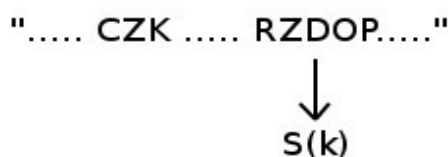
Zadaniem algorytmu jest znaleźć prawidłowe znaczenie dopełnienia (rzeczownika) w danym kontekście czasownikowym. Ze wszystkich możliwych znaczeń danego słowa w WordNecie należy wybrać takie, które najlepiej pasuje do danego słowa w danym kontekście. Poszukujemy zawsze znaczenia dopełnienia (rzeczownika) RZDOP w kontekście jego czasownika CZK. RZDOP ma  $n$  potencjalnych znaczeń w WordNecie.  $s(k)$  oznacza  $k$ -te znaczenie RZDOP w WordNecie, gdzie  $1 \leq k \leq n$ . SP jest semantycznym podobieństwem pomiędzy słowami w WordNetowej hierarchii JEST (Czymś). Strzałki zostaną użyte aby pokazać relację pomiędzy słowami lub pomiędzy słowem a jego znaczeniami.

Pierwsza reguła (HR1) mówi aby przeszukać tekst w poszukiwaniu pary czasownik-dopełnienie (CZK, RZDOP') w której RZDOP' jest semantycznie podobny do któregoś znaczenia słowa RZDOP w WordNecie i użyj tego znaczenia jako rozwiązania. [14]

**warunek**



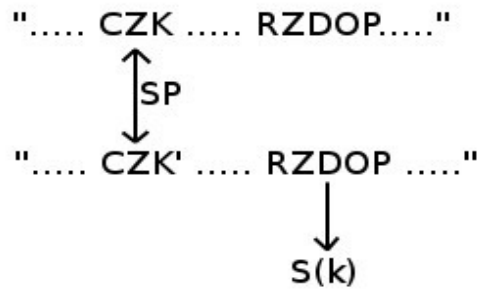
**rezultat**



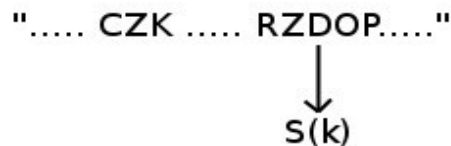
**Rys. 2.** Reguła heurystyczna nr 1

Druga reguła (HR2) nakazuje przeszukać tekst w poszukiwaniu pary (CZK', RZDOP), gdzie CZK' jest semantycznie podobne do CZK. Jeśli znaczenie słowa RZDOP w kontekście CZK' zostało już wcześniej znalezione należy uznać to znaczenie jako rozwiązanie. [14]

**warunek**



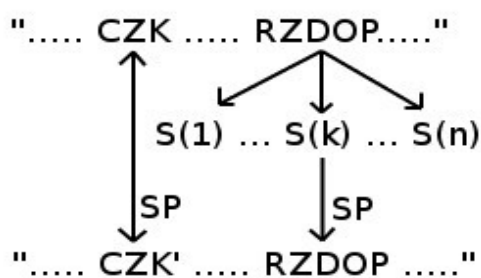
**rezultat**



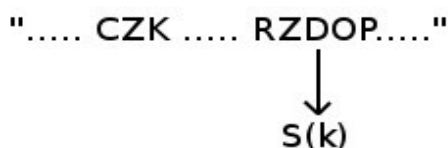
**Rys. 3.** Reguła heurystyczna nr 2

Trzecia reguła (HR3) mówi aby wyszukać w tekście pary (CZK', RZDOP'), gdzie CZK' jest semantycznie podobne do CZK, a RZDOP' jest semantycznie podobne do znaczenia słowa RZDOP w WordNecie. Znalezione znaczenie słowa RZDOP należy uznać jako rozwiązanie. [14]

warunek



rezultat



Rys. 4. Reguła heurystyczna nr 3

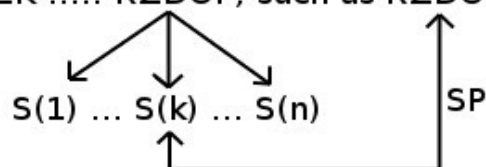
Jak łatwo zauważyć zasada HR2 nie zawsze jest w stanie znaleźć rozwiązanie gdyż bazuje na danych znalezionych wcześniej. Z kolei HR3 nakłada znacznie słabsze ograniczenia na podobieństwo semantyczne par czasownik-dopełnienie. W związku z tym zasadę HR1 należy uważać jako podstawową regułę algorytmu. Oprócz powyższych zasad bazujących na podobieństwie semantycznym pomiędzy parami czasownik-dopełnienie, wprowadzono też reguły związane ze strukturą syntaktyczną aby wspomóc rozpoznawanie znaczeń. [14]

