

**Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie**

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Automatyki



PRACA MAGISTERSKA

ŁUKASZ WRÓBEL

**AUTOMATYCZNA KLASYFIKACJA, GRUPOWANIE I
ZNAJDOWANIE UOGÓLNIŃ DLA JEZYKA POLSKIEGO.**

PROMOTOR:
dr Adrian Horzyk

Kraków 2011

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics
Institute of Automatics



MASTER OF SCIENCE THESIS

ŁUKASZ WRÓBEL

**AUTOMATIC CLASSIFICATION, CLUSTERING AND
SEARCHING FOR GENERALIZATIONS FOR POLISH
LANGUAGE.**

SUPERVISOR:
Adrian Horzyk Ph.D

Krakow 2011

Serdecznie dziękuję promotorowi, rodzinie i przyjaciołom za wsparcie podczas pisania tej pracy.

Spis treści

1. Wstęp	6
1.1. Cele pracy	6
1.2. Zawartość pracy	6
2. Wprowadzenie	8
2.1. Czym jest lingwistyka?.....	8
2.2. Lingwistyka komputerowa.....	8
2.3. Historia lingwistyki	9
2.4. Problemy i cele współczesnej lingwistyki komputerowej	10
2.5. Pająk internetowy.....	11
2.6. Ekstrakcja danych	14
3. Relacje między wyrazami w języku polskim	17
3.1. Części mowy	17
3.2. Relacje między wyrazami	18
4. Opis własnego rozwiązania	21
4.1. Założenia projektowe.....	22
4.2. Architektura systemu	22
4.2.1. Interfejs użytkownika.....	23
4.2.2. Pająk internetowy i ekstraktor danych	25
4.2.3. Biblioteka CLP.....	26
4.2.4. Specjalistyczny graf przyzwyczajień lingwistycznych	27
4.3. Opis działania aplikacji.....	29
4.3.1. Tryb online	29
4.3.2. Tryb offline.....	32
4.3.3. Symulator	32
4.4. Generacja specjalistycznego grafu LHG	33
5. Prezentacja działania aplikacji	36
6. Podsumowanie wyników pracy i wnioski	45
6.1. Podsumowanie	45
6.2. Dalsze możliwości rozwoju	46
6.3. Porównanie z rozwiązaniami o podobnej tematyce.....	47
7. Zakończenie	50

1. Wstęp

Komunikacja oraz przepływ informacji drogą elektroniczną są nieodzownym elementem współczesnego życia. W wielu sytuacjach niezwykle użyteczna jest automatyzacja tego procesu, nie tylko pod kątem wydajności i bezpieczeństwa, ale także poprawności lingwistycznej. O ile w przypadku gotowych szablonów wiadomości poprawność nie jest problemem, to gdy wymagana jest większa interaktywność, pojawiają się kłopoty.

Różne inteligentne czatboty posiadają wbudowane mechanizmy, które umożliwiają im wysoce interaktywną komunikację z użytkownikiem, która jednak często nie jest pozbawiona rozlicznych błędów zarówno składniowych, jak i logicznych. Niniejsza praca podejmuje tematykę automatycznego grupowania i klasyfikacji wyrazów w języku polskim w celu modelowania uogólnień i przechodniości właściwości obiektów w hierarchii słów.

Zakres pracy obejmuje zbudowanie specjalistycznego grafu lingwistycznego, agregującego słowa w zależności od stopnia uogólnienia pojęć, które reprezentują, na podstawie ich automatycznej klasyfikacji oraz grupowania. Do zbudowania tego grafu powinny być wykorzystane różne korpusy tekstów pozyskiwane z baz danych oraz Internetu. Celem pracy jest zbudowanie takiego grafu oraz pokazanie możliwości przenoszenia właściwości obiektów przez hierarchię klas tego grafu - modelując możliwości uogólniania, jakie zachodzą w ramach języka.

1.1. Cele pracy

Celem pracy jest zbudowanie specjalistycznego grafu przyzwyczajęń lingwistycznych (ang. *Language Habits Graph, LHG*)[9], który ukazuje relacje pomiędzy wyrazami. W szczególności skupia się on na uogólnianiu i na jego podstawie klasyfikowaniu słów oraz wyróżnianiu ich wspólnych właściwości. Zadaniem było stworzenie pająka internetowego, który przeszukując wyniki z wyszukiwarki internetowej buduje bazę wiedzy, wokół określonego przez użytkownika słowa. Na podstawie uzyskanych w ten sposób informacji, program tworzy sieć relacji dla podanego wyrazu. Sieć ta reprezentowana jest na grafie LHG w uproszczonej postaci, z uwzględnieniem siły (częstości występowania) określonych połączeń. Zamodelowana została w ten sposób zhierarchizowana struktura, która obrazuje grupowanie obiektów, jak i przechodność cech wspólnych wewnątrz danej klasy.

1.2. Zawartość pracy

1. W rozdziale 1 znajduje się wstęp zawierający krótką informację o tematyce i celu pracy.
2. Rozdział 2 opisuje zagadnienie lingwistyki komputerowej i ukazuje problematykę pracy w szerszej perspektywie.
3. Zagadnienia związanych z relacjami występującymi między wyrazami w języku polskim zostały opisane w rozdziale 3.
4. Rozdział 4 zawiera szczegółowy opis własnego rozwiązania.
5. Rozdział 5 przedstawia wyniki działania aplikacji na konkretnych przykładach.

6. W rozdziale 6 zamieszczone zostało podsumowanie wyników pracy oraz przemyślenia i wnioski.
7. Rozdział 7 jest zakończeniem, w którym zestawione są wyniki pracy z zamierzonymi celami.

2. Wprowadzenie

2.1. Czym jest lingwistyka?

Definicja 1 *Lingwistyka (językoznawstwo) jest to dział nauk humanistycznych badający istotę, budowę i rozwój języka[22].*

Język mówiony i pisany od wieków był przedmiotem badań i rozważań naukowych. Większość wysiłków koncentrowała się jednak na zgłębianiu jego ewolucji poprzez analizę dzieł literackich, czy też poszukiwaniu podobieństw pomiędzy różnymi językami i dialektami. Dopiero w XX wieku zaczęło się kształtować bardziej ustrukturalizowane i sformalizowane podejście do tego zagadnienia[22].

Zasadniczo lingwistyka dzieli się na teoretyczną i stosowaną. Językoznawstwo teoretyczne skupia się na badaniu aspektów współczesnej odmiany konkretnego języka i wyszukiwaniu oraz opisywaniu jego ogólnych zasad. Przedmiotem rozważań mogą być też ogólne reguły i aspekty wspólne dla wszystkich języków. Stosowana lingwistyka podejmuje problemy związane z wykorzystaniem wyników tych rozważań w innych dziedzinach. Bada język w szerszej perspektywie zarówno w porównaniu z innymi, jak i jego własnym kształtowaniem się przez lata[22].

Współczesne językoznawstwo korzystające z najnowszej technologii dało początek informatyzacji tej nauki. Otworzyła ona nie tylko nowe możliwości, ale i liczne wyzwania związane z elektroniczną komunikacją między ludźmi, a przede wszystkim człowieka z komputerem. Tak narodziła się lingwistyka komputerowa.

2.2. Lingwistyka komputerowa

Definicja 2 *Lingwistyka komputerowa to dział lingwistyki używający modeli komputerowych w celu testowania hipotez dotyczących mowy i języka oraz tworzenia programów komputerowych przetwarzających język naturalny[23].*

Ta ogólna definicja ukazuje jak bardzo lingwistyka komputerowa zakorzeniona jest w rozmaitych dziedzinach językoznawstwa. Zakres jej zastosowań jest jednak o wiele większy, gdyż obecna jest także między innymi w filozofii, psychologii, informatyce (np. sztuczna inteligencja) czy matematyce (np. rachunek prawdopodobieństwa, teoria automatów, logika matematyczna, statystyka, czy języki formalne)[23].

Dziedzinę tą można podzielić na część teoretyczną i praktyczną. Zagadnienia teoretyczne skupiają się na badaniu możliwości automatyzacji procesów związanych z klasyfikacją podzbiorów językowych, obliczaniu wydajności tych procesów, a także tworzeniu ich formalnych opisów. Praktyczne podejście do lingwistyki komputerowej owocuje natomiast powstawaniem algorytmów i złożonych modeli automatyzujących i przetwarzających, które znajdują zastosowanie w rozmaitych aspektach językowych[23].

Rozwój i cele stojące przed lingwistyką komputerową są ściśle związane z rozwojem technologii, jak i z rosnącymi potrzebami automatyzacji i formalizacji procesów językowych, które wykraczają poza możliwości człowieka. Coraz nowocześniejsze maszyny z dużą mocą obliczeniową są w stanie realizować nawet bardzo złożone i jeszcze niedawno zbyt kosztowne algorytmy, co z kolei sprawia, że zagadnienia związane ze sztuczną inteligencją są obecnie bardzo popularne.

2.3. Historia lingwistyki

Aby dobrze zrozumieć jak wielki postęp dokonał się w dziedzinie lingwistyki komputerowej warto zagłębić się w historię jej rozwoju. Daje to możliwość zaobserwowania trendów i zagadnień, które były, są i będą przedmiotem badań współczesnego językoznawstwa informatycznego.

Początki lingwistyki jako nauki o języku sięgają czasów starożytnych. Za jej kolebkę uważa się Indie, gdzie już w pierwszym tysiącleciu p.n.e. dokonywano tłumaczeń starożytnych tekstów religijnych. Za twórcę pierwszej gramatyki uważa się Paniniego, który około V-IV w. p.n.e. stworzył szereg pojęć i definicji oraz około 4 tysiące reguł językowych. Niektóre powstałe wtedy terminy takie jak rozróżnienie między głoskami, a literami, fleksja, słowotwórstwo, struktura wyrazu, czy klasyfikacja części mowy używane są do dnia dzisiejszego. Wszelkie wyjątki, które licznie pojawiają się w językach, takich jak łacina, czy greka Panini w swojej gramatyce opisał za pomocą bardzo szczegółowych reguł, które opisywały nawet pojedyncze przypadki[14].

W starożytnej Grecji i Rzymie skupiano się bardziej na ogólnych problemach języka, takich jak jego istota, stosunek do logiki, retoryki i poetyki. Ukształtowały się dwa odmienne punkty widzenia. Analogizm, który zakładał, że język ma charakter logiczny oraz anomalizm, który uznawał że język jest zmienny. Z tego okresu wywodzą się pojęcia etymologii (nauka o pochodzeniu i ewolucji znaczenia oraz formy wyrazów), teorii nominacji (nazywania świata za pomocą słów), onomatopei (dźwiękonaśladownictwa). Szczególnie rozwinęło się także językoznawstwo opisowe. Powstał podział na części mowy, opisano deklinacje, koniugacje i zasady opisu składni[14].

Dopiero w XVIII w. gramatyka języka łacińskiego przestała być powszechnie uważana za uniwersalną, do czego przyczyniły się badania Abu Hayan at Tauhaidi'ego, który postulował, że na proces kształtowania języka mają wpływ kultura i historia narodu. Zapoczątkowało to pojawianie się filologii narodowych: m.in. francuskiej, niemieckiej, angielskiej i słowiańskiej[14].

XIX w. to rozwój językoznawstwa historycznego, który rozpoczęło odkrycie przez William'a Jones'a sanskrytu (starożytnego języka Indii). Jego znaczne podobieństwo do języków europejskich zainicjowało badania nad komparatywistyką (lingwistyką porównawczą). Określone zostały regiony i narody, których formy komunikacji mają wspólne korzenie historyczne, tworząc tak zwane rodziny językowe[20].

Wiek XX to okres strukturalizmu i formalizacji zasad językowych. Zamiast do genezy zaczęto przywiązywać większą wagę do opisu struktury. F. de Saussure uważał, że każdy znak zbudowany jest z dwóch elementów: znaczącego (jego reprezentacja np. obraz, czy słowo) i znaczonego (pojęcie z nim związane). Język zdefiniowano więc jako dwuklasowy system semiotyczny, który służy do komunikacji[19]. Powstał także pierwszy matematyczny opis języka, który za pomocą formalnych narzędzi matematycznych takich jak dowody, twierdzenia i rachunki stworzył drzewiastą strukturę wszystkich wyrażen poprawnych w danej gramatyce. N. Chomsky wprowadził gramatykę transformacyjno-generatywną, która jest systemem określającym skończoną ilość reguł. Z ich pomocą rozmówca jest w stanie utworzyć poprawne zdanie, którego nigdy wcześniej nie słyszał, a odbiorca stwierdzić, czy zdanie, które usłyszał jest poprawne w danym języku, czy nie[13].

Formalizacja zasad lingwistycznych stworzyła podwaliny pod powstanie językoznawstwa informatycznego, które wykorzystując ściśle określone reguły umożliwiło automatyczną analizę i przetwarzanie tekstu, wyszukiwanie, a także budowanie poprawnych form językowych przez programy komputerowe.

Początkowo w lingwistyce komputerowej dominowały dwa podejścia. Pierwsze oparte było na metodach symbolicznych (teoria generatywna, sztuczna inteligencja), natomiast drugie bazowało na metodach statycznych takich jak analiza tekstu, rozpoznawanie liter itp[6]. Powstały modele statystyczne służące do syntezy mowy, języki programowania oparte o logikę formalną i metody wnioskowania (np. Prolog), a także aplikacje sterowane komendami w języku naturalnym.

Lata 90 to dynamiczny rozwój Internetu oraz mocy obliczeniowej i pojemności pamięciowej komputerów. Pojawiają się złożone aplikacje wykorzystujące zaawansowane algorytmy oparte m.in. o statystykę. W odpowiedzi na zapotrzebowanie na programy zajmujące się ekstrakcją danych z Internetu, czy korpusów tekstów oraz narzędzi przetwarzających i analizujących tekst z wykorzystaniem metod sztucznej inteligencji powstało wiele rozwiązań komercyjnych[6].

2.4. Problemy i cele współczesnej lingwistyki komputerowej

Największym problemem, przed jakim staje współczesne językoznawstwo komputerowe jest *wieloznaczność*[6]. Dotyczy on fonetyki (homofonia - fonetyczna tożsamość różnych znaków językowych), morfologii (homonimia - posiadanie różnych znaczeń przez identyczne formy językowe), składni (niejednoznaczność przy określaniu części zdania), semantyki (idiomy, polisemia - posiadanie przez jedno słowo wielu znaczeń), pragmatyki (metafory, ironia). O ile człowiek korzystając z wyższego poziomu języka może w łatwy sposób rozwiązać te problemy, to już komputer, ma z tym spore kłopoty.

Według Bonnie Webber (2001) dwa podstawowe cele jakie stoją przed lingwistyką informatyczną to[11]:

- modelowanie ludzkiego rozumienia i generacji języka naturalnego jako systemu procesów przetwarzających informację (lingwistyka informatyczna)
- wyposażenie komputerów w mechanizmy analizy i generowania języka naturalnego w celu dostarczenia użytecznej usługi (stosowane przetwarzanie języka naturalnego - ang. *applied natural language processing*, inżynieria języka naturalnego - ang. *natural language engineering*, technologia językowa - ang. *language technology*)

Roland Hausser (2001) uważa, że lingwistyka komputerowa powinna się skupiać na modelowaniu komunikacji człowieka z komputerem, konstruując modele wyjaśniające naturalny przekaz informacji w sposób spójny funkcjonalnie, precyzyjny matematycznie i efektywny obliczeniowo[11].

Z punktu widzenia współczesnych wymagań najintensywniejsze prace prowadzone są nad[11]:

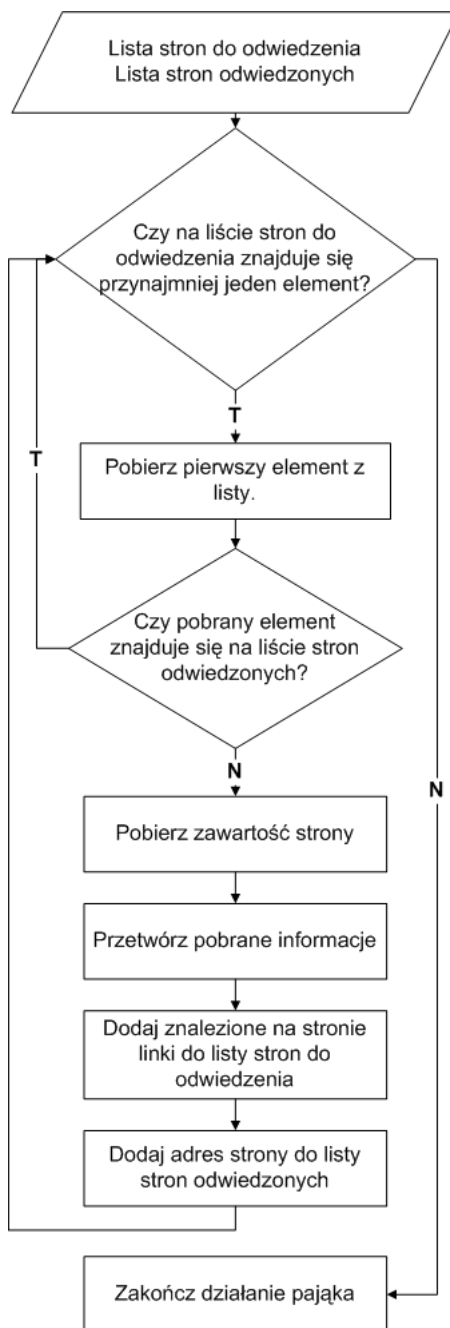
- aplikacją umożliwiającą wyszukiwanie specyficznych informacji na podstawie dokładnie opisanych wymagań co do rezultatów
- systemem potrafiącym uczyć się ze źródeł przyswajalnych przez człowieka, a następnie potrafiącym rozwiązać test sprawdzający zdobyta wiedzę, jaki rozwiązałby człowiek
- systemem do automatycznego tłumaczenia na inne języki naturalne, a także języki nieistniejące, ale opisane określonym zestawem reguł
- aplikacją umożliwiającą komunikację głosową człowieka z komputerem

Niewyczerpanym źródłem wiedzy dla programów wykorzystujących sztuczną inteligencję do wyszukiwania informacji jest Internet. Ogromu zawartych w nim informacji człowiek nie jest w stanie przetworzyć, natomiast posiadająca odpowiedni zestaw reguł oraz sposób wartościowania i ekstrakcji danych maszyna może podjąć się tego zadania. O ile już samo stworzenie narzędzia do inteligentnego pozyskiwania informacji nie jest zadaniem prostym, to dodatkowe trudności pojawiają się podczas przetwarzania danych. System, aby wykorzystać pozyskaną wiedzę w określonym celu, musi prawidłowo rozumieć zarówno posiadane dane, jak i cel ich wykorzystania. W szczególności stanowi to ogromny problem, gdy zadaniem jest przetłumaczenie tekstu na inny język. Znajomość reguł tworzenia poprawnych form nie gwarantuje wierności przekazu. Przykładem może być napisana przed wieloma laty rosyjska aplikacja, która tłumaczyła wyrażenia z języka angielskiego na rosyjski i później znów na angielski. Zamieniła ona wyrażenie „out of sight, out of mind” (co z oczu, to z serca) na „invisible idiot” (niewidzialny idiota) - out of mind zostało przetłumaczone jako „niespełna rozumu”, gdyż system nie zrozumiał semantyki całego wyrażenia[1].

Cele bezpośrednio związane z przedmiotem tej pracy to wyszukiwanie i wyodrębnianie informacji, ich analiza i na jej podstawie ukazanie zasad modelowania wybranej dziedziny języka. Następne podrozdziały to ogólny opis narzędzi i technik wykorzystywanych do rozwiązywania tych problemów wraz z przykładami istniejących implementacji.

2.5. Pająk internetowy

Definicja 3 Pająk (robot) internetowy (ang. *Web crawler, spider bot*) to program przeszukujący zawartość sieci Internet w zorganizowany i zautomatyzowany sposób[24].



Rysunek 2.1: Schemat działania pająka internetowego

Podstawowym zadaniem wszelkiego rodzaju robotów internetowych jest automatyczna nawigacja po stronach internetowych w sposób określony przez zadane reguły. Każdy pająk ma początkowo określoną pulę adresów, od której zaczyna przeglądanie. Odwiedzając kolejne strony przenosi adresy z tej listy na listę adresów już odwiedzonych. Na witrynie, przez którą przechodzi może szukać nowych linków, które dodaje do listy „do odwiedzenia”, pod warunkiem, że nie ma ich jeszcze na żadnej z list. Robot działa dopóki na liście „do odwiedzenia” znajduje się przynajmniej jeden adres. Specyficzne pająki mogą oczy-

wiście przechodzić przez te same strony wielokrotnie, lub działać w nieskończonej pętli, aby okresowo patrolować określone witryny[24].

Po wejściu na stronę robot sprawdza znajdujący się na serwerze plik robots.txt, który określa do jakiej części serwisu ma dostęp. Zawartość pliku to spis plików i katalogów opisana za pomocą protokołu Robot Exclusion Protocol, niedostępnych dla pająka. Administrator strony może w ten sposób ukryć część serwisu przed wszelkimi automatami, lub też całkowicie odmówić im dostępu, aby na przykład nie obciążały łącza. Plik ten jest jednak publicznie dostępny, więc zapisywanie w nim informacji na temat niepublicznych danych nie jest dobrym pomysłem. Roboty internetowe (w szczególności wszelkiego rodzaju malware, harvestery i spam boty) mogą po prostu ignorować plik robots.txt, więc nie jest on zabezpieczeniem absolutnym. Informacje dla pająka mogą znajdować się także w meta tagach, które określają zasady zachowania robota na danej stronie. Najczęściej stosowane tagi to NOINDEX (uniemożliwia indeksowanie zawartości strony) i NOFOLLOW (uniemożliwia bezpośrednie podążanie za linkami znajdującymi się na stronie)[16].

Pająki internetowe oprócz odwiedzania witryn służą przede wszystkim do zbierania informacji na nich umieszczonych. Wyszukiwarki internetowe wykorzystują rozmaite boty do indeksowania i sprawdzania aktualnej zawartości stron w sieci, aby wykorzystać uzyskane w ten sposób dane do zwrócenia jak najlepszej listy wyników. Często na potrzeby pobierania danych innych niż linki sprawdzany jest także język strony, czy kodowanie znaków, opisane w nagłówku, aby niepotrzebnie nie czytać całej treści[24].

