Akademia Górniczo-Hutnicza
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Rozprawa doktorska

Semi-automatyczna kontekstowa analiza i korekta tekstów z wykorzystaniem specjalistycznych grafów lingwistycznych

mgr inż. Marcin A. Gadamer

Promotor:
dr hab. Adrian Horzyk, prof. AGH

Kraków 2019

Pragnę złożyć najserdeczniejsze podziękowania mojemu promotorowi Panu dr hab. Adrianowi Horzykowi, Profesorowi AGH za okazaną pomoc, zrozumienie, życzliwość i cierpliwość podczas powstawania tej pracy. Szczególnie dziękuję za cenne uwagi i spostrzeżenia, dzięki którym rozprawa ta mogła powstać.

Pragnę szczególnie podziękować żonie Magdalenie za jej miłość, wsparcie okazywane każdego dnia, inspirację, wiarę, motywację i wyrozumiałość podczas prowadzonych badań i powstawania rozprawy doktorskiej.

Chcę również podziękować swojej rodzinie, a w szczególności Rodzicom za ich inspirację do napisania tej pracy oraz nigdy niegasnącą wiarę we mnie.

Streszczenie

Dostęp do informacji jest kluczowy dla dzisiejszego świata. To dzięki dostępowi do nich, możliwy jest nasz rozwój. Największym źródłem informacji jest Internet. Umożliwia on łatwy, szybki i tani dostęp do zasobów wiedzy z każdej dziedziny życia. Możliwość oceny danych, a następnie powiązania ich w celu późniejszego przetwarzania stanowi jednak problem. W związku z ogromną ilością nowych treści nieuniknionym jest pojawianie się w nich błędów. Tematyka sposobu korekty tekstu jest szeroko znana. Niestety nadal brakuje metod potrafiących dokonać jego automatycznej poprawy, biorąc pod uwagę jego semantykę.

Niniejsza rozprawa podejmuje wskazany powyżej problem przedstawiając możliwości wykorzystania nowego rozwiązania. Celem nadrzędnym pracy było zbudowanie innowacyjnego mechanizmu służącego do efektywnego, asocjacyjnego gromadzenia, kompresowania, a następnie przetwarzania tekstu, tak aby stworzyć algorytmy do jego późniejszej, semi-automatycznej korekty. Zaprojektowano specjalistyczny graf lingwistyczny, do którego budowy wykorzystano teksty pochodzące z różnych źródeł. Następnie jego działanie zweryfikowano poprzez eksperymenty pokazujące interesujące właściwości zaproponowanego rozwiązania. Należą do nich:

- korekta tekstu za pomocą zaimplementowanej metody pobudzeń asocjacyjnych;
- porównanie korekty tekstu z istniejącymi aplikacjami dla języka angielskiego i języka polskiego;
- porównanie korekty tekstu dla dzieł literackich.

Algorytmy oparte o zaproponowaną strukturę grafu zostały zatem przetestowane, a otrzymane rezultaty okazały się zadowalające. W badaniach porównano i potwierdzono skuteczność stworzonych metod do korekty tekstu wprowadzonego z różnego rodzaju błędami, jak również w kontekstowym zapamiętywaniu nauczonych wypowiedzi. Algorytmy były w stanie znaleźć, przetworzyć i powiązać relacje pomiędzy kolejnymi słowami w taki sposób, aby „zrozumieć" czytane zdanie w danym kontekście słownym.

W rezultacie algorytmy kontekstowej korekty tekstu wraz z zaproponowanym modelem zapisu zdań stały się warstwą logiki dla opracowanej aplikacji internetowej, w której użytkownik może wprowadzać tekst z błędami, a program, wykorzystując opracowane algorytmy, jest w stanie dokonać jego semi-automatycznej korekty. Specjalistyczny graf lingwistyczny może zatem w przyszłości być wykorzystywany dla różnych języków naturalnych stanowiąc, podstawę do tworzenia nowych rozwiązań korekty tekstu.

Abstract

Access to information is crucial for today's World. Thanks to it our development is possible. The biggest source of information is the Internet. It allows easy, fast, and quick access to the knowledge resources in every area of life. However, the ability to evaluate data and then link it for later processing is a problem. Due to the huge amount of new content, errors are inevitable. The subject of text correction is widely known. Unfortunately, there is still a lack of methods that can automatically improve it, taking into account its semantics.

This dissertation addresses the problem indicated above, presenting the possibilities of using the new solution. The main goal of the work was to build an innovative mechanism for effective associative collection, compression, and then processing of text to implement algorithms for its later semi-automatic correction. A specialized linguistic graph was designed, which was built using texts from various sources. Then its operation was verified by experiments showing interesting properties of the proposed solution. Belong to them:

- text correction using the implemented method of associative stimulation;
- comparison of text correction with existing applications for English and Polish;
- comparison of proofreading for literary works.

The algorithms based on the proposed graph structure have therefore been tested and the results obtained proved to be satisfactory. The studies compared and confirmed the effectiveness of the methods created to correct the text introduced with various types of errors, as well as in the context of memorizing the learned statements. The algorithms were able to find, process, and link relationships between words in such a way as to „understand" the sentence in a given context.

The research aimed at using texts available from various sources that were used to construct specialized linguistic graphs. Next, the proposed associative connections, together with the constructed structure, were used to implement the mechanisms of semi-automatic text correction.

As the result, the algorithms of contextual text correction together with the proposed model of storing sentences have become a layer of logic for the developed web application in which the user can enter text with errors, and the program, using the developed algorithms, can make its semi-automatic correction. Specialized linguistic graphs can be used for various natural languages, providing the basis for creating new text correction solutions.

Spis treści

Rozdział 1....6
Wstęp....6
Cel pracy....8
Zawartość rozprawy....8
Internet jako miejsce przechowywania danych....11
Sposoby magazynowania wiedzy....13
Rozdział 3....18
Przetwarzanie języka naturalnego....18
Etapy analizy tekstów....24
Narzędzia wykorzystywane podczas przetwarzania tekstu....28
Modele języka naturalnego....34
Metody wykorzystujące sieci neuronowe....44
Metody automatycznej predykcji sylab....51
Analiza sentymentu zdania....52
Oznaczanie części mowy....53
Rozpoznawanie nazw własnych....56
Rozdział 4....59
Graf jako struktura....59
Graf jako sposób zapisu zgromadzonych danych....61
Graf Przyzwyczajeń Lingwistycznych....66
Błędy językowe....81
Semi-automatyczne metody analizy i korekty tekstów....87
Metody statyczne....88
Zmodyfikowana metoda odległości edycyjnej....90
Zmodyfikowana metoda n-gramów....91
Metoda wykorzystująca pobudzenia asocjacyjne....93
Rozdział 5....96
Rodzaje opracowanych metod do korekty tekstu....96
Korekta tekstu za pomocą zaimplementowanej metody pobudzeń asocjacyjnych....99
Porównanie korekty tekstu z istniejącymi aplikacjami dla języka angielskiego....106
Porównanie korekty tekstu z istniejącymi aplikacjami dla języka polskiego....118
Porównanie korekty tekstu dla dzieł literackich....123
Rozdział 6....133
Podsumowanie....133
Otwarte spostrzeżenia badawcze....136
Bibliografia....138

Rozdział 1

Wstęp

W dzisiejszym świecie można łatwo zauważyć bardzo szybki postęp technologiczny, jak również związaną z nim ewolucję w wielu dziedzinach. Takiemu przekształceniu uległ m. in. sposób komunikacji międzyludzkiej oraz związana z nim przemiana języka. Początkowo ludzie komunikowali się za pomocą prostych słów. Wypowiedź docierała jedynie do osób znajdujących się w pobliżu mówiącego. Na odległość komunikacja była możliwa wyłącznie dzięki gestom oraz obrazom. Sposób komunikacji z biegiem czasu uległ jednak zmianie. Dzisiaj tradycyjne środki komunikacji odeszły na dalszy plan, oddając miejsce nowoczesnej technologii [1], [2].

Bardzo ważnym elementem w dzisiejszym świecie jest informacja oraz sposób jej przetwarzania. Dzięki informacji możliwy jest ciągły rozwój ludzkości. Niewątpliwie największym źródłem informacji jest Internet. W przeciągu kilku ostatnich lat powstała ogromna liczba nowych stron internetowych, forów, blogów, na których ludzie mogą wymieniać się informacjami. Przyczyniło się do tego powstanie, na początku XXI wieku, wielojęzycznego projektu internetowej encyklopedii działającej w oparciu o zasadę otwartej treści – Wikipedii – gdzie każdy człowiek może zamieszczać informacje na dowolny temat [3].

Tak więc problemem dzisiejszego świata nie jest już dostęp do informacji, ale sposób jej przetwarzania. Według Głównego Urzędu Statystycznego w 2018 r. odsetek osób korzystających z Internetu w Polsce wyniósł 77,5% [4]. Obecnie kwestią o wiele ważniejszą jest jakość treści oraz możliwość powiązania tej samej informacji występującej na kilku lub nawet kilkudziesięciu różnych stronach internetowych. Nie ma jednego sposobu, aby dane te ze sobą połączyć i skojarzyć. Zazwyczaj do tego celu wykorzystywane jest pasywne indeksowanie oraz ręczna praca. Osoba, która przegląda kolejne strony internetowe, może z łatwością połączyć podobne dane na wielu stronach dzięki swojej inteligencji i wiedzy. Wiele podobnych informacji w Internecie jest wyszukiwanych na zasadzie konkurencyjnej treści i podobieństwa, ale rzadko kiedy uzupełniają się one wzajemnie [5].

W dzisiejszych czasach powstaje bardzo duża liczba elektronicznych dokumentów, w których często występują błędy językowe. Istnieje więc potrzeba opracowania bardziej inteligentnej i kontekstowej korekty dla tekstu, który został wprowadzony z różnego rodzaju błędami. Na

pojawianie się tych błędów ma wpływ wiele różnorakich czynników [6]. Najczęściej błędy powstają z winy użytkownika, a czasami generuje je sam program komputerowy. Do najważniejszych błędów użytkownika można zaliczyć m. in.:

- brak znajomości zasad konstrukcji poprawnych zdań,
- brak pełnego przekazu treści (skróty myślowe, zdrobnienia),
- brak staranności przy wpisywaniu tekstu,
- pośpiech użytkownika podczas wprowadzania tekstu.

Do błędów, braków i niedoskonałości aplikacji komputerowych natomiast zalicza się:

- brak specjalistycznych algorytmów do obsługi języka polskiego,
- brak badania kontekstu wprowadzanego tekstu (kontekst wyrazów i zdań),
- brak badania wprowadzenia przypadkowo poprawnych/błędnych wyrazów,
- automatyczna korekta, która nie zawsze działa poprawnie w różnych procesorach tekstów.

Tematyka sposobu korekty tekstu jest szeroko znana, lecz nadal brakuje metod potrafiących dokonać automatycznej jego poprawy, biorąc pod uwagę semantykę. Obecnie stosowane są do tego celu różnorakie słowniki, wyznaczane dla słów odległości edycyjne oraz różne algorytmy, który skuteczność jest jednak nadal bardzo ograniczona..

Istnieje wiele metod oraz mechanizmów gromadzenia i przechowywania informacji. Modele językowe oparte na n-gramach słów (także wzbogacone o tagi POS – ang. *Part Of Speech tagging*) są dobrze znane od ponad 20 lat i używane w celu rozpoznawania mowy. Pośród metod służących budowaniu modeli do analizy tekstu można wymienić modele wykorzystujące gramatyki formalne i transakcyjne, systemy regułowe oraz systemy oparte na analizie korpusów tekstu. Niemniej jednak w znacznej części tych modeli brakuje analizy kontekstu wprowadzanych słów. Może się zdarzyć, że wprowadzone słowa są poprawne gramatycznie, lecz w zadanym kontekście okazują się nieprawidłowe [7]–[11].

Cel pracy

Celem pracy było uzyskanie wiedzy na temat możliwości przeprowadzenia efektywnej, semi-automatycznej, kontekstowej korekty różnych tekstów. Cel ten osiągnięto poprzez opracowanie innowacyjnego mechanizmu służącego do asocjacyjnego gromadzenia, kompresowania, a następnie przetwarzania tekstu (zdań) oraz zbudowanie algorytmów do korekty teksów wprowadzonych z różnego rodzaju błędami.

Badania miały na celu wykorzystać dostępne z różnych źródeł teksty dla wykonania algorytmów, które będą w stanie znaleźć, przetworzyć i powiązać relacje pomiędzy kolejnymi słowami w taki sposób, aby „zrozumieć" czytane zdanie w danym kontekście słownym. Następnie opracowane algorytmy wraz z zaproponowaną strukturą, służącą zapisowi dla tak „zrozumianego" kontekstowo tekstu, miały służyć automatycznej korekcie wprowadzanego tekstu.

Algorytm kontekstowej korekty tekstu wraz z zaproponowanym modelem zapisu zdań będą warstwą logiki dla aplikacji internetowej, w której użytkownik będzie mógł wprowadzać tekst z błędami, a program wykorzystując opracowane algorytmy, będzie mógł korygować tekst. Korekta ta odbywać się będzie w sposób automatyczny lub semi-automatyczny. Semi-automatyczna korekta tekstu oznacza, że ostateczną decyzję co do wyboru najkorzystniejszego rozwiązania spośród zaproponowanych przez algorytm podejmie autor tekstu. Wynika to z faktu, że algorytm korygujący tekst nie zna intencji piszącego i jedynie autor jest w stanie zdecydować, która z zaproponowanych korekt koresponduje z jego zamiarami. W pełni automatyczna korekta tekstu nie zawsze jest więc możliwa.

W rozprawie sformułowano następującą tezę:

Możliwe jest zbudowanie specjalistycznych grafów lingwistycznych na podstawie korpusów tekstów oraz algorytmów ich efektywnej analizy, pozwalających na przeprowadzenie poprawnej semi-automatycznej kontekstowej korekty różnych tekstów opartej na wiedzy zebranej w tych grafach, wykorzystując w trakcie ich tworzenia liczbę wystąpień poszczególnych słów w kontekście korygowanego tekstu.

Zawartość rozprawy

Rozprawa została podzielona na sześć rozdziałów.

W rozdziale pierwszym zawarto wstęp do niniejszej pracy. Zamieszczono w nim uzasadnienie ważności tematyki, jaką porusza, a także przedstawiono cel pracy.

W rozdziale drugim ujęto opis istniejących metod pozyskiwania danych. W tym fragmencie wymieniono sposoby utrwalania danych, ponadto scharakteryzowano najpopularniejsze rodzaje bazy danych, zarówno relacyjne, jak i nierelacyjne, które wykorzystywane są we współczesnych rozwiązaniach informatycznych.

Rozdział trzeci przedstawia wiedzę dotyczącą przetwarzania języka naturalnego w oparciu o zebrane źródła bibliograficzne. Wyróżniono w nim główne kategorie podejść do przetwarzania języka, jak również przedstawiono trudności, jakie wpływają na sposób działania algorytmów. W rozdziale tym opisane zostało prawo Zipfa, służące do określenia relacji pomiędzy częstotliwością występowania słowa a jego rangą. W dalszej części przedstawiono etapy, na jakie zazwyczaj podzielony jest proces analizy tekstu. W tym fragmencie wyróżniono narzędzia, które wykorzystywane są podczas przetwarzania tekstu. Ujęto i opisano, czym są wyrażenia regularne, segmentacja zdania, drzewa decyzyjne oraz minimalna odległość edycyjna. Opisano wykorzystywane modele języka naturalnego. Przedstawiono zarówno probabilistyczne modele językowe, jak również modele wektorowe. Scharakteryzowane zostały dodatkowo metody, które w swoim działaniu wykorzystują sieci neuronowe. W dalszej części opisano, czym jest analiza sentymentu zdania, na czym polega oznaczanie części mowy, a także rozpoznawanie nazw własnych.

W rozdziale czwartym opisano badania własne oparte na wykorzystaniu grafu jako struktury danych, która dzięki swoim właściwościom, może służyć do zapisu zgromadzonych tekstów. W tej części zostało wprowadzone pojęcie Grafu Przyzwyczajeń Lingwistycznych, czyli modelu zapisu kontekstu słownego. W rozdziale tym opisano opracowany przez autora, innowacyjny sposób przechowywania słów, jak również scharakteryzowano wykorzystane połączenia, występujące między nimi tak, aby móc odtworzyć kontekst słowny wprowadzonego zdania. Następnie przedstawiono, czym są błędy językowe oraz opisano ich typy. W kolejnym fragmencie przedstawiono wykonane na bazie Grafu Przyzwyczajeń Lingwistycznych semi-automatyczne metody analizy i korekty tekstu.

Rozdział piąty zawiera w głównej mierze porównanie metod wykorzystywanych do korekty tekstu. W tej części opisano szczegółowo proces korekty tekstu z wykorzystaniem opracowanych w tej pracy metod, a następnie przeprowadzono porównanie otrzymanych wyników z innymi aplikacjami, które służą korekcie tekstu. W rozdziale tym przedstawiono przykłady zarówno dla języka polskiego, jak i angielskiego, wykazując, iż zastosowana struktura grafowa wraz z wykonanymi metodami może służyć do korekty tekstu w różnych językach.

Szósty rozdział zawiera podsumowanie wyników pracy. Wskazano w nim celowość podjętych badań, jak również przedstawiono otwarte spostrzeżenia badawcze. Podjęta w pracy teza została potwierdzona: na podstawie korpusów tekstów oraz algorytmów ich analizy, możliwe jest zbudowanie specjalistycznych grafów lingwistycznych pozwalających na przeprowadzenie poprawnej semi-automatycznej kontekstowej korekty zdań, opartej na statystyce wystąpień tekstu w zadanym kontekście. W wyniku przeprowadzonych prac uzyskano efektywny model grafowy służący do zapisu wyrazów oraz znaków interpunkcyjnych występujących w badanym tekście. Opracowano metody umożliwiające pozyskanie tekstów z kilku źródeł w celu zbudowania podanego grafu. Następnie zaprojektowano i wdrożono niezależne metody służące do analizy i kontekstowej korekty tekstu. Na podstawie powyższych osiągnięć wykonano aplikację webową (serwis internetowy), która umożliwia skorzystanie z zaimplementowanych algorytmów. Badania wykazały, że opracowany graf wraz z zaproponowanymi mechanizmami korekty tekstu jest dokładniejszy w porównaniu z istniejącymi rozwiązaniami, o których mowa w rozdziale piątym. Graf Przyzwyczajeń Lingwistycznych potrafi w satysfakcjonującym stopniu odtworzyć kontekst zdań, które zostały wcześniej w nim zapisane. Osiągnięte rezultaty pozwalają na uznanie Grafu Przyzwyczajeń Lingwistycznych jako innowacyjnego rozwiązania, na bazie którego mogą zostać opracowane kolejne niezależne metody korekty tekstu w przyszłości.

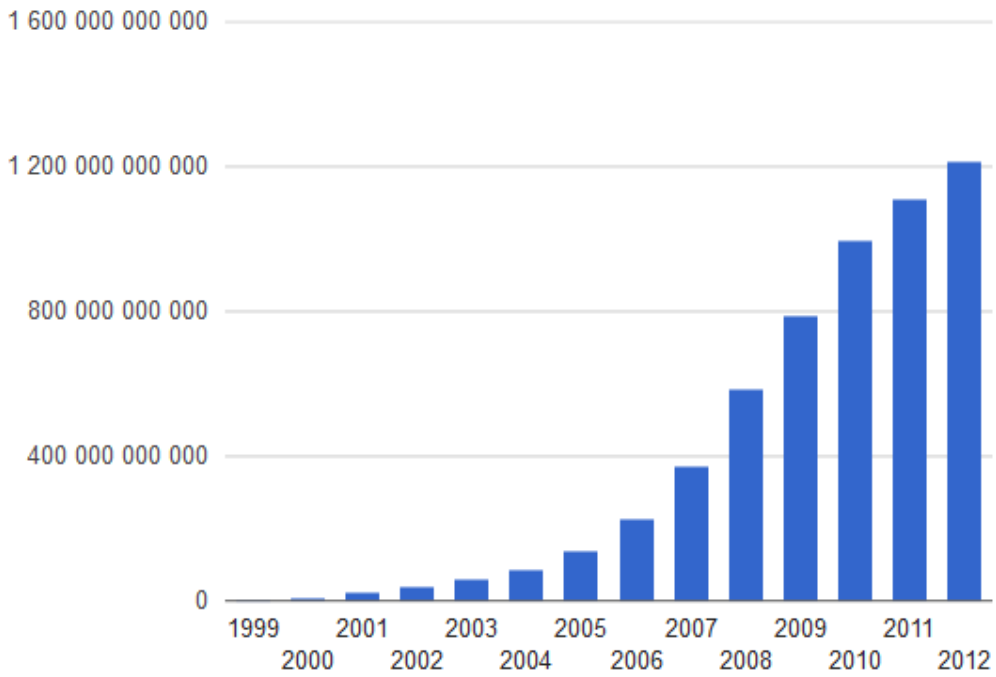
Rozdział 2

Internet jako miejsce przechowywania danych

Żyjemy w czasach swobodnego dostępu do Internetu. Ma on zasięg globalny, łączy wszystkie komputery podłączone do lokalnych i rozległych sieci komputerowych. To uniwersalny, ogólnoświatowy system wymiany informacji i komunikowania się. Jest złożonym zbiorem ogromnej ilości zasobów. Internet został uznany za przełomowe zjawisko w komunikacji społecznej. Jest też źródłem przewagi dla tych, którzy potrafią się nim posługiwać [3].

Zgodnie z danymi pochodzącymi z najbardziej popularnej wyszukiwarki na świecie – Google – w roku 2012 w ciągu sekundy przetwarzanych było ponad 40 000 zapytań, które to przekładają się na ponad 3,5 biliona zapytań dziennie i 1,2 tryliona zapytań rocznie [12]. Rysunek nr 1 pokazuje trend dla ilości rocznych wyszukiwań od początku jej powstania, właśnie do roku 2012.

Wyszukiwarka Google w całym 2016 roku obsłużyła już ponad 2 tryliony zapytań, co z kolei przekłada się na ponad 228 milionów wyszukań na godzinę i 60 tysięcy wyszukiwań na sekundę. Jak widać, z roku na rok liczba wyszukiwań wzrasta i nadal będzie wzrastać.



Rysunek 1: Przyrost ilości rocznych wyszukiwań dla wyszukiwarki google.com [13]

W sierpniu 2012 Amit Singhal, Senior Vice-President firmy Google, poinformował, że algorytmy znalazły już ponad 30 trylionów unikalnych adresów URL w sieci oraz dziennie przeszukują i indeksują dane z ponad 20 bilionów stron [13]. Dziś zindeksowanych stron internetowych przez Google jest już ponad 60 trylionów [14].

Należy nadmienić, że wyszukiwarka Google to niejedyna wyszukiwarka dostępna w Internecie. Zgodnie z danymi, jakie podaje portal www.gs.statcounter.com, w kwietniu 2019 udział w rynku najpopularniejszych wyszukiwarek w Polsce wyglądał następująco:

- Google – 98,62%,
- Yahoo – 0,59%,
- Bing – 0,52%,
- DuckDuckGo – 0,1%,
- Interia Katalog – 0,05%,
- Yandex RU- 0,03%.

Internet jest bardzo bogaty we wszelkiego rodzaju dane. Oznacza to, że w dzisiejszych czasach można w nim znaleźć informacje na prawie każdy temat. Popularnym stało się powiedzenie, że jeśli coś nie występuje w Internecie, to najprawdopodobniej w ogóle nie istnieje [15].

Wyszukiwarki internetowe stały się tak powszechne, ponieważ pomagają one rozwiązać jeden z najbardziej uciążliwych problemów - brak powiązania danych zawartych na stronach internetowych. Nie jest możliwe automatyczne połączenie danych z jednych stron internetowych z danymi na ten sam temat występujących na innych stronach. Jak możemy przeczytać na stronie wyszukiwarki Google: „*Internet jest jak stale powiększająca się biblioteka publiczna z miliardami zasobów bez centralnego systemu danych. Google po prostu gromadzi strony, po czym tworzy indeks, który pozwala na łatwe znajdowanie danych. Podobnie jak skorowidz na końcu książki indeks Google zawiera informacje o słowach i ich lokalizacji. Gdy czegoś szukasz, na najbardziej podstawowym poziomie nasze algorytmy wyszukują Twoje zapytanie w indeksie, by znaleźć odpowiednie strony […] Systemy indeksowania Google zapamiętują wiele różnych aspektów stron, np. datę publikacji, zamieszczone zdjęcia lub filmy itp. tworząc Graf wiedzy*” [16].

W związku z powstawaniem coraz większej ilości tekstów w formie elektronicznej (strony internetowe, blogi, e-maile, e-booki, dokumenty (pliki .txt, .doc, .docx, .pdf i inne), itp.) obserwuje się coraz większą liczbę błędów językowych, jakie są w nich zawarte. Powody ich powstania, jak

wskazano wyżej, są różne. Wynikają one z braku znajomości zasad poprawnej konstrukcji zdań, zastosowanych skrótów myślowych, zdrobnień lub wynikają z manualnych błędów osób je tworzących, takich jak brak staranności przy wpisywaniu tekstu lub z pośpiechu użytkownika.

Istnieje więc potrzeba opracowania bardziej inteligentnej i kontekstowej korekty tekstu, który został już wprowadzony z różnego rodzaju błędami.

Sposoby magazynowania wiedzy

W związku z coraz większą ilością powstających danych pojawiła się konieczność powstania sposobu ich magazynowania. Z jednej strony dane te należy tak przechowywać, aby w łatwy sposób można było je zapisywać, modyfikować, odczytywać i usuwać. Z drugiej strony wskazane jest opracowanie takiego systemu zapisu danych, aby mogły być one wzajemnie ze sobą powiązane. Na początku dane można było magazynować w zwykłych plikach płaskich (takich jak np. pliki .txt, .doc). Niewątpliwą zaletą stosowania takich plików jest łatwość zapisu, odczytu i edycji. Niemniej istotnym ograniczeniem dla powstających programów jest brak możliwości wprowadzenia w łatwy sposób relacji występujących pomiędzy nimi, jak również szybkość wyszukiwania informacji w tych plikach. W związku z tym odpowiedzią na te problemy stały się relacyjne bazy danych [17].

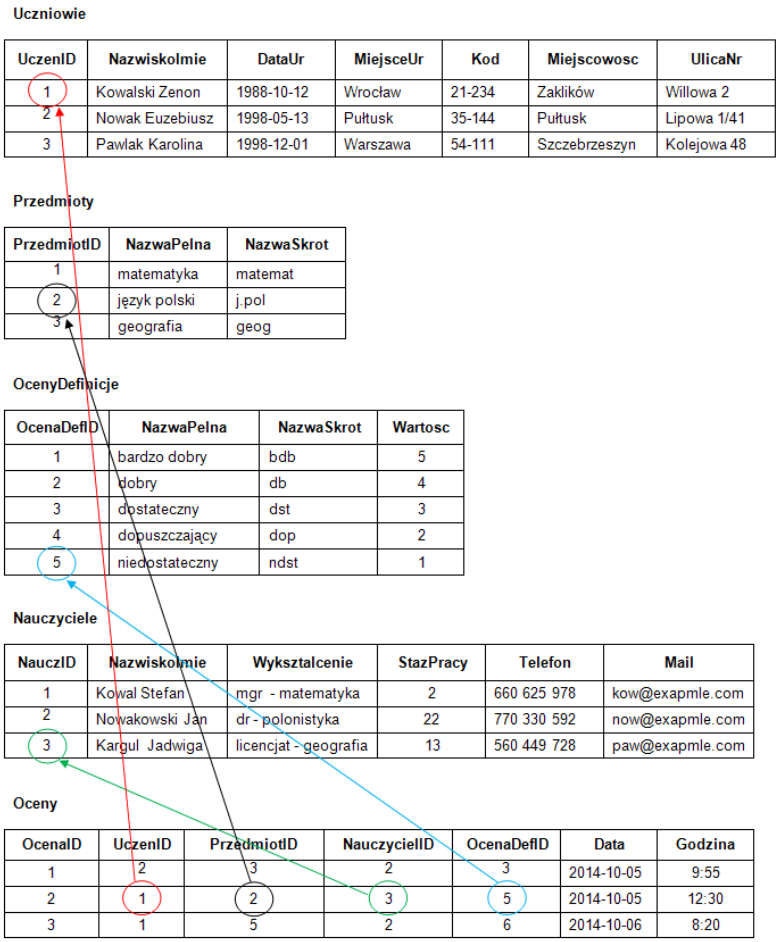
Najwcześniejsze znane użycie terminu baza danych miało miejsce w listopadzie 1963, kiedy odbyło się sympozjum pod nazwą „Development and Management of a Computer-centered Data Base" [18], sponsorowane przez System Development Corporation. Termin ten stał się powszechnie używany w Europie we wczesnych latach siedemdziesiątych XX wieku.

W latach sześćdziesiątych XX wieku został opracowany pierwszy system zarządzania bazami danych. Wcześniej przetwarzanie danych było oparte na kartach dziurkowanych i taśmach magnetycznych [19].