Reguła czwarta (HR4) nakazuje wyszukać w tekście struktury „such as” (ang. taki jak), która znajduje się za parą (CZK, RZDOP). Dopełnienie RZDOP' stojące za „such as” z pewnością jest powiązane relacją synonimiczną lub hiponimiczną z jakimś znaczeniem słowa RZDOP w WordNecie. Znalezione tak znaczenie należy uznać jako rozwiązanie. Należy zauważyć, że reguła ta obejmuje swym działaniem jedynie bardzo szczególny przypadek w tekście, a więc jej wykorzystanie jest dość ograniczone. [14]



warunek

"..... CZK ..... RZDOP, such as RZDOP'....."



rezultat

"..... CZK ..... RZDOP, such as RZDOP'....."

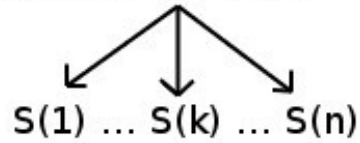


Rys. 5. Reguła heurystyczna nr 4

Reguła piąta (HR5) mówi aby znaleźć w tekście równorzędną strukturę „CZK and CZK” lub „CZK or CZK”, której dopełnieniem jest RZDOP. Jeśli znaczenie słowa RZDOP w kontekście CZK' zostało już znalezione to należy uznać to znaczenie jako rozwiązanie. Reguła ta podobnie jak HR2 bazuje na znalezionych wcześniej danych ale jej wykorzystanie jest znacznie ograniczone do konkretnej struktury (podobnie jak HR4), a zatem przydatność jej jest znacznie zmniejszona. [14]

warunek

"..... CZK ..... RZDOP ....."



"..... CZK' ..... RZDOP ....."

"..... CZK and/or CZK' ..... RZDOP ....."

rezultat

"..... CZK ..... RZDOP ....."



Rys. 6. Reguła heurystyczna nr 5

### 6.2.2. Przebieg algorytmu

Założmy, że rzeczownik  $W_n$  ma  $n$  znaczeń w WordNecie. Algorytm będzie próbował znaleźć prawidłowe znaczenie rzeczownika  $W_n$  w kontekście czasownika  $W_v$  lub też w parze czasownik-dopełnienie ( $W_v, W_n$ ). Znaczenie  $(W_n, k)$  oznacza  $k$ -te znaczenie słowa  $W_n$  w WordNecie gdzie  $0 \leq k \leq 1$ . Synonimia pomiędzy słowami oznacza, że oba słowa są w tym samym synsecie lub jedno słowo jest bezpośrednim węzłem nadrzędnym drugiego w WordNetowej hierarchii JEST (Czymś). Hiponimia oznacza że jedno słowo jest węzłem podrzędnym drugiego. Relacja równorzędna oznacza że jedno słowo jest węzłem równorzędnym drugiego słowa. Pewność wyniku jest zdefiniowana jako liczbę  $CF$ , gdzie  $0 \leq CF \leq 1$ . Wszelkie opisane tu oznaczenie pozostały takie same jak w oryginalnym dokumencie Li, Szpakowicza i Matwina dla przejrzystości i wygody referencji.

Algorytm podejmuje próbę odnalezienia właściwego znaczenia danego słowa na

osiem różnych sposobów, przechodząc przez osiem kolejnych kroków. Jeśli w danym kroku uda się odnaleźć znaczenie słowa pozostałe kroki algorytmu są pomijane.

W pierwszym kroku algorytm poszukuje  $W_n$  w WordNecie. Jeśli  $W_n$  ma tylko jedno znaczenie  $(W_n, 1)$  w WordNecie, znaczenie słowa  $W_n$  w danym kontekście czasownika  $W_v$  to znaczenie  $(W_n, 1)$ . Pewność w tej sytuacji wynosi 1.

W drugim kroku algorytm poszukuje pary czasownik-dopełnienie  $(W_v, W_n')$  w parsowanym tekście takiej, że  $W_n'$  jest synonimem lub hiponimem znaczenia  $(W_n, k)$ . Znaczenie słowa  $W_n$  w kontekście  $W_v$  to znaczenie  $(W_n, k)$ . Pewność tego wyniku to 0,9.

Krok trzeci jest analogiczny do kroku drugiego ale należy użyć relacji równorzędnej zamiast synonimii lub hiponimii pomiędzy  $W_n'$  a znaczeniem  $(W_n, k)$ . Pewność wynosi tutaj 0,8.

W kroku czwartym należy wyszukać w parsowanym tekście parę czasownik-dopełnienie  $(W_v', W_n)$ , gdzie  $W_v'$  jest połączone z  $W_v$  relacją synonimii, hiponimii lub równorzędną, oraz znaczenie  $W_n$  w jego kontekście  $W_v'$  jest już znane. W ten sytuacji znaczenie  $W_n$  w kontekście  $W_v$  jest również znaczeniem  $(W_n, k)$ . Pewność tego wyniku wynosi 0,7.

