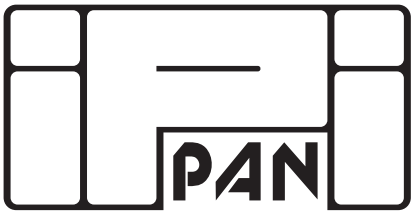
Instytut Podstaw Informatyki
Polskiej Akademii Nauk



Piotr Przybyła

Odpowiadanie na pytania w języku polskim z użyciem głębokiego rozpoznawania nazw

Rozprawa doktorska
napisana pod kierunkiem
dr hab.
Agnieszki Mykowieckiej

Warszawa 2014

Badania zrealizowano dzięki stypendium naukowemu i stażowemu w ramach projektu „Technologie informacyjne: badania i ich interdyscyplinarne zastosowania”, realizowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego, Program Operacyjny Kapitał Ludzki (Umowa nr UDA-POKL.04.01.01-00-051/10-00).

Szczególne podziękowania składam kolegom
i koleżankom z Instytutu, w szczególności
RK, AP, PT, HŁ, OM, JP, AC, BT, RS i MW,
dzięki którym nauczyłem się wielu rzeczy
leżących daleko poza tematyką niniejszej pracy.

Streszczenie

Niniejsza praca dotyczy znanego w dziedzinie przetwarzania języka naturalnego problemu odpowiadania na pytania (ang. *Question Answering*, QA). Zadanie to polega na budowie w pełni automatycznego systemu komputerowego zdolnego do przyjmowania pytań w języku naturalnym (np. polskim) i udzielania odpowiedzi w tym samym języku. Problem ten pełni szczególnie istotną rolę w przypadku systemów komputerowych udostępniających informacje osobom o niskich kwalifikacjach informatycznych (np. w przypadku wyszukiwarek internetowych).

W pracy przedstawiono opis systemu QA dla języka polskiego, nazwanego RAFAEL (ang. *RApid Factoidal Answer Extracting aLgorithm*), zdolnego do interpretacji pytań o proste fakty, odnajdywania informacji w zbiorze dokumentów tekstowych bez ograniczeń dziedzinowych i udzielania zwięzłych odpowiedzi. Pewne własności języka polskiego, tj. fleksyjność i swobodny szyk zdania, sprawiają, że niektóre elementy zadania (np. wstępny wybór dokumentów i dopasowywanie zdań) wymagają specjalnego podejścia i sprawiają trudności nieobecne w przypadku języka angielskiego, dla którego budowana jest większość systemów QA.

W odróżnieniu od innych badań z tej dziedziny, w niniejszej pracy najwięcej uwagi poświęcono problemowi rozpoznawania nazw. Zamiast powszechnie stosowanego podejścia, bazującego na rozpoznawaniu nazw własnych (ang. *Named Entity Recognition*, NER), zaproponowano jego uogólnienie, nazwane głębokim rozpoznawaniem nazw (ang. *Deep Entity Recognition*, DeepER). W tej wersji odnajdywanym w tekstach nazwom, potencjalnie stanowiącym odpowiedzi, nie są przypisywane szerokie kategorie NE, ale szczegółowe synsety z ontologii WordNet. Pozwala to na precyzyjniejsze wybieranie kandydujących wzmianek i zapewnienie pełnej zgodności z ograniczeniami wyrażonymi w pytaniu. Co więcej, poszerza to zakres obsługiwanych pytań o te dotyczące bytów takich jak zwierzęta, urządzenia czy związki chemiczne, leżących poza tradycyjnymi kategoriami NE.

Ewaluacja systemu przeprowadzona została w dwóch krokach – w pierwszym wykorzystano zbiór 1130 pytań z teleturnieju *Jeden z dziesięciu*, podczas gdy treść polskiej Wikipedii posłużyła za korpus tekstów. Dzięki technice DeepER, możliwe stało się także automatyczne sprawdzenie poprawności tych odpowiedzi, które sformułowano inaczej niż wzorcowe. Pozwoliło to na przeprowadzenie serii eksperymentów mierzących poprawność odpowiedzi w różnych konfiguracjach. W drugiej części uzyskane optymalne parametry wykorzystano w ręcznej ewaluacji wydajności systemu RAFAEL na oddzielnym zbiorze 576 pytań. Uzyskane wyniki pokazują, że użycie zaproponowanego podejścia znacząco poprawia udzielane odpowiedzi, głównie poprzez uwzględnienie pytań niedostępnych dla dotychczasowych metod.

Abstract

Title: *Question answering in Polish using deep entity recognition*

This thesis addresses a well-known problem in the domain of Natural Language Processing (NLP): Question Answering (QA). The task is to create a fully automatic computer system, capable of accepting questions in natural language (e.g. Polish) and returning answers using the same language. The problem plays an important role in computer systems that are expected to be used by people of low computer skills (e.g. search engines).

The work presents a description of a QA system for Polish, called RAFAEL (RApid Factoidal Answer Extracting aLgorithm), which can interpret factoid questions, find answers in a plain-text open-domain corpus and formulate them in concise and short manner. Some of the properties of Polish, i.e. rich nominal inflection and free word order make NLP sub-tasks, e.g. relevant documents selection and sentence matching, more challenging.

Unlike in other studies in the domain, the focus of this work is on a problem of entity recognition. In place of traditional Named Entity Recognition (NER) approach, its generalisation, called Deep Entity Recognition (DeepER), is proposed. Instead of using few general NE categories, entities are described by a set of WordNet synsets, to which they belong. It leads to much more precise candidate mention selection, according to question constraints. Moreover, a range of answered questions broadens, as those beyond the traditional NE categories (e.g. animals, devices, chemical compounds) are also accepted.

System evaluation consists of two steps – in the first part a set of 1130 questions from a Polish TV quiz show is used, while contents of Polish Wikipedia serve as a knowledge base. Thanks to DeepER technique, it is possible to automatically check validity of an answer even when it is formulated differently than expected. The method let to perform a series of experiments, measuring answering accuracy in varying configurations. Secondly, the obtained parameters were used to manually evaluate RAFAEL performance on previously unseen 576 questions. The results show that using DeepER approach substantially improves the answers, mainly by handling questions lying beyond the reach of existing methods.

Spis treści

Rozdział 1

Wprowadzenie

W ostatnich latach jesteśmy świadkami błyskawicznej rewolucji w technikach przetwarzania informacji. W miarę postępu technologicznego coraz większa część zadań związanych z tym obszarem przejmowana jest przez systemy komputerowe. Znamienne, że choć dzisiejsze rozwiązania różni wiele od tych, które zapoczątkowały te przemiany, pewien rodzaj problemów zdaje się trwać bez zmian. Trudności te wynikają z nieuchronnego konfliktu na linii człowiek – maszyna, powodowanego przez używanie odmiennych języków. Sposób wyrażania informacji w językach naturalnych, takich jak polski, diametralnie różni się od tego, jaki może być skutecznie przetwarzany przez systemy informatyczne. Jak dotąd trud przekroczenia tej bariery i nagięcia swojego języka do możliwości drugiej strony ponosili użytkownicy, zmuszeni do odbycia długiej edukacji przed wykonaniem najprostszych działań z użyciem komputera.

Jasnym wydaje się, że skoro komputery istnieją by ułatwiać ludziom życie, powinno to obejmować również metody komunikacji. Tego samego zdania byli pisarze literatury science-fiction i futurologicznej, którzy przedstawiali zdolność *mózgów elektronowych* do swobodnego posługiwania się ludzką mową, nie tyle jako spodziewane osiągnięcie, co raczej oczywistość (Lem, 1981). Dlaczego te przewidywania się nie sprawdziły? Zadanie przetwarzania języka naturalnego (ang. *Natural Language Processing* – NLP) stanowi wielkie wyzwanie ze względu na właściwości samego języka: kontekstowość, wieloznaczność, nieprecyzyjność i niepełność. Szczególne trudności sprawia ta ostatnia cecha – oznacza ona, że wypowiadane zdania zwykle nie niosą znaczenia bezpośrednio (*explicite*), lecz w połączeniu z ogólną wiedzą o świecie (*implicite*). Przykładowo, rozważmy następujące zdania: *Słonie zjadły wszystkie znalezione owoce. Wydają się być zadowolone z posiłku*. Z tego fragmentu nie wynika wprost, kto odczuwał zadowolenie – słonie czy owoce; z punktu widzenia składniowego podmiotem drugiego zdania może być każde z tych słów. Człowiek nie dostrzega tu niejednoznaczności, ponieważ jest przekonany, że owoce nie są zdolne do odczuwania czegokolwiek, w szczególności zadowolenia. Wiedza ta nie jest jednak dostępna dla systemu komputerowego. Co więcej, nie jest zapewne także wprost zapisana w takim źródle, jak np. encyklopedia czy słownik.

Mimo niezaprzeczalnych trudności, warto podejmować próby automatycznego przetwarzania języka naturalnego w celu budowy przyjaznych interfejsów użytkownika, nie tylko ze względu na wygodę korzystania z nich. Trzeba pamiętać, że wielu ludzi nie ma dostępu do solidnej edukacji w zakresie technik informatycznych. Gdy znaczna część wiedzy jest tworzona, przekazywana, przechowywana i udostępniana w sieci Internet, brak odpowiednich umiejętności wyklucza z tego obiegu na podobnej zasadzie, jak dawniej analfabetyzm

– stąd termin *cyfrowe wykluczenie* (ang. *digital exclusion*). Jako przykład rozważmy podstawowe narzędzie internetowe, tj. wyszukiwarkę tekstową. Aby sprawnie się nią posługiwać, trzeba opanować np. algebrę Boole'owską dla łączenia słów kluczowych, poznać słowa i znaki specjalne, charakterystyczne dla konkretnej wyszukiwarki. Niektórym z ważnych zasad formułowania zapytań daleko do intuicyjności, np. nie dla każdego oczywiste jest, że warto podawać słowa kluczowe, o których wiemy, że *nie* wystąpią na stronach, których *nie* oczekujemy. Wielu użytkownikom Internetu zapewne znacznie wygodniej byłby po prostu zadać pytanie, na które poszukują odpowiedzi. Choć tradycyjne wyszukiwarki nie są w stanie ich poprawnie interpretować, zapytania takie stanowią znaczną część ruchu. Na przykład w 2010 roku w rosyjskiej wyszukiwarce *Yandex* zadawano ich 3 miliony każdego dnia (Yandex.ru, 2010). Choć nie na wszystkie tak zgłaszane pytania można znaleźć krótką odpowiedź (najczęściej pytano *Jak schudnąć?*), to z pewnością wzbogacenie systemu wyszukiwania o rozumienie pytań w języku naturalnym zauważalnie zwiększyłoby satysfakcję użytkowników (Etzioni, 2011). Takie wzbogacenie interfejsu przyniosłoby korzyść również w przypadku systemów, których użytkownicy są wykształconymi specjalistami, ale niekoniecznie w zakresie informatyki (np. lekarze korzystający z bazy wiedzy medycznej).

Na podstawie powyższych rozważań łatwo wysnuć wniosek, że jednymi z najbardziej oczekiwanych przez użytkowników systemów realizujących komunikację w języku naturalnym są systemy odpowiadania na pytania. W niniejszej pracy przedstawiono wyniki badań realizujących następujące cele z tego obszaru:

1. Budowa pierwszego systemu zdolnego do automatycznego udzielania zwięzłych odpowiedzi na pytania o proste fakty na podstawie bazy tekstów w języku polskim bez ograniczeń dziedzinowych.
2. Zidentyfikowanie tych etapów procesu odpowiadania, dla działania których szczególne znaczenie mają własności języka polskiego, jak również eksperymentalne zweryfikowanie najlepszych rozwiązań.
3. Zaproponowanie i zastosowanie nowej metody rozpoznawania nazw, stanowiącej uogólnienie rozpoznawania nazw własnych i używającej synsetów WordNet w miejsce tradycyjnie rozumianych kategorii nazw własnych. Pozwala to na precyzyjniejsze określanie typu i uwzględnianie bytów spoza tych kategorii.
4. Opracowanie metody budowy bazy danych, kojarzącej nazwy (własne lub ogólne) z odpowiednimi synsetami, poprzez interpretację definicji encyklopedycznych w języku naturalnym i budowa takiego zasobu z wykorzystaniem polskiej Wikipedii.

Rozdział 2 zawiera szczegółową definicję podjętego problemu, określanego jako otwarto-dziedzinowe odpowiadanie na pytania o proste fakty w języku polskim na podstawie bazy tekstowej. System wykonujący to zadanie, nazwany RAFAEL (ang. *RApid Factoidal Answer Extracting aLgorithm*), został opisany w rozdziale 3. Przy jego budowie szczególną uwagę zwrócono na kwestię rozpoznawania nazw, opracowując nowe podejście do tematu, opisane w rozdziale 4 i nazwane głębokim rozpoznawaniem nazw (ang. *Deep Entity Recognition* – *DeepER*). Praca ta nie byłaby kompletna bez ilościowej oceny uzyskanych re-

zultatów. W rozdziale 5 umieszczono opis eksperymentów, mających na celu poznanie zachowania systemu w różnych warunkach oraz wyniki ostatecznej ewaluacji. Rozdział 6 składa się z analizy uzyskanych wyników, propozycji dalszych prac i podsumowania. Na końcu umieszczono dodatek, zawierający definicje niektórych pojęć.

Rozdział 2

Zadanie odpowiadania na pytania

O zadaniu odpowiadania na pytania (ang. *question answering* – QA) mówimy w sytuacjach, w których oczekujemy od systemu komputerowego przyjmowania od użytkowników żądań w postaci pytań w języku naturalnym (np. polskim) i udzielania odpowiedzi sformułowanych w języku naturalnym. Niniejszy rozdział poświęcono wskazaniu miejsca, jakie w tej szerokiej tematyce zajmuje problem rozważany w niniejszej pracy. Najpierw przedstawiono historię rozwoju dziedziny z uwzględnieniem dotyczących tego zagadnienia zawodów organizowanych przy konferencjach. Następnie szczegółowo zdefiniowano problem rozwiązywany w niniejszej pracy. Inne systemy realizujące takie samo (lub bardzo zbliżone) zadanie zaprezentowano pod kątem ogólnej struktury systemu (podrozdział 2.3) i obsługi języków słowiańskich (podrozdział 2.4). Ponieważ zdefiniowany problem stanowi jedynie fragment szerokiej dziedziny, w ostatniej części zamieszczono przegląd rozwiązań, które, choć pozostają poza zakresem niniejszej pracy, związane są z ogólnie sformułowanym zadaniem QA i dlatego też warte są uwagi. Rozdział ten stanowi rozwinięcie wcześniejszej pracy (Przybyła, 2012); omówienie dziedziny z punktu widzenia języka polskiego przedstawił także Vetulani (2002, 2004).

2.1. Zarys historii

Pierwszy system typu QA – *BASEBALL* (Green i inni, 1961) – powstał w roku 1961 w *Massachusetts Institute of Technology* przy wsparciu armii USA. Był on zdolny do odpowiadania na pytania z ograniczonej dziedziny, tj. przebiegu gier w baseball w lidze USA rozegranych w ciągu jednego roku. *BASEBALL* wykorzystywał bazę danych zawierającą jedynie informacje o dacie i miejscu meczu, występujących w nim drużynach i wyniku. Analiza pytania prowadziła do wypełnienia specjalnej listy specyfikacyjnej (ang. *spec list*), zawierającej zarówno informacje znane, jak i wskazanie, których elementów dotyczy pytanie. Przykładowo, pytaniu *Gdzie drużyna Red Sox grała siódmego lipca?* odpowiada następująca lista par atrybut=wartość:

- Miejsce = ?,
- Drużyna = Red Sox,
- Miesiąc = Lipiec,
- Dzień = 7.

Analiza pytania polegała na sprawdzaniu kolejnych słów w słowniku zawierającym przypisane im fragmenty listy. Przykładowo, słowu *gdzie* odpowiada para *Miejsce = ?*. System uwzględniał również słowa niejednoznaczne, np. *Boston* może odnosić się zarówno do miejsca rozgrywki (*Miejsce =*

Boston), jak i określać jedną z występujących drużyn (*Drużyna = Red Sox*). Każdemu słowu tego typu w słowniku przypisano również procedurę ujednoznaczniającą, korzystającą z kontekstu jego wystąpienia. Uwzględniono także proste parsowanie powierzchniowe, dzięki któremu rozpoznawano wyrażenia wielosłowne, posiadające własne wpisy w słowniku. Informacja przypisana do niektórych słów może oznaczać modyfikację wartości atrybutów przypisanych w wyniku rozpoznania innego słowa w tekście, np. słowo *wygrywająca* modyfikuje element typu *Drużyna* do postaci *Drużyna_(wygrywająca)*. Przy niektórych czasownikach, np. *pokonać*, aby właściwie przypisać modyfikatory, trzeba dokonać szerszej analizy struktury zdania. W tym wypadku oznacza to, że modyfikator *(wygrywająca)* dołączamy do atrybutu odpowiadającego podmiotowi zdania, a *(przegrywająca)* do atrybutu odpowiadającego dopełnieniu. Dzięki ustalonemu porządkowi elementów zdania w języku angielskim można tego dokonać w większości przypadków bazując jedynie na kolejności słów.

Po powstaniu listy specyfikacyjnej odpowiedni proces (ang. *processor*) poszukiwał w bazie danych rozgrywek, których opis można uzgodnić z listą. System przyjmował na wejściu pytanie zapisane na kartach perforowanych, a wyjście przekazywał na drukarkę. *BASEBALL* był w stanie odpowiadać na dość złożone pytania, np. *Które drużyny zagrały przynajmniej 4 mecze w marcu?* lub *Czy każda drużyna grała przynajmniej raz na każdym stadionie w każdym miesiącu?*. Autorzy planowali, że po umożliwieniu bezpośredniej komunikacji systemu z użytkownikiem, uda się rozszerzyć funkcjonalność programu o zadawanie pytań o znaczenie nieznanych słów i w konsekwencji rozszerzanie słownictwa.

W latach sześćdziesiątych ubiegłego wieku powstało więcej systemów QA, które odpowiadały na pytania z ograniczonej dziedziny na podstawie bazy wiedzy o ustalonej strukturze; ich przegląd przedstawił Simmons (1970). Podejmowano także próby przełożenia znaczenia zdań w języku naturalnym na logikę pierwszego rzędu, co pozwalałoby odnajdywać odpowiedzi z użyciem narzędzi logiki predykatów. Niestety, konieczność ręcznego przygotowania reguł takiego przekształcenia uniemożliwiała zastosowanie ich do tekstów z nieograniczonej dziedziny.

W późniejszym czasie uwagę zaczęła przyciągać rozwijająca się sieć WWW, traktowana jako źródło potencjalnych odpowiedzi. Na przykład w systemie *START* (Katz, 1997) korpus dokumentów pobranych z Internetu przekształcano do bazy tzw. T-wyrażeń w postaci trójek <obiekt_1 relacja obiekt_2>. Powstawały one w wyniku analizy zdań w dokumentach tekstowych. Zdanie wyjściowe najpierw rozbijano do poziomu tzw. *zdań prostych* (ang. *kernel sentences*), zawierających co najwyżej jeden czasownik. Następnie w wyniku ich analizy opracowywano drzewa rozbioru, z których wydobywano podstawową informację o podmiocie, orzeczeniu i dopełnieniu. Ta trójka elementów tworzyła właśnie podstawowe T-wyrażenie. Inne elementy zdania (przymiotniki, konstrukcje przyimkowe, etc.) także mogą być źródłem T-wyrażeń, np. w wyrażeniu przyimkowym rolę relacji pełni przyimek. Funkcję obiektu w T-wyrażeniu może realizować inne T-wyrażenie. Wykorzystując tę możliwość, całą treść zdania złożonego wyrażano przez jedno potencjalnie wielokrotnie złożone T-wyrażenie.

W ramach odpowiadania na pytanie użytkownika, system przekształcał je do postaci T-wyrażenia, przy czym w miejsce poszukiwanej treści wstawiany był odpowiedni zaimek. Potem następowało przejrzenie całej bazy wiedzy, aż do znalezienia pasującego T-wyrażenia. Jako odpowiedź zwracano zdanie, na podstawie którego powstało odnalezione T-wyrażenie. Proces taki umożliwiał uzyskiwanie wiarygodnych odpowiedzi, jednak problem stanowiła redundancja języka naturalnego – pozwala on na zapisanie tej samej informacji na wiele sposobów. Jeśli pytanie sformułowane było istotnie inaczej niż zdanie wprowadzające daną informację, to odpowiedź nie mogła być udzielona. Autor proponował jako rozwiązanie S-przekształcenia, które sprowadzają różne konstrukcje językowe o tym samym lub bardzo zbliżonym znaczeniu do identycznego T-wyrażenia. Przykładowo, wyrażenie 1 implikuje wyrażenie 2:

1. <<obiekt_1 *surprised* obiekt_2> *with* obiekt_3> (*obiekt 1 zaskoczył obiekt 2 obiektem 3*)[1]
2. <obiekt_3 *surprised* obiekt_2> (*obiekt 3 zaskoczył obiekt 2*)

Łatwo zauważyć, że powyższa własność dotyczy nie tylko czasownika *zaskakiwać*, ale także *złościć*, *rozczarowywać*, *irytować*, itp. Słowa te tworzą klasę czasowników emocjonalnych i podane S-przekształcenie dotyczyło każdego z nich. Przygotowanie systemu do uwzględniania przeformułowań w języku naturalnym obejmowało zatem dwa kroki – grupowanie słów w klasy i tworzenie dla każdej z nich możliwych S-przekształceń. Podobnie jak w przypadku systemu *BASEBALL*, autor nie zamieścił żadnej ilościowej oceny wydajności systemu. Podał natomiast przykłady pytań, na które udało się wskazać odpowiedź, np. *Ile masowych grobów znajduje się w Bośni?*, *Czy dziś pada w Belgradzie?*, *Jak porozumienie z Dayton dzieli Bośnię?*

Wydarzeniem, które podziałało stymulująco na społeczność badaczy systemów QA, były konkursy odpowiadania na pytania w języku angielskim towarzyszące konferencjom TREC (ang. *Text REtrieval Conference*) od roku 1999 do 2007. Opisy zadań opublikowali Voorhees (1999, 2000, 2001, 2002, 2003), Voorhees i Dang (2005) oraz Dang i inni (2006, 2007). Umożliwiły one porównanie wielu rozwiązań tego samego problemu, polegającego na odnajdowaniu odpowiedzi w ustalonym korpusie tekstów z nieograniczonej dziedziny. W roku 2007 zadawano 515 pytań podzielonych na 70 serii, z których każda dotyczyła jednego, wybranego tematu (np. *Guiness Brewery* czy *Jon Bon Jovi*). Pytania należały do jednej z trzech kategorii:

- **FACTOID** – pytania, na które odpowiedzią była pojedyncza nazwa[2], np. *Którego dnia nastąpił wstrząs?*[3],
- **LIST** – pytania, na które odpowiedzią była lista nazw, np. *Jakie kraje zostały dotknięte trzęsieniem?*,

[1] Próba tłumaczenia tego przykładu ukazuje trudności, na jakie się napotyka, stosując tak proste podejścia do języków o swobodnym szyku zdania – w polskim zdaniu brakuje odpowiednika słowa *with*, gdyż informacja ta przekazywana jest poprzez użycie nazwy obiektu 3 w narzędniku.

[2] Zobacz słownik pojęć w Dodatku.

[3] Przykłady pochodzą z serii o trzęsieniu ziemi w Pakistanie w październiku 2005, użytej na TREC 2007 (Dang i inni, 2007).

- **OTHER** – pytania, na które należało odpowiedzieć poprzez podanie interesującego faktu, który nie stanowił odpowiedzi na żadne z pytań w ramach tej samej serii.

W rywalizacji uczestniczyli reprezentanci dwóch firm i ośmiu uniwersytetów. Głównym wskaźnikiem wydajności w przypadku pytań FACTOID była prawidłowość odpowiedzi oceniana ręcznie przez sędziego. Co ciekawe, najlepsze wyniki (70,6% i 49,4%) odnotowały rozwiązania komercyjne, zaś systemy akademickie osiągnęły znacznie słabsze rezultaty (poniżej 30 %). Podobnie sytuacja wyglądała w przypadku pytań typu LIST, gdzie mierzono miarę F1 zwracanych list. Najlepsze wyniki wyniosły 0,479 i 0,324, zaś reprezentacje uniwersytetów odnotowały wartości poniżej 0,15.

W 2008 roku zadanie TREC QA przekształciło się w *Opinion Question Answering* na konferencji TAC (*Text Analysis Conference*), gdzie cel stanowiło odnajdywanie opinii w korpusie blogów z uwzględnieniem ich wydźwięku (Dang, 2008). Interesujące wyzwanie wielojęzycznego odpowiadania na pytania (*Cross-Lingual Question Answering* – CLQA) postawiono na warsztatach NTCIR (*NII Test Collection for IR Systems*). Pytania przypominały kategorię FACTOID z TREC, a dodatkowa trudność polegała na konieczności włączenia w system modułu tłumaczącego, gdy pytanie sformułowano w innym języku niż bazę wiedzy (Sasaki i inni, 2007). W eksperymentach uwzględniono angielski, japoński i chiński. Analogiczne zadanie zrealizowano na *9th Cross-language evaluation forum* (CLEF 2008), na którym brano pod uwagę aż 11 języków, w tym jeden słowiański – bułgarski. Testowano systemy w 43 z możliwych 121 kombinacji (Forner i inni, 2008). W następnej edycji tej konferencji zadania CLQA zostało zastąpione przez ResPubliQA (Peñas i inni, 2009). Problem polegał na udzielaniu odpowiedzi na 500 pytań dotyczących europejskiej legislacji. Oprócz prostych faktów[4], dotyczyły one także definicji pojęć, przyczyn wydarzeń, celów działań i opisów procedur. Bazę wiedzy stanowił podzbiór wielojęzycznego korpusu równoległego JRC-ACQUIS, zawierającego całość aktów prawnych Unii Europejskiej w 22 językach. Nietypowo wyglądało zadanie *Question Answering for Machine Reading Evaluation* (Q4MRE) na CLEF 2012 (Peñas i inni, 2012). System otrzymywał potencjalne odpowiedzi na pytania, a zadanie polegało na wybraniu z 5 jednej prawidłowej. Dzięki temu nacisk położono nie na wyszukanie odpowiedniego fragmentu w tekście, jak zazwyczaj, ale na głębsze zrozumienie jego znaczenia. Dane przygotowano w językach: angielskim, niemieckim, włoskim, rumuńskim, hiszpańskim, arabskim i bułgarskim. Inną okazję do porównania różnych podejść stanowił program AQUAINT (*Advanced QUestion Answering for Intelligence*), realizowany w latach 2002-2006 i finansowany przez rząd USA (National Institute of Standards and Technology, 2010). Miał on na celu sprawdzenie możliwości wykorzystania QA w celach wywiadowczych, w związku z tym pytania i teksty dotyczyły polityki międzynarodowej, terroryzmu, etc.

Kolejne, po otwartych konkursach, ważne wydarzenie w historii QA, to opracowanie przez zespół z działu badawczego IBM systemu *Watson* (Ferrucci i inni, 2010). System ów uczestniczył w teleturnieju *Jeopardy!* na żywo i z łatwością pokonał zwycięzców poprzednich edycji. Teleturniej ten wyróżnia się

[4] Zobacz słownik pojęć w Dodatku.

odwróconą rolą pytań i odpowiedzi – zawodnikom przedstawia się wskazówki w formie twierdzącej i oczekuje się zadania pasującego pytania. Przykładowo, zamiast zwykłego pytania *Który europejski władca pokonał armię Imperium Osmańskiego pod Wiedniem?* pojawia się podpowiedź *Ten europejski władca pokonał armię Imperium Osmańskiego pod Wiedniem.*, a zawodnik powinien zareagować pytaniem *Kim był Jan III Sobieski?* Zasada ta nie ma tak dużego znaczenia dla budowy systemu, jak obowiązujące w tym turnieju ograniczenie czasowe. Gracz ma 5 sekund na zgłoszenie się do odpowiedzi, jednak zazwyczaj trwa to znacznie krócej, gdyż tylko pierwszy zgłaszający się ma szansę udzielenia odpowiedzi. Kluczową rolę odgrywa zatem szybkie oszacowanie pewności odpowiedzi. W każdej rozgrywce zadaje się pytania w ramach sześciu kategorii po pięć pytań. Najczęściej kategorie odpowiadają pewnym zakresom tematycznym i ich nazwa może stanowić wskazówkę; mogą jednak obejmować trudniejsze problemy w postaci zagadek, gier słownych czy skojarzeniowych.

Strukturę systemu *Watson* omówiono w podrozdziale 3.1. Precyzja udzielanych przez system odpowiedzi zależy od tego, na jaką część z zadawanych pytań są one udzielane – dla całkowitego pokrycia zbioru testowego otrzymano wartość około 65%, zaś wybieranie tylko 20% najpewniejszych pytań podniosło wynik powyżej 95%. Zwycięstwo *Watsona* nad ludźmi w teleturnieju przyciągnęło znaczną uwagę – firma IBM wskazuje na analogie z pokonaniem mistrza szachowego Garriego Kasparowa przez komputer IBM *Deep Blue*.

Opracowanie i implementacja systemu *Watson* zostały poprzedzone wysunięciem przez jego autorów propozycji budowy *QA Open Collaboration Framework* (Ferrucci i inni, 2009), czyli platformy organizacyjno-programowej, dzięki której przedstawiciele instytucji badawczych i komercyjnych mogliby wydajnie współpracować w tej dziedzinie. Aby nie tylko wiarygodnie porównywać rezultaty, ale i wymieniać się komponentami systemów, konieczne jest opracowanie wspólnego modelu architektury, wykorzystującego elementy o otwartym kodzie źródłowym.

W kontekście możliwości porównywania rezultatów trzeba także wspomnieć o polskiej inicjatywie. Zespół z Politechniki Wrocławskiej (Marcińczuk i inni, 2013a) publicznie udostępnił zbiór 4721 pytań, zebranych z rubryki *Czy wiesz . . . ?* z polskiej Wikipedii, a następnie oczyszczonych i ujednoliconych. Dodatkowo, do każdego z pytań przypisano oczekiwaną odpowiedź. Jedynym ograniczeniem z punktu widzenia wykorzystania tego zbioru jako danych testowych w niniejszej pracy jest rozumienie odpowiedzi jako artykułu źródłowego, a nie krótkiego łańcuchu znaków, zawierającego oczekiwaną informację. Zbiór ewaluacyjny dla RAFAELa (zob. podrozdział 5.1) powstał przez wzbogacenie właśnie tego zestawu pytań o niezbędne dane.

Na koniec warto zwrócić uwagę na pracę z 2001 roku (Burger i inni), w której przedstawiono przegląd otwartych kwestii w dziedzinie. Spojrzenie na nie z perspektywy kilkunastu lat ukazuje, z jakim wyzwaniem mamy do czynienia – większość spośród wymienionych problemów wciąż pozostaje nierozwiązana. W kontekście analizy pytania wyszczególniono tam następujące zadania:

- dopuszczenie większej złożoności pytań,

- rozumienie i uwzględnianie kontekstu,
- nieograniczony zakres tematyczny.

Jeszcze trudniej przedstawiają się wymagania co do udzielanych odpowiedzi, gdyż powinny one:

- pochodzić z wielu źródeł różniących się typem i językiem,
- łączyć zebrane przesłanki i rozstrzygać sprzeczności,
- wzbogacać zdobyte dane o interpretacje i wyciągać wnioski.

W pracy przedstawiono czterostopniową hierarchę użytkowników systemów QA, bazującą na ich wymaganiach wobec systemu, sformułowanych między innymi w postaci powyższych postulatów. Obejmuje ona następujące poziomy:

1. użytkownik nieprofesjonalny,
2. zaawansowany użytkownik nieprofesjonalny,
3. początkujący dziennikarz,
4. profesjonalny analityk.

Aktualnie dostępne rozwiązania wypełniają jedynie wymagania z pierwszego poziomu. Pytania typowe dla czwartego poziomu to np. *Jakie środki podejmuje Microsoft, aby utrzymać wiodącą pozycję na rynku oprogramowania?* lub *Dlaczego firmy farmaceutyczne używają oprogramowania Oracle?*, wymagające syntezy wiedzy z wielu źródeł.

2.2. Definicja zadania

Określenie rozwiązywanego w pracy problemu jako pełnego zagadnienia QA, czyli budowy systemu komputerowego zdolnego do odpowiadania na dowolne pytania w języku naturalnym, na które odpowiedź znajduje się w zadanej bazie wiedzy, uniemożliwiłoby znalezienie rozwiązania. Problem taki byłby zbyt trudny – Shapiro (1992) określa go jako należący do klasy AI-zupełnych (ang. *AI-complete*). Przez analogię do problemów NP-zupełnych oznacza to, że rozwiązanie takiego zadania jest równoznaczne budowie sztucznej inteligencji. Łatwo to zrozumieć, gdy uzmysłowimy sobie, że niemalże cała ludzka wiedza została zakodowana w formie tekstowej, a zatem interfejs językowy prawie nie ogranicza oczekiwanych możliwości systemu. Także Turing (1950) uważał, że zdolność maszyny cyfrowej do swobodnej rozmowy w języku naturalnym wystarczy do uznania jej za myślącą, co doprowadziło do zaproponowania słynnego (choć kontrowersyjnego) *testu Turniga*.

Problem stanowiący przedmiot rozważań w niniejszej pracy można określić jako **otwarto-dziedzinowe odpowiadanie na pytania o proste fakty w języku polskim na podstawie bazy tekstowej**. Oznacza to, że oczekiwane rozwiązanie powinno obejmować:

- interpretowanie pytań zadanych w języku naturalnym,
- wykorzystywanie jako źródła informacji zbioru tekstów należących do dowolnej dziedziny,

- odnajdywanie i udzielanie odpowiedzi w formie możliwie krótkiego wyrażenia w języku naturalnym,
- operowanie na tekstach i pytaniach w języku polskim.

Aby precyzyjnie określić kontekst, w jakim funkcjonuje system, trzeba opisać trzy rodzaje danych, za pośrednictwem których komunikuje się on z użytkownikiem. Są to: pytania, reprezentujące potrzeby użytkownika; teksty, wśród których poszukiwane będzie rozwiązanie; oraz przekazywana użytkownikowi odpowiedź. Poniżej przedstawiono szczegółową specyfikację wymienionych elementów składowych, z którą muszą być również zgodne dane używane do ewaluacji systemu.

2.2.1. Pytania

Na potrzeby niniejszej pracy pytanie zdefiniujmy jako pojedyncze krótkie zdanie, będące poprawnym pytaniem w języku polskim, na które odpowiedź stanowi prosta informacja zawarta w bazie tekstów. Takie krótkie informacje określamy mianem *prostych faktów* (ang. *factoid*, patrz słownik pojęć w Dodatku). Na podane wyżej sformułowanie nakładamy także następujące dodatkowe ograniczenia:

- Informacja w tekście może być inaczej sformułowana, podana nie wprost lub wymagać połączenia fragmentów z kilku tekstów.
- Odpowiedź powinna być możliwa do określenia bez przeprowadzania złożonego rozumowania lub obliczeń; jedynie na podstawie prostej analizy tekstu.
- Pytanie samo w sobie wystarcza do znalezienia odpowiedzi w tekście – nie uwzględnia się żadnych informacji o kontekście, w jakim zostało zadane, czy dziedzinie, której dotyczy.

Jak widać z powyższego, bierzemy pod uwagę tylko pewien podzbiór zbioru wszystkich możliwych pytań. W szczególności wykluczone zostają pytania wymagające odpowiedzi opisowej, np. *Czym jest globalne ocieplenie?*, jak również żądania nie będące pytaniami, np. zaczynające się od *Podaj . . .* , *Wymień . . .* itp. O ile pierwszy z tych wymogów pozostawia wiele ciekawych zagadnień poza zakresem niniejszych badań (część wspomniano w podrozdziale 2.5), o tyle drugi upraszcza jedynie analizę pytania bez istotnego ograniczania zbioru możliwych pytań.

Z tak określonego podzbioru pytań o proste fakty wyodrębniamy te, na które można odpowiedzieć podając pojedynczą nazwę bytu[5] odnalezioną w tekście i nazywamy je *pytaniami o nazwy bytów* (ang. *entity questions*). RAFAEL udziela odpowiedzi tylko na takie pytania, ale ograniczenie to ma niewielki wpływ na osiągane wyniki, ponieważ stanowią one zdecydowaną większość w używanych zbiorach ewaluacyjnych. Dokładny opis kategorii pytań znajduje się w części 3.3.1, zaś ich rozkład w zbiorach używanych do ewaluacji w części 5.1.2.

[5] Zobacz słownik pojęć w Dodatku.

2.2.2. Teksty

Poziom trudności rozważanego zadania zależy w dużym stopniu od charakterystyki tekstów, z których wydobywane będą odpowiedzi. Poniżej opisano te własności dużych baz tekstowych, dla których uzasadnione jest budowanie systemów QA. Stanowią one również wymagania, jakie powinna spełniać baza tekstów, aby mogła posłużyć do wiarygodnego przebadania wydajności systemu:

- **językowa poprawność**: Pojawienie się w tekście błędów składniowych lub ortograficznych, skrótów, kolokwializmów czy braków interpunkcyjnych zwykle znacząco utrudnia automatyczną analizę. Niestety, wymaganie poprawności językowej wyklucza część treści opublikowanych na stronach internetowych (szczególnie tworzonych przez użytkowników, np. fora, sieci społecznościowe, czaty).
- **informatywność**: To wymaganie dość oczywiste – od bazy tekstowej oczekujemy, aby zawierała wiele faktów, o które warto pytać.
- **obfitość**: Systemy QA mają zastosowanie głównie tam, gdzie ilość tekstu do przetworzenia przekracza możliwości przyswojenia przez człowieka. Wiąże się z tym zjawisko *szumu informacyjnego*, czyli obecność nadmiaru informacji pozornie związanej z tematem, ale w rzeczywistości nie zawierającej poszukiwanej wiedzy.
- **wielo-dziedzinowość**: Systemy zdolne pracować z tekstami z dowolnej dziedziny są istotnie trudniejsze do opracowania. W celu uniknięcia dopasowania się systemu do pewnego typu tekstów, charakterystycznych dla jednego tematu, dlatego dopuszcza się obecność zarówno pytań, jak i tekstów bez ograniczeń dziedzinowych.
- **wiarygodność**: Oczekiwanie to jest oczywiste, jeśli chcemy, aby udzielane odpowiedzi spełniały wymagania użytkowników. W niniejszej pracy przyjęto założenie, że wiarygodność bazy tekstów została zweryfikowana przy jej przygotowywaniu i nie należą do celów systemu takie zadania jak wykrywanie sprzeczności czy szacowanie wiarygodności tekstów.

Podane powyżej wymagania były brane pod uwagę na etapie wyboru korpusu do testowania systemu. W przypadku zadań w języku angielskim najczęściej używa się archiwów prasowych – gwarantują one dużą ilość informacji, jak również dobrą jakość języka. Z drugiej strony, są one niestety dość odległe w stylu od tego języka, który można znaleźć w sieci WWW, stanowiącej największe obecnie wyzwanie dla badań NLP. W niniejszej pracy zdecydowano się na kompromis w postaci polskiej Wikipedii (szczegóły w podrozdziale 5.1), która łączy wysoką informatywność z dosyć dobrą jakością języka, szczególnie biorąc pod uwagę jej tworzenie przez „zwykłych” użytkowników Internetu.

2.2.3. Odpowiedzi

Zwracane przez systemy QA odpowiedzi, podobnie jak te udzielane przez człowieka, mogą przybierać różne formy. Należy zatem określić, jakiej postaci od-

powiedzi oczekujemy. Można wyróżnić następujące poziomy szczegółowości (dla pytania *Kiedy Albert Einstein opublikował szczególną teorię względności?*[6]):

1. Identyfikator dokumentu zawierającego odpowiedź, tj. *Wikipedia:Albert_Einstein*. Po stronie użytkownika wciąż pozostaje trud odnalezienia żądanej informacji – znaczny, jeśli weźmiemy pod uwagę długość konkretnego tekstu.
2. Pojedyncze zdanie z bazy tekstów, zawierające odpowiedź. Niestety, nie zawsze w dokumencie takie się znajduje. Tak jest i w tym wypadku – data pojawia się w zdaniu *Rok 1905 jest określany jako Annus mirabilis (cudowny rok) Einsteina.* Dopiero w dalszej części akapitu znajdują się odwołania do szczególnej teorii względności (przykład ten szerzej omówiono w punkcie 6.1.3).
3. Forma zawierająca wyłącznie odpowiedź, tj. w tym wypadku określenie czasu: *Rok 1905*.
4. Pełne zdanie sformułowane na potrzeby odpowiedzi, tj. *Albert Einstein opublikował szczególną teorię względności w roku 1905.*

Łatwo zauważyć, że satysfakcja użytkownika rośnie w miarę przechodzenia do wyższych poziomów sposobu formułowania odpowiedzi. W niniejszej pracy zdecydowano się na poziom 3., czyli sformułowanie zwięzłej odpowiedzi zawierającej jedynie opis faktu, o który pytano. Po pierwsze, odpowiedź taka jest w większości przypadków dla człowieka całkowicie wystarczająca, a często nawet preferowana. Po drugie, próba sformułowania pełnego poprawnego zdania wymagałaby zaangażowania technik generowania wypowiedzi w języku naturalnym (ang. *Natural Language Generation* – NLG), które, jako stanowiące odrębne, nierozwiązane jeszcze zadania, leżą poza zakresem tej pracy. Warto także zwrócić uwagę, że w literaturze spotyka się również rozwiązania pośrednie, korzystające z wielozdaniowych fragmentów – np. w pracy Marcińczuka i innych (2013b) jako odpowiedź zwracany jest najlepszy dokument, tj. posiadający najwyższą ocenę. Aby jednak obliczyć tę ocenę, dzieli się tekst na kilkuzdaniowe ustępy, których podobieństwo do pytania wpływa na ocenę całego dokumentu.

2.3. Stosowane podejścia

Istnieje wiele sposobów klasyfikacji systemów QA, które bazują głównie na różnicach w definicji rozwiązywanego zadania. Ten aspekt podziału omówiony jest w podrozdziale 2.5. Inny typ podziału wprowadzono w pracy przeglądowej (Andrenucci i Sneiders, 2005), w której wyróżniono trzy grupy rozwiązań: bazujące na przetwarzaniu języka naturalnego, wyszukiwaniu dokumentów oraz dopasowywaniu wzorców. Jeżeli ograniczymy się do prac nad problemami zbliżonymi do przedstawionego w podrozdziale 2.2, to największe zróżnicowanie opracowywanych rozwiązań dotyczy strategii poszukiwania zdań pasujących do zadanego pytania. Mamy tu do czynienia z czterema nurtami badań:

[6] Odpowiedzi z artykułu Wikipedii `http://pl.wikipedia.org/wiki/Albert_Einstein`.

1. dopasowywanie wzorców,
2. dopasowywanie strukturalne,
3. dopasowywanie semantyczne,
4. dopasowywanie statystyczne.

Podział ten nie jest ścisły, ponieważ istnieją rozwiązania łączące cechy kilku z powyższych nurtów.

2.3.1. Dopasowywanie wzorców

Podejście bazujące na dopasowywaniu wzorców jest najstarsze i najprostsze – polega ono na opracowaniu wzorców odpowiedzi dla każdego typu pytań i poszukiwaniu w tekście zdań, dla których zachodzi zgodność ze wzorcem[7]. Biorąc pod uwagę możliwości języków naturalnych, założenie, że zdanie zawierające odpowiedź przyjmie z góry ustaloną postać, może wydać się naiwne (szczególnie w przypadku języków o swobodnym szyku zdania (Przepiórkowski, 2007)). Jednak, gdy jako zbioru danych użyjemy sieci WWW (to podejście jest opisane dokładniej w punkcie 2.5.1), to dzięki jej redundantności bywa ono spełnione.

Za przykład zastosowania dopasowywania wzorców w korpusie WWW może posłużyć praca Ravichandrana i Hoviego (2002). W tym rozwiązaniu baza wzorców przypisanych poszczególnym typom pytań tworzona jest w procesie iteracyjnym. Algorytm rozpoczyna od początkowego zestawu słów zgodnych z danym typem pytania, np. dla pytania o rok narodzin będą to nazwiska osób i lata. Następnie na tej podstawie pobiera się z korpusu zdania zawierające wszystkie słowa z zapytania. Z tych zdań wydobywa się powtarzające się frazy, np. *urodził się*, *przyszedł na świat*. Wyrażenia te mogą posłużyć do wydobycia nowych par osoba-data narodzin i wykonania następnego przebiegu algorytmu, dołączającego kolejne wyrażenia. Gdy system otrzymuje pytania od użytkownika, z powstałej bazy pobiera się wzorce zgodne z typem pytania, pozwalające na uwzględnienie wielu różnych sformułowań przy poszukiwaniach odpowiedzi. Podobny schemat (*bootstraping*), także do korpusu WWW, zastosowali Duclaye i inni (2003), przy czym do obliczeń użyto algorytmu maksymalizacji wartości oczekiwanej (ang. *expectation maximisation*, EM).

Jeśli chodzi o systemy realizujące zadania podobne do tego podejmowanego w niniejszej pracy, to dopasowywanie wzorców stosuje się znacznie rzadziej. Na przykład Miliaraki i Androutsopoulos (2004) użyli ich do odpowiadania na pytania z konferencji TREC 2003, ale tylko w ramach jednego typu, tj. definicji. Przykładowy wzorzec postaci *X and other Y* (*X i inne Y*) pozwala z dużym prawdopodobieństwem uznać, że *Y* opisuje *X* – np. *Paris and other European capitals* (*Paryż i inne stolice europejskie*). W systemie *WEBCLOPEDIA* (Hovy i inni, 2001) wzorce znalazły zastosowanie w pierwszej z trzech technik dopasowywania. Pojawia się tam też propozycja automatycznego ich pozyskiwania w drodze uczenia się (autorzy nie opisali jednak szczegółów tego procesu). Także Soubbotin i Soubbotin (2002) wykorzystali wzorce znacznie krótsze niż całe zdania. Przykładowo, w tekście może pojawić się fraza *FBI Di-*

[7] Dopasowywanie wzorców to również jedna z technik klasyfikacji pytań – więcej o tym w podrozdziale 3.3.1.

rector Louis Freeh (dyrektor FBI Louis Freeh). Wzorzec, który można tu zastosować, to ciąg następujących składników: słowo składające się z wielkich liter, słowo oznaczające stanowisko oraz imię i nazwisko. Po rozpoznaniu takiego elementu jesteśmy w stanie odpowiedzieć na pytanie: *Kto jest dyrektorem FBI?*.

2.3.2. Dopasowywanie strukturalne

Do drugiej kategorii należą takie rozwiązania, w których dopasowywanie pomiędzy pytaniem i odpowiedzią odbywa się z uwzględnieniem ich struktury, przy czym mowa o strukturze bogatszej niż sama kolejność słów (uwzględniana przez poprzednio omawianą kategorię bazującą na wzorcach). Kluczowym elementem takich rozwiązań, decydującym o ich własnościach i wydajności, będzie zatem typ owej struktury.