Relacyjny model danych bazuje na matematycznej teorii mnogości, w szczególności na pojęciu relacji. Relacją wg definicji jest dowolny podzbiór iloczynu kartezjańskiego skończonej liczby zbiorów. Na modelu relacyjnym oparta jest relacyjna baza danych (ang. *Relational Database*), czyli taka baza, w której dane są przedstawione w postaci relacyjnej. W modelu tym dane dzielimy pomiędzy tabele, z której każda reprezentuje inną domenę. Następnie określa się relacje istniejące pomiędzy takimi tabelami, które łączą dane w logiczną całość, zrozumiałą dla osób korzystających z bazy [20]. Przykładem relacyjnej bazy danych może być elektroniczny

dziennik z ocenami, którego schemat został przedstawiony na rysunku 2. [21]. W takim systemie możemy wyróżnić następujące tabele:

- Uczniowie – tabela agregująca informację o uczniach (każdy wiersz tej tabeli to inny uczeń wraz ze swoimi danymi);
- Przedmioty – tabela definiująca wszystkie przedmioty w danej szkole;
- OcenyDefinicje – tabela definiująca oceny (ich pełne nazwy, nazwy skrócone, jak również wartości liczbowe);
- Nauczyciele – tabela definiująca nauczycieli pracujących w danej szkole;
- Oceny – tabela, w której łączymy wszystkie informacje z poprzednich tabel, tj. definiując wpis w takiej tabeli, otrzymujemy informację o tym, który uczeń, z jakiego przedmiotu i kiedy uzyskał ocenę oraz który nauczyciel dokonał takiego wpisu.



Uczniowie

UczenID	Nazwiskolmie	DataUr	MiejsceUr	Kod	Miejscowosc	UlicaNr
1	Kowalski Zenon	1988-10-12	Wrocław	21-234	Zaklików	Willowa 2
2	Nowak Euzebiusz	1998-05-13	Pułtusk	35-144	Pułtusk	Lipowa 1/41
3	Pawlak Karolina	1998-12-01	Warszawa	54-111	Szczebrzeszyn	Kolejowa 48

Przedmioty

PrzedmiotID	NazwaPelna	NazwaSkrot
1	matematyka	matemat
2	język polski	j.pol
3	geografia	geog

OcenyDefinicje

OcenaDefID	NazwaPelna	NazwaSkrot	Wartosc
1	bardzo dobry	bdb	5
2	dobry	db	4
3	dostateczny	dst	3
4	dopuszczający	dop	2
5	niedostateczny	ndst	1

Nauczyciele

NauczID	Nazwiskolmie	Wyksztalcenie	StazPracy	Telefon	Mail
1	Kowal Stefan	mgr - matematyka	2	660 625 978	kow@exapmle.com
2	Nowakowski Jan	dr - polonistyka	22	770 330 592	now@exapmle.com
3	Kargul Jadwiga	licencjat - geografia	13	560 449 728	paw@exapmle.com

Oceny

OcenaID	UczenID	PrzedmiotID	NauczycielID	OcenaDefID	Data	Godzina
1	2	3	2	3	2014-10-05	9:55
2	1	2	3	5	2014-10-05	12:30
3	1	5	2	6	2014-10-06	8:20

Rysunek 2: Przykład tabel i ich połączeń w relacyjnej bazie danych [21]

Oczywiście, relacyjne bazy danych to niejedyny model, za pomocą którego można w wydajny sposób zapisywać dane i relacje między nimi. Obecnie coraz popularniejszymi modelami są nierelacyjne bazy danych z ang. *NoSQL*.

Akronim pochodzi od słów „*Not only Structured Query Language*". Taki typ bazy danych nie tyle eliminuje klasyczny, relacyjny model, co doskonale go uzupełnia. Rosnąca w krótkim czasie popularność rozwiązań typu BigData leży u podstaw dynamicznego rozwoju modelu NoSQL. Nierelacyjne bazy danych w przeciwieństwie do klasycznych silników pozwalają na szybką analizę danych nieustrukturyzowanych i badanie korelacji pomiędzy nimi. W tradycyjnej bazie schemat i relacje są narzucone z góry. Za pomocą odpowiednich zapytań SQL możemy uzyskać strukturalne odpowiedzi mieszczące się we wcześniej opisanych ramach [22], [23].

Można wyróżnić co najmniej 5 typów baz danych nierelacyjnych [24]–[26]:

1. Column – najbardziej podobny typ do relacyjnej bazy danych. Od relacyjnych baz danych wyróżnia go wielkość przechowywanych danych. Ten typ bazy NoSQL nazywany jest również typem BigTable. Jego przykładem może być Cassandra, Druid lub HBase.
2. Document – typ bazy danych, w której przechowujemy całość informacji jako jeden wpis – dokument. Przykładem takiego typu bazy danych może być Apache CouchDB lub MongoDB.
3. Key-value – typ bazy danych, w której wartości przechowywane są za pomocą relacji klucz – wartość. Podając określony klucz otrzymujemy daną wartość. W świecie IT taki typ baz danych możemy porównać do używania klasycznej mapy, w której owa relacja klucz - wartość również występuje. Przykładem takiego typu może być baza danych Couchbase, Dynamo, Redis, jak również MemcacheDB.
4. Graph – typ bazy danych, w której nieusystematyzowane dane połączone są różnymi typami relacji. Zarówno dane nie muszą posiadać różnej struktury, jak również połączenia pomiędzy tymi danymi mogą znacząco się różnić. Taki typ bazy danych można porównać do klasycznego grafu zawierającego pewne informacje i połączenia między nimi, występuje on np. w AllegroGraph lub Neo4j.
5. Multi-model – jest to najbardziej nieusystematyzowany typ nierelacyjnej bazy danych. W tym typie mogą tak naprawdę wystąpić połączenia pomiędzy danymi znane zarówno z relacyjnych baz danych, jak i poprzednich typów nierelacyjnych baz danych. Doskonałym przykładem takiego typu może być baza danych OrientDB.

Dzisiaj większość dużych organizacji konstruuje swoje autorskie narzędzia do magazynowania i zarządzania danymi. Nie oznacza to, że codziennie budowane są nowe typy baz danych. Częściej wykorzystuje się wiele typów rozwiązań do jednego problemu. Takie podejście pozwala skorzystać z potencjału każdego z narzędzia osobno. Rozsądne ich połączenie w całość daje możliwość uzyskania ogromnego zysku, którego nie dałoby się uzyskać wykorzystując tylko jeden typ rozwiązania.

Przykładem połączenia wielu typów baz danych w celu optymalnego magazynowania informacji może być architektura baz danych portalu Twitter (internetowy serwis informacyjny i społecznościowy, w którym ludzie komunikują się w krótkich wiadomościach zwanych tweetami). Serwis ten wykorzystuje [27], [28]:

- relacyjną bazę danych MySQL – taka baza danych w dużym stopniu służy do podstawowego przechowywania wiadomości oraz danych użytkowników;
- FlockDB - jest to wewnętrzna grafowa baza danych, która stosowana jest do przechowywania informacji o połączeniach i interakcjach pomiędzy użytkownikami;
- Memcached – baza danych służąca jako pamięć podręczna (ang. *cache*), służy ona zapamiętywaniu najczęściej wykorzystywanych danych przez serwis tak, aby użytkownik podczas interakcji z systemem otrzymywał odpowiedź jak najszybciej;
- Cassandra – w niej Twitter generuje unikalne identyfikatory, które potrzebne są dla każdej wiadomości pochodzącej z serwisu;
- Gizzard – wewnętrzna implementacja bazy danych, która obecnie nie jest już utrzymywana przez serwis;
- Apache Lucene – baza danych, na której oparte jest wyszukiwanie wszelkiego rodzaju danych (użytkowników, wiadomości, słów kluczowych) w systemie;
- HBase oraz Hadoop – bazy danych, o których użyciu również informuje Twitter, zaznaczając, że są one bardzo mocno wykorzystywane, jednak nie są podane szczegóły wykorzystania;
- Redis – w tej bazie danych Twitter przechowuje oś czasu, na której jest w stanie umieścić wiadomości użytkowników w prawidłowej kolejności.

Jak widać, informacje o użytkownikach, wiadomościach i interakcjach między nimi są rozsiane pomiędzy wiele systemów. Większość portali oferuje także swoim użytkownikom dostęp do tych danych poprzez udostępnienie swojego API (ang. *Application Programming Interface* -

interfejs programowania aplikacji, interfejs programistyczny aplikacji). Zadaniem takiego interfejsu programowania aplikacji jest dostarczenie odpowiednich specyfikacji podprogramów, struktur danych, klas obiektów i wymaganych protokołów komunikacyjnych tak, aby użytkownik końcowy mógł otrzymać informację, o jaką zapytał program [29].

Przykładami portali, które udostępniają swoje API, mogą być:

- Twitter i Twitter API,
- Facebook wraz z Social Graph API (na bazie Open Graph API),
- GitHub wraz z GitHub REST API v3,
- LinkedIn z własnym modelem agregowania informacji o użytkownikach i ich udostępniania.

Jak można zauważyć jest wiele metod magazynowania informacji. Samo przechowywanie danych nie jest już problemem. Szeroki dostęp do urządzeń pamięci masowej (np. dysków twardych) oraz ich niska cena decydują o tym, że obecnie można przechowywać informacje o nieograniczonej objętości.

Najnowsze trendy pokazują, że warto gromadzić przeróżne dane, często nieustrukturyzowane, które być może początkowo wydają się nieistotne, ale w ostatecznym rozrachunku dostarczają niesamowicie wartościowych informacji biznesowych. Nie dziwią więc w obecnych czasach informacje, że banki oprócz przechowywania informacji o saldzie na koncie bankowym oraz poszczególnych transakcjach zaczynają gromadzić dane o pogodzie danego dnia [30], [31].

Rozdział 3

Przetwarzanie języka naturalnego

Jeśli niemal nieograniczonym stało się przechowywanie luźnych danych, niekiedy w ogóle nie połączonych ze sobą, to problemem, jaki zaczął się pojawiać, jest sposób interpretowania tych danych. Wiele informacji to dane surowe – zarówno liczby opisujące dane zagadnienie, jak również zdania w języku naturalnym. Niestety komputer sam z siebie nie jest w stanie zrozumieć języka naturalnego. Potrzebne są narzędzia, które umożliwią mu wydobycie z szeregu liter przydatnych informacji.

Odpowiedzią na te problemy jest gałąź sztucznej inteligencji zajmująca się przetwarzaniem języka naturalnego (ang. *Natural Language Processing, NLP*). Pomaga ona komputerom zrozumieć, interpretować i manipulować ludzkim językiem. NLP czerpie z wielu dyscyplin, w tym informatyki i lingwistyki komputerowej, dążąc do wypełnienia luki między komunikacją ludzką a zrozumieniem komputerowym [32], [33].

Językiem komputera był i nadal jest kod maszynowy. Na najniższym poziomie komunikaty generujące działania komputera przekładane są na ciąg zer i jedynek. Od lat ludzkość stara się poprawić sposób „komunikacji" z komputerami. W latach dwudziestych XX wieku programiści używali kart dziurkowanych do komunikacji z pierwszymi komputerami. Był to żmudny manualny proces, który został zrozumiany przez stosunkowo małą liczbę osób [34]. W obecnych czasach możemy porozumiewać się z urządzeniami, wykorzystując język naturalny. Urządzenia, takie jak Amazon Echo lub Google Home, są w stanie zapamiętać wypowiadane do nich zdanie, następnie dzięki skomplikowanym algorytmom wydobyć z nich sens i odpowiedzieć użytkownikowi na zadane pytanie lub prośbę [35]. Przykładem może być wypowiedź „Hej Google, odtwórz muzykę rockową", a po chwili z głośnika będziemy mogli słyszeć ulubione fragmenty muzyczne z gatunku rocka. Interakcja ta mogła w całości być zrealizowana, gdy urządzenie:

- uaktywniło się na charakterystyczną frazę – w tym przypadku słów „Hej Google",
- zapamiętało wypowiadane do niego zdanie,
- przeanalizowało komunikat, wydobywając z niego kluczowe informacje,
- wykonało akcję,
- przekazało informacje zwrotne.

Całość tej operacji zajęła kilka sekund, a było to możliwe dzięki przetwarzaniu języka naturalnego wraz z innymi elementami sztucznej inteligencji, takimi jak uczenie maszynowe i głębokie uczenie.

To jeden z przykładów czynności, w których przetwarzanie języka naturalnego może być pomocne. Do innych czynności można zaliczyć m. in.:

- uzyskanie odpowiedzi na pytania – przykładem może być zapytanie aplikacji: „Hej Google, kto napisał Pana Tadeusza?". Jako odpowiedź usłyszymy: „Adam Mickiewicz".
- ekstrakcja informacji i zapisanie jej w uporządkowanej formie. Jeśli otrzymujemy e-mail z zaproszeniem na spotkanie, to algorytmy korzystające z NLP mogą automatycznie dodać takie wydarzenie do naszego internetowego kalendarza, wydobywając wszystkie wartościowe informacje odnośnie spotkania, takie jak godzina i data spotkania, miejsce oraz temat rozmowy. W tym przykładzie możliwe jest wydobycie z tekstu informacji, która doskonale pasuje w strukturyzowaną formę wydarzenia [36].
- wydobycie informacji wraz z analizą sentymentalną (ang. *sentiment analysis*) wypowiedzi. Jako przykład takiego działania może być pozyskanie informacji o konkretnym modelu aparatu fotograficznego lub smartphona. Algorytmy są w stanie przeanalizować wiele stron internetowych, na których użytkownicy dyskutują odnośnie danego modelu urządzenia. Następnie możliwe jest wydobycie informacji o cechach takiego przedmiotu wraz z informacjami, czy wypowiedź posiada np. ton pozytywny lub negatywny. Dokonując takiej analizy, użytkownikowi końcowemu przedstawiana jest tabela z informacjami o wadach i zaletach wybranego modelu [37].
- translacja wypowiedzi - ten typ działania NLP opiera się na możliwości automatycznego tłumaczenia zdania w jednym języku na język docelowy.

Podczas analizy tekstów można napotkać na wiele trudności, które wpływają na sposób i jakość działania algorytmów [38], [39]. Do takich trudności można zaliczyć:

- niejednoznaczność wyrazów - wynikającą z tego, iż wiele słów w języku polskim, jak i angielskim ma wiele interpretacji. Znajomość pełnego kontekstu wypowiedzi (najczęściej całego zdania) powoduje jednoznaczność wypowiedzi. Przykładem może być słowo zamek, które w zależności od całego zdania definiuje, czy chodzi o „zamek w drzwiach", „zamek, w którym mieszka piękna królewna", „zamek w spodniach", czy „zamek w meczu hokejowym".

- używanie „niestandardowego” języka - wiele tekstów, które powstają (najczęściej w internecie) może być różnie zapisywanych z uwagi na specyficzne działanie danego serwisu internetowego czy w zależności od grupy osób, która ten tekst czyta. Przykładem może być wpisywanie wypowiedzi, tzw. tweetów w portalu Twitter. Do oznaczenia kluczowych słów używa się symbolu #, a aby wspomnieć o danej osobie, jej identyfikator należy poprzedzić symbolem @. Stopniowo używanie emotikon, takich jak np. „uśmieszków” :-) lub :-(również zaczyna być powszechne.
- stosowanie idiomów, czyli konstrukcji językowych właściwych tylko danemu językowi, niedających się dosłownie przetłumaczyć na inny język. Jako przykład idiomów w języku polskim można wymienić:
  - w tym sęk – w tym problem,
  - trafić kulą w płot - powiedzieć coś nietrafnie,
  - piąte koło u wozu – osoba lub rzecz przeszkadzająca,
  - urwanie głowy – pośpiech.

  Również w innych językach występują idiomy, poniżej przykłady dla języka angielskiego:

  - beat around the bush – unikać powiedzenia czegoś dosłownie,
  - miss the boat – być za późno, spóźnić się.
- używanie neologizmów lub archaizmów – neologizmy to wyrazy nowe, które służą zazwyczaj do nazywania nowych pojęć i przedmiotów. Archaizmami z kolei nazywamy wyrazy stare, które wyszły już z powszechnego użycia. Przykładem neologizmów mogą być słowa „nokaut”, „atelier”, „wygooglować”, natomiast przykładem archaizmów mogą być słowa „waćpan”, „jeno”, „wżdy”.
- występowanie w tekście skomplikowanych nazw, takich jak np. „Borne Sulinowo”, gdzie słowo Borne odnosi się do pełnej nazwy miasta i nie powinno być użyte rozłącznie ze słowem Sulinowo.

Jak widać, jest to spory problem podczas analizy tekstów, gdyż potrzebna jest znajomość zarówno języka, w jakim tekst został napisany, jak również potrzebna jest wiedza odnośnie słów, które w nim występują. Dopiero połączenie tych dwóch zasobów umożliwi pełniejsze analizowanie dostępnych tekstów.

W przetwarzaniu języka naturalnego można również wydzielić trzy główne kategorie podejść [40]:

- podejście bazujące na wszelkiego rodzaju zasadach (ang. *rule-based approaches*). W takim podejściu można wyróżnić różnego rodzaju gramatyki bezkontekstowe oraz wyrażenia regularne;
- podejście bazujące na rachunku prawdopodobieństwa oraz bazujące na uczeniu maszynowym (ang. *machine learning*). Do metod wykorzystywanych w takim podejściu możemy zakwalifikować metodę największej wiarogodności (ang. *maximum likelihood estimation*) oraz klasyfikatory, np. liniowe (ang. *linear classifiers*);
- kolejnym podejściem jest wykorzystanie uczenia głębokiego (ang. *Deep Learning*) wraz z wszelkiego rodzaju sieciami neuronowymi (splotowe sieci neuronowe, rekurencyjne sieci neuronowe).

Do przetwarzania języka naturalnego można podejść również z drugiej strony – analizując statystykę słów w zdaniach. Mając zbiór tekstów (korpusów tekstów), można na ich podstawie spróbować „nauczyć się" leksykalnych i strukturalnych właściwości języka, zamiast analizować tekst, wyłącznie używając do tego celu, np. etykietowania części mowy (ang. *part of speech labels*) [40].

Ciekawa właściwość słów została zaobserwowana podczas badania korpusów tekstów. George Kingsley Zipf zauważył, że częstość występowania słów w języku naturalnym jest odwrotnie proporcjonalna do ich pozycji w rankingu [41]. Obserwacja ta została nazwana nazwiskiem odkrywcy, a bazuje na frekwencyjnym słowniku dla badanego języka naturalnego. Słownik ten zawiera słowa wraz z częstotliwością ich występowania. Za przykładem [42] w tabeli 1. przedstawiono najczęściej występujące słowa wg Słownika Frekwencyjnego Polszczyzny Współczesnej [43].

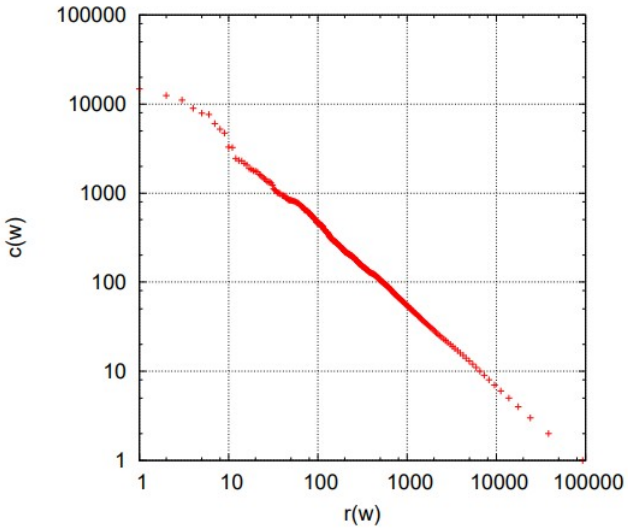
Tabela 1: Korpus Słownika Frekwencyjnego Polszczyzny Współczesnej

Ranga r(w)	**Częstość c(w)**	**Słowo w**
1	14767	w
2	12473	i
3	11093	się
4	8750	na
5	7878	nie

6	7605	z
7	6004	do
8	5233	to
9	4675	że
10	3292	jest

Jak można zauważyć, częstość oznacza liczbę wystąpień badanego słowa w analizowanym korpusie tekstowym, natomiast rangą można nazwać liczbę porządkową słowa na liście rankingowej. Najczęściej występujące słowo posiada rangę równą 1, następne słowo w kolejności występowania posiada rangę 2 itd.

Na rysunku 3. przedstawiono w skali logarytmicznej zależność częstotliwości występowania słów do ich rangi.



Rysunek 3: Zależność logarytmiczna częstotliwości występowania słów do ich rangi [42]

Prawo Zipfa można więc interpretować w następujący sposób:

Jeżeli słowo w_1 ma rangę 10 razy większą niż słowo w_2, to słowo w_1 ma częstość 10 razy mniejszą niż słowo w_2.

Powyższą interpretację można zapisać za pomocą wzoru:

$$c_w \approx \frac{A}{r_w} \tag{1}$$

gdzie $\mathbf{c_w}$ oznacza częstość występowania słowa w, $\mathbf{r_w}$ oznacza pozycję słowa w rankingu, natomiast literą **A** definiujemy pewną stałą liczbową.

Zależność ta posiada niezwykle prostą interpretację. Im większa ranga, tym mniejsza częstość słowa w korpusie. Zależność jest oczywiście równoważna stwierdzeniu, że dla każdego słowa w tekście iloczyn rangi i częstości jest stały [44]. Co więcej okazało się, że rozkład ten jest w zasadzie uniwersalny i niezależny od języka.

Rozkład częstości jest zwykle charakterystyczny dla badanych tekstów. Istnieje stosunkowo niewiele słów, które bardzo często pojawiają się w treści dokumentu oraz dużo słów, które występują bardzo rzadko. Słowa występujące bardzo często to w większości zaimki i przyimki, które samodzielnie nie przekazują praktycznie żadnej informacji [45].

Prawo Zipfa czasem jest określane jako zasada Pareto w lingwistyce [46]. Rozkłady spełniające powyższe prawo występują nie tylko w lingwistyce. Można do nich zaliczyć:

- wspomniane wcześniej prawo Pareto (zasada 80/20) dla rozkładu np. dochodów ludności,
- prawo Lotki dla rozkładu cytowań artykułów naukowych,
- prawo Gibrata dla rozkładu wielkości miast,
- prawo Zipfa jest stosowane również w teorii muzyki. Ważne jest ono w kontekście badań struktur muzycznych, ich hierarchicznej organizacji i stanów emocjonalnych (smutek, gniew, poczucie szczęścia itp.),
- liczba trzęsień ziemi rośnie od największych do najsłabszych według zależności potęgowej,
- funkcja rozkładu galaktyk w gromadach ma charakter potęgowy,
- erupcje wulkaniczne i ich rozmiary są zgodne z rozkładem Zipfa, to samo dotyczy rozmiarów wysp,
- proteiny i sieci metaboliczne posiadają własność niezmienności względem skali - topologię, dla której charakterystyczne są rozkłady Zipfa [44].

Uzupełnieniem prawa Zipfa jest uściślenie Mandelbrota, polegające na określeniu dokładniejszego modelu relacji, w oparciu o funkcje wykładniczą [47]. Dokładniej dla pewnych stałych *B, d, P* relacja między częstością a rangą wynosi:

$$\log(c_w) = \log(P) - B * \log(r_w + d) \tag{2}$$

Podczas analizy tekstu zauważyć można, że pomiędzy niektórymi słowami powstają naturalne relacje, często słowa występują w pewnych grupach, niektóre przed lub po innych

wyrazach. Taka wiedza kolokacyjna może być punktem wejścia do głębszej analizy semantyki zdań. W szczególności wykorzystanie modeli statystycznych daje dobre rozwiązanie problemu niejednoznaczności: modele statystyczne są solidne, w dobry sposób generalizują i dobrze zachowują się w obecności błędów językowych, jak i nowych danych.

Tak więc metody statystyczne wykorzystywane w analizie języka naturalnego doprowadziły do sukcesu, jakim jest ujednoznacznienie w systemach wielkoskalowych wykorzystujących język naturalny. Ponadto często można oszacować parametry modeli statystycznych NLP automatycznie z korpusów tekstowych. Ta zdolność automatycznej nauki nie tylko zmniejsza ludzki wysiłek w tworzeniu systemów NLP, ale porusza interesujące kwestie naukowe dotyczące nabywania języka ludzkiego [40].

Etapy analizy tekstów

Proces analizy tekstów składa się z wielu etapów. W zależności od celu, jakiemu ma służyć dany algorytm, niektóre z wymienionych etapów mogą występować w innej kolejności, a niektóre etapy mogą zostać pominięte w całym procesie. Poniżej zostały przedstawione etapy analizy tekstu w celu wydobycia informacji semantycznej, a więc poznania właściwego znaczenia tekstu [47] – [49].

1. Pierwszym etapem jest podzielenie dostępnego tekstu na zdania. Taki podział może być zadaniem bardzo prostym (wystarczy rozdzielić wprowadzony tekst, wykorzystując znaki końca zdania – kropka, pytajnik, wykrzyknik) i równocześnie bardzo wymagającym, podczas którego należy wykorzystać wiedzę o składni danego języka. Bezpieczniej również posługiwać się terminem podziału tekstu na wypowiedzenia niż na zdania, ponieważ termin „wypowiedzenie” obejmuje swoim zasięgiem wszelkiego rodzaju równoważniki zdań, jak i pojedyncze słowa.
2. Kolejnym etapem jest podzielenie każdego wypowiedzenia na tokeny. Tokenem będzie najczęściej pojedyncze słowo. W zależności od języka, którego dotyczy tokenizacja, proces ten może być mniej lub bardziej skompilowany. Dla języka polskiego w zdaniu „Alicja przeszła po jasnozielonej trawie.” możemy chcieć rozdzielić słowo jasnozielonej na dwa tokeny „jasno” oraz „zielona”. Podobnie możemy postąpić dla języka angielskiego w stosunku do słowa „wanna”, dla którego możemy otrzymać dwa tokeny „want” oraz „to”.
3. Następnym etapem może być normalizacja tekstu. Jest to proces przetwarzania tekstów, podczas którego przeprowadzane jest jego uspójnienie w celu ułatwienia dalszej interpretacji. Istnieje kilka rodzajów normalizacji tekstu. Możemy do nich zaliczyć zmianę

wielkości liter, normalizację skrótów, wyrażeń numerycznych, jak również znaków specjalnych, zmianę znaków interpunkcyjnych oraz usuwanie (lub zmienianie) znaków diakrytycznych. Przykładem takiej operacji będzie zamiana zdania „Zam. na os. Jana III Sobieskiego 55B/3." na zdanie „Zamieszkały na osiedlu Jana trzeciego Sobieskiego pięćdziesiąt pięć B przez trzy" [51]. Normalizacja tekstu wykonywana jest zazwyczaj po to, aby zmniejszyć rozmiar modelu, w oparciu o który analizujemy tekst.

4. W dalszej kolejności należy rozpoznać obecne w wypowiedzi byty nazwane (ang. „*named entities*"). Byty nazwane to, w uproszczeniu, byty reprezentowane przez nazwy własne – w odróżnieniu od rzeczowników pospolitych. Ich rozpoznanie nie zawsze jest łatwe. Często zależy od kontekstu. Nierzadko w wyniku takiej analizy dostajemy kilka hipotez. Przykładowo, „house" to po angielsku „dom", ale może to być również nazwa serialu lub nazwisko jednego z głównych bohaterów danego serialu. Po tym kroku analizy powinniśmy wiedzieć, która część naszego wypowiedzenia to byt nazwany i jakiego typu jest to byt (miejsce, osoba, tytuł, …).
5. Następnym etapem, jaki można przeprowadzić podczas analizy tekstu, może być ujednolicenie form wyrazów. Zastosować można tutaj dwie metody – lematyzacji lub stemmingu. Lematyzacja to sprowadzenie słowa do jego podstawowej postaci. W przypadku czasownika będzie do bezokolicznik, w przypadku rzeczownika – mianownik liczby pojedynczej. Do wykonania tego zadania potrzebny jest słownik lub rozbudowany zestaw reguł fleksyjnych dla danego języka. Stemming z kolei to obcięcie wszelkiego rodzaju przedrostków i przyrostków, mające na celu dotarcie do nieodmiennego „rdzenia" reprezentującego wyraz. Sam rdzeń niekoniecznie jest poprawnym słowem. Algorytm stemmera nie musi być zależny od języka. Niestety po takim zabiegu utracony zostaje pełny kontekst wypowiedzi, co może nie być akceptowalne dla różnych algorytmów. Podczas tego etapu możemy również rozpoznać części mowy wraz z informacją o danych odmianach. Głównie taką informację zwraca zastosowany lematyzator. Analiza tego typu może być oparta na słowniku, może również wykorzystywać końcówki fleksyjne. Przykładowo dla języka polskiego, jeśli dane słowo kończy się na „*ować*", to można przyjąć, że słowem tym jest czasownik.

6. Po tak przeprowadzonych etapach następuje analiza składniowa wprowadzonego wyrażenia, czyli tzw. parsowanie tekstu. Często na tym etapie można również przeprowadzić analizę semantyczną wypowiedzenia, a więc wydobyć znaczenie tekstu. Do metod używanych podczas analizowania tekstu można zakwalifikować wyrażenia regularne, jak i gramatyki formalne (oparte na teorii Chomskiego [52]), które mogą być tworzone manualnie lub

automatycznie jak również uczenie maszynowe. Uczenie to w odróżnieniu od gramatyk formalnych nie potrzebuje zdefiniowanych reguł, a wykorzystywane są wnioski wyciągnięte podczas trenowania systemu przy użyciu wybranej metody (np. *CRF - Conditional Random Field*) na dużej ilości odpowiednio opisanego tekstu.

7. Wynikiem przeprowadzenia wcześniejszych etapów jest otrzymanie reprezentacji wyrażenia poprzez dane semantyczne. Dzięki tak przeprowadzonemu procesowi dla każdego tokenu, który opisuje słowo z wprowadzonego tekstu, zostaje przyporządkowane jedno lub więcej powiązań.

Oczywiście to jeden ze sposobów analizy tekstów. Dla przykładu, gdy zadaniem programu jest ekstrakcja z dostępnego teksu kluczowych informacji, poszczególne etapy mogą wyglądać następująco:

1. Pierwszym etapem, podobnie jak poprzednio, będzie segmentacja zdań, a więc rozbicie tekstu na osobne zdania. Można założyć, że każde zdanie jest osobną myślą lub pomysłem. Dlatego też można analizować zdanie po zdaniu, gdyż każde stanowi pełny przekaz komunikatu.