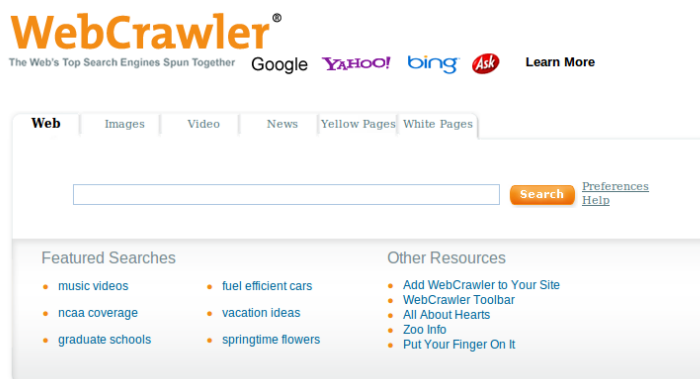
Kilka przykładowych istniejących implementacji robotów internetowych:

- **Googlebot**

Jest to pająk wykorzystywany przez popularną wyszukiwarkę do indeksowania stron w internecie. Działa on w dwóch wersjach: deep crawl (wędruje po witrynach gromadząc i odwiedzając jak najwięcej znalezionych adresów) oraz fresh crawl (odwiedza zmieniające się często strony, w celu aktualizacji związanych z nimi informacji). Aby Googlebot mógł odwiedzić daną stronę, musi istnieć do niej link na innej, przez którą już przechodził. Gromadzi on także statystyki takie jak ilość wystąpień adresu danej strony, która następnie określa jej pozycję w wynikach wyszukiwania. Największym problemem związanym z Googlebotem jest wysokie wykorzystanie transferu, co może doprowadzić nawet do czasowego zawieszenia strony z powodu przekroczenia ograniczeń transferowych. Google udostępnia administratorom narzędzie Webmaster Tools, za pomocą którego można ustawić m.in. częstotliwość wizyt pająka na stronie[22].

- **WebCrawler**

Metawyszukiwarka internetowa agregująca najlepsze wyniki z google, yahoo, ask.com, bing search i innych popularnych wyszukiwarek. Umożliwia wyszukiwanie także obrazków, plików audio i wideo. WebCrawler był pierwszą wyszukiwarką umożliwiającą szukanie w pełnym tekście dokumentów, a nie tylko indeksowanie[25].



Rysunek 2.2: Graficzny interfejs metawyszukiwarki WebCrawler.

- IBM Web Fountain

Web Fountain to jeden z pierwszych projektów, których celem jest skatalogowanie i interpretacja nieustrukturalizowanych lub częściowo ustrukturalizowanych danych tekstowych z Internetu. Jego głównym zadaniem jest poszukiwanie wzorców i zależności na potrzeby statystyczne oraz prezentacji trendów w czasie rzeczywistym[21].



Rysunek 2.3: Graficzny interfejs aplikacji Web Fountain.

- WGet

Jest to program pobierający zasoby z serwerów internetowych będący częścią projektu GNU. Wspiera on obecnie transfer za pomocą protokołów HTTP, HTTPS i FTP. Umożliwia pobieranie rekursywne, konwersję linków umożliwiającą przeglądanie lokalnych plików HTML w trybie offline oraz wspiera proxy. WGet był jednym z pierwszych programów wykorzystujących nagłówki HTTP do wznawiania przerwano transferu. Pająk internetowy wykorzystywany jest przede wszystkim do pobierania rekursywnego, które dokładnie odwzorowuje strukturę zasobów na serwerze. Domyślnie WGet bierze pod uwagę informacje zawarte w pliku robots.txt, ale można go też uruchomić z parametrem `-e robots=off`. Istnieje także możliwość pobierania tylko plików nowszych niż zapisane lokalnie, a także filtrowania zasobów po nazwach za pomocą wyrażeń regularnych[2].

```

anesdaq@amesdaq-dev:~/ugets$ wget -S http://item.slide.com/r/1/102/i/
--16:47:27-- http://item.slide.com/r/1/102/i/BouMSTDK7j8G-MPLdewuVYnPFgI-r0zg
=> `BouMSTDK7j8G-MPLdewuVYnPFgI-r0zg'
Resolving item.slide.com... 64.132.34.79, 64.132.34.72
Connecting to item.slide.com|64.132.34.79|:80... connected.
HTTP request sent, awaiting response...
HTTP/1.0 200 OK
Content-Length: 3453440
Content-Disposition: inline; filename="novo.jpg"; modification-date: Wed, 19 Mar 2008 17:53:35 GMT
Expires: Wed, 24 Sep 2008 23:30:30 GMT
Server: ImageMangle/0.2 Python/2.4.4
Last-Modified: Wed, 19 Mar 2008 17:53:35 GMT
Content-Type: image/jpeg
Date: Fri, 28 Mar 2008 23:46:27 GMT
Connection: keep-alive
Length: 3,453,440 (3.3M) [image/jpeg]

100%[=====] 3,453,440 1,26M/s

16:47:30 (1,26 MB/s) - `BouMSTDK7j8G-MPLdewuVYnPFgI-r0zg' saved [3453440/3453440]
anesdaq@amesdaq-dev:~/ugets$

```

Rysunek 2.4: Wynik działania programu WGet.

- Crawler4j

Jest to prosty, open source'owy interfejs w javie, który pozwala bardzo łatwo i szybko stworzyć własnego pająka internetowego. Udostępnia dwie metody do przeglądania: shouldVisit (określa kryteria definiujące, czy dany URL powinien zostać odwiedzony, czy nie) i visit (określa co ma zostać zrobione w momencie odwiedzenia strony). Dodatkowo należy zdefiniować kontroler, który określi początkowy zbiór adresów, miejsce zapisu danych, czy ilość równoległych wątków[3].

2.6. Ekstrakcja danych

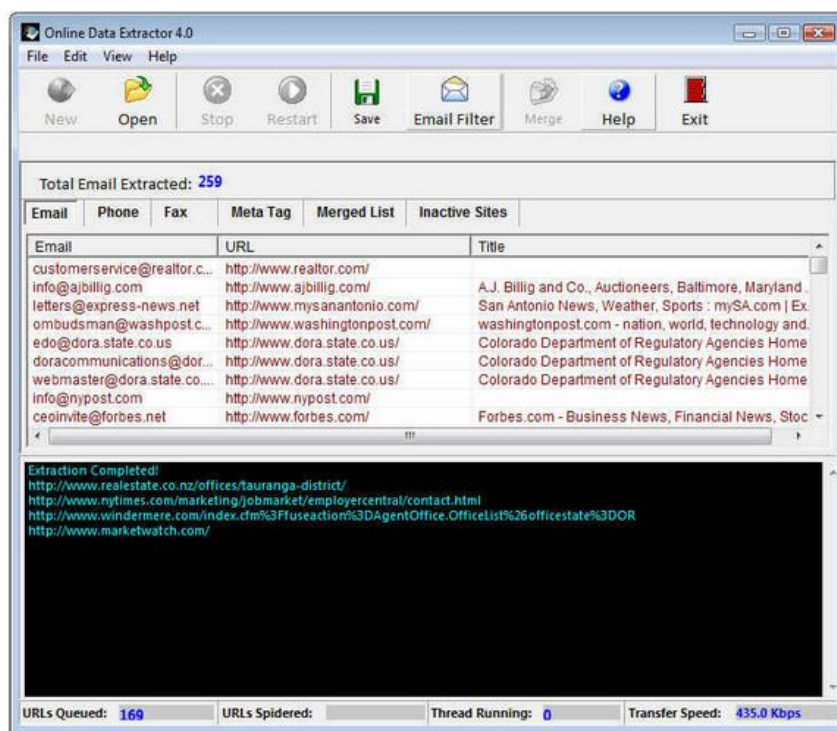
Znalezienie strony, na której mogą być zamieszczone poszukiwane informacje, to dopiero połowa procesu ich pozyskiwania. Są one zazwyczaj otoczone kodem html, który ułatwia użytkownikowi ich przeglądanie z użyciem przeglądarki, ale komplikuje proces ekstrakcji właściwego tekstu przez program. Wydobywanie danych może odbywać się na dwa sposoby. Pierwszy zakłada, że użytkownika interesuje konkretna informacja, której program poszukuje z wykorzystaniem np. słów kluczowych, czy wyrażeń regularnych. Drugi sposób to ekstrakcja całości tekstu w celu zbudowania przez użytkownika bazy wiedzy, z punktu widzenia której istotne jest pozyskanie jak największej ilości danych. Bazę tą wykorzystuje między innymi data mining, który na jej podstawie wyszukuje trendy i formułuje predykcje przyszłych zmian.

Aby uzyskane w ten sposób informacje mogły zostać dalej przetwarzane nie mogą zawierać elementów języka html. Niezbędny jest zatem parser, który znajdzie i usunie wszystkie tagi html, czy skrypty języka JavaScript zostawiając tylko istotne informacje. Teoretycznie możliwe jest wykorzystanie wyrażenia regularnego, znajdującego niepożądane elementy, lecz może ono nie uwzględnić częstych błędów w strukturze stron, jak np. znak < wewnątrz taga, czy też fragmenty tekstu przypominające tagi (równania matematyczne, kod w różnych językach programowania).

Przykłady ekstraktorów danych i parserów html:

- Online Data Extractor

Jest to program, który pozwala na wyszukiwanie danych kontaktowych głównie na potrzeby firm, takich jak adresy e-mail, numery telefonu, czy fax, a także adresy URL i meta tagi). Informacje te znajdują się z pomocą pająka internetowego przeglądającego określone strony, a także korzystającego z pomocy popularnych wyszukiwarek. Umożliwia eksport zgromadzonych danych między innymi do plików txt, xls, csv i htm, a także pozwala na zaawansowaną integrację danych do celów data miningu[10].



Rysunek 2.5: Zrzut ekranu obrazujący działanie aplikacji Online Data Extractor.

- HTML Text Extractor

HTML Text Extractor to komercyjne narzędzie działające pod Windowsem, które umożliwia wydobywanie kodu html dowolnej strony internetowej, w tym także tych, w których zablokowany jest podgląd źródła, czy też są zaszyfrowane. Ekstrakcja następuje w momencie odwiedzenia strony i daje możliwość wyodrębnienia, zaznaczenia i skopiowania samego tekstu, czy to z całej witryny, czy też jej fragmentu[4].



Rysunek 2.6: Zrzut ekranu obrazujący działanie aplikacji HTML Text Extractor.

Na potrzeby aplikacji zajmujących się przetwarzaniem i analizą języka naturalnego, a także pogłębiania wiedzy na temat kultury i literatury tworzy się korpusy tekstów, które zawierają tekst pokazujący wyszukiwane wyrazy i konstrukcje językowe zastosowane w rzeczywistym kontekście. Często też do korpusów udostępnionych online dołączone są mniej lub bardziej zaawansowane narzędzia wyszukiujące. Popularne źródła korpusów tekstów to m.in.:

- PELCRA

Projekt PELCRA realizowany przez Katedrę Języka Angielskiego Uniwersytetu Łódzkiego przy współpracy z Departamentem Językoznawstwa i Współczesnego Języka Angielskiego Uniwersytetu w Lancaster opracowuje korpusy tekstów w języku polskim i angielskim na potrzeby badań m.in. nad leksykografią, translatoryką i lingwistyką (zarówno ogólną jak i komputerową)[12].

Korpusy wchodzące w skład projektu PELCRA:

- Narodowy Korpus Języka Polskiego - Jeden z największych i najpopularniejszych korpusów języka polskiego współtworzony przez Instytut Podstaw Informatyki PAN, Uniwersytet Łódzki, Wydawnictwo Naukowe PWN, oraz Instytut Języka Polskiego PAN.
- Korpus referencyjny języka polskiego PELCRA - Zawiera on około 93 miliony segmentów słów tekstu. Jego częścią jest dostępny zarówno w wersji tekstowej jak i audio Korpus Polszczyzny Konwersacyjnej, który składa się z nagranych rozmów przypadkowych osób uzupełnionych o dane dotyczące uczestników konwersacji.
- ENHIG - Korpus historyczny zawierający teksty staroangielskie, oznakowane pod względem składniowym próbki poezji, prozy i tłumaczeń stworzony na potrzeby prowadzenia badań nad szykiem zdania.

Ośrodek frazeologizmu występuje w korpusie 17898 razy. Podajemy wyniki na podstawie wszystkich współwystąpień. Znalaziono 156 kolokacji spełniających zadane kryteria.				
#	Kolokacja	Pasujące współwystąpienia	Ogółem	Chi ²
1.	goły	gołym_niebem (692), gołe_niebo (14), gołego_nieba (1), niebie_gołym (1), nieba_gołe (1), gołymi_niebem (1),	710	2,833,319.26
2.	rozgwieździć	rozgwieźdzone_niebo (56), rozgwieźdzonym_niebem (23), rozgwieźdzonego_nieba (12), rozgwieźdzonym_niebie (8), niebo_rozgwieźdzone (3), rozgwieźdzone_nieba (1), rozgwieźdzonemu_niebu (1), niebem_rozgwieźdzonym (1),	105	1,452,093.52
3.	rozgwieźdzony	rozgwieźdzone_niebo (56), rozgwieźdzonym_niebem (23), rozgwieźdzonego_nieba (12), rozgwieźdzonym_niebie (8), niebo_rozgwieźdzone (3), rozgwieźdzone_nieba (1), rozgwieźdzonemu_niebu (1), niebem_rozgwieźdzonym (1),	105	1,452,093.52
4.	bezczmurny	bezczmurne_niebo (41), bezczmurnym_niebie (40), bezczmurnego_nieba (21), niebo_bezczmurne (11), bezczmurnym_niebem (8), nieba_bezczmurnego (7), niebie_bezczmurnym (3), bezczmurnemu_niebu (1), niebu_bezczmurnemu (1),	133	1,363,222.63
5.	zachmurzyć	zachmurzone_niebo (44), zachmurzonego_nieba (12), zachmurzonym_niebie (10), niebo_zachmurzone (4), zachmurzonemu_niebu (3), zachmurzonym_niebem (2), niebem_zachmurzonym (1),	76	760,710.15
6.	gwiazdzisty	niebo_gwiazdziste (33), gwiazdziste_niebo (20), gwiazdzistego_nieba (12), gwiazdzistym_niebie (7), gwiazdzistym_niebem (7), nieba_gwiazdzistego (5), niebem_gwiazdzistym (4), gwiazdziste_nieba (2),	90	464,033.9
7.	wygwieździć	wygwieźdzone_niebo (18), wygwieźdzonym_niebem (4), niebo_wygwieźdzone (3), wygwieźdzonego_nieba (2),	27	357,390.39
8.	jasne	jasnego_nieba (249), jasnym_niebie (10), niebie_jasnym (3), niebu_jasnemu (2), jasnym_niebem (2), niebem_jasnym (1), niebie_jasnych (1), jasnemu_niebu (1),	269	289,363.09
9.	błękitny	błękitne_niebo (57), błękitnego_nieba (42), błękitnym_niebie (18), błękitnym_niebem (15), niebo_błękitne (13), błękitnemu_niebu (2), nieba_błękitnymi (1), niebu_błękitne (1),	149	206,065.92
10.	pochmurny	pochmurne_niebo (13), pochmurnego_nieba (10), pochmurnym_niebie (6), niebo_pochmurne (5), pochmurnym_niebem (2), niebie_pochmurnym (1),	37	162,107.31
11.	siódmy	siódmym_niebie (108), siódme_niebo (56), siódmego_nieba (12), niebo_siódma (2), siódmym_niebem (1),	179	80,382.78
12.	gwieździsty	gwieździste_niebo (5), gwieździstego_nieba (2), gwieździstym_niebie (2), gwieździstemu_niebu (1),	10	65,356.2

Rysunek 2.7: Kolokacje wyrazu niebo na podstawie Narodowego Korpusu Języka Polskiego.

- IPI PAN

Korpus Polskiej Akademii Nauk zawiera obecnie około 250 milionów segmentów i dostępny jest darmowo. Do wyszukiwania służy dostępny na licencji GPL, dedykowany program Poliqarp[5].

- Wydawnictwo Naukowe PWN

Wydawnictwo PWN udostępnia swój korpus językowy odpłatnie. Zawiera on ponad 40 milionów słów z fragmentów 386 różnych książek, 977 numerów 185 różnych gazet i czasopism, 84 nagranych rozmów, 207 stron internetowych oraz kilkuset ulotek reklamowych. Wersja demonstracyjna dostępna darmowo składa się z 7.5 miliona słów[26].

3. Relacje między wyrazami w języku polskim

3.1. Części mowy

Głównie ze względów praktycznych wyrazy w języku polskim podzielono na klasy językowe zwane potocznie częściami mowy. Podział ten zależy od szeregu właściwości fleksyjnych, semantycznych i składniowych leksemów. Z punktu widzenia morfologii istotne jest, czy przyporządkowane do danej klasy wyrazy są odmienne i w jaki sposób. Klasyfikacja semantyczna wprowadza natomiast podział słów na samoznające (autosemantyczne) i nie mające znaczenia (synsemantyczne)[18].

Definicja 4 *Części mowy to kategorie gramatyczno-leksykalne, które[18]:*

- *mają te same cechy morfologiczne tzn. każdy leksem należący do danej części mowy ma podobną budowę, podobnie się odmienia np. rzeczowniki przez przypadki i liczby, przymiotniki przez rodzaje, przypadki i liczby,*
- *pełnią identyczną funkcję składniową tzn. występują w roli orzeczenia, podmiotu, przydawki itd.,*
- *mają podobne znaczenie lub niosą podobną treść ogólną np. nazywają zjawiska atmosferyczne, procesy, wielkości, cechy, liczby, stany itd.,*
- *wchodzą w określone relacje (związki) z innymi wyrazami np. przysłówek z czasownikiem tworzą związek przynależności.*

Tradycyjny podział wyróżnia dziesięć części mowy[18]:

- *Rzeczownik*
- *Przysłówek*
- *Przymiotnik*
- *Przyimek*
- *Liczebnik*
- *Czasownik*
- *Spójnik*
- *Zaimek*
- *Partykuła*
- *Wykrzyknik*

Na potrzeby pracy wykorzystane zostały tylko te części mowy, które uznane zostały za niosące najwięcej istotnych informacji oraz pełniące ważne funkcje w zdaniu, a przede wszystkim bezpośrednio wykorzystywane w procesie kojarzenia. Z punktu widzenia kognitywistyki największe znaczenie mają wyrazy, z którymi bezpośrednio można skojarzyć określony obraz oraz takie, za pomocą których można obraz ten opisać.

3.2. Relacje między wyrazami

Aby lepiej zrozumieć prawa, jakie rządzą relacjami między słowami w języku polskim, należy najpierw wprowadzić definicje pojęć takich jak zdanie i składnik, które są niezbędne z punktu widzenia analizy składniowej.

Definicja 5 *Zdanie to odcinek tekstu od wielkiej litery do kropki lub znaku jej równoważnego (pytajnika, wykrzyknika). W odniesieniu do formy mówionej jest to pojedynczy odcinek intonacyjny.[27]*

Definicja 6 *Składnik to najmniejszy element, jaki da się wyodrębnić w procesie analizy składniowej, mogący pełnić jako całość funkcję składniową i wchodzić w związki z innymi takimi elementami.[27]*

Najczęściej mamy do czynienia z sytuacją, w której składnik pokrywa się z wyrazem. Bywają jednak przypadki, gdy ta funkcja składniowa charakteryzuje połączenie wyrazów np. *na pewno, niech będzie, Jan Kowalski*. Takie połączenie nazywane jest składnikiem złożonym.

Wyróżniamy trzy zasadnicze kategorie grup syntaktycznych: podrzędne, współrzędne i egzocentryczne.[15] W grupie podrzędnej członem nadrzędnym jest składnik, którego nie można z niej usunąć, natomiast człon podrzędny niesie dodatkowe informacje, ale nie jest niezbędny z punktu widzenia poprawności składniowej wyrażenia w zdaniu[15]. Przykładowo dla zdania *Babcia upiekła pyszne ciasteczka*. fraza *pyszne ciasteczka* jest grupą podrzędną, gdzie składnikiem nadrzędnym jest wyraz *ciasteczka*, a podrzędnym *pyszne*, gdyż zdanie *Babcia upiekła ciasteczka*. jest poprawne, a *Babcia upiekła pyszne*. już nie. Ten rodzaj grupy składniowej wykorzystany został w pracy, gdyż niesie on najwięcej użytecznych informacji z punktu widzenia relatywnego opisywania rzeczywistości.

Grupa współrzędna zwana także szeregiem charakteryzuje się tym, że składniki są równorzędne i można ją zredukować do jednego składnika (nie będącego spójnikiem)[15]. Przykładowo w zdaniu *Babcia upiekła ciasteczka dla mnie i brata*. szeregiem będzie fragment *mnie i brata*, gdzie współrzędnymi składnikami są wyrazy *mnie* oraz *brata*, a *i* jest spójnikiem. Poprawne są więc zdania *Babcia upiekła ciasteczka dla mnie*. oraz *Babcia upiekła ciasteczka dla brata*., ale *Babcia upiekła ciasteczka dla i*. już nie.

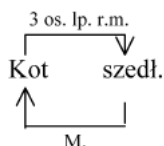
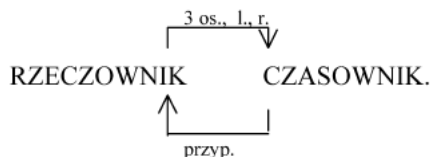
Trzecia grupa zwana egzocentryczną składa się z dwóch składników, z których żaden nie jest redukwalny, a składnik nadrzędny wyznacza formę podrzędnego[15]. Przykładem niech będzie zdanie *Babcia upiekła dla nas ciasteczka*., gdzie grupę egzocentryczną tworzą słowa *dla nas*, gdyż ani zdanie *Babcia upiekła nas ciasteczka*., ani *Babcia upiekła dla ciasteczka*. nie jest poprawne, a wyraz *dla* definiuje przypadek zaimka *my*.