W kroku piątym trzeba w parsowanym tekście odnaleźć parę czasownik-dopełnienie  $(W_v', W_n')$ , gdzie  $W_v'$  jest powiązany z  $W_v$  relacją synonimii, hiponimii lub równorzędną oraz  $W_n'$  jest powiązany relacją synonimii lub hiponimii ze znaczeniem  $(W_n, k)$ . Znaczenie  $W_n$  w kontekście  $W_v$  to znaczenie  $(W_n, k)$ . Pewność tego wyniku jest równa 0,6.

Krok szósty jest analogiczny do piątego lecz należy użyć relacji równorzędną zamiast synonimii lub hiponimii pomiędzy  $W_n'$  oraz znaczeniem  $(W_n, k)$ . Pewność tego wyniku to 0,5.

W kroku siódmym należy w parsowanym tekście znaleźć strukturę „such as” znajdująca się za parą czasownik-dopełnienie  $(W_v, W_n)$  oraz dopełnienie ten struktury  $W_n'$ , które jest synonimem lub hiponimem znaczenia  $(W_n, k)$ . Znaczenie  $W_n$  w kontekście  $W_v$  to w tej sytuacji znaczenie  $(W_n, k)$ . Pewność tego wyniku to 0,9.

W ósmym kroku należy w parsowanym tekście odnaleźć równorzędną strukturę czasownikową „ $W_v$  and  $W_v'$ ” albo „ $W_v$  or  $W_v'$ ”, której dopełnieniem jest rzeczownik  $W_n$  i znaczenie  $W_n$  w kontekście  $W_v'$  jest już znane. Znaczenie  $W_n$  w kontekście  $W_v$  jest takie

samo jak jego znaczenie w kontekście  $Wv'$ . Pewność tego wyniku wynosi 0,9.

Należy zauważyć, że krok ósmy powyższego algorytmu zakłada rozbicie struktury *czasownik and czasownik rzeczownik* na dwie struktury *czasownik rzeczownik*. W przypadku niniejszej pracy okazało się to niepotrzebne i dlatego krok ten jest całkowicie pomijany w wersji algorytmu zaimplementowanej w aplikacji towarzyszącej pracy.

### 6.3. Właściwości WordNetu wykorzystane w pracy

Spośród wielu możliwości jakie oferuje WordNet tej pracy wykorzystano w szczególności takie, które pozwalają imitować wypowiedzi ludzkie. Niektóre z nich odpowiadają właściwością jakimi charakteryzują się ontologie, o czym wspomniano w rozdziale 5, inne natomiast opierały się na cechach właściwych tylko bazie WordNet. Należy też zaznaczyć, że wszelkie działania aplikacji będącej częścią tejże pracy dzielą się na dwie zasadnicze części: próba rozpoznania znaczenia słów w wypowiedzi użytkownika oraz działania tworzenia odpowiedzi biorące pod uwagę rozpoznane słowa.

Przede wszystkim należy wymienić możliwość wyszukiwania synonimów. Synonimy, jako wyrazy o tym samym znaczeniu mogą być używane naprzemiennie. Pozwala to na uniknięcie powtórzeń oraz na zaimplementowanie mechanizmu parafrazowania tekstu. Parafraza opiera się tu na zamianie rozpoznanych słów na ich losowe synonimy.

Następną właściwością, jest możliwość wyszukiwania meronimów, czyli, jak opisano w rozdziale 4, części składowych danego obiektu. Jeśli chodzi o ta właściwość, to WordNet jest stosunkowo ubogi. Wiele pojęć, po których można by się spodziewać wielu części składowych posiada ich bardzo mało, często jedną, lub żadnej. Jednak już jedna taka część pozwala na odniesienie się do tegoż szczegółu danego obiektu i kontynuowanie rozmowy.

W pracy wykorzystano też hiponimy, czyli pojęcia podrzędne do pojęcia badanego. Pozwala to na podawanie bardziej szczegółowych przykładów związanych z danym pojęciem. Jak pokazały próby jest to właściwość, która bardzo przydaje się do imitowania ludzkich wypowiedzi.