2. Kolejnym etapem będzie tokenizacja słów, czyli rozbicie wypowiedzenia na oddzielne słowa lub tokeny. Przy takim przetwarzaniu tekstu znaki interpunkcyjne również traktujemy jako oddzielne tokeny, ponieważ interpunkcja ma również duże znaczenie.

3. Następnym krokiem będzie przyjrzenie się każdemu tokenowi i próba ustalenia konkretnej części mowy dla każdego tokena. W tym etapie program będzie próbował ustalić, czy dany token to rzeczownik, czasownik, przymiotnik itd. Znajomość roli każdego słowa w zdaniu pomoże dowiedzieć się, o czym ono mówi. Najczęściej w tym etapie wykorzystuje się gotowe modele części mowy, które zostały pierwotnie przeszkolone przez podawanie milionów zdań, a część mowy dla każdego słowa została wcześniej poprawnie oznaczona. Modele te są całkowicie oparte na statystykach. Oznacza to, że algorytm nie jest w stanie stwierdzić, co oznaczają słowa w taki sam sposób, w jaki robią to ludzie. Na podstawie podobnych zdań i słów, które zostały przetworzone wcześniej, algorytm stara się oznaczyć część mowy.

4. Gdy poszczególnym tokenom zostały przyporządkowane części mowy, kolejnym etapem będzie przekształcenie tokenów do formy podstawowej. W wielu językach słowa pojawiają się w różnych formach. Pomimo odmiennej formy fleksyjnej, oznaczają one to samo. Przykładem mogą być zdania „*Mam pięknego czarnego psa.*”, „*Ala dała psu dużą kość.*”. Pomimo użycia dwóch słów – psa i psu, w obu zdaniach chodzi o wyraz podstawowy

„pies”. Jak zostało to opisane powyżej, proces ten nosi nazwę lematyzacji. Wykonywana jest ona zazwyczaj poprzez przeglądanie tabeli lematów form słów w oparciu o ich część mowy.

5. Po wykonaniu lematyzacji można wykonać „odszumianie” tekstu. W zdaniach występuje wiele słów wypełniających, które pojawiają się bardzo często, przez co podczas robienia statystyk tekstowych można uznać, że słowa te pojawiają się znacznie częściej niż inne. Niektóre mechanizmy NLP oznaczą je jako słowa zatrzymujące, czyli słowa, które można odfiltrować przed wykonaniem jakiejkolwiek analizy statystycznej. W języku angielskim przykładem takich słów są: „a”, „an”, „the”.
6. Kolejnym etapem będzie analiza zależności – czyli parsowanie tekstu. Wynik tego działania to ustalenie, jak poszczególne słowa odnoszą się do siebie nawzajem. Jego celem jest także zbudowanie drzewa, które przypisuje jedno słowo macierzyste do każdego słowa występującego w zdaniu. W związku z tym na tym etapie tworzone jest drzewo zależności, gdzie zazwyczaj głównym korzeniem drzewa będzie główny czasownik w zdaniu. Oprócz takiej identyfikacji można również spróbować przewidzieć rodzaj relacji, która istnieje między dwoma słowami. Podobnie jak przy ustalaniu części mowy za pomocą modelu uczenia maszynowego, analizowanie zależności działa również poprzez podawanie słów do modelu uczenia maszynowego i generowanie wyniku. Niejednokrotnie jednak analizowanie zależności między słowami jest szczególnie skomplikowanym zadaniem.
7. Przedostatnim krokiem będzie rozpoznawanie obecnych w wypowiedzi bytów nazwanych (ang. *named entities*). Celem Named Entity Recognition (NER) jest wykrywanie i oznaczanie rzeczowników za pomocą reprezentowanych przez nich rzeczywistych treści.
8. Po tak przeprowadzonym procesie zdanie wejściowe zostało przekształcone na tyle, że dostępne są dla niego użyteczne dodatkowe dane. Rozpoznane zostały części mowy dla każdego słowa, zostało wyznaczone, jak słowa odnoszą się do siebie wzajemnie oraz które słowa mówią o nazwanych bytach. Ostatnim etapem jest odnalezienie i połączenie słów, które odnoszą się do tego samego słowa, a w tekście występują pod postacią zaimków - słów takich, jak on, ona i to. Są to skróty, które używane są, aby nie powtarzać słowa głównego. Ludzie naturalnie śledzą, co te słowa reprezentują na podstawie kontekstu.

Jak zostało przedstawione powyżej, dla osiągnięcia różnych rezultatów wykorzystuje się różne etapy analizy tekstu. Można jednak zauważyć, że większość z nich jest wspólna dla wszystkich zadań dotyczących przetwarzania języka naturalnego.

Narzędzia wykorzystywane podczas przetwarzania tekstu

Wyrażenia regularne

Najprostszym narzędziem podczas pracy z łańcuchami znaków jest zastosowanie wyrażeń regularnych (ang. *regular expressions*). Opisują one łańcuchy symboli za pomocą odpowiednich wzorców. Mogą one określać zarówno zbiór pasujących łańcuchów, jak również wyszczególniać jedynie istotne części łańcucha. Narzędzie to znalazło bardzo szerokie zastosowanie również w procesie przetwarzania tekstu. Za ich pomocą można w łatwy sposób opisywać wzorce tekstu, a następnie sprawdzić, czy podany ciąg znaków pasuje do wzorca lub czy w tekście pojawiają się wystąpienia danego wzorca. Stosując wyrażenia regularne można również wyszukać, podzielić czy zmodyfikować istniejący już łańcuch znaków [53].

Aby zapisać wyrażenie regularne, można posłużyć się następującą konwencją:

- [abc] – poszukiwane jest wystąpienie litery a, b lub c (dowolny znak ze wskazanego zbioru);
- [^abc] – poszukiwane jest wystąpienie liter za wyjątkiem liter a, b lub c (negacja, dowolny znak różny od tych w zbiorze);
- [a-zA-Z] – poszukiwany jest zakres liter od a do z (małe litery) oraz od A do Z (duże litery) (dowolny znak z zakresu);
- [0-9] – poszukiwane jest wystąpienie cyfry od 0 do 9.

Można również skorzystać przy zapisie z tzw. kwantyfikatora wystąpień:

- X? - poszukiwany ciąg znaków występuje raz lub wcale,
- X* - poszukiwany ciąg znaków nie występuje lub występuje przynajmniej raz,
- X+ - poszukiwany ciąg znaków występuje raz lub więcej razy,
- X{n} – poszukiwany ciąg znaków występuje dokładnie n razy.

Znając sposób zapisu wyrażeń regularnych, można sprawdzić np. występowanie kodu pocztowego dzięki zapisowi: [0-9]{2}-[0-9]{3}.

Segmentacja zdania

Jak można zauważyć, część ze znaków specjalnych, takich jak „!” (wykrzyknik) lub „?” (pytajnik), ma jasno określone miejsce w tekście. Przez co ich wystąpienie jest dość jednoznacznie określone. Występują jednak znaki specjalne, takie jak „.” (kropka), która w tekście może pojawić

się w wielu miejscach i w zależności od wystąpienia, pełni różną rolę, a więc wprowadza pewną niejednoznaczność w tekście. Przykładowe zastosowania kropki:

- wystąpienie kropki na końcu zdania kończy je;
- wystąpienie kropki po skrótach wyrazów takich, jak godz. (godzina), prof. (profesor), ul. (ulica) kończy jedynie dany skrót;
- wystąpienie w liczbach np. w języku angielskim 5.3, .08% jest informacją o danej licznie i rozdziela znak jedności od znaku dziesiątek.

Aby rozróżnić, czy występujący znak specjalny kończy zdanie, można posłużyć się manualnie opracowanymi regułami, wyrażeniami regularnymi lub nawet algorytmami uczenia maszynowego.

Drzewa decyzyjne

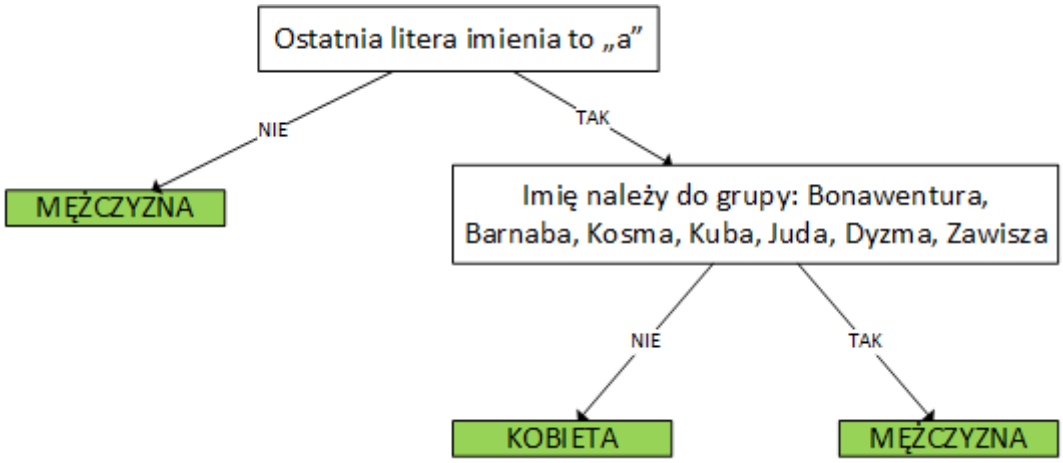
Kolejnym narzędziem, jakie można zastosować podczas przetwarzania tekstu, to zbudowanie drzewa decyzyjnego, którego największą zaletą jest jego przejrzysta struktura. Struktura ta jest równocześnie prosta do zrozumienia i wdrożenia.

Drzewa decyzyjne (ang. *Decision Tree, DT*) są dobrze znaną metodologią klasyfikacji oraz przewidywania w uczeniu maszynowym. Modelem tej metody jest drzewo, w którym każdy węzeł jest decyzją, a każdy liść reprezentuje klasę wyjściową (etykietę lub dystrybucję). Węzeł może mieć dowolną liczbę dzieci, ale zazwyczaj większość algorytmów implementuje jedynie drzewa binarne z pytaniami binarnymi. Klasyfikatory drzew decyzyjnych są bardzo popularnym wyborem, głównie dlatego, że są łatwe do trenowania, a dzięki analizie struktury drzewa - można łatwo zweryfikować pewne założenia lub uzyskać lepsze zrozumienie korpusów oraz zadań, w przeciwieństwie do np. sieci neuronowych, w jakich model jest matrycą liczb, które nie dają wglądu. Drzewa DT były szeroko stosowane w zadaniach przetwarzania języka naturalnego i mówionego, takich jak: tagowanie, nazwane rozpoznawanie jednostek (NER), konwersja liter na dźwięk (LTS), kategoryzacja tekstu, jak również estymacja parametrów statystycznej parametrycznej syntezy mowy.

Jedną z wad drzew decyzyjnych jest ich stosunkowo skromna wydajność w zadaniach klasyfikacyjnych. Obecne najnowocześniejsze podejścia do przetwarzania języka naturalnego i innych dziedzin badawczych wykorzystują bardziej wydajne metodologie, takie jak metody

wektorów nośnych (ang. *Support Vector Machines, SVM*), warunkowe pola losowe (*CRF*) i złożone architektury sieci neuronowych [54].

Przykład drzewa decyzyjnego określającego, czy imię pochodzenia polskiego należy do kobiety, czy do mężczyzny zamieszono na ilustracji 4.



Rysunek 4: Przykład drzewa decyzyjnego

Minimalna odległość edycyjna

Kolejnym narzędziem wykorzystywanym podczas pracy ze słowami jest zastosowanie metody minimalnej odległości edycyjnej (ang. *minimum edit distance*). Metoda ta wykorzystywana jest podczas rozwiązania problemu podobieństwa łańcuchów tekstu. Za jej pomocą można określić, jak podobne są dwa ciągi znaków [32], [55]. Jednym z zastosowań wspomnianej metody jest korekta tekstu. Niech przykładowym wyrazem będzie wyraz „granut”. Wyraz ten nie występuje w słowniku języka polskiego. Aby przeprowadzić korektę tekstu i poprawić błędne słowo, poszukiwane są wyrazy „podobne” do zadanego. Przykładowo mogą to być słowa:

- granat - kolor ciemnoniebieski, granatowy;
- granit - skała głębinowa barwy szarej, zbudowana z ziaren kwarcu, skaleni i miki;
- grant - dotacja na projekt badawczy lub artystyczny, przyznawana przez organizacje publiczne.

Niestety, problematycznym wydaje się określenie, które z zaproponowanych słów jest właściwe. Problem podobieństwa łańcuchów pojawia się także przykładowo w biologii dla dwóch sekwencji nukleotydów. Idea podobieństwa łańcuchów lub podobieństw sekwencji pojawia się również w procesie tłumaczenia maszynowego, podczas ekstrakcji informacji, jak również w procesie rozpoznawania mowy.

Można stwierdzić, iż minimalna odległość edycji między dwoma ciągami to minimalna liczba operacji edycji, takich jak:

- wstawiania,
- usuwania,
- podstawiania,

które są potrzebne do przekształcenia jednego ciągu znaków w drugi

Przykładowo liczba operacji, jakie należy przeprowadzić, aby zmienić słowo *orczyk* w słowo *oracz,* wynosi 3. Do przeprowadzenia takiego zabiegu należy bowiem usunąć ze słowa orczyk litery „y" jak i „k", a następnie wstawić literę „

Jeśli mamy podane dwa słowa, to możemy w prosty sposób wyznaczyć minimalną odległość edycyjną. Działanie algorytmu polega na wyszukaniu ścieżki, przez którą rozumiemy sekwencję edycji od łańcucha początkowego do końcowego. Stanem początkowym jest ciąg znaków (słowo początkowe), dostępne są również operacje – wstawiania, usuwania oraz podstawienia. Algorytm poprzez zastosowanie dostępnych przekształceń zamienia słowo początkowe w końcowe. Kosztem ścieżki będzie suma kosztów wykonanych operacji. Ścieżka o najmniejszym koszcie jest minimalną odległością edycyjną dla podanych dwóch ciągów znaków.

Jeśli założymy, że operacja podstawienia będzie wynosić dwa, a operacje wstawienia i usunięcia jeden, to taką odległość nazywamy odległością Levenshteina [56].
Aby zdefiniować odległość Levensteina, zakładamy, że posiadane są dwa łańcuchy znaków:

- łańcuch X o długości n (X=[1...n]),
- łańcuch Y o długości m (Y=[1...m]).

Wobec czego możemy zdefiniować macierz odległości D(i,j) przez następującą rekurencję, przedstawioną na rysunku 5.:

$$\text{Dla kazdego } i = 1...M$$

$$\text{Dla kazdego } j = 1...N$$

$$D(i,j) = min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1) + \begin{cases} 2\text{: jesli } X(i) \neq Y(j) \\ 0\text{: jesli } X(i) = Y(j) \end{cases} \end{cases}$$

Rysunek 5: Mechanizm wyznaczania odległości Levenshteina

Rozwinięciem odległości edycyjnej jest ważona odległość edycyjna, a więc taka miara odległości, dla której wprowadzone zostały dodatkowo odpowiednie wagi. Jest to bardzo pomocne, np. dla zagadnień związanych z poprawą pisowni na komputerze. Oczywistym jest, że niektóre litery są częściej wpisywane błędnie niż inne. Na rysunku 6. przedstawiono macierz z częstotliwością występowania pomyłek przy wpisywaniu liter dla języka angielskiego. Można zauważyć, że litera „E" dużo częściej jest mylona z literami „A", „I" lub „O" niż innymi. Z drugiej strony nie zdarzają się pomyłki z zamianą litery „A" z literą „B" [57]. Zatem ograniczenia domeny, w tym przypadku dotyczą nie tyle pisowni, co rozmieszczenia liter na klawiaturze i powodują istotną zmianę w tradycyjnym wyznaczaniu odległości edycyjnej. Warto więc dla takich przypadków do tradycyjnego algorytmu dodać wspomniane wcześniej wagi, dodając odpowiedni koszt dla różnych przejść, który należy dodatkowo sprawdzić.

Podczas korekty tekstu algorytmy posługują się również słownikami, w których zapisane są prawidłowe wyrazy dla danego języka. Zbiór tych słów jest zbiorem skończonym, wobec czego istnieje kilka podejść do obliczania odległości edycyjnej pomiędzy terminami słownikowymi, a terminami zapytania [58]. Są to:

- podejście naiwne – odległość obliczana jest pomiędzy wprowadzonym słowem a każdym terminem słownikowym. Metoda ta jest bardzo wolna, jak również potrzebuje bardzo dużo zasobów, np. komputera;
- podejście Petera Norviga (ang. *Peter Norvig's Approach*) – w podejściu tym wyprowadzane są wszystkie możliwe terminy, dla których odległość edycyjna jest mniejsza bądź równa 2 od terminu zapytania. Metoda ta jest szybsza od podejścia naiwnego (gdyż nie przeszukujemy wszystkich słów w całości). Niemniej łączna suma operacji algorytmu pozostaje nadal wysoka. Przykładowo dla słowa składającego się z 9 znaków algorytm ten zwraca 114 324 propozycji poprawy [56];
- podejście Faroo (ang. *Faroo's Approach*) – możliwe są usunięcia tylko z odległością edycyjną mniejszą lub równą 2 zarówno od terminu zapytania, jak i każdego terminu słownikowego. Podejście to jest o trzy rzędy wielkości szybsze w porównaniu do poprzedniego [60].

Tego rodzaju algorytmy mogą wykorzystywać również kilka innych aspektów, takich jak:

- sortowanie – sugestie sortowane są najpierw według (ważonej) odległości edycji, a następnie według częstotliwości występowania danego słowa lub liczby wyników, które sugerowane zapytanie zwróci dla danego indeksu;

- wykrywanie języka – w metodzie tej wykorzystywany jest dodatkowo specyficzny słownik sprawdzania pisowni dla wskazanego lub wykrytego języka;
- słownik – tak jak w poprzedniej metodzie wykorzystywany jest słownik (statyczny lub dynamiczny) do sprawdzania pisowni. Dynamiczny słownik może zostać wygenerowany lub uzupełniony z indeksu wyszukiwarki lub zapytań wprowadzonych przez użytkowników. W tym przypadku, jeśli częstotliwość słowa dla określonego terminu przekracza określony próg, słowo to jest dodawane do słownika. Możliwe jest również użycie samego indeksu wyszukiwania jako słownika sprawdzania pisowni [58];
- model Markowa - alternatywą dla słownika opartego na pisowni są metody statystyczne, np. ukryte modele Markowa [62].

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Rysunek 6: Macierz użycia błędnych liter dla języka angielskiego [57]

Warto zauważyć, że poprawa pisowni to niejedyny obszar, w którym można zastosować ważoną odległość edycyjną. Również w biologii, ze względu na specyfikę budowy organizmów, niektóre rodzaje operacji (specyficzne usunięcia lub dodawania) są bardziej prawdopodobne niż inne. Kolejnym przykładem wykorzystania tej cechy jest algorytm Soundex, który używa fonetyki do indeksowania nazw według dźwięku [63].

Poniżej przedstawiono zmodyfikowany algorytm służący do obliczenia ważonej odległości edycyjnej.

Inicjalizacja:

$D(0,0) = 0$

$D(i,0) = D(i-1,0) + \mathrm{del}[x(i)]; 1 < i \leq N$

$D(0,j) = D(0,j-1) + \mathrm{ins}[y(i)]; 1 < j \leq M$

Rekurencyjna zależność:

$$D(i,j) = \begin{cases} D(i-1,j) & + \mathrm{del[x(i)]} \\ D(i,j-1) & + \mathrm{ins[y(j)]} \\ D(i-1,j-1) & + \mathrm{sub[x(i),\ y(j)]} \end{cases}$$

Warunek końca:

D(N,M) oznacza wyznaczoną odległość.

Modele języka naturalnego

Klasyfikacja tekstów wydaje się być zagadnieniem, które ma bardzo wiele praktycznych zastosowań. Można tutaj wymienić [56]:

- przypisanie autorstwa tekstu danej osobie,
- rozpoznawanie płci autora - ostatnie badania dotyczące identyfikacji płci pokazały, że możemy przyjrzeć się liczbie zaimków i innych cech. Liczba determinantów, liczba fraz rzeczownikowych jest subtelnością wskazującą na różnicę między pisarzami płci męskiej, a żeńskiej. Pisarki mają tendencję do używania większej liczby zaimków, a pisarze mają tendencję do używania większej liczby faktów i determinantów w swoich frazach rzeczownikowych,
- wykrywanie spamu w korespondencji e-mailowej,
- przypisywanie kategorii i tagów do różnego rodzaju publikacji, np. książek w sklepie czy artykułów w Internecie,
- automatyczne segregowanie zleceń w systemie CRM (ang. *Customer Relationship Management*).

Aby zdefiniować pojęcie klasyfikacji tekstów jako wartości początkowe, powinniśmy posiadać: dokument – oznaczony symbolem *d* oraz ustalony zbiór cech $c = \{c_1, c_2, \ldots, c_j\}$. Wynikiem działania algorytmu klasyfikacji będzie przewidywanie klasy *c* z zadanego zestawu klas dla nieznanego dokumentu d_2.

Najprostszą możliwą metodą klasyfikacji tekstu jest użycie manualnie zdefiniowanych wcześniej reguł. Na przykład, gdy wykrywany jest spam, to możliwe jest posiadanie listy złych adresów e-mail, tzw. czarnej listy, z których ten spam pochodzi. Można również przeszukiwać treść korespondencji, wykrywając w niej zwroty często używane przy fałszywej korespondencji. Przykładowo warto zwrócić uwagę na wyrażenia „wygrałeś nagrodę" lub „zostałeś wybrany". Jeśli lista reguł i zasad jest starannie opracowana przez eksperta, to algorytm taki może uzyskać wysoką dokładność. Niemniej budowanie i utrzymywanie takich manualnie zapisanych zasad jest kosztowne. Dużo częściej w budowie systemu klasyfikacji tekstu, zamiast manualnej pracy eksperta, wykorzystywane jest uczenie maszynowe z nadzorem.

Można więc zdefiniować mechanizm klasyfikacji tekstów dla takiego przypadku w oparciu o następujące dane: oprócz dokumentu *d* oraz ustalonego zbioru cech $c = \{c_1, c_2, \ldots, c_j\}$ wykorzystywać można również zestaw szkoleniowy, a więc użycie dokumentów, które wcześniej zostały manualnie przyporządkowane do określonej klasy. Celem uczenia maszynowego jest utworzenie klasyfikatora, który mapuje nowy nieznany dokument do jednej z istniejących klas. Definicja klasyfikacji tekstów z uczeniem maszynowym pod nadzorem została zaprezentowana poniżej.

Dane wejściowe:

- dokument $\boldsymbol{d}$;
- ustalony zbiór klas $\boldsymbol{c = (c_1, c_2, \ldots, c_j\}}$;
- zestaw treningowy składający się z $\boldsymbol{m}$ ręcznie oznaczonych dokumentów $\boldsymbol{(d_1,c_1),\ldots, (d_m, c_m)}$.

Wyjście:

- klasyfikator przypisujący dokument do klasy $\gamma : d \rightarrow c$

Istnieje wiele rodzajów klasyfikatorów uczenia maszynowego. Do tej grupy można zaliczyć m. in.:

- naiwny klasyfikator bayesowski (*Naive Bayes*),

- regresja liniowa i logarytmiczna (*Simple Linear Regression, Log Regression*),
- metoda k najbliższych sąsiadów (*k-Nearest Neighbour*),
- maszyna wektorów nośnych (*Support-vector machines, SVM*).

Bez względu na to, jaki klasyfikator zostanie użyty, zadaniem klasyfikacji jest pobranie dokumentu, następnie wydobycie z niego potrzebnych danych, na podstawie których będzie można zbudować klasyfikator, określający do jakiej klasy należy dany dokument.

Warto wspomnieć o naiwnym klasyfikatorze bayesowskim. Klasyfikuje on przypadki ze względu na łączne prawdopodobieństwo wystąpienia w danej klasie posiadanych przez nie cech, czyli wartości atrybutów niedecyzyjnych. Klasyfikator ten jest oparty na twierdzeniu Bayesa dotyczącym prawdopodobieństwa warunkowego zdarzeń. Pomimo posiadania założeń o niezależności wartości atrybutów niedecyzyjnych jest bardzo często wykorzystywany. W wielu rzeczywistych sytuacjach naiwne klasyfikatory Bayesa posiadają najwyższą trafność. Dodatkowo jest to bardzo szybki algorytm [64].

Probabilistyczne modele językowe

Celem probabilistycznego modelowania języka jest przypisanie pewnego prawdopodobieństwa do zdania. Na przykład w tłumaczeniu maszynowym w zależności od przypisanego prawdopodobieństwa do zdania, możliwe jest odróżnienie, czy jest to poprawne, czy niepoprawne tłumaczenie. Również podczas procesu rozpoznawania mowy jedna fraza powinna uzyskać dużo większe prawdopodobieństwo poprawności niż fraza, która brzmi podobnie jedynie fonetycznie, a w rzeczywistości oznacza zupełnie coś innego [56].

Celem modelu językowego jest więc obliczenie prawdopodobieństwa zdania lub ciągu słów. Posiadając pewną sekwencję słów od wyrazu w_1 do wyrazu w_n, można wyznaczyć ich prawdopodobieństwo używając następującego wzoru:

$$P(W) = P(w_1, w_2, w_3, w_4, ..., w_n) \tag{3}$$

Podobnie można wyznaczyć prawdopodobieństwo nadchodzącego słowa. Chcąc wyznaczyć prawdopodobieństwo wystąpienia słowa w_5 w sekwencji w_1, w_2, …, w_5, prawdopodobieństwo będzie to określone następująco:

$$P(w_1, w_2, w_3, w_4, w_5) = P(w_5|w_1, w_2, w_3, w_4) \tag{4}$$

Można więc zdefiniować model językowy jako model, który wyznacza jedno z prawdopodobieństw

$$P(W) \text{ lub } P(w_n|w_1, w_2, ..., w_{n-1}) \tag{5}$$

gdzie **W** oznacza ciąg słów. Łączne prawdopodobieństwo całego ciągu lub prawdopodobieństwo warunkowe ostatniego słowa podane w poprzednich słowach można nazwać modelem językowym [65].

W celu wyznaczenia prawdopodobieństwa dla całego zdania *P(W)* można skorzystać z zasady łańcuchowego prawdopodobieństwa (ang. *Chain Rule of Probability*), matematycznie zdefiniowaną następująco:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{6}$$

Po przekształceniach wzór na prawdopodobieństwo wszystkich elementów będzie wyglądał następująco:

$$P(x_1, x_2, x_3, ..., x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)...(Px_n|x_1, ..., x_{n-1}) \tag{7}$$

Oznacza on, że prawdopodobieństwo całego wyrażenia P(W) można zapisać w następujący sposób:

$$P(w_1, w_2, ..., w_n) = \prod P(w_i|w_1, w_2, ..., w_{i-1}) \tag{8}$$

Łańcuchy Markowa

Okazuje się jednak, że nie jest to zbyt efektywna metoda, gdyż istnieje zbyt wiele wszystkich możliwych zdań, które należałoby kiedykolwiek oszacować. Nie ma też możliwości uzyskania wystarczającej ilości danych, aby wyznaczyć wszystkie liczby wszystkich możliwych zdań dla danego języka. Zamiast tego zastosowane zostało uproszczone założenie, zwane założeniem Markowa - od nazwiska pomysłodawcy Andrieja Markowa. Założenie to można zdefiniować następująco:

$$P(w_1, w_2, ..., w_n) \approx \prod P(w_i|w_{i-k}, w_{i-k+1}, ..., w_{i-1}) \tag{9}$$

W praktyce sprowadza się ono do zdefiniowania prawdopodobieństwa jedynie dla kilku słów, a nie dla całego zdania. Najprostszy przypadek modelu Markowa nazwany został unigramem. W modelu tym prawdopodobieństwo całej sekwencji słów szacowane jest na podstawie prawdopodobieństw jedynie poszczególnych słów (ang. *unigrams*) [66].

Nieco bardziej inteligentnym modelem jest model dwuskładnikowy, tzw. *bigram*. W modelu tym podczas określania prawdopodobieństwa całej sekwencji słów używamy jednego słowa

poprzedzającego. Oczywiście można rozszerzyć model n-gramowy na trigramy, czyli 3-gramy, 4-gramy itp.

Ogólnie można przyjąć, że modelowanie języka za pomocą n-gramów jest niewystarczające. Powodem tego może być fakt, że w wielu zdaniach zależności pomiędzy słowami mają duże odległości. Przykładem może być zdanie „Komputer, który właśnie umieściłem w serwerowni na piątym piętrze, wczoraj wieczorem się zepsuł". W zdaniu tym słowa „zepsuł się" odnoszą się do słowa „komputer". Odległość pomiędzy tymi słowami w zdaniu jest duża [56].

Okazuje się jednak, że w praktyce często możemy uniknąć modeli wielo-gramowych, ponieważ informacje lokalne, zwłaszcza przy zastosowaniu 3-gramów i 4-gramów, okazują się już wystarczające dla wielu przypadków.

Dla oszacowania prawdopodobieństwa bigramu można posłużyć się wzorem:

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \tag{10}$$

Wynika z niego, że prawdopodobieństwo słowa w(i) jest wyznaczane, biorąc jedynie pod uwagę poprzednie słowo w(i-1). W pierwszej kolejności zliczane jest, ile razy słowo w(i-1) oraz słowo w(i) występują razem. Następnie wynik ten dzielony jest przez liczbę wszystkich wystąpień słowa w(i-1).