Definicja 7 *Akomodacja syntaktyczna (łac. accomodare - przystosowywać) oznacza dostosowanie się do siebie jednostek w zdaniu. Dostosowanie to polega najczęściej na przyjmowaniu konkretnych wartości kategorii gramatycznych (takich jak rodzaj, liczba, przypadek) przez jednostkę wchodzącą w konstrukcję z inną jednostką.[27]*

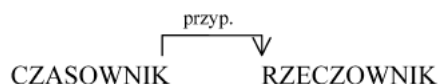
Akomodacje syntaktyczne podzielić można na jednostronne, gdy składnik nadrzędny wymaga od podrzędnego dostosowania formy fleksyjnej, wzajemne, gdy każdy ze składników ma wpływ na formę drugiego, a także międzyfrazowe, w których o formie podrzędnika decyduje nadrzędnik z innej frazy. Reguły zdefiniowane w ramach akomodacji syntaktycznych morfologicznych definiują warunki, które muszą być spełnione przez formy wyrazowe, aby mogły one istnieć wspólnie w jednym zdaniu[27]. Przykładowa taka reguła dla leksemów *kot* i *iść* wyglądała by jak na diagramie 3.1.

Wyraz nadrzędny (*kot*) wymusza na podrzędnym (*iść*) odpowiednią formę gramatyczną: liczbę, rodzaj i osobę. Pozostałe elementy takie jak czas, czy tryb zależą od źródła pozajęzykowego, jakim jest kontekst, do którego odniesiona jest wypowiedź. Uogólniając powyższy przykład można przedstawić ogólną regułę kształtującą zależności między podmiotem a orzeczeniem w zdaniu jak na diagramie 3.2.

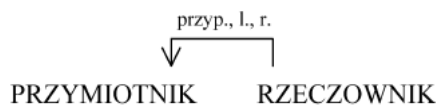
Tego typu zależności akomodacyjnych można zaobserwować dla innych części mowy, jak pokazano na przykładowych diagramach 3.3 i 3.4.

Rysunek 3.1: Przykład oddziaływania akomodacyjnego dla leksemów *kot* i *iść*.

Rysunek 3.2: Schemat uogólnienia oddziaływań akomodacyjnych między podmiotem i orzeczeniem.

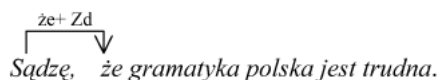


Rysunek 3.3: Schemat uogólnienia oddziaływań akomodacyjnych między czasownikiem i rzeczownikiem.



Rysunek 3.4: Schemat uogólnienia oddziaływań akomodacyjnych między przymiotnikiem i rzeczownikiem.

W gramatyce języka polskiego występuje również akomodacja typu frazy zdaniowej, w której czasownik wymaga zdania podrzędnego poprzedzonego spójnikiem lub zdania pytajno-zależnego np. wyraz *sądzić* w znaczeniu przypuszczać lub wyrażać przekonanie wymaga zdania podrzędnego rozpoczynającego się spójnikiem *że* (*Sądzę, że gramatyka polska jest trudna.*).



Rysunek 3.5: Przykład akomodacji typu frazy zdaniowej.

Zarówno w logice jak i w lingwistyce często wykorzystywane jest pojęcie konotacji.

Definicja 8 W składni języka polskiego konotacją nazywamy zjawisko polegające na tym, że występujący leksem niejako „zapowiada” pojawienie się innych leksemów lub konstrukcji składniowych.[27]

Przykładem może być wypowiedź *Wczoraj mój pies...*, która pozbawiona dokończenia rodzi pytanie *Co się stało z twoim psem?*. Naturalnym jest więc, że bezpośrednio po usłyszeniu takiego niepełnego fragmentu słuchacz oczekiwał będzie czasownika, lub innych uzupełnień które odpowiedzą na jego niezadane pytanie. Najczęściej konotacja syntaktyczna dotyczy zapowiadania określonego typu konstrukcji,

a rzadziej wymusza konkretny leksem, lub określone jego właściwości. W powyższym przykładzie wyraz *pies* „zapowiada” czasownik lub określoną grupę czasownikową, o strukturze, która wynika z cech akomodacyjnych i konotacyjnych danego czasownika[27].

Istnieje prosta metoda sprawdzenia, czy dany składnik podlega konotacji. Jeżeli usunięcie rzeczownego składnika sprawi, że konstrukcja stanie się niepoprawna, czy to w sensie składniowym, czy logicznym, oznacza to że jest on konotowany (wymagany)[27]. Jako przykład usunięcie ze zdania *Wczoraj mój pies pogryzł moje ulubione pantofle*. składnika *pogryzł* spowoduje otrzymanie niepoprawnej struktury: *Wczoraj mój pies moje ulubione pantofle*. Wyraz *pies* jest również konotowany. Zdanie *Wczoraj mój pogryzł moje ulubione pantofle*. w określonym kontekście (zapewniając odpowiedni podmiot domyślny) mogłoby zostać uznane za poprawne, to jednak bezpośrednio znaczenie zostało zmienione. Niekonotowane są natomiast składniki: *wczoraj*, *mój*, *moje* i *ulubione*.

Z. Klemensiewicz w swojej interpretacji budowy zdania, wprowadził następujący dychotomiczny podział związków między składnikami w zdaniu:

- związek główny - związek między podmiotem i orzeczeniem
- związki poboczne - wszystkie pozostałe związki w zdaniu

Powyższy podział czyni podmiot i orzeczenie najważniejszymi składnikami zdania. Klemensiewicz przypisywał dominującą rolę podmiotowi, który uznał za „nadrzędnik związku orzekającego”[7]. Orzeczenie niesie informację o stanie podmiotu, lub wykonywanej przez niego czynności. Wymagane jest jednak uprzednie stwierdzenie istnienia podmiotu. Z punktu widzenia czysto strukturalnego czasownik uznawany jest za nadrzędnik zdania, gdyż jest on jego elementem koniecznym i wystarczającym[27].

Związki poboczne kategoryzowane są następująco:

- związek zgody
- związek rządu
- związek przynależności

Związek zgody zachodzi na przykład między rzeczownikiem i przymiotnikiem lub między dwoma rzeczownikami. Podrzędnik każdorazowo dostosowuje swoją formę fleksyjną do formy nadrzędnika[27]. Przykładowo rzeczownik *kot* narzuca określony rodzaj, liczbę i przypadek przymiotnikowi *perski* - *kota perskiego*, *kotu perskiemu*, *kocie perskim*. W przypadku dwóch rzeczowników sytuacja wygląda podobnie: *pies bernardyn*, *psu bernardynowi*, *psem bernardynem*.

W przypadku związku rządu (głównie z czasownikiem) od rzeczownika wymagany jest ściśle określony przypadek[27]. Przykładem niech będzie wyrażenie *jechać pociągiem*, lub *karmić psa*. Rzeczowniki *pociąg* i *pies* muszą wystąpić odpowiednio w narzędniku i bierniku, aby zdanie było składniowo poprawne. Związek rządu może również zachodzić między rzeczownikiem, a przyimkiem: *przy drodze*, *za górami*, czy też między dwoma rzeczownikami: *szklanka soku*, *chwila namysłu*. Szczególnym przypadkiem związku rządu jest sytuacja, w której oprócz określonego przypadku rzeczownika wymagany jest również odpowiedni przyimek zespalający go z czasownikiem, na przykład: *idziemy na grzyby*, czy też *czytam o lingwistyce*.

Związek przynależności, który charakteryzuje się tym, że forma wyrazu podrzędnego nie jest wymuszana przez wyraz nadrzędny, występuje głównie między czasownikiem, a nieodmienną częścią mowy, na przykład przysłówkiem[27]. Przykładowo: *idę szybko*, *wpadnę jutro*.

4. Opis własnego rozwiązania

Wraz z narodzinami sztucznej inteligencji powstały nowe możliwości automatycznej analizy tekstu. Czatboty, które do tej pory mogły jedynie wykonywać ściśle zaprogramowane operacje w z góry określonych przypadkach zyskały nowe możliwości, które czyniły je nieporównanie bardziej elastycznymi. Dzięki możliwości wnioskowania, program, któremu dostarczono odpowiednio dużą ilość danych był w stanie sam określić reguły stosowane w trakcie rozmowy, a nawet uczyć się ich na bieżąco. Wciąż jednak nie są one niezawodne, a z nowymi możliwościami pojawiły się też nowe wyzwania i problemy. Od idealnego czatbota oczekuje się, żeby pomyślnie przeszedł test Turinga. Oznacza to, że prowadzący z nim konwersację człowiek nie powinien się zorientować, że rozmawia z programem komputerowym.

Aby sprostać temu wyzwaniu czatbot musi spełnić szereg warunków poprawnościowych. Przede wszystkim, budowane przez program zdania muszą mieć prawidłową składnię w języku polskim. Wykorzystywana w tym celu jest tak zwana gramatyka generatywna stworzona przez Noama Chomsky'ego, która jest skończonym zbiorem reguł umożliwiających zbudowanie wszystkich możliwych, poprawnych zdań. Chatbot jest dzięki niej w stanie zweryfikować, czy dana forma jest poprawna składniowo, a także taką formę utworzyć. Jest to możliwe nawet, gdy zdanie podlegające weryfikacji zostało napotkane po raz pierwszy. Sama poprawność językowa nie jest jednak wystarczająca, aby prowadzić rozmowę z człowiekiem. Nie mniej ważna, a o wiele bardziej skomplikowana jest warstwa logiczna wypowiedzi. Słowa, które padają podczas rozmowy często mają więcej niż jedno znaczenie, które nie zawsze program komputerowy jest w stanie wywnioskować z kontekstu zdania, czy nawet większej części konwersacji. Porównania, przenośnie, idiomy, czy nawet pospolite przysłowia lub powiedzenia są poważną przeszkodą, która wciąż stanowi wyzwanie dla programistów.

Aplikacja, która jest przedmiotem tej pracy skupia się na sferze semantycznej rozmowy pomiędzy rzeczonym czatbotem, a użytkownikiem. Dotyka problemu rozumienia znaczenia słów i relacji pomiędzy nimi. W szczególności modeluje strukturę, która umożliwi programowi zbadanie kontekstu używanych przez użytkownika słów przez pryzmat posiadanej bazy wiedzy, aby lepiej zrozumieć ich znaczenie.

Człowiek, który usłyszy lub przeczyta określony wyraz natychmiast kojarzy go z odpowiadającym mu obrazem. Dzięki temu od razu jest w stanie połączyć otrzymany obraz z przypisanymi mu (a więc też i słowu, które reprezentuje) cechami. Daje mu to możliwość opisanego tego co zobaczył, a także porównania z innym słowem (czyli de facto przypisanym mu obrazem). Ta naturalna metoda agregacji jest główną inspiracją przedstawionego dalej rozwiązania. Jakkolwiek program komputerowy nie jest w stanie wyobrazić sobie obrazu w sposób tak plastyczny jak człowiek, to jednak nie jest to przeszkodą, by za pomocą grupowania słów z nim związanych był go w stanie z porównywalną dokładnością opisać.

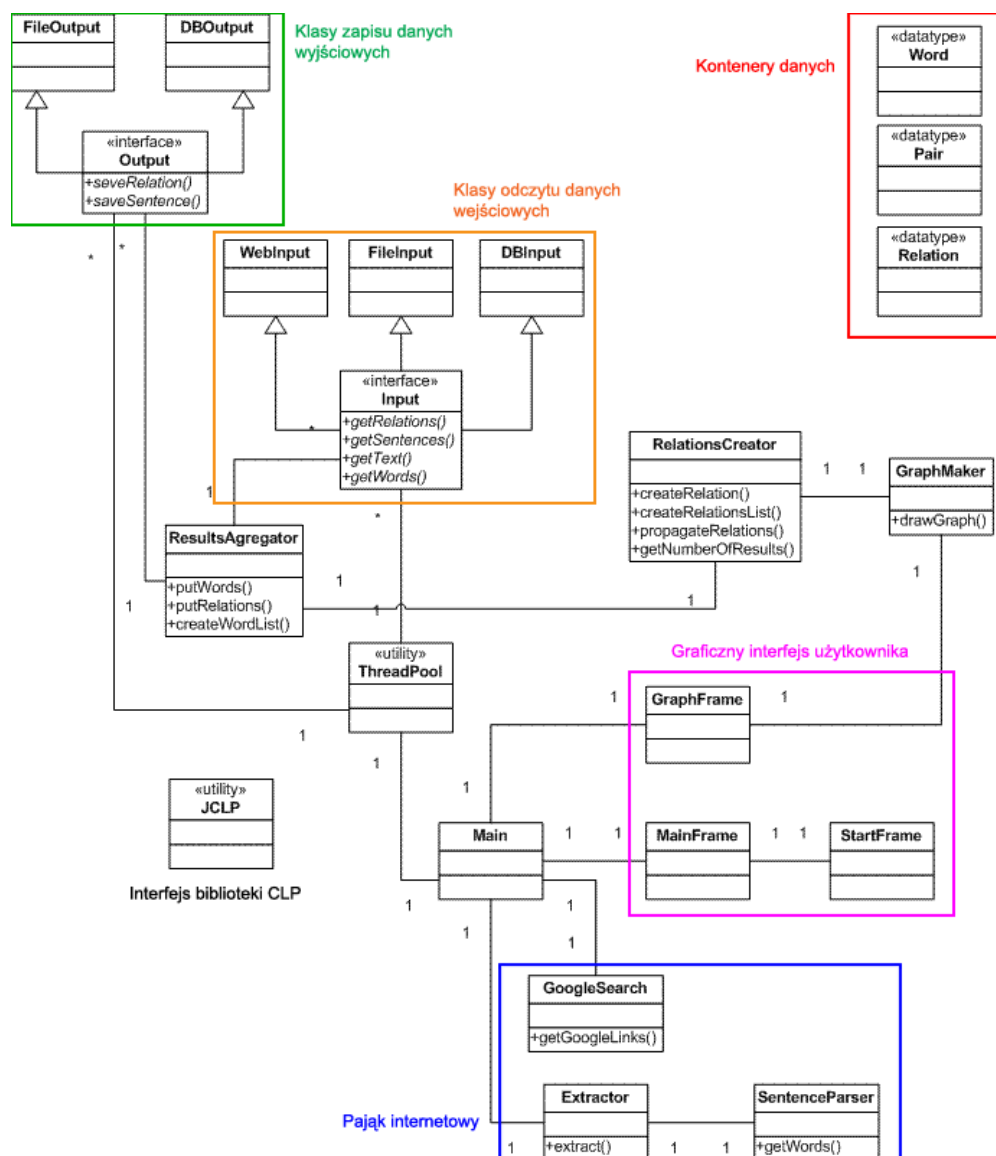
Napisana przeze mnie aplikacja wykorzystuje pająka internetowego, z którego pomocą budowana jest baza językowa będąca podstawowym źródłem wiedzy programu. W zależności od określonych przez użytkownika słów będących korzeniami drzewa, na podstawie przechowywanego tekstu powstaje lista słów, które potencjalnie mogą być z nimi w relacji. Następnym krokiem jest wyodrębnienie tych wyrazów, które łączą się z korzeniami w poprawne językowo i logicznie zależności. Ostatnim etapem działania programu jest zaprezentowanie zdefiniowanych w ten sposób połączeń na grafie przyzwyczajonej lingwistycznych. Szczegółowa realizacja wymienionych kroków, a także ich działanie zostaną opisane w kolejnych podrozdziałach.

4.1. Założenia projektowe

Niniejszy program zakłada, że podane przez użytkownika słowo (korzeń) jest poprawnym słowem w języku polskim napisanym bez błędów. Poprawa błędów i weryfikacja poprawności nie są w zakresie wymagań projektu. Ze względu na ograniczenia zakresu pracy aplikacja obsługuje wyłącznie relacje pomiędzy rzeczownikami, czasownikami, przymiotnikami i przysłówkami. Poprawność relacji badana jest w oparciu o wyniki z wyszukiwarki Google, która nakłada dodatkowe ograniczenia dotyczące maksymalnej ilości zapytań. Problem ten opisany został bardziej szczegółowo w dalszej części pracy.

4.2. Architektura systemu

Aplikacja składa się z czterech głównych elementów: pająka internetowego, modułu odpowiedzialnego za tworzenie relacji, specjalistycznego grafu LHG oraz interfejsu użytkownika. Pająk jest elementem opcjonalnym, gdyż użytkownik ma możliwość stworzenia bazy wiedzy na podstawie innego źródła niż Internet. Uogólniony diagram klas zaprezentowany został na rysunku 4.1.

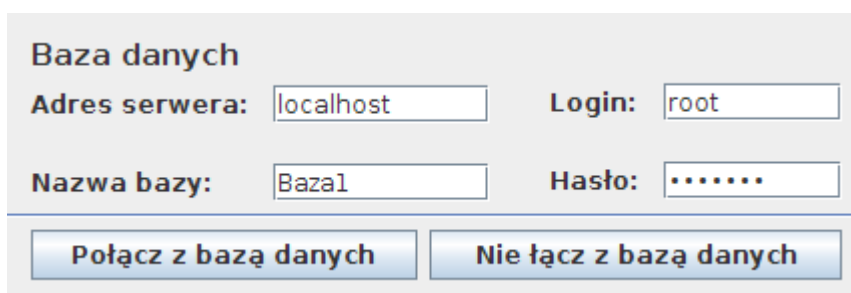


Rysunek 4.1: Diagram klas

4.2.1. Interfejs użytkownika

Podstawowym zadaniem interfejsu użytkownika jest umożliwienie konfiguracji parametrów aplikacji. Dzięki niemu można zdefiniować wejścia i wyjścia danych, a także sposób pozyskiwania i wyświetlania informacji, zależnie od preferencji użytkownika, lub ograniczeń systemu (na przykład brak dostępu do Internetu).

Bezpośrednio po uruchomieniu programu wyświetlone zostaje okno połączenia z bazą danych pokazane na zrzucie ekranu 4.2. Aby nawiązać łączność należy podać adres serwera, nazwę bazy, a także login i hasło. Połączenie z bazą danych nie jest wymagane i można z niego zrezygnować.



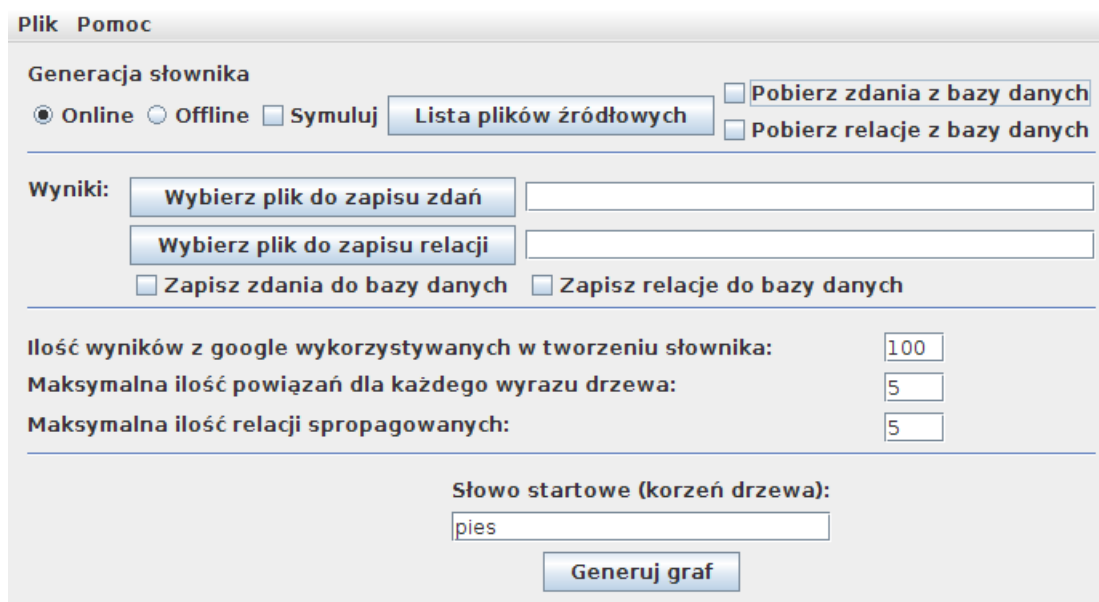
Baza danych

Adres serwera: **Login:**

Nazwa bazy: **Hasło:**

Rysunek 4.2: Okno połączenia z bazą danych

Główne okno programu zaprezentowane na zrzucie ekranu 4.3 jest nieco bardziej skomplikowane. Najważniejszym elementem jest pole służące do podania słów, wokół których budowany będzie specjalistyczny graf przyzwyczajęń lingwistycznych. Każde słowo wpisane po przecinku wykorzystane zostanie podczas wyszukiwania stron internetowych, w oparciu o które powstanie baza wiedzy, a także przy tworzeniu relacji. Możliwe do wpisania są wyłącznie wyrazy znajdujące się w bazie słownikowej biblioteki CLP opisaną dokładniej w rozdziale 4.2.3.