W systemie *SHAPAQA* (Buchholz i Daelemans, 2001) zdania podlegają parsowaniu powierzchniowemu, dzięki czemu wyodrębnione zostają grupy składniowe. Ich dopasowanie do pytania może może nastąpić na dwa sposoby. W pierwszym z nich wykrywane są relacje składniowe między grupami (typu podmiot, orzeczenie, okolicznik czasu itp.), a od odpowiedzi oczekuje się, żeby znajdowała się w strukturze analogicznej do zaobserwowanej w pytaniu. Sposób drugi nakłada niższe wymagania – wystarczy, że analizowana grupa posiada odpowiedni typ nazwy własnej[8]. Zgodnie z tym, czego należało się spodziewać, dopasowywanie struktury skutkuje wyższą precyzją odpowiedzi, zaś ograniczenie się do typu nazwy własnej lepszym pokryciem (liczbą pytań, na które mogła być udzielona odpowiedź).

Harabagiu i inni (2001) wykorzystują pełne drzewo rozbioru zdania, uzyskane przy wykorzystaniu odpowiedniego parsera. Oczywiście nie sposób oczekiwać odpowiedniości struktury pytania i odpowiedzi; szczególnie, że w tej pracy, w odróżnieniu od poprzednio omawianej, nie wykorzystuje się całej sieci WWW, a korpus TREC. Z tego powodu drzewo podlega znacznemu uproszczeniu i przekształceniu do struktury zależnościowej, w której nie rozróżnia się typów krawędzi. Dopiero taka reprezentacja pytania i potencjalnej odpowiedzi podlega dopasowaniu i pomiarowi podobieństwa.

Pełne drzewo rozbioru bierze się pod uwagę także w systemie *InSicht* (Hartrumpf, 2004), odpowiadającym na pytania w języku niemieckim. W tym wypadku inaczej rozwiązano problem rozluźnienia wymaganej zgodności zdań – zamiast upraszczania struktury pytania, generuje się wiele jego przekształceń, odpowiadających możliwym przeformułowaniom. Przykładowo, czasownik *zabić* można zastąpić przez *zginąć* bez dużej zmiany znaczenia, o ile pamięta się o modyfikacji ról innych składników zdania (tu dopełnienie stanie się podmiotem). Łatwo zauważyć, że liczba możliwych przekształceń tego typu jest znaczna – dla zbioru testowego reguły czysto składniowe generowały średnio 6,5 dodatkowych struktur zdania, zaś uwzględnienie zmian słów przy zachowaniu znaczenia (jak w powyższym przykładzie) podniosło tę liczbę do 215. Opracowany system odpowiadał prawidłowo na 80 ze 197 pytań ze zbioru CLEF 2004.

[8] Zobacz słownik pojęć w Dodatku.

2.3.3. Dopasowywanie semantyczne

O dopasowywaniu semantycznym można mówić wtedy, gdy przy porównywaniu zdań bierze się pod uwagę nie tylko same słowa i relacje składniowe między nimi, ale i ich znaczenie. Aby było to możliwe, konieczne jest użycie zasobów zewnętrznych. Podstawową informację o możliwych różnych znaczeniach słów z danego języka naturalnego[9] zawierają bazy typu WordNet, np. *Princeton WordNet* (Fellbaum, 1998), które gromadzą słowa w *synsety*[10], zawierające znaczenia słów reprezentujące to samo pojęcie. Zbiory te połączone są relacjami różnego typu, zgodnymi z relacjami między pojęciami im odpowiadającymi. Przykładowo, synset zawierający słowa *śmigłowiec* i *helikopter* połączony jest relacją hiperonimii z synsetem odpowiadającym pojęciu *pojazd*. Jak można wykorzystać tę wiedzę w odpowiadaniu na pytania? Jeśli pytanie dotyczy osoby, która wsiadła do helikoptera w określonych okolicznościach, a w tekście pojawia się w tym miejscu śmigłowiec lub pojazd, to możemy uznać, że zachodzi dopasowanie, pomimo, że brak całkowitej zgodności na poziomie słów (tak działa przykładowo miara podobieństwa zdań, którą zastosowali Boni i Manandhar (2003)).

Użycie WordNetu do ustalania podobieństwa zdań w korpusie znajdziemy też we wspominanej już pracy (Harabagiu i inni, 2001). Opisany tam system podejmuje próbę uzgodnienia uproszczonych drzew rozbioru pytania i kandydującej odpowiedzi. Dwa węzły drzewa zostają dopasowane nie tylko wtedy, gdy są dosłownie równe, ale także wtedy, gdy nie jest puste przecięcie zbiorów zawierających ich synonimy, hiperonimy i formy pochodne. Dopuszcza się również zmianę struktury zdania, analogicznie do systemu *InSicht* omawianego wyżej.

Wspomniane w zarysie historii (podrozdział 2.1) rozwiązania bazujące na semantyce formalnej, tj. poszukujące uzgodnienia pomiędzy formułami logicznymi reprezentującymi pytanie i bazę wiedzy, pojawiają się także współcześnie. Przykład takiego rozwiązania znajdziemy w systemie *COGEX* (Moldovan i inni, 2003). Aby przekształcić pytania na formułę logiczną, najpierw przeprowadzanie jest głębokie parsowanie, w wyniku którego powstaje pełne drzewo rozbioru. Proces należy do nurtu dopasowywania strukturalnego, ale w tym rozwiązaniu stanowi jedynie etap przejściowy. W dalszej kolejności uzyskana struktura zostaje przekształcona do postaci ciągu predykatów, ewentualnie poprzedzonych kwantyfikatorami. Poszukiwanie odpowiedzi odbywa się z użyciem programu do dowodzenia twierdzeń. Aby było możliwe uzgodnienie logicznych reprezentacji pytania i odpowiedzi, system wyposażono w dużą liczbę aksjomatów (w postaci formuł logicznych), informujących o faktach takich jak np. logiczna równoważność formy czynnej i biernej. Tak głęboka analiza okazała się konieczna tylko dla części pytań – 98 z 415 z analizowanego zbioru TREC-2002 wymagało użycia opisanych tu metod.

PowerAnswer (Moldovan i inni, 2006), stanowiący rozwinięcie systemu *COGEX*, oprócz formuł logicznych wykorzystuje również zależności semantyczne między słowami w dopasowywanych zdaniach. Poza opisanymi wyżej relacja-

[9] Inny przykład zasobów zawierających informacje semantyczne stanowią ontologie skupiające się na nazwach własnych, przedstawione w podrozdziale 4.1.

[10] Zobacz słownik pojęć w Dodatku.

mi między synsetami, bierze pod uwagę również krótkie ich definicje, obecne w angielskim WordNecie. Kolejne źródło danych wykorzystywanych przy przekształceniach formuł to łańcuchy leksykalne, przedstawione we wcześniejszej pracy (Moldovan i Novischi, 2002). Pomysł ten polega na uwzględnieniu wszystkich relacji pomiędzy rozważanymi synsetami, a nie samej hiperonimii. Miarą ich powiązania znaczeniowego będzie zatem długość najkrótszej ścieżki, łączącej odpowiadające im wierzchołki w grafie relacji. Rozwiązanie to omówiono szerzej w punkcie 3.5.2.

Inny sposób na wykorzystanie informacji o znaczeniu stanowią ramy semantyczne, które zastosowali Moreda i inni (2011). Ramy semantyczne określają role poszczególnych składników zdania, takie jak wykonawca czynności (np. *strzelec*), przedmiot czynności (np. *pocisk*), sposób wykonywania (np. *głośno*), określenie czasu (np. *po chwili*) lub przestrzeni (np. *zza rzeki*). Istnieje wiele różnych zestawów ról – autorzy cytowanej pracy wybrali *PropBank* (Palmer i inni, 2005). W każdym ze zdań w analizowanych dokumentach zostają oznaczone role semantyczne, po czym stosuje się odpowiednie reguły, określające role pełnione przez odpowiedź na podstawie typu pytania. Przykładowo, dla pytania typu *Kiedy ... ?*, poszukujemy składników pełniących rolę *AM-LOC* (położenie). Testowy zbiór pytań powstał w oparciu o zestawy TREC-8 i TREC-9, przy czym rozbudowano je o ręcznie przygotowane pytania niedotyczące nazw własnych. Dzięki zastosowaniu ról semantycznych udało się odnaleźć również odpowiedzi niebędące nazwami własnymi, co podniosło pokrycie zbioru testowego z 47% (w wersji bazującej na nazwach własnych) do 65%.

Do systemów z dopasowywaniem semantycznym należy również prezentowany w niniejszej pracy RAFAEL. W odróżnieniu jednak od większości rozwiązań opisanych poprzednio, analiza znaczenia nie jest stosowana do oceny podobieństwa kontekstu. Zamiast tego sprawdza się znaczenie samej odpowiedzi, aby ocenić czy jest ona zgodna z ograniczeniami wyrażonymi w pytaniu. Opis tej techniki, tj. głębokiego rozpoznawania nazw, przedstawiony jest w rozdziale 4.

2.3.4. Dopasowywanie statystyczne

Ostatnia kategoria systemów QA wyróżniona ze względu na sposób dopasowywania zdań wiąże się z użyciem narzędzi statystycznych. W tym podejściu budowany jest zbiór cech, reprezentujących m.in. wszystkie przesłanki świadczące o podobieństwie zdań. Może to być np. długość zdania, obecność nazw własnych, własności gramatyczne itp. Jeśli dysponujemy pewną grupą pytań wraz z oczekiwanymi odpowiedziami, to możemy ich użyć w charakterze danych treningowych dla narzędzi statystycznych. Analiza taka może wskazać nie tylko najlepszą odpowiedź na nowe pytanie, ale także najistotniejsze spośród zaproponowanych cech.

Przykładowo, Ittycheriah (2007) wykorzystuje modele z maksymalną entropią. Oznacza to, że w przypadku, gdy brakuje nam wiedzy, wypełniamy braki w taki sposób, by zachować maksymalną entropię modelowanego rozkładu. Dane treningowe pochodziły m.in. z baz TREC-8 i TREC-9. Podobny moduł odnajdziemy również w systemie *JAVELIN QA* (Ko i inni, 2007), gdzie

wartości poszczególnych miar podobieństwa odpowiadają cechom, a model regresji logistycznej zwraca prawdopodobieństwo, że analizowane zdanie stanowi odpowiedź. Do trenowania użyto 1760 pytań z baz TREC-8-12.

Inne niż przedstawione powyżej zastosowanie statystyki do odpowiadania na pytania może polegać na wnioskowaniu o zależnościach semantycznych na podstawie rozkładu wystąpień słów w korpusie. Rozwiązanie takie przedstawili Yih i inni (2013), zaliczając je do miar podobieństwa słów z uwzględnieniem ich znaczenia, stosowanych w QA. Miara ta opiera się o obserwację otoczeń wystąpień określonych słów w dużym korpusie. Im bardziej podobny skład tych otoczeń, tym bliższe powiązanie rozważanych słów. Dodatkowo, do wydobywania relacji hiperonimii oprócz WordNetu (nie zawierającego nazw własnych) wykorzystano bazę *Probase* (Wu i inni, 2012). *Probase* jest ontologią zawierającą aż 2,7 miliona pojęć.

Elementy podejścia statystycznego znajdują się również w architekturze systemu *IBM Watson* (Ferrucci i inni, 2010), którą szerzej opisano w podrozdziale 3.1.

2.4. Rozwiązania dla języków słowiańskich

Przed omówieniem budowy systemu RAFEL warto przedstawić jeszcze jedno spojrzenie na istniejące rozwiązania, tym razem pod kątem języka analizowanych tekstów. Wprawdzie żaden z systemów wymienionych poniżej nie realizuje zadania, które zdefiniowano w podrozdziale 2.2, ale ich twórcy musieli zmierzyć się z trudnościami podobnymi do tych, które pojawiły się przy budowie omawianego tu systemu. Opracowano go bowiem dla języka polskiego, w którym szyk zdania jest dość swobodny, zaś dodatkową informację o rolach słów przekazuje ich forma fleksyjna. Cecha ta jest zasadniczo wspólna dla całej rodziny języków słowiańskich (Przepiórkowski, 2007).

Pierwszy polski system QA przedstawiono w 1985 roku, kiedy Vetulani (1988) opracował polskojęzyczny interfejs do bazy *ORBIS*, zawierającej informacje o układzie słonecznym. Baza składała się z reguł w języku *PROLOG*. Rolą powstałego modułu o nazwie *POLINT* było przekształcenie pytania w języku naturalnym na odpowiednie zapytanie. Kolejny wczesny polski system operujący na tekstach o ograniczonej tematyce powstał w 2002 roku (Duclaye i inni, 2002) i odpowiadał na pytania z zakresu informacji biznesowej. Warty uwagi jest system *POLINT-112-SMS* (Vetulani i inni, 2010), którego celem jest analiza treści wiadomości SMS w celu pozyskiwania z nich informacji o zagrożeniach. Opracowany system jest zdolny do przyjmowania bardzo wielu takich wiadomości i łączenia zawartych w nich informacji w celu uzyskania kompleksowej wiedzy o zaistniałej sytuacji. *POLINT-112-SMS* zawiera również moduł QA, ale obsługuje on jedynie pytania z dziedziny zastosowania systemu, czyli bezpieczeństwa publicznego.

Bliżej obszaru badań niniejszej pracy leżą systemy dopuszczające zadawanie pytań z dowolnej dziedziny. Zbiory testowe dla takich rozwiązań często powstają poprzez przetłumaczenie ogólnodostępnych baz, np. pochodzących z konferencji TREC. Takie rozwiązania dla języka bułgarskiego pod nazwą *Socrates* przedstawił Tanev (2004). Ograniczył się on do następujących rodzajów

pytań: o definicje, położenie w czasie i przestrzeni. Przetwarzanie przebiega w trzech standardowych krokach: analiza pytania, wyszukiwanie dokumentów i ekstrakcja odpowiedzi. Analiza pytania polega jedynie na oznaczeniu części mowy – określenie typu pytania nie jest konieczne, ponieważ od użytkownika wymaga się, by podał, do której z kategorii ono należy. Dokumenty pochodzą z dwóch źródeł: bułgarsko-językowej bazy encyklopedycznej *VIDA* i wyszukiwarki *Google*, przy czym wykorzystuje się tylko fragmenty wskazane na liście wyników, bez otwierania stron. Do odnajdywania zdań zgodnych z pytaniem używa się ręcznie przygotowanych wzorców, zaś ranking odpowiedzi powstaje przez zliczanie powtarzających się słów we wszystkich dopasowanych fragmentach. Nie rozpoznaje się zatem nazw własnych ani nie uwzględnia informacji semantycznej. Ewaluację przeprowadzono na 100 pytaniach ze zbioru TREC 2001, przetłumaczonych na bułgarski. Dla 52 pytań udało się znaleźć prawidłową odpowiedź wśród pierwszych 5 w rankingu. Wskaźnik MRR (ang. *Mean Reciprocal Rank*), czyli średnia odwrotność pozycji prawidłowej odpowiedzi[11] wyniósł 0,433.

Inny system dla języka bułgarskiego, *BulQA* (Simov i Osenova, 2005), podlegał testom na konferencji CLEF 2005. W początkowym etapie przetwarzania rozwiązanie to przypomina RAFAELa – korpus tekstowy jest znakowany na poziomie morfosyntaktycznym i grup składniowych, a analiza pytania uwzględnia zarówno zaimek pytajny, jak i centrum pytania (nazwane *head word*). Różnice pojawiają się na etapie odnajdywania odpowiedzi, gdzie wykorzystano gramatyki częściowe, pełniące funkcję wzorców przypisanych do poszczególnych typów pytań. Wydajność systemu okazała się dosyć niska – w trakcie ewaluacji skierowano do systemu 200 pytań i otrzymano 37 prawidłowych odpowiedzi, 16 błędnych i 3 niedokładne. Poprzednio ten sam zespół brał udział w CLEF 2004, na którym zadawano pytania po bułgarsku do anglojęzycznego tekstu (Osenova i inni, 2004). Doświadczenie to doprowadziło do wydzielenia modułu tzw. *interfejsu*, który skupia wszystkie elementy systemu zależne od języka korpusu tekstów. Dzięki temu można łatwo wprowadzić obsługę nowego języka naturalnego.

Także dla języka słoweńskiego opracowano system QA (Čeh i Ojsteršek, 2009). Realizowane przez niego zadanie polega na odpowiadaniu na pytania studentów, dotyczące funkcjonowania wybranego wydziału uczelni, na podstawie baz danych, arkuszy kalkulacyjnych i usług sieciowych. Obserwacja pytań zadawanych przez użytkowników z użyciem dotychczas dostępnych technik (fora, e-maile) wskazała na pewne problemy. Wyodrębniono trzy kategorie pytań: krótkie i zwięzłe zapytania, pytania wielokrotne (odnoszące się do tej samej kwestii, ale różniące się informacją dodatkową) i opisowe (zawierające szczegółowy opis problemu). Ponadto zaobserwowano różne problemy charakterystyczne dla języka kolokwialnego: błędy ortograficzne, słowa obcojęzyczne, emotikony, opuszczanie diakrytów, braki w interpunkcji, pojawianie się skrótów i skrótowców. Czasem informacja zawarta w pytaniu była niewystarczająca do udzielenia odpowiedzi, a czasem niepotrzebnie powtórzona.

Pytania podzielono na 6 typów ogólnych (jednostka, miejsce, człowiek,

[11] Jeśli prawidłowa odpowiedź jest na pozycji k w rankingu, to jej odwrotność (ang. *reciprocal rank*) wynosi $\frac{1}{k}$. MRR powstaje przez uśrednienie tych wartości dla wszystkich pytań.

liczba, opis, skrót) i 31 szczegółowych. Typ określa się metodą wykorzystującą uczenie maszynowe, po czym na jego podstawie powstają możliwe inne sformułowania pytania oraz oczekiwane wzorce odpowiedzi. Rozwiązywany tu problem różni się od omawianych wcześniej zarówno ograniczoną dziedziną (kwestie związane z konkretnym wydziałem uczelni), jak i nietekstowymi źródłami wiedzy. Niestety, w cytowanej pracy nie przedstawiono żadnej ilościowej oceny wydajności.

Walas i Jassem (2010) przedstawili polski system o nazwie *Hipisek*. Akceptuje on jedynie pytania o czas, miejsce i osobę. W wyniku analizy pytania powstaje struktura nazwana *QQuery*, odpowiadająca modelowi pytania w RAFAELu, zawierająca typ, przedmiot pytania, czynność tego przedmiotu, oraz okoliczności. Analiza wykorzystuje dwie techniki: zbiór reguł w formalizmie parsera powierzchniowego *Spejd* (Przepiórkowski, 2008) i procedury heurystyczne. Także wybór zdania stanowiącego odpowiedź bazuje na funkcji heurystycznej. Bierze ona pod uwagę przedmiot pytania (pole `TOPIC` w *QQuery*) i okoliczności (pole `CONSTRAINTS` w *QQuery*), dla których sprawdzane jest ich wystąpienie w zdaniu i jego najbliższym sąsiedztwie. Funkcja ta wykorzystuje wagi ustalone ręcznie na podstawie doświadczeń. Głównym przedmiotem pracy jest zbadanie wpływu włączenia rozpoznawania nazw własnych do przebiegu algorytmu. Oznacza to, że w zdaniu odpowiadającym na pytanie określonego typu musi występować nazwa własna tego samego typu. Warta uwagi jest procedura ewaluacji, gdyż oczekiwane odpowiedzi na pytania podano w formie wyrażeń regularnych, co pozwala na częściowe rozwiązanie problemu różnic w sformułowaniach (zob. automatyczną ewaluację w podrozdziale 5.2). Eksperymenty przeprowadzone na zbiorze 65 pytań wykazały, że liczba poprawnych odpowiedzi wzrosła po dodaniu obsługi nazw własnych z 19 na 24. Niestety, bardzo mała wielkość danych testowych oraz niezdefiniowana baza wiedzy (zebrana przez robota internetowego) czynią porównanie rezultatów bardzo trudnym. W późniejszych pracach (Walas i Jassem, 2011; Walas, 2012) autorzy skupili się się na wykorzystaniu rozumowania przestrzennego do odpowiadania na pytania typu tak/nie; ich rozwiązanie szerzej omówiono w sekcji 2.5.3. Tematyka ta została rozwinięta również w rozprawie doktorskiej (Walas, 2014).

Rozwiązanie, które przedstawili Piechociński i Mykowiecka (2005) (szerszy opis we wcześniejszej pracy magisterskiej (Piechociński, 2005)) leży najbliżej zakresu niniejszej rozprawy, ponieważ zawiera analizę treści Wikipedii, a ewaluacja wykorzystuje przetłumaczone pytania z bazy TREC. Główna różnica wynika z faktu, że system ten silnie opiera się na strukturze Wikipedii, a wiec nie można go wykorzystać do dowolnego korpusu tekstowego. Pierwszy krok analizy pytania ma na celu odgadnięcie tytułu wpisu w encyklopedii, w treści którego będzie poszukiwana odpowiedź. Zakłada się, że tę rolę może pełnić pojawiająca się w pytaniu nazwa własna, przy czym za taką uznaje się pierwszą napotkaną sekwencję słów zaczynających się od wielkich liter. Jeśli takiej brakuje, zastępuje ją sekwencja nominalna wybrana według heurystyki bazującej na oczekiwanej kolejności słów w zdaniu. Wskazany w ten sposób artykuł Wikipedii dzielony jest na zdania i każdemu z nich przypisuje się wynik punktowy wynikający z pojawienia się w nim słów z pytania. Stosuje się również ręcznie opracowany słownik, wskazujący możliwe zamienniki słów,

np. *napisał* i *dzieło*. Dwa typy pytań, dotyczące daty i miejsca urodzin i śmierci, obsługiwane są przez specjalne reguły. W procesie ewaluacji zadano 200 pytań pobranych ze zbioru TREC-8 i przetłumaczonych na polski. Spośród nich 29 otrzymało prawidłowe odpowiedzi.

Niestandardowe podejście do wzorców odpowiedzi przedstawiono w czeskim otwarto-dziedzinowym systemie QA (Konopík i Rohlík, 2010). Wykorzystano tam zbiór wzorców przypisanych do poszczególnych typów pytań, ale zaproponowano również ich półautomatyczne wydobywanie z wyników wyszukiwania. Uczenie rozpoczyna się od zbioru par pytanie-odpowiedź tego samego typu, np. dotyczących daty śmierci. Pytanie przekształca się na zapytanie do wyszukiwarki, przy czym uwzględnia się zastępowanie słów ich synonimami. Uzyskane wyniki mogą posłużyć do wydobycia nowych wzorców, te zaś do odnalezienia nowych par pytanie-odpowiedź tego samego typu. Wzorce służące do rozpoznawania typu pytania nie podlegają uczeniu. Ten schemat postępowania bardzo przypomina *bootstraping*, który zastosowali Ravichandran i Hovy (2002) (omówienie w punkcie 2.3.1). Ewaluacja polegała na ręcznym przygotowaniu 100 pytań testowych i sprawdzeniu uzyskanych odpowiedzi, spośród których 64% było prawidłowych, 7% częściowo prawidłowych, a 29% błędnych.

Peshterliev i Koychev (2011) w ich rozwiązaniu dla bułgarskiego skoncentrowali się na semantycznym dopasowywaniu pytań i odpowiedzi z użyciem parsowania zależnościowego. Autorzy zwracają uwagę na sytuacje, gdy modele dopasowania ignorujące strukturę zdania zawodzą. Przykładowo, rozważmy pytanie *Kto napisał Kandyda?*. Jeśli weźmiemy pod uwagę reprezentację *bag of words*[12] z użyciem lematyzacji[13], to zdania *Wolter napisał Kandyda* i *Kandyd napisał list do Kunegundy* wydają się być tak samo użyteczne jako odpowiedź, a do tego oba zawierają nazwę bytu zgodnego z typem pytania, tj. osoby. Tymczasem drugie z nich różni się zasadniczo od pytania ze względu na fakt, że *Kandyd* jest w nim podmiotem, a nie dopełnieniem. Aby rozwiązać ten problem, autorzy przeprowadzają analizę zależnościową zarówno pytania, jak i potencjalnych odpowiedzi i rozszerzają wymagania zgodności słów o zgodność ich ról gramatycznych. W tym przykładzie oznacza to, że akceptowane będą tylko te zdania, w których *Kandyd* jest dopełnieniem, a poszukiwane określenie osoby występuje jako podmiot. W wyniku ewaluacji, w której wykorzystano 100 pytań opracowanych na podstawie zbioru TREC 2007, okazało się, że dodanie omawianego dopasowywania struktury podnosi precyzję o 9,3%.

W najnowszej publikacji z zakresu polskiego QA Marcińczuk i inni (2013b) przedstawili opis prac nad zadaniem analogicznym do badanego tutaj, jednak za odpowiedź uznaje się tam pojedynczy dokument, więc ocenie podlega jedynie precyzja wyszukiwania. Odzwierciedlają to miary: a@n, czyli udział pytań, dla których prawidłowa odpowiedź znajduje się wśród pierwszych n wyników i MRR. Poziom odniesienia w pracy stanowią wyniki wyszukiwania powstałe przez przekształcenie pytania do zapytania do systemu wyszu-

[12] Reprezentacja tekstu nazywana *bag of words* (ang. worek słów) polega na tym, że bierze się pod uwagę tylko liczbę wystąpień poszczególnych słów, ignorując ich kolejność.

[13] Lematyzacja oznacza konwersję słów do lematów, czyli form bazowych (zobacz słownik pojęć w Dodatku).

kiwania *Apache Solr*[14], bazującego na *Lucene*, i użycie domyślnego schematu rangowania. Pierwsze usprawnienie, nazwane MCSW (ang. *Maximum Cosine Similarity Weighting*), polega na zastosowaniu tego samego schematu, ale do k-zdaniowych fragmentów. Wynik przypisywany do dokumentu to maksimum z wyników składających się na niego fragmentów. Drugie podejście o nazwie MSW (ang. *Minimal Span Weighting*) wykorzystuje w funkcji rangującej długość najkrótszego fragmentu zawierającego wszystkie słowa z zapytania. Obie te techniki mają na celu to samo: premiowanie tych dokumentów, w których poszukiwane słowa pojawiają się blisko siebie. Może to oznaczać bowiem, że te wystąpienia mają ze sobą jakiś związek – tak, jak to dzieje się w pytaniu. Ewaluacja na wspominanym już zbiorze pytań typu *Czy wiesz . . . ?* (Marcińczuk i inni, 2013a) wykazała, że najlepsze rezultaty daje technika MCSW, podnosząc wskaźnik a@1 z 26,09% (poziom odniesienia) do 38,63%. Zysk maleje w miarę zwiększania n, a dla a@200 wyniki są takie same we wszystkich rozwiązaniach (87,29%).

Ponieważ dla niektórych języków słowiańskich brakuje ważnych narzędzi lingwistycznych, publikuje się także wyniki rozwiązań częściowych, obejmujących tylko część zadań systemu QA, jak np. wyszukiwanie dokumentów dla macedońskiego (Armenska i inni, 2010), klasyfikacja pytań dla chorwackiego (Lombarović i inni, 2011) czy ocena odpowiedzi dla rosyjskiego (Solovyev, 2013).

2.5. Inne problemy

W podrozdziale 2.2 wyznaczono granice zadania, które jest przedmiotem rozważań w niniejszej pracy, jednak część problemów związanych z systemami QA znajdujących się poza tymi granicami jest tak interesująca, że całkowite ich pominięcie czyniłoby niniejszą pracę niepełną. Dodatkowo niektóre z rozwiązań zaproponowanych w systemach realizujących zadanie QA w odmiennej wersji zostały wykorzystane w systemie RAFAEL, stąd odniesienia do nich pojawią się również w kolejnych rozdziałach. W niniejszym podrozdziale pokrótce przedstawiono takie wybrane problemy i przykładowe systemy, stanowiące próby ich rozwiązania.

2.5.1. Sieć WWW

Systemy QA wykorzystujące jako źródło danych sieć WWW (ang. *web-based QA*) są istotnie różne od tych korzystających z bazy tekstowej (ang. *textual QA* lub *text-based QA*), których przykład stanowi RAFEL. Jedna z głównych cech charakterystycznych problemu odpowiadania na pytania na podstawie danych tekstowych to redundantność języka, tj. istnienie wielu form zapisu tej samej informacji. Oznacza to, że dla danego typu pytania (np. o datę śmierci wskazanej osoby) nie zawsze możemy liczyć na to, że zawierający tę informację tekst będzie miał formę pasującą do sformułowanego schematu (np. *X Y zmarł w roku . . .*). Jednak gdy zbiór tekstów jest bardzo duży, popularne fakty zapewne znajdują się tam wielokrotnie i szansa na trafienie na oczekiwane wyrażenie

[14] `http://lucene.apache.org/solr/`

rośnie. Opracowanie metody znalezienia odpowiedzi na pytanie o taki fakt jest zatem znacznie łatwiejsze.

W systemach przeszukujących sieć WWW często wykorzystuje się dopasowywanie wzorców – prostą, ale wystarczająco wydajną technikę szukania fragmentu z potencjalną odpowiedzią. Za przykład może posłużyć praca, w której Ravichandran i Hovy (2002) wykorzystują wzorce do odpowiadania na pytania z bazy TREC-10. Co interesujące, opracowany system jest zdolny do uczenia się nowych wzorców na podstawie obserwacji wyrażeń, w jakich pojawiają się w tekście nazwy bytów zidentyfikowane jako odpowiedzi (szerszy opis w punkcie 2.3.1). Autorzy próbowali zastosować tę technikę do zamkniętej bazy tekstów (z tego samego zadania TREC), jednak osiągnęli znacznie gorsze rezultaty, gdyż opracowane wzorce powtarzały się tam rzadziej.

Praca zespołu badawczego firmy *Microsoft* (Brill i inni, 2002) opisująca rozwiązanie pod nazwą *AskMSR* pokazuje, jak prosty może być system QA, jeśli wykorzystuje zewnętrzną wyszukiwarkę WWW (do tego celu użyto *Google*). W systemie tym pytanie zostaje przekształcone na zbiór słów kluczowych skierowanych do wyszukiwarki. Fragmenty prezentowane na liście wyników (treść strony nie jest brana pod uwagę) podlegają filtrowaniu, a następnie zliczane są n-gramy[15] w celu wskazania tych najczęstszych, potencjalnie stanowiących odpowiedź. Oczywiście, często ta odpowiedź jest nieprawidłowa, dlatego autorzy stosują drzewo decyzyjne do szacowania oczekiwanej poprawności wyniku. Drzewo powstało z użyciem następujących cech pytania: unigramów i bigramów słów, długości pytania, liczby wielkich liter, liczby *stop-words*[16], ich udziału procentowego w w zbiorze słów i długości najdłuższego słowa. Współczynnik korelacji pomiędzy zaobserwowaną poprawnością odpowiedzi a szacunkami drzewa decyzyjnego wyniósł 0,363. Wykorzystanie *Google* zamiast budowy własnego indeksu to popularne rozwiązanie, można je często odnaleźć w systemach obsługujących języki mniej bogate w zasoby, jak czeski (Konopík i Rohlík, 2010) czy bułgarski (Tanev, 2004), ale nie tylko (zob. np. Buchholz i Daelemans (2001) czy Paşca (2002)). Na marginesie warto wspomnieć, że zdarzają się również nieszablonowe zastosowania wyszukiwarki *Google* – przykładowo Ahn i inni (2004) stosują ją do przesiewania zbioru kandydatów na odpowiedzi, uznając frazy, które skutkują pustą listą wyników wyszukiwania, za bezsensowne i niewarte dalszego przetwarzania.

Ciekawe wykorzystanie sieci WWW, które jednak jest zgodne z zasadami zadania TREC, przedstawili Katz i inni (2003). W przypadku pytań o proste fakty przeszukują oni sieć WWW, a następnie wykonują rzutowanie otrzymanej odpowiedzi (ang. *answer projection*) na korpus wskazany w zadaniu (tu AQUAINT). Ma to na celu znalezienie dokumentu, który mógłby posłużyć jako źródło. W ten sposób wykorzystują redundancję sieci WWW, spełniając jednocześnie wymogi zadania, które zaprojektowano jako bazujące na korpusie tekstowym.

Podejście polegające na przekształcaniu dokumentów do zbioru trójek postaci <obiekt_1 relacja obiekt_2>, przedstawione przez Katza (1997) i zre-

[15] Zobacz słownik pojęć w Dodatku.

[16] *Stop-words* są to słowa tak powszechne, że zazwyczaj pomija się je przy wyszukiwaniu, np. *jest*, *i*, *też* itp.

ferowane w zarysie historii (podrozdział 2.1) znalazło zastosowanie także niedawno. Fader i inni (2013) do odpowiadania na pytania używają bazy 600.000 takich trójek, powstałej przez zastosowanie algorytmu *ReVerb* (Fader i inni, 2011) do korpusu *ClueWeb09*, zawierającego ponad miliard stron WWW z 2009 roku. Także tu zasady konwersji pytania na spodziewane schematy trójek są uzyskiwane w procesie uczenia się. Autorzy zwracają jednak uwagę na fakt, że używane relacje są zaszumione, nieznormalizowane i niejednoznaczne, co utrudnia udzielanie prawidłowych odpowiedzi na ich podstawie. System osiągnął pokrycie na poziomie 42% i precyzję 77%, ale do ewaluacji użyto tylko tych pytań, dla których odpowiedź na pewno znajduje się w bazie trójek.

W przypadku systemów korzystających z zewnętrznej wyszukiwarki, istotną rolę odgrywa zakres analizowanej treści. Zazwyczaj wyniki wyszukiwania mają postać listy odnalezionych stron wraz z krótkimi ich fragmentami, zawierającymi poszukiwane słowa. Często do odnajdywania odpowiedzi wykorzystuje się właśnie te fragmenty, rezygnując z reszty treści dokumentów. Podejście to posiada pewne zalety: nie ma konieczności przetwarzania złożonego pliku HTML ani wyodrębniania z całości tekstu zdań istotnych z punktu widzenia pytania. Z drugiej strony, przytoczone fragmenty są na tyle krótkie, że mogą nie zawierać pełnych zdań. Stanowi to problem, gdy brana jest pod uwagę struktura zdania, np. w przypadku opisywanego w punkcie 2.3.3 wykorzystania ról semantycznych (Moreda i inni, 2011). Z tego powodu dla wspomnianego systemu opracowano rozszerzenie, polegające na zastąpieniu fragmentów dokumentów ich streszczeniami(Lloret i inni, 2011). Składają się one z kilku pełnych zdań z treści strony, najlepiej odzwierciedlających jej treść i najbliższych tematycznie do zapytania. Ewaluację przeprowadzono na tym samym, co w poprzedniej pracy, zbiorze testowym. Zgodnie z oczekiwaniami, omawiane usprawnienie wpłynęło na znaczne polepszenie wyników: pokrycia z 29% na 45% i precyzji z 49% na 73%.

2.5.2. Inne typy pytań

Znalezienie w sposób automatyczny odpowiedzi na pytania, które wykraczają poza kategorię prostych faktów, to duże wyzwanie, ponieważ w tym celu często trzeba rozumieć tekst w znacznie większym stopniu niż przy samym wyszukiwaniu nazw. Dobry przykład stanowią pytania o przyczynę, rozpoczynające się zaimkiem *dlaczego* (ang. *why-questions*), na które odpowiedzi poszukują Oh i inni (2013). Jak można by się spodziewać, punkt wyjścia stanowią wyrażenia regularne rozpoznające konstrukcje typu *z powodu*, *ponieważ* itp. Rozstrzygnięcie, które z tych zdań stanowią rzeczywiste konstrukcje przyczynowe, jest zadaniem klasyfikacji kontekstowej, do rozwiązania którego wykorzystuje się warunkowe pola losowe (ang. *conditional random fields* – CRFs). Klasyfikator przyjmuje na wejściu trzy kolejne zdania, po czym oznacza słowa w nich zawarte jako opisujące przyczynę, skutek lub inne. Dzięki własnościom pól losowych w procesie trenowania klasyfikator uwzględnia również informacje o kontekście, na przykład to, że po słowie *ponieważ* następuje opis przyczyny, choć słowo to samo w sobie do niego nie należy. Budowane modele korzysta-

ją z cech opracowanych na podstawie wystąpień wzorców konstrukcji zdań przyczynowych oraz informacji morfologicznych i składniowych.

Inny rodzaj pytań w pracy dotyczącej ich klasyfikacji uwzględnił Hermjakob (2001). Określa je jako *abstract QTargets* i należą do nich przykładowo pytania o przyczyny sławy osoby lub rzeczy, rozwinięcia skrótowców, synonimy itp.

Szczególną kategorią, dla której istnieją działające systemy QA, są pytania o definicje. Przykładowo, Paşca (2002) bazuje na wyszukiwaniu wzorców, ale wykorzystuje także wnioskowanie oparte na *WordNecie*. Polega ono na pobraniu listy hiperonimów pojęcia, którego definicji poszukujemy, i wykorzystaniu jej do oceniania kandydatów na odpowiedź. Pojawienie się jednego z nich w analizowanym fragmencie wskazuje, że w istocie definiuje on pojęcie wyjściowe. Podejście bazujące jedynie na wzorcach zastosowano także w module definicyjnym systemu zaprezentowanego na konferencji TREC 2003 (Katz i inni, 2003). Odpowiadanie na pytania nie jest jedynym zastosowaniem dla poszukiwania definicji w tekście, które stanowi interesujące zadanie samo w sobie. Przykładowe rozwiązanie tego problemu dla języków słowiańskich (bułgarskiego, czeskiego i polskiego) bazuje na parsowaniu powierzchniowym (Przepiórkowski i inni, 2007). Stosunkowo niska uzyskana precyzja (23%) wskazuje na wysoki poziom trudności zadania.

Ostatnim, nietypowym, rodzajem pytań jest kategoria *OTHER* z zadania TREC. Jak wspomniano wyżej, chodzi tu o podanie nowego interesującego faktu w ramach danego bloku tematycznego, przy czym wyklucza się informacje, których dotyczą inne pytania w tym bloku. Przykład takiego pytania podano w podrozdziale 2.1 – blok dotyczy trzęsienia ziemi w Pakistanie w 2005 roku i odpowiadając na pytanie w kategorii *OTHER* należy przedstawić fakt dotyczący tego zdarzenia. Opis jednego z możliwych podejść do tego problemu zawiera np. praca przedstawiająca system *QUARTZ* (Ahn i inni, 2004). Kluczową rolę odgrywa tu oszacowanie istotności wydobytych faktów. Do tego celu wykorzystuje się zewnętrzny korpus (Wikipedia). Fakty, które mają podobne odpowiedniki w obu bazach tekstów (źródłowej i dodatkowej) uznaje się za ważne. Aby oszacować to podobieństwo, przedstawia się oba fragmenty jako zbiory słów i oblicza miarę podobieństwa Jaccarda (1901).

2.5.3. Nietekstowe bazy wiedzy

Źródłem wiedzy, z którego korzysta system QA, nie zawsze jest zbiór tekstów w języku naturalnym. Znacznie łatwiej użyć sformalizowanej postaci przechowywania wiedzy, która jest prostsza w obsłudze przez automatyczny program. W takiej sytuacji komponent analizy języka ogranicza się do interpretacji pytania i jego przetłumaczenia do zapytania w odpowiedniej formie. Podejście takie cieszyło się popularnością zwłaszcza w początkach rozwoju dziedziny. Najlepszym przykładem jest pierwszy system QA – *BASEBALL* (Green i inni, 1961), gdzie baza wiedzy ma postać hierarchicznej listy par klucz-wartość, dopasowanej do wybranej wąskiej dziedziny tematycznej. Również pierwsze polskie rozwiązanie QA należy do tej kategorii (Vetulani, 1988). Tu także mamy do czynienia z ograniczoną dziedziną (układ słoneczny), zaś za format przechowywania danych posłużyły reguły języka *PROLOG*.

W pewnych sytuacjach szczególna dziedzina pytań narzuca również nietypową postać bazy wiedzy. Przykładem takiej sytuacji jest realizacja zadania odpowiadania na pytania w języku polskim opisana w pracy (Walas, 2012). Pytania te dotyczą okoliczności przestrzennych określonych wydarzeń. Jako źródło danych wykorzystano tu bazę *RCC-5* (Walas i Jassem, 2011), przechowującą zbiór lokalizacji geograficznych, jak również związków między nimi, takich jak zawieranie, nakładanie się czy równość. Dzięki temu zbudowany system QA jest w stanie przy odpowiadaniu na pytania posiłkować się wnioskowaniem o zależnościach przestrzennych. Przykładowo, jeśli pytanie brzmi *Czy X urodził się w Polsce?*, a wiemy, że X urodził się w Krakowie, to istnienie w bazie odpowiednich zależności pomiędzy Polską i Krakowem umożliwia udzielenie odpowiedzi twierdzącej.

Innym podejściem, zastosowanym w pracy (Atzeni i inni, 2004), jest stworzenie ontologii, zawierającej w ściśle określonej formie całą wiedzę, która może być przedmiotem pytań. Oczywiście, zadanie takie jest wykonalne tylko w przypadku bardzo ograniczonej dziedziny – w tym wypadku oparto się na zawartości stron internetowych kilku uniwersytetów. Powiązanie rozwiązanie można zobaczyć w słoweńskim systemie QA (Čeh i Ojsteršek, 2009). Podobnie jak w poprzedniej pracy, pozyskiwane informacje mogą dotyczyć funkcjonowania uczelni. Tym razem jednak zamiast ontologii korzystano z wielu heterogenicznych źródeł: dwóch odrębnych baz danych, plików *MS Excel* i usług sieciowych.

Najbardziej nietypowym rodzajem wiedzy, na którym można oprzeć się przy poszukiwaniu odpowiedzi są popularne w ostatnim czasie społecznościowe serwisy QA (ang. *community QA*). Ich działanie polega na wyświetlaniu nadesłanych pytań i zachęcaniu użytkowników do udzielania odpowiedzi. Ponieważ często pojawia się bardzo wiele reakcji na jedno zapytanie, a ich jakość różni się znacznie, główne zadanie polega na wybraniu tej, która najlepiej posłuży pytającemu. Przykładowe rozwiązanie, pracujące na archiwum *Yahoo! Answers* (Bian i inni, 2008), korzysta z bardzo wielu cech, takich jak podobieństwo leksykalne pytania i odpowiedzi, ich długości, głosy oddane na odpowiedź, a także historia poczynań odpowiadającego. Serwisy tego typu wykorzystuje się również w tradycyjnym podejściu, gdyż mogą posłużyć jako źródło pytań, np. *WikiAnswers* w (Fader i inni, 2013).

2.5.4. Rozwiązania częściowe

Ostatnią kategorią systemów rozwiązujących zadania znacząco różne od rozważanego w niniejszej pracy, a jednak zasługującą na wspomnienie, są rozwiązania częściowe, tj. realizujące tylko fragment całego systemu QA. Ponieważ pewne pomysły tam zawarte stanowiły inspirację przy opracowaniu elementów RAFAELa, ich szczegółowe omówienie znajduje się w odpowiednich rozdziałach. Można je podzielić na trzy grupy, odpowiadające trzem podzadaniom QA.

Pierwsza kategoria rozwiązań częściowych to systemy, które przyjmują na wejściu pytanie, ale w odpowiedzi ograniczają się do poziomu pierwszego według hierarchii z podrozdziału 2.2.3, czyli podania listy rankingowej dokumentów. W przeglądzie systemów QA (Andrenucci i Sneiders, 2005) zalicza się je

do dziedziny wyszukiwania informacji, gdyż można je widzieć jako dodatkowy moduł wyszukiwarki, interpretujący żądania w formie pytań. Za przykład może posłużyć praca (Marcińczuk i inni, 2013b), w której autorzy opisują modyfikację rankingu rezultatów systemu *Apache Solr*. Opis technik używanych przy realizacji tego zadania, stanowiącego jeden z etapów przedstawionego tu systemu, znajduje się w podrozdziale 3.3.2.

Innym zadaniem, stanowiącym element systemu QA, ale łatwym do wyodrębnienia, jest określenie typu pytania. Problem ten można widzieć jako przykład zadania klasyfikacji, co umożliwia również jednoznaczną ilościową ocenę wydajności. Przykłady takich rozwiązań przedstawiono w pracach (Przybyła, 2013b; Li i Roth, 2002; Hermjakob, 2001), a szczegółowe omówienie problemu umieszczono w podrozdziale 3.3.1.

Istnieją także rozwiązania stanowiące niejako dopełnienie tych wspomnianych powyżej, tzn. posiadające wszystkie elementy *oprócz* analizy pytania. Jak w tym przypadku wyglądają dane wejściowe? Za przykład może posłużyć nam system *SHAPAQA* (Buchholz i Daelemans, 2001), w którym używa się formularza HTML, składającego się z pól opatrzonych następującymi etykietami: *kto/co?*, *co zrobił?, kogo/co?*, *kiedy?*, *gdzie?*, *dlaczego?*, *jak?* i *z czym?*. Użytkownik wypełnia je słowami opisującymi jego wiedzę na temat odpowiednich aspektów pytania. Część pól można pozostawić pustymi, a w miejsce poszukiwanych danych wpisywany jest znak zapytania. System próbuje odnaleźć zdania pasujące do takiego wzorca i zwrócić szukaną informację. Przykładowo, dla zdania *Kiedy wynaleziono telefon?* otrzymujemy listę:

- co zrobił = *wynaleziono*,
- kogo/co = *telefon*,
- kiedy = ?,

zaś wszystkie pozostałe pola są puste. Odpowiedź zawiera tylko propozycję wypełnienia zawartości pola ze znakiem zapytania, czyli *W 1876 r*. Zapis ten przypomina listę specyfikacyjną z systemu *BASEBALL* (Green i inni, 1961), przedstawionego w podrozdziale 2.1, jednak tam stanowiła ona wewnętrzną reprezentację pytania, a tutaj je zastępuje.

2.6. Podsumowanie

Liczba i różnorodność przedstawionych systemów odpowiadających na pytania może robić wrażenie, jednak ich analiza prowadzi do spostrzeżenia, że osiągnięcie pozytywnych rezultatów było możliwe jedynie dzięki narzuceniu ograniczeń na rozwiązywany problem. Wciąż nie istnieje, nawet dla języka angielskiego, w pełni kompletny system QA, zdolny do odpowiadania na dowolne pytania na podstawie bazy tekstów w języku naturalnym. Wydaje się, że najbliżej tego celu znajdują się systemy, które odnosiły sukcesy na konferencjach TREC, jednak jak dotąd nie pokazano, czy radzą sobie równie dobrze poza ściśle określonym środowiskiem testowym. Próby takie podejmuje w przypadku systemu *Watson* firma IBM, przystosowując go do pracy z bazą wiedzy medycznej, jednak dotąd wiadomo niewiele o rezultatach tego wdrożenia.