W Internecie można znaleźć wiele baz zawierających n-gramy. Jednym z przykładów może być baza Google N-Grams (https://books.google.com/ngrams). Zbiory danych zawarte w tej bazie zostały wygenerowane w lipcu 2009 r. (wersja 1) oraz w lipcu 2012 r. (wersja 2). Google przetworzyło 1 024 908 226 229 słów tekstu z czego otrzymano 1 176 466 663 sekwencji 5-gramów, które pojawiają się co najmniej 40 razy. Istnieje również 13 588 391 unikalnych słów, które występują więcej niż 200 razy [67]. Również na portalu GitHub.com można otrzymać listę 10 000 najczęściej występujących słów w języku angielskim - https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt.

Każde narzędzie służące do przetwarzania języka naturalnego musi zostać ocenione w perspektywie poprawności działania. Tak samo dzieje się z modelami językowymi. Ogólnie rzecz biorąc, można uznać, że dobry model językowy to taki, który jest lepszy/trafniejszy w znajdowaniu dobrych zdań i ich przewidywaniu. Dokładniej oznacza to, że modele językowe są lepsze, gdy przypisują wyższe prawdopodobieństwo dla prawdziwych lub też częściej występujących zdań niż dla zdań niegramatycznych lub takich, które są rzadziej obserwowane. Aby wykonać takie

sprawdzenie, powstałe modele są trenowane na podstawie zestawu treningowego (zestawu uczącego), a następnie testowana jest wydajność tak nauczonego modelu na danych, które jeszcze nie były dostępne (widoczne). Dane treningowe i testowe muszą być rozłącznymi zbiorami, dodatkowo dane testowe nie są prezentowane w trakcie tworzenia modelu. Zapewniając takie warunki, można stwierdzić, że model został rzetelnie oceniony.

Do oceny modelu mogą posłużyć rożnego rodzaju metryki. Efektywnym sposobem oceny modelu A i B jest dodanie ich do istniejącego systemu, np. korekty tekstu lub rozpoznawania mowy. Po uruchomieniu systemu otrzymamy pewną dokładność dla systemu działającego z modelem A, a także dla systemu z działającym modelem B. Metrykami, jakimi można sprawdzać te dwa modele, może być jak wiele błędnie napisanych słów były w stanie poprawić lub ile słów zostało przetłumaczonych poprawnie. W zależności od tego, który model ma większą dokładność, będzie to lepszy model językowy dla danego zadania. Podsumowując, lepszy model językowy, to taki, który przypisuje większe prawdopodobieństwo temu, co rzeczywiście się dzieje [40].

Podsumowując, N-gramy sprawdzają się tylko w przypadku przewidywania słów, jeśli korpus testowy wygląda bardzo podobnie do korpusu treningowego. Jeśli za korpus do nauki użyjemy zdań, które napisał Szekspir, a za korpus, na którym będziemy weryfikować model, użyjemy zdań napisanych w gazecie codziennej, to żaden model nie będzie w stanie działać efektywnie, przez co słowa nie będą poprawnie przewidywane.

Problemem, z którym należy sobie poradzić, są wyrażenia, które nigdy nie wystąpiły w korpusie podczas nauki modelu, a występują w zbiorze wyrazów testowych. Podczas tej sytuacji prawdopodobieństwo dla takich n-gramów będzie wynosiło zero. Należy więc zastosować pewne mechanizmy tak, aby z jednej strony dla rzadko występujących n-gramów prawdopodobieństwo było minimalne (ale nadal większe od zera). Z drugiej jednak strony otrzymany rozkład prawdopodobieństwa powinien być poprawny tak, aby po normalizacji wynosił dalej łącznie jeden [56].

Istnieje kilka metod radzenia sobie z takimi sytuacjami. Najprostszą metodą jest metoda wygładzania Laplace'a (ang. *Laplace Smoothing*), nazywana również metodą *Add One Estimation*. W metodzie tej zamiast stosować podstawowy wzór do liczenia prawdopodobieństwa:

$$P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \tag{11}$$

modyfikowany jest on w taki sposób, że do licznika dodawana jest jedynka, a do mianownika dodawana jest pewna stała V, która określa liczbę wszystkich słów w zbiorze. Wzór na wygładzanie Laplace'a wygląda w następujący sposób:

$$P_{Add\ -1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V} \tag{12}$$

Jak się okazuje metoda ta jest bardzo prosta, nie jest natomiast w praktyce często używana. Wygładzania Laplace'a stosowane są jedynie w problemie klasyfikacji tekstu, w których liczba bigramów nie jest duża. Metoda ta nie nadaje się w dużej mierze do poprawnego modelowania języka [65].

Lepszą metodą może być metoda nazwana wycofaniem lub odcięciem Katz'a (ang. *Katz backoff)*. Jak wiadomo posługiwanie się dłuższymi n-gramami daje lepsze wyniki. Niemniej, gdy w zbiorze uczącym nie ma wystarczającej ilości danych, Katz zauważył, że najpierw należy użyć dłuższych n-gramów, a następnie wykorzystać krótszy kontekst i posłużyć się krótszymi n-gramami do oszacowania prawdopodobieństwa.

Działanie tej metody opiera się na badaniu n-gramu (np. 3-gramu). Gdy dany n-gram występuje bardzo rzadko w tekście, to algorytm „wycofuje się" i sprawdza, czy n-1-gram dla danych słów nie pojawia się częściej. Jeśli n-1-gram pojawia się bardzo rzadko, można wrócić nawet do unigramu. Tak więc metoda ta, zgodnie z nazwą, zakłada możliwość wycofania się i skorzystania z mniejszego kontekstu, który natomiast pojawia się częściej.

Pokrewną metodą jest interpolacja. W interpolacji wykorzystywane są, w zależności od przypadku, zmieszane informacje pochodzące z trigramów, bigramów i unigramów. W praktyce okazuje się, że interpolacja daje lepsze wyniki niż metoda wycofania.

Liniowa interpolacja dla trigramów, bigramów i unigramów może wyrażać się wzorem:

$$\begin{aligned} P(w_n|w_{n-1}w_{n-2}) &= \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ &+ \lambda_2 P(w_n|w_{n-1}) \\ &+ \lambda_3 P(w_n) \end{aligned} \tag{13}$$

gdzie $\sum_i \lambda_i = 1$.

W celu wyznaczenia prawdopodobieństwa stosuje się prawdopodobieństwo wystąpienia wspomnianych wcześniej trzech n-gramów. Modyfikacją tego algorytmu jest możliwość

wykorzystania dodatkowo kontekstu słownego i w zależności od niego wyznaczenia parametrów lambda.

$$\begin{aligned} P(w_n|w_{n-1},w_{n-2}) &= \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-1},w_{n-2}) \\ &+ \lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) \\ &+ \lambda_3(w_{n-2}^{n-1})P(w_n) \end{aligned} \tag{14}$$

W zmodyfikowanej metodzie interpolacji, oprócz korpusów służących do nauki i do sprawdzenia poprawności algorytmu, wykorzystuje się trzeci korpus słowny służący do pierwszych testów modelu. Zasada działania wygląda tak, że po nauczeniu modelu wykorzystywany jest dodatkowy korpus, na którym sprawdzane jest nauczenie modelu i ustawiane są wspomniane wcześniej współczynniki lambda tak, aby prawdopodobieństwa były wyznaczane z jak najwyższą dokładnością. Dla tak nauczonego modelu, wzbogaconego dodatkowo przez sprawdzenie na dodatkowym korpusie testowym, można rozpocząć wyznaczanie prawdopodobieństw dla finalnego zestawu testowego.

Oczywiście to niejedyna możliwość poprawy wyznaczania prawdopodobieństwa. Kolejnym sposobem jest stosowanie w procesie nauki takich n-gramów, które w całym korpusie tekstów występują przynajmniej określoną liczbę razy. W takim przypadku odrzucane są korpusy niepewne (z małą ilością wystąpień).

Następnym sposobem zaproponowanym przez T. Brants et al. w publikacji [68] jest stosowanie algorytmu tzw. *Stupid Backoff*. Polega on na tym, że dla n-gramów o dużej częstotliwości wystąpienia stosuje się podstawowy estymator Maximum Likelihood (metodę największej wiarygodności), a w pozostałych przypadkach metodę wycofania (czyli wyznaczenie n-gramu niższego rzędu) z dodatkową wagą. Wzór opisanego algorytmu wygląda następująco:

$$S(w_i|w_{i-k1}^{i-1}) = \begin{cases} \frac{c(w_{i-k+1}^{i})}{c(w_{i-k1}^{i-1})} & \text{jesli } c(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i|w_{i-k+2}^{i-1}) & \text{w przeciwnym wypadku} \end{cases} \tag{15}$$

$$S(w_i) = \frac{c(w_i)}{N} \tag{16}$$

Niemniej najczęściej wykorzystywaną metodą jest metoda rozszerzonej interpolacji Kneser-Ney'a (ang. *Extended Interpolated Kneser-Ney*). W metodzie tej zauważono, że niektóre słowa, pomimo licznej łącznej sumy wystąpień w całym korpusie słownym, występują bardzo często jedynie w jasno określonym otoczeniu. Przykładem takich słów mogą być słowa Kong oraz

Francisco. Słowo Kong występuje zazwyczaj jedynie ze słowem King, tworząc nazwę postaci King Kong, a słowo Francisco ze słowem San, stanowiąc nazwę miasta w USA. Tak więc słowo Kong nie występuje często względem różnych kontekstów, które mogą przed nim istnieć. W odróżnieniu od tych słów w korpusie tekstu możemy znaleźć słowa, które pojawiają się rzadko, ale mogą wystąpić w wielu kontekstach słownych.

Prawdopodobieństwo kontynuacji jednego słowa słowem następnym będzie więc wyrażało się wzorem:

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1,w} > 0\}| \tag{17}$$

Ogólny wzór rozszerzonej interpolacji Kneser-Ney'a dla bigramów można zapisać wobec tego w podanej niżej postaci:

$$P_{KN}(w|w_{i-1}) = \frac{max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i) \tag{18}$$

gdzie współczynnik lambda określa się wzorem:

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})}|\{w : c(w_{i-1}, w) > 0\}| \tag{19}$$

Podsumowując, metoda Kneser-Ney'a jest skuteczna, jeśli chodzi o eliminację zerowych prawdopodobieństw dla n-gramów wcześniej niewidzianych. Jest to również metoda bardzo często używana w rozpoznawaniu mowy i tłumaczeniu maszynowym.

Model wektorowy

Kolejnym sposobem modelowania języka jest próba przedstawienia tekstu w postaci wektora. Aby tego dokonać, najpierw musi zostać określony pewien słownik lub lista termów indeksujących. Term może oznaczać dowolne elementy tekstu, np. formy tekstowe czy wyrazowe. Zakłada się, że poszczególne termy są nieskorelowane i tworzą bazę pewnej przestrzeni wektorowej. W modelu wektorowym dokumenty ze zbioru, w którym dokonywane jest wyszukiwanie, są traktowane jako liniowe kombinacje termów. Jeżeli zostaną wprowadzone współczynniki dla takiej kombinacji, to otrzymany zostanie wektor, który reprezentuje dokument w wybranej przestrzeni. Współrzędne takiego wektora odpowiadają wadze, jaką nadaje się poszczególnym termom w dokumencie. Są one zazwyczaj proporcjonalne do liczby wystąpień

termu w tekście dokumentu. Jeżeli dla któregoś termu wartość jest równa zero, oznacza to, że term nie występuje w dokumencie lub jest dla niego bez znaczenia [46].

Istnieje wiele sposobów obliczania współrzędnych wektora odpowiadającego konkretnemu dokumentowi. Wszystkie podstawowe metody bazują na wykorzystaniu trzech współczynników, jakimi można opisać dokument oraz jego związek z innymi dokumentami. Należą do nich:

- częstotliwość termu (ang. *term frequency, tf*) - liczba wystąpień termu w dokumencie,
- częstotliwość w dokumentach (ang. *document frequency, df*) - liczba dokumentów w kolekcji zawierających dany term,
- częstotliwość w kolekcji (ang. *collection frequency, cf*) - liczba wszystkich wystąpień termu w kolekcji dokumentów.

Częstotliwość termu odnosi się do określonego termu i dokumentu, natomiast pozostałe dwa parametry określa się dla poszczególnych termów. Należy zauważyć, że parametry *df* i *cf* mogą być obliczone, tylko jeśli w ogóle istnieje kolekcja dokumentów.

Częstotliwość termu, czyli liczba jego wystąpień w dokumencie, wydaje się być dobrym wyznacznikiem ważności termu dla danego dokumentu. Im wyższa jest jej wartość, tym większa szansa, że term dobrze charakteryzuje zawartość dokumentu. Często zależność reprezentacji wektora od częstotliwości termu jest spłaszczana przez zastosowanie pierwiastka lub logarytmu. Drugi parametr, czyli liczba dokumentów, w których term występuje, odzwierciedla popularność danego wyrazu w kolekcji oraz jego specyfikę lub powszedniość, a co za tym idzie, wartość przy rozróżnianiu dokumentów. Dlatego często stosuje się go także do obliczania wag termów w dokumencie, a tym samym współrzędnych wektorów. Najczęściej spotykaną formą częstotliwości w dokumentach jest jej postać odwrotna (ang. *inverse document frequency, idf*).

Kombinacja przedstawionych dwóch parametrów, częstotliwości termu i odwrotnej częstotliwości w dokumentach, oznaczana *tf.idf* jest najchętniej stosowanym schematem ważenia termów.

Zatem wzór na wagę termów może wyglądać następująco:

$$\text{waga} = tf * log\frac{N}{df} \tag{20}$$

gdzie **N** oznacza liczbę wszystkich dokumentów w kolekcji.

Podstawowym zagadnieniem wyszukiwania w modelu wektorowym jest określenie „odległości” pomiędzy dokumentami. Zakłada się, że odległość ta jest miarą bliskości znaczeniowej

dwóch tekstów, a więc tego, czy opisują tą samą informację. Dzięki reprezentacji dokumentów jako wektorów, możliwe jest obliczanie odległości w oparciu o mocne podstawy matematyczne.

W modelu wektorowym podobieństwo dwóch wektorów określa się zazwyczaj poprzez miary związane z odległością tych wektorów w przestrzeni. Najczęściej stosowana jest miara cosinusowa. Opiera się ona na obliczeniu cosinusa kąta pomiędzy wektorami i może być przedstawiona w postaci wzoru:

$$cos(q,d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{21}$$

Miara cosinusowa to miara, w której wyższa wartość oznacza większe podobieństwo wektorów. Należy zauważyć, że jeżeli wektory są znormalizowane, to wyznaczenie miary cosinusowej sprowadza się do obliczenia iloczynu skalarnego wektorów. Miara ta jest popularna ze względu na takie własności jak: niezależność wartości od rozmiaru dokumentu, prostota i łatwość interpretacji [69].

Metody wykorzystujące sieci neuronowe

Najnowsze trendy wykorzystywane w systemach i aplikacjach do przetwarzania języka naturalnego w oparciu o głębokie uczenie się (ang. *Deep Learning*) zostały zaprezentowane m. in. w artykule przez Young'a i współpracowników [70]. Niektóre tematy obejmują:

- wzrost rozproszonych reprezentacji (np. *word2vec*),
- konwolucyjne, rekurencyjne i rekursywne sieci neuronowe (ang. *convolutional, recurrent, recursive neural networks*),
- aplikacje w uczeniu się ze wzmocnieniem (ang. *reinforcement learning*),
- niedawne postępy w nauce reprezentacji zdań bez nadzoru (ang. *unsupervised learning*),
- łączenie modeli głębokiego uczenia się ze strategiami zwiększania pamięci (ang. *deep learning*).

Przez długi czas większość metod stosowanych do badania problemów NLP wykorzystywała modele płytkiego uczenia się oraz czasochłonne, ręcznie wyznaczane cechy. Dzięki niedawnej popularności i powodzeniu osadzania wyrazów (ang. *word embeddings*) modele oparte na sieciach neuronowych osiągnęły coraz to lepsze wyniki w różnych zadaniach w

porównaniu z tradycyjnymi modelami uczenia maszynowego, takimi jak SVM lub regresja logistyczna [71].

Osadzanie słów - word embeddings

Wektory dystrybucyjne, zwane także osadzaniem słów, opierają się na tak zwanej hipotezie dystrybucyjnej - słowa pojawiające się w podobnym kontekście mają podobne znaczenie. Umieszczanie słów jest wstępnie wyszkolone na testowym korpusie tekstu, którego celem jest przewidzenie słowa na podstawie jego kontekstu, zazwyczaj przy użyciu płytkiej sieci neuronowej.

W przeszłości rozproszone reprezentacje były intensywnie wykorzystywane do badania różnych zadań NLP, ale zaczęły zyskiwać na popularności dopiero po wprowadzeniu ciągłego worka słów (ang. C*ontinuous Bag-Of-Words, CBOW*) i modeli nie n-gramowych. Uzyskanie popularności było możliwe, ponieważ pozwalały one na osadzanie wysokiej jakości wyrazów, a także dlatego, że mogłyby zostać użyte do semantycznej kompozycji.

Modele word2vec

Około 2013 roku Mikolav et al. w pracy [72] zaproponowali modele CBOW i nie n-gramowe, tzw. skip-gram.

CBOW jest neuronowym podejściem do konstruowania osadzania słów. Jego celem jest obliczenie prawdopodobieństwa warunkowego słowa docelowego, biorąc pod uwagę słowa występujące w zadanym kontekście dla podanego rozmiaru okna.

Model skip-gramu jest natomiast neuronowym podejściem do konstruowania osadzania słów, w którym celem jest przewidzenie słów dla otaczającego kontekstu (tj. prawdopodobieństwa warunkowego) z podaniem centralnego słowa docelowego. Dla obu modeli wymiar osadzania słów jest określony przez obliczenie (w sposób nienadzorowany) dokładności predykcji.

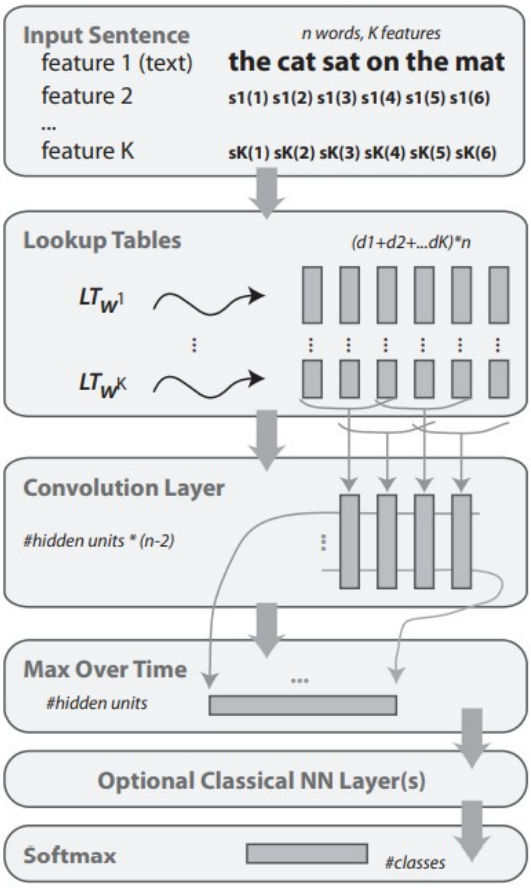
Innym ograniczeniem dla modeli word2vec jest to, że użycie mniejszych rozmiarów okien powoduje podobne osadzenie dla kontrastujących ze sobą słów, takich jak „dobry” i „zły”, co nie jest pożądane. Kolejnym zastrzeżeniem dotyczącym techniki osadzania słów jest to, że zależą one w dużej mierze od systemu, w którym są używane. Rozwiązaniem tego problemu może być ponowne nauczenie specyficznych osadzeń dla każdego nowego zadania. Zazwyczaj jednak operacja taka jest kosztowna obliczeniowo. Okazuje się również, że może być ona skuteczniej rozwiązana za pomocą np. próbkowania negatywnego.

Osadzanie liter - character embeddings

W przypadku zadań takich, jak oznaczanie części mowy (POS tagging) lub rozpoznawanie nazwanych jednostek (NER), przydatne okazuje się sprawdzanie informacji morfologicznych dla danego słowa. Ważna jest wobec tego sama kolejność znaków, jak również ich kombinacje. Jest to szczególnie pomocne w przypadku języków, które bogate są w morfologię (portugalski, hiszpański, chiński).

Konwolucyjna sieć neuronowa

Sieć ta (ang. *Convolutional Neural Network, CNN*) jest oparta jest na neuronach (ang. *neural-based approaches*), reprezentujących funkcję, która stosowana jest do tworzenia słów lub n-gramów w celu wydobycia cech wyższego poziomu. Powstałe w ten sposób abstrakcyjne cechy, zostały skutecznie wykorzystane między innymi do analizy sentymentów, tłumaczenia maszynowego i odpowiedzi na pytania. Collobert i Weston [73], jako jedni z pierwszych, zastosowali sieci CNN do zadań związanych z przetwarzaniem języka. Celem ich metody było przekształcenie słów w reprezentację wektorową za pomocą tabeli wyszukiwań (ang. *lookup table*), co zaowocowało prymitywnym podejściem do osadzania słów, które uczą się wag podczas samego treningu sieci. Podejście to zostało zobrazowane na rysunku 7.



Rysunek 7: Działanie konwolucyjnej sieci neuronowej (CNN) w zadaniu NLP [73]

Aby wykonać modelowanie zdań z siecią CNN, zdania są najpierw tokenizowane na słowa, które następnie są przekształcane w macierz osadzania słów, tj. w warstwę osadzania wejściowego. Następnie na wejściowej warstwie osadzania stosowane są filtry splotowe, które polegają na zastosowaniu filtra dla wszystkich możliwych rozmiarów okna tak, aby utworzyć tzw. mapę obiektów (ang. *feature map*). Następnie wykonywana jest operacja max-poolingu, która przeprowadzana jest na każdym filtrze w celu uzyskania wyjścia o stałej długości i zmniejszeniu danych wyjściowych. Procedura ta ostatecznie tworzy reprezentację zdania.

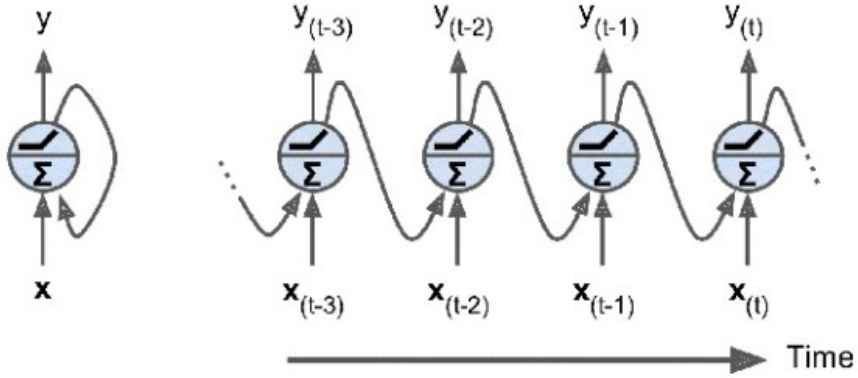
Jedną z podstawowych wad sieci CNN jest brak możliwości zamodelowania zależności dla dużych odległości, co jest ważne dla wielu zadań NLP. Rozwiązaniem tego problemu jest połączenie sieci CNN z opóźnionymi w czasie sieciami neuronowymi (*TDNN*), które umożliwiają skorzystanie z większego zakresu kontekstu podczas fazy treningu. Do innych typów sieci CNN, które okazały się skuteczne dla różnych zadań NLP, takich jak predykcja sentymentu i klasyfikacja typu pytania, można zaliczyć dynamiczną splotową sieć neuronową (*DCNN*). Sieć ta wykorzystuje

dynamiczną strategię łączenia k-max, w której filtry mogą dynamicznie dostosowywać zakresy zmiennych podczas przeprowadzania modelowania zdań.

Sieci CNN są również wykorzystywane do bardziej złożonych zadań, w których wykorzystywane są różne długości tekstów. Do zadań takich należy: analiza sentymentów, kategoryzacja tekstu i wykrywanie sarkazmu. Miejsca, w których sieci CNN okazały się przydatne, to między innymi dopasowywanie dokumentów do zapytań, rozpoznawanie mowy, tłumaczenie maszynowe (do pewnego stopnia) oraz reprezentacje pytań-odpowiedzi.

Rekurencyjna sieć neuronowa

Sieć ta (ang. *Recurrent Neural Network, RNN*) jest specjalistycznym podejściem opartym również na neuronach, które są skuteczne w przetwarzaniu sekwencyjnym informacji. Model ten stosuje rekurencyjnie obliczenia do każdej instancji sekwencji wejściowej uwarunkowanej wcześniejszymi obliczonymi wynikami. Sekwencje te są zazwyczaj reprezentowane przez wektor o ustalonej wielkości tokenów. Rysunek 8. ilustruje prosty model sieci RNN na osi czasu.



Rysunek 8: Model rekurencyjnej sieci neuronowej (RNN) na osi czasu [101]

Główną zaletą sieci takiego rodzaju jest zdolność do zapamiętywania wyników poprzednich obliczeń i wykorzystywanie tych informacji w bieżących obliczeniach. To sprawia, że sieci RNN nadają się do modelowania zależności kontekstowych na wejściach o dowolnej długości. Sieci te wykorzystywane są w zadaniach tłumaczenia maszynowego, do podpisywania obrazów oraz modelowania języka.

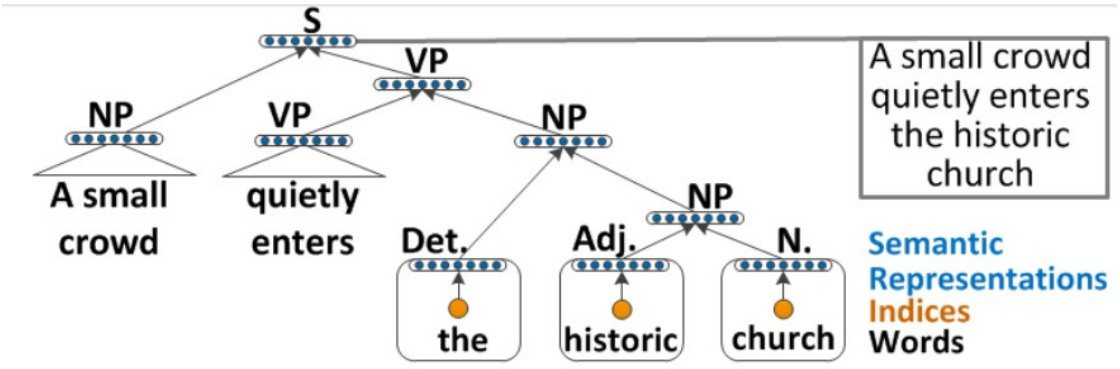
W porównaniu z modelem sieci CNN model sieci RNN jest podobnie skuteczny lub nawet nieco lepszy w określonych zadaniach. Dzieje się tak dlatego, że sieci te modelują różne aspekty danych, co tylko czyni je skutecznymi w zależności od zastosowania.

Innymi wariantami sieci RNN są sieci o długotrwałej pamięci krótkotrwałej (LSTM), sieci rezydualne (ResNets) i sieci okresowo zamknięte (GRU). Przykładowo sieć LSTM składa się z trzech bramek (wejściowych, wyjściowych i zapomnienia). Oblicza ona stan ukryty poprzez kombinację tych trzech bramek. GRU to sieci podobne do sieci LSTM, ale składają się tylko z dwóch bramek. Mimo to są bardziej wydajne, ponieważ posiadają mniejszą złożoność obliczeniową, szczególnie w trakcie uczenia.

Sieci RNN są używane w wielu aspektach NLP, takich jak:

- klasyfikacja na poziomie słów (np. NER),
- modelowanie języka,
- klasyfikacja na poziomie zdania (np. biegunowość sentymentu),
- dopasowywanie semantyczne (np. dopasowanie komunikatu do odpowiedzi kandydata w systemach dialogowych),
- generowanie języka naturalnego (np. tłumaczenie maszynowe, podpisywanie obrazów).

Podobnie jak sieć Recurrent Neural Networks, sieci Recursive Neural Networks są naturalnymi mechanizmami do modelowania danych sekwencyjnych. Dzieje się tak, ponieważ język można postrzegać jako strukturę, w której słowa i wyrażenia tworzą frazy wyższego poziomu. W takiej strukturze nieterminalny węzeł jest reprezentowany przez reprezentację wszystkich jego węzłów potomnych. Rysunek 9. ilustruje poniższą prostą rekursywną sieć neuronową (Recursive Neural Network).



Rysunek 9: Przykład Recursive Neural Network dla zadania NLP [102]

W podstawowej rekurencyjnej sieci neuronowej, funkcja kompozycyjna (tj. sieć) łączy składniki w podejściu oddolnym (bottom-up), aby obliczyć reprezentację wyrażeń wyższego rzędu. Inna odmiana tej sieci to rekurencyjna neuronowa sieć tensorowa (RNTN). Jej zaletą jest większa interakcja między wektorami wejściowymi, która sprzyja uniknięciu dużych parametrów. Rekurencyjne sieci neuronowe wykazują dużą elastyczność. Sieci te łączone są z sieciami LSTM w celu rozwiązania problemów, takich jak np. zanikanie gradientu.

Rekurencyjne sieci neuronowe są wykorzystywane do różnych celów, takich jak:

- rozbiór gramatyczny zdania,
- wykorzystanie reprezentacji na poziomie fraz do analizy sentymentów,
- klasyfikacja relacji semantycznych,
- powiązanie zdań.

Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem (ang. *Reinforcement Learning*) obejmuje metody uczenia maszynowego, w którym wykorzystywani są tzw. agenci do wykonywania dyskretnych działań, po których następuje etap nagrody.