Plik Pomoc

Generacja słownika

Online Offline Symuluj Pobierz zdania z bazy danych

Pobierz relacje z bazy danych

Wyniki:

Zapisz zdania do bazy danych Zapisz relacje do bazy danych

Ilość wyników z google wykorzystywanych w tworzeniu słownika:

Maksymalna ilość powiązań dla każdego wyrazu drzewa:

Maksymalna ilość relacji spropagowanych:

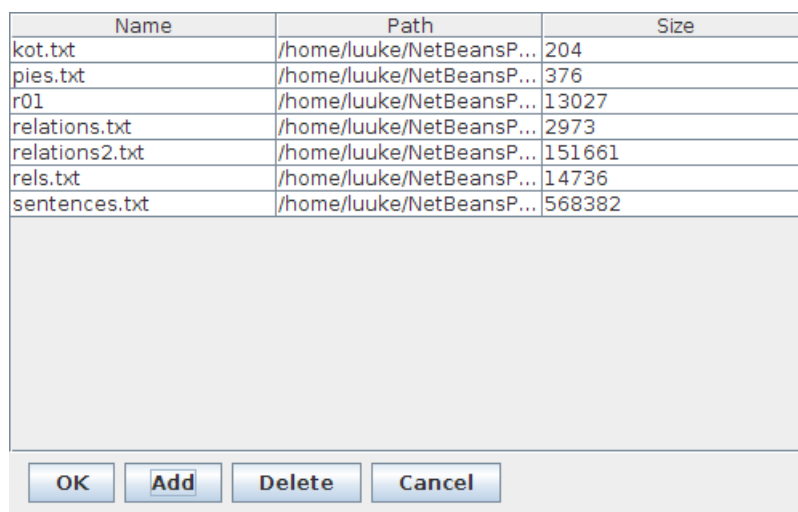
Słowo startowe (korzeń drzewa):

Rysunek 4.3: Główne okno programu

Pośród opcji możliwych do konfiguracji za pomocą głównego okna programu jest także tryb pracy aplikacji. Działanie systemu możliwe jest w jednym z czterech stanów, które definiują sposób pozyskiwania informacji, budowania bazy wiedzy, a także znacząco wpływają na wyświetlony finalnie graf. Tryby pracy aplikacji dzielą się na:

- **ONLINE** - Jest to domyślny tryb działania, w którym aplikacja wykorzystuje połączenie z Internetem w celu zbudowania korpusu tekstu i oceny poprawności utworzonych relacji. Możliwe jest także wykorzystanie plików i bazy danych w celu odczytu i zapisu danych.
- **OFFLINE** - Tryb ten pozwala wyłącznie na zaprezentowanie relacji, które zapisane są w pliku lub bazie danych. Nie wykorzystuje on połączenia z Internetem.
- **Symulator ONLINE** - Symulator pozwala na utworzenie grafu relacji na podstawie korpusu tekstu bez wykorzystywania Internetu do ich ewaluacji. Umożliwia to zaprezentowanie przykładowego grafu, który jest w stanie zobrazować interesujące informacje, jednocześnie nie korzystając z ograniczonych zasobów wyszukiwarki internetowej Google.
- **Symulator OFFLINE** - Tryb ten umożliwia zasymulowanie pełnego działania programu bez dostępu do Internetu z wykorzystaniem korpusów tekstów znajdujących się w plikach lub bazie danych.

W głównym oknie programu zdefiniować można także źródła danych wejściowych. Możliwe jest wczytanie tekstu lub relacji zarówno z plików jak i bazy danych. Okno prezentujące listę plików z danymi do wczytania prezentuje zrzut ekranu 4.4. Jeżeli plik zaczyna się od taga *RELATIONS*, aplikacja usiłuje odczytać zapisane w nim relacje, w przeciwnym wypadku zostanie on uznany jako korpus tekstu. Informacje te można uzyskać także z bazy danych pod warunkiem, że wcześniej ustanowione zostało połączenie. Można je zdefiniować ponownie wybierając z menu *Plik*, a następnie *Połącz z bazą danych*.



Name	Path	Size
kot.txt	/home/luuke/NetBeansP...	204
pies.txt	/home/luuke/NetBeansP...	376
r01	/home/luuke/NetBeansP...	13027
relations.txt	/home/luuke/NetBeansP...	2973
relations2.txt	/home/luuke/NetBeansP...	151661
rels.txt	/home/luuke/NetBeansP...	14736
sentences.txt	/home/luuke/NetBeansP...	568382

Rysunek 4.4: Wybór plików zawierających relacje lub korpusy tekstów.

W trybie *ONLINE* możliwe jest określenie liczby linków jaka zostanie przeglądnięta dla każdego słowa kluczowego. Adresy stron odwiedzonych przez pająka zapisywane są do pliku, aby aplikacja ponownie nie pobierała informacji z tego samego źródła. Użytkownik ma jednak możliwość ręcznego opróżnienia pliku z linkami.

Ważnymi parametrami z punktu widzenia wyświetlania informacji na grafie są parametry definiujące maksymalną liczbę połączeń, jakie zostaną wyświetlone dla pojedynczego słowa, a także maksymalną liczbę połączeń, jakie zostaną wyświetlone dla relacji utworzonych na podstawie pseudo-logicznego rozumowania. Parametry te pozwalają kontrolować ilość informacji na grafie i czynią go bardziej czytelnym. Bardziej szczegółowy opis sposobu przedstawiania danych na grafie LHG znajduje się w rozdziale 4.4.

4.2.2. Pająk internetowy i ekstraktor danych

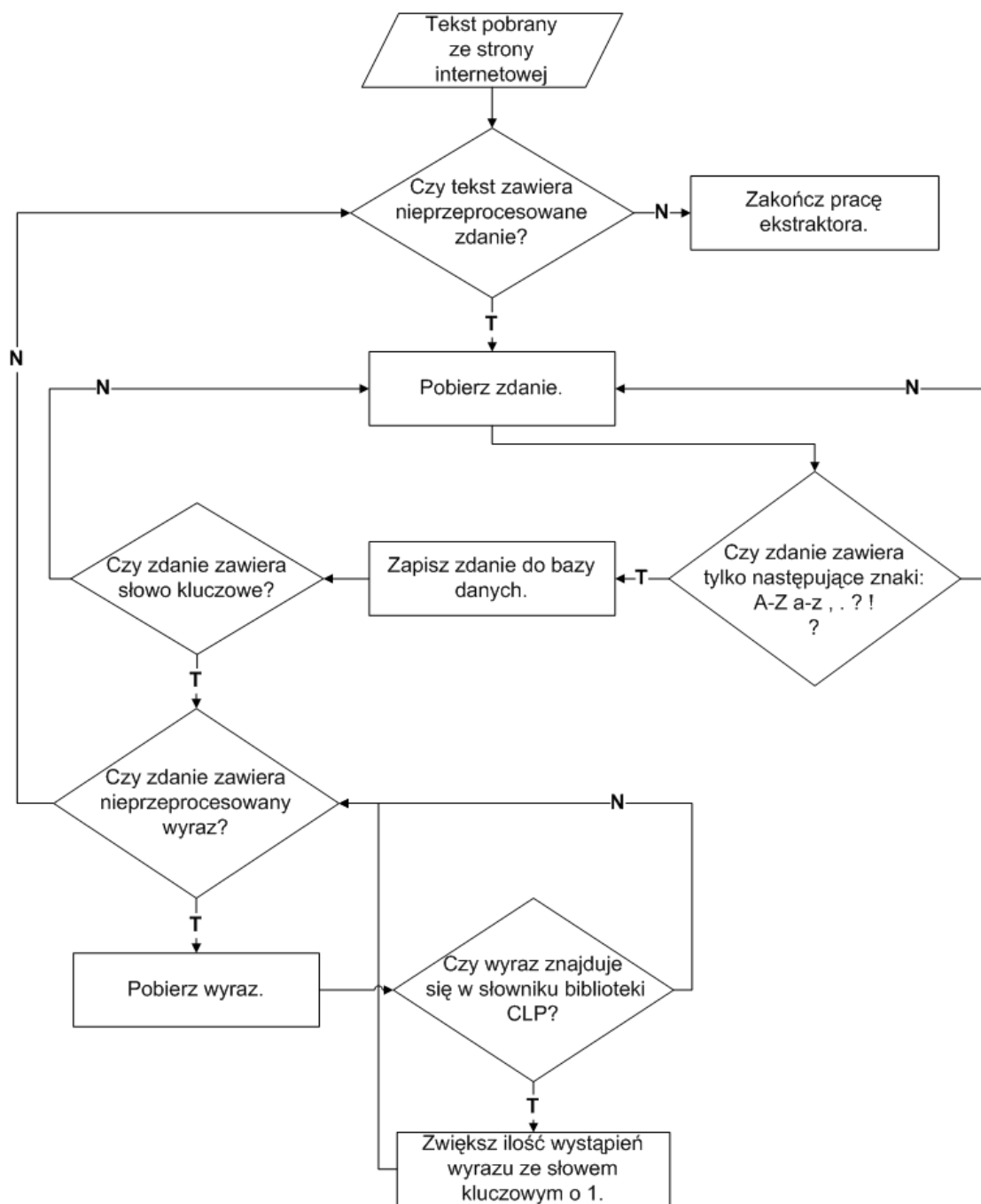
Zdecydowałem się napisać własną, uproszczoną wersję pająka internetowego, która wykorzystuje tylko absolutnie podstawowe funkcjonalności, a jednocześnie w pełni zaspokaja potrzeby aplikacji. Nie zdecydowałem się na żadne z już istniejących rozwiązań, także aby mieć pełną kontrolę nad wykonywanymi przez pająka operacjami. Podstawową różnicą w stosunku do większości crawlerów jest fakt, że aplikacja nie podąża za napotkanymi linkami, a tylko przechodzi przez adresy z określonego źródła, co znacznie skraca czas budowania bazy wiedzy i ogranicza ilość pobieranych, choć zbędnych w danym momencie danych.

Pająk rozpoczyna swoje działanie od znalezienia wszystkich linków na stronie startowej i zapisaniu ich na listę adresów do odwiedzenia. Następnie tworzona jest określona liczba wątków, które po odwiedzeniu usuwają kolejne elementy listy. Pająk wykorzystuje framework `Htmlparser`, który udostępnia metodę pozwalającą na pobranie wyłącznie tekstu z danej witryny pozbawionego tagów HTML. Uzyskany w ten sposób tekst, który wciąż jeszcze może zawierać wiele niepożądanych elementów przekazany zostanie do ekstraktora, który zajmie się jego dokładnym przetworzeniem. Po przekazaniu tekstu strony do ekstraktora wątek dodaje ją na listę odwiedzonych i przechodzi to przetwarzania kolejnego elementu z listy oczekujących, lub kończy działanie, gdy taki element nie istnieje.

Ekstraktor danych, którego schemat działania widoczny jest na diagramie 4.5, jest integralną częścią pająka internetowego. Jego zadaniem jest usunąć wszelkie niepożądane elementy pozostałe po usunięciu tagów HTML i wyodrębnić tekst, który następnie będzie mógł być wykorzystany w głównej części aplikacji. Pierwszym problemem jaki napotkałem było określenie definicji „czystego tekstu”. Wbrew pozorom pozbycie się elementów języka HTML to dopiero początek procesu jego pozyskiwania. Podstawowym komponentem, z którego zbudowany jest każdy tekst jest forma zdaniowa. Przyjąłem, że najmniejsza taka forma musi składać się przynajmniej z dwóch wyrazów, aby wносить jakąkolwiek wartość do bazy wiedzy. Z punktu widzenia relacji wyrażenie złożone z pojedynczego słowa jest mało interesujące.

Dla zmniejszenia stopnia skomplikowania problemu oraz zachowania w miarę przewidywalnej budowy zdań przyjąłem, że powinny się one składać wyłącznie z określonych znaków. Dopuszczalny zbiór znaków zawierał wyłącznie duże i małe litery alfabetu polskiego, a także białe znaki, kropki, wykrzykniki, pytajniki i przecinki. Wszelkie elementy spoza tego zbioru dyskwalifikują zdanie jako źródło informacji. To dość restrykcyjne ograniczenie uzasadnione jest tym, że znaki takie jak -, :, czy () są w stanie znacząco zmienić strukturę zdania, a także utrudnić automatyczne sprawdzenie jego poprawności. Przykładem może być znak pominięcia fragmentu tekstu podczas cytowania: (...), czy też ewentualne pozostałości po tagach HTML, lub fragmentach skryptów które nie zawsze zostaną dokładnie usunięte.

Największym problemem, z jakim zetknąłem się podczas tworzenia ekstraktora było określenie granicy między formami zdaniowymi w ciągłym tekście. Teoretycznie zdanie w języku polskim zaczyna się wielką literą, a kończy kropką, znakiem zapytania lub wykrzyknikiem. Dla programu komputerowego taka definicja jest jednak niewystarczająca. W poprawnym zdaniu, mogą bowiem wystąpić wielkie litery, które nie są początkiem nowego, jak i kropki, które tego zdania nie kończą. Dobrym przykładem może być wypowiedź: "Pan prof. Kowalski udał się na zasłużony urlop." Program szukając nowego zdania znalazłby literę P w słowie Pan, którą słusznie uznałby za jego początek, ale za koniec uznałby kropkę następującą po skrócie prof. Zaproponowane przeze mnie rozwiązanie wyszukuje co prawda zdania zgodnie z tą intuicyjną definicją, ale odrzuca te, które zakończone są skrótem jak na przykład prof., gen., gł., np., itp., a wyszukiwanie następnego rozpoczyna się od kolejnej nie będącej elementem żadnego skrótu kropki. Pozyskane w ten sposób zdania zostają zapisane do bazy danych i plików zdefiniowanych przez użytkownika, a także wykorzystane w procesie budowania grafu relacji.



Rysunek 4.5: Schemat działania ekstraktora danych

4.2.3. Biblioteka CLP

Niezwykle istotnym z punktu widzenia poprawności syntaktycznej elementem aplikacji jest słownik fleksyjny CLP. Został on stworzony przez Katedrę Informatyki Akademii Górniczo-Hutniczej i Katedrę Lingwistyki Komputerowej Uniwersytetu Jagiellońskiego, a jego autorami są Wiesław Lubaszewski, Henryk Wróbel, Marek Gajęcki, Barbara Moskal, Paweł Pietras, Piotr Pisarek i Teresa Rokicka. Dostępny jest on w formie elektronicznej bazy danych SFJP (Słownik Fleksyjny Języka Polskiego) i biblioteki CLP dla języka C.

Podstawowe funkcjonalności biblioteki to:

- rozpoznawanie wyrazu na podstawie jego dowolnej formy fleksyjnej

- dostarczanie informacji o wyrazie
- wygenerowanie form fleksyjnych dla danego wyrazu
- dostarczanie informacji o formie wyrazu

Każdy wyraz w bazie danych reprezentowany jest przez niepowtarzalny numer, dzięki któremu można go jednoznacznie zidentyfikować, wraz z jego etykietą fleksyjną. Dla każdego słowa znajdującego się w bazie biblioteka jest w stanie zwrócić jego numer, lub tablicę numerów, jeśli dopasowanie nie jest jednoznaczne. Na podstawie etykiety fleksyjnej możliwe jest określenie, na przykład części mowy i wzorca odmiany wyrazu. Identyfikator służy do wyznaczenia formy podstawowej danego słowa, a także wszystkich jego możliwych form fleksyjnych wraz z jego pozycją wśród nich. W bazie słownika znajduje się ponad 120 tysięcy wyrazów. Jest ona na początku ładowana do pamięci RAM, co znacznie przyspiesza wyszukiwanie. Istniejąca wersja działa wyłącznie pod systemem GNU/Linux.

4.2.4. Specjalistyczny graf przyzwyczajień lingwistycznych

Głównym zadaniem uproszczonej postaci grafu LHG, która jest wynikiem działania aplikacji jest pokazanie użytkownikowi możliwie jak najbardziej reprezentatywnego fragmentu powstałych relacji. Do wizualizacji wykorzystałem bibliotekę JUNG, gdyż oferuje dość bogate możliwości prezentacji i sporą interakcję z użytkownikiem. Wierzchołki grafu reprezentują formy bazowe słów, które wchodzą w skład relacji, natomiast krawędzie odpowiadają relacjom między połączonymi przez nie wyrazami. Kolor każdego wierzchołka zależy od tego, jaką częścią mowy jest słowo, które reprezentuje. Szczegóły dotyczące przyporządkowania kolorów do poszczególnych części mowy zostały zaprezentowane w tabeli 4.1.

Tablica 4.1: Kolory poszczególnych części mowy

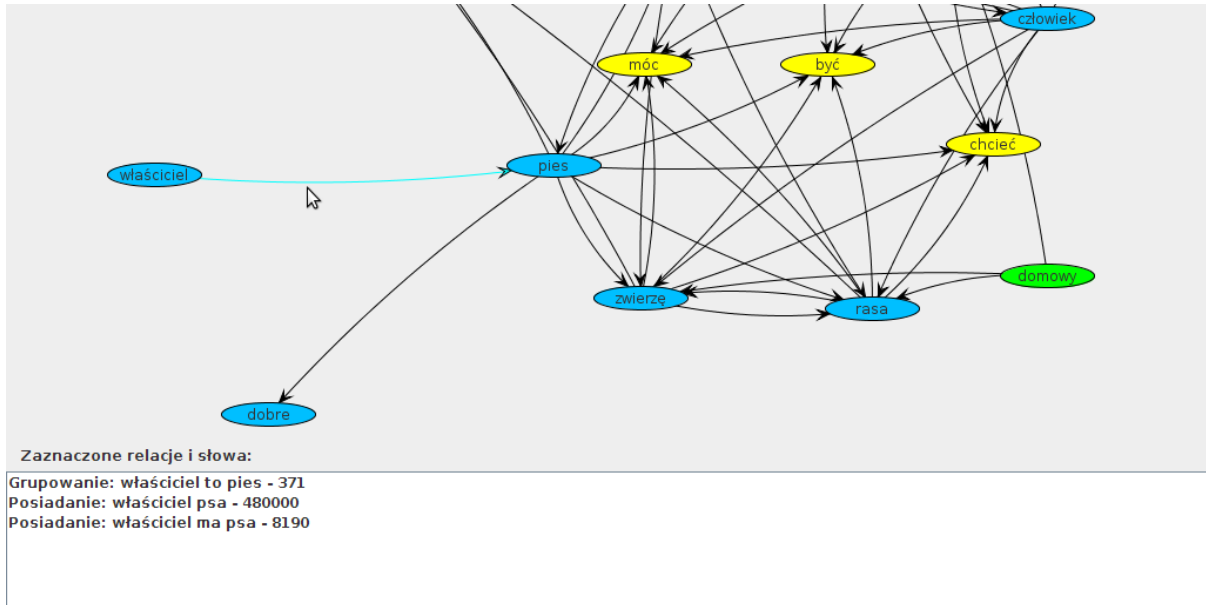
Część mowy	Kolor
rzeczownik	niebieski
czasownik	żółty
przymiotnik	zielony
przysłówek	turkusowy
liczebnik	pomarańczowy

Na grafie nie są prezentowane wszystkie powstałe i zaimportowane relacje, gdyż stałby się on nieczytelny. Parametr ustawiony przez użytkownika w głównym oknie programu określa maksymalną liczbę krawędzi, które są wyświetlane dla pojedynczego wierzchołka. Wszystkie mniej lub bardziej poprawne połączenia między słowami zaprezentowane są na jednej skierowanej krawędzi. Po kliknięciu na nią użytkownik może odczytać wszystkie relacje, które reprezentuje, ich typ, a także nadać im wartość. Przykład wyświetlania szczegółowych informacji o krawędziach został przedstawiony na zrzutach ekranu 4.6 i 4.7. Prezentowane są wyłącznie informacje uznane za najbardziej wartościowe i dające najlepszy obraz działania systemu. Dokładny opis algorytmu decyzyjnego odpowiedzialnego za rysowanie krawędzi opisany został w sekcji 4.4.

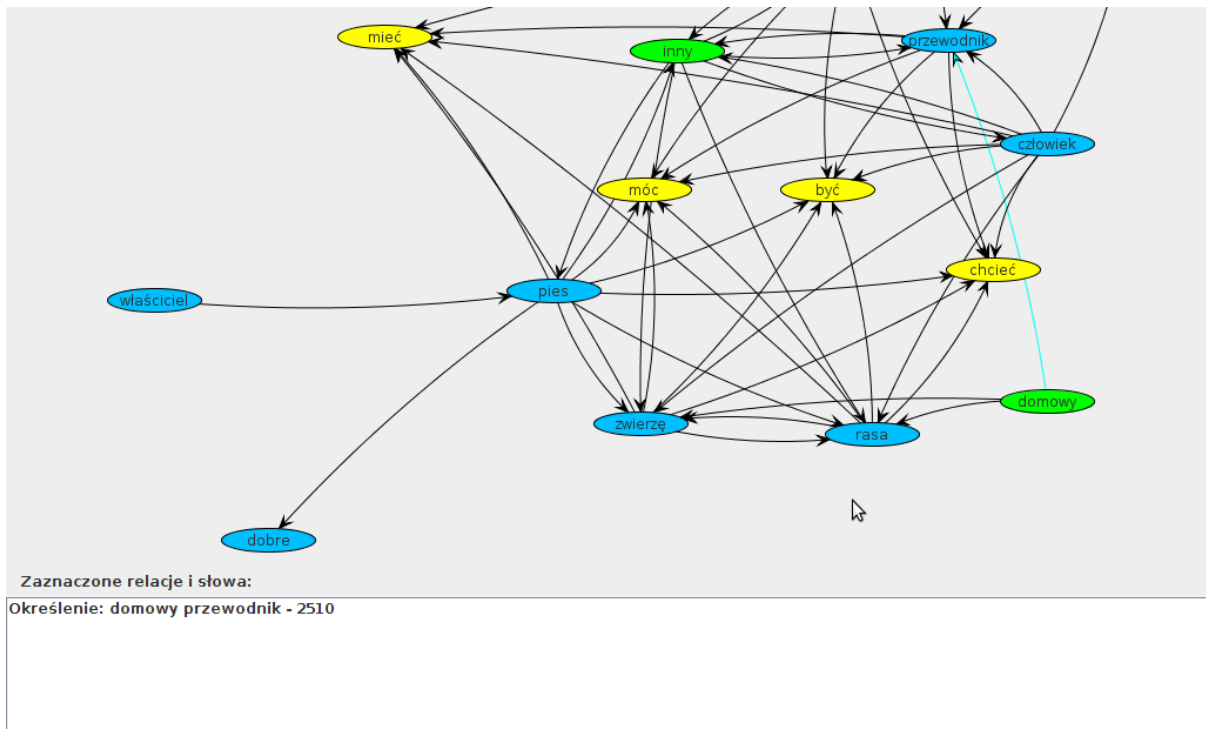
Użytkownik ma dodatkowo możliwość wyświetlenia relacji przechodnich. Są one wynikiem pseudologicznego rozumowania modelującego naturalną przechodniość cech w ramach hierarchii. Nie są one bezpośrednio uzyskane z tekstu źródłowego, ale powstają poprzez wyciągania wniosków z otrzymanej struktury, co maksymalizuje ilość uzyskanych informacji i dokładniej oddaje sposób ludzkiego myślenia. Dodatkowo możliwe jest uzupełnienie otrzymanego grafu nowymi słowami kluczowymi i relacjami przez wybór opcji *Dodaj więcej informacji* z menu Plik. Należy jednak pamiętać, że zaprezentowany zostaje zawsze wyłącznie ograniczony wycinek zbudowanej struktury, co sprawia, że niektóre oczeki-

wane połączenia mogą nie zostać wyświetlone, a zamiast nich zaprezentowane zostaną te, które zostały najlepiej ocenione.