Równie przydatną właściwością WordNetu zastosowaną w pracy są definicje, które

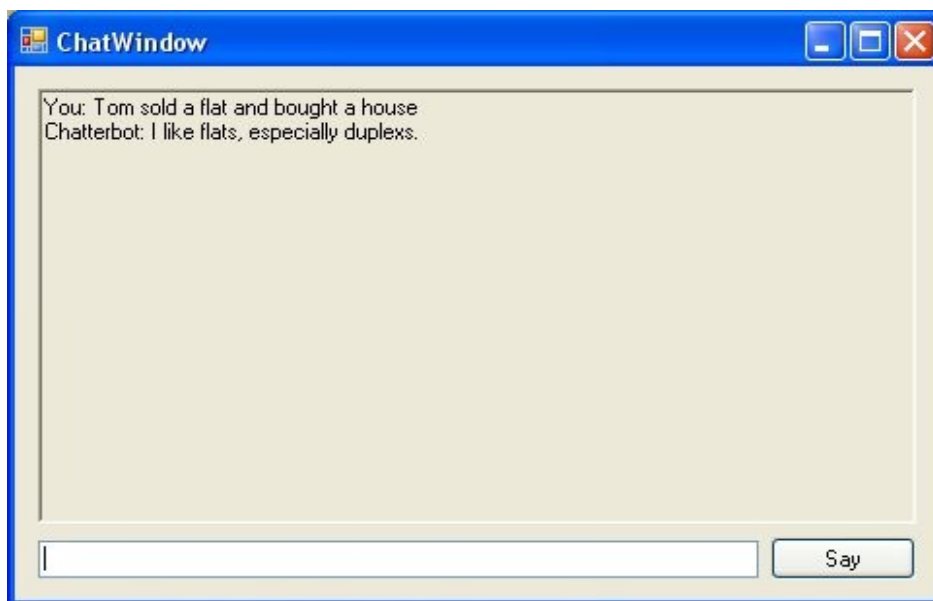
występują w przypadku każdego znaczenia. Przede wszystkim, mogą one służyć jako zbiór wiedzy na temat obiektów o których mowa w konwersacji, ale mogą być przydatne również w przypadku precyzowania znaczenia poszczególnych słów. Przypuśćmy, że algorytm rozpoznał znaczenie danego słowa, ale z bardzo małym współczynnikiem pewności. Można zatem zapytać użytkownika wprost czy ma na myśli to znaczenie, które związane jest z daną definicją. Zabieg taki może być również dobrą imitacją zachowań człowieka, ponieważ nawet w naturalnej konwersacji dwojga ludzi, czasami występują problemy z dokładnym zrozumieniem. Ponadto, definicje zawierają często przykładowe zdania zawierające badane słowo. Przykłady te również zostały użyte w aplikacji, choć ich przydatność jest raczej znikoma.

## 7. Opis aplikacji towarzyszącej pracy

Integralną częścią niniejszej pracy jest aplikacja napisana w technologii .NET w języku C#. Aplikacja jest napisana tak, aby zapewnić jej jak najlepszą przenośność. Główny silnik programu może być wykorzystany zarówno w aplikacji desktopowej jak i aplikacji internetowej osadzonej na stronie www, co zostało osiągnięte głównie dzięki uniwersalności środowiska .NET. W przypadku niniejszej pracy, program dołączony do niej ma postać aplikacji przeznaczonej dla środowiska MS Windows.

Podczas powstawania tej pracy, aplikacja ulegała ciągłym zmianom i modyfikacjom, tak aby na bieżąco testować wybrane algorytmy i podejścia do zagadnienia chatterbota. Z uwagi na to, pewne decyzje projektowe, które miały zastosowanie w początkowych etapach pisania pracy nie znajdują zastosowania w końcowym produkcie, jednak pozostały w nim z powodu trudności jakich przysporzyłaby potrzeba powtórnego napisania aplikacji od początku. Niektóre funkcje zostały zablokowane, tak aby końcowa aplikacja była jak najbardziej przejrzysta. Można się jednak na nie natknąć w kodzie źródłowym, który również stanowi część niniejszej pracy.

Aby móc zastosować silnik aplikacji w dowolnym programie (desktopowym lub sieciowym) wystarczy zapewnić dwie funkcjonalności: możliwość tekstowego wpisywania poleceń lub wypowiedzi przez użytkownika i przekazywania ich do silnika oraz możliwości wyświetlania odpowiedzi. Program demonstracyjny charakteryzuje się prostotą i przejrzystością. Jest podobny do uproszczonego komunikatora internetowego, co widać na poniższym obrazku.



Rys. 7. Okno aplikacji testowej

Jak można zaobserwować, aplikacja składa się z dużej przestrzeni przeznaczonej do wyświetlania historii rozmowy. W przypadku przepełnienia się ekranu pojawia się pasek przewijania umożliwiający przegląd całej historii. Poniżej znajduje się część służąca do wpisywania wypowiedzi użytkownika, oraz przycisk zatwierdzający wpisaną wypowiedź. Każde wpisanie tekstu przez użytkownika i zatwierdzenie go powoduje wpisanie tegoż tekstu do historii rozmowy oraz wygenerowanie odpowiedzi chatterbota. Trzeba tutaj zaznaczyć, że program nie jest przystosowany do radzenia sobie ze znakami interpunkcyjnymi lub łączeniem słów myślnikiem. Tekst rozdzielany jest na słowa, które z założenia oddzielane są znakami spacji. Taka konwencja jest wynikiem badawczej i eksperymentalnej natury tej aplikacji.