Metoda ta składa się z agenta (model generatywny oparty na sieci RNN), który współdziała ze środowiskiem zewnętrznym (słowa wejściowe). Agent podejmuje akcję na podstawie polityki (parametrów), która obejmuje przewidywanie następnego słowa w każdej sekwencji. Następnie agent aktualizuje swój stan wewnętrzny (ukryte jednostki sieci RNN). Trwa to aż do końca sekwencji, w której ostatecznie obliczana jest nagroda. Funkcje nagród różnią się w zależności od zadania. Przykładowo w zadaniu generowania zdania nagrodą może być przepływ informacji.

Uczenie bez nadzoru

Nauka reprezentacji zdań bez nadzoru (ang. *Unsupervised Learning*) obejmuje mapowanie zdań na wektory o stałym rozmiarze w sposób nienadzorowany. Reprezentacje rozproszone wychwytują właściwości semantyczne oraz składniowe z języka i są szkolone za pomocą zadania posiłkowego. Model ten jest szkolony przy użyciu struktury seq2seq, w której dekoder generuje sekwencje docelowe, a koder jest postrzegany jako ogólny ekstraktor funkcji.

Głębokie modele generatywne

Głębokie modele generatywne (ang. *Deep Generative Models*), takie jak wariacyjne autoenkondery (ang. *Variational Autoencoder, VAE*) i generatywne sieci kontradyktoryjne (ang. *Generative Adversarial Network, GAN*), są również stosowane w NLP do odkrywania bogatej struktury w języku naturalnym poprzez proces generowania realistycznych zdań z ukrytej przestrzeni kodu.

Powszechnie wiadomo, że automatyczne autodeterminatory zdania nie generują realistycznych zdań ze względu na nieograniczoną przestrzeń ukrytą. Modele VAE nakładają wcześniejszą dystrybucję na ukrytą przestrzeń ukrytą, umożliwiając modelowi wygenerowanie odpowiednich próbek. Modele VAE składają się z sieci kodera i generatora, które kodują dane wejściowe w przestrzeni ukrytej, a następnie generują próbki z przestrzeni ukrytej.

Modele generatywne są przydatne w wielu zadaniach NLP i mają charakter elastyczny. Na przykład zaproponowano generatywny model VAE oparty na sieci RNN w celu uzyskania bardziej zróżnicowanych i dobrze uformowanych zdań w porównaniu ze standardowymi autoenkoderami. Inne modele umożliwiły włączenie zmiennych strukturalnych (np. czasu i sentymentu) do kodu ukrytego, aby wygenerować wiarygodne zdania.

Sieci GAN, złożone z dwóch konkurujących ze sobą sieci - generatora i dyskryminatora – wykorzystano również do wygenerowania realistycznego tekstu. W tym przykładzie sieć LSTM została użyta jako generator, a sieć CNN jako dyskryminator, który rozróżniał rzeczywiste dane od wygenerowanych próbek. Sieć CNN reprezentuje natomiast klasyfikator zdań binarnych. Model ten był w stanie wygenerować realistyczny tekst po odpowiednim szkoleniu.

Metody automatycznej predykcji sylab

Bardzo ciekawe metody łączą się szczególnie z automatycznym rozpoznawaniem mowy. Podstawowym wyzwaniem jest możliwość rozpoznania przez komputer w czasie rzeczywistym słów, jakie użytkownik wypowiada. Metody te muszą uwzględnić m. in.:

- występujące zakłócenia,
- zniekształcenia niektórych wypowiadanych słów,
- specyficzny akcent danej osoby.

W związku z wyżej wymienionymi przyczynami automatyczne rozpoznawanie mowy z wykorzystaniem najlepszych metod waha się od 90 % do 95 % dla różnych języków naturalnych. Koszty dzisiejszych systemów do automatycznego rozpoznawania mowy są bardzo duże. Zawierają one bowiem wiele złożonych mechanizmów do filtrowania oraz transformacji sygnału wejściowego, jakim jest głos (dźwięk) danej osoby. W procesie automatycznego rozpoznawania mowy wykorzystywane są mechanizmy, których zadaniem jest sprawdzanie uzyskanego wyrazu w specjalistycznym słowniku danego języka naturalnego oraz kontrola jego konstrukcji m. in. syntaktycznej oraz semantycznej. Dzięki tym metodom wybierane jest słowo, dla którego uzyskano najbardziej prawdopodobny rezultat. Nowatorską metodę, służącą automatycznemu rozpoznawaniu mowy, zaproponował w swojej pracy prof. Adrian Horzyk [74]. Metoda ta opiera się na analizie sylab w wyrazach oraz na możliwości przewidywania (predykcji) następnych sylab. Dzięki takiemu podejściu możliwe będzie dokończenie wyrazów, występujących w określonym kontekście zdania. W metodzie tej wykorzystywane jest posortowane drzewo, zawierające możliwe sylaby wyrazu w określonym kontekście, względem szans wystąpienia danej sylaby.

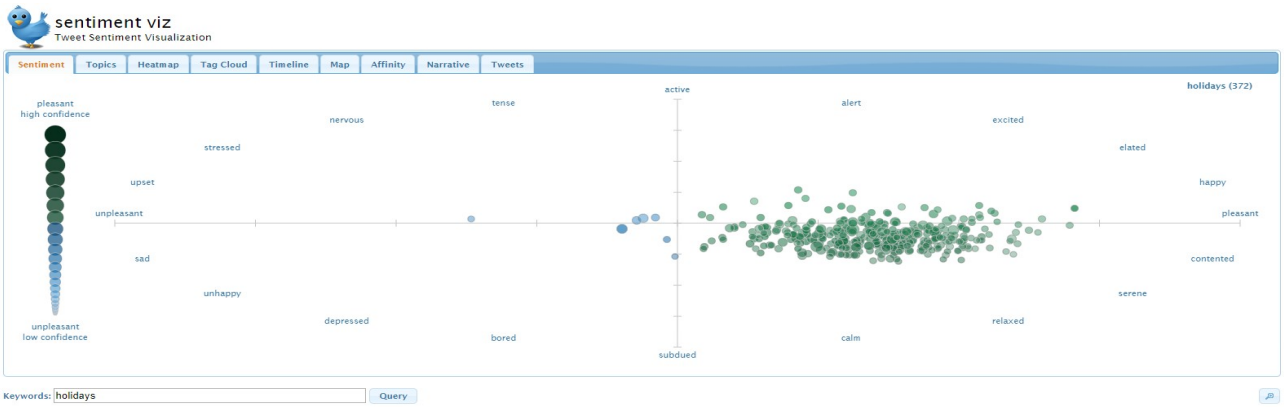
Pierwsze próby rozpoznawania słów w czasie rzeczywistym dla języka polskiego, uwzględniając metodę rozpoznawania sylab, miały skuteczność rzędu 76-92% [69].

Analiza sentymentu zdania

Jednym z ważnych zadań klasyfikacji tekstów jest określenie sentymentu (nastroju) zdania. W prostej formie może to być ocena, czy zdanie wyraża informację pozytywną czy negatywną.

Zautomatyzowanie procesu oceny treści zdania pod względem sentymentu przynosi wiele korzyści. Użytkownik zamiast analizować dziesiątki lub setki stron internetowych w poszukiwaniu informacji, np. o wybranym modelu aparatu fotograficznego, otrzyma skondensowaną informację o jego zaletach i wadach. Analizę sentymentu można również zastosować do rzeczy, które nie są produktami, na przykład do pomiaru zaufania konsumentów. Jednym ze źródeł danych, z którego korzystają aplikacje do analizy sentymentu, jest aktywność na portalu Twitter. Platforma ta pozwala w czasie rzeczywistym publikować krótkie informacje na temat aktualnych zdarzeń. Dzięki zastosowaniu w zdaniach słów kluczowych (tzw. hashtagów) można w łatwy sposób wyszukiwać konkretne informacje na zadany temat. Przykładem aplikacji, która analizuje sentyment zdania, może być aplikacja „Tweet Sentiment Visualization” (https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/). Na rysunku 10. przedstawiono

analizę sentymentu dla hashtagu „holidays”. Jak można zauważyć, w przeważającej większości zdania zawierające słowo w znaczeniu „wakacje” niosą pozytywne informacje.



Rysunek 10: Analiza sentymentu zdań ze słowem wakacje

Zadanie analizy sentymentu jest często spotykane pod różnymi nazwami:

- ekstrakcja lub eksploracja opinii,
- eksploracja sentymentu,
- analiza subiektywności.

Oznaczanie części mowy

Oznaczanie części mowy (ang. *Part-of-Speech tagging, POS-tagging, POS*) to jedno z klasycznych zadań w dziedzinie przetwarzania języka naturalnego. Czynność ta sprowadza się do przyporządkowania dla sekwencji tokenów (zdania) sekwencji etykiet. Owe oznaczenie jest wykorzystywane do budowania drzew dla analizowanych tekstów. W drzewach tych dla każdego wyrazu wchodzącego w skład zdania oznaczana jest część mowy. Na podstawie takiego zabiegu można wyodrębnić relację między słowami. Tagowanie POS jest również niezbędne do budowania lematyzatorów, które są używane do zredukowania słowa do jego formy podstawowej [65].

Czynność oznaczania części mowy dla wyrazów nie jest jednak zadaniem prostym, ponieważ konkretne słowo może występować w innej części mowy w zależności od kontekstu, w którym to słowo jest używane.

Przykładem może być wyraz „myśli”. W zdaniu „Marysia myśli o wyjeździe na wakacje w lipcu.” badany wyraz jest czasownikiem, natomiast ten sam wyraz w zdaniu „Różne myśli chodzą Adamowi po głowie.” jest rzeczownikiem.

W języku angielskim można spotkać bardzo wiele słów, które są zarówno rzeczownikami i czasownikami w zależności od kontekstu. Przykładowo:

- answer - może zostać przetłumaczone jako rzeczownik *odpowiedź* lub jako czynność *odpowiedzieć*,
- book - oznacza *książkę* lub czynność *zarezerwowania*,
- cook- rzeczownik *kucharz* lub czasownik *gotować*,
- sand - rzeczownik *piasek* lub czynność *wyszlifować*.

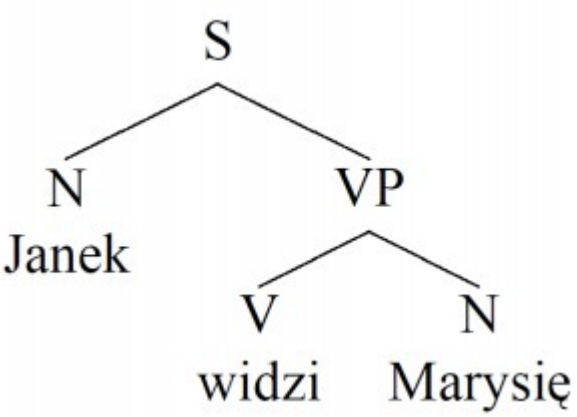
Istnieje kilka technik znakowania POS [75]:

- Metoda leksykalna (ang. *Lexical Based Methods*) - przydziela znacznik POS każdemu słowu występującemu w korpusie treningowym. W metodzie tej pomijana jest sekwencja wystąpień wyrazów w zdaniu;
- Metoda oparta na regułach - przypisuje znaczniki POS na podstawie reguł. Przykładowo może istnieć reguła, która mówi, że słowa w języku polskim kończące się na „ić” lub „ać”, oraz słowa w języku angielskim kończące się na „ing” mają być przypisane do czasownika. Techniki oparte na regułach mogą być stosowane wraz z podejściami leksykalnymi, aby umożliwić znakowanie słów, które nie są obecne w korpusie uczącym, ale znajdują się w danych testowych;
- Metoda probabilistyczna – przypisuje znaczniki POS na podstawie prawdopodobieństwa wystąpienia określonej sekwencji znaczników. Warunkowe pola losowe (ang. *Conditional Random Fields, CRF*) i ukryte modele Markowa (ang. *Hidden Markov Models, HMM*) są podejściami probabilistycznymi, służącymi do przypisania znacznika POS;
- Metoda głębokiego uczenia się (ang. *deep learning*) – użycie sieci neuronowych – np. Recurrent Neural Networks - w celu znakowania POS.

Zgodnie ze stroną https://universaldependencies.org/treebanks/pl-comparison.html [76] dla korpusów tekstów dla języka polskiego zostały wyróżnione następujące znaczniki POS:

- ADJ (przymiotnik, ang. *adjective*),
- ADP (przyimek, ang. *adposition*),
- ADV (przysłówek, ang. *adverb*),
- AUX(czasownik pomocniczy, ang. *auxiliary*),
- CCONJ (koordynowany spójnik, ang. *coordinating conjunction*),
- DET(wyznacznik, ang. *determiner*),
- INTJ (wykrzyknik, ang. *interjection*),
- NOUN (rzeczownik, ang. *noun*),
- NUM (numer, ang. *numeral*),
- PART (partykuła, ang. *particle*),
- PRON (zaimek, ang. *pronoun*),
- PROPN (nazwa własna, ang. *proper noun*),
- PUNCT (znak interpunkcyjny, ang. *punctuation*),
- SCONJ (spójnik podrzędny, ang. *subordinating conjunction*),
- SYM(symbol, ang. *symbol*),
- VERB(czasownik, ang. *verb*),
- X (inne, ang. *other*).

Dzięki podanym znacznikom można nadać etykiety dla każdego tokenu w zdaniu, tworząc drzewo, dzięki któremu zaprezentowana zostanie jego struktura. Przykładowo dla zdania „Janek widzi Marysię.” drzewo POS będzie wyglądało jak zaprezentowano to na rysunku 11.



Rysunek 11: POS tagging dla zdania w języku polskim

Ukryte modele Markowa

Łańcuchy Markowa bazują na założeniu, że aktualny stan zależy wyłącznie od stanu poprzedniego. Na łańcuchach tych bazują tzw. ukryte modele Markowa (ang. *Hidden Markov model, HMM*) [65], [66].

Ukryte Modele Markowa należą do grupy łańcuchów (sekwencji) stochastycznych Markowa. Cechą wyróżniającą te modele spośród innych łańcuchów Markowa jest to, że prawdopodobieństwo przejścia do następnego stanu, w dowolnej chwili, zależy tylko od stanu obecnego i nie zależy od historii. Na podstawie sekwencji obserwacji możemy dopiero odkryć, jakie stany ukryte wystąpiły w sekwencji [77].

Ukryty model Markowa można zastosować w procesie tagowania POS. Można określić, że stany ukryte odpowiadają częściom mowy, obserwacje to słowa tworzące strumień tekstu, a część mowy (tag) słowa zależy jedynie od części mowy słowa poprzedniego. W takim modelu można zauważyć, że słowo jest generowane jedynie przez swoją część mowy, co oznacza, iż jest zależne tylko od niej (a nie np. od poprzedniego słowa).

Rozpoznawanie nazw własnych

Wraz z rozpoznawaniem części mowy związane jest zagadnienie rozpoznawania nazw własnych (ang. *Named Entity Recognition, NER*). Zadanie to polega na stwierdzeniu, czy dane słowo lub ciąg wyrazów jest nazwą własną, a jeśli jest, to mechanizm przyporządkuje jej odpowiednią kategorię i ewentualnie podkategorię [78]. Przykładami przyporządkowania mogą być grupy:

- organizacja,
- osoba,
- lokalizacja,
- czas (data, godzina),
- ilość (liczba, kwota pieniężna, procent).

Nazwy własne bardzo często występują w dokumentach tekstowych. Poprawne ich rozpoznanie może mieć istotny wpływ na wynik dalszego przetwarzania w wielu zagadnieniach, np. w kategoryzacji dokumentów.

Najprostszą metodą rozpoznawania nazw własnych jest posłużenie się słownikami. Niemniej kilka z kategorii nazw własnych trudno byłoby umieścić w słowniku. Do takich nazw mogą należeć nazwy firm. Są one zbyt liczne i zbyt często dodawane są nowe. Poza tym mogą się one pojawiać w wielu różnych postaciach, np. w formie skróconej. Z tego powodu bezpośrednie stosowanie słownika może okazać się w wielu wypadkach niewystarczające.

Warto zauważyć, że w przypadku języków fleksyjnych, jakim jest język polski, większość nazw własnych podlega odmianie, w tym np. imiona i nazwiska. Co prawda w większości przypadków nazwa własna odmienia się tak jak rzeczownik o identycznej końcówce. Jednak dodatkowo utrudnia to prawidłowe rozpoznawanie i jest źródłem potencjalnych błędów przy znajdywaniu formy podstawowej [79].

Jedną z metod jest zastosowanie ukrytych modeli Markowa. Kolejnym przykładem może być model Maksymalnej Entropii Markowa (*MEMM*). Model ten jest bardzo podobny do ukrytych modeli Markowa. Łączy on cechy modeli HMM i maksymalnej entropii. Rozszerza on standardowy klasyfikator maksymalnej entropii, zakładając, że nieznane wartości, które mają być poznane, są połączone w łańcuchu Markowa, a nie warunkowo niezależne od siebie. Zaletą MEMM dla znakowania sekwencji jest to, że oferują one większą swobodę w wyborze funkcji do reprezentowania obserwacji. W sytuacjach znakowania sekwencji przydatne jest wykorzystanie wiedzy o domenie do projektowania funkcji specjalnego przeznaczenia. Modele maksymalnej entropii nie zakładają niezależności między cechami, ale generatywne modele obserwacji stosowane w modelach HMM owszem.

Kolejną zaletą MEMM w porównaniu do innych technik jest to, że szkolenie może być znacznie bardziej wydajne, gdyż oszacowanie parametrów rozkładów maksymalnej entropii

użytych dla prawdopodobieństw przejścia można wykonać dla każdego rozkładu przejścia w izolacji [80].

Innym przykładem algorytmów do rozpoznawania nazw własnych jest klasa metod nazwana warunkowymi polami losowymi (ang. *Conditional Random Fields, CRF)*. Jest to klasa metod modelowania statystycznego. Stosowana jest ona do rozpoznawania wzorców i w uczeniu maszynowym wykorzystywana jest do przewidywania strukturalnego. CRF są rodzajem dyskryminacyjnego, nieukierunkowanego probabilistycznego klasyfikatora. Kiedy dyskretny klasyfikator przewiduje etykietę dla pojedynczego wyrazu bez uwzględnienia wyrazów „sąsiadujących", CRF może brać pod uwagę kontekst, czyli otoczenie innych słów. Różnica między modelami dyskryminacyjnymi i generatywnymi polega na tym, że podczas gdy modele dyskryminacyjne próbują modelować rozkład prawdopodobieństwa warunkowego, to modele generatywne podejmują próbę modelowania wspólnego rozkładu prawdopodobieństwa. W CRF dane wejściowe są zestawem funkcji (liczb rzeczywistych). Pochodzą one z sekwencji wejściowej z wykorzystaniem wag, które powiązane są z cechami oraz z poprzednią etykietą. Zadaniem natomiast jest przewidzenie bieżącej etykiety. Wagi różnych funkcji będą określane w taki sposób, aby prawdopodobieństwo etykiet dla danych treningowych zostało zmaksymalizowane [81].

W CRF zdefiniowano zestaw funkcji, aby wyodrębnić odpowiednie etykiety dla każdego słowa w zdaniu. Niektóre przykłady funkcji to:

- Czy pierwsza litera słowa jest duża?
- Jaki jest przyrostek i przedrostek słowa?
- Jakie jest poprzednie słowo?
- Czy jest to pierwsze, czy ostatnie słowo zdania?
- Czy badanym słowem jest liczba?

Istnieje już wiele gotowych implementacji tej klasy rozwiązań, z których można skorzystać. Warto wymienić w tym miejscu biblioteki takie jak:

- CRF++ https://sourceforge.net/projects/crfpp/
- MALLET http://mallet.cs.umass.edu/
- GRMM http://mallet.cs.umass.edu/grmm/
- CRFSuite http://www.chokkan.org/software/crfsuite/

Rozdział 4

Graf jako struktura

Ze strukturą, jaką jest graf, spotykamy się w życiu codziennym. Jego przykład stanowić może mapa miasta. Wszystkie skrzyżowania w mieście oznaczyć można jako wierzchołki grafu, a drogi jako ich łączenia, czyli krawędzie grafu. Każda z dróg może być różnej długości, dlatego też krawędziom można przypisać odpowiednie wagi, które będą symbolizować liczbę kilometrów. Niektóre z dróg mogą być jednokierunkowe, a niektóre dwukierunkowe, stąd każdej krawędzi możliwy jest do przypisania kierunek. Z jednego skrzyżowania do drugiego można dostać się na więcej niż jeden sposobów, np. zamiast jechać najkrótszą, ale też najbardziej zakorkowaną drogą, można pojechać drogami dłuższymi, na których nie występuje takie natężenie ruchu. Dla grafu oznacza to, że może w nim istnieć pewien cykl [82].

Dzięki opisowi miasta i układu dróg został uzyskany opis grafu skierowanego. Ogólnie graf **G** składa się z niepustego zbioru wierzchołków, który oznaczany jest symbolem **V(G)** (ang. *verticies*) oraz zbioru krawędzi, oznaczonego symbolem **E(G)** (ang. *edges*). Liczba elementów zbioru V(G), czyli liczbę wierzchołków oznacza się zazwyczaj symbolem **n**, natomiast liczba elementów zbioru E(G), a więc liczbę krawędzi oznacza się symbolem **m**. Można wobec tego zapisać, że graf prosty wyraża się wzorem

$$G = (V(G), E(G)) \tag{22}$$

Grafem prostym nazywany jest graf G, składający się z niepustego i skończonego zbioru wierzchołków V(G) i skończonego zbioru krawędzi E(G), zawierającego różne, nieuporządkowane pary różnych elementów zbioru V(G). Oznacza to, że nie można poprowadzić krawędzi, która łączy dany wierzchołek z samym sobą. Dodatkowo, graf prosty może zawierać najwyżej jedną krawędź łączącą dane dwa wierzchołki [83].

Istnieje wiele rodzajów grafów. Gdy w grafie występują pętla, a więc krawędź łącząca wierzchołek z samym sobą oraz więcej niż jedna krawędź łączącą dwa wierzchołki (krawędzie wielokrotne), to struktura taka nazwana jest multigrafem. Gdy wierzchołki grafu przyjmują etykiety, czyli niosą pewną dodatkową informację w zależności od potrzeby modelu, to taki graf nosi nazwę grafu etykietowanego. Gdy graf zawiera wagi, można mówić o grafie ważonym. Wyróżnia się również grafy skierowane, a więc takie, w których krawędź prowadzi w określoną stronę, jak również grafy nieskierowane, gdy możliwe jest przejście pomiędzy wierzchołkami w

obydwie strony, a więc gdy krawędź (v_1, v_2) jest takim samym obiektem jak krawędź (v_2, v_1). Grafem spójnym nazywamy graf, w którym istnieje ścieżka (droga) łącząca dowolne dwa wierzchołki, natomiast graf niespójny charakteryzuje się tym, że istnieją wierzchołki, dla których takiej ścieżki (drogi) nie można wyznaczyć [84].

Za najstarszy przykład zastosowania grafów w rozwiązaniu postawionego problemu uznaje się zagadnienie mostów królewieckich. Opis tego problemu opublikował w 1736 roku Leonhard Euler. Był to rzeczywisty problem Królewca, przez który przepływała rzeka Pregoła. W jej rozwidleniach znajdowały się dwie wyspy, natomiast ponad rzeką zbudowanych było siedem mostów, z których jeden łączył obie wyspy, a pozostałe mosty łączyły wyspy z brzegami rzeki. Problemem, którym zainteresował się Euler, było udzielenie odpowiedzi na pytanie, czy można przejść kolejno przez wszystkie mosty tak, żeby każdy przekroczyć tylko raz. Euler wykazał, że jest to niemożliwe, a decyduje o tym nieparzysta liczba wylotów mostów zarówno na każdą z wysp, jak i na oba brzegi rzeki. Rozważył przy tym także ogólniejszy problem, starając się ustalić warunki, które muszą być spełnione, żeby dany graf spójny można było opisać linią ciągłą w taki sposób, by każda krawędź tego grafu była obwiedziona tylko raz. Euler pokazał, że jest to możliwe wtedy i tylko wtedy, gdy liczba wierzchołków tego grafu, w których spotyka się nieparzysta liczba krawędzi, wynosi 0 lub 2 [85]. Stąd też wziął się rodzaj grafów określanych mianem eulerowskich, a więc takich, w których możliwe jest skonstruowanie cyklu Eulera, czyli drogi, która przechodzi przez każdą jego krawędź dokładnie raz i wraca do punktu wyjściowego.

Teoria grafów to dział matematyki zajmujący się badaniem własności grafów. Zastosowanie tej teorii jest bardzo rozległe. Grafy pozwalają na przedstawienie informacji w zorganizowany sposób. Mogą zostać wykorzystane do modelowania informacji geograficznej: opisu map, planowania optymalnych dróg, obliczania odległości. Teoria grafów znajduje również swoje zastosowanie w wielu innych dziedzinach, takich jak fizyka, analiza i projektowanie oddziaływań społecznych czy bioinformatyka [86]. Za pomocą grafów można przedstawić sieć dróg i połączeń pomiędzy ważnymi punktami, dzięki czemu możliwe jest komputerowe znajdywanie najlepszej drogi z położenia początkowego do pożądanego celu. Algorytmy grafowe stanowią istotną część programów obsługujących urządzenia GPS. Przedstawienie w formie grafów sieci komputerowych pozwala na opracowanie oprogramowania usprawniającego trasowanie w Internecie. Za pomocą związanych z grafami pojęć można opisywać też m. in. rysunki obwodów, czy schematy blokowe, ponieważ przedstawiają one połączenia lub relacje, zachodzące między różnymi fragmentami wykresu [87].

Graf jako sposób zapisu zgromadzonych danych

Celem pracy jest m. in. opracowanie i wykorzystanie innowacyjnej metody do przechowywania wyrazów, uzyskanych na podstawie analizy korpusów tekstów tak, aby można było w możliwie łatwy sposób zapisać kluczowe informacje odnośnie samego wyrazu, jak i odnośnie kontekstu, w którym on występuje. Tak zapisane wyrazy wraz z informacją o kontekście, mają w dogodny sposób pozwalać na przygotowanie nowoczesnych algorytmów dla efektywnej analizy tekstu wprowadzonego z błędami, a następnie przeprowadzenia jego semi-automatycznej kontekstowej korekty.

Podstawowym zadaniem skutecznego mechanizmu automatycznej analizy korpusów tekstu, jest opracowanie poprawnej i skutecznej metody do ich przechowywania. Ważnym jest również, późniejsze wykorzystanie tej informacji w zależności od kontekstu. Istotnym aspektem jest także sama forma przechowywania informacji. Podczas pozyskiwania i przetwarzania tekstów duże znaczenie ma, aby powiązania występujące między słowami były automatycznie zapisywane tak, by w przyszłości można było z nich natychmiast skorzystać w zależności od kontekstu.

Chociaż komputery mają nieograniczony i możliwie bezpłatny dostęp do dużej ilości informacji (np. przez Internet), niestety nie są one jeszcze w stanie automatycznie wykorzystać tych danych do analizy lub rozumowania bez dodatkowych aplikacji. Z drugiej jednak strony, komputery są w stanie wykorzystywać różne informacje za pomocą dedykowanych programów i algorytmów, zaprojektowanych przez ludzi i ich inteligencję.

Inteligentne zachowanie wymaga znacznie więcej niż tylko przechowywanych zbiorów danych, które składają się na informacje. Zebrane dane nie tworzą wiedzy automatycznie, jeśli nie są aktywnie kojarzone z alternatywnymi danymi w ich kontekście. W tradycyjnych obliczeniach wciąż brakuje metod, które pozwalają na asocjacyjną reprezentację relacji danych, jak również na asocjacyjne obliczenia, które są naturalne dla ludzkiego mózgu. Tradycyjne obliczenia oddzielają reprezentację i przechowywanie informacji od ich późniejszego przetwarzania i eksploracji innych informacji. Co więcej, połączenia asocjacyjne w mózgu automatycznie wyzwalają różne przepływy danych na podstawie danych wejściowych, w zależności od kontekstu. Niestety bazy danych zawierają jedynie bardzo ograniczony pasywny zbiór relacji między danymi, a tylko algorytmy wyszukiwania mogą symulować niektóre procesy dedukcyjne. Tradycyjne algorytmy wciąż są dalekie od myślenia, tak jak robi to ludzki mózg w ułamku sekundy [88].

W związku z tą obserwacją podjęto próbę zamodelowania, a następnie zrealizowania nowatorskiej metody służącej do zapisu pojedynczych wyrazów wraz z kontekstem wypowiedzi

tak, aby możliwe było w późniejszym czasie jego odtworzenie. Pierwsze próby podjęte przez autora tej pracy wykorzystywały tradycyjną relacyjną bazę danych. W tabeli zamiast zapisu pojedynczych słów jako kolejnych rekordów zdecydowano się na zapis trójek wyrazów następujących po sobie. Dodano również informację o częstotliwości występowania dokładnie takiej trójki. Algorytm służący do zapisu tekstów posiadał więc następujące kroki:

1. Pierwszym etapem było podzielenie dostępnego tekstu na zdania.
2. Drugim krokiem było podzielenie wyodrębnionego zdania na tokeny. W etapie tym dokonano pewnych przekształceń teksów tak, aby z jednej strony dalej przechowywać pełny kontekst wypowiedzi, natomiast z drugiej strony ograniczyć liczbę wpisów w bazie danych. Przykładem takiego przekształcenia może być połączenie dla j. angielskiego przedimków, a więc wyrazów *a, an, the* ze słowem, którego one dotyczą.
3. Kolejnym krokiem było wprowadzenie pewnej normalizacji w tekście (ujednolicenie pisowni liter, usunięcie znaków specjalnych).
4. Po tak przeprowadzonych operacjach możliwe było zapisywanie do bazy danych trójek słownych. Jeśli zdanie składało się z dwóch lub jednego wyrazu, to zdania takie były pomijane.