W ostatnich latach łatwym do zaobserwowania trendem jest przeniesienie uwagi na rozwiązania poszukujące odpowiedzi w sieci WWW, a nie w ustalonym korpusie. Wynika to nie tylko z rosnącej ilości informacji dostępnych przez Internet, ale także z faktu, że redundantność takiego źródła czyni problemy związane z różnorodnością sformułowań mniej dokuczliwymi. Trzeba jednak pamiętać, że istnieją zastosowania, w których sieć WWW nigdy nie będzie zawierała potrzebnych informacji lub jej wiarygodność będzie niewystarczająca. Można tu wymienić wspieranie diagnostyki medycznej z uwzględnieniem informacji dotyczących poprzednio zdiagnozowanych przypadków, udzielanie informacji prawnych na podstawie odpowiednich aktów lub jakiekolwiek zastosowania baz niejawnych.

W tym świetle problem przedstawiony w niniejszym rozdziale można uważać za wciąż nierozwiązany. Przy projektowaniu zadania dla RAFAELa ustrzeżono się większości z istotnych ograniczeń omawianych systemów, pozostawiając tylko jedno, czyli udzielanie odpowiedzi w formie nazwy pojedynczego bytu. Ponieważ budowa systemu QA dla języka o bogatej składni stanowi duże wzywanie, dotąd rzadko podejmowane w literaturze, zdecydowano się na pracę z językiem polskim.

Rozdział 3

Budowa systemu RAFAEL

RAFAEL (ang. *RApid Factoidal Answer Extracting aLgorithm*) jest systemem komputerowym autonomicznie udzielającym odpowiedzi na pytania w języku polskim na podstawie bazy tekstów bez ograniczeń dziedzinowych. RAFAEL rozpoznaje wszystkie pytania o proste fakty, a odpowiada na te dotyczące nazw bytów. W poprzednim rozdziale szczegółowo przedstawiono realizowany typ zadania QA, w tym zaś znajduje się opis samego systemu. Najpierw przedstawiono schemat architektoniczny RAFAELa (podrozdział 3.1), a w kolejnych podrozdziałach wyjaśniono budowę poszczególnych modułów systemu.

Zazwyczaj przy budowie systemów QA najwięcej uwagi poświęca się problemowi dopasowywania zadanego pytania do zdań z korpusu w celu znalezienia tego, które zawiera odpowiedź (patrz podrozdział 2.3). Przy projektowaniu i budowie RAFAELa zaakcentowany został inny aspekt – wykrywanie nazw bytów, które mogą stanowić odpowiedź. Istniejący zestaw rozwiązań rozszerzono o nowe podejście, nazwane głębokim rozpoznawaniem nazw (ang. *Deep Entity Recognition – DeepER*), któremu poświęcono oddzielny rozdział 4.

3.1. Architektura

Choć istnieje ogromna różnorodność systemów QA, schemat architektoniczny większości z nich jest bardzo podobny i składa się z następujących elementów:

1. analiza pytania,
2. wyszukiwanie dokumentów podobnych tematycznie do pytania,
3. wybranie pewnej liczby dokumentów według rankingu,
4. wyszukiwanie w dokumentach zdań mogących stanowić odpowiedź,
5. wybór i zwrócenie odpowiedzi.

Przykłady takiej struktury odnajdziemy w wielu systemach (Moldovan i inni, 2000; Hovy i inni, 2000; Harabagiu i inni, 2000).

Zazwyczaj jest tak, że każdy etap wykonuje się tylko raz i efekt jego działania stanowi dane wejściowe dla następnego etapu. Od tej reguły odstąpili np. Harabagiu i inni (2001). W przedstawionym przez nich systemie wprowadzono pętle sprzężenia zwrotnego, tj. możliwość powrotu do wcześniejszego etapu, jeśli spełnione są określone warunki. Te warunki to zbyt duża lub zbyt mała liczba tekstów uzyskana po etapie wyszukiwania oraz brak zdania zgodnego z pytaniem na poziomie składniowym lub semantycznym. Wprowadzenie tych pętli poprawiło niemal dwukrotnie dokładność systemu.

W pewnych sytuacjach uzasadniona staje się modyfikacja przedstawionego schematu architektonicznego przez zrównoleglenie, tj. pobieranie odpowiedzi w kilku oddzielnych strumieniach i wybór najlepszych wyników. Podejście takie zastosowano w systemie *QUARTZ* (Jijkoun i De Rijke, 2004). Wykorzystywane tam źródła informacji to: przeszukiwanie tabel zawierających odpowiedzi na najbardziej oczywiste pytania (np. o stolicę kraju), dopasowywanie wzorców i wydobywanie n-gramów. Dodatkowo, wątki operujące na kolekcji dokumentów mogą równolegle poszukiwać odpowiedzi w różnych zbiorach (np. korpusie tekstowym i sieci WWW). Autorzy wskazali, że w przeprowadzonych eksperymentach podejście takie dało prawie 6-krotne zwiększenie pokrycia testowego zbioru pytań w porównaniu do najlepszego z pojedynczych źródeł. Dodatkowo, przetwarzanie równoległe ułatwia rozbudowę systemu poprzez dodawanie kolejnych metod lub źródeł danych, jak np. przetwarzanie Wikipedii (Ahn i inni, 2004).

Interesująco przedstawia się także architektura systemu *Watson* (Ferrucci i inni, 2010). System ten wykorzystuje wiele różnych technik analizy języka naturalnego na pięciu etapach: w analizie pytania, generowaniu hipotez, filtrowaniu, poszukiwaniu dla nich dowodów i syntezie. Jako całość *Watson* jest bardzo złożony, jednak wiodąca koncepcja wydaje się dość prosta – na każdym z etapów procesu wykorzystuje się wiele działających równolegle, redundantnych modułów. Każdy nich oprócz rezultatu działania zwraca również liczbę, odzwierciedlającą stopień pewności otrzymanego wyniku. Mechanizm wyboru pojedynczej decyzji na podstawie takiego wielogłosu wykorzystuje metody uczenia maszynowego, w szczególności meta-uczenia. Takie postępowanie nie byłoby możliwe bez dużego zbioru treningowego. Został on stworzony na podstawie zestawu około 200.000 pytań i odpowiedzi, pojawiających się w teleturnieju *Jeopardy!*, opracowanego i udostępnionego przez fanów tego programu[1].

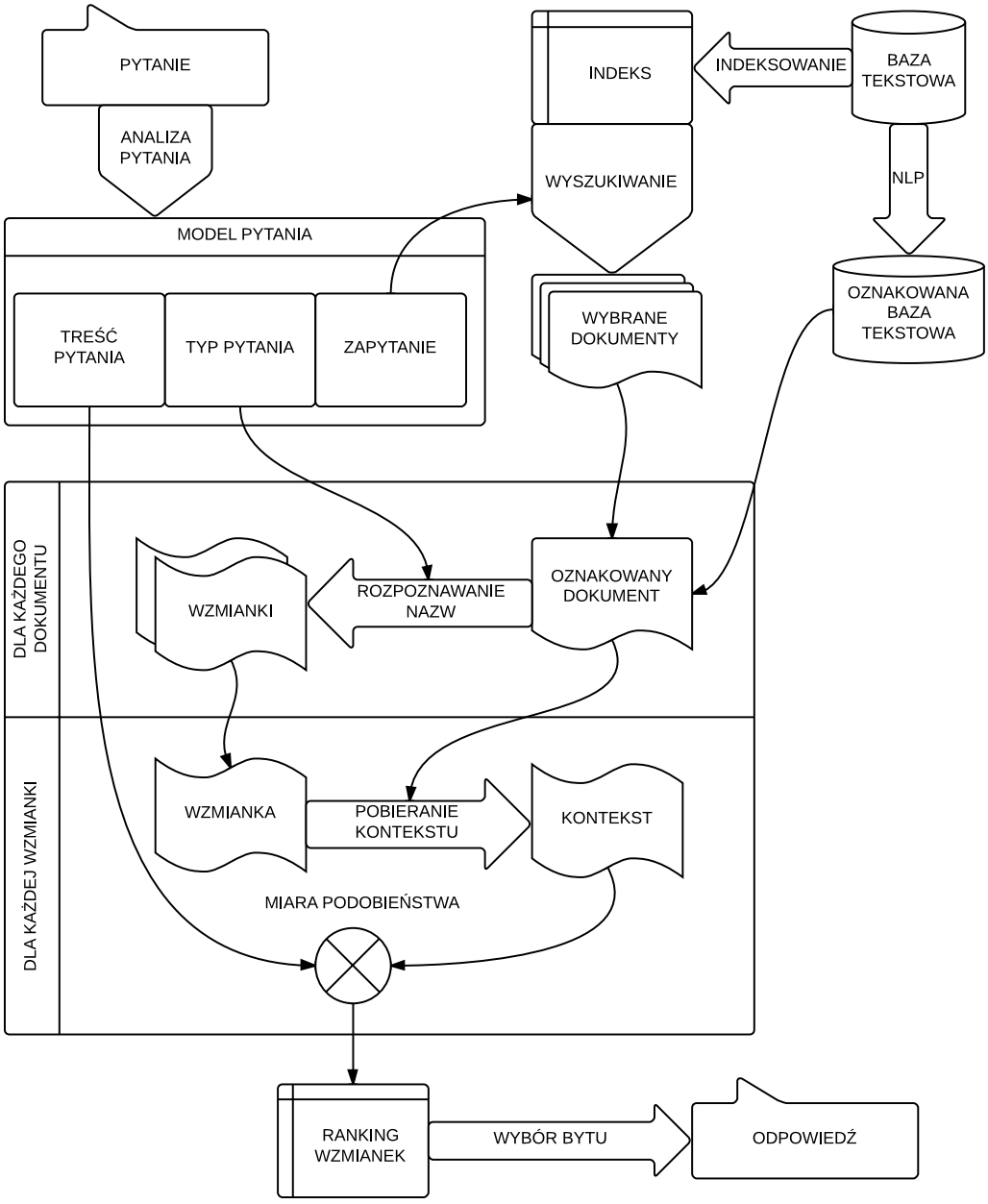
Inne ciekawe rozwiązanie w systemie *Watson* służy przyspieszeniu procesu odpowiadania. W turnieju *Jeopardy!* dużą rolę odgrywa kilkusekundowy czas zgłoszenia do odpowiedzi (tylko pierwszy zawodnik ma szansę na punkty), zaś czas przetwarzania pytania w pierwszej wersji systemu wynosił około 2 godzin. Ze względu na niezależność przetwarzania w poszczególnych wątkach procesu zdecydowano się na sprzętowe zrównoleglenie z użyciem 2500 rdzeni obliczeniowych. Ich pracą zarządza platforma *Apache UIMA*, zoptymalizowana do przetwarzania dużych ilości informacji tekstowych.

Architektura systemu RAFAEL, przedstawiona na rysunku 3.1, jest w dużym stopniu zgodna ze standardem obecnym w literaturze. Proces udzielania odpowiedzi składa się z sześciu głównych etapów:

1. przygotowanie bazy tekstowej,
2. analiza pytania,
3. wyszukiwanie dokumentów,
4. rozpoznawanie nazw,
5. ocena wzmianek[2],
6. generowanie odpowiedzi.

[1] `http://j-archive.com/`

[2] Zobacz słownik pojęć w Dodatku.



Rysunek 3.1. Architektura systemu odpowiadania na pytania RAFAEL.

Podstawę działania systemu stanowi zbiór dokumentów tekstowych. Pierwszym etapem przetwarzania danych jest przygotowanie ich do użycia poprzez utworzenie indeksu niezbędnego do późniejszego przeszukiwania. Następnie wykorzystywany jest zestaw narzędzi przetwarzania języka naturalnego do oznakowania każdego dokumentu dodatkowymi informacjami (morfologicznymi, składniowymi itd.). Gdy pojawia się pytanie, zostaje ono poddane analizie, której rezultat stanowi *model pytania*. Jest to struktura danych zawierająca pozyskane z pytania informacje trojakiego rodzaju: typ i treść pytania oraz zapytania do modułu wyszukiwania. Jak widać na schemacie, pytanie nie jest już później wykorzystywane. Zapytania używa się do ograniczenia bazy tekstów do małego podzbioru (tematycznie zgodnego z pytaniem). Wyselekcjonowane teksty trafiają następnie do modułu rozpoznawania nazw, rozpoznającego w nich wzmianki nazw zgodnych z typem pytania (np. określenia czasu, gdy pytanie zaczynało się od słowa *kiedy*). Aby ocenić zgodność wzmianki z pytaniem, z jej sąsiedztwa w dokumencie wybierany jest pewien zbiór słów tworzących kontekst i porównywany z pozyskaną na etapie analizy treścią pytania. Powstaje ranking odpowiedzi, uszeregowany według zgodności pytania i kontekstu wzmianki w tekście. Generowanie odpowiedzi polega na wybraniu jednej lub wielu z nich (w najprostszym rozwiązaniu pierwszej) i zwróceniu użytkownikowi wraz z dodatkowymi informacjami o źródle odpowiedzi.

Następne podrozdziały zawierają szczegółowy opis kolejnych etapów tego procesu.

3.2. Baza tekstów

Zanim przejdziemy do omawiania przebiegu przetwarzania pytania, zacznijmy od czynności, które mogą odbyć się wcześniej, tj. wstępnego przygotowania bazy tekstów. Jak wyjaśniono w punkcie 2.5.3, istotną cechą odróżniającą RAFAELa od innych systemów odpowiadających na pytania są niskie wymagania co do tego zasobu. Źródło udzielanych odpowiedzi stanowi po prostu zbiór tekstów w języku polskim, zapisanych w formie plików tekstowych w kodowaniu UTF-8. Jedyna struktura tekstu, z jakiej można skorzystać przy analizie, to podział na akapity poprzez znaki nowej linii.

Prosta struktura bazy oznacza, że jej przeszukiwanie może się okazać bardzo kosztowne obliczeniowo. Implementując system prawdziwie pozbawiony ograniczeń dziedzinowych należy oczekiwać, że zbiór tekstów będzie pokrywał szeroki zakres tematów, co skutkuje jego znacznymi rozmiarami. Przykładowo, polska Wikipedia, wykorzystana w niniejszej pracy do ewaluacji, zawiera około 900 tysięcy artykułów i 127 milionów słów, czyli 1,06 GB tekstu (szczegóły w punkcie 5.1.1). Poddawanie pełnej analizie wszystkich tekstów w trakcie poszukiwania odpowiedzi na każde zadane pytanie skutkowałoby nieakceptowalnymi czasami działania. Powszechnie stosowane rozwiązanie problemu wydajności przetwarzania polega na wprowadzeniu do systemu wyszukiwarki tekstowej, której rolą jest wybranie pewnej grupy dokumentów, należącej do tego samego zakresu tematycznego, co pytanie. Oczywiście wybór ten ma z konieczności charakter przybliżony, ale za to odbywa się bardzo szybko. Dopiero tak wyselekcjonowane dokumenty podlegają dalszej analizie.

Dodatkową zaletę tego rozwiązania stanowi ograniczenie wpływu szumu, tj. zdań, które pozornie odpowiadają na pytanie, ale w rzeczywistości odnoszą się do zupełnie innego kontekstu czy nawet dziedziny wiedzy.

W pełnej analizie dotyczącej jedynie wybranych dokumentów wykorzystywana jest informacja pozyskana przez znakowanie tekstu narzędziami lingwistycznymi. Proces ten można wykonać w ramach wstępnego przetwarzania całej bazy lub też już po zadaniu pytania w odniesieniu do wskazanych dokumentów. Wybór któregoś z tych rozwiązań zależy od konkretnego zastosowania – w omawianej implementacji ze względu na ewaluację zdecydowano się na znakowanie wstępne. Wymagało to jednorazowego poświęcenia długiego czasu (około dwóch tygodni ciągłej pracy dla oznakowania polskiej Wikipedii na komputerze osobistym) i przechowywania dużej ilości danych (po skompresowaniu 23GB), natomiast wielokrotnie powtarzane eksperymenty odbywały się znacznie szybciej. Rozwiązanie ze znakowaniem „na żądanie" można znaleźć np. w systemach *Webclopedia* (Hovy i inni, 2000) i *LASSO* (Moldovan i inni, 2000), zaś znakowanie wstępne w systemie *BulQA* (Simov i Osenova, 2005).

Podsumowując, przygotowanie bazy tekstów dla systemu RAFAEL składa się z dwóch kroków: opracowania indeksu i znakowania.

3.2.1. Indeks

W systemie RAFAEL do wyszukiwania dokumentów zgodnych tematycznie z pytaniem zastosowano wyszukiwarkę *Lucene* 3.6[3]. Na podstawie zbioru tekstów przygotowuje się dwa indeksy pełnotekstowe:

- bazujący na formach ortograficznych[4] słów (w oryginalnej postaci),
- bazujący na fragmentach generowanych z użyciem wbudowanego w *Lucene* stemera[5] *Stempel* (Galambos, 2001).

Dodatkowo dla każdego dokumentu w indeksie przechowywany jest również adres URL i tytuł, za który, z braku informacji o strukturze, uznaje się pierwszy akapit tekstu.

3.2.2. Znakowanie

Znakowanie ma na celu przygotowanie dokumentów w bazie do pełnej analizy poprzez wzbogacenie ich o informacje lingwistyczne, które będą niezbędne do odnalezienia odpowiedzi. Znakowanie odbywa się na następujących poziomach:

1. struktura odzwierciedlająca podział na paragrafy,
2. podział na segmenty[6],
3. tagowanie segmentów,
4. parsowanie powierzchniowe,

[3] Dostępna na stronie `http://lucene.apache.org/`.

[4] Zobacz słownik pojęć w Dodatku.

[5] *Stemer* służy do wybierania takich fragmentów słów, które są wspólne dla różnych jego form. Proces takiego przekształcania słów nazywamy *stemingiem*.

[6] Zobacz słownik pojęć w Dodatku.

5. lematyzacja grup składniowych,
6. nazwy własne.

Całe informacja lingwistyczna przechowywana jest w formacie będącym wariantem standardu *TEI P5* (korzystającego z XML), opracowanym na użytek Narodowego Korpusu Języka Polskiego (Przepiórkowski i inni, 2012).

Struktura

Ponieważ dane wejściowe mają postać plików tekstowych bez jakichkolwiek znaczników formatowania lub określających strukturę, na tym etapie możliwy jest jedynie podział na akapity zgodnie ze znakami nowej linii. Do wykonania tego zadania stworzony został skrypt w języku *Python*. Efektem jego działania jest plik `text_structure.xml` zgodny z formatem NKJP. Przykładowo dla artykułu o identyfikatorze `AA/00/0013`, widniejącego w Wikipedii pod tytułem *Abisynia*[7], wynik wygląda następująco:

```
<?xml version="1.0" encoding="UTF-8"?>
<?oxygen RNGSchema="NKJP_structure.rng" type="xml"?>
<teiCorpus xmlns:xi="http://www.w3.org/2001/XInclude"
        xmlns="http://www.tei-c.org/ns/1.0">
  <xi:include href="NKJP_header.xml"/>
  <TEI>
    <xi:include href="header.xml"/>
    <text>
    <group>
    <text decls="#bibl-1" xml:id="0013">
    <front>
      <docTitle>
        <titlePart type="main" xml:id="titlePart-1">0013
        </titlePart>
      </docTitle>
    </front>
    <body>
      <p xml:id="p-2">Abisynia</p>
      <p xml:id="p-4">Abisynia - dawna nazwa Etiopii,
    wywodząca się najprawdopodobniej od nazwy plemienia
    południowo arabskiego Habesz, które przywędrowało
    do Afryki przypuszczalnie jeszcze przed naszą erą.
    Uważa się, iż jedną z~przyczyn, dla których
    Etiopczycy zrezygnowali z~nazwy Abisynia i~
    wprowadzili nową (Etiopia) jest pogardliwe znaczenie
    formy "habesz" w~języku arabskim, oznaczające
    "mieszańca, włóczykija". Słowo to było używane w
    okresach waśni przez mieszkających w~kraju wyznawców
    islamu na określenie chrześcijan.</p>
      <p xml:id="p-5">Starą nazwę "Abisynia" zastąpiono
    formą "Etiopia", występującą m.in. w~tekstach
    biblijnych, gdzie nazwą tą opatrywany jest ten
    właśnie afrykański kraj. Wyraz Etiopia wywodzi się z
    greckiego "aithiopes", tj. "brunatne twarze". Mianem
    tym greccy podróżnicy określali ludy zamieszkujące
    obszar północno-wschodniej Afryki (tj. Sudan, Nubię
    oraz Etiopię właściwą). Ponieważ nazwa ta występuje
    w~oryginalnych tekstach biblijnych pisanych greką,
```

[7] `http://pl.wikipedia.org/wiki/Abisynia`

```
    została ona zaakceptowana jako oficjalna nazwa
    państwa.</p>
    </body>
    </text>
    </group>
    </text>
  </TEI>
</teiCorpus>
```

Znaczniki <p> odpowiadają akapitom – fragmentom rozdzielonym znakami nowej linii w tekście oryginalnym.

Segmentacja

Segmentacja polega na podziale akapitów na zdania i zdań na segmenty, stanowiące podstawowe jednostki oznakowanego tekstu. Najczęściej granice zdań wyznaczają kropki, a segmentów znaki odstępu, ale są od tej reguły wyjątki, jak np. kropki w skrótach lub słowa składające się z kilku segmentów (*żebyście* lub *chciałbym*). Segmentację przeprowadza analizator morfosyntaktyczny *Morfeusz Polimorf* 0.82 (Woliński i inni, 2012). Plik zawierający rezultat jego działania (`ann_segmentation.xml`) dla przykładowego artykułu wygląda następująco[8]:

```
(...)
<p corresp="text_structure.xml#p-2" xml:id="segm_p-2">
  <s xml:id="segm_p-2.1-s">
    <!-- Abisynia -->
    <seg
     corresp="text_structure.xml#string-range(p-2,0,8)"
     xml:id="segm_p-2.1-seg"/>
  </s>
</p>
<p corresp="text_structure.xml#p-4" xml:id="segm_p-4">
  <s xml:id="segm_p-4.1-s">
    <!-- Abisynia -->
    <seg
     corresp="text_structure.xml#string-range(p-4,0,8)"
     xml:id="segm_p-4.2-seg"/>
    <!-- - -->
    <seg
     corresp="text_structure.xml#string-range(p-4,9,1)"
     xml:id="segm_p-4.3-seg"/>
    <!-- dawna -->
    <seg
     corresp="text_structure.xml#string-range(p-4,11,5)"
     xml:id="segm_p-4.4-seg"/>
    <!-- nazwa -->
    <seg
     corresp="text_structure.xml#string-range(p-4,17,5)"
     xml:id="segm_p-4.5-seg"/>
    <!-- Etiopii -->
    <seg
     corresp="text_structure.xml#string-range(p-4,23,7)"
```

[8] Ze względu na ograniczone miejsce przedstawiono tylko fragmenty z tego i następnych plików, pominięto także nagłówki i znaczniki zewnętrzne – są one analogiczne do tych w `text_structure.xml`.

```
      xml:id="segm_p-4.6-seg"/>
         (...)
   </s>
   <s xml:id="segm_p-4.2-s">
    <!-- Uważa -->
    <seg
     corresp="text_structure.xml#string-range(p-4,185,5)"
     xml:id="segm_p-4.28-seg"/>
    <!-- się -->
    <seg
     corresp="text_structure.xml#string-range(p-4,191,3)"
     xml:id="segm_p-4.29-seg"/>
(...)
```

Widać, że akapity (znaczniki <p>) zostały rozbite na zdania (znaczniki <s>), a te na segmenty (znaczniki <seg>). Dla każdego segmentu określono jego zakres, np. zapis `string-range(p-2,0,8)` oznacza, że opisywany segment zaczyna się od pierwszego znaku akapitu `p-2` i ma długość 8 znaków.

Tagowanie

Tagowanie polega na przypisywaniu każdemu z segmentów formy bazowej i interpretacji morfosyntaktycznej[9], składającej się z tzw. *tagów*. Dla języków o rozbudowanej fleksji, takich jak polski, proces ten przebiega dwuetapowo. W pierwszym kroku analizator bazujący na słowniku gramatycznym określa wszystkie możliwe pary lemat-interpretacja[10]. Następnie narzędzie ujednoznaczniania morfosyntaktycznego na podstawie kontekstu określa, która z tych par stanowi prawidłową interpretację danej formy ortograficznej. W RAFAELu możliwe interpretacje określa *Morfeusz Polimorf* 0.82, a ujednoznacznienia dokonuje tager transformacyjny *PANTERA* 0.9.1 (Acedański, 2010). Efekt ich działania (plik `ann_morphosyntax.xml`) składa się z opisów segmentów w następującej formie:

```
(...)
<seg corresp="ann_segmentation.xml#segm_p-4.4-seg"
        xml:id="morph_2.2.3-seg">
  <fs type="morph">
    <f name="orth">
      <string>dawna</string>
    </f>
    <f name="interps">
      <fs type="lex" xml:id="morph_2.2.3.1-lex">
        <f name="base">
          <string>dawny</string>
        </f>
        <f name="ctag">
          <symbol value="adj"/>
        </f>
        <f name="msd">
          <vAlt>
            <symbol value="sg:nom:f:pos"
            xml:id="morph_2.2.3.1.1-msd"/>
            <symbol value="sg:voc:f:pos"
```

[9] Zobacz słownik pojęć w Dodatku.

[10] Zobacz słownik pojęć w Dodatku.

```
            xml:id="morph_2.2.3.1.2-msd"/>
          </vAlt>
        </f>
      </fs>
    </f>
      <f name="disamb">
        <fs type="tool_report">
        <f fVal="#morph_2.2.3.1.1-msd" name="choice"/>
        <f name="interpretation">
          <string>dawny:adj:sg:nom:f:pos</string>
        </f>
      </fs>
    </f>
  </fs>
</seg>
(...)
```

Powyższy fragment zawiera opis słowa, którego forma ortograficzna ma postać *dawna* (`<f name="orth">`). Analizator morfosyntaktyczny rozpoznał, że jest to przymiotnik (element `<symbol value="adj"/>` w znaczniku `<f name="ctag">`) o formie bazowej *dawny* (`<f name="base">`). Możliwe są dwie zbliżone interpretacje: mianownik rodzaju żeńskiego w liczbie pojedynczej i stopniu równym (`sg:nom:f:pos`) lub analogiczna interpretacja z wołaczem (`sg:voc:f:pos`)[11]. W procesie ujednoznaczniania za właściwą została uznana pierwsza z nich (`<f name="disamb">`).

Gdy słownik analizatora morfosyntaktycznego nie zawiera danego słowa, np. w przypadku nazwy własnej, przypisuje mu specjalny znacznik `ign`. W takiej sytuacji *PANTERA* wykorzystuje moduł zgadujący opis morfosyntaktyczny o nazwie *Odgadywacz* z tagera *TaKIPI* 1.8 (Piasecki, 2007).

Parsowanie powierzchniowe z lematyzacją

Parsowanie powierzchniowe ma na celu wskazanie w zdaniu prostych struktur łączących kilka słów; np. rzeczownik i odnoszące się do niego przymiotniki tworzące razem grupę nominalną. Jest to zatem poziom pośredni pomiędzy traktowaniem segmentów indywidualnie a tworzeniem pełnego drzewa rozbioru zdania. Do tego zadania wykorzystany został parser regułowy *Spejd* 1.3.7 (Przepiórkowski, 2008). Rozpoznaje on grupy zagnieżdżone oraz wskazuje centra semantyczne i składniowe w każdej grupie. Zbiór reguł sterujących działaniem parsera tworzy *gramatykę* – w RAFAELu użyto gramatyki dla korpusu NKJP z dnia 14.02.2012.

Lematyzacja oznacza konwersję słów i wyrażeń wielosłowowych do lematów, czyli form bazowach. Funkcję tę w odniesieniu do segmentów pełni tager, zaś w przypadku grup składniowych można wykorzystać parser, ale konieczna jest zmiana reguł gramatyki. Wynika to z faktu, że często lemat grupy składniowej nie stanowi złożenia lematów jej składników. Na przykład rozważmy wyrażenie *zielonej gęsi*, dla którego złożenie lematów składników brzmi *zielony gęś*, a prawidłowy lemat grupy – *zielona gęś*. W niniejszej pracy wykorzystano modyfikacje opracowane do tego celu przez Degórskiego (2012), rozbudowa-

[11] W przebiegu znakowania wykorzystano *tagset* (zbiór tagów) z projektu NKJP, wyjaśnienie ich znaczenia można odnaleźć we wspomnianej pracy (Przepiórkowski i inni, 2012).

ne dla potrzeba RAFAELa. Rezultat działania parsera powierzchniowego (plik `ann_groups.xml`) wygląda następująco:

```
(...)
<seg xml:id="groups_2.2-s_14">
<!--  rule="PrepNG: Prep + NG" -->
  <fs type="group">
    <f name="orth">
      <string>przed naszą erą</string>
    </f>
    <f name="type">
      <symbol value="PrepNG"/>
    </f>
  </fs>
  <ptr type="synh" target="ann_words.xml#words_2.2-s_24"/>
  <ptr type="semh" target="#groups_2.2-s_15"/>
</seg>
<seg xml:id="groups_2.2-s_15">
<!--  rule="NGa: Adj(Adj) + Noun" -->
  <fs type="group">
    <f name="orth">
      <string>naszą erą</string>
    </f>
    <f name="type">
      <symbol value="NGa"/>
    </f>
    <f name="base">
      <string>ADJ(nasz,f,pos) era</string>
    </f>
  </fs>
  <ptr type="nonhead"
  target="ann_words.xml#words_2.2-s_25"/>
  <ptr type="head" target="ann_words.xml#words_2.2-s_26"/>
</seg>
(...)
```

W powyższym przykładzie widzimy opis wyrażenia *przed naszą erą*. Zewnętrzna grupa przysłówkowa (`<symbol value="PrepNG"/>` w znaczniku `<f name="type">`) ma dwa składniki: przysłówek *przed*, będący centrum składniowym (`<ptr type="synh">`) oraz grupę wewnętrzną, będącą centrum semantycznym (`<ptr type="semh">`). Wewnętrzna grupa nominalna (`<symbol value="NGa"/>` w znaczniku `<f name="type">`) zawiera słowo *nasz* i pełniące rolę obu centrów *era*. W przypadku grupy wewnętrznej określono również formę bazową – zapis `ADJ(nasz,f,pos) era` oznacza, że tworzy ją przymiotnik *nasz* w rodzaju żeńskim i stopniu równym wraz ze słowem *era*. W trakcie wczytywania tego pliku przez RAFAELa ciąg ten zostanie przekształcony do oczekiwanej postaci *nasza era*.

Nazwy własne

Celem ostatniego etapu znakowania jest wykrycie obecnych w tekście nazw własnych i określenie ich typu. Do tego celu w RAFAELu wykorzystuje się dwa dostępne narzędzia: *Nerf* 0.1 (Savary i Waszczuk, 2010) i *Liner2* 2.3 (Marcińczuk i Janicki, 2012). Ponieważ zadanie to ma kluczowe znaczenie z punktu widzenia całego problemu odpowiadania na pytania, omówiono je szczegóło-

wo w punkcie 3.4.1. Tutaj przedstawiono jedynie wyniki działania tych dwóch narzędzi na przykładowym tekście. Oba programy działają na zasadzie uczenia maszynowego, a więc do przeprowadzania znakowania konieczny jest model reprezentujący informacje zebrane w procesie trenowania. W tym celu wykorzystano modele dystrybuowane wraz z programami, powstałe z użyciem ręcznie oznakowanych korpusów treningowych o rozmiarze 1 miliona słów (*Nerf*) lub 2000 dokumentów (*Liner2*).

Rezultat rozpoznawania narzędziem *Nerf* (plik `ann_named.xml`) wygląda następująco:

```
(...)
<seg xml:id="named_2.3-s_n2">
  <fs type="named">
    <f name="type">
      <symbol value="placeName"/>
    </f>
    <f name="subtype">
      <symbol value="country"/>
    </f>
    <f name="derived">
      <fs type="derivation">
        <f name="derivType">
          <symbol value="relAdj"/>
        </f>
      </fs>
    </f>
    <f name="orth">
      <string>arabskim</string>
    </f>
  </fs>
  <ptr target="ann_morphosyntax.xml#morph_2.3.31-seg"/>
</seg>
(...)
```

W powyższym przykładzie widać rozpoznanie słowa *arabskim* jako nazwy własnej o kategorii ogólnej `placeName` (miejsce) i szczegółowej `country` (kraj). Segment ten stanowi formę pochodną (`<f name="derived">`) typu przymiotnikowego (`<symbol value="relAdj"/>`).

Liner2 obsługuje dane wejściowe i wyjściowe w formacie CCL, zatem w RAFAELu konieczne było zastosowanie modułu konwersji do formatu NKJP, który zrealizowano w formie skryptu w języku *Python*. Wynikowy plik z rozpoznanymi nazwami (`ann_named_liner.xml`) wygląda następująco:

```
(...)
<seg xml:id="named_2.2-s_n1">
  <fs type="named">
    <f name="type">
      <symbol value="city_nam"/>
    </f>
    <f name="orth">
      <string>Abisynia</string>
    </f>
  </fs>
  <ptr target="ann_morphosyntax.xml#morph_2.2.1-seg"/>
</seg>
<seg xml:id="named_2.2-s_n2">
```

Nazwa	*Nerf*	*Liner2*
Abisynia	-	city_nam
Etiopia	orgName	country_nam
Habesz	persName:surname	tech_nam
Afryka	geogName	city_nam/continent_nam
Etiopczyk	placeName:country(persDeriv)	nation_nam
arabski	placeName:country(relAdj)	-
afrykański	geogName(relAdj)	-
grecki	placeName:country(relAdj)	-
Sudan	-	country_nam
Nubia	-	country_nam

Tabela 3.1. Wyniki rozpoznawania nazw własnych w przykładowym tekście. Dla każdej z nazw (zapisanej w postaci podstawowej) podano, jakie typy przypisały jej narzędzia *Nerf* i *Liner2*.

```
  <fs type="named">
    <f name="type">
      <symbol value="country_nam"/>
    </f>
    <f name="orth">
      <string>Etiopii</string>
    </f>
  </fs>
  <ptr target="ann_morphosyntax.xml#morph_2.2.5-seg"/>
</seg>
(...)
```

Słowo *Abisynia* zostało rozpoznane jako nazwa miasta (city_nam), a *Etiopii* jako nazwa kraju (country_nam).

Wyniki działania tych narzędzi różni nie tylko forma zapisu, ale także zestaw rozpoznanych słów. W tabeli 3.1 podano, jakie etykiety przypisano nazwom własnym w przykładowym tekście. Tylko jedno ze słów (*Etiopczyk*) zostało poprawnie rozpoznane przez oba narzędzia. W innych przypadkach albo brakowało rozpoznania w jednym z rozwiązań (np. *arabski*, *Sudan*) albo przypisywano niewłaściwą kategorię (np. *Etiopia* jako organizacja). Dodatkowo problem stanowi wybór typu, gdy możliwa jest więcej niż jedna interpretacja danej nazwy. Dotyczy to słowa *Afryka*, które może być nazwą wsi, oraz *Abisynia* – miejscowości o takiej nazwie jest w Polsce aż 10. Problem ten powróci jako jedna z istotnych przyczyn udzielania błędnych odpowiedzi na pytania przez system (punkt 6.1.1).

3.3. Analiza pytania

Pierwszym procesem, który zostaje uruchomiony po wprowadzeniu przez użytkownika pytania, jest jego analiza. Jej cel stanowi wydobycie z pytania wszelkich danych, które będą potrzebne do znalezienia odpowiedzi. Struktura gromadząca te informacje, nazywana *modelem pytania*, składa się z następujących elementów:

1. *Typ pytania* – opisuje wiedzę na temat oczekiwanego rodzaju odpowiedzi. Dzięki temu wiadomo, jakiego typu informacja może być brana pod uwagę przy poszukiwaniach. Typ określa się na trzech poziomach szczegółowości:
   - *Ogólny typ pytania* – klasyfikuje pytania ze względu na sposób, w jaki można szukać odpowiedzi.
   - *Typ nazwy własnej* – określony wtedy, gdy, zgodnie z ustalonym wcześniej typem ogólnym, pytanie dotyczy nazwy własnej. Listę rozróżnianych typów ogólnych i nazw własnych podano w punkcie 3.3.1.
   - *Synset centrum pytania* – określa synset w ontologii WordNet, który reprezentuje centrum pytania. Ten element jest wykorzystywany przy głębokim rozpoznawaniu nazw.
2. *Zapytanie do wyszukiwania* – używane do przeszukania indeksu w celu odnalezienia zbioru dokumentów potencjalnie zgodnych tematycznie z pytaniem.
3. *Treść pytania* – zbiór słów z pytania, które spodziewamy się znaleźć również w tekście w miejscu, w którym znajduje się szukana odpowiedź.

Znaczenie wymienionych składników zostanie szczegółowo omówione w kolejnych sekcjach. Warto zwrócić uwagę, że na tym etapie przetwarzania system porzuca pytanie w formie tekstowej – nie będzie ono już potrzebne, gdyż zostało w całości zastąpione przez jego model. Pewne elementy przedstawionych w tym podrozdziale rozwiązań i eksperymentów opublikowano we wcześniejszych pracach (Przybyła, 2013a,b).

3.3.1. Klasyfikacja pytań

Klasyfikacja pytania ma na celu odkrycie jego typu, określającego oczekiwaną postać odpowiedzi. W tym celu dla zdefiniowanych kategorii pytań buduje się klasyfikator, przydzielający odpowiednią etykietę na podstawie analizy pytania.

Kategorie

Dla potrzeb RAFAELa pytania o proste fakty podzielono na następujące grupy, odpowiadające ogólnemu typowi pytania:

- Pytania o **potwierdzenie** (oznaczane etykietą `VERIFICATION`), na które można odpowiedzieć *Tak* lub *Nie*, np. *Czy gen. Władysław Sikorski zginął w katastrofie lotniczej?*,
- Pytania **wyboru** (`WHICH`), odpowiedź na które polega na wyborze jednej z zaproponowanych opcji, np. *Co spowodowało wypadek gen. Władysława Sikorskiego: przypadek czy celowe działanie?*,
- Pytania o **nazwę własną** (`NAMED_ENTITY`), wymagające wskazania bytu określanego nazwą własną, np. *Skąd wystartował ostatni lot gen. Władysława Sikorskiego?*,

- Pytania o **nazwę ogólną**[12] (UNNAMED_ENTITY) są podobne do powyższych, ale poszukiwany byt nie mieści się w kategoriach określanych przez nazwy własne, np. *Jaki stopień wojskowy posiadał Władysław Sikorski?*,
- Pytania o **inną nazwę** (OTHER_NAME), gdzie poszukujemy innej nazwy określającej byt wskazany w pytaniu, np. *Jaki pseudonim gen. Władysław Sikorski nosił w Związku Walki Czynnej?*,
- Pytania o **wiele nazw własnych** (MULTIPLE), odpowiedź na które stanowi lista nazw własnych bytów spełniających podane kryteria, np. *Kto uczestniczył w ostatnim locie gen. Władysława Sikorskiego?*

Zgodnie z definicją prostych faktów podaną w punkcie 2.2.1, wykluczamy pytania wymagające odpowiedzi złożonej typu *Czym jest . . . ?* lub *Dlaczego . . . ?*. Dodatkowo pominięto rzadkie typy mieszane, np. o nazwy własne różnych typów (*Gdzie i kiedy zginął . . . ?*) lub łączące nazwy własne i ogólne (*Jaki stopień wojskowy i od kiedy posiadał . . . ?*).

Wprowadzona w punkcie 2.2.1 klasa pytań o nazwy bytów składa się z dwóch z powyższych kategorii – pytań o nazwę ogólną i własną. W obu przypadkach odpowiedzieć można odnajdując poszukiwaną nazwę w tekście. Różnica tkwi w przynależności tej nazwy do grupy nazw własnych. Omawiany moduł analizy pytania rozpoznaje typ ogólny dla wszystkich pytań o proste fakty, jednak RAFAEL jest w stanie udzielać odpowiedzi tylko na te dotyczące nazw bytów.

Tradycyjne kategorie nazw własnych, wprowadzone w cyklu konferencji *Message Understanding Conference* (Grishman i Sundheim, 1996), to miejsca (LOC), osoby (PER), organizacje (ORG) i inne (MISC). Podział ten jest jednak zbyt ubogi, a nie istnieje inny powszechnie używany standard. W niniejszej pracy przyjęto szeroki zakres nazw własnych, obejmujący również określania liczbowe. W systemie rozróżniane są zatem następujące typy:

- miejsce (oznaczane etykietą PLACE),
- kontynent (CONTINENT),
- rzeka (RIVER),
- jezioro (LAKE),
- góra (MOUNTAIN),
- łańcuch górski (RANGE),
- wyspa (ISLAND),
- archipelag (ARCHIPELAGO),
- morze (SEA),
- ciało niebieskie (CELESTIAL_BODY),
- państwo (COUNTRY),
- region, stan, województwo, itp. (STATE),
- miejscowość (CITY),
- narodowość (NATIONALITY),
- osoba (PERSON),
- imię (NAME),
- nazwisko (SURNAME),
- zespół muzyczny (BAND),

[12] Zobacz słownik pojęć w Dodatku.

- dynastia (DYNASTY),
- organizacja (ORGANISATION),
- przedsiębiorstwo (COMPANY),
- wydarzenie historyczne lub cykliczne (EVENT),
- określenie czasu (TIME),
- stulecie (CENTURY),
- rok (YEAR),
- okres (PERIOD),
- liczba (COUNT),
- wielkość fizyczna (QUANTITY),
- nazwa modelu lub egzemplarza pojazdu (VEHICLE),
- imię zwierzęcia (ANIMAL),
- tytuł dzieła (TITLE).

Łatwo zauważyć, że powyższe kategorie nie są od siebie całkowicie niezależne i odrębne. Mamy tu do czynienia z zawieraniem znaczeniowym (np. kraje i miejscowości są miejscami), zawieraniem fizycznym (np. archipelag składa się z wysp, a zespół muzyczny z ludzi) oraz zawieraniem notacyjnym (określenie czasu zawiera rok, a osoby – imię lub nazwisko). Konflikty te rozstrzygamy, stosując najwęższą możliwą kategorię, np. etykietę PLACE nadajemy tylko tym pytaniom, które nie określają typu miejsca (*Gdzie . . . ?*) lub odnoszą się do typu miejsca nieuwzględnionego przez inne kategorie (*W której dzielnicy . . . ?*).

Określanie typu pytania

Zadanie wyboru jednej z etykiet na podstawie pytania stanowi przykład problemu *klasyfikacji tekstów*. Przykład to jednak szczególny z dwóch względów. Po pierwsze, badane teksty są niezwykle krótkie (średnio 10 słów w zbiorze treningowym opisanym w punkcie 5.1.2), co czyni macierz reprezentacji *bag of words*[13] wyjątkowo rzadką. Po drugie, hierarchiczność zestawu kategorii może także wpływać na klasyfikację.

Do rozwiązania problemu określania kategorii pytań jako zadania klasyfikacji stosowano różne rozwiązania, sprawdzające się w innych zastosowaniach uczenia maszynowego. W pracy dotyczącej języka chorwackiego Lombarović i inni (2011) wykorzystali metody takie jak:

- maszyny wektorów podpierających,
- drzewa decyzyjne,
- algorytm modelowania językowego,
- algorytm k najbliższych sąsiadów.

Ze względu na dużą liczbę cech stosowano ich selekcję z użyciem następujących miar istotności:

- informacji wzajemnej,
- statystyki χ^2,
- częstości występowania słowa w dokumentach.

[13] Reprezentacja tekstu w postaci *bag of words* polega na tym, że ignorujemy kolejność słów i zapisujemy tekst jako ciąg liczb, z których każda odpowiada liczbie wystąpień pewnego słowa.

Poruszono także kwestię sposobu reprezentacji słów w modelu, która ma szczególne znaczenie w przypadków języków z bogatą odmianą (np. słowiańskich) – zgodnie z oczekiwaniami, zastosowanie stemingu lub lematyzacji przyniosło polepszenie rezultatów.

Li i Roth (2002) zwrócili szczególną uwagę na generowanie cech dla klasyfikatora pytań. Do ich wytworzenia wykorzystano następujące źródła:

- formy ortograficzne słów,
- części mowy,
- wyrażenia wielowyrazowe (np. grupy nominalne),
- nazwy własne,
- centrum pytania,
- listy słów spodziewanych w pytaniach określonych typów.

Dla zbioru treningowego powstało około 200.000 cech; tak duża i rzadka przestrzeń parametrów może stanowić problem dla większości klasyfikatorów, dlatego wprowadzono specjalnie przystosowane narzędzie *SNoW* (*Sparse Network of Winnows*). Najpierw przypisywano do pytania kategorię ogólną, a następnie szczegółową. Niestety, wykorzystanie hierarchiczności zestawu kategorii nie przyniosło korzyści w stosunku do zwykłej klasyfikacji. Taki sam wniosek pojawił się w pracy, w której między innymi badano podobne rozwiązanie dla języka chorwackiego (Lombarović i inni, 2011).

Zazwyczaj, ze względu na małą liczbę dostępnych pytań, zamiast modeli uzyskanych w procesie uczenia maszynowego stosuje się ręcznie opracowane reguły. Punkt wyjścia stanowi tutaj fakt, że bardzo często zaimki rozpoczynające pytanie jednoznacznie określają jego typ. Przykładowo, pytanie typu *Kiedy...?* z pewnością można zaliczyć do kategorii `NAMED_ENTITY:TIME`. Tego typu przypadki można obsługiwać za pomocą dopasowywania zbioru wyrażeń regularnych, z których każde odpowiada pewnemu typowi. Do takiego podejścia ograniczyli się np. Lee i inni (2005). Zastosowany przez nich zbiór zawierał aż 1273 wzorce.

Zaimki pytajne, które nie determinują jednoznacznie typu pytania, to na przykład *który* i *jaki*. Gdy się pojawią, konieczne staje się przeanalizowanie grupy nominalnej, która po nich następuje. Grupę tę nazywamy *centrum pytania* (ang. *question focus*), gdyż określa, co jest sednem pytania. Przykładowo, w pytaniu *W której francuskiej miejscowości urodził się Vincent van Gogh?* centrum stanowi fraza *francuskiej miejscowości*, skąd można wnioskować, że pytanie dotyczy właśnie miejscowości. Taką analizę pytania przeprowadzono na przykład w pracy Ittycheriaha (2007), wykorzystując pełne drzewo rozbioru pytania. Istotną rolę odgrywa wybór głównego słowa z grupy – w tym wypadku typowi nazwy własnej `CITY` odpowiada *miejscowość*, a nie *francuska miejscowość*.