Odpowiedź programu może zostać wygenerowana na trzy różne sposoby. Pierwsze dwa stanowią odpowiedzi na polecenia *disambiguate* (rozpoznań) oraz *paraphrase* (parafrazuj). W przypadku rozpoczęcia tekstu użytkownika od tych słów, program stara się, w przypadku pierwszego rozkazu, rozpoznać znaczenie słów stojących za poleceniem i podać ich definicje oraz ,w przypadku drugiego rozkazu, rozpoznać znaczenie słów, a potem dokonać parafrazy tekstu. Trzeci rodzaj odpowiedzi jest używany we wszystkich przypadkach, które nie zaliczają się do dwóch opisanych powyżej. Jest to próba udzielenia odpowiedzi symulującej ludzką wypowiedź i nawiązującej w jakiś sposób do tekstu wprowadzonego przez użytkownika.

Należy zaznaczyć, że rozpoznawanie znaczenia słów wykonano z zastosowaniem opisanego w poprzednim rozdziale algorytmu Xiaobin, który nie jest pozbawiony wad. Przede wszystkim przystosowany on jest do rozpoznawania tylko i wyłącznie rzeczowników. W niniejszej pracy zastosowano prosty algorytm określający część mowy wyrazu co zapobiega wyszukiwaniu znaczeń słów, których nie powinien, takich jak czasowniki, rodzajniki, przyimki, itp. Ponadto, program stosuje listę słów, zapisaną w pliku xml, które są pomijane w procesie rozpoznawania znaczenia.

Proces tworzenia odpowiedzi nawiązującej do tekstu użytkownika dzieli się na kilka etapów. Na początku następuje rozpoznanie znaczeń słów. Następnie program próbuje stworzyć po kilka zdań, dla każdego z rozpoznanych słów, które są z nimi powiązane. Etapem końcowym jest losowe wybranie jednej z wygenerowanych odpowiedzi i zwrócenie jej jako wynik działania algorytmu.

Tworzenie wszystkich rodzajów odpowiedzi oparte jest na przygotowanych w pliku xml szablonach odpowiedzi. Początkowo program wyszukuje relacji między słowami lub słów powiązanych z danym pojęciem, po czym uzupełnia nimi wybrany losowo szablon z odpowiedniej kategorii szablonów. Najpierw program próbuje wyszukać meronim dla każdego z rozpoznanych słów, następnie hiponim, a na koniec skorzystać z definicji znaczenia. Trzeba zwrócić uwagę, że gdy program szuka meronimów bądź hiponimów, wielokrotnie znajduje on bardzo wiele tychże, lecz wybiera tylko po jednym z każdego. Przed wstawieniem znalezionych słów do szablonu, czasami (w zależności od szablonu) następuje jeszcze ich zamiana na liczbę mnogą, bądź dodanie odpowiedniego rodzajnika nieokreślonego. Jeśli algorytm nie rozpozna żadnego słowa, lub nie uda mu się znaleźć żadnych powiązanych znaczeń aby wygenerować odpowiedzi, następuje wylosowanie jednego z tekstów ogólnych aby uniknąć braku jakiegokolwiek odpowiedzi.

### **7.1. Kod osób trzecich wykorzystany w pracy**

Podczas tworzenia aplikacji testowej wykorzystano kilka algorytmów i rozwiązań udostępnionych publicznie do wykorzystania. Pozwoliło to pominąć etap tworzenia pewnych niezbędnych narzędzi i skupienie się na istocie zagadnienia. Przede wszystkim wykorzystano gotową bibliotekę służącą to komunikacji z bazą danych WordNet. Jest to



aplikacja udostępniona przez Malcolma Crowe'a i Troya Simpsona, którzy są związani z projektem WordNet.Net [20]. Biblioteka ta pozwala na odpytywanie bazy WordNet, podobnie jak można to czynić za pośrednictwem strony www tejże bazy. Daje też szereg dodatkowych możliwości, takich jak na przykład pytanie o konkretne części mowy. Ponadto zapewnia odpowiednie klasy, które reprezentują wszystkie najważniejsze obiekty WordNetu, takie jak synset czy znaczenie. Umożliwia to łatwą i intuicyjną integrację z dowolną aplikacją.