Dla każdej trójki słów (wyraz poprzedzający, wyraz rozważany, wyraz następny) można było analizować jej najbliższy kontekst słowny z uwzględnieniem konkretnych form fleksyjnych wszystkich słów [89]:

- dla słowa poprzedzającego kontekstem są dwa wyrazy występujące po nim;
- dla słowa aktualnego kontekstem jest słowo poprzedzające je oraz słowo następne;
- dla słowa następnego kontekstem są dwa wyrazy występujące przed nim.

Jednym z głównych zadań podczas zbierania, a następnie zapisywania tekstów, było obserwowanie i analizowanie wybranej metody do przechowywania zebranych informacji. Jak można było się spodziewać, liczba rekordów w bazie danych rosła wraz ze wzrostem ilości przetworzonych korpusów tekstów. Interesujące zatem wydawało się sprawdzenie, czy przyrost liczby rekordów będzie miał charakter liniowy, czy będzie można od pewnego momentu zauważyć, że pewne konteksty słowne, w tym przypadku trójki słów, będą się powtarzać. W takim przypadku zamiast dodawać nową trójkę słowną, zwiększony zostanie licznik dla istniejącej. Maksymalną liczbę potrójnych kombinacji słów, które mogą wystąpić w danym języku, można oszacować jako:

$$\binom{liczbaWszystkichSlow}{3}$$

Niemniej rzeczywista liczba kombinacji sensownych potrójnych słów jest znacznie mniejsza niż maksymalna liczba trójek słów w każdym języku. Podczas badania wskazanej metody przechowywania informacji odnotowano dwa istotne fakty:

1. W przypadku najczęściej używanych sekwencji słów zmieniana była tylko częstotliwość pojawienia się danej trójki. Tym samym do bazy danych nie zostawał dodany nowy rekord, a jedynie zaktualizowana została kolumna odpowiadająca ilości wystąpień danej trójki słów.
2. Podczas czytania tekstów (szczególnie takich, które pochodziły z prywatnych, czy też nieoficjalnych stron internetowych – np. blogów) niektóre zdania były niepoprawne. W zdaniach tych użyte były błędne lub pospolite słowa, pojawiały się błędy ortograficzne itp. Zaobserwowano jednak, że w przypadku wystarczająco dużej liczby przeanalizowanych i zapisanych tekstów, można znaleźć wartość progową dotyczącą częstotliwości występowania, która będzie oddzielać wątpliwe trójki słów, których nie należy używać do korekty, od tych poprawnych.

Podczas badań zaobserwowano, że wzrost liczby trójek w czasie jest ograniczony i z czasem proces ich przyrostu spowalnia. Aby sprawdzić, jak szybko następuje przyrost, przeprowadzono kilka eksperymentów. Badania te oparto na analizie kontekstów słownych z książek, opowiadań i stron internetowych. Najważniejszym czynnikiem było zweryfikowanie tezy o występowaniu dwóch zakresów:

1. Kontekstu słów trójek, które pojawiły się tylko raz. Zgodnie z oczekiwaniami, procentowa liczba takich słów powinna się zmniejszyć w miarę czytania kolejnych tekstów.
2. Kontekstu słów trójek, które pojawiły się więcej niż dziesięć razy. W tym przypadku, procentowa liczba trójek słów powinna wzrosnąć.

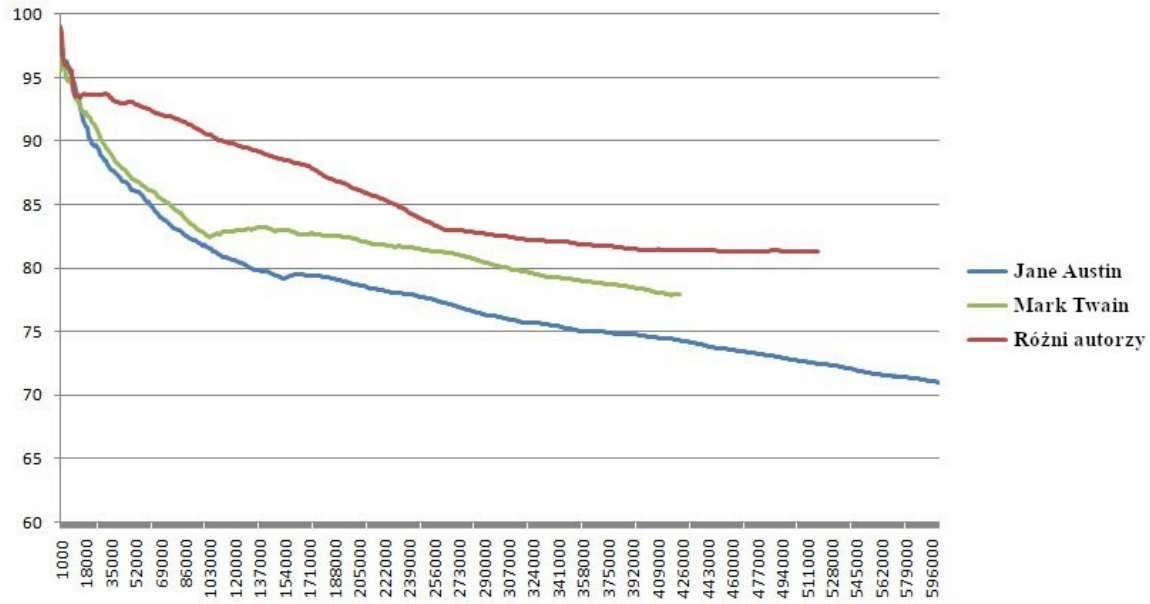
Pierwsze wyniki uzyskano, badając dzieła tego samego autora. Następnie przeprowadzono badania dla różnych tekstów, różnych autorów – zarówno dla języka polskiego jak i angielskiego. Celem tych prac była weryfikacja hipotezy, że przyrost liczby trójek słownych jest niezależny od języka. Zarówno dla języka polskiego jak i angielskiego zostały dostrzeżone te same właściwości.

Uzyskane wyniki potwierdziły wcześniej ustaloną hipotezę. Dla 45 różnych opowiadań różnych autorów uzyskano następujące wyniki:

- liczba potrójnych słów, które występują tylko raz, spadła do 78,24%;

- liczba potrójnych słów, które wystąpiły ponad dziesięć razy, wzrosła do 5,64%.

Należy zauważyć, że pozycje te zostały napisane przez różnych autorów, którzy używają swojego, indywidualnie określonego słownictwa, jak również niektórych słów i zwrotów częściej niż innych. Eksperymenty te wykazały, że można zaobserwować podobny trend wśród uzyskanych wyników. Wraz z czasem maleje procentowa liczba słów trójek występujących tylko raz. Taką tendencję zaprezentowano na rysunku 12.

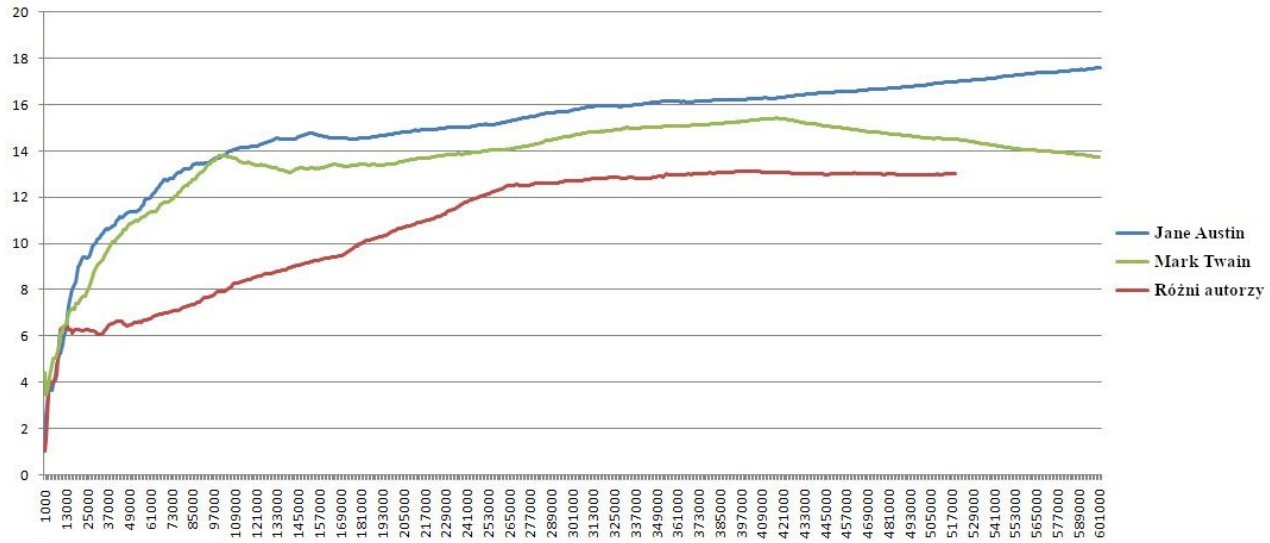


Rysunek 12: Procentowe występowanie pojedynczych słów trójek w zależności od liczby wyrazów.

Z drugiej strony procentowa liczba słów trójek, występujących więcej niż raz, rośnie w czasie. W celu dokładniejszej analizy zbiór takich trójek podzielono na kilka zakresów:

- trójka słowna pojawia się od dwóch do pięciu razy,
- trójka słowna pojawia się od sześciu do dziesięciu razy,
- trójka słowna pojawia się częściej niż dziesięć razy.

Po przeprowadzeniu eksperymentów stwierdzono, że najwolniejszy wzrost dotyczy trójki, która wystąpiła ponad dziesięć razy. Najszybszy wzrost kontekstu dotyczył potrójnych słów, które wystąpiły od dwóch do pięciu razy. Na rysunku 13. przedstawiono wzrost słów trójek, które wystąpiły od sześciu do dziesięciu razy.



Rysunek 13: Procentowe występowanie słów trójek w zakresie 6-10 razy w zależności od liczby wyrazów

Wyniki te wskazują na poprawne założenia dla zastosowanej metody. Z jednej strony zaproponowany sposób przechowywania danych zawiera informację o częstotliwości występowania słów w ich specyficznych formach fleksyjnych, z drugiej strony wykazano, że liczba słów trójek nie wzrasta liniowo w miarę pojawiania się nowych tekstów, a jest spowalniana poprzez zastosowania oznaczenia częstości występowania tych słów. Dodatkowo poprzez agregację najbliższych słów uzyskano możliwość odtworzenia najbliższego kontekstu.

Dzięki przeprowadzonym eksperymentom i wykazaniu słuszności zastosowanej metody, podjęto próbę wykonania jeszcze bardziej wydajnej metody zapisu i reprezentacji tekstów. W tym celu opracowany został nowy model ich reprezentacji. Głównymi potrzebami, dla jakich należało zbudować model, było:

- możliwość zapisu w wydajny sposób słów występujących w zdaniu;
- możliwość zapisu kontekstu słownego dla całej wypowiedzi;
- możliwość łatwego przeszukiwania modelu w celu odszukania określonego słowa;
- możliwość optymalnego zapisu słów, które już wcześniej wystąpiły;
- możliwość zapisu dodatkowych informacji o słowach, specyficznych zarówno dla samego słowa, jak również dla całego wyrażenia;
- możliwość odtworzenia zapisanych wcześniej kontekstów słownych;
- możliwość łatwej wizualizacji zebranych danych;

- możliwość opracowania algorytmów służących kontekstowej korekcie tekstu z wykorzystaniem powstałego modelu.

Zauważono, że modelem takim może być struktura grafowa. Jest to doskonały model do przechowywania słów i reprezentowania kontekstów ich użycia. Dla grafu można wprowadzić kilka rodzajów krawędzi (symbolizujących różne relacje), tak więc można w łatwy sposób zapisać różne konteksty słowne. Ponadto każdy z wierzchołków, jak i krawędzi może posiadać swoje unikalne właściwości, które posłużą do reprezentowania specyficznych informacji o słowach i wyrażeniach. Kolejną korzyścią z zastosowania takiego modelu, może być łatwość poruszania się w nim – dla każdego wierzchołka będzie istnieć skończona liczba krawędzi, po których będzie możliwe poruszanie się. Graf bądź jego wycinek można łatwo wizualizować. Dodatkowo jest to na tyle znana struktura, że na jej podstawie możliwe jest wykonanie algorytmów służących kontekstowej korekcie tekstów.

Graf Przyzwyczajeń Lingwistycznych

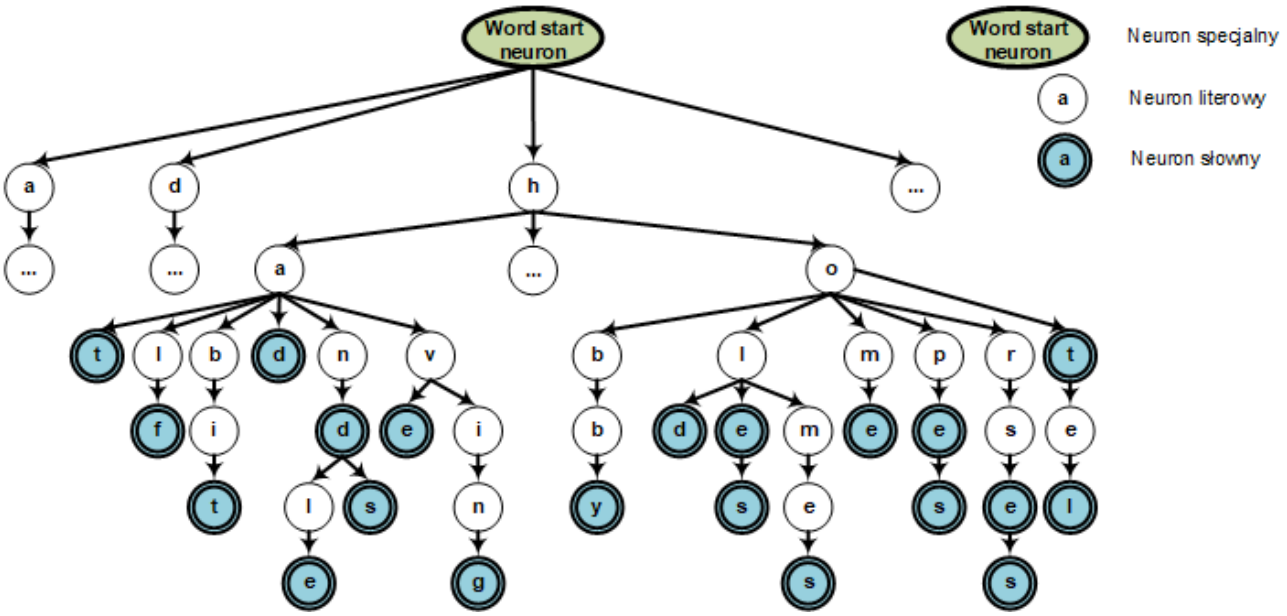
Zaproponowany przez autora grafowy model do reprezentacji słów wraz z kontekstami całych wypowiedzi został nazwany Grafem Przyzwyczajeń Lingwistycznych (ang. *Linguistic Habit Graph, LHG*). Inspiracją do jego powstania był model ludzkiego mózgu, który jest niezwykle efektywnym narzędziem do przetwarzania języka naturalnego. Jego neurony potrafią się aktywować wielokrotnie w ciągu jednej sekundy, rozwiązując niełatwe zadania. Średnio istnieje około kilku tysięcy połączeń dla każdego neuronu. W rzeczywistości utalentowani oraz uzdolnieni ludzie posiadają jeszcze więcej połączeń w niektórych obszarach mózgu. Oznacza to, że połączenia odgrywają istotną rolę w myśleniu, inteligencji, a także w przetwarzaniu danych. Połączenia te wydają się być nieprzypadkowe. Co więcej, odzwierciedlają one bardzo ważną relację między informacjami przechowywanymi w neuronach. Ogromna liczba połączeń w mózgu jest w stanie odtworzyć wiele powiązanych danych i ich różnych kombinacji, które reprezentowane są przez neurony. Mózg nie tylko zapamiętuje te relacje, ale także pozwala na automatyczną i autonomiczną aktywację danych w wybranych kontekstach zewnętrznych opartych na wiedzy. Tak więc dane przechowywane w mózgu są powiązane ze sobą nawzajem. Powiązania te można ponownie aktywować przy użyciu odpowiednich kontekstów, które pobudzają odpowiednie neurony. W biologii neurony kumulują sygnały w czasie i przestrzeni. Dzięki tej zdolności, nie tylko neurony bezpośrednio wcześniej pobudzone biorą udział w wzbudzaniu i aktywacji kolejnych neuronów, ale także wzbudzenie to następuje przez neurony, które zostały pobudzone jeszcze wcześniej. Ta bardzo ważna cecha została również wykorzystana podczas budowy modelu grafowego [90].

Zaproponowana grafowa struktura zapisu danych służy zgromadzeniu i zapisaniu informacji, pochodzących z wielu źródeł, operowaniu jak największą ilością słów w danym kontekście i generacji powiązań pomiędzy nimi. Konstrukcja takiego grafu pozwala w łatwy sposób zapisać i aktywnie powiązać między sobą słowa występujące w różnych kontekstach. Tak maksymalnie rozbudowany i maksymalnie ogólny graf, został nazwany Grafem Przyzwyczajeń Lingwistycznych, gdyż w łatwy sposób można w nim zapisać konteksty słowne zdań zapisanych przez ludzi. Niewątpliwymi zaletami takiego grafu skierowanego są: możliwość zastosowania go dla różnych języków, ciągłe uzupełnianie poprzez czytanie z kolejnych źródeł tekstowych oraz lokalne optymalizacje dla aktualnie badanego tekstu, w celu jego poprawy. Graf ma tę zaletę w porównaniu z innymi rozwiązaniami, iż liczba nowo dodanych wierzchołków będzie wraz z aktualizacją grafu o nowe teksty maleć, a nowe powiązania można modelować dodając jedynie krawędzie, różnego typu, pomiędzy kolejnymi wierzchołkami lub też aktualizując samą etykietę istniejącej już krawędzi i/lub wierzchołka.

Na podstawie badań i eksperymentów ostatecznie graf ten składa się z kilku rodzajów wierzchołków, jak również z kilku typów krawędzi:

- Wierzchołki symbolizujące poszczególne litery w wyrazie nazywane są neuronami literowymi (ang. *letter neurons*).
- Ostatni wierzchołek literowy dla danego wyrazu tworzy słowo. Został on więc dodatkowo wyróżniony i nazwany neuronem słownym (ang. *word neuron*). Dzięki temu, wyszukując w grafie jedynie neurony słowne, otrzymane zostaną wszystkie słowa. Dla każdego takiego neuronu dodana została informacja nie tylko o słowie, ale również o jego właściwościach.
- Wprowadzono również neurony specjalne (ang. *special neurons*), które wykorzystywane są przy zapisie całego zdania. Takimi neuronami są wierzchołki, które symbolizują początek słowa, początek zdania, jak również zakończenie zdania.

Każde słowo, zapisane w grafowej strukturze, reprezentowane jest przez połączoną sekwencję neuronów. Zaczyna się ono od neuronu specjalnego, określającego początek słowa, a następnie przechodzi przez połączone neurony literowe. Ostatni neuron oznaczany jest jako neuron słowny i dla każdego słowa występuje dokładnie jeden taki neuron w całym grafie. Rysunek 14. przedstawia bardzo mały wycinek grafu, w którym umieszczono jedynie informacje o pojedynczych słowach.



Rysunek 14: Wycinek Grafu LHG, w którym zapisano poszczególne słowa

Dzięki takiej formie zapisu słów w grafie, można je w jednoznaczny sposób wyszukać postępując techniką z „góry na dół” (ang. *top down)* lub techniką „z dołu do góry” (ang. *bottom up*). Jeśli potrzeba sprawdzić, czy dane słowo występuje w grafie, to proces ten rozpoczyna się od wyszukania neuronu specjalnego oznaczającego początek słowa. Następnie pod uwagę brana jest pierwsza litera słowa i sprawdzane jest, czy występuje połączenie (krawędź) od neuronu specjalnego do wybranego neuronu literowego, oznaczającego pierwszą literę słowa. Jeśli takie połączenie występuje, to sprawdzane jest występowanie kolejnego połączenia tym razem między pierwszą, a drugą literą słowa. Jeśli natomiast takie połączenie nie występuje, oznacza to, że takie słowo nie wystąpiło jeszcze w grafie i ewentualnie należy je dodać (całe słowo lub jego część). Technika wyszukiwania „z dołu do góry” sprawdza, jakie słowo reprezentowane jest przez wskazany neuron słowny. W tym celu w pierwszej kolejności należy wybrać neuron słowny, a następnie wyszukać neuron literowy, który prowadzi do wskazanego neuronu słownego. Tym sposobem słowo odtwarzane jest od końca do początku. Przykładem z rysunku 14. może być wybór neuronu słownego „e” dla słowa „home”:

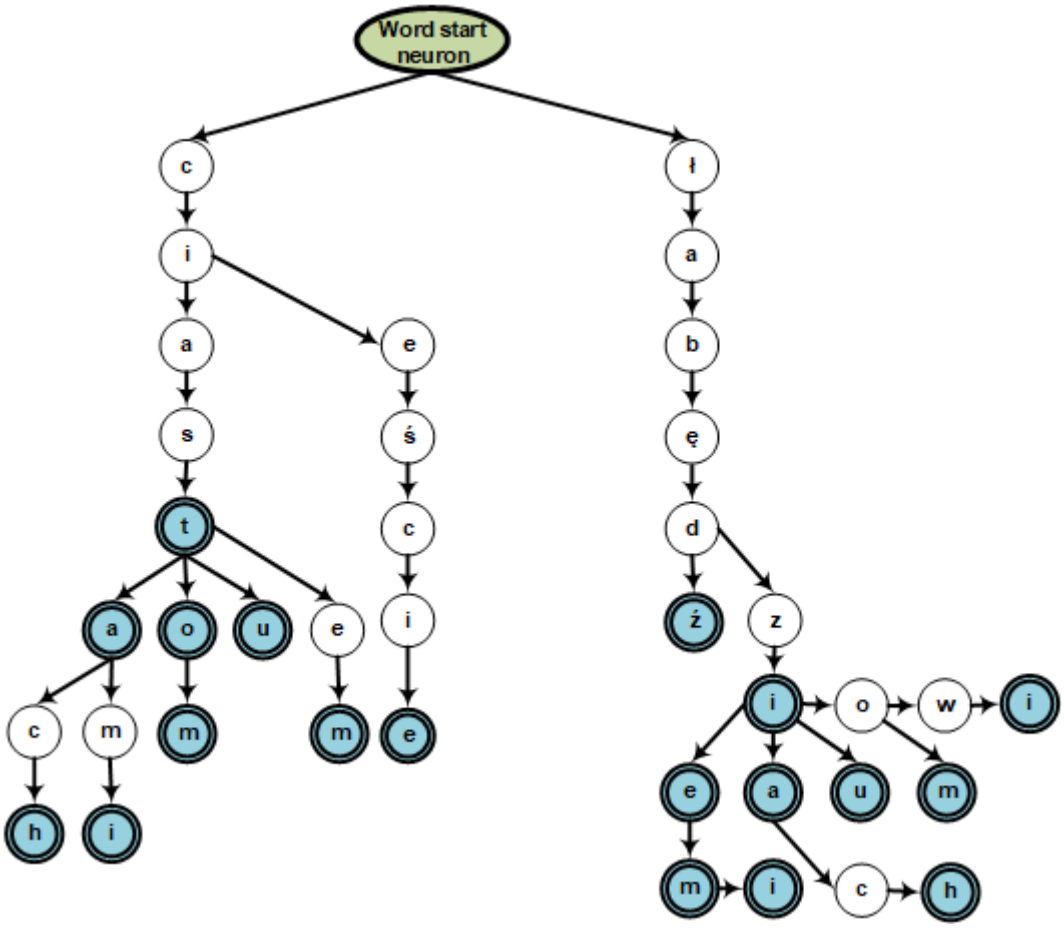
- dla wierzchołka „e” poprzednikiem jest neuron „m”,
- dla neuronu „m” poprzednikiem jest neuron „o”,
- dla neuronu „o” poprzednikiem jest neuron „h”,
- dla neuronu „h” poprzednikiem jest neuron specjalny „word start neuron”.

Wobec tego uzyskano ścieżkę e → m → o → h → word start, która po odwróceniu kolejności zaprezentuje szukane słowo „home”.

Dzięki takiej formie zapisu uzyskano również bardzo ciekawą właściwość, szczególnie dla słów w języku polskim, które podlegają odmianie przez przypadki. Każde słowo zapisywane jest oddzielnie w grafie, niemniej jednak taki sposób zapisu posiada dwie istotne zalety:

1. Łączna liczba wszystkich wierzchołków, potrzebnych do zapisu wszystkich słów, jest dużo mniejsza, niż liczba wszystkich liter we wszystkich słowach.
2. Może istnieć pewien wspólny rdzeń, dla zapisanych słów, a wszystkie formy fleksyjne (odmienione przez przypadki) zlokalizowane są względnie blisko w przestrzeni w tym grafie.

Na rysunku 15. przedstawiono fragment grafu LHG dla słów ciasto oraz łabędź wraz z ich odmianą. Jak można zauważyć dla takich „słów podobnych”, istnieje możliwość zgrupowania ich i nadania im takiej samej etykiety.



Rysunek 15: Zapis słów wraz z odmianą w grafie LHG

Jak zostało opisane wcześniej, neurony literowe posiadają dodatkowe opisy poprzez dodanie im pewnych cech. Przykładem takich cech może być informacja, czy słowo powinno zaczynać się od małej czy dużej litery, czy występuje przecinek dla danego słowa, a także czy słowo może kończyć się kropką, wykrzyknikiem lub znakiem zapytania. Każdy taki neuron zawiera także swój licznik (etykieta *word*), informujący o częstości występowania danego słowa, który jest aktualizowany podczas konstruowania grafu LHG dla nowych kontekstów.

Zauważono, że gdyby nie przeprowadzać na etapie analizy zdania takiego ujednolicenia, to w grafie występowałyby dodatkowe wierzchołki (np. znaki specjalne, takie jak kropka, przecinek, dwukropek, średnik, cudzysłów, nawias, itp.), które jedynie wprowadzałyby pewne zaburzenia do grafu. Oczywiście skonstruowany model grafowy byłby w stanie zapisać te znaki, jednak ponieważ występują one bardzo często (każde zdanie musi się kończyć pewnym znakiem – kropką, pytajnikiem lub wykrzyknikiem), to do tych i z tych wierzchołków występowałoby bardzo wiele krawędzi, które jedynie utrudniałyby przeszukiwanie grafu. Zdecydowano wobec tego zastąpić osobne wierzchołki odpowiednimi właściwościami, które zostają dopisane do słów występujących w bezpośrednim otoczeniu. Wobec tego każdy neuron reprezentujący słowo może zawierać dodatkowo etykiety takie jak:

- wyraz może rozpoczynać zdanie – oznaczone etykietą *startSentence,*
- wyraz może rozpoczynać się dużą literą - etykieta *startWithBigLetter,*
- wyraz może rozpoczynać się małą literą - etykieta *startWithSmallLetter,*
- wyraz może kończyć zdanie kropką - etykieta *endWithDot,*
- wyraz może kończyć zdanie wykrzyknikiem - etykieta *endWithExclamationMark,*
- wyraz może kończyć zdanie pytajnikiem - etykieta *endWithQuestionMark,*
- wyraz może być poprzedzony przecinkiem - etykieta *followedByComma,*
- po wyrazie może wystąpić przecinek - etykieta *mayEndWithComma,*
- po wyrazie może wystąpić średnik - etykieta *mayEndWithSemicolon,*
- po wyrazie może wystąpić dwukropek - etykieta *mayEndWithColon,*
- po wyrazie może wystąpić wielokropek - etykieta *mayEndWithEllipsis,*
- wyraz może być poprzedzony otwarciem nawiasu - etykieta *followedByStartingBracket,*
- po wyrazie może wystąpić otwarcie nawiasu - etykieta *mayEndWithStartingBracket,*

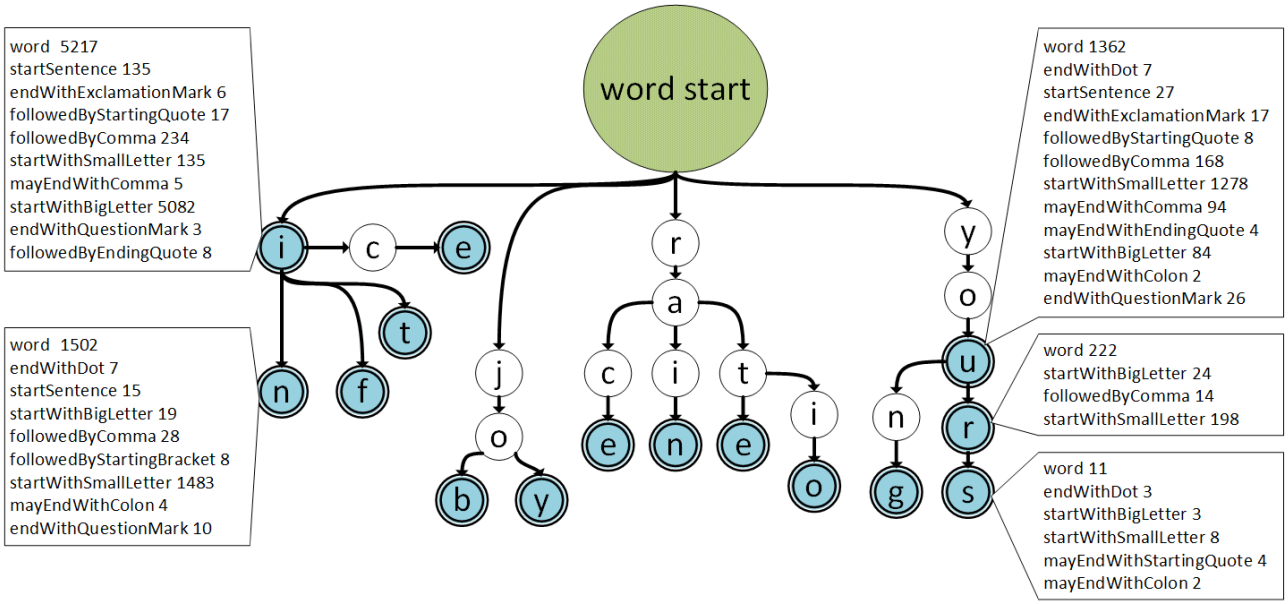
- wyraz może być poprzedzony zamknięciem nawiasu - etykieta *followedByEndingBracket,*
- po wyrazie może wystąpić zamknięcie nawiasu - etykieta *mayEndWithEndingBracket.*

Dodatkowo wyróżniono specyficzne właściwości dla słów w języku angielskim. Zauważono bowiem, że w języku tym występujące przedimki *a, an, the,* które również można przekształcić w podobne cechy dla słów. Wobec tego dla języka angielskiego wprowadzono dodatkowo etykiety:

- wyraz może być poprzedzony przedimkiem a – etykieta *followedByA,*
- wyraz może być poprzedzony przedimkiem an – etykieta *followedByAn,*
- wyraz może być poprzedzony przedimkiem the – etykieta *followedByThe.*

Dla każdej z tych etykiet przypisano również licznik, który jest zwiększany w przypadku zaobserwowania takiej samej cechy słowa, analizując kolejne korpusy tekstów. Dla każdego neuronu słownego może zostać przypisana dowolna etykieta. Nie wprowadzono bowiem mechanizmu wykluczeń etykiet.