Klasyfikując niejednoznaczne pytania często trzeba wyjść ponad poziom składni zdania i uwzględnić również znaczenie słów. Rozważmy zdanie *Który francuski postimpresjonista uciął sobie ucho?*. W tym wypadku nie wystarczy odnaleźć centrum pytania – słów *francuski postimpresjonista*. Trzeba jeszcze wiedzieć, że określają one osobę, czyli byt kategorii `PERSON`. Klasyczne rozwiązanie bazujące na liście słów przypisanych do typów pytań (np. *miejscowość* do `CITY`) powoduje konieczność zgromadzenia bardzo wielu możliwych

Wyrażenie regularne	Zbiór typów
`^Czy[ ,](.*)\?$`	`TRUEORFALSE`
`^(.*[ ,])?[Ii]le[ ,](.*)\?$`	`COUNT,PERIOD,QUANTITY`
`^(.*[ ,])?[Kk]omu[ ,](.*)\?$`	`PERSON,COUNTRY,COMPANY,BAND,...`
`^W którym wieku[ ,](.*)\?$`	`CENTURY`
`^Jaki nosi tytuł[ ,](.*)\?$`	`TITLE`
`^(.*[ ,])?[Cc]zym[ ,](.*)\?$`	`UNNAMED_ENTITY`

Tabela 3.2. Przykłady ze zbioru 104 jednoznacznych reguł klasyfikatora pytań, bazujących na wyrażeniach regularnych i przypisanych im zbiorach możliwych typów.

określeń dla ogólnych typów pytań, takich jak `PERSON`. Lepiej zamiast tego skorzystać z bazy WordNet, w której synsety odpowiadające postimpresjoniście i osobie łączy łańcuch hiperonimii[14]. Wystarczy go prześledzić, by określić typ pytania w takim przypadku (Harabagiu i inni, 2001).

Dla potrzeb RAFAELa przeanalizowano, zaimplementowano i przetestowano wszystkie opisane powyżej możliwości:

1. podejście regułowe,
   a) dopasowywanie wzorców,
   b) analizę znaczeniową centrum pytania,
2. uczenie maszynowe.

Realizacja podejścia regułowego

W podejściu regułowym bazuje się na ręcznie opracowanej procedurze klasyfikacyjnej, biorącej pod uwagę możliwe sformułowania pytań określonych typów.

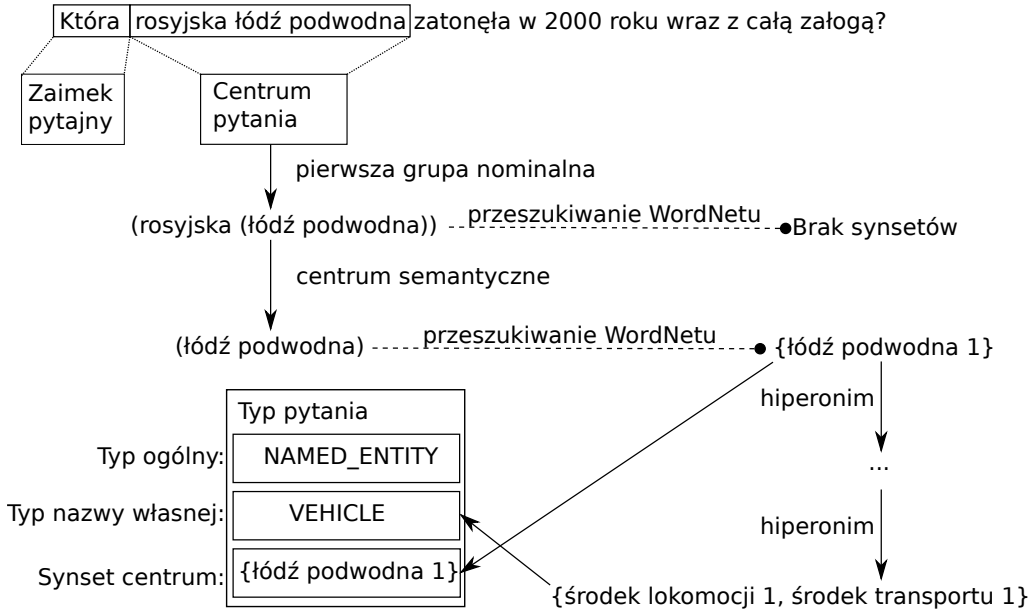
Podstawę analizy metodą dopasowywania wzorców, przedstawionej dalej jako algorytm 1, stanowią 104 wyrażenia regularne, odpowiadające konstrukcjom jednoznacznie wskazującym właściwą kategorię. Typów przypisanych do wyrażenia może być więcej niż jeden, ponieważ niektóre konstrukcje pasują do wielu z nich. Czasem nawet całe zdanie nie wystarczy, by wskazać dokładnie jeden typ. Przykładowo, odpowiedź na pytanie *Kto wywołał drugą wojnę światową?* może zawierać pojęcie typu `COUNTRY`, `NATIONALITY`, `ORGANISATION` lub `PERSON`. Wyrażenia opracowano ręcznie na podstawie ogólnej wiedzy językowej o zaimkach pytajnych w języku polskim i ich znaczeniu. Duża liczba wzorców wynika z konieczności uwzględnienia odmiany zaimków (*kto*, *kogo*, *komu*, etc.). Przykładowe wzorce przytoczono w tabeli 3.2.

Analiza pytania rozpoczyna się od próby dopasowania do niego każdego z wzorców. Jeśli dla jednego z nich się to uda, przypisany mu zbiór typów jest zwracany jako wynik klasyfikacji i kończy to przebieg algorytmu. Jeśli zaś żaden z tych wzorców nie pasuje do zdania, stosowany jest drugi zbiór, zawierający 72 wyrażenia odpowiadające niejednoznacznym konstrukcjom pytajnym, wymagającym analizy centrum pytania. Przykłady takich wzorców przedstawiono w tabeli 3.3. Jeśli i ten zbiór nie zawiera żadnego pasującego

[14] Hiperonimia to relacja szerszego znaczenia. Przykładowo słowo *zwierzę* jest hiperonimem słowa *ssak*.

Wyrażenie regularne	Zbiór typów
`^()Jak nazywa (się)[ ,](.*)\?$`	NAMED_ENTITY/UNNAMED_ENTITY
`^()Jaka (jest)[ ,](.*)\?$`	NAMED_ENTITY/UNNAMED_ENTITY
`^()Co (jest)[ ,](.*)\?$`	NAMED_ENTITY/UNNAMED_ENTITY
`^()(Który)[ ,](.*)\?$`	NAMED_ENTITY/UNNAMED_ENTITY
`^()(Jakiemu)[ ,](.*)\?$`	NAMED_ENTITY/UNNAMED_ENTITY

Tabela 3.3. Przykłady ze zbioru 72 niejednoznacznych reguł klasyfikatora pytań, dla których konieczna jest analiza centrum pytania.



Rysunek 3.2. Przykładowy przebieg procesu określania typu pytania o niejednoznacznej konstrukcji.

wyrażenia, moduł zwraca pusty model pytania, co oznacza rezygnację z poszukiwania odpowiedzi. W przeciwnym wypadku przystępuje się do badania centrum pytania.

W przypadku konstrukcji niejednoznacznych do określenia typu pytania konieczna jest analiza jego centrum z wykorzystaniem bazy WordNet. Dla języka polskiego istnieją dwie jej wersje: *plWordNet* (Piasecki i inni, 2009; Maziarz i inni, 2012) i *PolNet* (Vetulani i Kochanowski, 2014). W omawianym zadaniu użyty został *plWordNet* 2.1 ze względu na znacznie większy zasób słów – przykładowo zawiera on 112.000 synsetów rzeczownikowych, podczas gdy w *Polnecie* jest ich 11.700. Ta sama wersja bazy była używana także w innych komponentach systemu RAFAEL, które wymagały bazy WordNet.

Postępowanie w przypadku konstrukcji niejednoznacznych, przedstawione dalej jako algorytm 2, wymaga wcześniejszego oznakowania treści pytania z użyciem narzędzi, które omówiono w punkcie 3.2.2 (bez poziomu nazw własnych). Tabela 3.4 przedstawia przykład takiego oznakowania trzech prostych pytań ze zbioru treningowego. Skoro dopasowano jeden z niejednoznacznych wzorców, musi się odbyć proces poszukiwania centrum pytania, przedstawiony na przykładzie na rysunku 3.2. Słowo, po którym powinno ono nastąpić,

Z którymi państwami graniczy Lesotho?

segmenty	Z	którymi	państwami	graniczy	Lesotho
lemat	z	który	państwo	graniczyć	Lesotho
część mowy	prep	adj	subst	fin	subst
tagi	inst:nwok	pl:inst:n:pos	pl:inst:n	sg:ter:imperf	sg:gen:n
grupy 1		NGa które państwo			NG Lesotho
grupy 2	PrepNG				

Jakich pseudonimów używał Stefan Żeromski?

segmenty	Jakich	pseudonimów	używał	Stefan	Żeromski
lemat	jaki	pseudonim	używać	Stefan	Żeromski
część mowy	adj	subst	praet	subst	subst
tagi	pl:gen:m3:pos	pl:gen:m3	sg:m1:imperf	sg:nom:m1	sg:nom:m1
grupy	NGa jaki pseudonim			NGs Stefan Żeromski	

Stolicą którego państwa jest Chartum?

segmenty	Stolicą	którego	państwa	jest	Chartum
lemat	stolica	który	państwo	być	Chartum
część mowy	subst	adj	subst	fin	subst
tagi	sg:inst:f	sg:gen:n:pos	sg:gen:n	sg:ter:imperf	sg:nom:m3
grupy 1		NGa które państwo			NG Chartum
grupy 2	NGg stolica którego państwa				

Tabela 3.4. Przykładowe pytania podzielone na segmenty wraz z oznakowaniem na poziomie analizy morfosyntaktycznej (lemat, część mowy i tagi) oraz grup składniowych (typ i lemat grupy oraz podkreślenie centrum semantycznego). Dla czytelności pominięto znak zapytania na końcu pytań i warstwę słów składniowych, nieużywaną w RAFAELu.

określane jest przez wyrażenie regularne (środkowa para nawiasów we wzorcach z tabeli 3.3) – w naszym przykładzie jest to zaimek pytajny *Która*. Dodatkowo, przed samym centrum może wystąpić jedno z *wyrażeń wprowadzających*, np. *spośród*, *z*, *typ* itp. W takim przypadku kontynuujemy poszukiwania za tym wyrażeniem (analogiczne rozwiązanie zastosował też Ittycheriah (2007)). Dzięki tej technice w pytaniu *Jaki typ gwiazdy może powstać w wybuchu supernowej wielkości 8-10 mas słońca?* identyfikujemy centrum pytania w słowie *gwiazda*, a nie *typ*. Pierwsza grupa nominalna napotkana przy dalszym przeglądaniu zdania zostaje uznana za centrum pytania. W przykładzie na rysunku jest to *rosyjska łódź podwodna*. Pobieramy lemat tej grupy (w przykładzie bez zmian) i poszukujemy leksemu[15] o tym samym brzmieniu w bazie WordNet. Jeśli nie ma takiego leksemu, badana grupa zostaje zastąpiona przez jej centrum semantyczne (*łódź podwodna* zastępuje *rosyjską łódź podwodną*) i ponawiamy proces aż do odnalezienia leksemu w WordNecie lub wyczerpania hierarchii grupy. Jeżeli WordNet zawiera jakieś znaczenie danego słowa lub frazy, pobieramy listę bezpośrednich i pośrednich hiperonimów synsetu, do którego należy odnaleziony leksem[16]. Dla każdego z typów nazw własnych ręcznie wybrano listę synsetów, które odpowiadają mu znaczeniowo, np. dla typu `VEHICLE` jest to synset <środek lokomocji.1, środek transportu.1>. Jeśli jeden z nich znajdzie się w analizowanym zbiorze hiperonimów, nadajemy pytaniu typ ogólny `NAMED_ENTITY` i typ nazwy własnej zgodny z odnalezionym synsetem, np. `VEHICLE`. Gdy takiej zgodności brakuje lub żadna z grup w łańcuchu centrów semantycznych nie posiada odpowiednika w WordNecie, zwracamy ogólny typ `UNNAMED_ENTITY`. Gdy zamiast grupy nominalnej napotykamy grupę liczebnikową, przypisujemy jedynie typ ogólny `MULTIPLE`, np. dla pytań typu *Które dwa państwa toczyły wojnę stuletnią?*.

W dalszej części niniejszego podrozdziału omówiono pseudokody funkcji realizujących opisane powyżej postępowanie.

Przedstawiona jako algorytm 1 funkcja *getTypePatterns* przyjmuje na wejściu treść pytania i zwraca jego model, zawierający typ oraz treść pytania. Realizuje ona podejście bazujące jedynie na jednoznacznych wzorcach pytań i wykorzystuje następujące elementy:

removeInBrackets() usuwa cały tekst pomiędzy nawiasami zwykłymi, kwadratowymi lub klamrowymi.

unambiguousPatterns to lista zawierająca 104 jednoznaczne wzorce pytań, których przykłady przedstawiono w pierwszej kolumnie tabeli 3.2.

match(*$pattern$*,*$text$*).group(*k*) dla dopasowania wzorca $pattern$ do tekstu $text$ pobiera jego fragment odpowiadający k-tej parze nawiasów w wyrażeniu regularnym wzorca. W tym wypadku pobierane są grupy pierwsza i trzecia, ponieważ słowa stanowiące treść pytania mogą zarówno poprzedzać konstrukcję pytajną, jak i następować po niej. Przykładowo w zdaniu *Na czyje życzenie przebudowano pałac Na Wodzie?* pierwsza grupa obejmuje słowo *Na*, a trzecia *życzenie przebudowano pałac Na Wodzie*.

[15] Zobacz słownik pojęć w Dodatku.

[16] Problem wieloznaczności centrum i możliwe rozwiązania omówiono dalej.

```
Input: text – nieoznakowana treść pytania
Output: model – model pytania z uzupełnionymi polami
        odpowiadającymi typowi i treści pytania
begin
    text := removeInBrackets(text);
    foreach pattern in unambiguousPatterns do
        if pattern matches text then
            model.content := match(pattern,text).group(1) ∪
            ↪match(pattern,text).group(3) ;
            model.type.general:=generalTypes(pattern);
            model.type.ne:=neTypes(pattern);
            model.type.focus:=null;
            return model;
        end
    end
    return null;
end
```

Algorytm 1: Funkcja **getTypePatterns**(*text*) – analizuje pytanie i określa jego typ z użyciem jednoznacznych wzorców.

generalTypes() i ***neTypes()*** pobierają ogólny typ pytania i typ nazwy własnej przypisane do wzorca – przykłady znajdują się w drugiej kolumnie tabeli 3.2.

Przedstawiona jako algorytm 2 funkcja *getTypePatternsWordNet* także przyjmuje na wejściu treść pytania i zwraca jego model, zawierający typ oraz treść pytania. W odróżnieniu od *getTypePatterns* uwzględnia niejednoznaczne wzorce i określa typ nazwy własnej na podstawie analizy centrum pytania z użyciem bazy WordNet. Oprócz poprzednio wymienionych wykorzystuje następujące nowe elementy:

ambiguousPatterns to lista zawierająca 72 niejednoznaczne wzorce pytań, których przykłady znajdują się w pierwszej kolumnie tabeli 3.3.

end() zwraca numer ostatniego znaku wyodrębnionego fragmentu tekstu.

skipQuestionPrefixes() znajduje w tekście wyrażenia poprzedzające centrum pytania, tj. grupy przysłówkowe i wyrażenia wprowadzające (lista w tabeli 3.5), i zwraca numer pierwszego znaku po ich zakończeniu,

firstNominalChunk() znajduje w tekście pierwszy rzeczownik, grupę nominalną lub grupę liczebnikową. W przypadku, gdy w tym samym miejscu w tekście rozpoczyna się kilka takich składników (ze względu na zawieranie), wybiera najdłuższy z nich.

isNumericalGroup() sprawdza, czy dana grupa jest grupą liczebnikową.

hypernyms() zwraca zbiór pośrednich i bezpośrednich hiperonimów podanych synsetów.

nePairs to lista par synsetów i typów nazw własnych, skojarzonych ze sobą na zasadzie zbieżnego znaczenia. Na przykład typowi nazwy własnej `SEA` odpowiada synset <morze.1>, a `PERSON` – <osoba.4>. Czasami kilka

```
Input: text – oznakowana treść pytania
Output: model – model pytania z uzupełnionymi polami
        odpowiadającymi typowi i treści pytania
begin
    model.type.general:=null;
    model.type.ne:=null;
    model.type.focus:=null;
    model.content:=null;
    text := removeInBrackets(text);
    foreach pattern in ambiguousPatterns do
        if pattern matches text then
            model.content := match(pattern,text).group(1) ∪
            ↪match(pattern,text).group(3) ;
            focusStart := match(pattern,text).group(2).end()+1;
            break;
        end
    end
    if focusStart==null then
        return null;
    focusStart := skipQuestionPrefixes(focusStart,text);
    chunk := firstNominalChunk(focusStart,text);
    model.type.general:=UNNAMED_ENTITY;
    if chunk==null then
        return model;
    if isNumericalGroup(chunk) then
        model.type.general:=MULTIPLE;
        return model;
    end
    model.content := model.content \ chunk;
    model.type.focus=extractSynsets(chunk);
    if model.type.focus==null then
        return model;
    hypernyms=hypernyms(model.type.focus);
    foreach nePair in nePairs do
        if nePair.parent ∈ hypernyms then
            model.type.general := NAMED_ENTITY ;
            model.type.ne := nePair.neTypes ;
            break;
        end
    end
    return model;
end
```

Algorytm 2: Funkcja **getTypePatternsWordNet**(*text*) – analizuje pytanie i określa jego typ z użyciem niejednoznacznych wzorców i bazy WordNet.

Wyrażenia wprowadzające
z
spośród
gromada
klasa
rząd
grupa
odmiana
seria
rodzaj
typ
gatunek

Tabela 3.5. Wyrażenia wprowadzające, które są pomijane w procesie wyszukiwania centrum pytania (używane w *skipQuestionPrefixes()* w algorytmie 3).

par może zawierać ten sam typ NE, np. do TITLE przypisano <tytuł.1> i <dzieło.2, praca.6>.

Przedstawiona jako algorytm 3 funkcja *extractSynsets* przyjmuje na wejściu grupę nominalną lub rzeczownik i zwraca synsety WordNet, którym odpowiada ten fragment tekstu. Działa ona rekurencyjnie, pozbywając się zbędnych części i szukając najdłuższej podgrupy, która ma swój odpowiednik w bazie WordNet. Wykorzystuje ona następujące elementy:

lemmatise() zwraca lemat grupy nominalnej.
inWordNet() sprawdza, czy dla danego łańcucha znaków istnieje leksem o takiej nazwie w WordNecie.
getLexemes() zwraca listę leksemów WordNet o podanej nazwie.
allSynsets() zwraca wszystkie synsety skojarzone z różnymi wariantami podanego leksemu.
isCoordination() sprawdza, czy dany element stanowi grupę koordynacyjną.
isGroup() sprawdza, czy dany element jest grupą.
semanticHead to wyróżniony element grupy, tj. centrum semantyczne.

W powstałym modelu pytania zachowujemy informację o dopasowanym synsecie centrum (składnik $model.type.focus$). Zostanie ona wykorzystana w technice głębokiego rozpoznawania nazw (rozdział 4) – w naszym przykładzie pozwoli to na branie pod uwagę tylko występujących w tekstach łodzi podwodnych, a nie wszystkich pojazdów.

Realizacja uczenia maszynowego

Jako punkt odniesienia dla oceny osiągniętych rezultatów, zaimplementowano również proste rozwiązanie bazujące na uczeniu maszynowym. Trzeba zwrócić uwagę, że ze względu na odmienną strukturę zdań rozwiązania opracowane dla języka angielskiego mogą w przypadku polskiego dawać gorsze rezultaty. Przykładowo rozważmy dwie wersje tego samego pytania w języku angielskim:

```
Input: chunk – grupa nominalna lub pojedynczy rzeczownik
Output: synsets – synsety WordNet wydobyte na podstawie chunk
begin
    lemma := lemmatise(chunk);
    if inWordNet(lemma) then
        return getLexemes(lemma).allSynsets();
    else if isCoordination(chunk) then
        synsets := {};
        foreach element in chunk do
            synsets := synsets ∪ extractSynsets(element);
        end
        return synsets;
    else if isGroup(chunk) then
        return extractSynsets(chunk.semanticHead);
    else
        return {};
end
```

Algorytm 3: Funkcja **extractSynsets**(*chunk*) – rekurencyjnie wydobywa synsety z grupy nominalnej lub rzeczownika.

- czynną: *Which composer created the Jupiter Symphony?*,
- bierną: *By which composer was the Jupiter Symphony created?*

Posiadają one 6 wspólnych słów i dwa odmienne, co odpowiada dużemu podobieństwu w modelu *bag of words*. Jest to zgodne z oczekiwaniami, skoro oznaczają dokładnie to samo. Dla porównania przeanalizujmy te wersje w języku polskim:

- czynną: *Który kompozytor napisał Symfonię "Jowiszową"?*,
- bierną: *Przez którego kompozytora została napisana Symfonia "Jowiszowa"?*

Nie zawierają one ani jednego takiego samego słowa (w sensie łańcucha znaków). Dla uzyskania niezawodnej oceny zgodności konieczny staje się zatem steming lub lematyzacja.

Przed zastosowaniem klasyfikatora każde pytanie przetworzono analizatorem morfosyntaktycznym i tagerem, opisanymi w punkcie 3.2.2. Na tej podstawie powstały cechy odpowiadające:

- występowaniu lematów w zdaniu,
- interpretacjom morfologicznym pierwszych pięciu słów.

Cechy z obu grup mają charakter binarny, np. cecha `kot` przyjmuje wartość `1`, jeśli w zdaniu wystąpiło przynajmniej jedno słowo o takim lemacie. Cechy morfologiczne uzupełnia się o pozycję, jaką słowo o danym tagu zajmuje w zdaniu, na przykład cecha `adj.sg.nom.m1.pos.0` ma wartość `1`, jeśli pierwszym słowem zdania jest przymiotnik rodzaju męskiego osobowego

Klasyfikator	Pokrycie	Precyzja	Ogółem
dopasowywanie wzorców	36,15%	95,37%	34,48%
wzorce + semantyczna analiza centrum	98,33%	80,95%	79,60%
drzewo decyzyjne	100%	67,02%	67,02%
las losowy	100%	72,91%	72,91%

Tabela 3.6. Wyniki czterech badanych klasyfikatorów pytań: udział sklasyfikowanych przypadków (pokrycie), udział poprawnych etykiet (precyzja) i iloczyn tych dwóch wskaźników, czyli udział poprawnej klasyfikacji w całości zbioru (ogółem).

w liczbie pojedynczej, mianowniku i stopniu równym. Dla pierwszego zdania z tabeli 3.4 otrzymamy następujące wartości cech:

- `z`=1,
- `który`=1,
- `państwo`=1,
- `graniczyć`=1,
- `Lesotho`=1,
- `?`=1,
- `prep.inst.nwok.0`=1,
- `adj.pl.inst.n.pos.1`=1,
- `pl.inst.n.2`=1,
- `sg.ter.imperf.3`=1,
- `sg.gen.n.4`=1,
- pozostałe=0.

W ramach eksperymentów wstępnych podejmowane były też próby z uwzględnianiem informacji semantycznych w postaci hiperonimów słów z pytania, ale powodowało to gwałtowny rozrost przestrzeni cech, co przy stosunkowo niewielkim zbiorze treningowym znacząco pogarszało wydajność klasyfikacji.

Wykorzystano dwa klasyfikatory: ze względu na interpretowalne rezultaty zastosowano przycinane drzewa decyzyjne, zaś lasy losowe wprowadzono z powodu dobrych wyników uzyskiwanych przy ich wykorzystaniu do analizy problemów wielowymiarowych. Użyto ich implementacji w środowisku R (R Core Team, 2013) – drzew z pakietu `mvpart` (przycinanie według reguły *1-se*) i lasów (z 500 drzew) z pakietu `randomForest` (Liaw i Wiener, 2002).

Ewaluacja

Do oceny jakości opracowanych rozwiązań wykorzystano zbiór 1130 pytań opisany w punkcie 5.1.2. Porównano kategorie przypisane przez klasyfikatory z typami ogólnymi i nazw własnych określonymi ręcznie. Ponieważ dla pewnych przypadków klasyfikator regułowy może nie przydzielić żadnej kategorii, do oceny wydajności zastosowano dwie miary:

- pokrycie (ang. *recall*), tj. udział pytań, do których przypisano jakikolwiek typ,
- precyzję (ang. *precision*), tj. udział poprawnych etykietowań w zbiorze pytań z przypisanym typem.

Przyczyny błędnej klasyfikacji	Liczba
Wieloznaczne centrum pytania	99
Wybór typu egzemplarza zamiast ogólnej grupy	55
Brak reguł dla typu `OTHER_NAME`	14
Uboga detekcja typu `MULTIPLE`	12
Zbyt złożona składnia pytania	12
Niewystarczające wzorce	11
Niedostateczna lista synsetów przypisanych do typu	10
WordNet nie zawiera potrzebnych relacji hiperonimii	8
Błąd tagowania	6
Błąd płytkiego parsowania	1

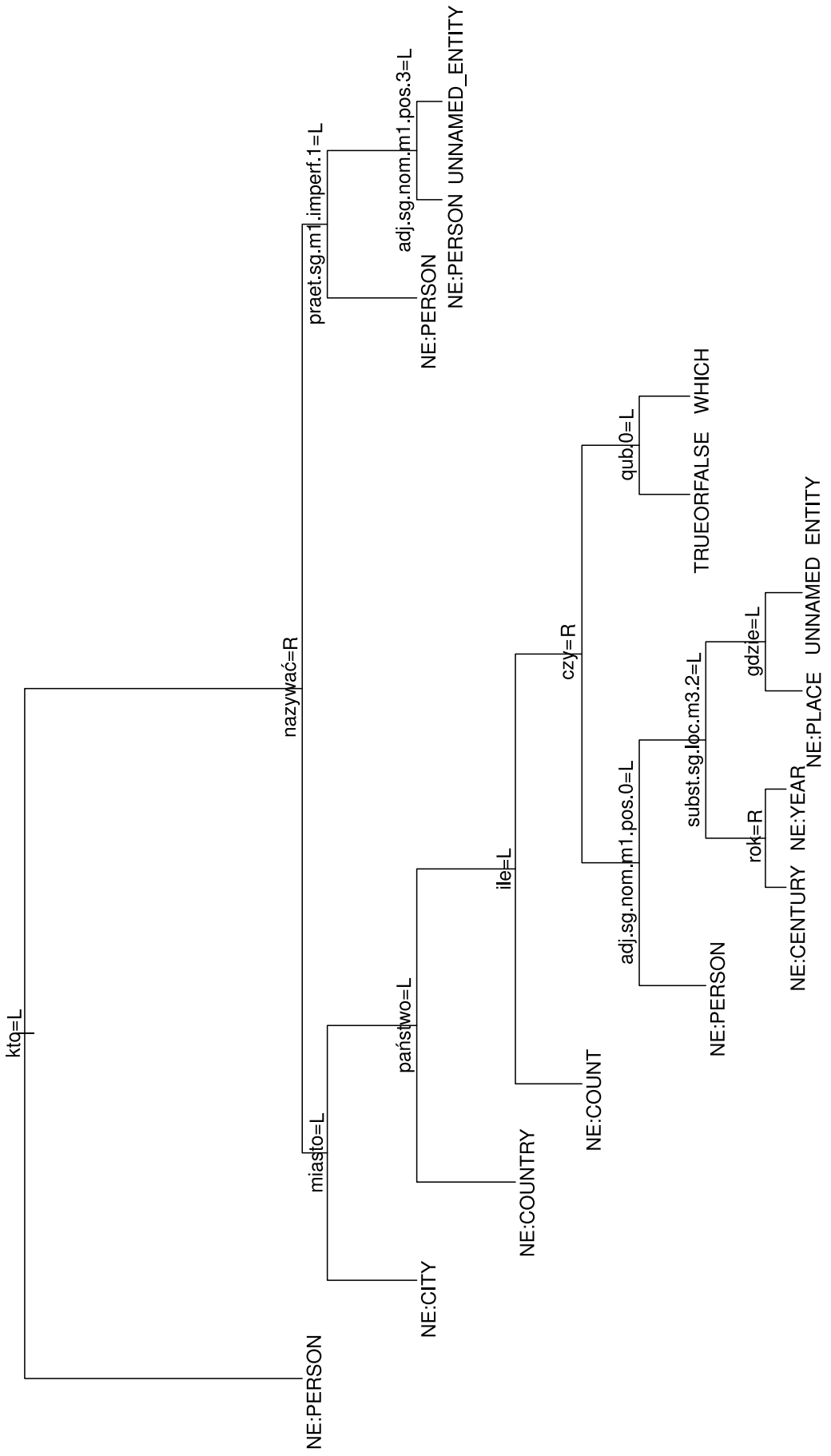
Tabela 3.7. Przyczyny błędnej klasyfikacji pytań w klasyfikatorze z semantyczną analizą centrum według liczby przypadków, w których wystąpiły.

Tabela 3.6 zawiera wyniki dwóch technik regułowych: dopasowywania wzorców i dopasowywania wzorców z analizą centrum pytania oraz dwóch technik uczenia maszynowego: drzew decyzyjnych i lasów losowych[17]. Rezultaty wyrażono w wartościach pokrycia, precyzji i udziału poprawnych klasyfikacji ogółem. Jak widać, jednoznaczne wzorce udaje się dopasować jedynie do 36% pytań, natomiast w tych przypadkach otrzymujemy bardzo wysoką precyzję. Dołączenie do zbioru wzorców niejednoznacznych sprawia, że niemalże dla każdego pytania udaje się wskazać typ. Określanie typu poprzez analizę centrum wiąże się jednak z ryzykiem popełniania błędu – co piąty wynik jest nieprawidłowy.

Niestety, uczenie maszynowe nie daje w tym przypadku bardzo dobrych rezultatów. Zgodnie z oczekiwaniami, las losowy osiąga wyższą precyzję od drzewa decyzyjnego, jednak jest to wciąż mniej niż podejścia regułowe. W przypadku tego zadania poziom odniesienia, czyli udział najliczniejszej klasy, wynosi 33%. Na rysunku 3.3 przedstawiono przycięte drzewo decyzyjne, zbudowane na całych danych treningowych. Obok oczywistych zależności, np. że wystąpienie słowa *kto* wskazuje na typ `NAMED_ENTITY:PERSON`, znajdziemy także bardziej złożone, np. że pojawienie się w zdaniu słowa *czy* oznacza typ `TRUEORFALSE`, gdy zajmuje ono pierwszą pozycję i `WHICH` w przeciwnym wypadku. Można wskazać także reguły trudniejsze do zrozumienia, np. użycie cechy `adj.sg.nom.m1.pos.3` w skrajnym prawym odgałęzieniu drzewa oznacza uzależnienie decyzji od tego, czy czwarte słowo zdania jest przymiotnikiem rodzaju męskoosobowego w liczbie pojedynczej, mianowniku i stopniu równym. Warto zwrócić uwagę, że zastosowany zestaw cech jest bardzo prosty – wydaje się, że jego rozbudowanie np. o informacje semantyczne przy jednoczesnym filtrowaniu przestrzeni cech mogłoby przynieść wzrost wydajności.

Aby poprawić rezultaty klasyfikacji, poddano analizie przypadki błędnego przypisania kategorii przez klasyfikator regułowy i określono przyczyny niepowodzenia. Wyniki przedstawia tabela 3.7. Wyróżniają się dwa problemy: wieloznaczność centrum i wybór między egzemplarzem a klasą. Drugi

[17] Dla wiarygodności wyniku rezultaty uczenia maszynowego sprawdzano w drodze 100-krotnej walidacji krzyżowej.



Rysunek 3.3. Przycięte drzewo decyzyjne zbudowane w oparciu o dane treningowe. Wartość `L` oznacza, że wystąpienie danej cechy determinuje pójście prawą gałęzią, zaś `R` – lewą.

Sposób łączenia zbiorów hiperonimów	Precyzja klasyfikacji
Suma zbiorów	80,95%
Część wspólna zbiorów	72,00%
Głosowanie	79,07%
Tylko pierwsze znaczenie	84,35%

Tabela 3.8. Precyzja określania typu pytania w klasyfikatorze z analizą centrum w zależności od sposobu łączenia list hiperonimów pochodzących od różnych znaczeń słowa.

z nich możemy zilustrować następującą parą pytań: *Jaki pies jako pierwszy poleciał w kosmos?* i *Jaki pies ma najsilniejszy uścisk szczęki?*. Pierwsze pytanie odnosi się do imienia konkretnego egzemplarza klasy *pies*, co odpowiada typowi `ANIMAL`, zaś drugie do ogólnej grupy, tj. rasy, czyli należy do typu `UNNAMED_ENTITY`. Jak widać, nie sposób rozstrzygnąć tej wątpliwości na podstawie analizy centrum pytania, skoro w tych przypadkach jest ono takie samo. Problem ten nie ma znaczenia w przypadku techniki głębokiego rozpoznawania nazw (rozdział 4), w której zarówno słowu *Łajka*, jak i *pudel*, przypisane zostaną synsety połączone łańcuchem hiperonimii z synsetem *pies*. Ponadto w takich przypadkach nawet człowiekowi może być trudno określić intencje pytającego (tj. jakiej informacji oczekuje) bez znajomości kontekstu.

Najpoważniejsza przyczyna problemów to wieloznaczność centrum. Wiąże się ona z faktem, że część słów ma wiele znaczeń, a zatem badany leksem może występować w WordNecie w kilku wariantach, należących do różnych synsetów. Jako przykład rozważmy pytanie *Jaka jednostka służy do pomiary natężenia prądu?* Stanowiącemu centrum pytania słowu *jednostka* odpowiada w bazie *plWordNet* pięć synsetów. Wśród nich znajduje się oczekiwany <jednostka.4, jednostka miary.1, miano.2, miara.3>, ale również np. <jednostka.2, człowiek.1, osoba.1, istota ludzka.1, persona.2>. Synsety te mają oczywiście różne listy hiperonimów i, co z tego wynika, mogą prowadzić do różnych typów nazw własnych lub ogólnych.

Skoro nie wiemy, które ze znaczeń jest właściwe w konkretnym przypadku, poszukujemy takiego sposobu na scalenie zbiorów hiperonimów, który uwzględni różne możliwości. Można wskazać następujące opcje:

- **suma**, czyli branie pod uwagę hiperonimów wszystkich znaczeń (takie rozwiązanie stanowi punkt wyjścia przedstawiony w algorytmie 3),
- **przecięcie**, czyli branie pod uwagę jedynie hiperonimów wspólnych dla wszystkich znaczeń,
- **głosowanie**, czyli uwzględnianie hiperonimów występujących dla większości znaczeń,
- **wybór pierwszego znaczenia**, przy którym bierzemy pod uwagę tylko listę hiperonimów pierwszego wariantu leksemu, który zwykle odpowiada najczęstszemu znaczeniu.

Przedstawione warianty przetestowano na wspomnianych wyżej danych treningowych według tego samego schematu i obliczono precyzję klasyfikacji. Wyniki zawiera tabela 3.8. Przyjęta początkowo strategia uwzględniania sumy

nie jest najgorszą, ale jeszcze lepsze wyniki osiągamy wybierając po prostu pierwszy wariant leksemu.

Podsumowując, dla zadania klasyfikacji pytań najlepsze wyniki otrzymano stosując podejście regułowe z dopasowywaniem wzorców, wsparte semantyczną analizą centrum pytania uwzględniającą tylko pierwsze jego znaczenie. Taka też wersja klasyfikatora pytań była używana przy ewaluacji systemu RAFAEL, opisanej w rozdziale 5.

3.3.2. Generowanie zapytania

Jak wspomniano w punkcie 3.2.1, w procesie odpowiadania na pytania ze względów wydajnościowych całościowej analizie podlega jedynie pewien podzbiór korpusu, wybrany na podstawie tematycznej zgodności z pytaniem. Aby możliwe było dokonanie tej wstępnej selekcji, konieczna jest konwersja pytania na zapytanie do wyszukiwarki pełnotekstowej – tę rolę w RAFAELu pełni *Lucene*.

Możliwe rozwiązania problemu

Zagadnieniu wstępnej selekcji dokumentów poświęca się niewiele uwagi w pracach z obszaru QA, podczas gdy ma ono kluczowe znaczenie dla odnalezienia odpowiedzi. Jeżeli bowiem zbiór stanowiący wynik działania tego etapu nie będzie zawierał dokumentu z odpowiedzią, to system jej nie odnajdzie, niezależnie od złożoności wykorzystywanych w dalszym przetwarzaniu metod.

Zapytanie powstaje na ogół przez przekształcenie pytania na listę słów kluczowych połączonych operatorem Boole'owskim. Jeśli zastosujemy tutaj koniunkcję, to ograniczymy się tylko do dokumentów wykazujących bardzo duże podobieństwo do pytania. Takie podejście ma sens w przypadku korzystania z zasobu tekstów o dużej redundancji, jak np. sieć WWW. Tego typu rozwiązanie, bazujące na wyszukiwarce *Google*, zaprezentowali Brill i inni (2002).

Kiedy jednak mamy do dyspozycji mniejsze zbiory tekstowe, tak utworzone zapytanie może nie pasować do żadnego z dokumentów. Trzeba zatem wziąć pod uwagę inne sformułowania żądanej informacji. Korzysta się z bardziej liberalnej konstrukcji z operatorem alternatywy, jak również różnych technik modyfikacji zapytania.

Po pierwsze, można usuwać mniej istotne spośród słów kluczowych. W systemie *LASSO* (Moldovan i inni, 2000) zastosowano aż osiem heurystyk wskazujących porządek istotności słów. Jako ważne wybiera się po kolei:

1. wyrażenia w cudzysłowach,
2. rozpoznane nazwy własne,
3. wieloskładnikowe grupy nominalne,
4. modyfikatory przymiotnikowe grup,
5. pojedyncze rzeczowniki z modyfikatorami,
6. wszystkie rzeczowniki,
7. czasowniki,
8. centrum pytania.

W systemie łączącym techniki wyszukiwania w sieci WWW i własnym korpusie tekstów (Katz i inni, 2003) słowa kluczowe usuwa się zgodnie z rosną-

cym wskaźnikiem IDF (ang. *Inverse Document Frequency*), a więc w pierwszej kolejności pozbywamy się takich, które występują powszechnie. W rozwiązaniu dla języka słoweńskiego (Čeh i Ojsteršek, 2009) usuwane są ciągi słów z początku lub końca zdania, do momentu aż pozostały fragment zostanie rozpoznany jako klucz obecny w indeksie.

W niektórych systemach rozszerza się zapytanie o formy pochodne obecnych w nim słów, co ma szczególne zastosowanie w przypadku języków fleksyjnych. Takie rozwiązanie, zarówno w odniesieniu do odmiany, jak i derywacji, odnajdziemy w już wspominanej pracy (Katz i inni, 2003). Bardziej interesującą technikę zastosowano w systemie *Webclopedia* (Hovy i inni, 2000) – tam do zapytania dodawane są pobrane z WordNetu synonimy słów, które wystąpiły w pytaniu.

Implementacja w RAFAELu

Implementacja procesu generowania zapytania w systemie RAFAEL wykorzystuje przedstawione powyżej rozwiązania. Punkt wyjścia stanowi połączenie operatorem alternatywy wszystkich niekonstrukcyjnych słów pytania, czyli tych objętych pierwszą i trzecią parą nawiasów we wzorcach umieszczonych w tabelach 3.2 i 3.3. Nie wykorzystano żadnych technik ważenia słów ani usuwania *stop-words*, ponieważ wyszukiwarka *Lucene* bazuje na wskaźniku TF/IDF (Papineni, 2001), który uwzględnia niższy priorytet powszechnych słów. W innym wariancie używa się operatora koniunkcji, a jeśli w wyniku nie są zwracane żadne dokumenty, to ponawia się wyszukiwanie z alternatywą.

Pewnej uwagi wymaga usuwanie centrum pytania, zastosowane na przykład w systemie *LASSO* (Moldovan i inni, 2000). Rozważmy ponownie zdanie *Który francuski postimpresjonista uciął sobie ucho?*. Stanowiąca tu centrum grupa nominalna *francuski postimpresjonista* nie musi się pojawić w dokumencie, zastąpiona po prostu przez poszukiwane nazwisko. Z drugiej strony, jej obecność może stanowić pomocną wskazówkę.

Dodatkowo, korzystanie z bazy WordNet umożliwia również uwzględnianie synonimów: każde słowo rozpoznane jako leksem zostaje zastąpione zagnieżdżoną alternatywą wszystkich leksemów z synsetów odpowiadających możliwym jego znaczeniom.

Odrębnym problemem jest fleksyjny charakter języka polskiego. Oznacza on, że forma ortograficzna słowa w pytaniu może różnić się od tej obecnej w tekście, jeśli tylko różnią się ich role składniowe w zdaniach, do których należą. Możliwe są cztery rozwiązania tej kwestii:

- wyszukiwanie form fleksyjnych,
- pełne dopasowanie,
- steming,
- zapytania rozmyte.

Wyszukiwanie form fleksyjnych polega na tym, że każde słowo w zapytaniu zastępuje się przez zapytanie zagnieżdżone, zawierające wszystkie możliwe formy fleksyjne tego słowa połączone operatorem alternatywy. Niestety, dla języka o bogatej fleksji, takiego jak polski, powoduje to wielokrotne wydłużenie zapytania i spowolnienie wyszukiwania. Z tego powodu wyszukiwania form fleksyjnych nie uwzględniono w budowie zapytań w systemie RAFAEL.

Pełne dopasowanie oznacza, że za zgodne z zapytaniem uznaje się tylko słowa o tych samych formach ortograficznych. Stemming, jak wspomniano w części 3.2.1, jest zaimplementowany w *Lucene* (Galambos, 2001) i został uwzględniony przy budowie indeksu. Przy jego użyciu zarówno słowa w zapytaniu, jak i w tekście podlegają tej samej transformacji do ciągu znaków zbliżonego do rdzenia słowa i dopiero w tej postaci zostają porównane. Inną opcję stanowią dostępne w tym samym systemie zapytania rozmyte (ang. *fuzzy queries*). Słowo w tekście uznaje się za zgodne ze słowem w takim zapytaniu wtedy, jeśli odległość edycyjna Levenshteina (1966) pomiędzy nimi jest mniejsza od zadanego progu. Problem ten nie odgrywa dużej roli w języku angielskim. Większość autorów, np. Hovy i inni (2000), używa prostego stemera, np. Portera (1980). Dla języków o bardziej rozwiniętej fleksji rzeczownikowej stemer nie sprawdza się tak dobrze. Zastosowano go np. w przypadku systemu QA dla języka bułgarskiego (Peshterliev i Koychev, 2011), gdzie odmiana przez przypadki występuje w bardzo zredukowanej formie.

Na sposób działania zapytań rozmytych wpływa parametr określający dopuszczalną maksymalną odległość edycyjną między dopasowywanymi łańcuchami. Optymalna jego wartość zależy od własności języka implementacji, w tym wypadku polskiego. Ograniczenie to można określić na trzy różne sposoby:

- względnie,
- bezwzględnie,
- bezwzględnie ze stałym prefiksem.

W trybie względnym maksymalna odległość wyrażona jest jako ułamek długości słowa (np. 30%) – im zatem ono dłuższe, tym więcej liter może się różnić. W trybie bezwzględnym określa się po prostu pewną stałą liczbę zmiennych liter, niezależną od długości słowa. W rzeczywistości *Lucene* nie udostępnia takiej możliwości, jednak można ją łatwo symulować, wyliczając dla każdego słowa kluczowego indywidualną wartość rozmycia względnego na podstawie jego długości[18]. Tryb bezwzględny ze stałym prefiksem działa analogicznie jak poprzedni, jednak zmiany w ramach dozwolonej odległości k mogą dotyczyć tylko ostatnich k znaków słowa, tj. dla słowa o długości l aby doszło do dopasowania pierwsze $(l - k)$ znaków musi być identycznych. Motywowane jest to głównie względami wydajnościowymi, jednak odpowiada również własnościom języka polskiego, w którym różnice morfologiczne znacznie częściej dotyczą końcówek słów niż początków.

Przedstawiona jako algorytm 4 funkcja *generateQuery* realizuje opisaną wyżej procedurę. Przyjmuje ona na wejściu składniki modelu pytania: treść i centrum, a zwraca zapytanie gotowe do przekazania modułowi wyszukiwania na bazie *Lucene*. Dodatkowo utworzone zapytanie zależy od szeregu parametrów, określających jego budowę ogólną i sposób dopasowywania składników. Znaczenie nazw obecnych w pseudokodzie jest następujące:

newLuceneQuery() tworzy puste zapytanie *Lucene*.

[18] W procedurze tej przyjęto, że wielkość rozmycia nie może stanowić więcej niż połowy długości słowa.

```
Input: content – łańcuch znaków zawierający treść pytania
Input: focus – łańcuch znaków zawierający centrum pytania
Input: firstAnd – określa, czy zapytanie powinno być koniunkcją
Input: focusRemoval – określa, czy z zapytania będzie usuwane
       centrum
Input: synonyms – określa, czy zapytanie będzie zawierać synonimy
Input: matchingMode – tryb dopasowywania (PLAIN, STEMMING
       lub FUZZY)
Input: fuzziness – dopuszczalny stopień rozmycia zapytania: ułamek
       długości słowa dla rozmycia względnego lub całkowita liczba
       znaków dla bezwzględnego
Input: fixedPrefix – określa, czy przy rozmyciu ma zostać zachowany
       stały przedrostek
Output: query – utworzone zapytanie
begin
    query = newLuceneQuery();
    if firstAnd then query.operator=AND;
    else query.operator=OR;
    if mode==STEMMING then query.index=stemmed;
    else query.index=plain;
    if focusRemoval then content.remove(focus);
    foreach word in split(content) do
        keywords={ word };
        if synonyms and inWordNet(word) then
            foreach synset in getLexemes(word).allSynsets() do
                keywords.addAll(synset.lemmas);
            end
        end
        term=newLuceneTerm();
        if length(words)>1 then term.operator=OR;
        foreach keyword in keywords do
            if mode == PLAIN or mode == STEMMING then
                term.add(keyword);
            else if mode == FUZZY then
                if fuzziness<1 then
                    term.addFuzzy(keyword,fuzziness,fixedPrefix);
                else
                    term.addFuzzy(keyword,word.length/fuzziness,
                    ↪ fixedPrefix);
                end
            end
        end
        query.add(term);
    end
    return query;
end
```

Algorytm 4: Funkcja **generateQuery**() – tworzy zapytanie *Lucene* na podstawie treści pytania.

query.index określa, do którego z dwóch indeksów opisanych w punkcie 3.2.1 zostanie skierowane zapytanie *query*.

remove() usuwa z danego ciągu znaków jego fragment – w tym wypadku centrum pytania z treści.

split() dzieli łańcuch znaków na fragmenty według wystąpienia znaków odstępu.

inWordNet() ***getLexemes()*** i ***allSynsets()*** mają takie samo znaczenie, jak w funkcji *extractSynsets* (algorytm 3).

newLuceneTerm() tworzy pusty *term*, czyli fragment zapytania, mogący być pojedynczym słowem kluczowym lub ich sekwencją z operatorem (w przypadku synonimów).

addFuzzy(keyword,fuzziness,fixedPrefix) dodaje do *termu* słowo kluczowe *keyword*, które może być dopasowane z rozmyciem względnym o wielkości podanej w parametrze $fuzziness$. Parametr *fixedPrefix* określa, czy przy dopasowywaniu ma być zachowany stały przedrostek.