Innym gotowym rozwiązaniem jakie okazało się pomocne jest algorytm oznaczania części mowy w zdaniu napisany przez Erica Brilla z Massachusetts Institute of Technology (M.I.T.) dostępny na stronie Marka Watsona [21]. Jest to prosty mechanizm analizujący głównie przyrostki słów oraz wzajemną relację i kolejność słów. Na potrzeby aplikacji testowej tego projektu koniecznym była drobna modyfikacja algorytmu Erica Brilla, która jednak ograniczała się do postaci danych, na których pracuje algorytm, bez ingerencji w jego przebieg i logikę.

## 7.2. Analiza działania aplikacji

Na początku analizy zostanie przedstawionych kilka fragmentów dialogów jakie powstały podczas testów aplikacji. Są to przykłady zarówno dobre (w sensie trafności odpowiedzi, logiczności oraz podobieństwa do wypowiedzi naturalnych człowieka) jak i te gorsze. Pod każdym przykładem zostanie umieszczony skrótowy opis.

*You: my uncle sold a cat and bought a dog last week*

*Chatterbot: I like dogs, especially hunting dogs.*

*You: interesting are you a hunter or you simply like hunting dogs*

*Chatterbot: I like hunters, especially trappers*

*You: tell me what is a trapper*

*Chatterbot: Did you know that a trapper is someone who sets traps for animals (usually to obtain their furs) ?*

Ten przykład pokazuje jak duży potencjał drzemie w WordNecie oraz jego właściwościach. Przechodzenie podczas rozmowy od psa do psa myśliwskiego lub od myśliwego do trapera sprawia wrażenie dużej naturalności.

*You: my mother took money from the bank and bought a house last month*

*Chatterbot: Talking about houses reminds me of a poem that starts with the words: he has a house on Cape Cod*

*You: who wrote this poem was it a famous writer*

*Chatterbot: I like writers, especially Bucks.*

W tym fragmencie widać zastosowanie przykładowego zdania z WordNetowej definicji słowa – jest ono wstawione w pierwszej wypowiedzi chatterbota po dwukropku. Jest to niestety jeden z niewielu przypadków kiedy zastosowanie tego (potencjalnie wyrwanego z kontekstu) zdania daje zadowalający wynik.

*You: paraphrase my mother borrowed money from the bank and bought a house*

*Chatterbot: my mother borrowed money from the banking company and bought a house*

*You: paraphrase do not eat bread and hamburgers unless you drink enough water*

*Chatterbot: do not eat staff of life and burger unless you drink enough H2O*

Powyższe przykłady parafrazy pokazują jak ważne jest rozpoznanie sensu słów. W pierwszym przypadku jedynie słowo bank zostało rozpoznane. W drugim przykładzie były to słowa chleb, hamburger oraz woda.

*You: disambiguate an old farmer had some ducks and cows on his farm*

*Chatterbot: Sentence 1:*

*an - unknown/not important*

*old - unknown/not important*

*farmer - unknown/not important*

*had - unknown/not important*

*some - unknown/not important*

*duck - unknown/not important*

*and - unknown/not important*

*cows - domesticated bovine animals as a group regardless of sex or age; "so many head of cattle"; "wait till the cows come home"; "seven thin and ill-favored kine"- Bible; "a team of oxen"*

*on - unknown/not important*

*his - unknown/not important*

*farm - workplace consisting of farm buildings and cultivated land as a unit; "it takes several people to work the farm"*

W tym fragmencie zaprezentowano działanie funkcji rozpoznawania znaczenia słów. Dwa istotne rzeczowniki zostały rozpoznane poprawnie. Niestety nie zostało rozpoznane słowo farmer oraz kaczka, co może być spowodowane zbyt ubogim kontekstem.

Wyniki działania aplikacji są bardzo interesujące. W wielu przypadkach produkuje ona odpowiedzi, które wydają się być bardzo naturalne i wręcz zadziwiają. Często produkowane są też odpowiedzi dość komiczne. Zdecydowanie dobrym zabiegiem jest wykorzystanie meronimów i hiponimów. Dowodzi to przydatności użycia struktur takich jak ontologie (choć tutaj były to własności WordNetu, a nie ontologia sensu stricte) w aplikacjach chatterbotowych. Ontologie zapewniają aplikacji pewien zasób wiedzy z której może korzystać i dzięki temu konstruować odpowiedzi, zamiast korzystać z uprzednio przygotowanych przez twórcę programu.