Jeżeli użytkownik chce zobaczyć wszystkie utworzone i pobrane z bazy wiedzy relacje, ma taką możliwość przez wybór opcji *Pokaż listę podstawowych relacji* z menu Plik. Dodatkowo, jeżeli chce zobaczyć relacje, które powstały na skutek wnioskowania, ma możliwość wyboru opcji *Pokaż listę wszystkich relacji*. Przykładowa lista, jaka zostanie zaprezentowana została pokazana na zrzucie ekranu 4.8.



Rysunek 4.6: Szczegóły krawędzi - grupowanie, posiadanie



Rysunek 4.7: Szczegóły krawędzi - określenie

Typ relacji	Wyrażenie	Ocena ▼
Zdolność	człowiek jest	1020000
Zdolność	człowiek może	5970000
Zdolność	człowiek ma	4600000
Zdolność	osoba ma	3480000
Zdolność	osoba jest	2680000
Określenie	jeden człowiek	2250000
Zdolność	pies jest	1630000
Grupowanie	człowiek jest osobą	1550000
Zdolność	człowiek musi	1280000
Zdolność	pies ma	1010000
Określenie	każdy człowiek	985000
Zdolność	pies musi	946000
Posiadanie	pies rasy	934000
Określenie	człowiek jest inny	901000
Zdolność	pies lubi	526000
Określenie	każdy pies	524000
Określenie	człowiek jest jeden	444000
Zdolność	pies może	428000
Określenie	inny człowiek	407000
Zdolność	człowiek wie	398000
Zdolność	zwierzę jest	358000
Zdolność	osoba musi	348000
Określenie	człowiek jest cały	342000
Grupowanie	człowiek jest zwierzęciem	297000
Zdolność	sąsiad ma	293000
Zdolność	osoba może	214000
Określenie	pies jest inny	197000
Zdolność	przewodnik jest	155000
Określenie	osoba jest inny	147000
Posiadanie	osoba ma swoje	143000
Zdolność	osoba lubi	139000
Określenie	osoba jest jeden	134000
Posiadanie	pies ma swoje	126000
Zdolność	rasa jest	116000
Zdolność	zwierzę ma	109000
Zdolność	osoba chce	101000

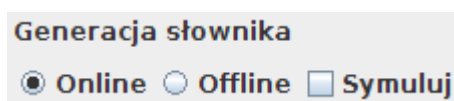
OK

Rysunek 4.8: Lista relacji

4.3. Opis działania aplikacji

Celem działania aplikacji jest stworzenie uproszczonej postaci grafu przyzwyczajęń lingwistycznych, który ilustruje hierarchię wyrazów wraz z ich określeniami oraz pokazuje przechodność niektórych cech w ramach tej struktury. Powstały graf jest wyłącznie uproszczoną odmianą grafu LHG, gdyż prezentuje wyłącznie bazowe formy wyrazów, a także wyłącznie niektóre typy relacji charakterystycznych dla LHG[9]. Z punktu widzenia czatbota, może to być cenne źródło wiedzy, które podczas rozmowy można wykorzystać do analizy kontekstu, w jakim użyte zostało jakieś słowo, zadawania pytań o określone cechy wyrazu, czy też jego opisywania.

4.3.1. Tryb online



Rysunek 4.9: Wybór trybu online

Po wybraniu trybu online (zrzut ekranu 4.10), do wyszukiwarki Google zostaje wysłane zapytanie

o podane słowa-korzenie. Ze zwróconej listy wyników pobierana jest określona w parametrach konfiguracyjnych przez użytkownika liczbę linków, które następnie przekazywane są do kolejnych wątków pająka. Po określeniu, czy strona nadaje się jako źródło informacji następuje ekstrakcja tekstu. Powstaje lista zdań, które trafiają do określonego na początku miejsca docelowego bazy wiedzy dla tekstu. Wątki działają dopóki na liście adresów do odwiedzenia znajdują się jakieś elementy. Adresy, które zostały odwiedzone wypisane są w logach: 4.3.1.

```
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://eszukam.pl/
posokowiec-bawarski-szczenie-pies/debnica-kaszubska/324579/
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://eszukam.pl/
posokowiec-bawarski-szczenie-pies/debnica-kaszubska/324579/
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://eszukam.pl/
posokowiec-bawarski-szczenie-pies/debnica-kaszubska/324579/
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://tanio.pl/Pies-
Baskervilleow-Artur-Conan-Doyle-Audiobook---1-68234-12277036.html
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://tanio.pl/Pies-
Baskervilleow-Artur-Conan-Doyle-Audiobook---1-68234-12277036.html
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://tanio.pl/Pies-
Baskervilleow-Artur-Conan-Doyle-Audiobook---1-68234-12277036.html
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://www.
niepelnosprawni.pl/ledge/x/10686
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://www.
niepelnosprawni.pl/ledge/x/10686
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://www.
niepelnosprawni.pl/ledge/x/10686
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://klarka.blog.onet.
pl/jaki-pan-taki-pies,2,ID419994422,n
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://klarka.blog.onet.
pl/jaki-pan-taki-pies,2,ID419994422,n
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://klarka.blog.onet.
pl/jaki-pan-taki-pies,2,ID419994422,n
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://pies.org.pl/
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://pies.org.pl/
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://pies.org.pl/
2011-03-05 14:34:55 [INFO] Pobieranie słów: http://www.filmweb.pl/
film/Pies+andaluzyjski-1929-906
2011-03-05 14:34:55 [INFO] Pobieranie zdań: http://www.filmweb.pl/
film/Pies+andaluzyjski-1929-906
2011-03-05 14:34:55 [INFO] Pobieranie tekstu: http://www.filmweb.pl/
film/Pies+andaluzyjski-1929-906
```

W następnym kroku wyniki działania pająka oraz wybrane przez użytkownika źródła tekstu są analizowane i wybierane są z nich zdania zawierające podane przez użytkownika słowa w ich dowolnej formie fleksyjnej. Wyszukiwane są wszystkie zwrócone przez bibliotekę CLP formy dla korzeni. Powstaje w ten sposób nowa lista zdań. Wykorzystując bibliotekę CLP wszystkie słowa są następnie sprowadzane do formy bazowej, dzięki czemu ich formy fleksyjne nie będą traktowane jako niezależne słowa. Są one następnie zliczane i sortowane malejąco w zależności od częstości występowania. Dla każdego korzenia powstaje oddzielna lista, a sam korzeń oczywiście znajduje się na jej szczycie, jako że wystąpił on w każdym przetworzonym zdaniu. Za nim zazwyczaj umiejscowione są popularne przymyki np. z, do, w, na i spójniki np. i, a, lub. Program filtruje jednak z pomocą biblioteki CLP otrzymane wyniki, tak aby pozostały wyłącznie określone części mowy. Aplikacja skupia się na najbardziej istotnych i niosących najwięcej informacji, a zatem na rzeczownikach, czasownikach, przymiotnikach, przysłówkach i liczebnikach. Rozróżnienia, jaką częścią mowy jest dane słowo dokonuje biblioteka CLP.

Jako, że biblioteka CLP dopasowując wyraz porównuje go z wszelkimi możliwymi formami, nie zawsze zwrócony wynik odpowiada kontekstowi zdania. Przykładowo często występujące słowo *kilka* identyfikowane jest w pierwszej kolejności jako rzeczownik (jest to nazwa ryby), w drugiej jako przymiotnik, a dopiero w trzeciej kolejności jako zaimek. Podobna sytuacja występuje w przypadku bardzo często używanego słowa *jest*, którego forma bazowa *być* również uznawana jest przez CLP jako rzeczownik (dopełniacz liczby mnogiej rzeczownika *bycie*), a dopiero kolejny zwrócony wynik dopasowania jest czasownikiem. Problem ten rozwiązany został przez rozróżnienie na etapie przetwarzania tekstu nie tylko słów, ale także ich części mowy, co sprawia, że homonimy są rozróżnialne i traktowane jako oddzielne wyrazy.

W tak posortowanej liście wyrazy znajdujące się wysoko mają dużą szansę być w relacji ze słowem kluczowym, jako że często były użyte z nim w jednym zdaniu. Takie statystyczne podejście utrudnia co prawda stwierdzenie koneksji z rzadziej używanymi słowami, ale omija problemy związane z załościami składni języka polskiego. Nie jest konieczne budowanie skomplikowanego parsera zdań z wyszukanyymi regułami uwzględniającymi niuanse językowe. Związki między wyrazami ustalane są w ramach jednego zdania. Frazy, w których klucz został zamieniony zaimkiem lub wystąpił jako podmiot domyślny w ogóle się tu nie znajdują, gdyż zostaną odfiltrowane jako niezawierające korzenia.

Elementy z posortowanej listy potencjalnych kandydatów do relacji ze słowem kluczowym są kolejno w odpowiedniej formie łączone z korzeniem (także w odpowiedniej formie) z wykorzystaniem szablonów relacji. W zależności od typu relacji do utworzonej w ten sposób formy może zostać dodany spójnik. Szablony relacji definiują typowe zależności pomiędzy dwoma wyrazami występującymi w języku polskim i określają części mowy jakie wchodzą w ich skład, ich formę fleksyjną oraz kolejność. Nie muszą być one skomplikowane, ani brać pod uwagę innych słów, które mogłyby zaburzyć ustalony schemat. Ważne jest, aby sam układ i forma dwóch składowych wyrazów był poprawny.

Dla każdego schematu utworzona przez niego fraza jest przekazywana do wyszukiwarki Google, po czym przypisywana jest jej wartość zależna od ilości zwróconych wyników zapytania. Dopóki więc dana relacja między dwoma wyrazami występuje w niezmienionej, dokładnie takiej jak określona formie odpowiednio często na stronach indeksowanych przez wyszukiwarkę, program zakłada, że jest ona poprawna. Każda para wyrazów (korzeń i słowo z listy wyrazów często z nim występujących w jednym zdaniu) może być w potencjalnie dowolnej zależności, a więc dla odpowiednich części mowy sprawdzane są wszystkie możliwe szablony relacji. Te, dla których Google zwróci odpowiednio dużą liczbę wyników zostają uznane za poprawne. Szablony powinny odzwierciedlać najczęściej występujące struktury, gdyż każdy kolejny zwiększa liczbę zapytań o poprawność utworzonej relacji, co bezpośrednio przekłada się na maksymalną ilość uzyskanych wartościowych informacji.

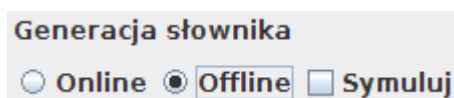
Przykładowo dla korzenia *pies* i słowa z listy *łapa*, jako że oba wyrazy są rzeczownikami sprawdzone zostaną możliwe relacje pomiędzy dwoma rzeczownikami. Tak więc dla relacji posiadania o szablonie *MIANOWNIK1 + ma + BIERNIK2* ułożona zostanie fraza *pies ma łapę*, dla której Google zwróci X wyników. Sprawdzona zostanie też relacja odwrotna, a zatem *łapa ma psa*. Kolejną potencjalną relacją pomiędzy dwoma rzeczownikami jest grupowanie, czyli przypadek, gdy jeden wyraz należy do podzbioru (czyli jest przykładem) wyrazu drugiego. Szablon tej zależności wygląda następująco: *MIANOWNIK1 + jest + NARZĘDNIK2*, a zatem *pies jest łapą*, czy też *łapa jest psem* nie zwrócą zbyt wielu wyników. Dla pary *kot* i *zwierzę* fraza *kot jest zwierzęciem* oceniona zostanie wysoko, a odwrotność *zwierzę jest kotem* już nie.

Algorytm sprawdza poprawność relacji wysyłając kolejne zapytania do wyszukiwarki do momentu, aż nie wyczerpią się wyrażenia, które wymagają oceny, lub nie zostanie zwrócony błąd oznaczający przekroczenie maksymalnej ilości zapytań. Wynikiem jego działania jest lista relacji z przyporządkowanymi im ocenami. Każda uzyskana w ten sposób informacja jest cenna. Otrzymane wyniki dają informacje nie tylko o tym, które wyrażenia są poprawne, ale także o tym, które poprawne nie są. Jako, że nigdy nie ma absolutnej pewności, czy dana relacja ma sens, czy nie, przyporządkowana jej ocena pozwala ustalić wyłącznie prawdopodobieństwo. Informacje o utworzonych relacjach wypisywane są w logach aplikacji: 4.3.1.

Listing 4.1: Fragment logów wyświetlanych podczas tworzenia relacji.

```
2011-03-05 13:33:28 [INFO] osoba lubi: 139000
2011-03-05 13:34:36 [INFO] osoba ma rasę: 0
2011-03-05 13:34:38 [INFO] cały jak osoba: 0
2011-03-05 13:34:40 [INFO] osoba wie: 93500
2011-03-05 13:34:42 [INFO] osoba jest sąsiadem: 0
2011-03-05 13:34:44 [INFO] osoba to swoje: 0
2011-03-05 13:34:45 [INFO] lud ma osobę: 0
2011-03-05 13:35:04 [INFO] osoba zrobi: 19400
2011-03-05 13:35:37 [INFO] człowiek jest osobą: 1550000
2011-03-05 13:36:44 [INFO] każdy jak człowiek: 26800
2011-03-05 13:37:15 [INFO] swoje jest zwierzęciem: 0
2011-03-05 13:37:16 [INFO] swoje jest rasą: 0
2011-03-05 13:37:18 [INFO] swoje jest sąsiadem: 0
2011-03-05 13:37:20 [INFO] swoje jest człowiekiem: 0
2011-03-05 13:37:33 [INFO] człowiek wie: 398000
2011-03-05 13:37:47 [INFO] oko to zwierzę: 0
2011-03-05 13:37:49 [INFO] oko to rasa: 0
2011-03-05 13:37:50 [INFO] oko to sąsiad: 0
2011-03-05 13:38:27 [INFO] nowy sąsiad: 50500
2011-03-05 13:38:28 [INFO] nowy człowiek: 29200
2011-03-05 13:38:30 [INFO] oko jest zwierzęciem: 0
2011-03-05 13:38:32 [INFO] oko jest rasą: 0
2011-03-05 13:38:34 [INFO] oko jest sąsiadem: 0
2011-03-05 13:38:35 [INFO] oko jest człowiekiem: 0
2011-03-05 13:40:17 [INFO] człowiek musi: 1280000
2011-03-05 13:41:14 [INFO] oko człowieka: 50500
2011-03-05 13:41:16 [INFO] pasterski jak zwierzę: 0
2011-03-05 13:41:18 [INFO] pasterski jak rasa: 0
2011-03-05 13:41:48 [INFO] człowiek jest cały: 342000
```

4.3.2. Tryb offline



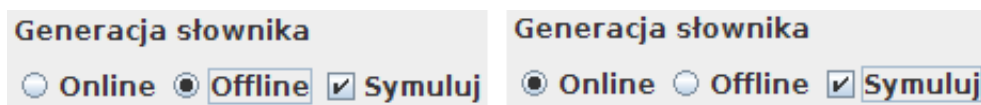
Rysunek 4.10: Wybór trybu offline

Główną różnicą między trybem offline, a trybem online jest brak konieczności łączenia się z Internetem. W tym stanie aplikacja nie wykorzystuje pająka internetowego i nie wylicza ocen relacji, a jedynie wyświetla te, które zostały pobrane z bazy danych lub plików. Działanie programu w tym trybie jest naturalnie o wiele szybsze. Jako, że nie jest możliwe określenie poprawności relacji, użytkownik nie musi podawać słowa kluczowego. Graf zostanie zbudowany na podstawie informacji, które już zostały zebrane. W trybie tym istnieje także możliwość wczytania relacji z plików i ich zapis do bazy danych.

4.3.3. Symulator

Tryb symulatora zaimplementowany został wyłącznie na potrzeby prezentacji idei działania aplikacji. Nadaje on nowo utworzonym relacjom stałe oceny poprawności, aby zostały one wyświetlone na

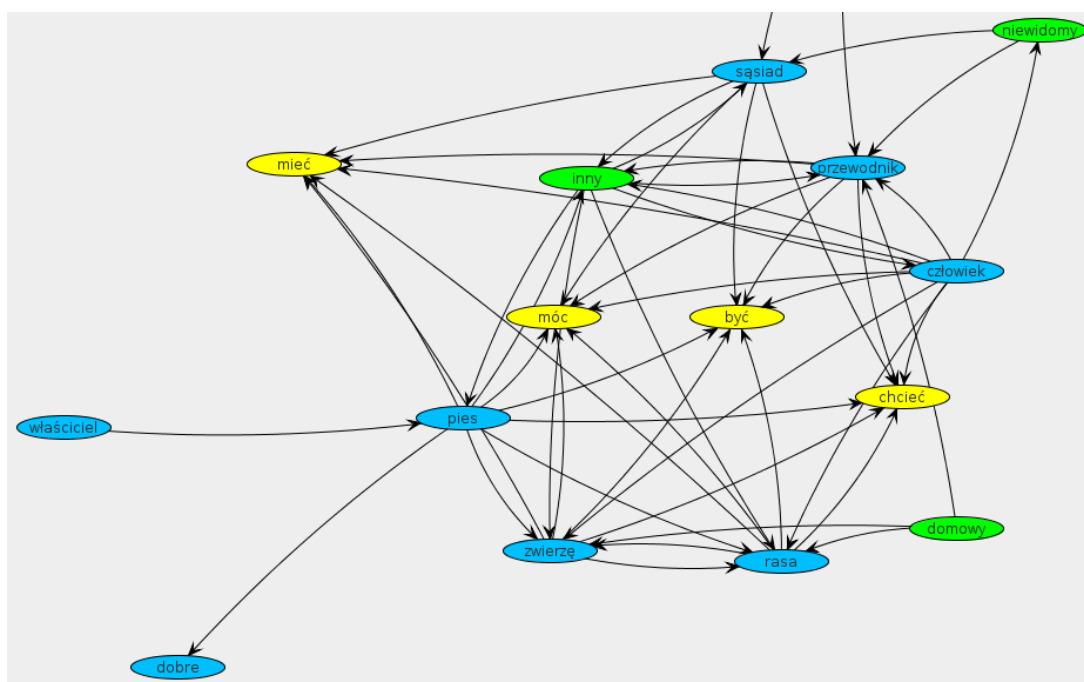
grafie. Ustalanie poprawności w ten sposób oczywiście nie ma nic wspólnego z rzeczywistością, pozwala jednak przedstawić sposób działania programu i wszystkie jego funkcje bez konieczności korzystania z mechanizmu sprawdzania poprawności. Zrzut ekranu 4.11 przedstawia wybór symulacji sprawdzania poprawności odpowiednio dla trybu offline i online.



Rysunek 4.11: Wybór trybu pracy z symulatorem

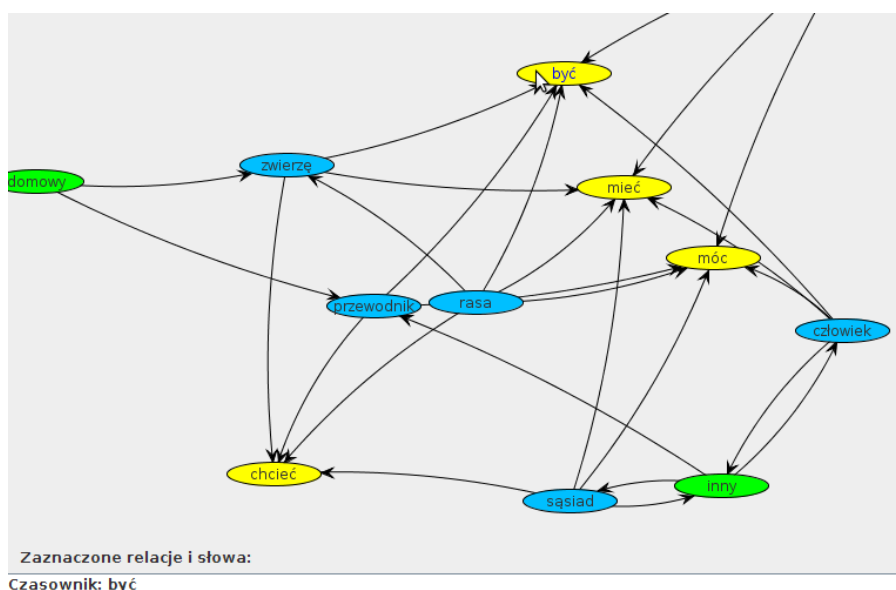
4.4. Generacja specjalistycznego grafu LHG

Ostatnim etapem działania aplikacji jest wygenerowanie grafu przyzwyczajęń lingwistycznych w uproszczonej postaci, który zaprezentuje znalezione relacje z przypisanymi do nich wartościami. Po zakończeniu przetwarzania danych powstaje lista relacji złożona zarówno z wczytanych przez użytkownika, jak i uzyskanych na podstawie analizowanego tekstu. Słowa, które wchodzą w skład wyrażen stają się węzłami i zapisane są w formie bazowej. W zależności od tego, jaką część mowy reprezentują, przyporządkowany jest im odpowiedni kolor zgodnie z tabelą: 4.1. Słowa z tym samym źródłosłowem, lecz będące różnymi częściami mowy są też różnymi wierzchołkami. Przykład grafu obrazującego zróżnicowanie kolorów w zależności od części mowy, jaką reprezentuje słowo w wierzchołku pokazuje zrzut ekranu 4.12.



Rysunek 4.12: Przykładowy graf z wierzchołkami będącymi różnymi częściami mowy.