Prześledźmy wykonanie funkcji na przykładzie wspomnianego pytania *Który francuski postimpresjonista uciął sobie ucho?*

1. Klasyfikator pytania dopasowałby do niego niejednoznaczny wzorzec z tabeli 3.3:
   `^()(Który)[ ,](.*)\?$`.
2. Zgodnie z algorytmem 2, treść pytania stanowią jego fragmenty obejmowane przez pierwszą i trzecią parę nawiasów wzorca, czyli w tym przypadku otrzymujemy tekst:
   `francuski postimpresjonista uciął sobie ucho`
3. Podstawowa wersja zapytania wygląda następująco (w *Lucene* zapis `()~k` oznacza, że spośród słów w nawiasach przynajmniej `k` musi zostać dopasowane w odnalezionym tekście):
   `(francuski postimpresjonista uciął sobie ucho)~1`
4. Dalsze modyfikacje zależą od tego, który wariant budowy zapytania jest realizowany:
   a) Z usunięciem centrum pytania:
      `(uciął sobie ucho)~1`
   b) Z operatorem koniunkcji:
      `(francuski postimpresjonista uciął sobie ucho)~5`
   c) Z synonimami:
      `(francuski postimpresjonista uciął sobie (ucho uszko nausznik)~1)~1`
5. Niezależnie od powyższego wyboru, możliwe są również różne strategie dopasowywania słów (przykłady dla wariantu podstawowego):
   a) Według form ortograficznych:
      `(francuski postimpresjonista uciął sobie ucho)~1`
   b) Z użyciem stemingu:
      `(francuski postimpresjonista uciąć ucho)~1`
   c) Z dopasowaniem rozmytym (rozmycie względne wielkości 30%[19]):

[19] W konwencji zapisu *Lucene* liczba przy słowie kluczowym oznacza, jaka część jego długości musi pozostać niezmienna.

```
(francuski~0.7 postimpresjonista~0.7 uciął~0.7
sobie~0.7 ucho~0.7)~1
```

d) Z dopasowaniem rozmytym (rozmycie bezwzględne o długości 3 znaków)[20]:

```
(francuski~0.6666667 postimpresjonista~0.8235294
uciął~0.7 sobie~0.4 ucho~0.5)~1
```

Już na tak prostym przykładzie widać podstawowy problem związany z generowaniem synonimów na podstawie bazy WordNet – brak ujednoznaczniania sensów słów powoduje, że do zapytania zostają włączone synonimy niewłaściwych znaczeń. Problem ten może być jednak skorygowany na etapie poszukiwania odpowiedzi, gdyż miara oceny wzmianek (punkt 3.5.2) synonimów nie uwzględnia i wyższy wynik otrzymają zdania zawierające słowo *ucho*, a nie *nausznik*.

Ewaluacja

Omawiany moduł zaimplementowano z uwzględnieniem różnych opcji budowy zapytania: priorytetu operatora koniunkcji, usuwania centrum pytania, dodawania synonimów; jak również różnych strategii dopasowywania słów. Konieczne stało się sprawdzenie, która wersja przyniesie najlepsze rezultaty w danych warunkach – w szczególności dla tekstów w języku polskim. W tym celu ponownie wykorzystano zbiór treningowy, przedstawiony w części 5.1.2. Do każdego z 1057 pytań[21] ręcznie przypisano jeden dokument wzorcowy, zawierający odpowiedź.

Dla poszczególnych wariantów zapytania wykonywano wyszukiwanie i obserwowano pozycję oczekiwanego dokumentu na liście rankingowej, zwróconej przez *Lucene*. Nie mają tu zastosowania miary przywiązujące dużą wagę do pozycji dokumentu w rankingu (np. MRR), ponieważ RAFAEL pobiera dużą grupę dokumentów ze szczytu listy i traktuje je jednakowo. Wobec tego wskaźnikiem wydajności strategii tworzenia zapytań jest udział pytań, dla których oczekiwany dokument znalazł się wśród pierwszych N, gdzie N jest liczbą dokumentów branych pod uwagę w dalszej analizie. Określenie optymalnej wartości tej liczby było celem jednego z eksperymentów przeprowadzanych na całości systemu, opisanych w podrozdziale 5.3. Dla potrzeb testowania wyodrębnionego modułu generowania zapytania przyjęto $N = 100$. Średnią pozycję w ramach tej setki zamieszczono wyłącznie w celach poglądowych. Tak zdefiniowane pokrycie jest miarą nadmiernie pesymistyczną, gdyż w zbiorze mogą istnieć dokumenty inne niż wzorcowy, a zawierające odpowiedź na pytanie. Żeby mieć całkowicie pewną ocenę pokrycia, należałoby ręcznie określić wszystkie dokumenty zawierające odpowiedź, jednak dla korpusu tej wielkości stanowi to zadanie niezwykle czasochłonne.

Uzyskane wyniki przedstawione są w tabeli 3.9. Co może wydać się zaskakujące, najlepsze rezultaty osiągane są dla wersji podstawowej, tj. prostego zapytania typu OR. Wprowadzenie operatora koniunkcji znacząco zmniejsza

[20] Postać tekstowa wygląda identycznie, niezależnie, czy zostaje zachowany stały prefiks czy też nie.

[21] Dla pozostałych taki dokument nie istniał, tzn. odpowiedź wymagała połączenia treści z kilku dokumentów.

Zapytanie \ Dopasowanie	Dokładne	Steming	Rozmyte
podstawowe – alternatywa	69,97% 14,32	80,08% 12,90	**82,19%** **12,36**
najpierw koniunkcja	57,94% 11,36	57,07% 8,80	34,84% 7,07
usuwanie centrum	62,75% 14,65	71,99% 14,00	73,34% 12,84
dodane synonimy	47,06% 21,42	65,64% 15,47	58,71% 16,00

Tabela 3.9. Wydajność wyszukiwania dla czterech strategii modyfikacji zapytania i trzech technik dopasowywania. Liczby odzwierciedlają pokrycie, tj. udział pytań, dla których oczekiwany dokument znalazł się wśród pierwszych 100, i średnią pozycję tego dokumentu na liście.

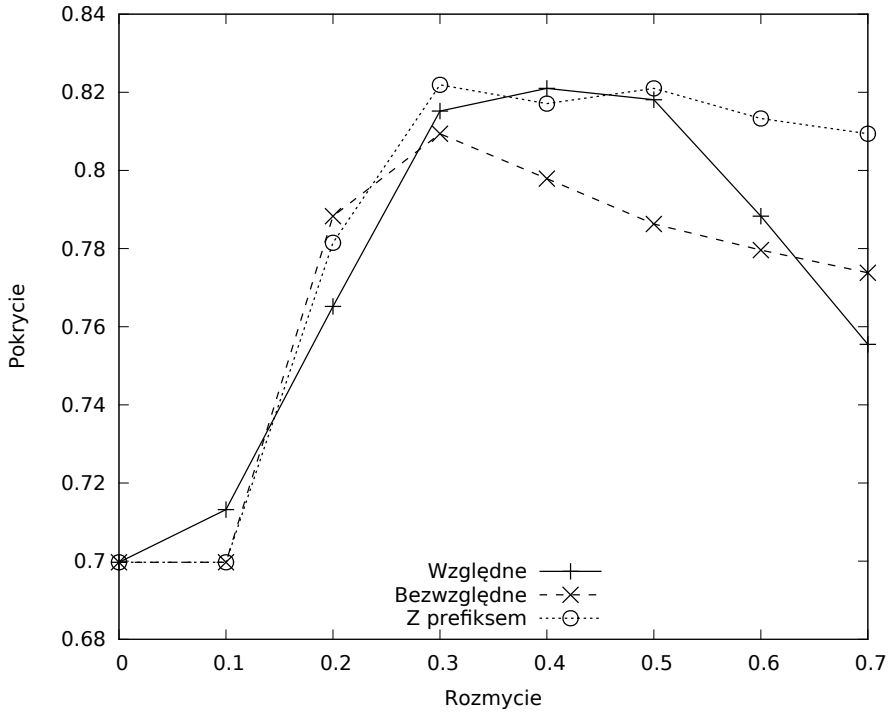
pokrycie – korpus jest zbyt mały, by można było liczyć na to, że znajdziemy dokument zawierający identyczne sformułowania. W zamian otrzymujemy co prawda znacznie lepszą pozycję na liście rankingowej, ale dla systemu nie ma ona znaczenia. Także usunięcie centrum pytania i dodanie synonimów zauważalnie pogarsza rezultaty. W tym ostatnim przypadku daleka średnia pozycja na liście pozwala zrozumieć, skąd bierze się problem – pojawia się bardzo wiele dokumentów pasujących do zapytania i system rangowania *Lucene* nie określa ich kolejności optymalnie. Być może wzięcie pod uwagę liczniejszego niż 100 elementów zbioru uczyniłoby tę modyfikację uzasadnioną.

Jeśli chodzi o sposób dopasowywania słów, podejście dokładne przynosi najsłabsze wyniki – znów gra tu rolę mała wielkość i redundancja korpusu. Proste zapytania rozmyte zaowocowały lepszym pokryciem niż korzystający z wiedzy lingwistycznej steming. Należy jednak zwrócić uwagę, że zastosowany stemer *Stempel* ustępuje innym rozwiązaniom dla języka polskiego (Weiss, 2005). Zdecydowanie najlepiej sprawdzają się zapytania rozmyte. W tabeli 3.9 uwzględniono wyniki dla najlepszej konfiguracji rozmycia, natomiast rysunek 3.4 pokazuje pokrycie dla wyszukiwania przy różnych typach i wielkościach rozmycia. Widać, że najlepsze rezultaty uzyskano dla rozmycia bezwzględnego o długości 3 znaków ze stałym prefiksem. Ten rodzaj rozmycia jest przy tym najmniej kosztowny obliczeniowo, co korzystnie wpływa czas działania systemu.

3.3.3. Wyodrębnianie treści pytania

Ostatnim krokiem analizy pytania jest wyodrębnienie treści, czyli wskazanie tych słów, które będą porównywane z kontekstem odpowiedzi według określonej miary podobieństwa. Mamy tu do czynienia z podobnymi uwarunkowaniami, co przy tworzeniu zapytania do wyszukiwania, więc zastosowano ten sam schemat – pobierane są słowa, które nie należą do konstrukcji pytania, czyli są obejmowane przez pierwszą i trzecią parę nawiasów w tabelach 3.2 i 3.3.

Podobnie jak poprzednio, także i tu należy uwzględnić problem usuwania



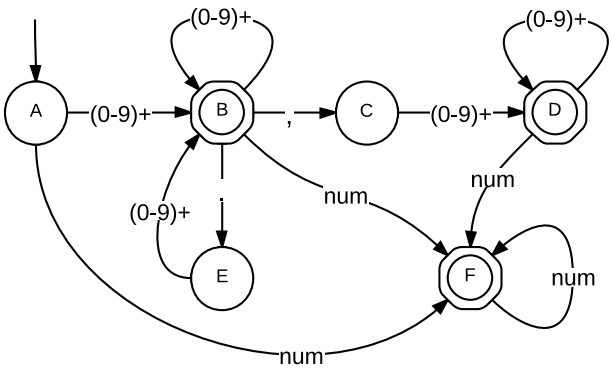
Rysunek 3.4. Pokrycie, czyli udział pytań, dla których odnaleziono oczekiwany dokument, w zależności od typu rozmycia i jego wielkości. Dla przeliczenia wartości względnych na bezwzględne założono długość słowa 10.

centrum pytania. Wydaje się, że o ile jego treść może pojawić się w odnalezionym dokumencie, to raczej nie w bezpośrednim kontekście odpowiedzi, dlatego centrum jest usuwane z treści pytania. Dla pytania *Który francuski postimpresjonista uciął sobie ucho?* otrzymamy treść w postaci następującego zbioru: {*uciął*, *sobie*, *ucho*}. Dzięki temu zdanie *Vincent van Gogh uciął sobie ucho będąc w domu publicznym.* zostanie ocenione wysoko, chociaż nie zawiera określenia *francuski postimpresjonista*.

3.4. Oznaczanie wzmianek

Zidentyfikowanie typu pytania oznacza, że wiadomo już, jakiego typu informacji należy szukać w bazie tekstów. Przypomnijmy, że RAFAEL odnajduje odpowiedzi na pytania należące do klasy pytań o nazwy bytów, co odpowiada typowi nazwy własnej (`NAMED_ENTITY`) lub ogólnej (`UNNAMED_ENTITY`). Odpowiedzi na nie można udzielić poprzez wskazanie jednej z nazw wyselekcjonowanych wstępnie w dokumentach. Gdy otrzymane pytanie nie należy do tej klasy, RAFAEL nie jest w stanie udzielić odpowiedzi.

Na tym etapie system analizuje wszystkie dokumenty, które zostały zwrócone jako rezultat wyszukiwania, i wskazuje wzmianki odnoszące się do bytów należących do szukanego typu. Można do tego problemu zastosować trzy podejścia – tradycyjne rozpoznawanie nazw własnych, będące głównym przedmiotem tej pracy głębokie rozpoznawanie nazw, oraz podejście hybrydowe, łączące wzmianki pochodzące z obu źródeł.



Rysunek 3.5. Automat służący do rozpoznawania liczb w module *Quant*. Stany końcowe otoczono podwójnym okręgiem.

Rezultatem omawianego etapu jest przypisana do każdego dokumentu lista wzmianek, z których każda obejmuje jeden lub więcej segmentów z treści tego dokumentu, tworzących nazwę bytu. Ponieważ jedna z odnalezionych wzmianek zostanie wybrana jako odpowiedź na pytanie, moduł ten ma kluczowe znaczenie dla wydajności całości systemu. Z tego powodu różne warianty jego realizacji eksperymentalnie przebadano w rozdziale 5.

3.4.1. Rozpoznawanie nazw własnych

Najbardziej rozpowszechnione w literaturze podejście do wydobywania nazw w tekście obejmuje wyrażenia określane jako *nazwy własne* (ang. *named entities*, zob. objaśnienie w Dodatku). Proces rozpoznawania ich w tekście określa się mianem rozpoznawania nazw własnych (ang. *Named Entity Recognition*, NER). Uwzględniane kategorie różnią się w różnych narzędziach, jednak w uproszczeniu można powiedzieć, że obejmują one takie nazwy, które łatwo rozpoznać w tekście dzięki ustalonej konwencji zapisu – przede wszystkim osoby, miejsca czy organizacje, czasem także liczby lub daty.

W ramach systemu RAFAEL użyto trzech narzędzi typu NER: *Nerf*, *Liner2* i *Quant*. *Nerf* (Savary i Waszczuk, 2012) to narzędzie opracowane w ramach budowy Narodowego Korpusu Języka Polskiego, bazujące na liniowych warunkowych polach losowych (ang. *Conditional Random Fields*, CRFs). Rozpoznaje 13 typów nazw własnych i uwzględnia możliwość zagnieżdżenia (np. *Jan Długosz* w *Uniwersytet im. Jana Długosza*). *Liner2* (Marcińczuk i Janicki, 2012) także wykorzystuje pola losowe, jednak rozróżnia aż 56 typów nazw. Liczbę tę można zredukować do 5, jeśli w danym zastosowaniu większą rolę odgrywa wysoka precyzja rozpoznawania niż wąskie kategorie. Znakowanie z wykorzystaniem obu tych narzędzi odbywa się na etapie wstępnego przygotowania bazy wiedzy (opis w punkcie 3.2.2), zatem w ramach odpowiadania na zadane pytanie wystarczy sięgnąć do danych skojarzonych z wybranym dokumentem.

Jako że żadne z przedstawionych narzędzi nie rozpoznaje liczb ani wielkości, do których często odnoszą się pytania, na potrzeby RAFAELa stworzono od podstaw nowe rozwiązanie pod nazwą *Quant*. Działa ono w sposób następujący:

- Każdy ciąg segmentów, rozpoznawany przez zadany automat skończony, zostaje oznaczony jako typ `number`,
- Każdy ciąg segmentów oznaczony jako `number`, po którym następuje jednostka miary, oznaczamy etykietą `quantity`.

Automat skończony rozpoznający polską notację liczb przedstawiono na rysunku 3.5. Bierze on pod uwagę następujący typy segmentów:

1. **(0-9)+** – łańcuch znaków składający się z samych cyfr,
2. **.** – kropka, separator grup cyfr,
3. **,** – przecinek, separator dziesiętny,
4. **num** – słowne wyrażenie liczby, tj. segment oznaczony przez tager jako liczebnik.

Aby uniknąć problemów z wydobywaniem jedynie części określenia liczbowego, obecnych np. w systemie *BulQA* (Simov i Osenova, 2005), wzorzec dopasowuje się w trybie zachłannym, tj. segmenty dołącza się tak długo, jak to możliwe. Metoda ta pozwala rozpoznać dużą różnorodność wyrażeń liczbowych, np. *10 tysięcy*, *kilka milionów* czy *1.698,88*.

Aby rozstrzygnąć, czy słowo następujące po liczbie określa jednostkę miary, pobiera się jego lemat. Następnie w bazie WordNet szuka się leksemu o tej samej nazwie i bada listę hiperonimów (bezpośrednich i pośrednich) wszystkich synsetów, do których on należy. Jeśli zawiera ona synset <jednostka miary.1>, to cały ciąg uzyskuje etykietę `quantity`. W ten sposób można wykryć wyrażenia takie jak *piętnaście kilogramów* czy *5 000 watów*. Ponieważ *plWordNet* wśród synonimów nazw jednostek miary zawiera również ich skrócone formy, także określenia typu *15 kg* czy *5 kW* zostaną poprawnie zinterpretowane.

Moduł *Quant* wykorzystano również do odnajdywania liczb w wypowiedziach sejmowych (Przybyła i Teisseyre, 2014). Średnia liczba wystąpień wyrażeń liczbowych w zdaniu posłużyła jako jedna z cech, pozwalających rozpoznawać płeć, przynależność partyjną, wykształcenie i wiek przemawiającego posła na podstawie zapisu jego wypowiedzi.

Używając zewnętrznych narzędzi NER trzeba pamiętać, że ich kategoryzacja nazw własnych różni się od tej, która pochodzi z analizy pytania. Tabela 3.10 przedstawia odwzorowanie między czterema klasyfikacjami nazw: typami pytań oraz typami przypisanymi przez *Nerf*, *Liner2* i *Quant*. Na zestawieniu tym można zaobserwować, że:

1. wiele typów NE pytania nie ma odpowiedników ani w *Nerf*, ani w *Liner2*,
2. dla wszystkich nazw geograficznych narzędzie *Nerf* przypisuje jedną kategorię `geogName`, co może wpływać na precyzję odpowiedzi,
3. w przypadku typów `CENTURY` i `YEAR`, *Nerf* rozpoznaje tylko ogólniejszy typ `date`,
4. *Liner2* nie rozróżnia `NAME` i `SURNAME`, rozpoznając tylko ogólny typ nazwy osoby – `person_nam`.

O ile braków niektórych typów nie można zrekompensować, to ostatnie dwa problemy są łatwe do przezwyciężenia. W tym celu wprowadzono dodatkową procedurę, która określa wartość roku i stulecia na podstawie rozpozna-

nej daty oraz imię i nazwisko z określenia osoby. Wykorzystuje się przy tym założenie, że pierwszy segment z ciągu oznaczonego etykietą `person_nam` to imię, a ostatni to nazwisko.

3.4.2. Głębokie rozpoznawanie nazw – DeepER

Głębokie rozpoznawanie nazw, stanowiące alternatywę dla NER, omówiono szczegółowo w rozdziale 4. Technika ta znakuje wzmianki synsetami, do których należą wskazywane przez nie byty, co pozwala na wyjście poza kategorie nazw własnych zarówno wgłąb, dzięki rozróżnianiu bardziej szczegółowych typów, jak i wszerz, przez uwzględnienie nazw bytów nienależących do tradycyjnych kategorii NE. DeepER można zastosować do wszystkich pytań o nazwy bytów (kategorie `UNNAMED_ENTITY` i `NAMED_ENTITY`) z wyłączeniem liczb, wielkości i dat.

Aby uruchomić rozpoznawanie z użyciem tej techniki, wymagany jest synset centrum. Jeśli do klasyfikacji pytania nie użyto semantycznej analizy centrum, np. w przypadku jednoznacznej konstrukcji *Gdzie . . . ?*, to model nie zawiera tego elementu. W tym wypadku określony jest natomiast typ nazwy własnej, z którego można wywnioskować oczekiwany synset. Do określania typu nazwy własnej na podstawie synsetu wykorzystywana jest lista zawierająca takie skojarzenia, oznaczona jako *nePairs* w algorytmie 2. Przykładowo, typowi NE `PLACE` odpowiada synset <obszar.1, strefa.1, terytorium.1, zona.1, obręb.1, rejon.3>. Na omawianym etapie można wykorzystać ten zasób w przeciwnym kierunku, tj. do określania synsetu na podstawie typu nazwy własnej. Dzięki temu dla pytania *Gdzie . . . ?*, dla którego znany jest typ NE `PLACE`, możemy przypisać wymieniony powyżej synset i wykorzystać technikę DeepER do odnalezienia wzmianek zgodnych z pytaniem.

3.4.3. Rozwiązanie hybrydowe

Ostatnia możliwość rozpoznawania nazw, jaką przewidziano w projekcie RAFAELa, to łączenie wyników z kilku narzędzi. W tym celu uruchamia się je niezależnie i tworzy zbiór wzmianek poprzez zsumowanie uzyskanych list. Powinno to spowodować wzrost pokrycia, gdyż można w ten sposób objąć większą różnorodność pytań, jednak w zamian traci się część precyzji. Rozwiązaniem tego problemu może być stosowanie do każdego typu takiego narzędzia, które radzi sobie z nim najlepiej (zobacz propozycje dalszych prac w podrozdziale 6.2).

3.5. Wybór odpowiedzi

Gdy dysponujemy już listą rozpoznanych w dokumentach wzmianek zgodnych z typem pytania, musimy wybrać tę spośród nich, która będzie stanowić podstawę odpowiedzi. Decyzji tej dokonuje się na podstawie treści pytania, wyodrębnionej na etapie analizy pytania (punkt 3.3.3). Treść zawiera dodatkowe informacje o poszukiwanym bycie, np. okoliczności wydarzenia, atrybuty lub funkcje osoby czy położenie, w przypadku obiektów geograficznych. W przyjętym podejściu zakłada się, że słowa zawarte w treści powtórzą się również

Typ NE pytania	**Typ Nerf**	**Typ Liner2**	**Typ Quant**
PLACE	placeName:*		
CONTINENT	geogName	continent_nam	
RIVER		river_nam	
LAKE			
MOUNTAIN			
RANGE		mountain_nam	
ISLAND		island_nam	
ARCHIPELAGO			
SEA		sea_nam	
CELESTIAL_BODY		astronomical_nam	
COUNTRY	placeName:country	country_nam	
STATE	placeName:region	admin1_nam admin2_nam admin3_nam historical_region_nam country_region_nam	
CITY	placeName:settlement	city_nam	
NATIONALITY	placeName:country	nation_nam	
PERSON	persName	person_nam	
NAME	persName:forename		
SURNAME	persName:surname		
BAND	orgName	band_nam	
DYNASTY	persName:addName		
ORGANISATION	orgName	organization_nam institution_nam political_party_nam	
COMPANY	orgName	company_nam	
EVENT		event_nam	
TIME	date		
CENTURY			
YEAR			
PERIOD			quantity
COUNT			number
QUANTITY			quantity
VEHICLE			
ANIMAL			
TITLE		title_nam media_nam	

Tabela 3.10. Zestawienie typów nazw własnych pochodzących z analizy pytania i dostępnych w zastosowanych narzędziach NER.

w tekście w sąsiedztwie wzmianki stanowiącej odpowiedź. Trzeba zatem dla każdej kandydującej wzmianki wydobyć pewne słowa z otoczenia, tworzące kontekst (punkt 3.5.1). Na koniec wykorzystując miarę podobieństwa dwóch zbiorów segmentów (punkt 3.5.2) określa się zgodność treści pytania i kontekstu odpowiedzi. Uzyskany wynik liczbowy stanowi podstawę dla budowy rankingu wzmianek i sformułowania odpowiedzi (punkt 3.5.3).

3.5.1. Wybór kontekstu

Celem procesu wybierania kontekstu jest wskazanie tych słów z treści dokumentu, które są powiązane z analizowaną wzmianką. Niestety, bez pełnego zrozumienia tekstu, można to zrobić w sposób zaledwie przybliżony. Spotyka się dwa podejścia do problemu, bazujące na:

- **zdaniach**, gdzie za kontekst służy pojedyncze zdanie, w którym wystąpiła wzmianka,
- **ciągach segmentów**, gdzie kontekstem może być każda ciągła sekwencja M segmentów, potencjalnie przekraczająca granice zdań, zawierająca wzmiankę.

Obie te możliwości posiadają swoje zalety. Poleganie na strukturze zdań zapewnia pewien semantyczny związek pomiędzy wzmianką, a tak utworzonym kontekstem. Jest to także najpopularniejsza metoda (zastosowali ją np.Yih i inni (2013)). Z drugiej strony, użycie ciągu segmentów pozwala na wpływanie na jego długość, która powinna być proporcjonalna do długości pytania, a dokładnie jego treści (zob. punkt 3.3.3). W tym podejściu dla każdej wzmianki powstaje wiele kontekstów, różniących się położeniem początku ciągu względem wzmianki. Budowę kontekstu jako ciągu segmentów znajdziemy np. w pracy Katza i innych (2003), którzy użyli wartości $M = 140$ bajtów, lub systemie *Webclopedia* (Hovy i inni, 2000), przy czym autorzy nie podali wielkości okna.

Wybór kontekstu wiąże się także z inną kwestią, tj. niejawnymi odwołaniami do bytów. Przyjmują one postać anafory, czyli zastępowania słów zaimkami, np. *on*, *wtedy*; lub elipsy, czyli całkowitego ich pomijania, np. w zdaniu *Urodził się w Warszawie.* Problem ten można rozwiązywać wykorzystując specjalizowane narzędzia do analizy koreferencji, ale odpowiednie rozwiązanie dla języka polskiego (Kopeć i Ogrodniczuk, 2012) w momencie budowy systemu RAFAEL pozostawało jeszcze we wczesnej fazie rozwoju. Obserwacje odwołań w korpusie treningowym wskazują, że w przypadku artykułów encyklopedycznych zdecydowana większość niejawnych odwołań odnosi się do bytu wspomnianego w tytule, zatem problem ten można częściowo zneutralizować poprzez automatyczne dołączanie go do kontekstu. W systemie RAFAEL zaimplementowano wszystkie opisane możliwości wybierania kontekstu: oparte na ciągach lub zdaniach oraz z dodaniem tytułu artykułu lub bez niego. Oczywiście kontekst nie zawiera wzmianki, wokół której powstaje.

Rozważmy omawiane rozwiązania na przykładzie próby odpowiedzi na pytanie: *Gdzie Albert Einstein był zatrudniony jako zastępca nauczyciela?*. Na etapie analizy pytania zostanie wyodrębniona jego treść w postaci następującego zbioru segmentów: {*Albert, Einstein, był, zatrudniony, jako, zastępca, na-*

uczyciela}. Odpowiedź zawiera fragment z artykułu Wikipedii zatytułowanego *Albert Einstein*[22]:

> *21 lutego 1901 r. Einstein przyjął obywatelstwo szwajcarskie. Mając już dyplom wykładowcy nauk ścisłych zaczął szukać pracy. Starał się bezskutecznie o asystenturę u wykładającego w ETHZ Webera, później u Hurwitza i Wilhelma Ostwalda. Dopiero w maju 1901 r. został zatrudniony na krótko jako zastępca nauczyciela w szkole średniej w Winterthur w Szwajcarii. W tym czasie zajmował się tam ruchem materii względem eteru i kinetyczną teorią gazów.*

Załóżmy, że słowo *Winterthur* zostało poprawnie rozpoznane jako wzmianka zgodna z typem PLACE i program przystępuje do wybierania jej kontekstu. Opierając się na zdaniach można otrzymać następujące zbiory segmentów:

- bez tytułu: *Dopiero w maju 1901 r. został zatrudniony na krótko jako zastępca nauczyciela w szkole średniej w w Szwajcarii.*
- z tytułem: *Albert Einstein Dopiero w maju 1901 r. został zatrudniony na krótko jako zastępca nauczyciela w szkole średniej w w Szwajcarii.*

Dla utworzenia ciągów segmentów przyjmijmy $M = 7$, co odpowiada długości treści pytania. Bez dodania tytułu powstaje następujący zbiór możliwych kontekstów:

- *jako zastępca nauczyciela w szkole średniej w*
- *zastępca nauczyciela w szkole średniej w w*
- *nauczyciela w szkole średniej w w Szwajcarii*
- ...
- *w Szwajcarii . W tym czasie zajmował*

Analogicznie po dodaniu tytułu otrzymujemy:

- *Albert Einstein jako zastępca nauczyciela w szkole średniej w*
- *Albert Einstein zastępca nauczyciela w szkole średniej w w*
- *Albert Einstein nauczyciela w szkole średniej w w Szwajcarii*
- ...
- *Albert Einstein w Szwajcarii . W tym czasie zajmował*

Na tym przykładzie widać, jak pomocne może być dodawanie tytułu artykułu do kontekstu. W rozważanym zdaniu *Einstein* pojawia się jako podmiot domyślny, zaś jego nazwisko znajduje się dopiero w odległości 50 segmentów od omawianej wzmianki.

Wyniki badań wpływu sposobu generowania kontekstu i jego rozmiaru na wydajność całego systemu przedstawiono w rozdziale 5.

3.5.2. Miara podobieństwa

Miara podobieństwa ciągów segmentów jest niezbędna, aby wskazać wzmiankę, której kontekst jest w największym stopniu zgodny z treścią pytania. Na

[22] http://pl.wikipedia.org/wiki/Albert_Einstein

wejściu tego procesu są zatem dwa zbiory segmentów, przy czym ich rozmiary mogą się różnić. Do określenia ich podobieństwa wykorzystuje się indeks Jaccarda (1901), zdefiniowany następująco:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{3.1}$$

gdzie:

- A to zbiór segmentów z treści pytania,
- B to zbiór segmentów z kontekstu odpowiedzi,
- $J(A,B)$ to wartość indeksu Jaccarda, zawierająca się w przedziale od 0 (dla całkowicie rozłącznych zbiorów) do 1 (dla identycznych zbiorów).

Powyższe sformułowanie nie uwzględnia faktu, że istotność słów różni się. Przykładowo, występowanie w obu zbiorach segmentu *onomatopeja* więcej mówi o ich podobieństwie, niż gdy w tej roli wystąpi słowo *on*. Aby to wyrazić, należy zmodyfikować wzór 3.1, wprowadzając wagi:

$$\mathrm{Sim}_w(A,B) = \frac{\sum_{i \in A \cap B} w_i}{\sum_{i \in A \cup B} w_i}, \tag{3.2}$$

gdzie:

- A i B to zbiory segmentów jak wyżej,
- w_i to dodatnia liczba rzeczywista, określająca wagę i-tego segmentu,
- $Sim_w(A,B)$ to wartość ważonej miary podobieństwa, zawierającej się w przedziale od 0 do 1.

Aby obliczyć $A \cap B$ i $A \cup B$, trzeba najpierw zdefiniować relację równości dla segmentów. Możliwe są tutaj dwa rozwiązania:

1. równość **form ortograficznych**,
2. równość **lematów**.

Zwróćmy uwagę, że wykorzystywanie przez indeks Jaccarda sumy i iloczynu zbiorów wyklucza możliwość użycia ciągłych miar podobieństwa segmentów, np. opartych o odległość edycyjną. W systemie RAFAEL zastosowano miarę opartą o równość lematów.

Waga w_i odzwierciedla istotność i-tego słowa. W tej roli stosuje się odwrotną częstość termów w dokumentach (ang. *Inverse Document Frequency*, IDF), czyli technikę stosowaną powszechnie w wyszukiwaniu informacji (Papineni, 2001). Pod dodaniu czynnika skalującego (dla lepszej interpretowalności) definicja wag wygląda następująco:

$$w_i = \frac{\log \frac{|D|}{|\{d : i \in d\}|}}{\max_j \log \frac{|D|}{|\{d : j \in d\}|}}, \tag{3.3}$$

gdzie:

- D to zbiór wszystkich dokumentów w korpusie,
- d to pojedynczy dokument,
- w_i to waga i-tego dokumentu – liczba rzeczywista zawierająca się w przedziale od 0 (dla słowa obecnego w każdym dokumencie) do 1 (dla słowa obecnego w najmniejszej liczbie dokumentów).

Podobnie jak w przypadku miary podobieństwa zbiorów segmentów, wagi obliczać można dla lematów lub form ortograficznych. Dla zachowania spójności także i tu wybrano lematy. Do obliczenia wag konieczne było określenie liczby wystąpień słów w dokumentach całego korpusu; część tych danych przedstawiono w celach poglądowych w punkcie 5.1.1.

Indeks Jaccarda często stosuje się do określania podobieństwa zdań, zob. np. system opublikowany przez Ahna i in. (2004). Z kolei w przypadku porównywania zdania z całym dokumentem popularna jest miara kosinusowa, jednak wymaga ona takiej reprezentacji danych, w której dane wejściowe stanowią wektory o równej długości. Używano jej np. w pracy Marcińczuka i innych (2013b), gdzie w wydajności wskazywania najlepszego dokumentu ustępowała ona mierze MSW (ang. *Minimum Span Weighting*), która uwzględnia odległości pomiędzy dopasowywanymi słowami w tekście (więcej szczegółów w podrozdziale 2.4).

Czasem miarę rangowania ciągu segmentów komplikuje fakt, że obliczając ją uwzględnia się równocześnie stopień zgodności potencjalnej odpowiedzi z typem pytania – tak jest np. w systemie *Webclopedia* (Hovy i inni, 2000). W takiej wersji jako odpowiedź mogą zostać zwrócone nazwy, które nie są zgodne z pytaniem co do typu, ale za to ich kontekst bardzo przypomina treść pytania. W systemie RAFAEL ocena typu odpowiedzi odbywa się oddzielnie, na etapie wyboru wzmianek (podrozdział 3.4).

Wiele interesujących technik pomiaru podobieństwa słów, wychodzących poza poziom łańcuchów znaków i uwzględniających znaczenie, zaprezentowano ostatnio na konferencji ACL (Yih i inni, 2013). Korzysta się tu z relacji, takich jak synonimia, antonimia, hiperonimia, hiponimia i częste współwystępowanie. Uogólnienie tego podejścia stanowią wspominane już tutaj łańcuchy leksykalne (Moldovan i Novischi, 2002), gdzie przy obliczaniu podobieństwa uwzględnia się wszystkie relacje w bazie WordNet, jak również przypisane tam do synsetów definicje. Dla każdego synsetu s_i inny synset należy do jego sąsiadów, jeśli spełnia przynajmniej jeden z warunków:

- jest wzmiankowany w definicji s_i,
- jest powiązany z s_i jakąkolwiek relacją,
- jest wzmiankowany w definicji synsetu s_j, powiązanego z s_i jakąkolwiek relacją,
- w jego definicji wzmiankuje się s_i.

Miarę podobieństwa dwóch pojęć stanowi długość najkrótszej ścieżki łączącej je w tak zdefiniowanym grafie sąsiedztwa, przy czym na dokładną wartość wpływają również wagi przypisane relacjom. Należy jednak zwrócić uwagę, że tego typu podejścia ignorują kontekst wystąpienia słów i mogą zawodzić, gdy odgrywa on dużą rolę, np. w przypadku konstrukcji z przeczeniem.

3.5.3. Formułowanie odpowiedzi

W ostatniej fazie procesu odpowiadania na pytanie system dysponuje listą wzmianek bytów zgodnych z ograniczeniami pytania, z których każda posiada kontekst o określonym podobieństwie do jego treści. Jak wskazać odpowiedź? Najbardziej oczywistym rozwiązaniem jest wybranie tej wzmianki,

której kontekst otrzymał maksymalny wskaźnik podobieństwa. Istnieje jednak inna możliwość, tj. agregacja wyników. W tym podejściu łączy się wartości indeksu przypisane wzmiankom, które odpowiadają tej samej odpowiedzi, np. poprzez sumę lub średnią. Przykładowo, takie rozwiązanie odnajdziemy w systemach dla bułgarskiego (Tanev, 2004) i czeskiego (Konopík i Rohlík, 2010). Wstępne testy nie wykazały jednak, aby w przypadku RAFAELa przyniosło to dobre wyniki, więc ograniczono się do prostego maksimum. Odpowiedź podana przez system składa się z następujących elementów:

1. odpowiedzi w sensie ścisłym, czyli krótkiego łańcucha znaków,
2. zdania źródłowego,
3. dokumentu źródłowego,
4. wskaźnika zaufania.

Określenie zdania i dokumentu źródłowego polega po prostu na wybraniu zdania i dokumentu, w których wystąpiła zwycięska wzmianka. Bardziej złożone podejścia byłyby uzasadnione, gdyby odpowiedź powstawała na podstawie wielu wzmianek lub gdy odpowiedź pozyskano spoza korpusu, w którym należy wskazać dokument źródłowy (Katz i inni, 2003).

Wskaźnik zaufania odzwierciedla, na ile dana odpowiedź uważana jest za prawidłową w świetle posiadanych informacji. Rolę tę pełni opisany w punkcie 3.5.2 ważony indeks podobieństwa kontekstu i treści pytania, który przyjmuje wartości od 0 (brak współwystępujących słów) do 1 (pełna zgodność po pominięciu odmiany i kolejności słów). Odpowiedź nie jest zwracana, jeśli nie odnaleziono żadnych wzmianek spełniających ograniczenia pytania. Tak samo można postąpić, jeśli wskaźnik zaufania najlepszej wzmianki znajduje się poniżej zdefiniowanego progu. W ten sposób odpowiemy na mniejszą liczbę pytań (niższe pokrycie), ale będą to odpowiedzi o większej pewności (wyższa precyzja).

Oszacowanie pewności odpowiedzi i ewentualna odmowa jej udzielenia odgrywa dużą rolę w teleturnieju *Jeopardy!*, a więc również w systemie *IBM Watson* (Ferrucci i inni, 2010), gdzie pozwala na osiągnięcie precyzji na poziomie 95% dla 20% pytań. Stosuje się ją jednak również przy „zwykłym" QA, np. przy odpowiadaniu na pytania o przyczynę (Oh i inni, 2013) osiągnięto w ten sposób precyzję 83,2%, ograniczając się do odpowiedzi na 25% pytań, które otrzymały najwyższą wartość wskaźnika zaufania.

3.6. Implementacja

System RAFAEL został zrealizowany jako program w języku *Java* 1.7 SE. Całość implementacji obejmuje 60 klas i interfejsów w 18 pakietach:

- `annotationsNKJP` – pakiet zawierający funkcje związane z generowaniem i wczytywaniem znakowania tekstu (punkt 3.2.2):
  - `AnnotatedText` – klasa reprezentująca oznakowany tekst,
  - `Group` – klasa reprezentująca grupę składniową,
  - `GroupOrWord` – interfejs reprezentujący cechy wspólne dla słów składniowych i grup składniowych,

  - — `NamedEntity` – klasa reprezentująca nazwę własną,
  - — `NamedEntityOrSegment` – interfejs reprezentujący cechy wspólne dla nazw własnych i segmentów,
  - — `NKJPReader` – klasa wykonująca wczytywanie plików XML znakowania w formacie NKJP do struktur w pamięci programu,
  - — `Paragraph` – klasa reprezentująca akapit tekstu,
  - — `ParserNKJPException` – klasa reprezentująca wyjątek procesu wczytywania tekstu,
  - — `Segment` – klasa reprezentująca segment,
  - — `Sentence` – klasa reprezentująca zdanie,
  - — `Word` – klasa reprezentująca słowo składniowe,
  - — `WordOrSegment` – interfejs reprezentujący cechy wspólne dla słów składniowych i segmentów,
- `answerer` – pakiet zawierający główne funkcje realizujące proces odpowiadania na pytanie:
  - — `Answerer` – interfejs, który musi wypełniać klasa odpowiadająca na pytania,
  - — `AnswererException` – klasa reprezentująca wyjątek procesu odpowiadania na pytania,
  - — `RankedAnswerer` – klasa realizująca moduł odpowiadający na pytania poprzez zwrócenie uporządkowanej listy odpowiedzi,
  - — `SimpleAnswerer` – klasa realizujący podstawowy proces odpowiadania na pytania,
- `context` – pakiet zawierający funkcje związane z generowaniem kontekstu dla wzmianki (punkt 3.5.1):
  - — `ComplexContext` – klasa realizująca wydobywanie kontekstu w postaci jednego zdania z tytułem,
  - — `ContextGenerator` – interfejs reprezentujący strategię wydobywania kontekstu,
  - — `ComplexContext` – klasa realizująca wydobywanie kontekstu w postaci ciągu segmentów z tytułem,
  - — `SingleSentenceContext` – klasa realizująca wydobywanie kontekstu w postaci jednego zdania,
- `corpus` – pakiet zawierający klasy reprezentujące korpus źródłowy (podrozdział 3.2):
  - — `Article` – klasa reprezentująca jeden artykuł,
  - — `Corpus` – klasa reprezentująca cały korpus źródłowy,
- `DeepER` – pakiet zawierający klasy realizujące technikę głębokiego rozpoznawania nazw (rozdział 4):
  - — `DeepEREntityPicker` – klasa realizująca rozpoznawanie nazw metodą DeepER (podrozdział 4.3),
  - — `Entity` – klasa reprezentująca deskryptor bytu (podrozdział 4.2),
  - — `EntityMention` – klasa reprezentująca wzmiankę określonego bytu w tekście,
  - — `EntityMiner` – klasa realizująca tworzenie biblioteki bytów na podstawie encyklopedii (podrozdział 4.2),
- `evaluation` – pakiet zawierający klasy realizujące ewaluację (rozdział 5):

  - — `EvaluationResult` – klasa przechowująca rezultat ewaluacji,
  - — `Evaluator` – klasa realizująca proces ewaluacji,
  - — `EvaluatorThread` – klasa realizująca współbieżny proces ewaluacji,
- `exec` – pakiet zawierający klasę umożliwiającą wywoływanie zewnętrznych narzędzi:
  - — `ExecCommand` – klasa wykonująca określone polecenie w systemie operacyjnym,
- `indexLucene` – pakiet zawierający klasę obsługującą indeks *Lucene* (punkt 3.2.1):
  - — `Index` – klasa reprezentująca indeks wyszukiwarki i umożliwiająca jego utworzenie,
- `main` – pakiet zawierający klasę główną programu:
  - — `Main` – klasa główna, wywoływana z linii poleceń i realizująca różne funkcje programu,
- `plWordNet` – pakiet zawierający klasy obsługujące kontakt z bazą *plWordNet*:
  - — `Lexeme` – klasa reprezentująca leksem,
  - — `ParserPlWNException` – klasa reprezentująca wyjątek procesu wczytywania bazy,
  - — `PlWNReader` – klasa realizująca wczytywanie bazy z pliku,
  - — `Synset` – klasa reprezentująca synset,
  - — `WordNet` – klasa przechowująca bazę,
- `questionAnalysis` – pakiet zawierający klasy realizujące analizę pytania (podrozdział 3.3):
  - — `questionQuery` – pakiet zawierający klasy realizujące generowanie zapytania (punkt 3.3.2):
    - — `FirstANDQueryGenerator` – klasa realizująca tworzenie zapytania z priorytetem koniunkcji,
    - — `QueryGenerator` – interfejs opisujący klasy generujące zapytanie,
    - — `SimpleQueryGenerator` – klasa realizująca podstawową strategię tworzenia zapytania,
    - — `SynonymQueryGenerator` – klasa realizująca tworzenia zapytania z synonimami,
  - — `questionType` – pakiet zawierający klasy realizujące klasyfikację pytań (podrozdział 3.3.1):
    - — `PatternQuestionParser` – klasa realizująca klasyfikację pytania na zasadzie dopasowywania wzorców,
    - — `QuestionParser` – interfejs opisujący klasy klasyfikujące pytania,
    - — `WNQuestionParser` – klasa realizująca klasyfikację pytania z analizą semantyczną centrum,
  - — `QuestionModel` – klasa przechowująca model pytania,
- `sentencePicking` – pakiet zawierający klasy służące do wybrania wzmianki na podstawie miary podobieństwa kontekstów (podrozdział 3.5):

  - `matchImportance` – pakiet zawierający klasy określające wagi dopasowania słów (punkt 3.5.2):
    - `FreqMatchImportance` – klasa obliczająca wagę dopasowania słów na podstawie ich częstości,
    - `LengthMatchImportance` – klasa obliczająca wagę dopasowania słów na podstawie ich długości,
    - `MatchImportance` – interfejs określający zadanie oceny wagi dopasowania słów,
  - `segmentDistance` – pakiet zawierający klasę określającą podobieństwo segmentów (punkt 3.5.2):
    - `EqualityDistanceMeasure` – klasa określająca podobieństwo segmentów na podstawie równości ich form ortograficznych lub bazowych,
    - `SegmentDistanceMeasure` – interfejs opisujący określanie podobieństwa segmentów,
  - `sentenceDistance` – pakiet zawierający klasę określającą podobieństwo kontekstów (punkt 3.5.2):
    - `JaccardDistanceMeasure` – klasa obliczająca podobieństwo kontekstów z użyciem indeksu Jaccarda,
    - `SentenceDistanceMeasure` – interfejs opisujący określanie podobieństwa kontekstów,
    - `SimpleDistanceMeasure` – klasa obliczająca podobieństwo kontekstów na podstawie liczby zgodnych segmentów,
  - `AnnotatedNamedEntityPicker` – klasa realizująca rozpoznawanie nazw z użyciem narzędzi NER (punkt 3.4.1),
  - `CenturyConverter` – klasa realizująca określanie stulecia na podstawie daty oznaczonej przez narzędzie NER (punkt 3.4.1),
  - `EntityPicker` – interfejs opisujący rozpoznawanie nazw (punkt 3.4),
  - `FirstLastNameConverter` – klasa realizująca określanie imienia i nazwiska na podstawie nazwy osoby oznaczonej przez narzędzie NER (punkt 3.4.1),
  - `UnsupportedEntityException` – klasa reprezentująca wyjątek polegający na braku obsługi danego typu pytania,
- `tools` – pakiet zawierający dodatkowe klasy pomocnicze:
  - `LemmatisatorViaGenerator` – klasa wspierająca określanie lematów grup składniowych,
  - `NumericalNEAdder` – klasa realizująca moduł *Quant*, czyli oznaczająca wzmianki wielkości i liczb.

RAFAEL do poprawnego działania wymaga zainstalowania w systemie programów umożliwiających znakowanie tekstów źródłowych i pytań (punkt 3.2.2). Są to:

- analizator morfosyntaktyczny *Morfeusz Polimorf* 0.82,
- tager *PANTERA* 0.9.1,
- parser powierzchniowy *Spejd* 1.3.7,
- narzędzie do rozpoznawania nazw własnych *Nerf* 0.1,
- narzędzie do rozpoznawania nazw własnych *Liner2* 2.3.