W przypadku znaczenia rozpoznanego prawidłowo, znakomita większość wyprodukowanych na jego podstawie zdań wygląda całkiem naturalnie i sensownie. Jeśli znaczenie zostanie rozpoznane nieprawidłowo, w najlepszym przypadku otrzymujemy odpowiedź komiczną, ale często jest ona raczej pozbawiona większego sensu biorąc pod uwagę kontekst. W początkowych etapach zdarzało się to dość często z uwagi na błędne klasyfikowanie słów jako rzeczowniki. Często pojawiały się wypowiedzi takie jak poniższe:

*You: what do you know about cars*

*Chatterbot: Talking about knows reminds me of a poem that starts with the words: he is always in the know*

*You: i was hoping that you could tell me something about cars or horns*

*Chatterbot: Did you know that a tell is a Swiss patriot who lived in the early 14th century and who was renowned for his skill as an archer?*

Widać tutaj ewidentnie problem z rozpoznawaniem które słowa są rzeczownikami a które innymi częściami mowy. W pierwszym przypadku jest to słowo *know*, natomiast w drugim słowo *tell*. Jest to o tyle rażące, że dla człowieka te słowa nie pozostawiają żadnej

wątpliwości co do ich funkcji w zdaniu. Wynikało to z tego, że algorytm tutaj zastosowany jest przeznaczony do rozpoznawania rzeczowników, co stanowi problem w sytuacji gdy trudno jest jednoznacznie zakwalifikować słowo jako rzeczownik (w języku angielskim wiele słów brzmiących tak samo może funkcjonować zarówno jako rzeczownik jak też czasownik). Problem udało się jednak w dużej mierze przezwyciężyć poprzez zastosowanie prostego algorytmu oznaczającego części mowy w zdaniu. To pozwoliło wyeliminować pewne słowa z procedury rozpoznającej jeszcze przed jej rozpoczęciem. Wpłynęło to pozytywnie zarówno na skuteczność algorytmu rozpoznającego znaczenie jak i na szybkość jego działania.

Wykorzystanie przykładowych zdań, jakie występują w niektórych definicjach znaczeń w WordNecie należy traktować raczej jako ciekawostkę. Często są to bowiem zdania dość abstrakcyjne i wyrwane z kontekstu. Trudno zatem przygotować szablon, który mógłby z powodzeniem z nich korzystać. Jednym z zastosowanych w aplikacji tego projektu szablonów jest „Rozmowa o [rozpoznane słowo] przypomina mi wiersz zaczynający się od słów: [przykładowe zdanie z definicji].” Jak widać w tym przypadku zdanie może być właściwie dowolne, jednak taki szablon nie powinien być stosowany zbyt często gdyż wypowiedzi chatterbota brzmią wtedy sztucznie.

### **7.3. Propozycje dalszych badań**

Przede wszystkim, aby móc skutecznie tworzyć wypowiedzi lub wnioskować bazując na wypowiedziach użytkownika niezbędne jest precyzyjne zrozumienie sensu zdań jakie dostarczane są aplikacji. Eksperymenty dowiodły, że zasadniczą trudnością było początkowo odróżnienie słów kluczowych dla zdania, czyli takich, których znaczenie jest najbardziej istotne dla zrozumienia sensu całego zdania. Rozwiązaniem okazał się algorytm potrafiący rozróżnić poszczególne części mowy jakimi są słowa w danym zdaniu, na podstawie ich położenia w tym zdaniu oraz względem siebie, jak też charakterystycznych końcówek. Innym aspektem, który wiąże się bezpośrednio z częściami mowy jest umiejętność rozbioru zdania na składowe jego części i utworzenie drzewa przedstawiającego zależności pomiędzy poszczególnymi słowami. Taka umiejętność dała by możliwość bardziej trafnego wyszukiwania znaczenia słów oraz pozwalałaby pomijać słowa nieistotne. Byłby to punkt wyjścia do stworzenia aplikacji w

pełni rozumiejącej wypowiedzi użytkownika.

Biorąc pod uwagę popularność języka AIML oraz bardzo dobre wyniki jakie osiąga on w przypadku prostych konwersacji w istniejących już chatterbotach, dobrym rozwiązaniem byłaby próba połączenia metod bardziej zaawansowanych, jak choćby te wykorzystane w niniejszej pracy, z prostymi metodami bodziec-reakcja takimi jak AIML. Można przypuszczać, że tego rodzaju hybryda dała by bardzo dobre rezultaty, ponieważ powieliłaby skuteczność AIML w przypadku wypowiedzi prostych i rozszerzyła jego możliwości, przystosowując program do radzenia sobie z wypowiedziami bardziej złożonymi i wymagającymi bardziej rozbudowanych algorytmów.

## 8. Podsumowanie

Jak należało przypuszczać, analiza która była przedmiotem tej pracy pokazała, że aplikacje chatterbotowe jakie dziś istnieją posiadają znaczną ilość wad, które zdecydowanie ograniczają ich funkcjonalność i wpływają negatywnie na stopień podobieństwa produkowanych przez nie wypowiedzi do wypowiedzi prawdziwego człowieka. Istotną wadą jest przede wszystkim podejście oparte wyłącznie na używaniu szablonów, które stosowane są zarówno do klasyfikowania wypowiedzi użytkownika jak i do tworzenia odpowiedzi chatterbota. Jest to prosta metoda bodziec-reakcja, która pozbawiona jest logicznego wnioskowania i produkowane za jej pośrednictwem odpowiedzi ograniczają się do takich, jakie zostaną przez programistę wpisane do skryptów z których korzysta aplikacja.