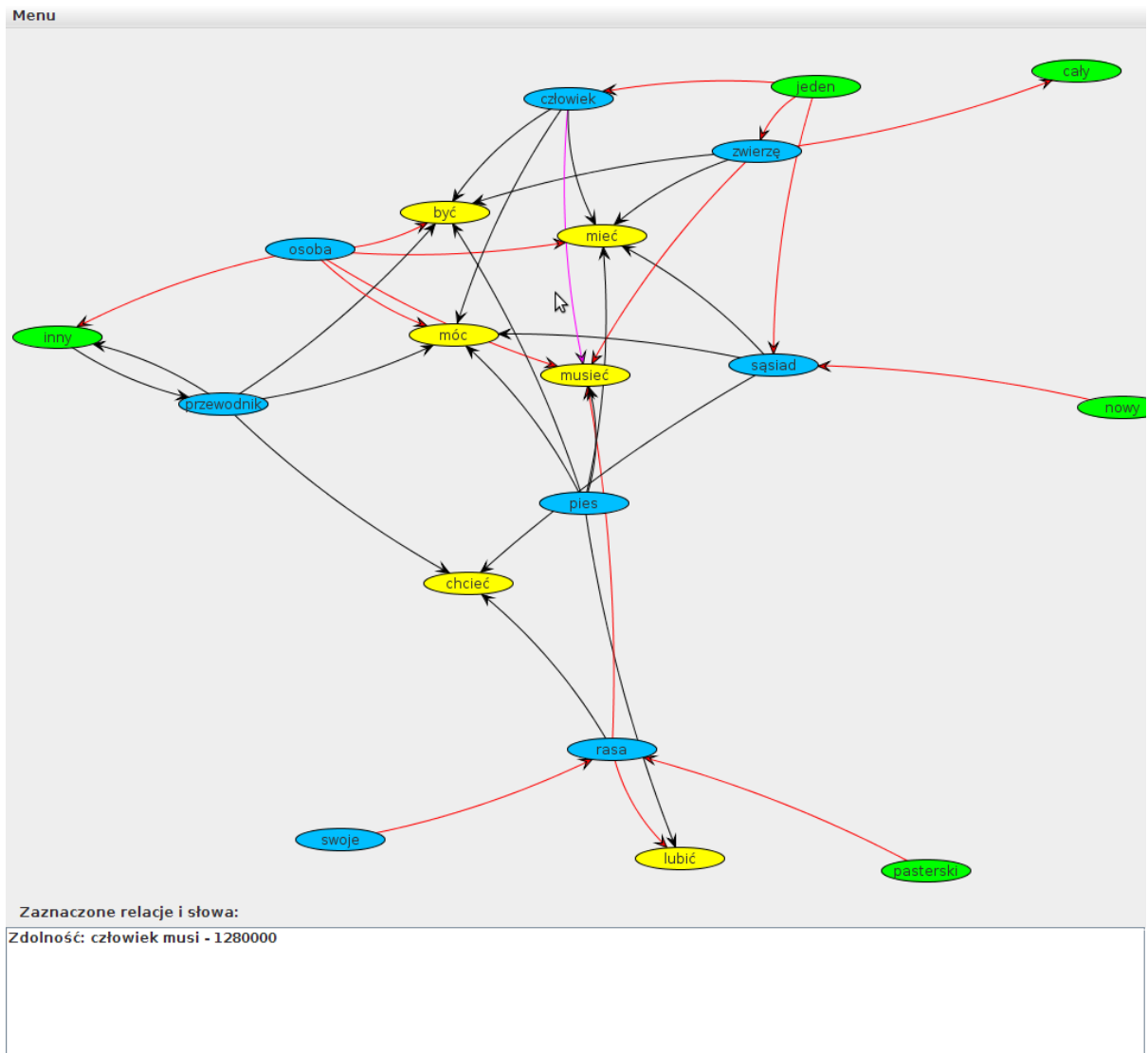
Poprzez kliknięcie na wierzchołku grafu użytkownik ma możliwość odczytania, jaką częścią mowy jest słowo, które dany wierzchołek reprezentuje. Informacja ta nie tylko określona jest za pomocą koloru zgodnego z tabelką 4.1, ale także jest wyświetlana w panelu informacyjnym, jak pokazuje zrzut ekranu 4.13.



Rysunek 4.13: Informacja o konkretnym wierzchołku wyświetlona na grafie.

Krawędzie grafu definiują połączenia między dwoma wyrazami. Nie odpowiadają one bezpośrednio relacji, ale raczej grupie relacji między wierzchołkami. Zwrot krawędzi określa rolę jaką dany węzeł pełni w ich połączeniu. Zawsze jest on skierowany od nadrzędnego wyrazu w wyrażeniu do podrzędnego. Między dwoma wierzchołkami może zatem wystąpić co najwyżej dwie krawędzie o różnych zwrotach. Relacje, które uznane zostały za niepoprawne (mają ocenę 0) nie są wyświetlane na grafie, co znacznie poprawia jego czytelność. Decyzja, czy dane relacje utworzą krawędź, która następnie zostanie wyświetlona na grafie zależy od średniej oceny relacji między słowami reprezentowanymi przez wierzchołki. Ich ilość jest ograniczona przez parametry opisane w sekcji 4.2.1. Tylko krawędzie o największej średniej ocenie relacji zostaną wyświetlone.

Relacje utworzone na podstawie analizy hierarchicznej struktury istniejących powiązań są wyświetlane tylko na żądanie użytkownika, aby nie zaciemniać grafu. Powstałe w ten sposób krawędzie oznaczone są kolorem czerwonym, aby łatwo można było je odróżnić i prześledzić ścieżkę rozumowania aplikacji. Liczba dorysowanych w ten sposób krawędzi zależy od zdefiniowanego przez użytkownika w głównym oknie programu parametru *Maksymalna ilość relacji spropagowanych* opisanego w sekcji 4.2.1. Graf z widocznymi spropagowanymi relacjami został zaprezentowany na zrzucie ekranu 4.14.

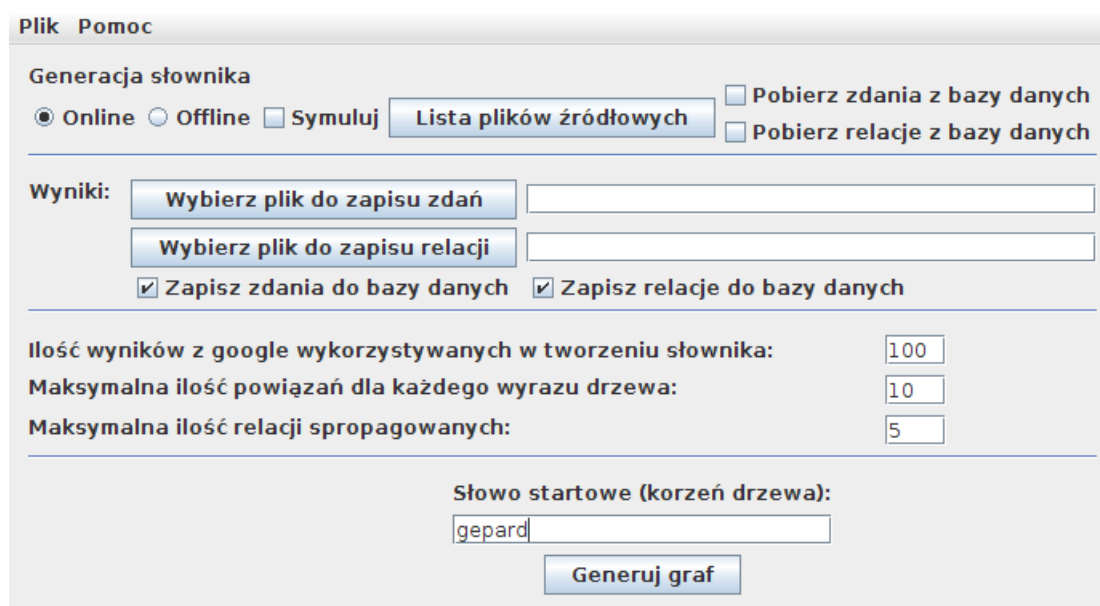


Rysunek 4.14: Widok pokazujący spropagowane relacje na grafie.

5. Prezentacja działania aplikacji

Aby dokładnie zaprezentować działanie aplikacji prześledźmy przykład jej wykorzystania w celu wygenerowania struktury relacji dla zadanych słów. Załóżmy, że czatbot rozmawiający właśnie z użytkownikiem chce przeanalizować otrzymaną wypowiedź pod kątem określonych słów kluczowych. Jest to konieczne, jeżeli chce nawiązać do cech charakterystycznych tematu rozmowy, lub też sformułować inteligentne pytanie. To, czy czatbot „zrozumie” niektóre elementy rozmowy zależeć będzie od tego jakie będą jego skojarzenia, czyli relacje, które posiada w bazie danych.

Przykładowo dla wyrazu *gepard* dane wejściowe aplikacji mogą wyglądać jak na zrzucie ekranu 5.1. Oczywiście możliwe jest wykorzystanie większej ilości zasobów (zarówno stron internetowych, jak i korpusów tekstów).



Plik Pomoc

Generacja słownika

Online Offline Symuluj Pobierz zdania z bazy danych Pobierz relacje z bazy danych

Lista plików źródłowych

Wyniki:

Zapisz zdania do bazy danych Zapisz relacje do bazy danych

Ilość wyników z google wykorzystywanych w tworzeniu słownika:

Maksymalna ilość powiązań dla każdego wyrazu drzewa:

Maksymalna ilość relacji spropagowanych:

Słowo startowe (korzeń drzewa):

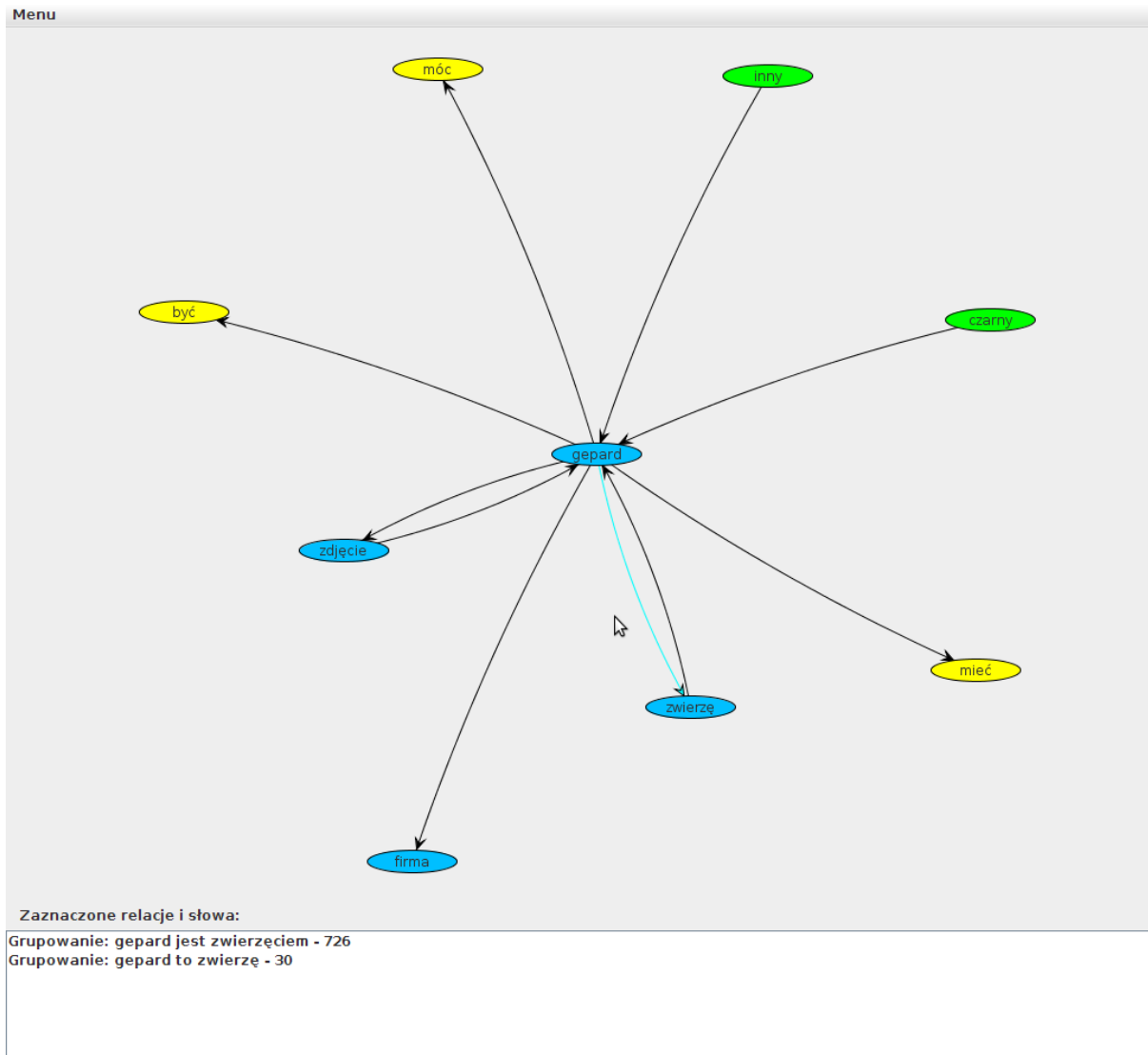
Rysunek 5.1: Przykładowe dane wejściowe dla programu.

Ze 100 wyników zwróconych przez wyszukiwarkę Google dla słowa *gepard* pobierany jest tekst, który następnie dzielony jest na zdania, a te z kolei na wyrazy, które pojawiają się w kontekście słowa *gepard*. Najczęściej pojawiające się wyrazy są następnie podstawiane do szablonów relacji wraz ze słowem *gepard* i ich poprawność jest sprawdzana za pomocą wyszukiwarki Google. Im więcej wyników zostanie zwróconych dla określonej frazy, tym częściej występuje ona w Internecie, co pośrednio przekłada się także na jej poprawność językową. Gdy do wyszukiwarki trafi zbyt wiele zapytań, tworzenie nowych relacji zostaje przerwane i wyświetlone zostają uzyskane do tej pory wyniki. Sytuację taką obrazuje wycinek logów 5.

```
2011-04-09 11:32:56 [INFO] używany gepard: 17
2011-04-09 11:32:58 [INFO] wielki jak gepard: 0
2011-04-09 11:32:58 [INFO] Zakończono tworzenie relacji podstawowych.
```

2011-04-09 11:32:59 [INFO] firma ma zdjęcie: 0
 2011-04-09 11:33:01 [INFO] firma ma zwierzę: 0
 2011-04-09 11:33:09 [ERROR] Przekroczono limit zapytań do google.
 2011-04-09 11:33:09 [INFO] człowiek to firma: 0
 2011-04-09 11:33:09 [INFO] Zakończono tworzenie relacji przechodnich.

Jak widać na zrzucie ekranu 5.2 utworzony został graf przyzwyczajęń lingwistycznych, który obrazuje wycinek relacji uzyskanych dla wyrazu *gepard* przy wykorzystaniu zdefiniowanej wcześniej bazy wiedzy. Zgodnie z parametrem, jaki został podany, maksymalnie 10 krawędzi łączy się z pojedynczym wierzchołkiem, co przy ocenach utworzonych relacji pozwala wyświetlić 8 wierzchołków reprezentujących słowa związane relacjami z korzeniem. Popularne czasowniki takie jak *być*, *móc* i *mieć* niebezpiecznie łączą się z wieloma rzeczownikami. Rzeczownik *firma* pojawia się, gdyż *GEPARD TRANS* jest nazwą dobrze rozreklamowanej w Internecie polskiej firmy przewozowej. Kolor czarny został przypisany gepardowi, choć w rzeczywistości odnosi się on raczej do cętek niż do samego geparda.



Rysunek 5.2: Graf relacji dla słowa *gepard*.

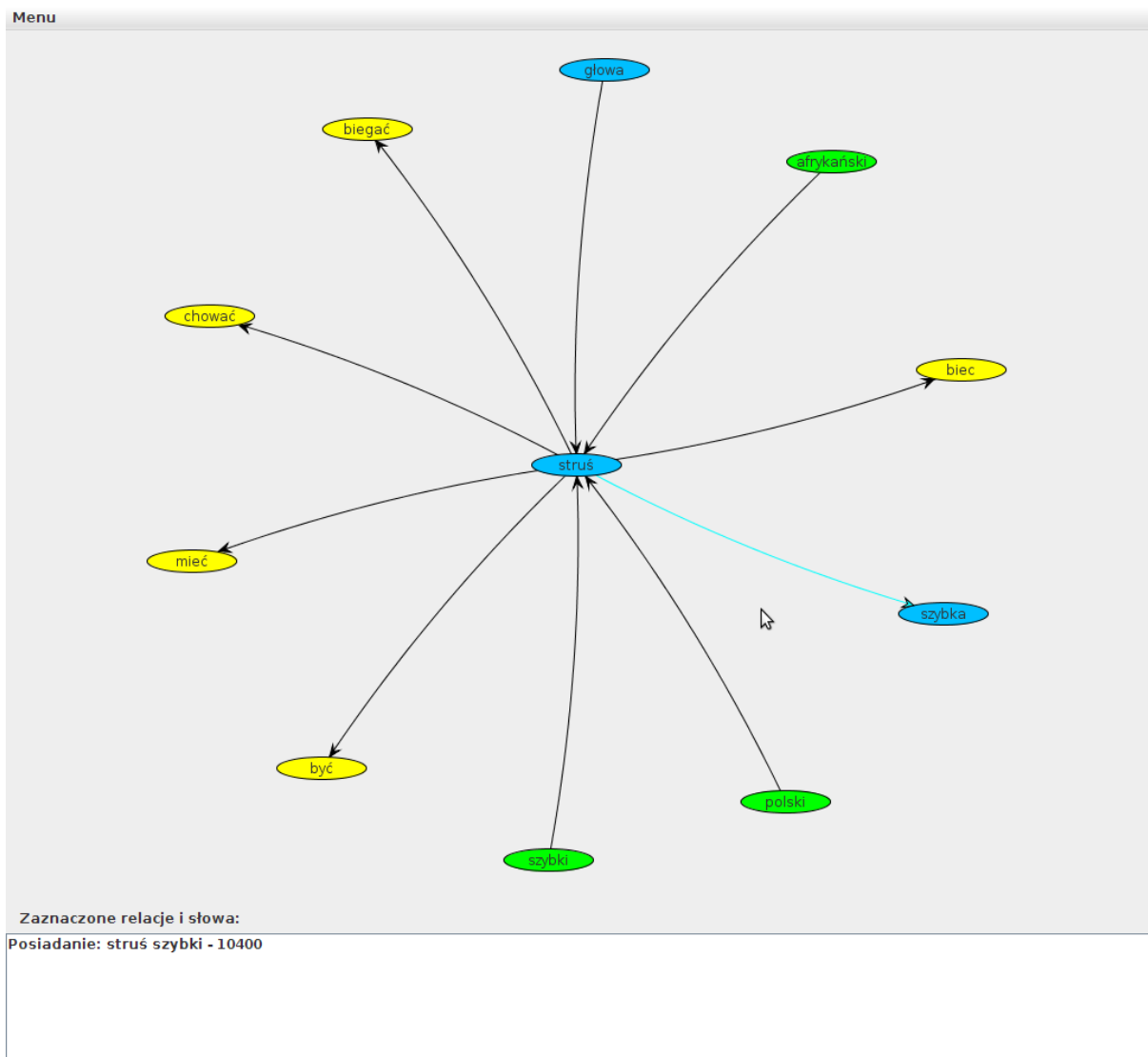
Na pełnej liście utworzonych relacji zaobserwować można relacje, które nie zostały zaprezentowane na grafie ze względu na zbyt niską wartość krawędzi, takie jak *używany gepard* (GEpard to nazwa dealera samochodów), czy *gepard zostanie*. Lista ta widoczna jest na zrzucie ekranu 5.3.

Typ relacji	Wyrażenie	Ocena ▼
Zdolność	gepard jest	9380
Zdolność	gepard może	1910
Zdolność	gepard ma	1690
Grupowanie	zwierzę to gepard	1260
Grupowanie	gepard jest firmą	1180
Określenie	czarny gepard	923
Grupowanie	gepard jest zwierzęciem	726
Posiadanie	gepard firmy	335
Posiadanie	gepard zdjęcia	163
Posiadanie	zdjęcie geparda	140
Określenie	inny gepard	64
Grupowanie	gepard to zwierzę	30
Zdolność	gepard zostanie	27
Określenie	używany gepard	17
Posiadanie	gepard ma zdjęcie	0
Posiadanie	gepard ma zwierzę	0
Grupowanie	człowiek to gepard	0
Określenie	czarny jak gepard	0
Posiadanie	gepard człowieka	0
Określenie	wszystek jak gepard	0
Posiadanie	człowiek geparda	0
Określenie	wielki gepard	0
Grupowanie	gepard jest kotem	0
Posiadanie	firma geparda	0
Grupowanie	zwierzę jest gepardem	0
Grupowanie	kot jest gepardem	0
Posiadanie	kot geparda	0
Grupowanie	firma jest gepardem	0

OK

Rysunek 5.3: Przykładowa lista relacji wygenerowanych dla słowa *gepard*.

Dla wyrazu *struś* baza wiedzy zbudowana została na podstawie 300 wyników z wyszukiwarki Google, co nieco poprawiło jakość utworzonych relacji. Graf LHG został zaprezentowany na zrzucie ekranu 5.4. Znow pojawiły się na nim czasowniki *być* i *mieć*, ale wyświetlone zostały także czasowniki bardziej charakterystyczne dla strusia, jak *chować*, czy *biegać*. Warto zauważyć, że biblioteka CLP wyraźnie rozróżnia słowa *biegać* i *biec*. Wyświetlone przymiotniki, jakie charakteryzują strusia to *afrykański*, *szybki* i *polski*. Brak rozróżnienia części mowy w momencie analizy zdania i konieczność wykorzystania wszystkich dopasowań zwróconych przez bibliotekę CLP skutkuje tym, że jako poprawna relacja uznane zostało wyrażenie *struś szybki*, gdzie *szybki* jest dopełniaczem wyrazu *szybka*. Google zwróciło wiele wyników, pomimo iż wydawałoby się, że kolejność wyrazów nie jest poprawna z gramatycznego punktu widzenia. Okazuje się jednak, że gra *Struś Szybki Lopez* jest na tyle popularna w Internecie, że algorytm sprawdzania poprawności uznaje tę relację za poprawną.



Rysunek 5.4: Graf relacji dla słowa *struś*.

Interesujące informacje można także odczytać z pełnej listy relacji zaprezentowanej na zrzucie ekranu 5.5. Mimo, iż relacja mówiąca, że struś jest ptakiem zwróciła kilkaset wyników, to jednak nie została zaprezentowana na grafie LHG ze względu na limit 10 krawędzi dla jednego wierzchołka. Przewidzenie, czy określony limit ograniczy wyświetlenie wartościowych informacji, gdy będzie zbyt mały, czy pozwoli na wyświetlenie niepotrzebnych relacji, zaciemniających graf, gdy będzie zbyt duży, jest trudne do przewidzenia. Na liście widoczne są także z pozoru niezrozumiałe relacje takie jak *kojot strusia*, czy *pędziwiatr to struś*. Nawiązują one jednak do popularnej bajki „Struś Pędziwiatr” studia Warner Bros.