Dodatkowo istnieje szereg innych zasobów, które są wykorzystywane w procesie odpowiadania na pytania. Są to:

- biblioteki *Apache Commons* w wersji 3.1,
- biblioteki *Lucene* 3.6,
- biblioteka *PATRICIA trie* 0.6,
- pliki zawierające częstości lematów i form ortograficznych do oceny wagi dopasowania,
- skrypty powłoki *bash*, wywołujące narzędzia znakowania,
- skrypt w języku *Python* generujący plik ze strukturą na podstawie znaków nowej linii,
- gramatyka do parsowania powierzchniowego z uwzględnieniem lematyzacji grup składniowych,
- skrypty w języku *Python* zapewniające konwersję między formatem znakowania NKJP a CCL, używanym przez program *Liner2*,
- baza *plWordNet* w wersji 2.1,
- pliki zawierające mapowanie między typami nazw własnych w narzędziach NER, a typami pytań,
- pliki zawierające jednoznaczne i niejednoznaczne wzorce do klasyfikacji pytań,
- plik zawierający mapowanie między synsetami WordNet a typami pytań.

Rozdział 4

Głębokie rozpoznawanie nazw

System odpowiadania na pytania w języku polskim RAFAEL różni się od innych opublikowanych rozwiązań nie tylko dostosowaniem do pracy z tekstami w języku o bogatej fleksji i swobodnym szyku zdań. W budowie systemów QA dla języka angielskiego szczególną uwagę zwraca się zazwyczaj na procedurę wyboru zdania, które wykazuje największe podobieństwo do pytania. Tymczasem przy projektowaniu i implementacji RAFAELa skupiono się na precyzyjnym rozpoznawaniu nazw. Dzięki temu znacznie mniejsza liczba zdań jest w ogóle brana pod uwagę i stosunkowo prosta miara podobieństwa (punkt 3.5.2) wystarcza do osiągnięcia dobrych rezultatów. W tym celu tradycyjne rozpoznawanie nazw własnych (NER) zastąpiono nową techniką, nazwaną głębokim rozpoznawaniem nazw (ang. *Deep Entity Recognition*, DeepER). Sedno rozwiązania polega na tym, że zamiast zaklasyfikować napotkaną w tekście nazwę do jednej z kilku lub kilkunastu predefiniowanych kategorii, przypisuje się jej synsety w bazie WordNet, opisujące typy bytów, do których się ona odnosi.

Jako przykład rozważmy następujące pytanie: *Który wygnany europejski monarcha powrócił do ojczyzny jako premier republiki?*. W tradycyjnym podejściu, przedstawionym w poprzednim rozdziale, analiza pytania prowadzi do wniosku, że poszukujemy określenia osoby. Stosujemy zatem jedno z narzędzi NER do wskazania w wybranych tekstach wszystkich nazw własnych osób i traktujemy je jako potencjalne odpowiedzi. Łatwo zauważyć, że przy tak ogólnym ograniczeniu liczba nazw branych pod uwagę będzie ogromna i opracowanie efektywnej funkcji oceny kandydujących zdań sprawi duże problemy. Użycie DeepER zmienia ten sposób postępowania już na etapie analizy – zachowujemy informację o tym, że pytanie dotyczyło monarchy. Informacja ta podana jest w formie synsetu z bazy *plWordNet*: <monarcha.1, koronowana głowa.1>. Pozwala to na ograniczenie analizy tekstu tylko do fragmentów wspominających o monarchach, a nie o jakichkolwiek osobach. W szczególności w tekście może występować określenie *Symeon II*, któremu przypisano synset <car.1>. Dzięki zawartej w WordNecie relacji hiperonimii między wymienionymi synsetami wiadomo, że odnaleziona nazwa może stanowić odpowiedź.

DeepER jest uogólnieniem i pogłębieniem NER również w innym sensie – pozwala rozpoznawać nazwy bytów nienależących do kategorii nazw własnych. Przykładowo załóżmy, że poszukujemy odpowiedzi na pytanie *Który ptak co roku migruje z Arktyki do Antarktyki i z powrotem?* Wyrażenie *rybitwa popielata*, stanowiące prawidłową odpowiedź, nie należy do nazw własnych, a więc nie może zostać wyodrębnione przez narzędzie typu NER. Jeśli jednak korzystamy z głębokiego rozpoznawania nazw, to wzmianki o tym gatunku zo-

staną oznaczone synsetem <ptak wodny.1>, którego relacja hiperonimii z synsetem centrum pytania <ptak.1> pozwoli na uwzględnienie *rybitwy popielatej* wśród potencjalnych odpowiedzi. Podejście DeepER działa zatem równie dobrze dla nazw własnych i ogólnych.

Skąd jednak wiemy, że *Symeon II* jest carem, a *rybitwa popielata* ptakiem wodnym? W typowych narzędziach NER klasyfikowanie odbywa się poprzez analizę cech uzyskanych na podstawie słów tworzących wzmiankę i jej bezpośredniego sąsiedztwa. Na przykład, następujące po sobie dwa słowa, rozpoczęte wielką literą, z których pierwsze zostało rozpoznane przez analizator morfologiczny jako rzeczownik rodzaju męskoosobowego, a drugie nie rozpoznane wcale, z dużym prawdopodobieństwem określają osobę. Takie cechy kontekstu sprawdzają się dobrze przy ogólnych kategoriach NER, ale nie wystarczą dla odróżnienia sztangisty od szachisty lub rybitwy od ryby.

Informację o typach i nazwach bytów można pozyskać z uprzednio przygotowanej bazy danych, nazwanej *biblioteką bytów*, w której byty reprezentowane są za pośrednictwem dwóch składników:

- zbioru reprezentacji tekstowych (nazw),
- zbioru synsetów.

W niniejszej pracy baza ta powstała poprzez analizę definicji encyklopedycznych bytów (do tego celu użyto polskiej Wikipedii). W poprzednim przykładzie wykorzystalibyśmy wpis[1]:

> *Symeon II, (...) – ostatni car Bułgarii (1943-1946), prawnik, politolog, menedżer i polityk, pierwszy w historii państw Europy Wschodniej monarcha obalony przez reżim komunistyczny, który został premierem republiki (2001-2005); założyciel i w latach 2001-2009 lider Narodowego Ruchu na rzecz Stabilności i Postępu.*

Proces wydobywania nazw i synsetów z takiej definicji, uwzględniający również strony przekierowań i ujednoznaczniań, przedstawiono szczegółowo w podrozdziale 4.2. Dysponując bazą bytów, wystarczy zlokalizować w tekście wzmianki odpowiadające nazwom bytów zgodnych z ograniczeniami pytania. Zadanie to (podrozdział 4.3) stanowi pewne wyzwanie ze względu na złożoną odmianę nazw własnych w języku polskim (Przepiórkowski, 2007).

Model DeepER oferuje jeszcze jedną usługę, tj. automatyczną ewaluację. Zazwyczaj systemy QA ocenia się poprzez ręczne sprawdzenie zgodności odpowiedzi: uzyskanej i oczekiwanej. Samo porównanie łańcuchów znaków nie wystarczy, ponieważ wiele bytów ma niejedną reprezentację tekstową, np. *Franciszek*, *Jorge Maria Bergoglio* lub *Bergoglio*. Odmiana tych nazw jeszcze bardziej komplikuje sprawę. Tymczasem korzystając z DeepER jesteśmy w stanie poddać procesowi rozpoznawania pary odpowiedzi i porównać *byty*, a nie *łańcuchy znaków*. Dzięki procesowi automatycznej ewaluacji (podrozdział 5.2) możliwe było szybkie przeprowadzenie wielu eksperymentów, których ręczna ocena wymagałyby ogromnego nakładu pracy.

[1] `http://pl.wikipedia.org/wiki/Symeon_II`

4.1. Pokrewne rozwiązania

Koncepcja głębokiego rozpoznawania nazw, przedstawiona powyżej, jest nowa, ale jej składniki można odnaleźć we wcześniej opublikowanych rozwiązaniach. W pierwszej kolejności należy wymienić istniejące bazy, mogące pełnić rolę analogiczną do biblioteki bytów w DeepER, tj. służyć do rozpoznawania w tekście wzmianek według precyzyjnie określonego typu. Są to ontologie, zawierające pojęcia i łączące je relacje: *Freebase* (Bollacker i inni, 2008), *BabelNet* (Navigli i Ponzetto, 2010), *DBpedia* (Bizer i inni, 2009) czy *YAGO* (Suchanek i inni, 2007). Nazwy pojęć są zwykle wyrażone w języku angielskim, zaś adaptacja takich narzędzi do języka polskiego stanowi trudne zadanie ze względu na złożoność problemów tłumaczenia i odmiany nazw własnych. Niedawno powstała tego typu baza zawierająca także polskie nazwy, o nazwie *Prolexbase* (Savary i inni, 2013), jednak na razie składa się z 40.000 nazw zgrupowanych w 34 typy. Dodatkowo, dostępne ontologie posiadają swoje własne kategorie pojęć, co przy włączeniu w system QA czyniłoby koniecznym określenie odwzorowania pomiędzy nimi, a typami pytań.

Samo sedno rozwiązania, tj. uzgadnianie synsetu przypisanego do pytania i kandydującej odpowiedzi pojawiło się w pracy Manna (2002). Choć technika analizy pytania wydaje się bardzo podobna, to budowa biblioteki bytów (nazywanej tam ontologią nazw własnych) różni się znacznie. Autor przeanalizował automatycznie 1 GB tekstów prasowych, biorąc pod uwagę pewne szczególne konstrukcje. Przykładowo wyrażenie *X, such as Y* (X, takie jak Y) oznacza, że *Y* najprawdopodobniej należy do kategorii *X*. Chociaż osiągnięta jakość bazy nie była najwyższa (47% poprawnych nazw dla innych kategorii niż ludzie), to pozwoliła ona na zauważalne zwiększenie pokrycia zbioru pytań (z 16,9% na 19,4%) przy zachowaniu precyzji odpowiedzi.

Znakowanie wzmianek w tekście za pomocą synsetów WordNet zastosowali również Na i inni (2002). W odróżnieniu od poprzednio omawianej pracy, w tym wypadku nie buduje się bazy nazw, ale przypisuje się kategorie tylko do kandydujących odpowiedzi. Wspólny element stanowi tutaj sposób wyboru synsetu – wykorzystuje się bezpośredni kontekst wystąpienia nazwy. Jak można się spodziewać, jeżeli przypisany synset jest hiponimem centrum pytania, to taka wzmianka jest brana pod uwagę przy wyborze odpowiedzi. Co ciekawe, stosowany algorytm uwzględnia również sytuację odwrotną, tj. gdy synset kandydującej odpowiedzi stanowi hiperonim centrum pytania. Wtedy uruchamia się dodatkowy mechanizm ujednoznaczniający, polegający na przeglądaniu innych wystąpień tej samej nazwy w korpusie w poszukiwaniu takich kontekstów, w których użyto bardziej precyzyjnego synsetu. System ewaluowano na zbiorze TREC-11, osiągając precyzję na poziomie 20%.

Pomysł, żeby zamiast poszukiwania ustalonych fraz wykorzystać definicje encyklopedyczne także już się pojawił, ale w kontekście innych zastosowań. Na przykład Toral i Muñoz (2006) przedstawili metodę automatycznej budowy gazetera (katalogu nazw własnych, szczególnie geograficznych) przez analizę ścieżki hiperonimii rzeczowników z pierwszego zdania definicji w Wikipedii. W odróżnieniu od niniejszej pracy, w rozwiązaniu tym wykonuje się rzutowanie na gruboziarniste kategorie nazw własnych i pomija uzyskany synset. Przykład innego zastosowania to rozpoznawanie nazw własnych wy-

korzystujące słowa z definicji jako dodatkowe cechy dla klasyfikatora, które przedstawili Kazama i Torisawa (2007). Narzędzie to odnajduje nazwy własne w tekście i przypisuje je do ogólnych kategorii (osoby, miejsca, organizacje, inne). Schemat działania jest typowy dla tego typu rozwiązań – wykorzystano klasyfikator CRF ze standardowym zestawem cech. Głównym przedmiotem badania było włączenie w proces rozpoznawania wiedzy pobranej z Wikipedii. W tym celu rozważany ciąg słów uznaje się za tytuł artykułu, przy czym, podobnie jak w RAFAELu, bierze się pod uwagę również strony przekierowujące i ujednoznaczniające. Z pierwszego zdania wpisu encyklopedycznego pobierane jest ostatnie słowo pierwszej grupy nominalnej (w angielskiej Wikipedii najczęściej jest to rzeczownik). Słowo to nie podlega żadnej analizie, lecz jest traktowane jako cecha dla klasyfikatora – na tej samej zasadzie, jak np. słowa z kontekstu klasyfikowanego wystąpienia nazwy. Kolejne cechy powstają przez pobieranie listy kategorii, do których należy odnaleziony artykuł. W ewaluacji wykorzystano zbiór z zadania CoNLL 2003; przyrost wydajności rozpoznawania (mierzony miarą F1) był stosunkowo nieduży – z 86,46% na 88,02%.

Inni badacze przyglądali się zadaniu klasyfikacji artykułów Wikipedii według kategorii nazw własnych, do których należy byt omawiany w definicji. Na przykład Dakka i Cucerzan (2008) zaproponowali rozwiązanie bazujące na łączeniu tradycyjnych technik klasyfikacji tekstów (reprezentacja *bag of words*) z kontekstem wzmianek klasyfikowanych bytów. Oznacza to, że jeśli próbujemy określić typ bytu opisywanego przez pewien artykuł, to warto wziąć pod uwagę zdania, w których pojawiają się łącza do tego artykułu. Przykładowo w opisie miejscowości może pojawić się zdanie takie, jak *Urodziła się tam Gwen Stefani*, gdzie ostatnie dwa słowa stanowią łącze do artykułu zatytułowanego *Gwen Stefani*. Po wyrażeniu *urodziła się tam* często występuje określenie osoby, a zatem jest to cecha, którą można wykorzystać w klasyfikacji. Okazało się jednak, że znacznie silniejsze są cechy wynikające z liczności określonych słów w całym tekście i pierwszym akapicie. Ponzetto i Strube (2007) analizowali kategorie przypisane do artykułu, uznając je za potencjalne źródło relacji *is-a* (hiperonimii). Wprawdzie 99% wpisów w Wikipedii posiada przypisaną przynajmniej jedną kategorię, ale nie wszystkie mają charakter hiperonimiczny. Przykładowo artykuł *Flaga Estonii* przynależy do kategorii *Flagi państw Europejskich* (tu relacja jest typu *is-a*), ale także *Estonia* (nie *is-a*). W celu rozróżnienia takich przypadków autorzy zastosowali klasyfikator heurystyczny. Kategorie Wikipedii używane były także jako cechy w zadaniu rozpoznawania nazw własnych (Richman i Schone, 2008), ale i w tym wypadku wymaga to opracowania reguł rozdzielających kategorie hiperonimiczne i niehiperonimiczne – w tej pracy sformułowanych ręcznie.

Niektórzy postrzegają Wikipedię jako doskonałe źródło danych trenujących dla klasyfikatorów NER. Rozwiązanie takie polega na klasyfikacji artykułów według kategorii nazw własnych, a następnie potraktowaniu linków do nich kierujących jako odwołań do bytu odpowiedniego typu. Obszerne badania na ten temat przedstawili Balasuriya i inni (2009). Do klasyfikacji wykorzystywali treść artykułu w reprezentacji *bag of words*, kategorie, słowa kluczowe, między-artykułowe i między-językowe łącza. Ponieważ w Wikipedii zwykle tylko pierwsza wzmianka o danym bycie wzbogacana jest odwołaniem do artykułu, konieczne było uzupełnianie kolejnych łączy. Dzięki zastosowaniu tej

procedury do całej treści anglojęzycznej Wikipedii powstał korpus treningowy równie dobry (w sensie jakości wytrenowanego na nim narzędzia NER), jak ręcznie oznakowany złoty standard.

Istnieją także rozwiązania leżące pomiędzy tradycyjnym rozpoznawaniem nazw własnych a techniką DeepER, tj. wprowadzające bogatsze kategorie NE przy zachowaniu podejścia do klasyfikacji bazującego na uczeniu maszynowym. Przykładowo, Ciaramita i Altun (2006) stworzyli tager przypisujący słowa w tekście do 41 kategorii, nazwanych „nadznaczeniami" (ang. *supersenses*). Owe nadznaczenia obejmują klasy nazw własnych, ale także inne grupy, np. rośliny, zwierzęta czy kształty. Autorzy zastosowali rzutowanie publicznie dostępnych korpusów z oznakowanymi nazwami własnymi na nowe kategorie i użyli ich do wytrenowania ukrytego modelu Markowa (HMM). Osiągnięta precyzja i pokrycie wyniosły około 77%.

4.2. Biblioteka bytów

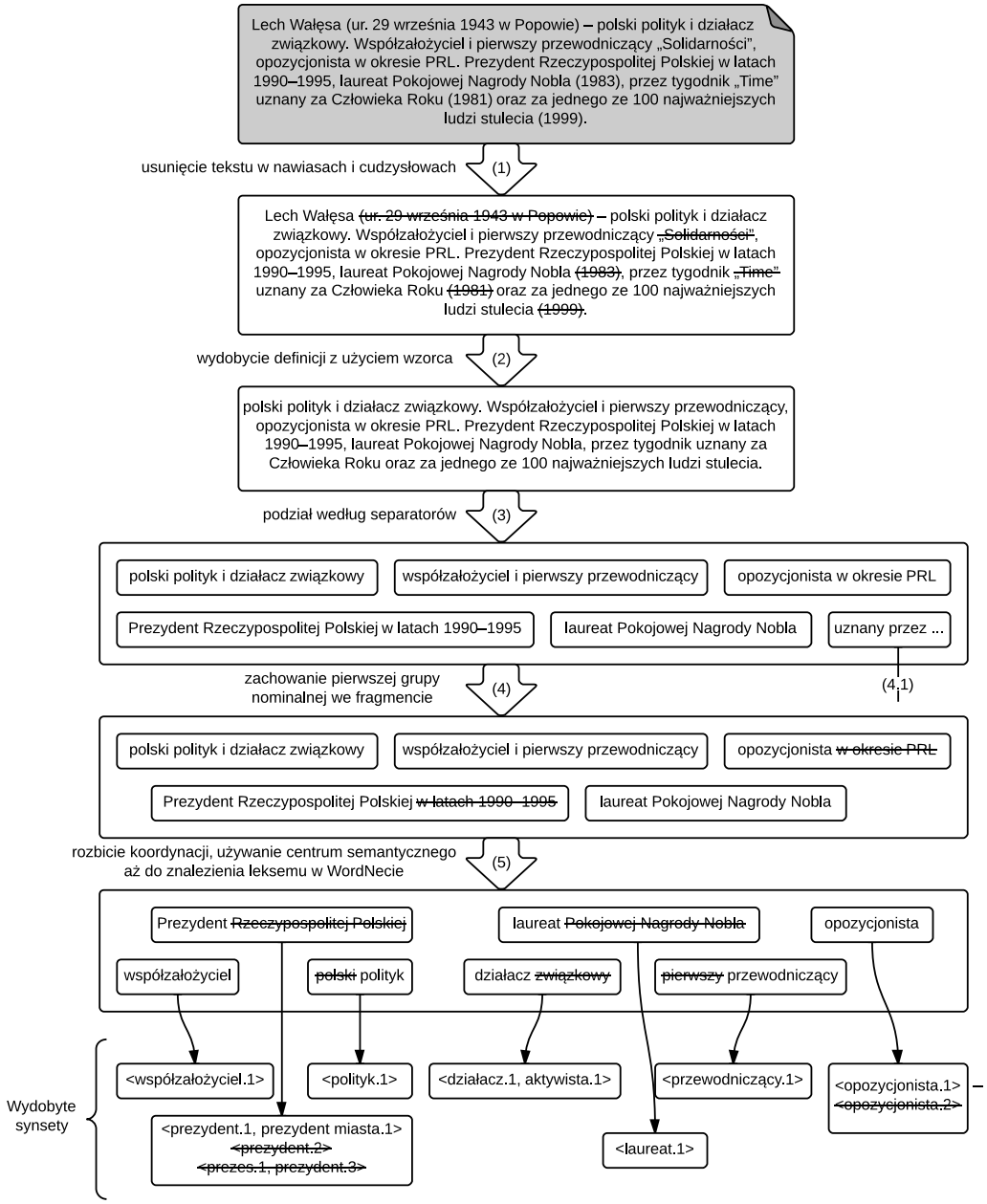
Biblioteka bytów zawiera wiedzę niezbędną dla rozpoznawania nazw techniką DeepER. Stanowi ona prostą listę deskryptorów bytów, z których każdy składa się z następujących elementów (na przykładzie wpisu nr 9751, opisującego prezydenta Bronisława Komorowskiego[2]):

- główna nazwa: *Bronisław Komorowski*,
- inne nazwy (aliasy): *Bronisław Maria Komorowski*, *Komorowski*,
- adres URL źródła,
- synsety WordNet:
  - <podsekretarz.1, podsekretarz stanu.1, wiceminister.1>,
  - <wicemarszałek.1>,
  - <polityk.1>,
  - <wysłannik.1, poseł.1, posłaniec.2, wysłaniec.1, posłannik.1>,
  - <marszałek.1>,
  - <historyk.1>,
  - <minister.1>,
  - <prezydent.1, prezydent miasta.1>.

Proces wydobywania informacji o bytach z korpusu źródłowego, wykonywany niezależnie od procesu odpowiadania na pytania, polega na analizie definicji encyklopedycznych. Biblioteka dla systemu RAFAEL powstała na bazie korpusu polskiej Wikipedii, który posłużył również za korpus źródłowy dla odpowiadania na pytania, opisany w punkcie 5.1.1. Korpus zawiera 857.952 artykuły, 51.866 stron ujednoznaczniających i 304.823 przekierowania. Wynikowa baza składa się z 809.786 bytów, opisanych przez 1.169.452 nazw (972.592 różne) i 1.264.918 synsetów (31.545 różnych). Zamiast Wikipedii można do tego celu użyć dowolnego zasobu zawierającego definicje, np. słownika. Istotna jest jedynie spójność słownictwa między korpusem encyklopedycznym a tematyką, której będą dotyczyć zadawane pytania.

Rysunek 4.1 przedstawia proces przypisywania synsetów WordNet słowom z pierwszego akapitu wpisu encyklopedycznego na przykładzie strony opisują-

[2] `http://pl.wikipedia.org/wiki/Bronis%C5%82aw_Komorowski`



Rysunek 4.1. Wydobywanie listy synsetów opisujących byt z definicji encyklopedycznej na przykładzie wpisu z Wikipedii, poświęconej Lechowi Wałęsie.

cej prezydenta Lecha Wałęsę[3]. W pierwszej kolejności usuwa się zbędne fragmenty tekstu (1). Dotyczy to tekstu w nawiasach i cudzysłowach, ale również nieistotnych wyrażeń wprowadzających, takich jak *jeden z* lub *typ* (podobnie postępują Kazama i Torisawa (2007), usuwając wyrażenia *kind of*, *sort of* itp.). Następnie nazwę bytu oddziela się od samej definicji poprzez dopasowanie jednego z wzorców (2). W przedstawionym przykładzie jest to najczęstszy z nich, czyli półpauza (–). Wszystkie wystąpienia separatorów (kropek, przecinków i średników) w definicji służą do podziału tekstu na fragmenty (3). Następne kroki wykorzystują płytkie parsowanie – tylko grupy nominalne pojawiające się na początku fragmentów przechodzą do dalszej analizy (4). Pierwszy fragment niespełniający tego warunku, jak również wszyscy jego następcy, zostaje wykluczony z procesu (4.1). Jeśli pojawiają się grupy koordynacyjne, to zostają rozbite na składniki. Dla każdej grupy w zbiorze sprawdza się, czy jej lematowi odpowiada jakiś leksem w bazie WordNet (5). Jeśli nie, to proces powtarza się, przy czym grupy, dla których nie odnaleziono leksemu, zostają zastąpione swoimi centrami semantycznymi. W przypadku słów wieloznacznych wykorzystuje się tylko pierwsze znaczenie.

Podobnie postępuje się ze stronami ujednoznaczniającymi. Przykładowo w polskiej Wikipedii hasło *Wałęsa* prowadzi do strony, która zawiera kilka wpisów przekierowujących, a wśród nich: *Lech Wałęsa (ur. 1943) – przywódca Solidarności, Prezydent RP w latach 1990–1995, laureat Pokojowej Nagrody Nobla*. W stosunku do tego wpisu stosuje się identyczny algorytm, a uzyskane nazwy (*Wałęsa*) i synsety dodaje się do istniejącego deskryptora. W tym wypadku synsety z ujednoznacznienia zawierają się w zbiorze synsetów z pełnego artykułu, ale bardzo często zdarza się, że zawarte tam opisy są zwięźlejsze i łatwiejsze w automatycznej interpretacji niż pełne definicje.

Cały proces jest jednak w ogólności bardziej złożony, niż można pokazać na tak prostym przykładzie. Składa się on z następujących kroków:

Krok 0 Przygotowanie korpusu – format danych i proces znakowania jest analogiczny[4], jak przy przygotowywaniu bazy wiedzy, opisywanym w punkcie 3.2.2. Różnica tkwi w zakresie korpusu – wykorzystuje się nie tylko zwykłe artykuły, ale także strony przekierowań i ujednoznaczniania.

Krok 1 Z każdego artykułu pobiera się pierwszy akapit i stosuje do niego funkcję *readDefinition*. Jeśli uzyskany deskryptor bytu posiada niepustą listę synsetów, zostaje dodany do biblioteki. Adres artykułu zapisuje się jako adres źródła, a tytuł jako główną nazwę. Jeśli część ze stron przekierowujących wskazuje na ten byt, ich nazwy wzbogacają jego listę aliasów.

Krok 2 Każda strona ujednoznaczniająca zostaje podzielona na fragmenty odpowiadające poszczególnym ujednoznacznieniom i stosuje się do nich funkcję *readDefinition*. Gdy wskazują one na byty obecne już w bibliotece, dopisuje się do odpowiednich deskryptorów uzyskane synsety i nazwę strony źródłowej. W przeciwnym przypadku powstaje nowy deskryptor bytu na podstawie uzyskanych danych. Dodanie nazw z przekierowań przebiega jak powyżej.

Krok 3 Uzyskana biblioteka zostaje zapisana w pliku.

[3] `http://pl.wikipedia.org/wiki/Lech_Wa%C5%82%C4%99sa`

[4] Nie wykorzystuje się jedynie znakowania na poziomie nazw własnych.

```
Input: text – pierwszy akapit definicji encyklopedycznej
Output: synsets – synsety opisujące byt
begin
    synsets := {};
    text := removeInBrackets(text);
    text := removeInQuotes(text);
    foreach pattern in definitionPatterns do
        if pattern matches text then
            definition := match(pattern,text).group(2);
            break;
        end
    end
    definition := skipDefinitionPrefixes(definition);
    parts := split(definition,seperators);
    foreach part in parts do
        chunk := firstGroupOrWord(part);
        if isNominal(chunk) then
            synsets := synsets ∪ extractSynsets(chunk);
        end
        else break;
    end
    return synsets;
end
```

Algorytm 5: Funkcja **readDefinition**(*text*) – interpretuje definicję i wydobywa synsety, do których przynależy opisywany byt.

Przedstawiona jako algorytm 5 funkcja *readDefinition* przyjmuje na wejściu pierwszy akapit z definicji i wydobywa z niego listę synsetów, jak na przykładzie z rysunku 4.1. Funkcja ta wykorzystuje następujące elementy:

removeInBrackets() usuwa cały tekst pomiędzy nawiasami zwykłymi, kwadratowymi lub klamrowymi (krok (1) w przykładzie).

removeInQuotes() usuwa tekst objęty pojedynczymi lub podwójnymi cudzysłowami (krok (1) w przykładzie).

definitionPatterns to lista zawierająca ciągi znaków oddzielające definiowane pojęcie od definicji, powstała przez konkatenację znaków definicji (pierwsza kolumna tabeli 4.1) z wyrażeniami definicyjnymi (druga kolumna tabeli 4.1) – krok (2) w przykładzie.

skipDefinitionPrefixes() usuwa wyrażenia poprzedzające pierwszą grupę nominalną, analogicznie do *skipQuestionPrefixes* w algorytmie 2, przy czym lista wyrażeń częściowo różni się (trzecia kolumna tabeli 4.1).

separators zawiera trzy znaki, które oddzielają fragmenty definicji: kropka, średnik i przecinek (użyte w kroku (3) w przykładzie).

firstGroupOrWord() zwraca najdłuższy element składniowy (słowo lub grupę), zaczynający się na początku danego fragmentu tekstu (krok (4) w przykładzie).

Znaki definicji (kod UTF)	Wyrażenia definicyjne	Wyrażenia wprowadzające
- (HYPHEN-MINUS)	(puste)	*gromada*
− (MINUS SIGN)	*to*	*klasa*
– (FIGURE DASH)	*jest*	*rząd*
– (EN DASH)	*jest to*	*grupa*
— (EM DASH)	*są*	*odmiana*
— (HORIZONTAL BAR)	*są to*	*seria*
	był	*rodzaj*
	był to	*typ*
	była	*gatunek*
	była to	*jeden z*
	było	*jedno z*
	było to	*jeden ze*
	były	*jedna ze*
	były to	*jedno ze*
	byli	*określenie*
	byli to	*nazwa*
		z wykształcenia
		z zawodu

Tabela 4.1. Trzy listy wyrażeń używanych do rozpoznawania fragmentów definicji: znaki definicji w złożeniu z wyrażeniami definicyjnymi oddzielają nazwę bytu od definicji (*definitionPatterns* w algorytmie 5), zaś wyrażenia wprowadzające poprzedzają grupy nominalne, z których wydobywa się synsety (używane w funkcji *skipDefinitionPrefixes*).

isNominal() sprawdza, czy podany element jest rzeczownikiem w mianowniku, grupą nominalną lub koordynacją grup nominalnych.

extractSynsets() to funkcja zwracająca synsety obecne w danym rzeczowniku lub grupie nominalnej, przedstawiona jako algorytm 3 w punkcie 3.3.1 i odpowiadająca krokowi (5) w przykładzie.

Kilka spośród podjętych decyzji wpływających na przedstawione procedury wymaga komentarza. Po pierwsze, zastosowane podejście różni się od tych rozwiązań, które wykorzystują definicje w reprezentacji *bag of words* (Dakka i Cucerzan, 2008; Ruiz-Casado i inni, 2005; Balasuriya i inni, 2009). Tutaj założono pewną strukturę definicji, tj. serię grup nominalnych oddzielonych separatorami. Co więcej, ponieważ kropka jest jednym z nich, seria ta może rozciągać się poza pierwsze zdanie, co w wielu definicjach ma zastosowanie (przykład na rysunku 4.1). Dzięki temu, że tekst został oznakowany na poziomie grup składniowych z uwzględnieniem ich lematyzacji, możliwe jest odnajdywanie w WordNecie całych fraz, np. *łódź podwodna*, a nie samych tylko rzeczowników, jak w innych pracach (Toral i Muñoz, 2006). Z powodu stosunkowo swobodnego szyku zdania w języku polskim nie można tu założyć, jak Kazama i Torisawa (2007), że grupa kończy się rzeczownikiem. Zamiast tego, w sytuacji, gdy cała grupa nie ma swojego odpowiednika w bazie WordNet, używa się jej centrum semantycznego.

Ponieważ wiele słów posiada więcej niż jedno znaczenie, do określenia właściwego synsetu konieczne staje się wybranie jednego z nich – w przyjętym

rozwiązaniu jest to zawsze znaczenie oznaczone liczbą 1. Pomysł ten bazuje na założeniu, że jako pierwsze podano najczęstsze znaczenie, jednak nie zawsze musi tak być, np. na rysunku 4.1 synset <prezydent.1, prezydent miasta.1> poprzedza synset <prezydent.2> (tj. prezydent kraju), podczas gdy w sytuacji braku informacji o kontekście to drugie znaczenie wydaje się bardziej oczywiste. Ważne jest, aby decyzje podjęte na tym etapie były spójne z budową analizatora pytań. Wyobraźmy sobie np. sytuację, w której moduł analizy definicji za domyślne znaczenie słowa *prezydent* uznaje <prezydent.2> i zgodnie z tą zasadą oznacza wzmianki w tekście, a model pytania *Który prezydent Polski objął urząd 22 grudnia 1990?* zawiera synset centrum <prezydent.1, prezydent miasta.1>. Wtedy oczywiście żadne wzmianki zgodne z modelem nie zostaną wskazane. W związku z tym decyzja o wyborze pierwszego znaczenia jest umotywowana najlepszymi wynikami tego wariantu w klasyfikacji pytań (punkt 3.3.1). Rozwiązanie takie znajdziemy także w innych zastosowaniach, np. generowaniu gazeterów (Toral i Muñoz, 2006).

W celu oceny jakości powstałej biblioteki bytów, jej zawartość porównano z ręcznie wybranymi synsetami dla 100 wylosowanych artykułów z Wikipedii. W 95 z nich pierwszy akapit zawierał opis bytu. Na podstawie tych haseł powstało 88 deskryptorów w bibliotece (pokrycie bytów 92,63%). Proces ręczny doprowadził do przypisania im 135 synsetów, podczas gdy automatycznie wydobyto ich 133. Spośród nich 106 było zgodnych (precyzja synsetów 79,70%), zaś 13 różniło się tylko wybranym znaczeniem. Zbiór ręczny zawierał również 16 synsetów bez odpowiednika w bibliotece (pokrycie synsetów 88,15%), w której znajdowało się natomiast 14 niewłaściwych pozycji.

Biblioteka bytów jest zapisana w postaci pojedynczego pliku tekstowego o rozmiarze 129,8 MB. Każdy z deskryptorów znajduje się w osobnej linii w postaci listy składników rozdzielonych znakami tabulacji[5]:

```
Aleksander Mniszek-Tchorznicki
Aleksander Mniszek-Tchorznicki
http://pl.wikipedia.org/wiki/?curid=2809169
3
4
Mniszek
Aleksander Mniszek-Tchorznicki
Aleksander Mniszek-Tchórznicki
597
1069
6809
5981
<lekarz1,doktor1,lek.1, rels=5 >
<hrabia1,graf1, rels=3 >
<sędzia1, rels=4 >
<urzędnik1,biuralista1, rels=3 >
```

Na początku widać główną nazwę i nazwę artykułu (w tym przypadku identyczne), adres URL źródła, liczbę nazw (3) i synsetów (4). Dalej następują kolejne nazwy i identyfikatory synsetów w bazie *plWordNet* 2.1. Na końcu umieszczono reprezentacje tekstowe synsetów – nie są one wykorzystywane przez program, ale ułatwiają przeglądanie listy przez człowieka i mogą się

[5] Dla czytelności przykładu przedstawionych w oddzielnych liniach.

okazać konieczne, gdyby w kolejnych wersjach *plWordNet* identyfikatory nie zostały zachowane.

4.3. Rozpoznawanie nazw

Użycie przygotowanej biblioteki bytów do wskazania wszystkich wzmianek zgodnych z typem pytania odbywa się na etapie oznaczania wzmianek, opisanym w podrozdziale 3.4, i zastępuje rozpoznawanie nazw własnych. Proces rozpoczyna się od załadowania pliku biblioteki do drzewa PATRICIA (Morrison, 1968) – drzewa prefiksowego, pozwalającego na szybkie przeszukiwanie przy niewielkich wymaganiach pamięciowych. W tej strukturze kluczem staje się każda z nazw głównych lub aliasów, zaś wartość to lista bytów powiązana z tą nazwą.

Po wczytaniu dokumentu sprawdza się, czy zawiera on tekst pasujący do jednej z nazw. Trzeba jednak wziąć pod uwagę, że tekst na wejściu składa się z grup składniowych i segmentów o określonych lematach, zaś klucze w drzewie to zwykłe łańcuchy znaków. Wykorzystuje się zatem trzy źródła kandydujących nazw:

- lematy grup składniowych i słów,
- lematy pojedynczych słów,
- formy ortograficzne słów.

Pierwsza z wymienionych technik jest niewystarczająca, ponieważ proces lematyzacji słów i grup składniowych często zawodzi i podaje niewłaściwe formy bazowe, szczególnie w przypadku nazw własnych. Poniżej podano przykłady takich słów i grup składniowych wraz z przypisanymi im lematami:

- *Gór Skalistych* ⟶ góra skalista
- *dużymi kulami gradowymi* ⟶ duży kul gradowy
- *ThyssenKrupp* ⟶ thyssenkruppa
- *Benito Mussoliniego* ⟶ Benito Mussoliniego
- *Leopolda Mozarta* ⟶ Leopold Mozarta
- *Eidgenössische Technische Hochschule* ⟶ eidgenössischy technischy hochschul
- *Winterthur* ⟶ Winterthura
- *doktorem honoris causa* ⟶ doktór honoris causa

Dodatkowo często zdarza się, że grupa składniowa (np. imię i nazwisko) wcale nie jest rozpoznana, a zatem nie można pobrać dla niej lematu. Aby prawidłowo rozpoznać nazwy w takich przypadkach trzeba dopuścić pewne różnice pomiędzy nazwą napotkaną w tekście (według któregokolwiek z podanych źródeł), a tą z biblioteki bytów. Uznaje się zatem, że dopasowanie tych dwóch łańcuchów zachodzi, jeśli spełniają one następujące warunki:

1. mają wspólny prefiks,
2. niedopasowana końcówka w żadnym z nich nie jest dłuższa niż 3 znaki,
3. wspólny prefiks ma większą długość niż 50% długości obu słów.

Dysponując listą wzmiankowanych bytów, RAFAEL sprawdza ich zgodność z modelem pytania. Pod uwagę brane są dwa składniki – typ ogólny i synset centrum pytania. Aby przejść test zgodności centrum, synset ten musi być (bezpośrednim lub pośrednim) hiperonimem jednego z synsetów wzmiankowanego bytu. Na przykład lista synsetów, przypisana do bytu o głównej nazwie *Jan III Sobieski*, zawiera element <król.1>, zatem pasuje do centrum pytania typu <władca.1, panujący.1, hierarcha.2, pan.1> poprzez ścieżkę hiperonimii <władca.1, panujący.1, hierarcha.2, pan.1> → <monarcha.1, koronowana głowa.1> → <król.1>. Dodatkowo, jeśli typ ogólny to `NAMED_ENTITY` różny od `QUANTITY` lub `COUNT`, to wymaga się, aby pierwszy segment wzmianki rozpoczynał się wielką literą. Wszystkie wzmianki spełniające te warunki zostają przekazane do dalszego przetwarzania, czyli oceny zgodności z treścią pytania.

Rozdział 5

Ewaluacja

Do oceny wydajności systemu QA niezbędne jest środowisko testowe składające się z korpusu tekstów źródłowych i zestawu pytań, dla których odpowiedzi znajdują się w tym korpusie. W podrozdziale 5.1 przedstawiono dane, które wykorzystano do testowania systemu RAFAEL. W procesie ewaluacji przygotowane pytania przekazuje się do systemu, po czym otrzymane odpowiedzi porównuje się do odpowiedzi opracowanych uprzednio przez człowieka na podstawie tego samego korpusu. Jak wspomniano we wstępie do rozdziału 4, choć nie wystarczy porównać reprezentacji tekstowych odpowiedzi, to technika głębokiego rozpoznawania nazw umożliwia zautomatyzowanie tego procesu (zostało to wyjaśnione w podrozdziale 5.2). Dzięki temu możliwe stają się liczne eksperymenty, których ręczna ocena wymagałoby ogromnego nakładu ludzkiej pracy. Ich wyniki opisano w podrozdziale 5.3.

Biorąc pod uwagę fakt, że rezultaty eksperymentów zostały użyte do ustalenia optymalnych wartości pewnych parametrów procesu odpowiadania na pytania, konieczne staje się przeprowadzenie ostatecznej oceny wydajności na oddzielnym zbiorze pytań. Aby zadanie było bardziej realistyczne, użyto do tego celu pytań z innego źródła, a więc nieco odmiennych w stylu i treści, i ręcznie oceniono prawidłowość wypowiedzi. Rezultaty przedstawiono w podrozdziale 5.4.

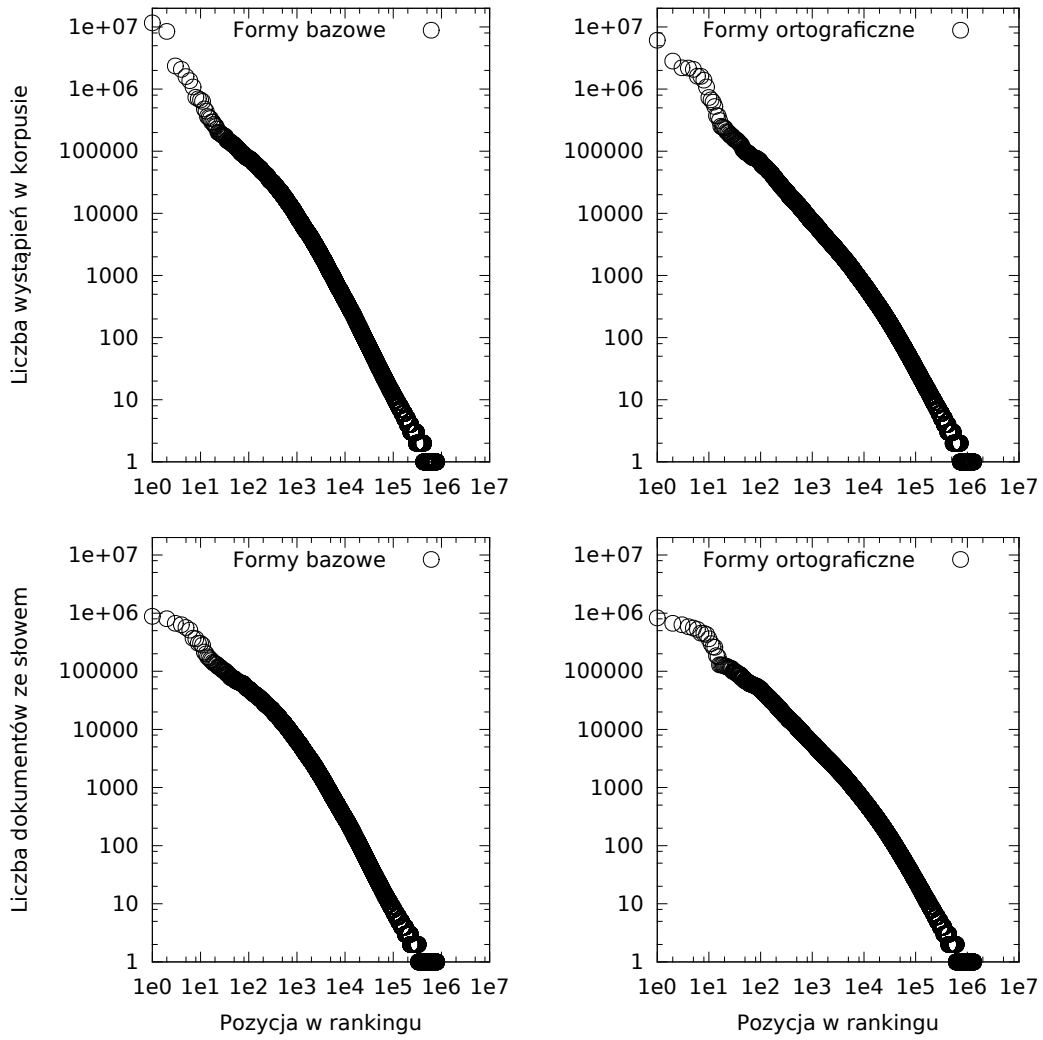
5.1. Dane

Dane używane do ewaluacji to zbiory trojakiego rodzaju: dokumenty źródłowe, pytania i odpowiedzi. Istnienie tego typu środowiska testowego jest niezwykle pomocne dla budowy i rozwoju systemów QA, ponieważ pozwala ilościowo ocenić wydajność systemu i porównywać różne rozwiązania. Niestety, jego przygotowanie wymaga znacznej ilości pracy, szczególnie przy odnajdywaniu odpowiedzi w korpusie. Dla języka angielskiego najczęściej w tym celu wykorzystuje się zbiory z konferencji TREC (więcej na ten temat w podrozdziale 2.1). Pierwszy tego typu zbiór dla języka polskiego został opublikowany dopiero niedawno (Marcińczuk i inni, 2013a) i rolę odpowiedzi pełni w nim nazwa dokumentu, który odpowiada na pytanie. Na bazie tych danych powstał omawiany poniżej zbiór ewaluacyjny.

5.1.1. Korpus źródłowy

Jako korpus źródłowy wykorzystano treść polskiej Wikipedii, pobraną ze strony projektu[1] jako pojedynczy zrzut bazy danych z dnia 3. marca 2013. Dane

[1] `http://dumps.wikimedia.org/plwiki/`



Rysunek 5.1. Rozkład popularności segmentów w zależności od ich pozycji w rankingu. Pokazano liczbę wystąpień w całym korpusie (górne wykresy) i liczbę dokumentów zawierających segment (na dole), zapisany jako forma bazowa (lewe wykresy) lub ortograficzna (po prawej).

te mają postać pliku XML o rozmiarze 4,7 GB, zawierającego artykuły w formacie *Wikitext*, opisującym formatowanie i nietekstowe elementy artykułów. W celu ich konwersji do czystego tekstu wykorzystano skrypt *Wikipedia Extractor*[2] w wersji 2.2. Program ten usuwa nie tylko formatowanie, ale również wszystkie elementy dodatkowe, np. listy, tabele, *infoboksy*[3], obrazy. Po ukończeniu procesu konwersji powstaje korpus o rozmiarze 1,06 GB, składający się z 895.486 dokumentów, zawierających 168.982.550 segmentów, podlegający następnie znakowaniu w procesie omówionym w punkcie 3.2.2.

Do policzenia wag segmentów wykorzystywanych w mierze podobieństwa zdań (punkt 3.5.2), potrzebne było zgromadzenie statystyk wystąpień form

[2] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[3] Infoboks (ang. *infobox*) to widoczna na stronie artykułu w Wikipedii ramka pod zdjęciem, zawierająca podstawowe fakty dotyczące obiektu. Na przykład w przypadku budynków umieszcza się tam ich położenie, projektanta, datę budowy i ew. zniszczenia.

bazowych i ortograficznych w całym korpusie. Powstałe cztery zbiory danych ukazano na rysunku 5.1 w standardowy sposób, tj. wykreślając liczbę wystąpień w zależności od miejsca słowa w rankingu, stosując skalę logarytmiczną na obu osiach. Widzimy, że zgodnie z prawem Zipfa (Newman, 2005) punkty układają się w przybliżeniu liniowo nie tylko w przypadku liczby wystąpień w korpusie (górne wykresy), ale i liczby dokumentów, w których pojawił się segment (dolne wykresy). Wszystkich form ortograficznych zanotowano 1.330.977, zaś lematów 799.431. Najpopularniejsza forma ortograficzna to kropka, występująca 11.664.360 razy w 883.649 dokumentach (czyli prawie we wszystkich). Pozostałe popularne segmenty to inne znaki interpunkcyjne, spójniki i przyimki.