Program będący integralną częścią niniejszej pracy umożliwił przetestowanie metod, które nie są powszechnie stosowane w chatterbotach, ale potencjalnie oferują możliwości jakie mogą być przydatne w tego rodzaju aplikacjach. Przeprowadzone testy, choć w dość ograniczonym zakresie, pokazały, że teza mówiąca o tym, iż pewne istniejące dziś technologie bazujące na semantyce języka i językowych bazach danych da się z powodzeniem wykorzystać w chatterbotach okazała się prawdziwa. Przede wszystkim pokazane zostało jak dużym potencjałem dysponuje baza danych WordNet, która jest powszechnie dostępnym i wciąż rozrastającym się źródłem informacji semantycznej, czego dowodem są interesujące zdania budowane przez program testowy czerpiący z tejże bazy wiedzy. Równie ciekawym zagadnieniem są z całą pewnością ontologie, które też mogą służyć programom chatterbotowym w budowaniu wypowiedzi bardziej złożonych i bazujących na logicznym wnioskowaniu. Aplikacja testowa nie korzystała z żadnej rozbudowanej ontologii a jedynie wykorzystwała pewne ograniczone właściwości WordNetu, które są w nim dostępne. Już to pokazało, że potencjalne możliwości jakie niesie z sobą wykorzystanie ontologii są bardzo duże, co można było zaobserwować na przykładnie zdań powstałych z wykorzystaniem zależności meronimicznych i hiponimicznych. Można zatem założyć, że użycie pełnej, rozbudowanej ontologii da o wiele lepsze rezultaty i szereg nowych możliwości do wykorzystania w chatterbotach.

Sugestie zawarte na końcu niniejszej pracy, dotyczące potencjalnych dróg rozwoju aplikacji chatterbotowych, pokazują, że dziedzina ta może się z powodzeniem rozwijać,

gdyż istnieje cały szereg technologii mogących korzystnie wpłynąć na potencjał programów konwersacyjnych. Widać też, że stosowane obecnie metody, takie jak język AIML, nie są idealne, ale w połączeniu z metodami bardziej zaawansowanymi mogą stanowić podstawę bardzo dobrych chatterbotów. Wnioski wyciągnięte podczas testów sugerują, że dobry chatterbot, powinien być połączeniem metod prostych (bodziec-reakcja) z metodami zaawansowanego wnioskowania z wykorzystaniem baz wiedzy semantycznej.

## 9. Literatura

- [1] <http://www.artificial-solutions.com> – Strona Artificial Solutions
- [2] <http://www.zabaware.com/webhal/hal6alpha.asp> – Zabaware HAL
- [3] <http://www.personalityforge.com/index.php> – The Personality Forge
- [4] W. Skrzypczak, *Analog-based Modelling of Meaning Representations in English*, Nicolaus Copernicus University Press, Toruń 2006.
- [5] <http://www.wikipedia.org/> – Wikipedia
- [6] E. Agirre, P. Edmonds, *Word Sense Disambiguation: Algorithms and Applications*, Springer, 2006.
- [7] <http://alicebot.blogspot.com/> - Alicebot blog
- [8] <http://www.jabberwacky.com> – Jabberwacky
- [9] <http://www.loebner.net/Prizef/loebner-prize.html> – Nagroda Loebnera
- [10] E. E. Smith, E. J. Shoben, L. J. Rips, *Structure and Process in Semantic Memory: A Featural Model For Semantic Decisions*, *Psychological Review*, 81(3), 1974, 214-241
- [11] <http://wordnet.princeton.edu/> - WordNet
- [12] N. Ide, J. Veronis, *Word Sense Disambiguation: The state of the art*, *Computational Linguistics* 24(1),1998, 1-40
- [13] <http://www.scholarpedia.org> – Scholarpedia
- [14] Xiaobin Li, S. Szpakowicz, S. Matwin, *A WordNet-based Algorithm for Word Sense Disambiguation*, *International Joint Conferences on Artificial Intelligence*, 1995, 1368-1374
- [15] <http://www.ontologyportal.org/> - Suggested Upper Merged Ontology
- [16] <http://www.alicebot.org> – Alicebot
- [17] <http://www.pandorabots.com> – Pandorabots
- [18] <http://cyc.com/cyc/opencyc> – OpenCyc
- [19] <http://www.inguaris.pl> – Inguaris



[20] <http://www.ebswift.com> – projekt WordNet.Net

[21] <http://www.markwatson.com/> - strona www Marka Watsona