Typ relacji	Wyrażenie	Ocena ▼
Zdolność	struś jest	12400
Posiadanie	struś szybki	10400
Zdolność	struś chowa	8540
Zdolność	struś ma	3440
Określenie	szybki jak struś	2660
Posiadanie	głowa strusia	813
Zdolność	struś biega	689
Określenie	szybki struś	577
Grupowanie	struś to ptak	388
Określenie	polski struś	274
Określenie	afrykański struś	242
Zdolność	struś biegnie	218
Określenie	strusi struś	186
Posiadanie	struś głowy	162
Grupowanie	struś jest ptakiem	83
Posiadanie	kojot strusia	45
Grupowanie	pędziwiatr to struś	26
Grupowanie	ptak to struś	14
Posiadanie	pędziwiatr strusia	0
Posiadanie	struś kojota	0
Określenie	strusi jak struś	0
Grupowanie	głowa to struś	0
Posiadanie	struś pędziwiatra	0
Grupowanie	pędziwiatr jest strusiem	0
Określenie	struś jest afrykański	0
Określenie	wszystek struś	0
Określenie	struś jest szybki	0
Określenie	struś jest strusi	0
Grupowanie	kojot to struś	0
Grupowanie	ptak jest strusiem	0
Grupowanie	głowa jest strusiem	0
Posiadanie	głowa ma strusia	0
Posiadanie	struś ma kojota	0
Grupowanie	szybka jest strusiem	0
Posiadanie	ptak ma strusia	0
Posiadanie	struś ma pędziwiatra	0
Określenie	polski jak struś	0
Określenie	struś jest polski	0
Grupowanie	szybka to struś	0
Określenie	wszystek jak struś	0
Grupowanie	kojot jest strusiem	0

OK

Rysunek 5.5: Przykładowa lista relacji wygenerowanych dla słowa *struś*.

Aby wygenerować więcej relacji przechodnich na podstawie relacji już istniejących wystarczy wczytać te relacje, czy to z plików, czy bazy danych. Nie jest konieczne wykorzystywanie pająka internetowego, więc ilość linków do pobrania można ustawić na 0. Jeżeli z bazy danych lub z pliku nie zostaną pobrane zdania, na podstawie których powstać mogą nowe relacje, aplikacja tworzyć będzie wyłącznie relacje na podstawie analizowania istniejącej hierarchii i wyciągania pseudo-logicznych wniosków. Przykładowa konfiguracja takich danych wejściowych zaprezentowana została na zrzucie ekranu 5.6.

Plik Pomoc

Generacja słownika

Online Offline Symuluj Pobierz zdania z bazy danych Pobierz relacje z bazy danych

Wyniki:

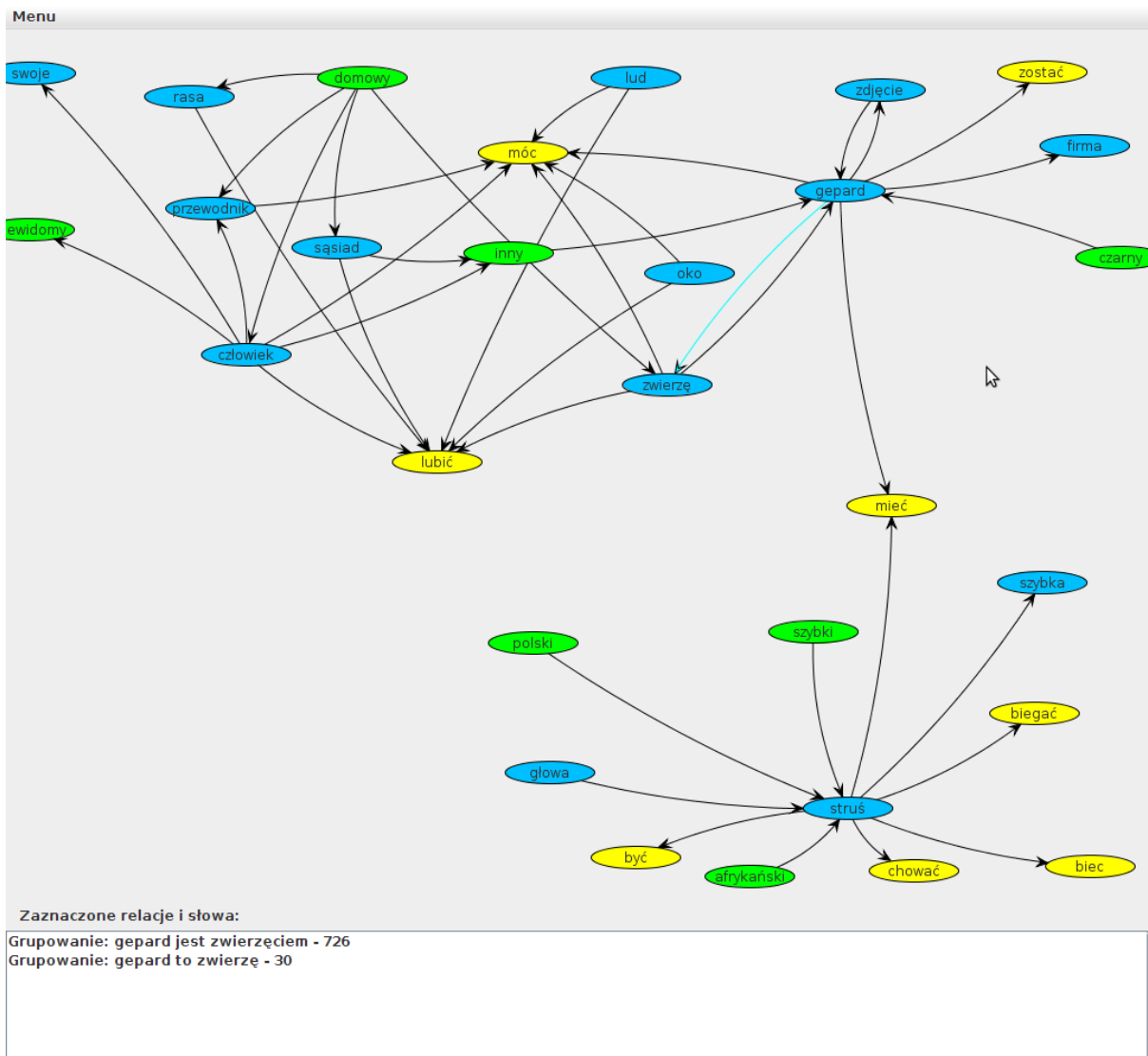
 Zapisz zdania do bazy danych Zapisz relacje do bazy danych

Ilość wyników z google wykorzystywanych w tworzeniu słownika:
Maksymalna ilość powiązań dla każdego wyrazu drzewa:
Maksymalna ilość relacji spropagowanych:

Słowo startowe (korzeń drzewa):

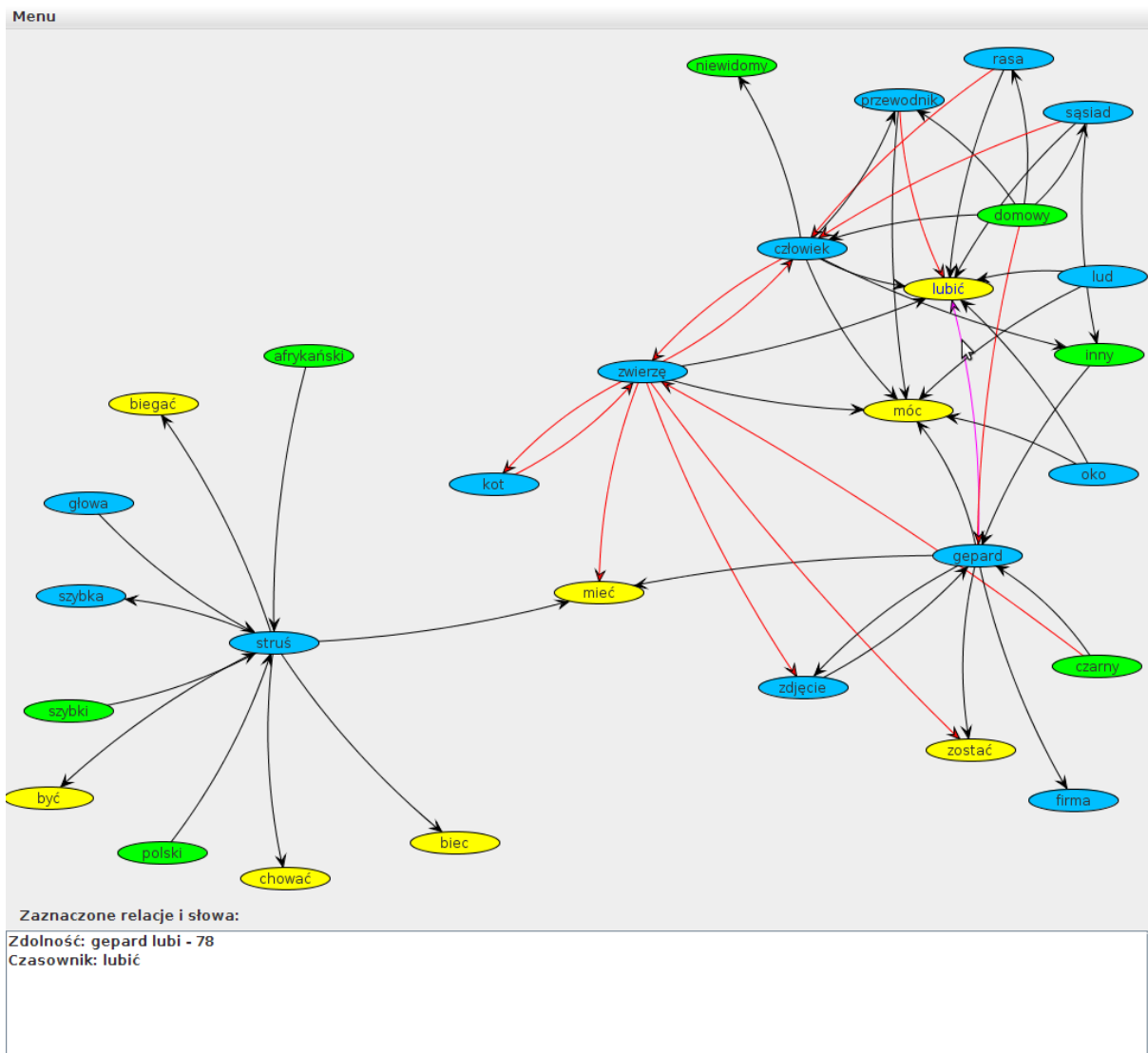
Rysunek 5.6: Przykładowe dane wejściowe dla generacji dodatkowych relacji przechodnich.

Po pobraniu relacji z bazy danych lub plików, graf przyzwyczajęń lingwistycznych rozrasta się, a słowa, które wcześniej charakteryzowały pojedyncze wyrazy łączą się z kilkoma innymi tworząc bardziej rozbudowaną strukturę. Nadal obowiązują jednak reguły dotyczące maksymalnej ilości krawędzi dla jednego wierzchołka, które ustalone zostały w momencie definiowania danych wejściowych. Taki bardziej rozbudowany graf zawierający zarówno relacje z dwóch poprzednich przykładów, jak i kilka innych pokazany został na zrzucie ekranu 5.7.



Rysunek 5.7: Przykładowy graf relacji dla kilku słów.

Większa struktura tego typu pozwala na podejmowanie przez aplikację prób utworzenia relacji na podstawie analizy hierarchii. Na zrzucie ekranu 5.8 widać owoce tego procesu. Wyświetlone zostały nowe krawędzie, które powstały na podstawie nowych propozycji relacji. Jak widać na zrzucie ekranu 5.2 poprawna jest relacja *gepard jest zwierzęciem*. Choć nie została ona wyświetlona na bardziej rozbudowanym grafie, to jednak została uwzględniona w procesie wnioskowania. Jako, że poprawna jest także relacja *zwierzę lubi*, aplikacja uznała łącząc te dwa fakty, że rzeczownik gepard i czasownik lubić także mogą być połączone relacją. Weryfikacja przeprowadzona na podstawie ilości wyników zwróconych przez wyszukiwarke Google potwierdziła poprawność takiego połączenia, co zaowocowało powstaniem nowej czerwonej krawędzi na grafie, obrazującej spropagowaną relację. Dokładna lista wszystkich relacji została zaprezentowana na zrzucie ekranu 5.9.



Rysunek 5.8: Przykładowy graf relacji wraz z relacjami przechodnimi dla kilku słów.

Typ relacji	Wyrażenie	Ocena ▼
Określenie	człowiek jest inny	962000
Zdolność	człowiek może	447000
Zdolność	oko może	128000
Zdolność	zwierzę ma	118000
Zdolność	człowiek lubi	66200
Grupowanie	człowiek to zwierzę	54700
Grupowanie	kot jest zwierzęciem	29200
Zdolność	przewodnik może	27100
Określenie	człowiek jest niewidomy	26200
Zdolność	zwierzę może	23300
Grupowanie	kot to zwierzę	22800
Grupowanie	człowiek jest przewodnikiem	19000
Zdolność	lud może	18400
Zdolność	struś jest	12400
Posiadanie	struś szybki	10400
Zdolność	struś chowa	8540
Określenie	sąsiad jest inny	7580
Grupowanie	zwierzę jest człowiekiem	7410
Zdolność	lud lubi	6680
Zdolność	zwierzę zostanie	6550
Grupowanie	sąsiad jest człowiekiem	6460
Grupowanie	zwierzę to kot	5220
Zdolność	struś ma	3440
Zdolność	oko lubi	2710
Określenie	szybki jak struś	2660
Zdolność	zwierzę lubi	2520
Określenie	domowy przewodnik	2420
Grupowanie	zwierzę jest kotem	2320
Zdolność	gepard może	1910
Zdolność	sąsiad lubi	1910
Grupowanie	zwierzę to człowiek	1840
Zdolność	gepard ma	1690
Grupowanie	zwierzę to gepard	1260
Grupowanie	rasa to człowiek	1130
Posiadanie	zwierzę człowieka	1120
Określenie	czarny gepard	923
Posiadanie	głowa strusia	813
Grupowanie	gepard jest zwierzęciem	726
Zdolność	struś biega	689
Określenie	czarny zwierzę	643
Zdolność	rasa lubi	643
Grupowanie	sąsiad to człowiek	590
Określenie	szybki struś	577
Posiadanie	zwierzę zdjęcia	507
Grupowanie	struś to ptak	388
Posiadanie	gepard firmy	335
Określenie	domowy zwierzę	295
Określenie	inny zwierzę	288
Określenie	polski struś	274
Określenie	afrykański struś	242
Zdolność	przewodnik lubi	220

Rysunek 5.9: Przykładowa lista relacji wygenerowanych dla kilku różnych słów.

W przykładach zaprezentowanych powyżej starałem się pokazać sposób działania aplikacji i uzyskiwane przez nią wyniki. Na efekt działania programu silnie wpływają trendy panujące aktualnie w Internecie, a także reklamy rozmaitych produktów, których nazwy często łączą słowa w nieoczekiwany sposób. Homonimy rozróżniane są poprzez wykorzystanie wszystkich części mowy zwróconych przez bibliotekę CLP i przekazanie ich jako zapytanie do wyszukiwarki w odpowiedniej formie. Dwa identycznie zapisane wyrazy w formie bazowej, będące jednak różnymi częściami mowy, rozróżnione zostaną na podstawie reguł akomodacyjnych.

6. Podsumowanie wyników pracy i wnioski

6.1. Podsumowanie

Współczesna technologia wciąż jest daleka od skonstruowania inteligentnego robota, który byłby w stanie idealnie komunikować się z ludźmi. Taka maszyna musi znać nie tylko gramatykę danego języka, ale także rozumieć znaczenie wypowiadanych słów na poziomie wyższym niż prosta poprawność językowa. Problem badania znaczenia słów, aby następnie poprawnie użyć ich podczas konwersacji nie może sprowadzać się wyłącznie do analizowania ich kontekstu w pojedynczym zdaniu lub wypowiedzi. Mimo, iż rozszyfrowanie formy frazeologicznej słowa nie jest zadaniem skomplikowanym, to sformułowanie nie tylko poprawnej gramatycznie, ale i logicznie wypowiedzi zależy już od większej liczby czynników. Podobnie jak człowiek, który po raz pierwszy usłyszał dane słowo, jest w stanie wywnioskować jego znaczenie z kontekstu zdania i następnie używać go poprawnie, tak i możliwe jest to dla czatbota. Niezbędnym jednak elementem, który warunkuje tę umiejętność jest posiadanie odpowiedniej wiedzy, na podstawie której stosując logiczne rozumowanie możliwe staje się uzyskanie nowych, poprawnych informacji. Stworzona przeze mnie aplikacja jest propozycją rozwiązania tego problemu. Dostarcza ona danych, które wykorzystane przez czatbota pozwolą mu zarówno na znacznie lepsze rozumienie analizowanych przez niego wypowiedzi, jak i formułowanie własnych logicznie poprawnych wyrażeń.

Największym atutem mojego rozwiązania jest fakt, że nie ogranicza się ono do informacji zawartych wyłącznie w korpusach tekstów. Jakkolwiek są one i długo pozostaną nieocenionym źródłem danych dla czatbotów, to ilość słów zawarta nawet w największych tego typu repozytoriach jest nieporównanie mniejsza niż mnogość form wyrazowych w Internecie. Do osiągnięcia zamierzonego celu niezbędna była baza wiedzy, która nie ogranicza się wyłącznie do pojedynczych wyrazów, ale także ich możliwych połączeń. Podczas, gdy korpusy tekstów liczą dziesiątki lub setki tysięcy wystąpień wielu popularnych wyrazów, zasoby oferowane przez Google słowa takie liczą w dziesiątkach, czy nawet setkach milionów.

Dodatkowym argumentem przemawiającym na korzyść korzystania bezpośrednio z wiedzy zawartej w Internecie jest fakt ciągłej zmienności umieszczonych tam tekstów. Język ewoluuje, a zabytki piśmiennictwa, które często są integralną częścią korpusów nie zawierają wielu współcześnie używanych form i wyrażeń. Z drugiej strony zawierają formy, które rzadko pojawiają się w mowie potocznej dzisiejszych czasów. Ale nawet znalezienie rzadkich, czy archaicznych sformułowań jest łatwiejsze, gdy ma się do dyspozycji całą wiedzę zgromadzoną w Internecie.

Spore niebezpieczeństwo dla poprawności językowej zgromadzonych danych stanowi sposób ich zapisu na stronach internetowych. W szczególnych przypadkach niepożądane znaki, lub fragmenty kodu mogą znaleźć się w tworzonej bazie wiedzy. Był to główny powód, dla którego zdecydowałem się dość restrykcyjnie podejść do filtrowania niepożądanych informacji. Nie wszystkie zagrożenia dla poprawności przechowywanych danych mają podłoże opierające się o nietypowe znaki lub ich kombinacje. Bardzo wiele portali internetowych zawiera treści niepozbawione błędów. Nie chodzi tylko o literówki w artykułach, czy esejach. Prawdziwą plagą stają się wpisy na rozmaitych forach internetowych i blogach, a także komentarze do wiadomości na portalach informacyjnych. Zarówno poziom ortografii, jak i stylistyki, a nawet logiki wielu wypowiedzi pozostawia wiele do życzenia. Wśród wpisów tego typu częstym zjawiskiem są boty reklamujące, które na wielu forach automatycznie tworzą posty składające się wyłącznie z pisanych jednym ciągiem haseł. Mając do dyspozycji wiele źródeł informacji pojedyncze

przypadki tego typu byłyby z pewnością niezauważalne. Proceder ten jest jednak powszechny i automatycznie propagowany przy pomocy złośliwych pajaków internetowych na bardzo wiele stron. Ilość tego typu zdegenerowanych danych może mieć wpływ na proces automatycznego decydowania o poprawności wypowiedzi.

Podstawowa wersja pajaka internetowego, którą zdecydowałem się zaimplementować spełnia swoje zadanie. Wyszukiwarka Google dostarcza wyniki dość dobrej jakości, co bezpośrednio przekłada się na jakość stron odwiedzanych przez robota. Nawet jeżeli pomijane są czasem zasoby, które potencjalnie mogłyby zostać wykorzystane do budowania bazy wiedzy, to najważniejszym aspektem działania pajaka jest zebranie reprezentatywnych źródeł. Źródła te mają posłużyć zgromadzeniu informacji o słowach, które jak najczęściej pojawiają się w kontekście wyrazu, który w danej chwili jest obiektem zainteresowania programu.

Ekstrakcja danych i tworzenie relacji przebiega bardzo sprawnie. Wąskim gardłem aplikacji jest zdecydowanie sprawdzanie poprawności relacji. Jako, że Google nie udostępnia API pozwalającego na dostęp do wyników wyszukiwania, konieczne jest pozyskiwanie ich bezpośrednio parsując otrzymaną stronę wynikową. Częste automatyczne zapytania są jednak bardzo skutecznie rozpoznawane przez algorytmy wyszukiwarki, co skutkuje blokadą możliwości korzystania z niej na kilka godzin, lub konieczności uzupełnienia formularza zwanego CAPTCHA (ang. Completely Automated Public Turing test to tell Computers and Humans Apart), którego zadaniem jest weryfikacja, czy użytkownik jest człowiekiem.

Zaproponowana przeze mnie metoda oceny poprawności relacji nie jest niezawodna. Jej główna niedoskonałość wiąże się z algorytmem wyszukiwającym Google, który możliwie jak najbardziej poszerza zadane przez użytkownika kryteria, aby zwrócić jak największą liczbę wyników. Ignorowane są zatem wszelkie znaki interpunkcyjne (zarówno w wyszukiwanej frazie, jak i potencjalnych wynikach), które mogą mieć znaczący wpływ na jakość rezultatu. Prosty przykładem jest sprawdzanie poprawności wyrażenia *kot to samochód*. Grupowanie takie wydaje się niedorzeczne, ale Google znajdzie to wyrażenie w tekście zawierającym kolejno zdania: *Obok przebiegł przerażony kot. To samochód przejeżdżający obok tak go przestraszył*. Fakt, że między *kot*, a *to samochód* jest kropka zostanie całkowicie zignorowany.