5.1.2. Pytania

Zgodnie z wyjaśnieniami we wstępie, dla większej wiarygodności wyników wykorzystano dwa oddzielne zbiory z pytaniami. Ponieważ oba dotyczą wiedzy ogólnej, w obu przypadkach za korpus źródłowy służy Wikipedia.

Zbiór treningowy

Zbiór treningowy powstał na podstawie 1500 opublikowanych pytań z teleturnieju *Jeden z dziesięciu* (Karzewski, 1997). Program ten ma dość typowy przebieg – uczestnikom zadawane są kolejno pytania z przygotowanego zbioru, na które należy odpowiedzieć w ciągu kilku sekund. Zazwyczaj oczekiwana odpowiedź ma bardzo zwięzłą formę. W pierwszej kolejności zbiór poddano filtrowaniu, usuwając pytania niespełniające warunków opisanych w punkcie 2.2.1. Najczęstsze problemy to:

- brak odpowiedzi w bazie wiedzy (zazwyczaj spowodowany deaktualizacją zagadnienia, gdyż pytania pochodzą z 1997 roku),
- konieczność wykonywania zadań takich jak tłumaczenie czy obliczenia,
- oczekiwana odpowiedź opisowa (np. *O czym nauką jest genetyka?*).

Po etapie filtrowania pozostało 1130 pytań, które wymagały jeszcze poprawek, tj.:

- aktualizacji zawartych informacji,
- skrócenia, gdy podano znacząco więcej danych niż potrzeba do jednoznacznego wskazania odpowiedzi (np. kilkuzdaniowy życiorys),
- sprowadzenia poleceń (*podaj*, *wymień* itp.) do postaci pytającej,
- doprecyzowania, gdy w pytaniu brakowało danych.

Tak powstały zbiór był wykorzystywany do wykonywania eksperymentów w trakcie rozwoju systemu, zarówno dotyczących analizy pytania (podrozdział 3.3), jak i poprawności udzielanych odpowiedzi w zależności od parametrów procesu (podrozdział 5.3).

Zbiór ewaluacyjny

Podstawę ostatecznej ewaluacji stanowi wspominany już zbiór opublikowany przez zespół z Pollitechniki Wrocławskiej (Marcińczuk i inni, 2013a). Pochodzi on z rubryki *Czy wiesz . . . ?* w polskiej Wikipedii, która służy zainteresowaniu

użytkownika odwiedzającego stronę główną encyklopedii i zachęceniu go do lektury wybranego artykułu. Przykładowe pytanie brzmi: *Czy wiesz, skąd pochodzi roślina ozdobna, w Polsce znana jako iksora szkarłatna?* i kieruje do artykułu pt. *Iksora szkarłatna*. Jeśli usuniemy początkowe wyrażenie *Czy wiesz*, otrzymujemy pytanie o prosty fakt. Dodatkową zaletę tego zbioru stanowi fakt, że znany jest artykuł źródłowy i można mieć dużą pewność, że w istocie znajduje się tam oczekiwana odpowiedź, gdyż to na jego podstawie opracowano pytanie. Eliminuje to najbardziej pracochłonny etap przygotowania zbioru, czyli odnajdywanie odpowiedzi w korpusie.

Autorzy zbioru pobrali 9580 pytań ze strony projektu i poddali je filtracji, usuwając pytania:

- powtarzające się,
- będące *de facto* stwierdzeniami, tj. typu *Czy wiesz, że . . . ?*,
- nie prowadzące do żadnego artykułu według stanu bazy z 13 marca 2013,
- złożone, np. *Gdzie i kiedy . . . ?*,
- wymagające kontekstu do udzielenia odpowiedzi,
- niezrozumiałe,
- odnoszące się bieżącej daty.

W rezultacie pozostało 4721 pytań, każde z przypisanym artykułem zawierającym odpowiedź. Do wykorzystania w ramach niniejszej pracy wylosowano z nich 1050 i poddano dodatkowej selekcji. Ze względu na przygotowanie zbioru tym razem wystarczyło jedynie wybrać pytania o proste fakty według kryteriów z punktu 2.2.1. Najliczniejszą grupę usuniętą na tym etapie stanowią problemy opisowe, takie jak *Dlaczego . . . ?* czy *W jaki sposób . . . ?*. W rezultacie w zbiorze pozostało 576 elementów, które nie wymagały już żadnej korekty. Zestaw ten wykorzystano tylko i wyłącznie do ewaluacji ostatecznej, opisanej w podrozdziale 5.4.

5.1.3. Odpowiedzi

Do każdego z pytań z obu zbiorów ręcznie przypisano następujące informacje (przykłady dla pytania *Gdzie wynaleziono system dziesiętny oraz używane dziś przez cały świat cyfry zwane arabskimi?*):

1. identyfikator (*40.7*),
2. ogólny typ pytania (`NAMED_ENITY`),
3. typ nazwy własnej (`PLACE`),
4. tytuł artykułu źródłowego z korpusu (*Cyfry arabskie*),
5. odpowiedź zwykłą (*W Indiach*),
6. odpowiedź uproszczoną (*Indie*).

Dwie formy odpowiedzi ułatwiają ewaluację RAFAELa i innych systemów QA, które w ogromnej większości pozwalają jedynie na wskazanie nazwy oczekiwanego bytu (*Indie*), ale nie potrafią sformułować odpowiedzi w pełni poprawnie (*W Indiach*). Dlatego właśnie to odpowiedź uproszczona jest wykorzystywana przy ewaluacji.

Pytanie		
Typ pytania	**Artykuł źródłowy**	**Odpowiedź**
W którym roku umarł Stefan Żeromski?		
NAMED_ENTITY:YEAR	Stefan Żeromski	1925
Jakie organella nadają barwę korzeniom marchwi?		
UNNAMED_ENTITY	Chromoplast	Chromoplasty
Jakiego wyznania jest większość mieszkańców Liechtensteinu?		
UNNAMED_ENTITY	Liechtenstein	Katolicyzm
Który z filozofów był twórcą „atomizmu”?		
NAMED_ENTITY:PERSON	Demokryt	Demokryt z Abdery
Czy Jacques Brel pochodził z Francji?		
TRUEORFALSE	Jacques Brel	Nie

Tabela 5.1. Przykładowe pytania ze zbioru treningowego wraz z ich typem ogólnym, typem nazwy własnej, tytułem artykułu źródłowego i odpowiedziami.

Przypisanie do pytań nie tylko odpowiedzi, ale i ich typów oraz artykułów źródłowych umożliwia testowanie modułu klasyfikacji pytań (punkt 3.3.1) i generatora zapytań do wyszukiwarki (punkt 3.3.2). Dodatkowo pozwala to na symulowanie perfekcyjnych wersji tych modułów przy analizie wydajności dalszych etapów procesu (podrozdział 5.3). Tabela 5.1 przedstawia kilka przykładowych pytań ze zbioru treningowego wraz z ich typami, odpowiedziami i artykułami źródłowymi.

Tabela 5.2 przedstawia wyrażoną procentowo liczbę pytań poszczególnych typów w zbiorach: treningowym i ewaluacyjnym. Rozkłady te różnią się znacząco – drugi z nich jest mniej różnorodny (niektóre typy, np. morze lub imię, nie występują ani razu) i zawiera mniej pytań o nazwy ogólne. Znacząco większa jest natomiast przewaga najpopularniejszego typu – dotyczącego osób (`PERSON`).

5.2. Ewaluacja automatyczna

Motywację dla wprowadzenia automatycznej ewaluacji przedstawiono we wstępie do rozdziału 4. Przypomnijmy, że wiele bytów ma różnorodne reprezentacje tekstowe i dlatego nie wystarczy porównać łańcuchów znaków, aby sprawdzić poprawność odpowiedzi. Zamiast tego należy sprawdzić, czy odnoszą się do tego samego bytu. Problem ten można częściowo obejść przez podawanie oczekiwanej odpowiedzi w formie wyrażenia regularnego (Walas i Jassem, 2010). Zwiększa to jednak nakład pracy przy przygotowywaniu każdej odpowiedzi, ponieważ konieczne jest uwzględnienie wszystkich możliwych jej sformułowań.

Lepsze rozwiązanie stanowi głębokie rozpoznawanie nazw, oparte na bibliotece zawierającej reprezentacje tekstowe przypisane do bytów. Sprawdzenie odpowiedzi przebiega wtedy następująco:

Krok 0 Na wejściu procesu przyjmuje się pytanie należące do jednego z ty-

Typ ogólny	Typ nazwy własnej	Zbiór treningowy	Zbiór ewaluacyjny
WHICH	-	2,39%	0,17%
TRUEORFALSE	-	2,21%	0,87%
MULTIPLE	-	2,57%	8,33%
OTHER_NAME	-	2,04%	0,87%
UNNAMED_ENTITY	-	32,30%	16,67%
NAMED_ENTITY	PLACE	2,83%	9,03%
	CONTINENT	0,35%	0,17%
	RIVER	0,97%	0,17%
	LAKE	0,80%	0,00%
	MOUNTAIN	0,35%	0,52%
	RANGE	0,18%	0,17%
	ISLAND	0,44%	0,17%
	ARCHIPELAGO	0,18%	0,00%
	SEA	0,18%	0,00%
	CELESTIAL_BODY	0,71%	0,17%
	COUNTRY	4,60%	0,35%
	STATE	0,62%	0,17%
	CITY	4,69%	0,35%
	NATIONALITY	1,06%	0,17%
	PERSON	22,92%	36,81%
	NAME	0,97%	0,00%
	SURNAME	0,88%	0,00%
	BAND	0,53%	0,17%
	DYNASTY	0,53%	0,17%
	ORGANISATION	1,86%	3,47%
	COMPANY	0,18%	0,52%
	EVENT	0,97%	1,39%
	TIME	0,18%	3,82%
	CENTURY	0,80%	0,00%
	YEAR	3,01%	0,69%
	PERIOD	0,09%	0,17%
	COUNT	2,74%	7,64%
	QUANTITY	0,53%	1,74%
	VEHICLE	0,88%	1,74%
	ANIMAL	0,09%	0,00%
	TITLE	3,36%	3,47%

Tabela 5.2. Udział poszczególnych typów w zbiorze treningowym (1130 pytań) i ewaluacyjnym (576 pytań).

pów, które obsługuje technika DeepER[4], oczekiwaną odpowiedź w formie łańcucha tekstowego i propozycję odpowiedzi w postaci wzmianki bytu w tekście.

Krok 1 Porównuje się formę ortograficzną wzmianki z oczekiwaną odpowiedzią – jeśli są równe, to odpowiedź zostaje uznana za prawidłową, co kończy algorytm.

Krok 2 Oczekiwanej odpowiedzi używa się jako klucza do drzewa PATRICIA (zobacz podrozdział 4.3), pobierając listę bytów, jakie mogą być przez nią wskazywane.

Krok 3 Dla wzmianki proponowanej jako odpowiedź uruchamiany jest pełen algorytm rozpoznawania nazw DeepER, zwracając w odpowiedzi listę bytów,

Krok 4 Oblicza się przecięcie zbiorów otrzymanych w krokach 2 i 3 – jeśli jest ono niepuste, to odpowiedź uznaje się za prawidłową.

Prześledźmy opisaną wyżej procedurę na przykładzie pytania: *Kto jest obecnie prezydentem Polski?*. Załóżmy, że oczekiwana odpowiedź została zapisana w formie *Bronisław Komorowski*, a system zwrócił tylko nazwisko: *Komorowski*. Porównanie łańcuchów znaków daje odpowiedź negatywną, zatem korzystając z drzewa nazw pobieramy listę bytów skojarzoną z oczekiwaną odpowiedzią. Zbiór ten zawiera trzech Bronisławów Komorowskich – pisarza, księdza i polityka. Zastosowanie procedury głębokiego rozpoznawania nazw do wzmianki wskazanej jako odpowiedź prowadzi do długiej listy bytów. Ponieważ procedura ta dopuszcza pewne odstępstwa od dokładnej równości znakowej, zawiera ona nie tylko mężczyzn noszących to nazwisko, ale i byty znalezione na stronie ujednoznaczniającej pt. *Komorowska*, tj. osiem kobiet i dwie wsie. Część wspólna tych dwóch list jest niepusta, gdyż zawiera dwóch Bronisławów Komorowskich[5].

Przedstawiona procedura wymaga trzech komentarzy. Po pierwsze – dlaczego wzmiankę kandydującą przeprowadza się przez pełen proces rozpoznawania, czyli uwzględniając również lematy, a w przypadku odpowiedzi wzorcowej bierze się pod uwagę tylko dokładną postać tekstową? Jest tak, ponieważ te pierwsze pochodzą z tekstu oznakowanego morfosyntaktycznie, a te drugie to wyrwane z kontekstu pojedyncze słowa. Można by wprawdzie przeprowadzić proces tagowania, ale pamiętajmy, że korzysta on z kontekstu, zatem przy jego braku jakość rozpoznawania byłaby niska. Jest to zatem kolejny z powodów, dla którego przygotowano odpowiedzi uproszczone (punkt 5.1.3), nie wymagające lematyzacji.

Choć opisywana metoda ewaluacji korzysta z techniki DeepER, to nic nie stoi na przeszkodzie, żeby z jej pomocą oceniać wyniki pozyskane w dowolny sposób, np. z pomocą tradycyjnego rozpoznawania nazw własnych (NER).

Na koniec zwróćmy ponownie uwagę na nasz przykład. Procedura doprowadziła do oczekiwanego wyniku pozytywnego, choć nie udało się jednoznacznie ustalić, o którego Bronisława Komorowskiego chodzi w pytaniu

[4] Czyli `UNNAMED_ENTITY` lub `NAMED_ENTITY` z wyłączeniem dat, liczb i wielkości.

[5] Trzeci Bronisław Komorowski (pisarz) niestety nie znajduje się na stronie ujednoznaczniającej *Komorowski*. Koncentracja edytorów Wikipedii na popularnych artykułach w budowie tego typu stron może powodować słabsze działanie techniki DeepER dla mniej znanych bytów.

lub odpowiedzi. Taka niejednoznaczność pozostaje jednak nawet w przypadku ręcznej oceny odpowiedzi i aby ją usunąć, należałoby dopuścić zadawanie pytań uściślających ze strony użytkownika. To już jednak inna dziedzina badań – systemy dialogowe (ang. *dialogue systems*).

5.3. Eksperymenty

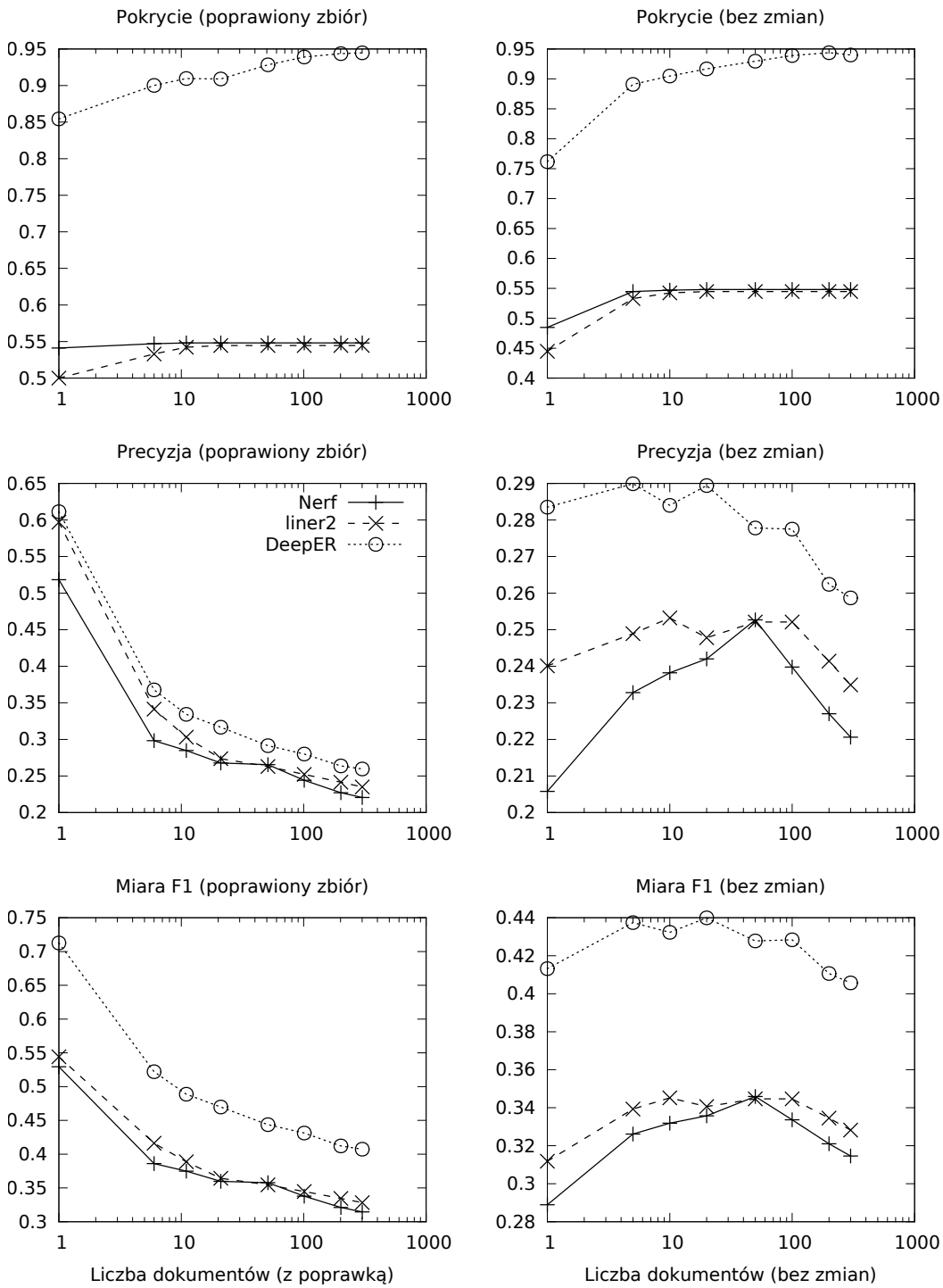
W niniejszym podrozdziale przedstawiono szereg eksperymentów polegających na uruchomieniu procesu odpowiadania na pytania w różnych wariantach i automatycznej ocenie wyników na podstawie zbioru treningowego. Mianem *pokrycia* (ang. *recall*) określa się udział procentowy pytań, na które system udzielił jakiejkolwiek odpowiedzi, w całym zbiorze, zaś *precyzja* (ang. *precision*) oznacza, jaka część z tych odpowiedzi jest prawidłowa. Miarę F1 oblicza się jako średnią harmoniczną powyższych wielkości.

Interpretując wydajność różnych technik analizy tekstu trzeba wziąć pod uwagę, że stanowią one końcowy etap całego procesu i silnie zależą od wyników analizy pytania, która nie jest doskonała. W szczególności, testy pokazały (podrozdział 3.3), że w najlepszym wariancie 15,65% pytań zostaje przypisany zły typ, a 17,81% wyników wyszukiwania nie zawiera oczekiwanego dokumentu. Wydaje się bardzo mało prawdopodobne, by moduł rozpoznawania nazw, na którym skupia się niniejsza praca, doprowadził do dobrych odpowiedzi w tych przypadkach. Aby przetestować ten moduł niezależnie od ewentualnych błędów popełnianych na poprzednich etapach przetwarzania pytania, można poprawić ich wyniki, sztucznie zastępując niewłaściwy typ pytania i/lub dołączając oczekiwany dokument do rezultatów wyszukiwania. Jest to możliwe dzięki dostępności tej informacji dla treningowego zbioru pytań (punkt 5.1.3). W ten sposób etap rozpoznawania nazw może być oceniany tak, jak gdyby analiza pytania działała idealnie. Trzeba tylko zwrócić uwagę, że takie podejście faworyzuje NER kosztem DeepER, ponieważ opis zbioru pytań zawiera typ nazwy własnej, ale brakuje w nim (wykorzystywanego przez głębokie rozpoznawanie nazw) synsetu określającego centrum pytania.

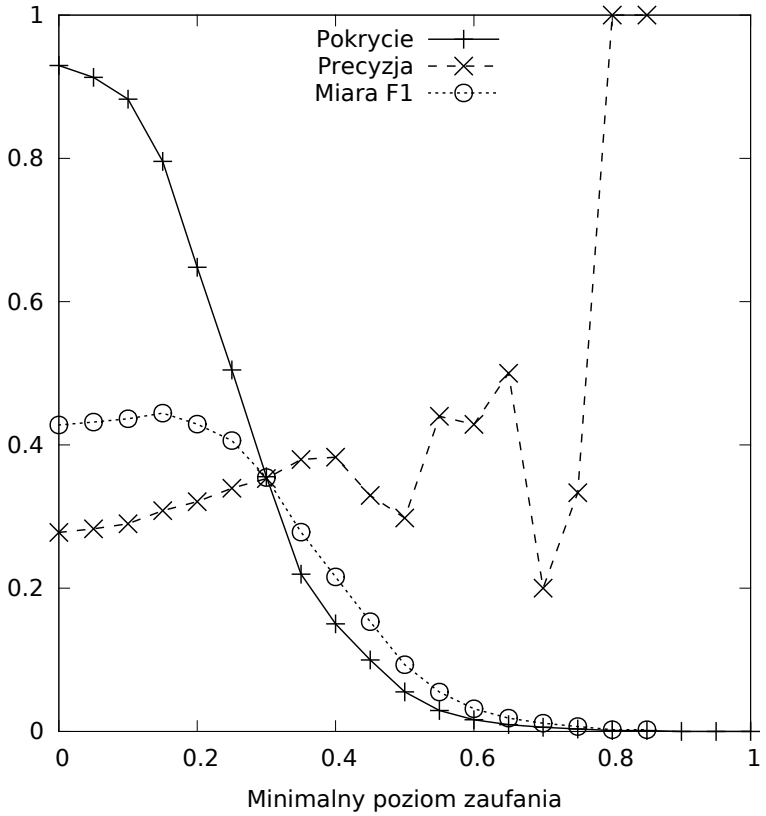
Kolejne podrozdziały opisują trzy grupy eksperymentów sprawdzających, jak na poprawność odpowiedzi wpływają kolejno: liczba dokumentów i techniki rozpoznawania nazw, minimalny poziom zaufania oraz sposób wydobywania kontekstu.

5.3.1. Wpływ liczby dokumentów i techniki rozpoznawania nazw

Celem pierwszego eksperymentu było sprawdzenie, w jaki sposób liczba dokumentów pobranych do analizy z wyników wyszukiwania wpływa na wydajność całego procesu. W tym eksperymencie błędy klasyfikacji pytań są poprawiane zgodnie z wyjaśnieniami podanymi powyżej. Przebadano dwa warianty: z uzupełnianiem zbioru o oczekiwany dokument i bez niego. Rysunek 5.2 przedstawia wyniki uzyskane dla różnych technik rozpoznawania nazw. Jak widać, jeśli pobrany zbiór zawiera oczekiwany dokument, to powiększanie go o nowe dokumenty nieznacznie polepsza pokrycie, ale precyzja spada dość gwałtownie. Wynika to z faktu, że dołączane artykuły w większości nie zawierają odpowie-



Rysunek 5.2. Wydajność odpowiadania na pytania w zależności od rozmiaru zbioru dokumentów pobieranych z wyszukiwarki i podlegających pełnej analizie. Wykresy dla uruchomień z poprawką gwarantującą obecność artykułu źródłowego w zbiorze (po lewej) i bez niej (po prawej) oraz trzech technik rozpoznawania nazw – tradycyjny NER (*Nerf*, *Liner2*) i DeepER.



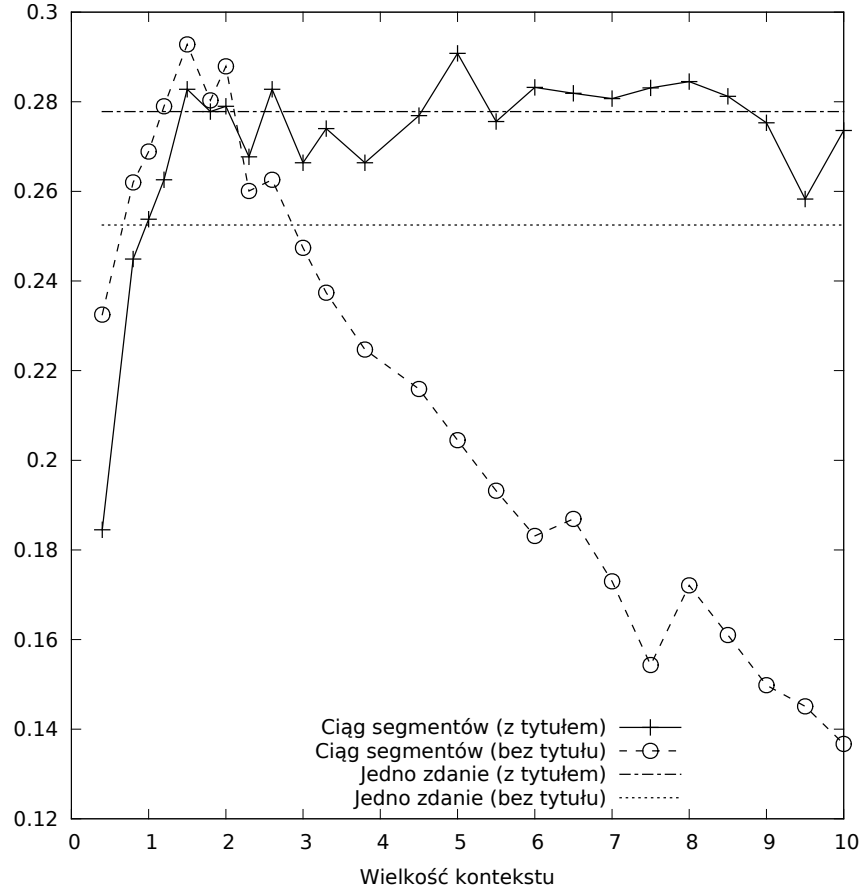
Rysunek 5.3. Poprawność odpowiedzi RAFAELa w zależności od minimalnego poziomu zaufania.

dzi, a wprowadzają jedynie szum. Z drugiej strony, w bardziej realistycznej sytuacji braku gwarancji obecności źródłowego dokumentu, rozszerzanie zbioru przynosi znaczne korzyści, szczególnie dla małych jego rozmiarów. Dla zbiorów większych niż 50 elementów szum zaczyna znowu dominować i osłabiać precyzję. Według miary F1 optymalna wielkość zbioru to 20 dokumentów.

W obu przypadkach głębokie rozpoznawanie nazw daje zauważalnie lepsze wyniki niż oba narzędzia NER. Jeśli chodzi o precyzję, to zysk jest niewielki, ale pokrycie różni się dwukrotnie. Łatwo to wyjaśnić – *Nerf* i *Liner2* nie są w stanie obsłużyć typu `UNNAMED_ENTITY`, którego udział w zbiorze treningowym wynosi 36%.

5.3.2. Wpływ minimalnego poziomu zaufania

Kolejny eksperyment miał na celu sprawdzenie, jak na otrzymywane rezultaty wpływa minimalny poziom zaufania (wartość indeksu Jaccarda), opisany w punktach 3.5.2 i 3.5.3. Jego regulacja mogłaby się okazać przydatna w zastosowaniach, w których oczekuje się wysokiej precyzji kosztem mniejszego pokrycia. Wykres 5.3 przedstawia wyniki ewaluacji otrymane z wykorzystaniem poprawek modelu pytania. Ogólnie system zachowuje się zgodnie z oczekiwaniami, jednak wartości liczbowe rozczarowują. W miarę podwyższania minimalnej wartości zaufania precyzja rośnie stosunkowo wolno, osiągając 38% na poziomie zaufania 0,40. Dzieje się to jednak w obszarze szybkiego



Rysunek 5.4. Wydajność RAFAELa w zależności od strategii wydobywania kontekstu, bazującej na pojedynczym zdaniu lub ciągu segmentów o określonej długości. Dla obu technik wykreślono wyniki z dodanym tytułem artykułu i bez niego.

spadku pokrycia do 22%, dlatego trudno uznać tę sytuację za korzystną. Według miary F1 optymalna wartość tego wskaźnika to 0,2.

5.3.3. Wpływ metody wydobywania kontekstu

Jeszcze jeden parametr wart przebadania to metoda wydobywania kontekstu opisana w punkcie 3.5.1. Przypomnijmy, że odpowiedź jest wybierana na podstawie podobieństwa treści pytania i kontekstu potencjalnej odpowiedzi. Kontekst ten może być utworzony jako pojedyncze zdanie lub ciąg segmentów o określonej długości. Dla obu tych rozwiązań bierze się również pod uwagę dodanie tytułu, często obecnego w kontekście za pośrednictwem anafory. Rysunek 5.4 przedstawia wartość precyzji (pokrycie nie zależy od kontekstu) dla tych czterech kombinacji. Długość ciągu wyrażono na poziomej osi jako wielokrotność długości treści pytania. Widać na nim, że uwzględnienie tytułu w istocie znacząco poprawia precyzję. Wpływ wyrażeń anaforycznych ujawnia się w przypadku ciągu segmentów – im większy rozmiar zastosujemy, tym gorsze otrzymujemy wyniki, podczas gdy uwzględnienie tytułu neutralizuje ten problem. Co ciekawe, najwyższą wartość precyzji system osiąga dla krótkie-

go ciągu segmentów (1,5 * długość treści pytania). Ponieważ jednak różnice są niewielkie, do ostatecznej oceny użyto prostszej wersji, czyli pojedynczego zdania z dodanym tytułem.

5.4. Wyniki końcowe

Dla przeprowadzenia ostatecznej oceny wykorzystano oddzielny zbiór pytań, opisany w punkcie 5.1.2. Nie zastosowano korekty analizy pytania, ponieważ testowano wydajność systemu jako całości. Klasyfikacja pytań odbywa się poprzez dopasowywanie wzorców i analizę centrum z wyborem pierwszego znaczenia. Zapytania powstają z użyciem prostej alternatywy z dopuszczalnym dopasowaniem rozmytym o długości 3 znaków (niezmienny prefiks). System wykorzystuje 20 pierwszych dokumentów z listy zwróconej przez wyszukiwarkę. Nie stosuje się minimalnego poziomu zaufania, a kontekst powstaje z pojedynczego zdania z dodaniem tytułu. Testowane rozwiązania różnią się jedynie zastosowanym podejściem do rozpoznawania nazw. W tym zakresie uwzględniono następujące warianty:

1. wyrażenia numeryczne i wielkości fizyczne (*Quant*),
2. narzędzia NER: *Nerf* i *Liner2*,
3. głębokie rozpoznawanie nazw,
4. podejście hybrydowe, łączące wzmianki pozyskane ze wszystkich wymienionych wyżej źródeł.

Tabela 5.3 przedstawia precyzję, pokrycie, miarę F1 i MRR dla otrzymanych wyników. Ostatnia z użytych miar jest powszechnie stosowana przy ewaluacji wyszukiwarek i uwzględnia całą listę proponowanych odpowiedzi, przy czym im dalszą pozycję na liście zajmuje prawidłowy wynik, tym niższe ma znaczenie. MRR dla zbioru pytań oblicza się jako średnią odwrotności pozycji prawidłowej odpowiedzi na liście rankingowej:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{r_i}, \tag{5.1}$$

gdzie:

- Q to zbiór pytań,
- r_i to pozycja prawidłowej odpowiedzi na liście rankingowej dla i-tego pytania.

Gdy na i-tej liście nie występuje oczekiwana odpowiedź, to przyjmuje się, że $\frac{1}{r_i} = 0$. Aby obliczyć miarę MRR, odpowiedzi uporządkowano według malejącego poziomu zaufania, czyli wartości indeksu podobieństwa Jaccarda. Ponieważ zwrócone listy mogą być bardzo długie, ich zgodność z oczekiwanym wynikiem sprawdzono używając ewaluacji automatycznej. Aby oszacować odchylenie standardowe, obliczono te wielkości na 500 podzbiorach pytań typu *bootstrap*[6]. Dodatkowo podano wartość precyzji według ewaluacji automatycznej.

[6] Oznacza to, że ze zbioru pytań Q wylosowano 500 prób ze zwracaniem o wielkości $|Q|$ i dla każdej z nich obliczono wskaźniki wydajności. Pierwiastek z wariancji otrzymanego

Jak widać w tabeli, tylko niewielka część pytań (9,5%) jest obsługiwana przez narzędzie *Quant*. Wynika to z małej liczności tego typu pytań w zbiorze testowym, co odzwierciedlają również wysokie wartości odchyleń. Rozpoznawanie bytów nazwanych znajduje zastosowanie w nieco mniej (*Liner2*) lub więcej (*Nerf*) niż połowie pytań. Kiedy użyjemy DeepER, wskaźnik pokrycia osiąga prawie 73%, przy czym różnica w precyzji w stosunku do technik NER nie jest istotna statystycznie. Lepsze pokrycie wynika z dużego udziału pytań typu `UNNAMED_ENTITY`, niedostępnych dla rozwiązań NER, w zbiorze testowym. Wskaźnik ten osiąga maksimum w przypadku rozwiązania hybrydowego (90%), ale dzieje się to kosztem niższej precyzji (33%). Z drugiej strony, jeśli weźmiemy pod uwagę całą listę odpowiedzi i wyrazimy jej poprawność miarą MRR, rozpoznawanie nazw własnych zyskuje przewagę. Gdy oceniać rozwiązania pod kątem miary F1, to najlepsze okazuje się rozwiązanie hybrydowe z wynikiem 0,48.

Tam, gdzie było to możliwe[7], podano dla porównania wyniki automatycznej ewaluacji. Zgodnie z oczekiwaniami procedura ta podaje zaniżoną wartość precyzji ze względu na nierozpoznawanie niektórych poprawnych odpowiedzi, ale różnica ta pozostaje poniżej 5%.

zestawu 500 wyników stanowi oszacowanie odchylenia standardowego wyniku dla całego zbioru.

[7] Odpowiedzi oznaczane przez programy *Quant* mogą zawierać wielkości lub daty, które nie mogą być sprawdzone przez automatyczną ewaluację.

	Pokrycie		Precyzja			Miara F1		MRR	
			ręcznie		automatycznie				
rozpoznawanie nazw	**wartość**	**odchylenie**	**wartość**	**odchylenie**	**wartość**	**wartość**	**odchylenie**	**wartość**	**odchylenie**
Quant	9,55%	1,19%	27,27%	5,6%	-	0,1415	0,0162	0,2671	0,0486
Nerf	56,25%	2,12%	34,88%	2,73%	-	0,4306	0,0213	0,3366	0,0229
Liner2	45,31%	2,05%	39,08%	2,90%	34,10%	0,4197	0,0188	0,4136	0,0270
DeepER	72,92%	1,88%	35,24%	2,23%	33,89%	0,4751	0,0214	0,3280	0,0199
Hybrydowe	89,58%	1,24%	33,14%	2,01%	-	0,4838	0,0221	0,3557	0,0188

Tabela 5.3. Poprawność odpowiadania na pytania przez system RAFAEL w zależności od zastosowanej techniki rozpoznawania nazw: tylko liczby i wielkości (*Quant*), nazwy własne (*Nerf*, *Liner2*), głębokie rozpoznawanie nazw (*DeepER*) i podejście hybrydowe. Przedstawiono wartości pokrycia, precyzji, miary F1, MRR oraz ich odchylenia standardowe. Podano również wartości precyzji uzyskane przez automatyczną ewaluację.

Rozdział 6

Wnioski

Niniejszy rozdział zawiera analizę uzyskanych rezultatów. Wskazano mocne strony zastosowanych rozwiązań i wyjaśniono najważniejsze przyczyny pojawiania się błędnych odpowiedzi. Następnie zaproponowano dalsze prace, które mogą polepszyć wydajność systemu. Podrozdział 6.3 podsumowuje przeprowadzone badania.

6.1. Analiza wyników

Największą zaletą zastosowania w RAFAELu głębokiego rozpoznawania nazw zamiast rozpoznawania nazw własnych jest znacząco lepsze pokrycie zbioru pytań. Tabela 6.1 pokazuje przykłady pytań, których analiza doprowadziła do ich odrzucenia przez techniki bazujące na NER, gdyż ich centra nie są obejmowane przez kategorie nazw własnych. Dla tych samych pytań użycie DeepER doprowadziło do uzyskania poprawnych rozwiązań.

Pod względem różnic w precyzji wyniki uzyskane na zbiorze treningowym znacząco różnią się od ostatecznych. Można to wyjaśnić różnicami w częstości występowania pytań poszczególnych typów w użytych zestawach pytań. Zbiór treningowy zawiera znacznie więcej pytań rozpoczynających się niejednoznacznym zaimkiem z następującym po nim centrum pytania, np. *Który poeta ... ?*. Pytania takie pozwalają uzyskać konkretny synset, co zwiększa precyzję w technice DeepER. Tymczasem elementy zbioru testowego znacznie częściej mają jednoznaczną konstrukcję, jak np. *Kto ... ?*, gdzie uzyskany synset odpowiada ogólnemu typowi nazwy własnej (`PERSON`).

Jako że RAFAEL jest pierwszym polskim systemem QA, który w odpowiedzi podaje nazwy bytów, a nie dokumenty, nie jest możliwe bezpośrednie porównanie go do żadnego istniejącego rozwiązania. Pamiętajmy jednak, że zbiór testowy powstał w oparciu o pytania opublikowane przez Marcińczuka i innych (2013a) i użyte do ewaluacji systemu wyszukiwania dokumentów (Marcińczuk i inni, 2013b), omówionego w podrozdziale 2.4. W systemie tym podstawowa konfiguracja osiągnęła wartość wskaźnika a@1 (udział pytań, na które odpowiedź zawiera pierwszy dokument, co odpowiada precyzji w tabeli 5.3) na poziomie 26,09%. Biorąc pod uwagę bliskość wystąpień słów kluczowych (metoda MCSW) udało się podnieść ten wynik do 38,63%. Widzimy zatem, że RAFAEL, choć rozwiązuje znacznie ambitniejsze zadanie, we wszystkich badanych wersjach osiąga lepsze wyniki niż system odniesienia. RAFAEL z narzędziem *Liner2* uzyskuje precyzję wyższą niż najlepsza metoda stosowana dotąd na tym zbiorze (MCSW).

Pytanie	
Centrum	**Odpowiedź**
Który żleb uważany jest za najbardziej lawiniasty w całych Karpatach?	
żleb	Pusty Żleb
Jaki owad pożera liście owadożernej rosiczki?	
owad	Piórolotek bagniczek
Jaki przyrząd pozwala na pomiar współczynnika załamania światła?	
przyrząd	Refraktometr Abbego
Jaki związek chemiczny służy do otrzymywania boru o wysokiej czystości?	
związek chemiczny	Jodek boru
Jaką metodą łamano szyfry Enigmy przed wynalezieniem cyklometru?	
metoda	Metoda rusztu

Tabela 6.1. Przykłady pytań, na które udało się znaleźć prawidłowe odpowiedzi tylko dzięki technice DeepER. Ich centra leżą poza kategoriami nazw własnych.

Inny punkt odniesienia stanowią wyniki osiągnięte przez badaczy uczestniczących w rywalizacji na konferencjach TREC. Zarówno ogólny typ zadania (odnajdywanie zwięzłych odpowiedzi w zadanym korpusie tekstowym), jak i natura zadawanych pytań (proste fakty) są zgodne z zadaniem postawionym przed RAFAELem. Oczywiście konkretne wartości nie mogą być bezpośrednio porównywane, ale warto zwrócić uwagę, że mamy do czynienia z tym samym rzędem wielkości – mediana dokładności[1] systemów nadesłanych na ostatnią konferencję TREC (Dang i inni, 2007) to około 24%, podczas gdy w przypadku RAFAELa (w wersji hybrydowej) wynik ten wynosi 30%.

Mimo to niska precyzja udzielanych odpowiedzi wskazuje, że daleko jeszcze do całkowicie zadowalającego rozwiązania postawionego problemu. Aby odnaleźć przyczyny błędów, część nieprawidłowych odpowiedzi poddano ręcznej analizie. Poniżej przedstawiono najczęściej pojawiające się problemy.

6.1.1. Błędy głębokiego rozpoznawania nazw

Technika *DeepER* jest ograniczana głównie przez brak ujednoznaczniania słów, co utrudnia wnioskowanie oparte o WordNet. Problem ten właściwie dotyczy wszystkich metod rozpoznawania nazw. Za przykład weźmy słowo *kot*. Oprócz zwierzęcia może ono oznaczać teatr, jezioro, wieś, odznakę (*KOT*) lub nazwisko jednej z dziesięciu osób wymienionych w polskiej Wikipedii. Człowiek zazwyczaj potrafi wybrać najbardziej prawdopodobne w danym kontekście znaczenie lub zakłada najczęstsze z nich, ale system traktuje wszystkie jako równo prawdopodobne. Bardzo wieloznaczne nazwy wprowadzają do procesu szum, ponieważ pasują do wielu pytań. Także Kazama i Torisawa (2007) uważają problem niejednoznacznych nazw za jeden z najpoważniejszych w przypadku rozpoznawania nazw z wykorzystaniem Wikipedii. Zwracają uwagę na fakt, że problem będzie narastał w miarę powiększania encyklopedii i wydłużania list artykułów przypisanych do pojedynczej nazwy.

[1] Dokładność rozumiana jest jako udział poprawnych odpowiedzi w całym zbiorze pytań, czyli iloczyn precyzji i pokrycia.

Brak ujednoznaczniania słów jest przyczyną, dla której po zastosowaniu głębokiego rozpoznawania nazw nie obserwujemy oczekiwanego przyrostu precyzji względem rozwiązań NER. Warto zwrócić uwagę na podobne zjawisko w pierwszym podejściu do zastosowania bogatszych niż nazwy własne kategorii do odpowiadania na pytania, które zaprezentował Mann (2002). Także w tamtym rozwiązaniu zaobserwowano znacząco większe pokrycie bez istotnej różnicy w precyzji. Jeden z pomysłów na walkę z problemem niejednoznaczności przedstawiono jako propozycję dalszych prac w podrozdziale 6.2.

Oprócz niedostatecznej precyzji rozróżniania bytów, problem stanowi również zbyt duża czułość rozpoznawania i wskazywanie wzmianek tam, gdzie w rzeczywistości ich nie ma. Rozważmy pytanie *Kto pracował jako misjonarz przed rozpoczęciem kariery malarskiej?*. W artykule zawierającym odpowiedź (*Vincent van Gogh*) znajdziemy następujące zdanie: *Od 1879 pracował jako misjonarz w górniczym zagłębiu w Belgii.*[2] Może wydać się zaskakującym fakt, że metoda DeepER prowadzi do rozpoznania w tym zdaniu nazwy własnej określającej osobę. Jest to *Od*, zwany także *Odur*, bóg z mitologii nordyckiej. Próbą walki z tym zjawiskiem jest opisany w podrozdziale 4.3 wymóg, by wzmianka oznaczana jako nazwa własna rozpoczynała się wielką literą. Widać jednak na tym przykładzie, że w przypadku pierwszego słowa w zdaniu to rozwiązanie nie wystarcza. Można także uzależnić oznaczanie wzmianki od zgodności z wynikami tagowania – w tym wypadku segment *Od* został oznaczony jako przyimek, a zatem nie może oznaczać osoby. Niestety, ponieważ bóg *Od* nie występuje w słowniku analizatora morfologicznego, w miejscu jego nazwy zawsze będzie rozpoznawany przyimek, a więc nie odpowiemy prawidłowo na pytanie *Który nordycki bóg był mężem Freyi?*

6.1.2. Błędy rozpoznawania nazw własnych

W przypadku modułu rozpoznawania nazw własnych (*Quant*) najwięcej błędów powoduje ignorowanie typów wielkości. Przykładowo, odpowiadając na pytanie *Ile metrów miała mieć wieża Eiffla według pierwszego projektu?*, natrafimy na artykuł *Wieża Eiffla* i zdanie *W 1884 został opracowany pierwszy projekt wieży o wysokości 300 m.* W zdaniu tym *Quant* odnajduje dwie liczby – *1884* i *300*. Otrzymują one tę samą pozycję w rankingu. W praktyce problem ten najczęściej przejawia się właśnie przez zwracanie roku jako odpowiedzi, gdy pytanie dotyczy wielkości innego rodzaju. Aby uniknąć takich błędów należałoby przygotować listę możliwych jednostek miary dla każdego typu wielkości. Dzięki temu byłoby jasne, że pytanie rozpoczynające się frazą *Ile metrów* dotyczy długości, a liczba 1884 nie może stanowić odpowiedzi, skoro nie następuje po niej odpowiednia jednostka. Rozwiązanie to nie jest bardzo skomplikowane koncepcyjne, ale wymaga zgromadzenia dużego zbioru pytań o wielkości różnych typów – w zbiorze testowym używanym w niniejszej pracy takich pytań jest tylko 6.

Zewnętrzne moduły NER, *Nerf* i *Liner2*, powodują błędy odpowiedzi poprzez nieoznaczanie części występujących w tekstach nazw własnych, jak również błędny wybór kategorii. Wyniki rozpoznawania nazw dla przykładowego

[2] Wszystkie przykłady w tym rozdziale pochodzą z polskiej Wikipedii: `http://pl.wikipedia.org/`

tekstu z punktu 3.2.2 umieszczono w tabeli 3.1. Ponieważ narzędzia te bazują na uczeniu maszynowym, jedną z przyczyn może być niedostateczna ilość danych trenujących. Przykładowo *Liner2* wykorzystuje 2000 dokumentów z ręcznie oznaczonych korpusów *KPWr* (Korpus Języka Polskiego Politechniki Wrocławskiej) i *CEN* (Corpus of Economic News), a rozpoznaje tak szczegółowe typy nazw własnych, jak nazwa wyspy czy regionu historycznego. Wydaje się prawdopodobne, że wśród tych dokumentów niewiele jest nazw tego typu i rozbudowanie danych treningowych wpłynęłoby pozytywnie na wydajność programu. Trudność stanowi tutaj oczywiście nakład pracy konieczny do ręcznego znakowania, ale np. Balasuriya i inni (2009) pokazali, jak można przekształcić korpus Wikipedii w dane trenujące dla narzędzi rozpoznawania nazw o równie wysokiej jakości, jak dane oznakowane ręcznie.