Na rysunku 16. przedstawiono wycinek grafu LHG wraz z etykietami dla niektórych słów w języku angielskim.



Rysunek 16: Wycinek grafu LHG wraz z etykietami słów

Graf Przyzwyczajeń Lingwistycznych zawiera również inne typy połączeń. Do tej pory opisane krawędzie łączyły poszczególne litery w grafie, dzięki czemu można było odtworzyć zapisane słowo. Główną siłą grafu LHG jest wprowadzenie nowego asocjacyjnego sposobu przechowywania, kompresji i przetwarzania zdań. Wobec czego w tym grafie, występują również krawędzie reprezentujące m. in. połączenia pomiędzy poszczególnymi wyrazami w zdaniu. Rozróżnić można cztery typy połączeń:

1. Połączenie ASEQ (asocjacja sekwencyjna) – połączenie takie występuje jako połączenie wyrazów występujących w zdaniu, w naturalnej kolejności (jeden bezpośrednio po drugim).

2. Połączenia ACON (asocjacja kontekstowa „w przód") – aby jednoznacznie móc odczytać pełny kontekst zdań zapisanych w grafie, konieczne jest dodanie połączeń kontekstowych. Łączą one nie bezpośrednio poprzednie słowa z następnym w taki sposób, aby móc wskazać jednoznacznie następstwo kolejnych słów w zdaniu. Połączenia te dodawane są jedynie wtedy, gdy aktywacja poprzedniego neuronu słownego jest niejednoznaczna i aby określić, który neuron powinien zostać aktywowany jako następny w zadanym kontekście. Formują one więc kontekst dla danego słowa nie tylko w oparciu o jego bezpośredniego poprzednika. Dodatkowo dla połączeń ACON wprowadzone zostało rozróżnienie na poziomy (rzędy) połączeń. Połączeniami pierwszego rzędu są połączenia ASEQ, połączeniami drugiego i wyższych poziomów są połączenia ACON. Celem takiego rozróżnienia jest możliwość stwierdzenia, w jakiej odległości kontekstowej występują wybrane słowa od siebie nawzajem.

3. Połączenia ACON_PREV (asocjacja kontekstowa „w tył") - są to połączenia kontekstowe bardzo podobne do połączeń ACON (asocjacji kontekstowych „w przód"). Zostały one wprowadzone w celu wyeliminowania niejednoznaczności. Łączą one poprzednie słowa z następnymi w taki sposób, aby móc wskazać jednoznacznie poprzedzający kontekst słów w analizowanym zdaniu. Dzięki tym połączeniom wraz z połączeniami ASEQ możliwe jest wprowadzenie niezależnego sprawdzania kontekstu zdania „od końca do przodu".

4. Połączenia AMOR (asocjacja morfologiczna) – połączenie takie zostało wprowadzone dla tekstów w języku polskim, gdyż dla tego języka możliwe było skorzystanie ze słownika morfologicznego. Połączenia takie występują pomiędzy każdym słowem wprowadzanym do grafu, a jego formą podstawową (również obecną w grafie).

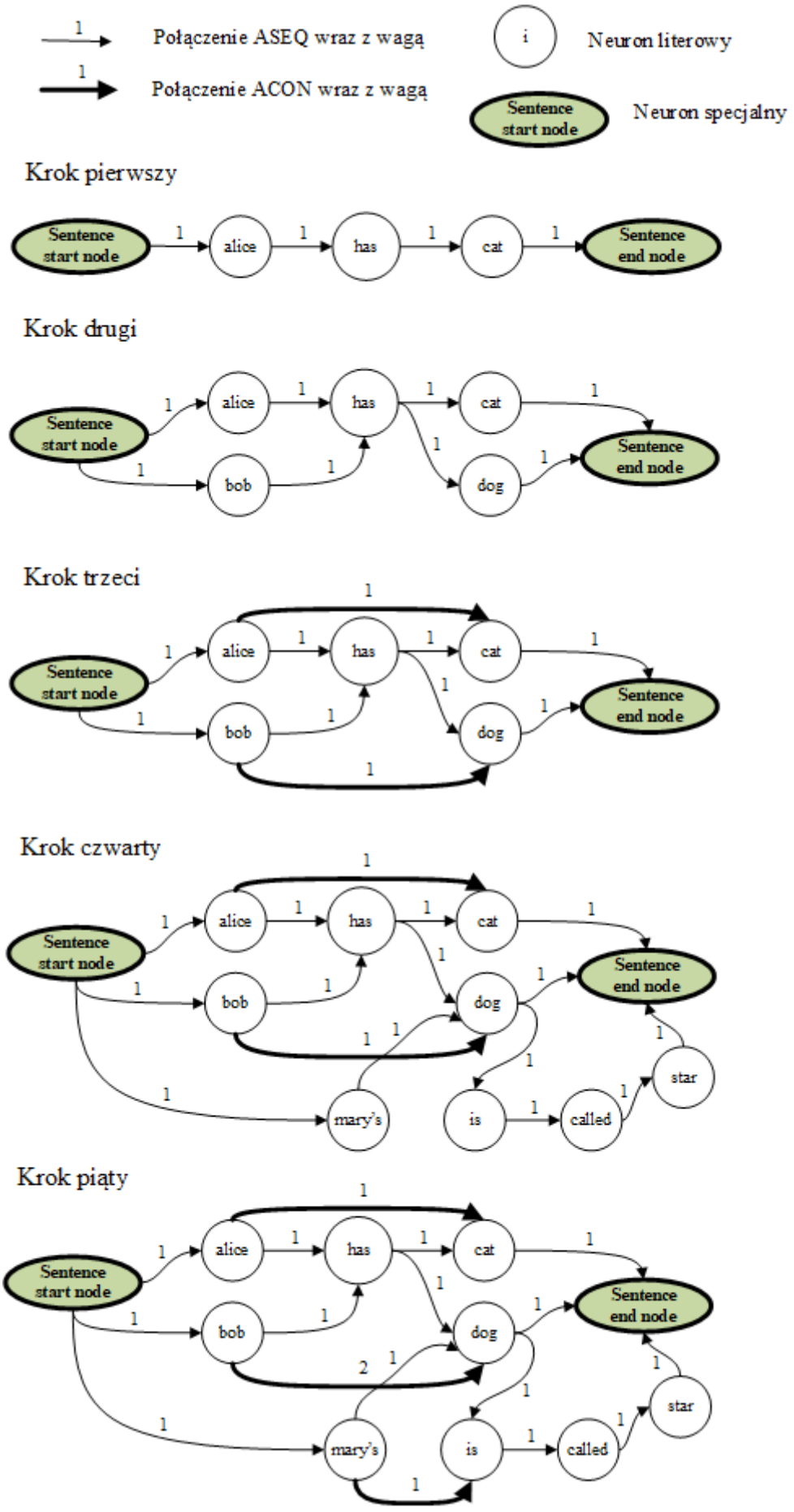
Każde wystąpienie przejścia pomiędzy połączeniami kontekstowymi odnotowywane jest poprzez wagę nadawaną takiemu połączeniu. Wobec czego, po przeczytaniu dużej ilości tekstów otrzymamy ścieżki, a więc konteksty, które są bardzo pospolite dla danego języka, dla danego

autora, dla danego zagadnienia (w zależności jakie teksty zostały dodane do grafu). Jest to kolejna zaleta wykorzystania takiego modelu do zapisu informacji o kontekstach słownych.

Na rysunku 17. przedstawiono kolejne kroki tworzenia połączeń asocjacyjnych „w przód" podczas dodawania następujących zdań do grafu:

- Alice has a cat.
- Bob has a dog.
- Mary's dog is called Star.

W celu większej przejrzystości na rysunkach tych przedstawione zostały tylko neurony specjalne oraz neurony słowne, symbolizujące pełne wyrazy (pominięto neurony literowe). Neurony te zostały zaprezentowane po przeprowadzeniu procesu ujednolicania (np. zamiana wszystkich liter w wyrazie na małe) oraz etykietowania (każdy neuron słowny zawiera dodatkowo zbiór właściwości).



Rysunek 17: Poszczególne kroki tworzenia asocjacji ASEQ oraz ACON w grafie

1. Krokiem pierwszym jest dodanie do pustego grafu zdania „Alice has a cat.”. Ponieważ graf jest pusty, a w dodawanym zdaniu nie występują powtórzenia słowne, to każdy wyraz łączony jest jedynie asocjacją sekwencyjną ASEQ. Warto zwrócić uwagę, że nie występuje w grafie wierzchołek dla słowa *„a”*. Jak zostało opisane wcześniej, przedimki tego typu zamieniane są na właściwości słowa stojącego bezpośrednio za nim, w tym wypadku będzie to dodatkowa właściwość słowa *„cat”*.

2. Drugim krokiem jest dodanie do grafu nowego zdania „Bob has a dog.”. Podczas dodawania odpowiednich neuronów słownych oraz krawędzi asocjacji sekwencyjnej można zwrócić uwagę, że dla słowa „has” pojawia się niejednoznaczność. Z neuronu tego wychodzą dwa połączenia sekwencyjne: jedno do słowa *cat*, a drugie do słowa *dog*. W związku z tym, bez dodatkowych asocjacji kontekstowych jednoznaczność kontekstu zostałaby utracona.

3. W kroku trzecim zatem po wykryciu niejednoznaczności kontekstowej następuje próba jej naprawy poprzez wprowadzenie połączeń kontekstowych ACON. Połączenia te mają za zadanie łączyć neurony słowne, poprzedzające niejednoznaczność, z neuronami słownymi występującymi za niejednoznacznością. W tym wypadku będą to dwa połączenia ACON pomiędzy dwójką słów „*alice*” i „*cat*” oraz pomiędzy „*bob*” a słowem „*dog*”. Po wprowadzeniu takiego uzupełnienia graf LHG nadal zawiera pełną informację odnośnie kontekstu dwóch różnych zdań.

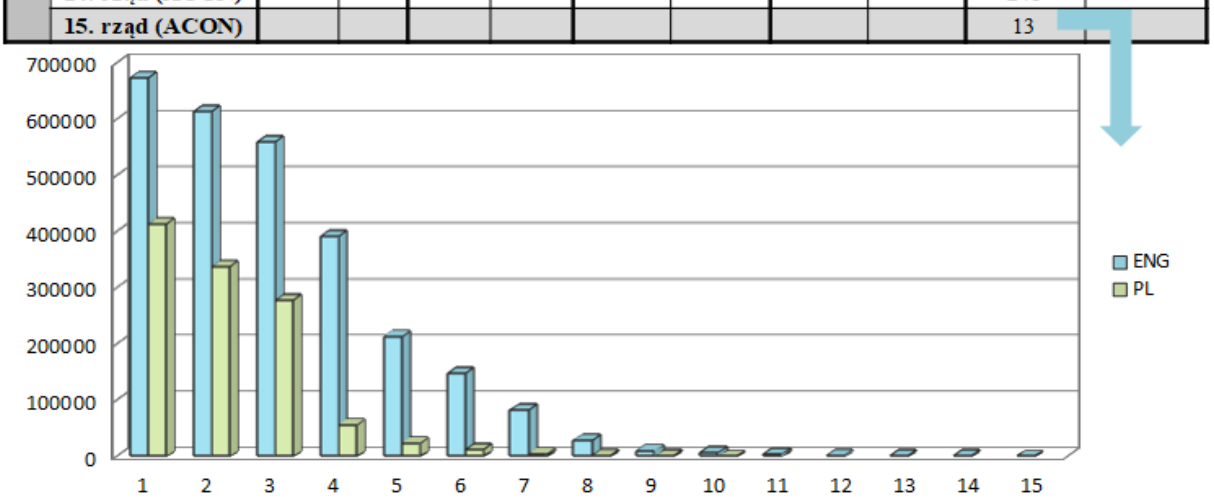
4. Następnie sprawdzane jest, czy nie występuje kolejna niejednoznaczność. Ponieważ w grafie nie wystąpił ten przypadek, można przystąpić do dodawania kolejnego zdania do grafu. W kroku tym do grafu dodane jest zdanie „Mary’s dog is called Star.”. Po dodaniu tego zdania sprawdzane jest, czy nie wystąpiła w grafie niejednoznaczność. Okazuje się, że dwuznaczność wystąpiła dla słowa „dog”. Ze słowa tego, w tym kroku, zostało dodane drugie połączenie ASEQ. W związku z tym należy, w następnym etapie, wprowadzić dodatkowe połączenia ACON.

5. W kroku piątym zostaje naprawiony kontekst dla zdań powodujących niejednoznaczność dla słowa „dog”. Podobnie jak w kroku 3. wyznaczane są słowa, którym należy dodać połączenia ACON, są to dwójki wyrazów „*bob*” i „*dog*” oraz „*mary’s*” i „*is*”. Jak można zauważyć, asocjacja kontekstowa dla pary „*bob*” i „*dog*” już istnieje. Wobec tego zwiększana jest jedynie waga tego połączenia. Dla pary „*mary’s*” oraz „*is*” tworzona jest nowa asocjacja ACON.

Zaprezentowany sposób analizy i zapisu korpusów tekstów, z wykorzystaniem pełnego kontekstu dla każdego ze zdań, okazał się poprawny. Mechanizm ujednolicenia kontekstu

wypowiedzi wykorzystuje połączenia asocjacji kontekstowych. W wyniku rozpoznania niejednoznaczności dla połączeń ASEQ bądź ACON, tworzone są połączenia ACON wyższego rzędu. W dalszej części pracy wykonano badania efektywności zapisu kontekstów słownych w grafie LHG. Sprawdzono, jak wiele krawędzi ACON i jakich rzędów tworzonych jest dla korpusów tekstu napisanego dla języka polskiego i angielskiego. Na rysunku 18. przedstawiono wyniki badania ilości asocjacji kontekstowych w zależności od liczby przeczytanych słów [91].

Ilość zdań		200		400		2000		4000		100000	
Język		ENG	PL	ENG	PL	ENG	PL	ENG	PL	ENG	PL
Ilość słów		2614	1391	11729	6689	25461	14633	50953	28948	961684	645858
Ilość liter		11452	7762	52459	35943	115564	78294	231917	152082	4288439	2514794
Ilość wierzchołków		941	804	2336	2994	3688	5634	5638	9564	64297	81348
Stopień kompresji:		0,4854	0,619	0,4941	0,6059	0,5079	0,597364	0,48056	0,583501	0,34828687	0,51949901
Rząd połączeń asocjacyjnych	1. rząd (ASEQ)	2423	1487	9289	6675	17969	13952	35184	26884	673338	413077
	2. rząd (ACON)	1579	531	7556	3132	16322	7550	35282	16694	613252	337528
	3. rząd (ACON)	894	132	6017	1321	13709	4040	28921	9397	558984	276919
	4. rząd (ACON)	56		1190	43	3707	346	8908	769	391357	56737
	5. rząd (ACON)			144		1124	2	3943	34	213736	24311
	6. rząd (ACON)			5		322		1787		147010	11565
	7. rząd (ACON)					28		672		81445	4973
	8. rząd (ACON)							129		28858	1413
	9. rząd (ACON)							3		10346	419
	10. rząd (ACON)									6242	71
	11. rząd (ACON)									3619	
	12. rząd (ACON)									1471	
	13. rząd (ACON)									619	
	14. rząd (ACON)									245	
	15. rząd (ACON)									13	



Rysunek 18: Rozkład ilości połączeń asocjacyjnych w zależności od ilości przetworzonych zdań

Jak można było oczekiwać, liczba połączeń asocjacyjnych dla wyższych rzędów zanika. Najwięcej jest połączeń pierwszego rzędu (ASEQ). Połączenia kontekstowe ACON dodawane są tylko wtedy, gdy kontekst dla poprzedniego neuronu słownego jest niejednoznaczny. Dzieje się tak,

ponieważ należy jednoznacznie określić, który neuron słowny powinien zostać aktywowany jako następny w kontekście poprzednio aktywowanego neuronu. W miarę dodawania kolejnych tekstów, występuje coraz więcej niejednoznaczności, tym samym konieczne są kolejne połączenia kontekstowe. Na rysunku 18. zostało również przedstawione bardzo ciekawe zestawienie dwóch różnych od siebie języków w kontekście występowania połączeń ACON. W języku polskim występuje odmiana słów przez przypadki. Najczęściej odmieniając słowo przez przypadki, otrzymujemy nowe słowa. Wobec czego będzie istniała mniejsza liczba ścieżek przez wybrane słowo niż dla języka angielskiego, w którym te słowa nie odmieniają się przez przypadki. Jako przykład można wskazać dwa zdania w języku polskim: „Marta czyta interesującą książkę. Ani ta książka również się podoba.”. W przykładzie tym, występują dwa różne słowa dla określenia książki – słowa „książkę” i „książka”. Ponieważ są to dwa odmienne słowa, nie wystąpi potrzeba dodania połączenia kontekstowego ACON. Natomiast te same dwa zdania w języku angielskim, będą brzmiały następująco: „Marta is reading an interesting book. Ania likes this book too.”. Jak można wskazać w zdaniach tych słowo książka występuje w tej samej formie „book”, wobec czego należy dla tego przykładu wprowadzić dodatkowe asocjacje kontekstowe ACON.

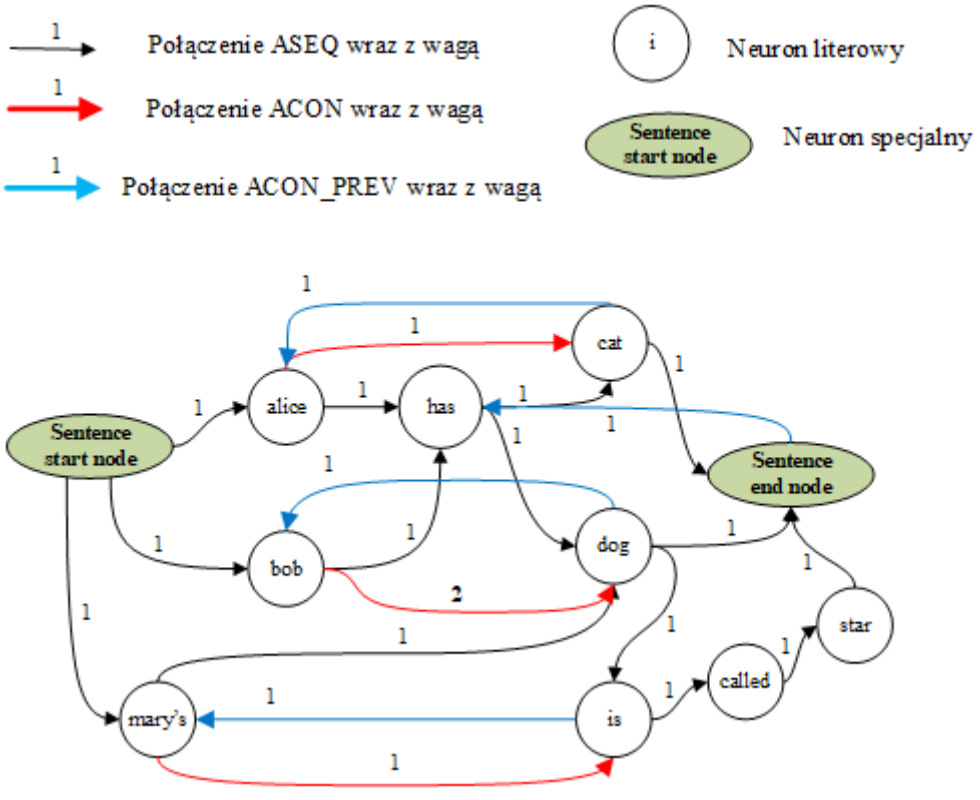
Kolejnym interesującym badaniem było sprawdzenie, czy w grafie wymagane jest przechowywanie pełnego kontekstu wypowiedzi, a więc połączeń ACON oraz ACON_PREV wszystkich potrzebnych rzędów. Początkowe badania uwzględniały jedynie połączenia kontekstowe „w przód”. Podczas prowadzonych eksperymentów zauważono, że nie jest możliwe odtworzenie pełnego kontekstu wypowiedzi we wszystkich przypadkach. Dzieje się tak m. in. wówczas, gdy niejednoznaczność występuje dla pierwszych wyrazów w zdaniu. Algorytm ich naprawy działa w ten sposób, iż łączy słowa poprzedzające wieloznaczność połączeniami ACON ze słowem występującym za niejednoznacznością. Gdy taka niejednoznaczność wystąpi na początku zdania, to nie jest możliwym wybór słowa poprzedzającego, ponieważ zdanie dopiero się rozpoczęło i takie słowo nie istnieje. Nie ma zatem dostępu do wcześniejszych słów. Dokonano więc usprawnienia, dzięki któremu wprowadzono dodatkowe połączenia kontekstowe „w tył” ACON_PREV. Aby wykryć moment konieczności utworzenia takiego połączenia, sprawdzana jest liczba połączeń tego samego typu, „wchodzących” do danego słowa. Gdy niejednoznaczność tak opisana powstanie, a więc więcej niż jedno połączenie danego typu będzie połączeniem dla danego słowa, konieczne jest dodanie połączenia kontekstowego „w tył”. Wówczas łączone jest słowo poprzedzające niejednoznaczność ze słowami występującymi za niejednoznacznością. Dzięki zastosowaniu połączeń kontekstowych zarówno ACON jak i ACON_PREV uzyskano interesujące możliwości:

- Możliwy jest zapis całego kontekstu wypowiedzi, zarówno w momencie gdy, niejednoznaczność kontekstowa wystąpi na początku, w środku lub na końcu zdania. W

przypadku wystąpienia niejednoznaczności w początkowych wyrazach zdania odtworzenie kontekstu możliwe jest dzięki połączeniom ACON_PREV. Gdy niejednoznaczność wystąpi w środku zdania, można zastosować niezależnie analizę połączeń ACON oraz ACON_PREV. Natomiast, gdy niejednoznaczność wystąpi w końcowych wyrazach, do odtworzenia kontekstu głównie zostaną wykorzystane połączenia ACON.

- Dzięki wprowadzeniu połączeń kontekstowych zarówno „w przód" jak i „w tył", możliwe zostało utworzenie oddzielnych algorytmów sprawdzających poprawność zdania. Pierwszy algorytm wykorzystuje połączenia ASEQ oraz ACON i bada zdanie „od początku do końca". Drugi analogiczny algorytm wykorzystuje połączenia ASEQ oraz ACON_PREV i niezależnie sprawdza wprowadzone zdanie „od końca do przodu".
- Ponieważ jednoznaczność kontekstu dla zdań oznaczana jest połączeniami ACON dwóch typów, możliwe jest zrezygnowanie z zapisu krawędzi wyższych rzędów. Po przeprowadzeniu badań okazało się, że połączenia szóstego i kolejnych rzędów występują tak rzadko, że można je pominąć.

Aby przedstawić konstrukcję grafu z wprowadzonymi połączeniami ACON_PREV na rysunku 19. przedstawiono ostateczny wygląd grafu LHG dla tych samych zdań, które zostały zaprezentowane wcześniej dla rysunku 17.



Rysunek 19: Graf LHG z połączeniami kontekstowymi ACON „w przód" oraz „w tył"

Podczas opracowywania połączeń asocjacyjnych w Grafie Przyzwyczajeń Lingwistycznych, a następnie metod korekty tekstu, inspirowano się rozwiązaniami spotykanymi w sieciach neuronowych. Rekurencyjne sieci neuronowe (RNN), w szczególności sieci LSTM lub sieci GRU są narzędziami, które często wykorzystywane są podczas prac związanych z rozpoznawaniem języka naturalnego. Zapewniają one dobre wyniki w wielu zadaniach NLP, w tym modelowaniu języka (LM), tłumaczeniu maszynowym, czy analizie nastrojów [92]. Inspiracją do powstania asocjacji kontekstowych ACON „w przód" oraz „w tył" były dwukierunkowe rekurencyjne sieci neuronowe (BRNN), które łączą dwie ukryte warstwy o przeciwnych kierunkach z tym samym wyjściem. W sieciach tych warstwa wyjściowa może jednocześnie uzyskiwać informacje z przeszłych (w tył) i przyszłych (w przód) stanów. W podobny sposób działają zaproponowane metody korekty tekstu. Błędy w zdaniu mogą występować zarówno na jego początku, jak i na końcu. Możliwe jest więc niezależne analizowanie zdania techniką „w przód", w celu wyznaczenia przyszłych słów na podstawie poprzednich, jak również dzięki technice „w tył", w celu wyznaczenia poprzednich słów, na postawie informacji o słowach następnych.

Niemniej sieci neuronowe posiadają pewne ograniczenia. Jedną z największych wad sieci RNN jest ich powolna nauka - szczególnie sieci LSTM na dużych zbiorach. Szybkość uczenia się można przyspieszyć na kilka sposobów, w tym przy użyciu wstępnie wyszkolonych modeli, przy użyciu szybszego algorytmu „softmax" lub stosując inną architekturę rozwiązania - splotowe sieci neuronowe (CNN), lub quasi-rekurencyjne sieci neuronowe (QRNN). W odróżnieniu od tych sieci, graf LHG konstruowany jest na bieżąco przez „czytanie" nowych zdań. Gdy w grafie wystąpi niejednoznaczność, należy przeanalizować i wrócić jedynie do tych zdań, które zostały już dodane do grafu, a w których wystąpiło szukane słowo. Nie jest konieczne analizowanie wszystkich wcześniejszych tekstów.

Podczas przewidywania następnego słowa w zdaniu wymagane jest sekwencyjne działanie algorytmu. Dla zdania „Ala ma kota", dla mechanizmu „w przód", najpierw przetworzone muszą zostać słowa „Ala" oraz „ma", a następnie słowo „kota". Tak więc, zaleta jaką jest działanie równoległe sieci neuronowych, nie będzie wykorzystywana. Jest to też jeden z powodów wolnego działania sieci RNN - każde słowo w ciągu wejściowym musi zostać przetworzone sekwencyjnie.

Dodatkowymi utrudnieniami związanymi z zastosowaniem sieci neuronowych jest ich zwykle niedeterministyczne ucznie się. Może się zdarzyć, że ta sama metoda szkolenia zastosowana do tego samego zestawu danych utworzy dwie różne sieci neuronowe. Sieci te będą mieć podobną wydajność, niemniej nie będą jednakowe. W odróżnieniu od sposobu budowy sieci neuronowych proces budowy grafu LHG jest deterministyczny. Posiadając zbiór danych uczących można

dokładnie określić jego konstrukcję. Graf taki możliwy jest również do zilustrowania w czytelny sposób w każdym momencie. To istotny aspekt, którego nie oferują sieci neuronowe, gdyż stosują one w swojej budowie warstwy ukryte.

Podobnie jak dla sieci neuronowych, w których informacje przechowywane są w całej sieci, tak połączenia pomiędzy kolejnymi neuronami słownymi przechowywane są w całym grafie. Zniknięcie lub usunięcie połączeń w jednym fragmencie grafu nie uniemożliwia funkcjonowania innych części grafu.