Problem stanowią także tytuły książek, filmów lub artykułów, które często celowo sformułowane są w sposób syntaktycznie lub semantycznie niepoprawny. Można by pomyśleć, że nie ma w tym nic złego. O rozmaitych komentarzach pod artykułami na stronach informacyjnych lub wpisach na forach, czy blogach można powiedzieć to samo. Niestety tytuły dzieł publicznie dostępnych są nieporównanie szerzej rozpowszechnione, co sprawia, że Google podczas wyszukiwania natrafi na nie o wiele częściej przeszukując rozmaite strony zawierające reklamy, zapowiedzi, czy recenzje.

6.2. Dalsze możliwości rozwoju

W aplikacji istnieje wciąż wiele obszarów, które można poprawić, aby polepszyć zarówno jakość otrzymywanych wyników, jak i wydajność działania. Główne elementy, które można rozwinąć w przyszłości to:

1. Algorytm sprawdzania poprawności relacji.

Ograniczenia narzucone przez Google są największym mankamentem zaproponowanego przeze mnie rozwiązania. Konieczność oczekiwania pomiędzy zapytaniami, a także możliwość blokady zapytań na kilka godzin znacznie obniża wydajność aplikacji. Niestety inne wypróbowywane przeze mnie wyszukiwarki internetowe zwracają nieporównanie mniej wyników, co w przypadku sprawdzania poprawności metodą statystyczną ma spore znaczenie. Być może w przyszłości znów dostępne będzie API, które pozwoli na zautomatyzowane korzystanie z wyszukiwarki Google, co z pewnością znacznie poprawi wydajność programu. Akceptowalnym rozwiązaniem byłyby także znacznie bardziej zaawansowany mechanizm wysyłający zapytania wykorzystujący częste zmiany przeglądark podawanych w nagłówku wysyłanego zapytania, a także korzystający z serwerów proxy.

2. Szablony relacji

Jeżeli problem skuteczniejszego mechanizmu sprawdzania poprawności relacji zostałby rozwiązany, to nic nie stałoby na przeszkodzie, aby wprowadzić więcej rodzajów relacji, które znacznie wzbogaciłyby wartość lingwistyczną tworzonej bazy wiedzy. Kolejnym krokiem w rozwoju tego elementu systemu mogłoby być wyodrębnienie spójników jako oddzielnych części mowy, a nie jak obecnie traktowanie ich jako „sztywnych” elementów zespalających połączenie, między bardziej znaczącymi częściami mowy. Na podstawie analizy składniowej tekstu możliwe byłoby dodanie automatycznego tworzenia szablonów relacji na podstawie często spotykanych struktur gramatycznych.

3. Słownik fleksyjny

Kolejną możliwością ulepszenia aplikacji jest wzbogacenie słownika fleksyjnego o nowe formy wyrazowe, a także bardziej przejrzysty sposób określania i definiowania odmiany słów. Możliwość dokładnego określenia, odmiana której cechy wyrazu nas interesuje byłaby znacznie praktyczniejsza niż wybór po indeksie w tablicy. Przykładowo mając wyraz *kot* w formie bazowej, chcąc uzyskać dopełniacz liczby mnogiej musimy odwołać się do określonego indeksu, który odpowiada żądanej formie. O wiele wygodniejsza byłaby możliwość podania parametrów formy, które nas interesują np. `zwrocForme("kot", DOPELNIACZ, L_MN)`.

6.3. Porównanie z rozwiązaniami o podobnej tematyce

Przykładowa, znaleziona w Internecie implementacja czatbota[17] napisana przez Tomasza Jędrzejewskiego wykorzystuje prosty algorytm tworzenia wypowiedzi. Na podstawie rozmowy z użytkownikiem, zapamiętuje pojawiające się zdania w postaci skierowanego grafu przedstawionego na rysunku 6.1. Graf ten łączy wierzchołki odpowiadające użytym wyrazom skierowanymi krawędziami, które odpowiadają połączeniom między nimi. Wierzchołki \$ i # oznaczają odpowiednio początek i koniec zdania. Generowanie wypowiedzi polega na wyborze losowej ścieżki pomiędzy elementami grafu od wierzchołka \$ do #.



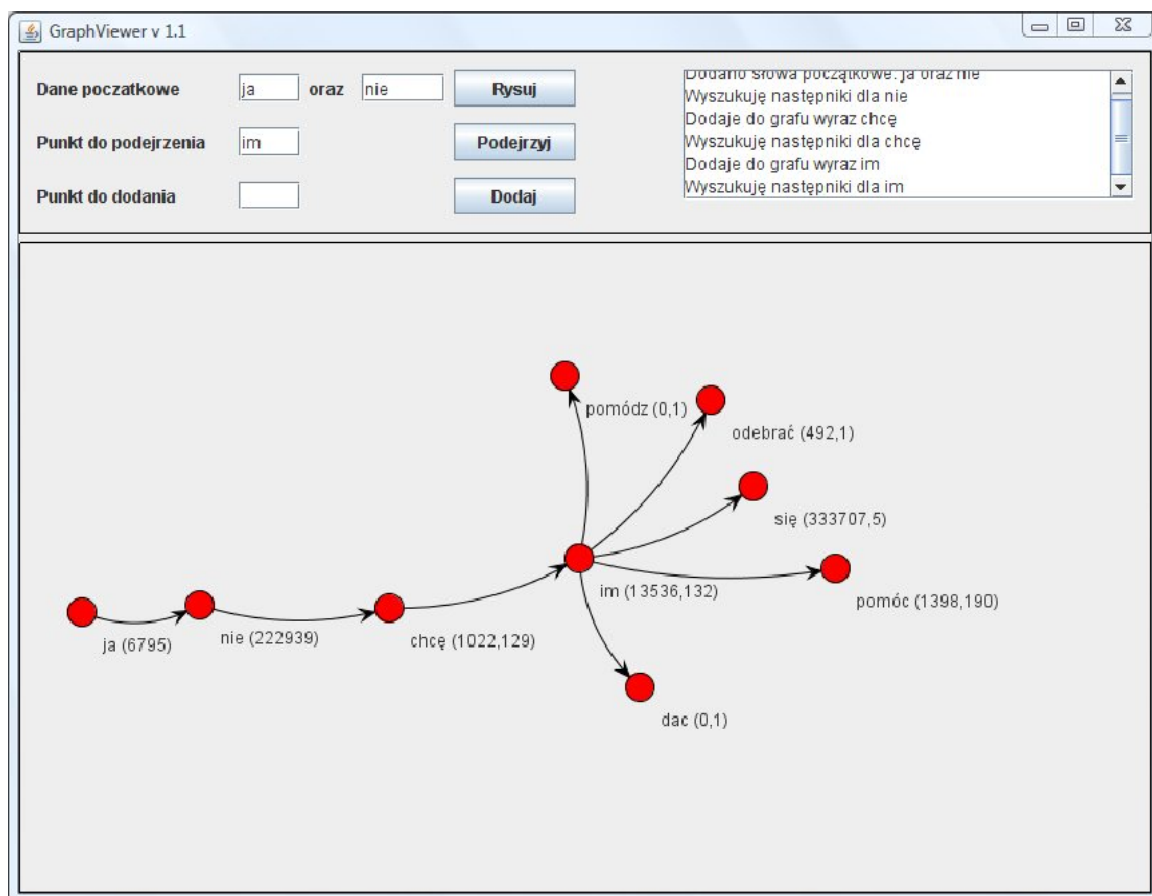
Rysunek 6.1: Graf obrazujący połączenia między wyrazami dla czatbota zaimplementowanego przez Tomasza Jędrzejewskiego.

Algorytm tego typu jest bardzo prosty, a generowane przez czatbota wypowiedzi bardzo często są niepoprawne nie tylko semantycznie, ale również składniowo, gdyż nie wykorzystywany jest żaden słownik morfologiczny. Baza wiedzy, na podstawie której tworzone są zdania jest ograniczona wyłącznie do danych zebranych podczas rozmowy z użytkownikiem. Zebranie dużej ilości danych z wielu konwer-

sacji nie poprawi jakości działania programu, gdyż losowy wybór ścieżki w grafie może powodować zapętlenia, lub powstawanie długich pozbawionych sensu zdań.

Rozwiązanie, które zaproponowałem korzysta ze zdecydowanie bogatszej bazy wiedzy, co pozwala na generowanie wypowiedzi z wykorzystaniem słów, które nie pojawiły się w trakcie rozmowy. Połączenia między wyrazami są poprawne syntaktycznie, a także odfiltrowywane są te, uznane za nie poprawne z punktu widzenia semantyki. O poprawności decyduje w głównej mierze popularność fraz w Internecie, zatem błędy choć są nieuniknione, pojawiać się będą znacznie rzadziej. Baza wiedzy budowana jest poprzez wyszukiwanie informacji związanych z konkretnym tematem. Bezpośrednio przekłada się to na jakość otrzymywanych wyników, które są tym dokładniejsze, im większa jest baza wiedzy, na podstawie której zostały wygenerowane.

Marcin Gadamer w swojej pracy[8] wykorzystał algorytm, który buduje graf przyzwyczajęń lingwistycznych na podstawie korpusów tekstów. Generowanie wypowiedzi odbywa się w interakcji z użytkownikiem, który wskazuje kolejne słowo z listy zaproponowanych możliwości. Przykładowy fragment grafu zaprezentowany został na rzucie ekranu 6.2. W bazie danych przechowywane są trójki wyrazów, z zachowaniem kolejności, w jakiej występowały w analizowanych zdaniach. Na podstawie dwóch pierwszych form wyrazowych z trójki prezentowane są możliwe poprawne formy kolejnej, wraz z określeniem częstości wystąpień.



Rysunek 6.2: Fragment grafu LHG z aplikacji Marcina Gadamera prezentujący cztery słowa wypowiedzi i proponowane następniki.

Algorytm ten dobrze sprawdza się podczas automatycznej korekcji tekstu, jednak słabiej spisująby się w przypadku automatycznej generacji zdań. Wrażliwość na zapętlenia, a także powstawanie zbyt długich wypowiedzi zostały rozwiązane przez interakcję z użytkownikiem. Problemem jest także ilość

danych przechowywanych w bazie wiedzy. Każde przeanalizowane zdanie rozkładane jest na $n-2$ trójek, gdzie n jest ilością wyrazów w zdaniu.

Generowana przez moją aplikację uproszczona postać grafu LHG skupia się w większym stopniu na relacjach między konkretnymi parami wyrazów. Każda informacja odnośnie takiego połączenia jest istotna i posiada wartość dla czatbota. Nawet jeżeli do bazy danych zapisywane są niepoprawne relacje, to jest to także ważny element wiedzy programu. Na podstawie zebranych danych nie da się co prawda utworzyć pełnych zdań w języku polskim, ale można je wykorzystać podczas procesu „rozumienia” sensu wypowiedzi użytkownika, a także generowania odpowiedzi zgodnej z tematem rozmowy.

7. Zakończenie

Cel jaki został postawiony w tej pracy to utworzenie uproszczonej wersji grafu przyzwyczajzeń lingwistycznych (LHG), który obrazuje relacje zachodzące między wyrazami w języku polskim, w szczególności modelując grupowanie wokół określonych cech i budowę hierarchicznej struktury w obrębie danej grupy. Osiągnięcie tego celu sprowadzone zostało do utworzenia trzech głównych modułów aplikacji oraz interfejsu użytkownika.

1. Pająk internetowy

Aby wykorzystać wiedzę zgromadzoną w Internecie niezbędne było wykorzystanie pająka internetowego, który analizując treść odwiedzanych stron pozyskuje informacje, które następnie wykorzystywane są do utworzenia bazy wiedzy. Utworzony przeze mnie robot wykorzystując wyniki z wyszukiwarki Google pobiera poprawne zdania w języku polskim i zapisuje je w bazie danych stale powiększając dostępne do wykorzystania repozytorium wiedzy. Jego zadaniem jest także rozkład zdań na pojedyncze słowa, które następnie wykorzystywane są w mechanizmie tworzenia relacji. Oprócz pająka dodałem także możliwość wykorzystania korpusów tekstów znajdujących się w plikach tekstowych, aby Internet nie był jedynym źródłem wiedzy, z którego aplikacja może korzystać.

2. Kreator relacji

Zasadniczy mechanizm działania aplikacji miał za zadanie z wyników dostarczonych przez pająka utworzyć strukturę, która modeluje relacje między słowami w języku polskim. Zastosowany przeze mnie algorytm zlicza ilość wystąpień wyrazów w kontekście słowa, wokół którego tworzony jest graf relacji. Najczęściej występujące są następnie podstawiane do pre-definiowanych szablonów, które zawierają oba elementy relacji w określonych formach fleksyjnych tworzących poprawne syntaktycznie wyrażenie. Poprawność takiego wyrażenia jest następnie sprawdzana za pomocą wyszukiwarki Google. Ilość zwróconych wyników określa „popularność” frazy w Internecie, co przekłada się na jej poprawność zarówno syntaktyczną, jak i semantyczną.

Ważną funkcją kreatora relacji jest także proces pseudo-logicznego rozumowania. Wykorzystując znane już poprawne relacje i ich hierarchiczną strukturę sprawdzane są potencjalne połączenia między słowami. Elementy należące do wspólnej grupy zestawiane są z cechami składników tejże grupy. Podobnie wyrazy, które posiadają wspólne cechy sprawdzane są pod kątem przynależności do wspólnej grupy.

3. Specjalistyczny graf przyzwyczajzeń lingwistycznych

Zadaniem uproszczonego grafu LHG było zilustrowanie modelu relacji w sposób czytelny dla użytkownika. Zostały na nim pokazane różne rodzaje połączeń między wyrazami oraz wartość liczbowa określająca ich poprawność. Relacje zapisane zostały jako wyrażenia składające się z dwóch wyrazów w odpowiedniej formie połączonych ewentualnym spójnikiem i zaprezentowane na skierowanych krawędziach grafu. Na żądanie użytkownika wyświetlone mogą zostać także relacje uzyskane z procesu tak zwanej „przechodniości”, czyli wykonanego wcześniej pseudo-logicznego procesu rozumowania i analizy istniejących połączeń.

Popularność tematyki sztucznej inteligencji jest obecnie bardzo duża, a zainteresowanie urządzeniami i aplikacjami zdolnymi porozumiewać się z człowiekiem jest szczególnie wysokie. Roboty, które komunikują się z użytkownikiem i na bieżąco uczą się poszerzając swoją bazę wiedzy przestają być wytworem filmów science fiction, a zaczynają pojawiać się już jako gotowe produkty. Wciąż jeszcze charakteryzowane są raczej jako nowinki techniczne, lub modne gadżety, ale wzrastające zapotrzebowanie może sprawić, że już niedługo napotykać je będziemy w codziennym życiu.

Zaproponowane przeze mnie rozwiązanie jest zaledwie małym krokiem na drodze do stworzenia inteligentnego czatbota. Daje ono jednak w miarę zadowalające rezultaty, które mogą zostać wykorzystane przy tworzeniu nie tylko poprawnych gramatycznie, ale i logicznie zdań. Wizja robota, który nie tylko formułuje poprawne zdania, ale także rozumie kontekst poszczególnych słów, jakich używa, jak i tych, które otrzymuje od rozmówcy, staje się coraz bardziej realna. Pomimo wielu problemów, jakie wciąż stoją przed programistami, powstanie czatbota, który pozytywnie przejdzie test Turinga jest tylko kwestią czasu.

Spis rysunków

2.1	Schemat działania pająka internetowego	11
2.2	Graficzny interfejs metawyszukiwarki WebCrawler.	12
2.3	Graficzny interfejs aplikacji Web Fountain.	13
2.4	Wynik działania programu WGet.	14
2.5	Zrzut ekranu obrazujący działanie aplikacji Online Data Extractor.	15
2.6	Zrzut ekranu obrazujący działanie aplikacji HTML Text Extractor.	15
2.7	Kolokacje wyrazu niebo na podstawie Narodowego Korpusu Języka Polskiego.	16
3.1	Przykład oddziaływania akomodacyjnego dla leksemów <i>kot</i> i <i>iść</i>	19
3.2	Schemat uogólnienia oddziaływań akomodacyjnych między podmiotem i orzeczeniem.	19
3.3	Schemat uogólnienia oddziaływań akomodacyjnych między czasownikiem i rzeczownikiem.	19
3.4	Schemat uogólnienia oddziaływań akomodacyjnych między przymiotnikiem i rzeczownikiem.	19
3.5	Przykład akomodacji typu frazy zdaniowej.	19
4.1	Diagram klas	22
4.2	Okno połączenia z bazą danych	23
4.3	Główne okno programu	23
4.4	Wybór plików zawierających relacje lub korpusy tekstów.	24
4.5	Schemat działania ekstraktora danych	26
4.6	Szczegóły krawędzi - grupowanie, posiadanie	28
4.7	Szczegóły krawędzi - określenie	28
4.8	Lista relacji	29
4.9	Wybór trybu online	29
4.10	Wybór trybu offline	32
4.11	Wybór trybu pracy z symulatorem	33
4.12	Przykładowy graf z wierzchołkami będącymi różnymi częściami mowy.	33
4.13	Informacja o konkretnym wierzchołku wyświetlona na grafie.	34
4.14	Widok pokazujący spropagowane relacje na grafie.	35
5.1	Przykładowe dane wejściowe dla programu.	36
5.2	Graf relacji dla słowa <i>gepard</i>	37
5.3	Przykładowa lista relacji wygenerowanych dla słowa <i>gepard</i>	38
5.4	Graf relacji dla słowa <i>struś</i>	39
5.5	Przykładowa lista relacji wygenerowanych dla słowa <i>struś</i>	40
5.6	Przykładowe dane wejściowe dla generacji dodatkowych relacji przechodnich.	41
5.7	Przykładowy graf relacji dla kilku słów.	42

5.8	Przykładowy graf relacji wraz z relacjami przechodnimi dla kilku słów.	43
5.9	Przykładowa lista relacji wygenerowanych dla kilku różnych słów.	44
6.1	Graf obrazujący połączenia między wyrazami dla czatbota zaimplementowanego przez Tomasza Jędrzejewskiego.	47
6.2	Fragment grafu LHG z aplikacji Marcina Gadamera prezentujący cztery słowa wypowiedzi i proponowane następniki.	48

Bibliografia

- [1] Geofferey Landis. Out of Sight, Out Of Mind. <http://www.geoffreylandis.com/sight.htm>, 1999.
- [2] GNU. WGet. <http://www.gnu.org/software/wget/>, 2010.
- [3] Google Code. crawler4j. <http://code.google.com/p/crawler4j/>, 2010.
- [4] ICONICO. HTML Text Extractor. <http://www.iconico.com/html extractor/>, 2010. [dostęp: 2010-05-08].
- [5] IPI PAN. Korpus IPI PAN. <http://korpus.pl/>, 2008.
- [6] M. Junczys-Dowmunt. Zastosowanie automatów skończonych stanów w przetwarzaniu języków naturalnych, 2008.
- [7] Z. Klemensiewicz. *Zarys składni polskiej*. Warszawa, 1961.
- [8] M. Gadamer. Automatyczna kontekstowa korekta tekstów na podstawie Grafu Przyzwyczajień Lingwistycznych (LHG) zbudowanego przez robota internetowego dla języka polskiego, Kraków, 2009.
- [9] M. Gadamer, A. Horzyk. Automatyczna kontekstowa korekta tekstów z wykorzystaniem grafu LHG. *Computer Science AGH, Vol. 10, pp. 39-57*, Kraków, 2010.
- [10] Online Data Extractor. Online data extractor tool. <http://www.onlinedataextractor.com>, 2010.
- [11] M. Piasecki. Cele i zadania lingwistyki informatycznej. *Metodologie językoznawstwa. Współczesne tendencje i kontrowersje*, 2007.
- [12] Piotr Pęzik, Narodowy Korpus Języka Polskiego . PELCRA. <http://www.nkjp.uni.lodz.pl/>, 2010. [dostęp: 2010-03-10].
- [13] Portal naukowy.pl. Gramatyka generatywna. http://www.naukowy.pl/encyklopedia/Gramatyka_generatywna, 2010.
- [14] Portal naukowy.pl. Historia lingwistyki. http://www.naukowy.pl/encyklopedia/Historia_lingwistyki, 2010.
- [15] Z. Saloni and M. Świdziński. *Składnia współczesnego języka polskiego*. PWN, 1985.
- [16] The Web Robots Pages. About /robots.txt. <http://www.robotstxt.org/robotstxt.html>, 2010.
- [17] Tomasz Jędrzejewski. Piszemy czatbota. <http://www.eioba.pl/a/1nrd/piszemy-chatbota>, 2007.
- [18] Wapedia. Klasyfikacja części mowy. http://wapedia.mobi/pl/Klasyfikacja_cz%C4%99%C5%9Bci_mowy, 2010.

-
- [19] Wikipedia. Ferdinand de saussure. http://pl.wikipedia.org/w/index.php?title=Ferdinand_de_Saussure&oldid=24025783, 2010.
- [20] Wikipedia. Historia językoznastwa. http://pl.wikipedia.org/w/index.php?title=Historia_j%C4%99zykoznawstwa&oldid=23236068, 2010.
- [21] Wikipedia. IBM WebFountain. http://en.wikipedia.org/w/index.php?title=IBM_WebFountain&oldid=391584874, 2010.
- [22] Wikipedia. Językoznastwo. <http://pl.wikipedia.org/w/index.php?title=J%C4%99zykoznawstwo&oldid=23891961>, 2010.
- [23] Wikipedia. Lingwistyka komputerowa. http://pl.wikipedia.org/w/index.php?title=Lingwistyka_komputerowa&oldid=23235735, 2010.
- [24] Wikipedia. Web crawler. http://en.wikipedia.org/w/index.php?title=Web_crawler&oldid=399429057, 2010.
- [25] Wikipedia. Web crawler. http://en.wikipedia.org/w/index.php?title=Web_crawler&oldid=398058393, 2010.
- [26] Wydawnictwo Naukowe PWN. Korpus Języka Polskiego Wydawnictwa Naukowego PWN. <http://korpus.pwn.pl/>, 2010.
- [27] P. Żmigrodzki. Związki między składnikami w zdaniu polskim. Szkoła języka i kultury polskiej Uniwersytet Śląski, Katowice.