6.1.3. Inne błędy

Inne problemy dotyczą w większości etapu dopasowywania zdania pytającego i stanowiącego potencjalną odpowiedź. Często zdarza się, że informacja podana w pytaniu w zwartej formie w tekście jest rozrzucona po wielu zdaniach. Przykładowo, dla pytania *Kiedy Albert Einstein opublikował szczególną teorię względności?*, odpowiedź znajdziemy we fragmencie artykułu o tytule *Albert Einstein*:

> *Rok 1905 jest określany jako Annus mirabilis (cudowny rok) Einsteina. Wtedy ten nieznany w środowisku fizyków szwajcarski urzędnik patentowy opublikował kilka prac przełomowych dla fizyki. Jego publikacja Zur Elektrodynamik bewegter Körper (O elektrodynamice ciał w ruchu) wprowadza nową teorię, nazwaną później szczególną teorią względności, która dzięki nowemu spojrzeniu na czas i przestrzeń rozwiązywała obserwowaną niezależność prędkości światła od obserwatora, wprowadzała związek między masą a energią.*

Widać, że informacja o opublikowaniu teorii jest dosyć oddalona w tekście od określenia czasu. Dodatkowo, wyjątkowo długie zdanie o publikacji zawiera dużo informacji niewymienionych w pytaniu. Jest to częsta przyczyna udzielania złych odpowiedzi na podstawie Wikipedii: bardzo szczegółowe opisy utrudniają odnalezienie najistotniejszych informacji.

Trudności sprawia również fleksyjny charakter języka polskiego, tj. odmiana nazw własnych. Ponieważ niemożliwym jest, aby słownik analizatora morfologicznego zawierał wszystkie nazwiska, w przypadku nieznanych nazw używany jest moduł odgadujący prawidłowy lemat i interpretację. Na przykład artykuł *Atom* zawiera następujące zdanie: *W 1913 roku Frederick Soddy, badając produkty rozpadu promieniotwórczego, odkrył, że atomy każdego pierwiastka mogą występować w kilku odmianach różniących się nieco masą atomową*. Segment *Frederick* otrzymuje prawidłową interpretację, natomiast segment *Soddy* zostaje uznany za formę rzeczownika *sodd* w liczbie mnogiej. Błędne oznakowanie pociąga za sobą konsekwencje również na wyższych poziomach; z powodu niezgodności liczby segmenty te nie zostaną połączone w grupę składniową, jak dzieje się to z innymi parami imię – nazwisko.

Jeżeli we wspólnym kontekście (np. w jednym zdaniu) występuje wiele nazw pasujących do poszukiwanego typu, model *bag of words* nie wystarcza do wskazania, która z nich powinna być zwrócona jako odpowiedź. Zwróćmy uwagę na pytanie *Z którym państwem Laos graniczy na wschodzie?*. Odpowiedź na to pytanie zawiera artykuł *Laos* w zdaniu *Na północy Laos graniczy z Mjanmą i Chinami, na wschodzie z Wietnamem, na południu z Kambodżą, a na zachodzie z Tajlandią.* Widać, że oprócz Laosu (który zostanie dopasowany do treści pytania) w zdaniu znajduje się jeszcze 5 państw. Ponieważ system RAFAEL ignoruje strukturę zdania, wszystkie one otrzymują tę samą pozycję w rankingu. Rozwiązaniem mogłoby być zastosowanie którejś z metod dopasowywania strukturalnego, omówionych w punkcie 2.3.2.

Najpowszechniejszy problem dotyczy oceny podobieństwa słów. Niektóre pary słów występujących w treści pytania i kontekście odpowiedzi mogą być ze sobą związane morfologicznie lub znaczeniowo, ale różnić się co do lematu, np. *łowca* i *łowiectwo* lub *myśliwy*. Możliwe podejścia do tego zagadnienia, wskazującego oczywisty kierunek dalszych prac, omówiono w następnym podrozdziale.

6.2. Dalsze prace

Systemy odpowiadania na pytania to jedne z najbardziej złożonych systemów budowanych w dziedzinie przetwarzania języka naturalnego. Także RAFAEL składa się z wielu wzajemnie zależnych modułów, dlatego trudno byłoby wymienić wszystkie sposoby, na które można go rozbudować i ulepszyć. Z powodu ograniczonych zasobów przy budowie systemu skupiono się na aspekcie rozpoznawania nazw, realizując funkcję podobieństwa pytania i kontekstu w formie dość podstawowej. Wydaje się naturalne, że w pierwszej kolejności to ten element powinien być rozbudowany. Analiza wyników w poprzednim podrozdziale wykazała, że obliczanie podobieństwa na poziomie lematów nie wystarcza – trzeba odnieść się do ich znaczenia. Podejścia semantyczne do problemu dopasowywania omówiono w punkcie 2.3.3. Oczywiście nie wszystkie takie rozwiązania można przenieść bezpośrednio na grunt języka polskiego ze względu na brak odpowiednich zasobów. Dysponujemy jednak polską bazą WordNet, która sprawdziła się w niniejszej pracy. Na zasobie tego typu opiera się wiele miar podobieństwa (Budanitsky i Hirst, 2006). Najprostsza z nich polega na uznaniu dwóch słów za identyczne, jeśli tylko odpowiadające im leksemy należą do tego samego synsetu. Rozwiązałoby to wspominany wyżej problem słów *łowca* i *myśliwy*, ale znów pojawia się tu zagadnienie wybierania znaczenia. Leksem *łowca* występuje w dwóch wariantach i to drugi z nich pozsiada oczekiwany związek ze słowem *myśliwy*. W praktyce w systemach QA zastosowano np. łańcuchy leksykalne (Moldovan i Novischi, 2002), gdzie związek dwóch słów ocenia się na podstawie długości najkrótszej ścieżki pomiędzy odpowiadającymi im synsetami w grafie WordNet (szersze omówienie w punkcie 3.5.2). Rozwiązanie to wykorzystuje również treść definicji, których niestety brakuje w przypadku większości synsetów w bazie *plWordNet*. Powiązanie słów można także określić na podstawie analizy ich sąsiedztwa w dużym

korpusie (Turney i Pantel, 2010) – duże podobieństwo kontekstów wystąpień może wskazywać na powiązanie znaczeniowe.

Zróżnicowana precyzja odpowiedzi i pokrycie pytań w zależności od wybranej techniki rozpoznawania nazw wskazuje, że dobre wyniki mogłoby przynieść używanie na tym etapie różnych rozwiązań w zależności od typu pytania. Wydaje się oczywiste, że niektóre narzędzia mogą się okazać lepsze przy rozpoznawaniu np. dat, a inne nazw geograficznych. Aby to jednak stwierdzić, trzeba dysponować zbiorem pytań, który zawiera wiele elementów każdego typu. Niestety, jak pokazano w tabeli 5.2, w wykorzystywanym zbiorze treningowym mamy do czynienia z bardzo nierównym rozkładem – niektóre typy zawierają zaledwie kilka pytań.

Jeśli chodzi o technikę DeepER, to główną przeszkodą na drodze do lepszej jakości rozpoznawania jest wieloznaczność słów. Oczywiście rozwiązaniem mogłoby być narzędzie zdolne do precyzyjnego ujednoznaczniania wszystkich słów w tekście, ale trudno oczekiwać takiej możliwości w najbliższej przyszłości. Warto natomiast rozważyć modyfikację techniki wyboru synsetu przy generowaniu biblioteki bytów. Zamiast zapisywać synsety odpowiadające dokładnie słowom w definicji można najpierw nieco przesunąć się w górę hierarchii hiperonimii, w pewnym stopniu rezygnując z dokładności na rzecz jednoznaczności. W przykładzie pokazanym na rysunku 4.1 oznaczałoby to, że zamiast wybierać między synsetem <prezydent.1, prezydent miasta.1> a <prezydent.2> decydujemy się na <urzędnik.1, biuralista.1>, który pokrywa oba te znaczenia.

Przedstawione powyżej pomysły wynikają z przeprowadzonej analizy wyników, jednak można wskazać ich więcej, np. agregacja wielu wzmianek odnoszących się do tego samego bytu, dokładniejsze szacowanie poziomu ufności, generowanie kontekstu z uwzględnieniem anafor itd.

6.3. Realizacja celów

W niniejszej pracy przedstawiono RAFAELa – system zdolny do odpowiadania na pytania o proste fakty na podstawie otwarto-dziedzinowej bazy tekstowej. Powróćmy teraz do założonych celów badań (rozdział 1) i przyjrzyjmy się, w jaki sposób zostały one zrealizowane.

1. Zbudowano pierwszy system działający na tekstach w języku polskim, zdolny do udzielania odpowiedzi nie w formie całych dokumentów, ale samych nazw poszukiwanych bytów. Przeprowadzone testy wykazały, że poprawność uzyskiwanych odpowiedzi jest zbliżona do analogicznych rozwiązań dla języka angielskiego.
2. Przy projektowaniu systemu korzystano z rozwiązań obecnych w literaturze, jednak większość z nich opracowana była z myślą o języku angielskim. Z tego powodu podczas budowy niektórych modułów (np. analizy pytania, wyszukiwania dokumentów i generowania kontekstu) konieczne było porównanie wydajności istniejących rozwiązań w przypadku zastosowania do tekstów w języku o bogatej fleksji i swobodnym szyku zdania, tj. polskim. W niektórych przypadkach stosunkowo

proste rozwiązania dały dobre rezultaty (np. zapytania rozmyte przy wyszukiwaniu), inne problemy z tego zakresu wymagają dalszych prac (np. dopasowywanie zdań o swobodnym szyku).

3. Szczególną uwagę zwrócono na moduł rozpoznawania nazw, który pozwala na wskazanie w analizowanych tekstach wszystkich wzmianek bytów spełniających wymagania narzucone w pytaniu. Obok tradycyjnego rozpoznawania nazw własnych została zaproponowana i zaimplementowana nowa technika, nazwana głębokim rozpoznawaniem nazw. Wyniki przeprowadzonych eksperymentów wskazują, że głębokie rozpoznawanie nazw pozwala na znaczne poprawienie jakości odpowiedzi poprzez rozszerzenie zakresu obsługiwanych pytań poza typy nazw własnych przy zachowaniu tej samej precyzji odpowiedzi.
4. Aby było możliwe zastosowanie głębokiego rozpoznawania nazw, wprowadzono metodę pozyskiwania skojarzeń nazw i synsetów z definicji encyklopedycznych, której zastosowanie do polskiej Wikipedii doprowadziło do powstania biblioteki bytów. Baza ta zawiera ponad 800 tysięcy deskryptorów bytów i umożliwia rozpoznawanie typów wzmianek w tekście na poziomie synsetów WordNet, a nie ogólnych kategorii nazw własnych.

W toku prowadzonych badań do ilościowej oceny osiągniętych rezultatów konieczne okazało się rozwinięcie metod ewaluacji. Doprowadziło to do osiągnięcia dwóch celów dodatkowych – nieplanowanych, ale wartościowych z punktu widzenia dalszego rozwoju dziedziny.

1. Skompletowano zestaw danych ewaluacyjnych, składający się z: dużego korpusu tekstowego na bazie polskiej Wikipedii, zbioru pytań o proste fakty, powstałego w oparciu o wcześniejsze publikacje, oraz odpowiedzi, ręcznie odnalezionych w korpusie. Zestaw ten może posłużyć do porównywania wydajności różnych systemów odpowiadania na pytania w języku polskim.
2. Dzięki technice głębokiego rozpoznawania nazw możliwa stała się automatyczna ewaluacja, która pozwala na rozpoznanie zgodności odpowiedzi oczekiwanej i otrzymanej nawet wtedy, gdy ich reprezentacje tekstowe całkowicie się różnią. Technika ta wspomogła określenie optymalnych ustawień niektórych parametrów procesu w drodze eksperymentów, których ręczna ocena byłaby zbyt długotrwała.

Uzyskanie streszczonych powyżej pozytywnych wyników otwiera nowe perspektywy badawcze. Po pierwsze, choć system RAFAEL jest kompletny, możliwości jego rozbudowy i modyfikacji są bardzo duże. Mogą one prowadzić nie tylko do lepszych rezultatów na zbiorach testowych, ale również praktycznych zastosowań w uzyskiwaniu informacji z tekstowych baz danych. Taką drogę – od odpowiadania na pytania z teleturnieju do produktu komercyjnego – przeszedł najbardziej znany system QA, *IBM Watson*. Po drugie, wykorzystywanie przez system licznych narzędzi i zasobów lingwistycznych umożliwia testowanie ich wpływu na wydajność systemu. Takie eksperymenty pozwalają pokazać, na ile różnice osiągane w testach syntetycznych danego narzędzia przekładają się na rezultaty w praktycznym zastosowaniu. Po trzecie, osiągnię-

cie przez RAFAELa określonych wyników na ściśle zdefiniowanych zbiorach testowych może stanowić punkt odniesienia dla konkurencyjnych rozwiązań. Udostępnienie takich możliwości powinno przyczynić się do ożywienia badań nad odpowiadaniem na pytania w języku polskim.

Dodatek – słownik pojęć

W niniejszym dodatku przedstawiono krótkie wyjaśnienia pojęć, które często pojawiają się w pracy, a ich znaczenie może nie być oczywiste dla Czytelnika lub wymagać doprecyzowania.

- **słowo** (inaczej wyraz) to *znak językowy mający jakieś znaczenie, odpowiadający jednemu pojęciu* (Dubisz, 2006). Termin ten jest wieloznaczny, dlatego używa się również następujących, węższych, pojęć:
  — **leksem** to *wyraz traktowany jako abstrakcyjna jednostka systemu słownikowego języka* (Dubisz, 2006). Leksem jest zatem słowem w sensie elementu słownika; może występować w kilku wariantach, odpowiadających różnym znaczeniom.
  — **forma bazowa** (zwana także lematem) to *podstawowa (hasłowa) forma danego leksemu* (Przepiórkowski i inni, 2012), np. dla rzeczownika to mianownik liczby pojedynczej, np. *kot*.
  — **forma ortograficzna** (zwana także formą powierzchniową) to postać w jakiej słowo pojawia się w tekście, czyli łańcuch znaków, np. *kotom*.
  — **interpretacja** to *informacja o części mowy (rzeczownik, czasownik itp.) i wartościach odpowiednich kategorii morfosyntaktycznych (rodzaju, przypadka, itp.)*. (Przepiórkowski, 2008). Zazwyczaj zapisuje się ją jako zbiór tagów morfosyntaktycznych, na przykład zapis *subst:pl:dat:m2* oznacza rzeczownik rodzaju męskiego ożywionego w liczbie mnogiej i celowniku.
  — **segment** to fragment tekstu, któremu można przypisać interpretację morfosyntaktyczną (Przepiórkowski, 2008). Zazwyczaj podział na segmenty jest oczywisty i bazuje na znakach odstępu i interpunkcyjnych, ale czasem jest bardziej skomplikowany, np. łańcuch *dałbym* ulega rozbiciu na *dał*, *by* i *m*[3]. W procesie analizy morfosyntaktycznej do segmentów przypisuje się możliwe interpretacje i lematy.
- **n-gram** to sekwencja *n* określonych słów następujących bezpośrednio po sobie, np. wyrażenie *dał kotu mleko* jest 3-gramem (trigramem).
- **byt** – *to, co jest, co istnieje w jakikolwiek sposób* (Dubisz, 2006; Stępień, 2001), np. *Warszawa*, *mleko*, *13* lub *Jan III Sobieski*.
- **nazwa** to *wyraz albo połączenie wyrazowe oznaczające kogoś lub coś; miano, nazwanie, określenie, termin* (Dubisz, 2006). Nazwę stanowi zatem słowo lub słowa, które wskazują na jakiś byt.

[3] Wynika to z faktu, że segmenty te można ułożyć w inny sposób bez zmiany znaczenia, np. *bym dał*.

- **nazwa własna** oznacza taką nazwę, która odnosi się do konkretnego bytu[4]. W niniejszej pracy lingwistyczną kategorię nazw własnych (rzeczowników własnych) rozszerzono o daty, liczby i wielkości, tak, by odpowiadało to najczęstszemu rozumieniu angielskiego pojęcia *named entities*. W polskiej literaturze przedmiotu obok takiego rozwiązania (Mykowiecka, 2007) spotyka się również określenia: byty nazwane (Przepiórkowski, 2008), jednostki nazewnicze (Przepiórkowski i inni, 2012) lub jednostki identyfikacyjne (Marcińczuk, 2012).
- **nazwa ogólna** to nazwa nienależąca do kategorii nazw własnych, a więc opisująca grupę bytów (inaczej *pojęcie*), np. *pies*.
- **wzmianka** to nazwa znajdująca się w określonym kontekście w dokumencie.
- **synset** to zbiór leksemów w określonych wariantach o identycznym lub prawie identycznym znaczeniu. Przykładowo, synset <zamek.6, suwak.1, zamek błyskawiczny.1, ekler.2, ekspres.3> grupuje leksem *zamek* w szóstym wariancie znaczeniowym, *suwak* w pierwszym itd.
- **prosty fakt** (ang. *factoid*) oznacza prostą i krótką informację o charakterze faktograficznym z dowolnej dziedziny. Dodajmy, że polskie słowo *faktoid* oznacza *kłamstwo lub półprawdę uznawaną za prawdę przez sam fakt ukazania się ich w druku* (Bralczyk, 2005), które to znaczenie jest częstsze w brytyjskiej odmianie języka angielskiego (Allen, 2000).

[4] Możliwa jest wieloznaczność, np. *Mruczek* lub *Praga*, jednak zazwyczaj nie jest ona zamierzona przez autora.

Bibliografia

Acedański, S. (2010). A morphosyntactic Brill Tagger for inflectional languages. *Proceedings of the 7th international conference on Advances in Natural Language Processing (IceTAL'10)*, strony 3–14. Springer-Verlag. (cyt. na s. 38)

Ahn, D., Jijkoun, V., Mishne, G., Müller, K., De Rijke, M., i Schlobach, S. (2004). Using Wikipedia at the TREC QA Track. *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*. National Institute of Standards and Technology (NIST). (cyt. na s. 25, 27, 32 oraz 74)

Allen, R., editor (2000). *The New Penguin English Dictionary*. Penguin Books. (cyt. na s. 116)

Andrenucci, A. i Sneiders, E. (2005). Automated Question Answering: Review of the Main Approaches. strony 514–519. IEEE Computer Society Washington. (cyt. na s. 15 i 28)

Armenska, J., Tomovski, A., Zdravkova, K., i Pehcevski, J. (2010). Information Retrieval Using a Macedonian Test Collection for Question Answering. *Proceedings of the 2nd International Conference ICT Innovations*, strony 205–214. Springer-Verlag. (cyt. na s. 24)

Atzeni, P., Basili, R., Hansen, D. H., Missier, P., Paggio, P., Pazienza, M. T., i Zanzotto, F. M. (2004). Ontology-based question answering in a federation of university sites: the MOSES case study. *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB'04)*. Springer-Verlag. (cyt. na s. 28)

Balasuriya, D., Ringland, N., Nothman, J., Murphy, T., i Curran, J. R. (2009). Named entity recognition in Wikipedia. *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, strony 10–18, Association for Computational Linguistics. (cyt. na s. 84, 89 oraz 110)

Bian, J., Liu, Y., Agichtein, E., i Zha, H. (2008). Finding the right facts in the crowd. *Proceeding of the 17th international conference on World Wide Web*, WWW '08. ACM Press. (cyt. na s. 28)

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., i Hellmann, S. (2009). DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165. (cyt. na s. 83)

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., i Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, strony 1247–1250. ACM Press. (cyt. na s. 83)

Boni, M. D. i Manandhar, S. (2003). The Use of Sentence Similarity as a Semantic Relevance Metric for Question Answering. *Proceedings of the AAAI*

Symposium on New Directions in Question Answering. AAAI Press. (cyt. na s. 18)

Bralczyk, J., editor (2005). *100 tysięcy potrzebnych słów*. Wydawnictwo Naukowe PWN. (cyt. na s. 116)

Brill, E., Dumais, S., i Banko, M. (2002). An analysis of the AskMSR question-answering system. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, volume 10, strony 257–264. Association for Computational Linguistics. (cyt. na s. 25 i 59)

Buchholz, S. i Daelemans, W. (2001). SHAPAQA: Shallow Parsing for Question Answering on the World Wide Web. *Proceedings of Euroconference on Recent Advances in Natural Language Processing RANLP2001*. (cyt. na s. 17, 25 oraz 29)

Budanitsky, A. i Hirst, G. (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47. (cyt. na s. 111)

Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R. J., Harabagiu, S., Israel, D., Jacquemin, C., Lin, C.-Y., Maiorano, S., Miller, G., Moldovan, D., Ogden, B., Prager, J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E. M., i Weishedel, R. (2001). Issues , Tasks and Program Structures to Roadmap Research in Question & Answering (Q & A). Technical report, NIST. (cyt. na s. 11)

Ciaramita, M. i Altun, Y. (2006). Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing - EMNLP '06*. Association for Computational Linguistics. (cyt. na s. 85)

Dakka, W. i Cucerzan, S. (2008). Augmenting Wikipedia with Named Entity Tags. *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008)*. Association for Computational Linguistics. (cyt. na s. 84 i 89)

Dang, H. T. (2008). Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. *Proceedings of Text Analysis Conference (TAC 2009)*. NIST. (cyt. na s. 10)

Dang, H. T., Kelly, D., i Lin, J. (2007). Overview of the TREC 2007 Question Answering track. *Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007*. NIST. (cyt. na s. 9 i 108)

Dang, H. T., Lin, J., i Kelly, D. (2006). Overview of the TREC 2006 Question Answering Track. *Proceedings of The Fifteenth Text REtrieval Conference (TREC 2006)*. NIST. (cyt. na s. 9)

Degórski, L. (2012). Towards the Lemmatisation of Polish Nominal Syntactic Groups Using a Shallow Grammar. *Proceedings of the International Joint Conference on Security and Intelligent Information Systems*, volume 7053 of *Lecture Notes in Computer Science*, strony 370–378. Springer-Verlag. (cyt. na s. 39)

Dubisz, S., editor (2006). *Uniwersalny słownik języka polskiego PWN*. Wydawnictwo Naukowe PWN. (cyt. na s. 115)

Duclaye, F., Sitko, J., Filoche, P., i Collin, O. (2002). A Polish Question-Answering System for Business Information. *BIS 2002, 5th International Conference on Business Information Systems, Poznań, Poland, 24-25*

April 2002, strony 209–212. Department of Information Systems, Poznań University of Economics. (cyt. na s. 20)

Duclaye, F., Yvon, F., i Collin, O. (2003). Learning paraphrases to improve a question-answering system. *Proceedings of the 10th Conference of EACL Workshop Natural Language Processing for Question-Answering*. Association for Computational Linguistics. (cyt. na s. 16)

Etzioni, O. (2011). Search needs a shake-up. *Nature*, 476(7358). (cyt. na s. 4)

Fader, A., Soderland, S., i Etzioni, O. (2011). Identifying relations for open information extraction. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, strony 1535–1545. (cyt. na s. 26)

Fader, A., Zettlemoyer, L., Etzioni, O., i Science, C. (2013). Paraphrase-Driven Learning for Open Question Answering. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, strony 1608–1618. Association for Computational Linguistics. (cyt. na s. 26 i 28)

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press. (cyt. na s. 18)

Ferrucci, D. A., Brown, E., Chu-carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., i Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79. (cyt. na s. 10, 20, 32 oraz 75)

Ferrucci, D. A., Nyberg, E., Allan, J., Barker, K., Brown, E., Chu-Carroll, J., Ciccolo, A., Duboue, P., Fan, J., Gondek, D., Hovy, E., Katz, B., Lally, A., Mccord, M., Morarescu, P., Murdock, B., Porter, B., Prager, J., i Heights, Y. (2009). Towards the Open Advancement of Question Answering Systems. Technical report, IBM. (cyt. na s. 11)

Forner, P., Peñas, A., Agirre, E., Alegria, I. n., Forăscu, C., Moreau, N., Osenova, P., Prokopidis, P., Rocha, P., Sacaleanu, B., Sutcliffe, R., i Sang, E. T. K. (2008). Overview of the CLEF 2008 multilingual question answering track. *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access (CLEF'08)*, strony 262–295. Springer-Verlag. (cyt. na s. 10)

Galambos, L. (2001). Lemmatizer for Document Information Retrieval Systems in JAVA. *Proceedings of the 28th Conference on Current Trends in Theory and Practice of Informatics (SOFSEM 2001)*, strony 243–252. Springer-Verlag. (cyt. na s. 35 i 61)

Green, B. F., Wolf, A. K., Chomsky, C., i Laughery, K. (1961). BASEBALL: an automatic question answerer. *Proceedings of western joint IRE-AIEE-ACM '61 computer conference*, strony 219–224. ACM Press. (cyt. na s. 7, 27 oraz 29)

Grishman, R. i Sundheim, B. (1996). Message Understanding Conference - 6: A Brief History. *Proceedings of the 16th conference on Computational linguistics - COLING '96*, volume 1, strony 466–471. Association for Computational Linguistics. (cyt. na s. 44)

Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., i Morarescu, P. (2001). The role of lexico-semantic feedback in open-domain textual question-answering. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*, strony 282–289. Association for Computational Linguistics. (cyt. na s. 17,

18, 31 oraz 47)

Harabagiu, S., Moldovan, D., Paşca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., i Morarescu, P. (2000). FALCON: Boosting Knowledge for Answer Engines. *Proceedings of The Ninth Text REtrieval Conference (TREC 2000)*, strony 479–488. NIST. (cyt. na s. 31)

Hartrumpf, S. (2004). Question Answering using Sentence Parsing and Semantic Network Matching. *Proceedings of the 5th conference on Cross-Language Evaluation Forum: Multilingual Information Access for Text, Speech and Images CLEF'04*, strony 512–521. Springer-Verlag. (cyt. na s. 17)

Hermjakob, U. (2001). Parsing and question classification for question answering. *Proceedings of the workshop on Open-domain question answering (ODQA '01)*, volume 12. Association for Computational Linguistics. (cyt. na s. 27 i 29)

Hovy, E., Gerber, L., Hermjakob, U., Junk, M., i Lin, C.-Y. (2000). Question Answering in Webclopedia. *Proceedings of The Ninth Text REtrieval Conference (TREC 2000)*. NIST. (cyt. na s. 31, 35, 60, 61, 71 oraz 74)

Hovy, E., Gerber, L., Hermjakob, U., Lin, C.-Y., i Ravichandran, D. (2001). Toward Semantics-Based Answer Pinpointing. *Proceedings of the first international conference on Human Language Technology research (HLT)*. Association for Computational Linguistics. (cyt. na s. 16)

Ittycheriah, A. (2007). A statistical approach for open domain question answering. *Advances in Open Domain Question Answering (Text, Speech and Language Technology)*, volume 32 of *Text, Speech and Language Technology*, strony 35–69. Springer-Verlag. (cyt. na s. 19, 46 oraz 50)

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579. (cyt. na s. 27 i 73)

Jijkoun, V. i De Rijke, M. (2004). Answer selection in a multi-stream open domain question answering system. (cyt. na s. 32)

Karzewski, M. (1997). *Jeden z dziesięciu - pytania i odpowiedzi*. Muza SA. (cyt. na s. 95)

Katz, B. (1997). Annotating the World Wide Web Using Natural Language. *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet RIAO 97*. (cyt. na s. 8 i 25)

Katz, B., Lin, J., Loreto, D., Hildebrandt, W., Bilotti, M., Felshin, S., Fernandes, A., Marton, G., i Mora, F. (2003). Integrating Web-based and corpus-based techniques for question answering. *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*. NIST. (cyt. na s. 25, 27, 59, 60, 71 oraz 75)

Kazama, J. i Torisawa, K. (2007). Exploiting Wikipedia as External Knowledge for Named Entity Recognition. *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (2007)*, strony 698–707. Association for Computational Linguistics. (cyt. na s. 84, 87, 89 oraz 108)

Ko, J., Si, L., i Nyberg, E. (2007). A Probabilistic Framework for Answer Selection in Question Answering. *Proceedings of the 2007 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics HLT-NAACL 2007*, strony 524–531. Association for Computational Linguistics. (cyt. na s. 19)

Konopík, M. i Rohlík, O. (2010). Question Answering for Not Yet Semantic Web. *Proceedings of the 13th International Conference on Text, Speech and Dialogue (TSD 2010)*, strony 125–132. Springer-Verlag. (cyt. na s. 23, 25 oraz 75)

Kopeć, M. i Ogrodniczuk, M. (2012). Creating a Coreference Resolution System for Polish. *The eighth international conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA). (cyt. na s. 71)

Lee, C., Wang, J.-H., Kim, H.-J., i Jang, M.-G. (2005). Extracting Template for Knowledge-based Question-Answering Using Conditional Random Fields. *Proceedings of the 28th Annual International ACM SIGIR Workshop on MFIR*, strony 428–434. ACM Press. (cyt. na s. 46)

Lem, S. (1981). *Golem XIV*. Wydawnictwo Literackie. (cyt. na s. 3)

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710. (cyt. na s. 61)

Li, X. i Roth, D. (2002). Learning Question Classifiers. *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, volume 1 of *COLING '02*. Association for Computational Linguistics. (cyt. na s. 29 i 46)

Liaw, A. i Wiener, M. (2002). Classification and Regression by randomForest. *R news*, 2:18–22. (cyt. na s. 55)

Lloret, E., Llorens, H., Moreda, P., Saquete, E., i Palomar, M. (2011). Text summarization contribution to semantic question answering: New approaches for finding answers on the web. *International Journal of Intelligent Systems*, 26(12):1125–1152. (cyt. na s. 26)

Lombarović, T., Šnajder, J., i Bašić, B. D. (2011). Question Classification for a Croatian QA System. *Proceedings of the 14th International Conference on Text, Speech and Dialogue (TSD 2011)*, strony 403–410. Springer-Verlag. (cyt. na s. 24, 45 oraz 46)

Mann, G. S. (2002). Fine-grained proper noun ontologies for question answering. *Proceedings of the 2002 workshop on Building and using semantic networks (SEMANET '02)*, volume 11. Association for Computational Linguistics. (cyt. na s. 83 i 109)

Marcińczuk, M. (2012). *Ekstrakcja informacji o relacjach semantycznych między jednostkami identyfikacyjnymi z dokumentów tekstowych*. rozprawa doktorska, Politechnika Wrocławska. (cyt. na s. 116)

Marcińczuk, M. i Janicki, M. (2012). Optimizing CRF-based Model for Proper Name Recognition in Polish Texts. *Proceedings of CICLing 2012, Part I*, strony 258–269. Springer-Verlag. (cyt. na s. 40 i 67)

Marcińczuk, M., Ptak, M., Radziszewski, A., i Piasecki, M. (2013a). Open dataset for development of Polish Question Answering systems. *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. Adama Mickiewicza. (cyt. na s. 11, 24, 93, 95 oraz 107)

Marcińczuk, M., Radziszewski, A., Piasecki, M., Piasecki, D., i Ptak, M. (2013b). Evaluation of a Baseline Information Retrieval for a Polish Open-domain Question Answering System. *Proceedings of the International*

Conference Recent Advances in Natural Language Processing (RANLP 2013), strony 428–435. Association for Computational Linguistics. (cyt. na s. 15, 23, 29, 74 oraz 107)

Maziarz, M., Piasecki, M., i Szpakowicz, S. (2012). Approaching plWordNet 2.0. *Proceedings of the 6th Global Wordnet Conference*. The Global WordNet Association. (cyt. na s. 48)

Miliaraki, S. i Androutsopoulos, I. (2004). Learning to identify single-snippet answers to definition questions. *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*. Association for Computational Linguistics. (cyt. na s. 16)

Moldovan, D., Clark, C., Harabagiu, S., i Maiorano, S. (2003). COGEX : A Logic Prover for Question Answering. *NAACL '03 Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, number June, strony 87–93. Association for Computational Linguistics. (cyt. na s. 18)

Moldovan, D., Harabagiu, S., Paşca, M., Mihalcea, R., Gîrju, R., Goodrum, R., i Rus, V. (2000). The structure and performance of an open-domain question answering system. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics - ACL '00*, strony 563–570. Association for Computational Linguistics. (cyt. na s. 31, 35, 59 oraz 60)

Moldovan, D. i Novischi, A. (2002). Lexical chains for question answering. *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*. Association for Computational Linguistics. (cyt. na s. 19, 74 oraz 111)

Moldovan, D., Paşca, M., i Surdeanu, M. (2006). Some Advanced Features of LCC's PowerAnswer. *Advances in Open Domain Question Answering (Text, Speech and Language Technology)*, volume 32, strony 3 – 34. Springer-Verlag. (cyt. na s. 18)

Moreda, P., Llorens, H., Saquete, E., i Palomar, M. (2011). Combining semantic information in question answering systems. *Information Processing & Management*, 47(6):870–885. (cyt. na s. 19 i 26)

Morrison, D. R. (1968). PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, 15(4):514–534. (cyt. na s. 91)

Mykowiecka, A. (2007). *Inżynieria lingwistyczna: Komputerowe przetwarzanie tekstów w języku naturalnym*. Wydawnictwo PJWSTK, Warszawa. (cyt. na s. 116)

Na, S.-H., Kang, I.-S., Lee, S.-Y., i Lee, J.-H. (2002). Question Answering Approach Using a WordNet-based Answer Type Taxonomy. *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*. NIST. (cyt. na s. 83)

National Institute of Standards and Technology (2010). Advanced Question Answering for Intelligence (AQUAINT). (cyt. na s. 10)

Navigli, R. i Ponzetto, S. P. (2010). BabelNet: building a very large multilingual semantic network. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, strony 216–225. Association for Computational Linguistics. (cyt. na s. 83)

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46:323–351. (cyt. na s. 95)

Oh, J.-h., Torisawa, K., Hashimoto, C., Sano, M., De Saeger, S., i Ohtake, K. (2013). Why-Question Answering using Intra- and Inter-Sentential Causal Relations. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, strony 1733–1743. Association for Computational Linguistics. (cyt. na s. 26 i 75)

Osenova, P., Simov, A., Simov, K., Tanev, H., i Kouylekov, M. (2004). Bulgarian-english question answering: adaptation of language resources. *Proceedings of the 5th conference on Cross-Language Evaluation Forum: multilingual Information Access for Text, Speech and Images (CLEF'04)*, volume 3491 of *Lecture Notes in Computer Science*, strony 458–469. Springer-Verlag. (cyt. na s. 21)

Palmer, M., Gildea, D., i Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. (cyt. na s. 19)

Papineni, K. (2001). Why inverse document frequency? *Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001 - NAACL '01*. Association for Computational Linguistics. (cyt. na s. 60 i 73)

Paşca, M. (2002). Processing Definition Questions in an Open-Domain Question Answering System. *Proceedings of the 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, strony 73–78. AAAI Press. (cyt. na s. 25 i 27)

Peñas, A., Forner, P., Sutcliffe, R., Rodrigo, A., Forăscu, C., Alegria, I. n., Giampiccolo, D., Moreau, N., i Osenova, P. (2009). Overview of ResPubliQA 2009: question answering evaluation over European legislation. *Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments (CLEF'09)*, strony 174–196. Springer-Verlag. (cyt. na s. 10)

Peñas, A., Hovy, E. H., Forner, P., Rodrigo, A., Sutcliffe, R. F. E., Sporleder, C., Forascu, C., Benajiba, Y., i Osenova, P. (2012). QA4MRE: Question Answering for Machine Reading Evaluation at CLEF 2012. *CLEF 2012 Evaluation Labs and Workshop Online Working Notes*. Springer-Verlag. (cyt. na s. 10)

Peshterliev, S. i Koychev, I. (2011). Semantic Retrieval Approach to Factoid Question Answering for Bulgarian. *Proceedings of the 3rd International Conference on Software, Services and Semantic Technologies (S3T 2011)*, strony 25–32. Springer-Verlag. (cyt. na s. 23 i 61)

Piasecki, M. (2007). Polish Tagger TaKIPI: Rule Based Construction and Optimisation. *Task Quarterly*, 11(1-2):151–167. (cyt. na s. 39)

Piasecki, M., Szpakowicz, S., i Broda, B. (2009). *A Wordnet from the Ground Up*. Oficyna Wydawnicza Politechniki Wroclawskiej. (cyt. na s. 48)

Piechociński, D. (2005). *Automatyczne udzielanie odpowiedzi na pytania zadawane w języku naturalnym*. rozprawa doktorska, Polsko-Japońska Wyższa Szkoła Technik Komputerowych. (cyt. na s. 22)

Piechociński, D. i Mykowiecka, A. (2005). Question answering in Polish using shallow parsing. *Computer Treatment of Slavic and East European Languages: Proceedings of the Third International Seminar, Bratislava, Slovakia*, strony 167–173. VEDA: Vydava- tel'stvo Slovenskej akadéme vied. (cyt. na s. 22)

Ponzetto, S. P. i Strube, M. (2007). Deriving a Large Scale Taxonomy from Wikipedia. *Artificial Intelligence*, 22:1440–1445. (cyt. na s. 84)

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137. (cyt. na s. 61)

Przepiórkowski, A. (2007). Slavonic information extraction and partial parsing. *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing Information Extraction and Enabling Technologies - ACL '07*. Association for Computational Linguistics. (cyt. na s. 16, 20 oraz 82)

Przepiórkowski, A. (2008). *Powierzchniowe przetwarzanie języka polskiego*. Akademicka Oficyna Wydawnicza EXIT. (cyt. na s. 22, 39, 115 oraz 116)

Przepiórkowski, A., Bańko, M., Górski, R. L., i Lewandowska-Tomaszczyk, B. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warszawa. (cyt. na s. 36, 39, 115 oraz 116)

Przepiórkowski, A., Degórski, L., Wójtowicz, B., Spousta, M., Kuboň, V., Simov, K., Osenova, P., i Lemnitzer, L. (2007). Towards the automatic extraction of definitions in Slavic. *Proceedings of the Workshop on BaltoSlavonic Natural Language Processing Information Extraction and Enabling Technologies ACL 07*. (cyt. na s. 27)

Przybyła, P. (2012). Issues of Polish Question Answering. Hryniewicz, O., Mielniczuk, J., Penczek, W., i Waniewski, J., editors, *Proceedings of the first conference 'Information Technologies: Research and their Interdisciplinary Applications' (ITRIA 2012)*, strony 122–139. Institute of Computer Science, Polish Academy of Sciences. (cyt. na s. 7)

Przybyła, P. (2013a). Question Analysis for Polish Question Answering. *51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Student Research Workshop*, strony 96–102, Sofia, Bulgaria. Association for Computational Linguistics. (cyt. na s. 43)

Przybyła, P. (2013b). Question Classification for Polish Question Answering. Kłopotek, M. A., Koronacki, J., Marciniak, M., Mykowiecka, A., i Wierzchoń, S. T., editors, *Proceedings of the 20th International Conference on Language Processing and Intelligent Information Systems (LP&IIS 2013)*, strony 50–56. Springer-Verlag. (cyt. na s. 29 i 43)

Przybyła, P. i Teisseyre, P. (2014). Analysing Utterances in Polish Parliament to Predict Speaker's Background. *Journal of Quantitative Linguistics*, 21(4):350–376. (cyt. na s. 68)

R Core Team (2013). R: A Language and Environment for Statistical Computing. Technical report, R Foundation for Statistical Computing, Vienna, Austria. (cyt. na s. 55)

Ravichandran, D. i Hovy, E. (2002). Learning surface text patterns for a Question Answering system. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, strony 41–47. Association for Computational Linguistics. (cyt. na s. 16, 23 oraz 25)

Richman, A. E. i Schone, P. (2008). Mining Wiki Resources for Multilingual Named Entity Recognition. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*. Association for Computational Linguistics. (cyt. na s. 84)

Ruiz-Casado, M., Alfonseca, E., i Castells, P. (2005). Automatic Assignment of Wikipedia Encyclopedic Entries to WordNet Synsets. *Advances in Web Intelligence*, 3528:380–386. (cyt. na s. 89)

Sasaki, Y., Lin, C.-j., Chen, K.-h., i Chen, H.-h. (2007). Overview of the

NTCIR-6 Cross-Lingual Question Answering (CLQA) Task. *Proceedings of NTCIR-6 Workshop Meeting*. (cyt. na s. 10)

Savary, A., Manicki, L., i Baron, M. (2013). Populating a multilingual ontology of proper names from open sources. *Journal of Language Modelling*, 1(2):189–225. (cyt. na s. 83)

Savary, A. i Waszczuk, J. (2010). Towards the Annotation of Named Entities in the National Corpus of Polish. *Procedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010*, strony 3622–3629. European Language Resources Association (ELRA). (cyt. na s. 40)

Savary, A. i Waszczuk, J. (2012). Narzędzia do anotacji jednostek nazewniczych. *Narodowy Korpus Języka Polskiego [Eng.: National Corpus of Polish]*, strony 225–252. Wydawnictwo Naukowe PWN. (cyt. na s. 67)

Shapiro, S. C. (1992). *Encyclopedia of Artificial Intelligence*. John Wiley & Sons. (cyt. na s. 12)

Simmons, R. F. (1970). Natural Language Question-Answering Systems: 1969. *Communications of the ACM*, 13(1):15–30. (cyt. na s. 8)

Simov, K. i Osenova, P. (2005). BulQA: Bulgarian–bulgarian question answering at CLEF 2005. *Proceedings of the 6th international conference on Cross-Language Evalution Forum: accessing Multilingual Information Repositories (CLEF'05)*, volume 4022 of *Lecture Notes in Computer Science*, strony 517–526. Springer-Verlag. (cyt. na s. 21, 35 oraz 68)

Solovyev, A. (2013). Dependency-Based Algorithms for Answer Validation Task in Russian Question Answering. *Proceedings of the 25th International Conference on Language Processing and Knowledge in the Web (GSCL 2013)*, strony 199–212. Springer-Verlag. (cyt. na s. 24)

Soubbotin, M. M. i Soubbotin, S. M. (2002). Use of patterns for detection of answer strings: A systematic approach. *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*, volume 11. NIST. (cyt. na s. 16)

Stępień, A. B. (2001). Byt: jego rodzaje, struktura, uwarunkowania. *Wstęp do filozofii*. Towarzystwo Naukowe Katolickiego Uniwersytetu Lubelskiego, Lublin. (cyt. na s. 115)

Suchanek, F. M., Kasneci, G., i Weikum, G. (2007). Yago : a large ontology from Wikipedia and WordNet. *Proceedings of the 16th international conference on World Wide Web - WWW '07*, strony 697–706. ACM Press. (cyt. na s. 83)

Tanev, H. (2004). Socrates - a Question Answering prototype for Bulgarian. *Recent Advances in Natural Language Processing: Selected Papers from RANLP 2003*, strony 377–386. John Benjamins. (cyt. na s. 20, 25 oraz 75)

Toral, A. i Muñoz, R. (2006). A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. (cyt. na s. 83, 89 oraz 90)

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236):433–460. (cyt. na s. 12)

Turney, P. D. i Pantel, P. (2010). From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188. (cyt. na s. 112)

Čeh, I. i Ojsteršek, M. (2009). Developing a question answering system for the

slovene language. *WSEAS Transactions on Information Science and Applications*, 6(9):1533–1543. (cyt. na s. 21, 28 oraz 60)

Vetulani, Z. (1988). PROLOG Implementation of an Access in Polish to a Data Base. *Studia z automatyki, XII*, strony 5–23. PWN. (cyt. na s. 20 i 27)

Vetulani, Z. (2002). Automatyczna interpretacja pytań i udzielanie odpowiedzi jako technologia multimedialna. *Multimedialne i sieciowe systemy informacyjne*, strony 11–22. Oficyna Wydawnicza Politechniki Wrocławskiej. (cyt. na s. 7)

Vetulani, Z. (2004). *Komunikacja człowieka z maszyną. Komputerowe modelowanie kompetencji językowej*. Akademicka Oficyna Wydawnicza EXIT. (cyt. na s. 7)

Vetulani, Z. i Kochanowski, B. (2014). "PolNet - Polish WordNet" project: PolNet 2.0 - a short description of the release. *Proceedings of the Seventh Global Wordnet Conference*, strony 400–404. Association for Computational Linguistics. (cyt. na s. 48)

Vetulani, Z., Marciniak, J., Vetulani, G., Dąbrowski, A., Kubis, M., Osiński, J., Walkowska, J., Kubacki, P., i Witalewski, K. (2010). *Zasoby językowe i technologia przetwarzania tekstu POLINT-112-SMS jako przykład aplikacji z zakresu bezpieczeństwa publicznego*. Wydawnictwo Naukowe UAM. (cyt. na s. 20)

Voorhees, E. M. (1999). The TREC-8 Question Answering Track Report. *Proceedings of The Eight Text REtrieval Conference (TREC 2000)*, volume 7. National Institute of Standards and Technology, NIST. (cyt. na s. 9)

Voorhees, E. M. (2000). Overview of the TREC-9 Question Answering Track. *Proceedings of The Ninth Text REtrieval Conference (TREC 2000)*. NIST. (cyt. na s. 9)

Voorhees, E. M. (2001). Overview of the TREC 2001 Question Answering Track. *Proceedings of the Tenth Text Retrieval Conference (TREC 2001)*. NIST. (cyt. na s. 9)

Voorhees, E. M. (2002). Overview of the TREC 2002 Question Answering Track. *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*. NIST. (cyt. na s. 9)

Voorhees, E. M. (2003). Overview of the TREC 2003 Question Answering Track. *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, volume 142. NIST. (cyt. na s. 9)

Voorhees, E. M. i Dang, H. T. (2005). Overview of the TREC 2005 Question Answering Track. *Proceedings of The Fourteenth Text REtrieval Conference (TREC 2005)*. NIST. (cyt. na s. 9)

Walas, M. (2012). How to answer yes/no spatial questions using qualitative reasoning? *13th International Conference on Computational Linguistics and Intelligent Text Processing*, strony 330–341. Springer-Verlag. (cyt. na s. 22 i 28)

Walas, M. (2014). *Wnioskowanie czasowo-przestrzenne w systemie Question Answering*. rozprawa doktorska, Uniwersytet im. Adama Mickiewicza w Poznaniu. (cyt. na s. 22)

Walas, M. i Jassem, K. (2010). Named entity recognition in a Polish question answering system. *Intelligent Information Systems*, strony 181–191. Publishing House of University of Podlasie. (cyt. na s. 22 i 97)

Walas, M. i Jassem, K. (2011). Spatial reasoning and disambiguation in the

process of knowledge acquisition. *Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, strony 420–424. Fundacja Uniwersytetu im. Adama Mickiewicza. (cyt. na s. 22 i 28)

Weiss, D. (2005). A Survey of Freely Available Polish Stemmers and Evaluation of Their Applicability in Information Retrieval. *2nd Language and Technology Conference, Poznań, Poland*, strony 216–221. Fundacja Uniwersytetu im. Adama Mickiewicza. (cyt. na s. 65)

Woliński, M., Miłkowski, M., Ogrodniczuk, M., Przepiórkowski, A., i Szałkiewicz, L. (2012). PoliMorf: a (not so) new open morphological dictionary for Polish. *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, strony 860–864. European Language Resources Association (ELRA). (cyt. na s. 37)

Wu, W., Li, H., Wang, H., i Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. *Proceedings of the 2012 ACM SIGMOD . . .* , strony 481–492. (cyt. na s. 20)

Yandex.ru (2010). Report on Yandex.ru queries. Technical report. (cyt. na s. 4)

Yih, W.-t., Chang, M.-w., Meek, C., i Pastusiak, A. (2013). Question Answering Using Enhanced Lexical Semantic Models. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, strony 1744–1753. Association for Computational Linguistics. (cyt. na s. 20, 71 oraz 74)

Spis rysunków

Spis tabel

Spis algorytmów