Podsumowując, zaimplementowany graf LHG jest bardzo skutecznym modelem do zapisu informacji o kontekście wypowiedzi, czyli o określonym porządku słów, mogącym wystąpić w zdaniach. Model ten jest rodzajem sieci neuronowej, która ma strukturę grafu. Składa się z neuronów (wierzchołków grafu), które reprezentują litery, całe wyrazy, jak również specjalne symbole oznaczające początek wyrazu, początek i koniec zdania. Struktura grafowa sieci neuronowej budowana jest dzięki kilkukrotnemu odczytowi korpusu tekstu. Wszystkie odczytane słowa są podzielone na osobne litery i przekształcone w neurony literowe, które są ze sobą połączone w sposób odzwierciedlający kolejność liter w tych słowach. Ta sama litera, może być reprezentowana przez kilka neuronów literowych, ale zawsze występuje w różnych kontekstach dla różnych słów. Każde słowo składające się z liter jest reprezentowane przez połączoną sekwencję neuronów literowych. Neuron literowy reprezentujący ostatnią literę w danym słowie jest również nazywany neuronem słownym. Neurony te powiązane są ze sobą na wiele sposobów, aby odzwierciedlić kolejność słów i kontekst wypowiedzi. Każde słowo jest reprezentowane dokładnie przez pojedynczy neuron słowny. Różne formy fleksyjne wyrazu są traktowane jako różne słowa i reprezentowane są poprzez różne neurony. Graf Przyzwyczajeń Lingwistycznych jest bardzo oszczędnym sposobem reprezentacji wielu słów. Dla słów, które posiadają ten sam korzeń, wystarczy dodać jedynie kilka neuronów literowych, aby utworzyć nowe słowo. Dużą zaletą tych grafów jest fakt, iż mogą być zupełnie automatycznie budowane w drodze „czytania" zdań z różnych korpusów tekstów. Im więcej tekstów graf LHG „przeczyta", tym lepiej jest w stanie wyrazić kontekst słów fleksyjnie oraz częstotliwościowo. Dzięki tej unikalnej właściwości, grafy LHG podczas ich stosowania w automatycznej korekcie tekstów są w stanie automatycznie eliminować różnego rodzaju błędy językowe z przetworzonych tekstów przy założeniu, iż błędy w korpusach tekstów będą występowały dużo rzadziej niż poprawne słowa.

Do automatycznej eliminacji błędów dochodzi na skutek ich rzadkiego występowania i niewielkiej wartości etykiety krawędzi. Można powiedzieć, iż błędy zostaną potraktowane jako swoiste artefakty słowne i ze względu na ich nieistotną częstotliwość nie będą brane pod uwagę

podczas korekty tekstów. Ponadto w ogólności grafy LHG mogą być też aktywnie czyszczone z krawędzi o niewielkich wartościach etykiet, co eliminuje możliwość błędnej akceptacji niepoprawnych językowo słów w kontekstach słownych, określonych w zbudowanym grafie LHG.

Błędy językowe

Głównym celem skonstruowania rozbudowanego, a tym samym bardzo ogólnego Grafu Przyzwyczajeń Lingwistycznych, jest opracowanie na jego podstawie innowacyjnych metod służących do korekty tekstu, zawierającego różnego rodzaju błędy. Model ten pozwala w łatwy sposób zapisać i aktywnie powiązać między sobą słowa występujące w różnych kontekstach. Na podstawie umiejscowienia tych słów w grafie, na podstawie unikalnych właściwości każdego ze słów oraz kontekstów, które tworzą, możliwe jest skuteczne zastosowanie grafu LHG do korekty tekstu.

Inteligentna semi-automatyczna korekta tekstu jest jednym z typowych, ważnych i praktycznych zadań obliczeniowych, które można osiągnąć wykorzystując nowatorskie metody, oparte na zaproponowanym rozwiązaniu. Struktura grafu jest w stanie zgromadzić nawyki językowe wielu osób, a następnie użyć ich do zdefiniowania składników języka, jak również do odtworzenia połączeń pomiędzy słowami w poprawnych kontekstach w sposób, w jaki ludzie robią to na co dzień. Do tego działania wykorzystywany jest fakt, iż neurony słowne mogą aktywować się wzajemnie. Co więcej, po przeprowadzeniu analizy, algorytmy mogą proponować najbardziej prawdopodobne kolejne słowa. Wszystko to służy do poprawy błędów we wprowadzonych tekstach.

Jak stwierdza A. Markowski [93], mianem błędu językowego obejmowane są zjawiska dość różnorodne, które odznaczają się odstępstwem od obowiązującej w danym momencie normy językowej, czyli takie innowacje, które nie znajdują uzasadnienia funkcjonalnego: nie usprawniają porozumiewania się, nie wyrażają nowych treści, nie przekazują na nowo, w inny sposób emocji nadawcy itd.

Błąd językowy można też określić jako taki sposób użycia jakiegoś elementu języka, który budzi sprzeciw jego świadomych użytkowników, gdyż pozostaje w sprzeczności z ich dotychczasowymi przyzwyczajeniami, a nie tłumaczy się funkcjonalnie. Witold Doroszewski we wstępie do Słownika poprawnej polszczyzny [94] napisał *„Wszelki błąd, a więc i błąd językowy, jest czymś niezamierzonym. Czynnością niezamierzoną jest czynność, której ktoś nie przewidział w*

myśli i nie pragnął, to znaczy czynność odbywająca się poza świadomością i poza aktywnym udziałem woli. Zwalczanie błędów to przeciwdziałanie irracjonalnym, inercyjnym (bezwładnym) skojarzeniom, będącym przyczyną wykolejeń we wszystkich dziedzinach języka".

Przytoczone powyżej przykłady zjawisk uznawanych za błędy językowe wskazują, że odstępstwa od normy, związane z posługiwaniem się językiem, są niejednorodne, należy je więc podzielić według kilku kryteriów na grupy bardziej spójne wewnętrznie.

Jednym z takich podziałów, może być dokonanie rozróżnienia na błędy zewnętrznojęzykowe i wewnętrznojęzykowe. Błędy zewnętrznojęzykowe to błędy zapisu - ortograficzne i interpunkcyjne. Są one związane z samą strukturą języka. W grupie tej można wyróżnić, np. zasady używania dużych i małych liter, których użycie na początku wyrazu jest motywowane względami znaczeniowymi lub składniowymi. Kolejnymi przykładami może być pisownia łączna bądź z łącznikiem, która może być uzasadniona budową słowotwórczą wyrazu. Stawianie znaków interpunkcyjnych ma natomiast swoje podstawy w składni. Wszystkie tego typu błędy zewnętrznojęzykowe nie naruszają jednak reguł systemowych czy też zasad rozwoju języka. Błędy wewnętrznojęzykowe natomiast bezpośrednio te reguły naruszają. Grupę tych błędów można dalej podzielić na błędy systemowe (językowe) i błędy użycia (stylistyczne). W obrębie błędów językowych mieszczą się błędy gramatyczne, leksykalne i fonetyczne.

Poniżej na podstawie [93] została umieszczona charakterystyka każdego z tych typów błędów i dokładniejsza ich klasyfikacja.

- Błędy wewnętrznojęzykowe – systemowe
  - Błędy gramatyczne
    - Błędy fleksyjne, które polegają na:
      - wyborze niewłaściwej postaci wyrazu, np. widnokręg zamiast widnokrąg, brzytew zam. brzytwa, ten kontrol zam. ta kontrola, wziąść zam. wziąć, szklanny zam. szklany, dokonywujący zam. dokonujący.
      - wyborze niewłaściwego wzorca odmiany, np. zaorać - zaoram zam. zaorze, tego zgorzela zam. tej zgorzeli, bardziej wysoki zam. wyższy.
      - wyborze niewłaściwej postaci tematu fleksyjnego, np. przyjacielom zam. przyjaciołom, o aniole zam. o aniele, kopła zam. kopnęła.
      - wyborze niewłaściwej końcówki fleksyjnej, np. diabłowi zam. diabłu, kapciów zam. kapci, znamieniowi zam. znamieniu, umią zam. umieją, bystszy zam. bystrzejszy.

- nieodmienianiu wyrazu, który ma swój wzorzec deklinacyjny, np. z Aleksandrem Fredro zam. z Aleksandrem Fredrą, jadę do Manchester, zam. jadę do Manchesteru.
- odmianie wyrazu, któremu nie można przypisać wzorca odmiany, np. wypić kubek kakaa zam. wypić kubek kakao.

- Błędy składniowe, które polegają na wyborze niewłaściwego wzorca składniowego, czyli niewłaściwym łączeniu form wyrazowych w jednostki tekstu. Mogą to być:
  - Błędy w zakresie związku zgody, np. do pokoju wszedł Adam i Grzegorz, zam. do pokoju weszli Adam i Grzegorz, pies i Zuzia wbiegły na podwórko zam. pies i Zuzia wbiegli na podwórko, przyszło czterdzieści trzy osoby zam. przyszły czterdzieści trzy osoby, rodzeństwo wyprowadzili się do Łodzi zam. rodzeństwo wyprowadziło się do Łodzi.
  - Błędy w zakresie związku rządu, np. używać dobrą szminkę zam. używać dobrej szminki, rozróżniać prawdę od fałszu, zam. rozróżniać prawdę i fałsz, brać się za malowanie pokoji zam. brać się do malowania pokoju, podobna nam zam. podobna do nas, mniejszy jak ty zam. mniejszy niż ty, mniejszy od ciebie.
  - Błędy w używaniu przyimków, np. przed i po obiedzie zam. przed obiadem i po obiedzie, warunki dla rozwoju eksportu zam. warunki do rozwoju eksportu, warunki rozwoju eksportu, jechać do Białorusi zam. jechać na Białoruś.
  - Błędy w zakresie używania wyrażeń przyimkowych, np. brak postępu w temacie kopalń zam. brak postępu w sprawie kopalń, pytania odnośnie reformy służby zdrowia zam. pytania dotyczące reformy służby zdrowia, odnośnie do reformy służby zdrowia.
  - Niepoprawne skróty składniowe, np. Organizuje i kieruje ruchem oporu na tych terenach zam. Organizuje ruch oporu na tych terenach i kieruje nim.
  - Niepoprawne konstrukcje z imiesłowowym równoważnikiem zdania, np. Zdając egzamin, został przyjęty na studia zam. Zdawszy egzamin, został przyjęty na studia, Po zdaniu egzaminu został przyjęty na studia, czytając takie reportaże, łzy napływają same do oczu zam. kiedy czyta się takie reportaże, łzy napływają same do oczu.
  - Konstrukcje niepoprawne pod względem szyku, np. wystąpienie kulturalnego attaché ambasady Francji zam. wystąpienie attaché kulturalnego ambasady Francji, wisząca groźba dyskwalifikacji nad zawodnikami zam. groźba

dyskwalifikacji wisząca nad zawodnikami, to w minionych latach nie zdarzało się zam. to w minionych latach się nie zdarzało.

- Zbędne zapożyczenia składniowe, np. założenia te wydają się być sensowne zam. Założenia te wydają się sensowne, wydarzyło się to nie wczoraj, a przed tygodniem zam. wydarzyło się to nie wczoraj, ale przed tygodniem.

- Błędy leksykalne
  - Błędy słownikowe (wyrazowe)
    - Używanie wyrazów w niewłaściwym znaczeniu, czyli zbędna neosemantyzacja, np. wnioskować jako „zgłaszać wniosek" choć właściwe znaczenie to „wysnuwać wniosek", „wyciągać wniosek", postument w znaczeniu pomnik, choć właściwe znaczenie słowa postument to „cokół", „podstawa pomnika", enigmatyczny jako synonim dla słowa „zdawkowy", „lapidarny", gdzie właściwe znaczenie tego słowa to „zagadkowy", „tajemniczy".
    - Mylenie znaczeń wyrazów podobnych brzmieniowo lub morfologicznie i ich niepoprawne wymienne używanie, np. klimatyzacja i aklimatyzacja, ewangelik i ewangelista, adaptować i adoptować, formować i formułować, monitować i monitorować, maksymalny i maksymalistyczny, efektowny i efektywny.
    - Posługiwanie się pleonazmami, np. całkowicie zlikwidować, cofnąć się do tyłu, akwen wodny, sopel lodu.
    - Naruszanie łączliwości wyrazu, np. „odnieść porażkę", gdy prawidłowo powinno być użyte sformułowanie „ponieść porażkę", „wyrządzać straty" zam. poprawnie: powodować straty.
    - Nadużywanie wyrazów modnych, takich jak asortyment, konsensus, opcja, pakiet, relacje, kreować, posiadać, serwować, adekwatny, poważny.
  - Błędy frazeologiczne
    - Zmiana formy frazeologizmów wskutek wymiany, redukcji lub uzupełnienia składu związku, np. ciężki orzech do zgryzienia, choć poprawnie należałoby stwierdzić „twardy orzech do zgryzienia", dolać oliwy bez dalszego członu, zam. „dolać oliwy do ognia".
    - Zmiana formy frazeologizmu wskutek zmiany postaci gramatycznej jednego ze składników, np. palić na panewce zam. spalić na panewce, szukać przy świecy zam. szukać ze świecą, plują sobie w brody zam. plują sobie w brodę.
    - Zmiana znaczenia frazeologizmu, np. „łowić ryby w mętnej wodzie" jako wyrażenie do określenia tłumaczenia się niejasno, nieprzekonująco. Poprawne

znaczenie tego frazeologizmu to „czerpać zyski z nieuczciwych, podejrzanych interesów”.

- Błędy słowotwórcze
  - Budowanie formacji niezgodnie z polskimi modelami słowotwórczymi, np. specgrupa zam. grupa specjalna, sport telegram zam. telegraficznie o sporcie.
  - Zastosowanie niewłaściwego formantu, np. głupość zam. głupota, babciowy zam. babciny.
  - Wybór niewłaściwej podstawy słowotwórczej, np. eurosejm, choć poprawnie powinno zostać użyte słowo europarlament. Sejm jest nazwą odnoszącą się jedynie do parlamentu polskiego oraz litewskiego.

- Błędy fonetyczne (w punkcie tym dla lepszego zobrazowania został zastosowany zapis fonetyczny).
  - Niepoprawna wymowa głosek, np. ą jako om na końcu wyrazu, np. [robotniką] zam. [robotnikom]), ś jako śi, np. [śiruba] zam. [śruba], przydechowe wymawianie głoski k, np. [k(h)oleiny].
  - Literowe odczytywanie wyrazów, np. [začęła] poprawnie: [začela].
  - Redukcja głosek i grup głoskowych, np. [tšea] poprawnie: [tšeba], pot. [čšeba] [kal'icja] poprawnie: [koal'icja].
  - Niepoprawne akcentowanie wyrazów i form wyrazowych, np. [at**mo**sfera], [robil'i**by**śmy], [pol'i**ty**ka), choć poprawnie słowa te powinny zostać odczytane z akcentem [atmo**sf**era], [ro**bi**l'ibyśmy]. [po**l**'ityka].

- Błędy wewnętrznojęzykowe – stylistyczne
  - Niewłaściwy dobór środków językowych w określonej wypowiedzi, niedostosowanie ich do charakteru i funkcji tej wypowiedzi.
    - Używanie elementów oficjalnych w wypowiedziach potocznych, np. Dokonałem zakupu maszynki do golenia, Skonsumowałeś na obiad cały makaron.
    - Używanie elementów potocznych w wypowiedzi o charakterze publicznym, np. Następnie zaobserwowałem, że człowiek ten wykopyrtnął się po wzięciu zakrętu, W wyniku obserwacji ustalono, że facet przemieszkiwał w wymienionym obiekcie.
    - Mieszanie elementów z kilku różnych stylów w jednej wypowiedzi. Wyróżnić można kilka stylów, np. styl wysoki, styl potoczny, styl urzędowy, styl książkowy, styl techniczny. Dobrą praktyką jest nie mieszanie ich w krótkich wyrażeniach.

        - Stylizacja językowa niemająca uzasadnienia w treści i charakterze stylowym wypowiedzi, np. Onegdaj nasi zawodnicy odbyli tylko jeden trening, Mamy tu duże utrudnienia, jako że samochody muszą jechać jedną stroną jezdni.
        - Naruszanie zasad jasności, prostoty i zwięzłości stylu.
- Błędy zewnętrznojęzykowe
    - Błędy ortograficzne
        - Używanie niewłaściwych liter i połączeń literowych w zapisie, np. ogurek, rzyczyć, druch, skoniczyć zam. ogórek, życzyć, druh, skończyć.
        - Niewłaściwa pisownia łączna lub rozdzielna, także niewłaściwe użycie łącznika, np. świeżomalowany, codzień, po środku, jasno żółty, le karz-dentysta zam. świeżo malowany, co dzień, pośrodku, jasnożółty, lekarz dentysta.
        - Niewłaściwe używanie dużych i małych liter na początku wyrazów, np. maria nowak, w ostatnią Sobotę, Sok z Czarnych Porzeczek.
    - Błędy interpunkcyjne
        - Brak właściwego znaku interpunkcyjnego, zwłaszcza przecinka.
        - Zbędne użycie znaku interpunkcyjnego.
        - Użycie niewłaściwego znaku interpunkcyjnego.

Innym kryterium podziału błędów językowych jest ich ranga, czyli to, w jakim stopniu naruszają normę językową. Z tego względu wszystkie błędy językowe można podzielić na rażące, pospolite i usterki językowe.

Błędy rażące naruszają podstawowe zasady poprawnościowe. Ich popełnienie powoduje zakłócenie podstawowej, komunikatywnej funkcji przekazu językowego. Tekst, który zawiera takie błędy, jest albo zupełnie niezrozumiały dla odbiorcy, albo przekazuje mu informacje niezgodnie z intencją nadawcy. W obu wypadkach prowadzi to do nieporozumień. Tekst pełen błędów rażących może nawet utrudniać zrozumienie treści, np. wskutek zawiłości czy rozwlekłości.

Błędy pospolite nie powodują na ogół nieporozumienia na poziomie semantycznym, jednakże, naruszając normę panującą w danym środowisku, narażają osobę, która je popełnia, na negatywną ocenę ze strony odbiorcy. Takim błędem jest np. brak odmiany nazwiska polskiego „Idę do pana Kościuszko” lub użycie błędnej formy, np. wzięłem, włanczać.

Usterki językowe to takie odstępstwa od normy, które naruszają ją tylko w niewielkim zakresie. Usterką będzie np. przestawny szyk zdania, użycie neologizmu w tekście o charakterze oficjalnym, który jeszcze jest niezakorzeniony w języku, posłużenie się formą przestarzałą w tekście potocznym albo wybór niewłaściwego kontekstowo wariantu leksykalnego.

Semi-automatyczne metody analizy i korekty tekstów

Graf Przyzwyczajeń Lingwistycznych może być szeroko stosowany do opracowywania metod sprawdzania pisowni i poprawiania tekstu dla głównych typów błędów wymienionych powyżej, za wyjątkiem błędów fonetycznych. Opracowany model został bowiem wykorzystany jedynie dla korekty tekstów pisanych, a nie do tekstów mówionych.

Opracowane i zaimplementowane przez autora metody korekty tekstu można podzielić ze względu na kilka cech, które posiadają. Pierwszego podziału można dokonać ze względu na czas, kiedy wykonywana jest analiza i korekta tekstu:

1. Pierwsza grupa zawiera więc metody, które mogą sprawdzać tekst podczas jego wprowadzania. Jeśli kontekst zdania jest niepoprawny, algorytmy korekty powinny natychmiast zaznaczyć wszystkie niepoprawne słowa i zaproponować poprawne ich zastąpienie.
2. Druga grupa metod szuka błędów dopiero po wprowadzeniu całego tekstu. Po wprowadzeniu zdań algorytmy powinny sprawdzić, czy wszystkie słowa są poprawnie umieszczone. Jeśli słowa występują w niewłaściwej kolejności, wykonane metody korekty tekstu powinny je znaleźć i zaznaczyć. Następnie podjęta powinna zostać próba ich automatycznego lub semi-automatycznego poprawienia poprzez zastąpienie nieprawidłowego słowa prawidłowym.

Kolejnym możliwym podziałem opracowanych algorytmów jest sposób ich działania. Tutaj również można podzielić je na dwie zasadnicze grupy:

1. Narzędzia służące statycznej analizie i korekcie tekstu.
2. Narzędzia wykorzystujące kontekst wypowiedzi, w celu przeprowadzenia dokładniejszej analizy i korekty tekstu.

Warto podkreślić, że algorytmy te stosowane są wspólnie. Tak więc raz wprowadzony tekst poddawany jest jednocześnie analizie statycznej jak i kontekstowej. Następnie wyniki analizy z każdej niezależnie działającej metody są zbierane i odpowiednio łączone w celu umożliwienia kontekstowej korekty tekstu. Gdy kilka algorytmów wykryje ten sam błąd i zaproponuje taką samą jego poprawę, wówczas wskazany fragment tekstu poprawiany jest automatycznie. Jeśli natomiast w wyniku działania algorytmów istnieje duże prawdopodobieństwo poprawy tekstu na kilka sposobów, wówczas użytkownikowi prezentowanych jest kilka najlepszych wyborów, a sam człowiek manualnie decyduje, która wersja poprawy powinna być wybrana.

Metody statyczne

Jak zostało wcześniej stwierdzone, podczas dodawania zdań do Grafu Przyzwyczajeń Lingwistycznych każdy neuron słowny zawiera informacje o słowie i jego właściwościach. Każdy z takich neuronów dla każdej cechy posiada własny licznik, który jest aktualizowany podczas konstruowania tego grafu. W tabeli 2. zostały zaprezentowane właściwości wyodrębnionych neuronów słownych po przeczytaniu zdania „Alice has a dog called Daisy!”.

Tabela 2: Tabela prezentująca właściwości neuronów słownych po przeczytaniu zdania

Neuron słowny	Właściwość neuronu
alice	*startASentence, startWithSmallLetter*
has	*startWithSmallLetter*
dog	*followedByA, startWithSmallLetter*
called	*startWithSmallLetter*
daisy	*startWithBigLetter, endWithExclamationMark*

Na podstawie właściwości opisanych powyżej opracowany został algorytm służący statycznej analizie, a następnie korekcji słów oraz interpunkcji w zdaniach. Głównym zadaniem tego algorytmu jest sprawdzenie, czy słowo powinno zaczynać się od małej czy dużej litery, czy przecinek jest obowiązkowy, a także czy zdanie zaczyna się od dużej litery, a kończy kropką, wykrzyknikiem lub znakiem zapytania. Mechanizm działania algorytmu opiera się na częstotliwości występowania pewnych właściwości danego słowa. Dla przykładu załóżmy, że słowo *alice* przytoczone powyżej reprezentowane jest przez zestaw poniższych cech wraz z odpowiednimi wagami:

- $word_{COUNTER}$ = 131 – waga określająca, ile razy słowo wystąpiło w kontekstach słownych. Oznacza to, że słowo „alice” wystąpiło 131 razy podczas tworzenia grafu LHG.
- $startSentence_{COUNTER}$ = 15 – waga określająca, ile razy słowo występowało jako pierwszy wyraz w zdaniu.
- $startWithSmallLetter_{COUNTER}$ = 17 – waga określająca, ile razy słowo zaczynało się od małej litery (gdy słowo znajduje się na początku zdania, zawsze jest oznaczone etykietami startWithSmallLetter oraz startSentence).
- $startWithBigLetterr_{COUNTER}$ = 114 – waga określająca, ile razy słowo zaczynało się od dużej litery.

Na podstawie powyższych właściwości algorytm korekcji interpunkcji może zakładać, że słowo *alice* zazwyczaj zaczyna się od dużej litery. Odbywa się to poprzez sprawdzenie nierówności opisanej wzorem poniżej.

$$\frac{\text{startWithBigLetter}_{COUNTER}}{(\text{word}_{COUNTER} - \text{startSentence}_{COUNTER})} \geq \text{bigLetterFactor} \tag{23}$$

We wzorze tym ujęto wagi następujących wystąpień:

- startWithBigLetterr$_{\text{COUNTER}}$ – liczba wystąpień słowa dużą literą,
- word$_{\text{COUNTER}}$ – ogólna liczba wystąpień słowa,
- startSentence$_{\text{COUNTER}}$ – liczba wystąpień słowa na początku zdania,
- bigLetterFactor – stała określona na podstawie eksperymentów, decydująca, czy wybrane słowo powinno rozpoczynać się dużą bądź małą literą.

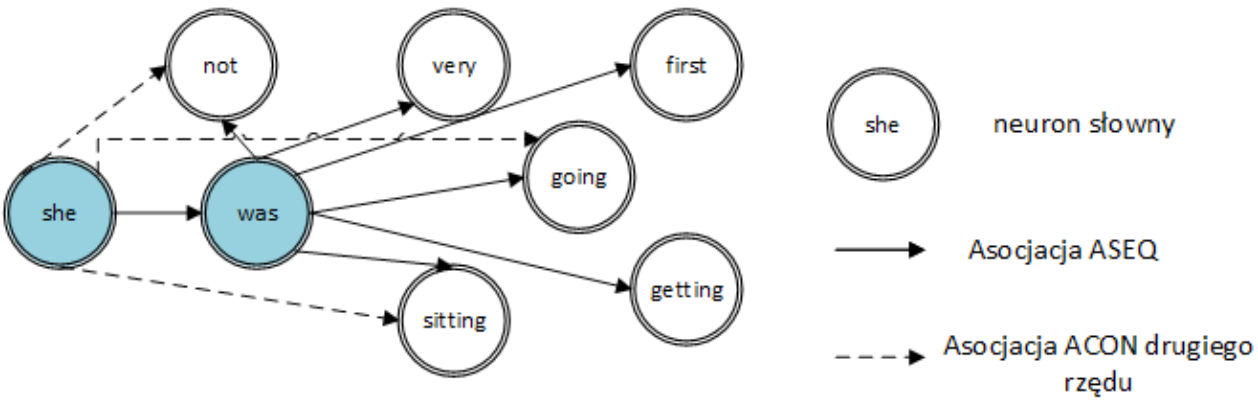
Uzyskane współczynniki dla podanych właściwości słowa *alice* można interpretować w następujący sposób: słowo pojawiło się 131 razy w tym 114 razy dużą literą, a jedynie 2 razy małą literą. Obliczając więc wskazaną nierówność, uzyskany został wynik 0,98. Tak więc jest bardzo prawdopodobne, wg wskazanego algorytmu, iż słowo *alice* powinno być napisane dużą literą. Po serii kolejnych eksperymentów współczynnik *bigLetterFactor* został ustalony dla aplikacji na wartość 0,87, aby uzyskać najlepsze wyniki korekcji. Oznacza to, że jeśli wynik oceny wystąpienia dużej litery jest równy bądź większy niż 0,87, to algorytm ten automatycznie uznaje wskazany wyraz jako wyraz pisany dużą literą [95].

Bardzo podobny algorytm zastosowano do sprawdzenia, czy przed lub po sprawdzanym wyrazie powinien zostać postawiony przecinek. W wyniku przeprowadzonych badań wykazano, że przecinek powinien wystąpić dla języka polskiego, m. in. przed słowami: aby, albowiem, aż, bo, chociaż, choć, czy, dlaczego, gdy, iż, jaki, jeśli, kiedy, kto, który, skąd, skoro, że, żeby. Wszystkie wyznaczone wyrazy przez ten algorytm powtarzają się w ogólnie przyjętych zasadach stawiania przecinka dla języka polskiego. Jedną z tych zasad jest sytuacja, w której przecinka używamy, aby oddzielić zdanie podrzędne (określające) od zdania nadrzędnego.

Zmodyfikowana metoda odległości edycyjnej

Jak zostało wspomniane w rozdziale 3, istnieją dwie powszechnie stosowane grupy metod dla zadania korekty tekstu. Pierwsza grupa opiera się na pomiarze odległości edycyjnej, druga grupa wykorzystuje modelowanie językowe za pomocą modeli n-gramowych. Te algorytmy również zostały zaimplementowane na podstawie własności grafu LHG.

Wyznaczenie odległości edycyjnej na ogół odbywa się poprzez obliczenie odległości między terminem zapytania a każdym terminem w słowniku językowym. Metoda ta jest bardzo kosztowna pod względem obliczeniowym, przez co także bardzo powolna. W podstawowej wersji tej metody należy przeszukać bowiem pełny słownik dla danego języka i przeprowadzić badanie odległości edycyjnej między badanym słowem, a słowem pochodzącym ze słownika. Oczywiście istnieją pewne usprawnienia, niemniej jednak w dalszym ciągu jest to mało efektywna metoda. Dzięki wykorzystaniu Grafu Przyzwyczajeń Lingwistycznych i występujących w nim połączeń asocjacji sekwencyjnej ASEQ, sprawdzanie odległości edycyjnej jest znacznie szybsze. W pierwszej kolejności analizowane są wszystkie połączenia ASEQ pomiędzy kolejnymi słowami wprowadzonymi przez użytkownika w danym kontekście. Jeśli nie znaleziono połączenia ASEQ między dwoma sąsiadującymi neuronami słownymi, algorytm bazujący na wyliczeniu odległości Damerau-Levenshteina bada tą odległość tylko między potencjalnie niepoprawnym słowem a wszystkimi słowami połączonymi relacją ASEQ lub ACON drugiego rzędu, wychodzącymi z poprzedniego neuronu słownego. Podejście to zwraca tylko kilka słów, dla których należy przeprowadzić badanie odległości edycyjnej. Jak wykazały badania, podejście to sprawdza się w znacznej liczbie przypadków. Na rysunku 20. przedstawiono kilka możliwych neuronów słownych wyznaczonych do badania dla kontekstu początkowego „she was...”.



Rysunek 20: Kolejne możliwe neurony słowne dla kontekstu początkowego „she was...”

Zmodyfikowana metoda n-gramów

Druga grupa metod wykorzystuje modele n-gramowe, których celem jest przypisanie określonego prawdopodobieństwa poprawności do danego zdania. Prawdopodobieństwo to można wykorzystać w tłumaczeniu maszynowym, korekcie pisowni i rozpoznawaniu mowy. Najprostszy sposób wykorzystuje model unigramowy, a więc taki, w którym do pomiaru poprawności całego zdania wykorzystywany jest kontekst jednego słowa. W tym przypadku prawdopodobieństwo jest liczone, jak pokazano we wzorze poniżej, gdzie w_i oznacza słowo na pozycji i-tej.

$$P(w_1 w_2 ... w_n) \approx \prod_i P(w_i) \tag{24}$$

Bardziej zaawansowane metody wykorzystują model bigramowy zamiast modelu unigramowego. Prawdopodobieństwo zdania przy użyciu modelu bigramowego jest zdefiniowane tak, jak pokazano na poniższych wzorach, gdzie w_i oznacza słowo na pozycji i-tej, a $c(w_{i-1}, w_i)$ jest licznikiem, ile razy sekwencja (w_{i-1}, w_i) wystąpiła we wszystkich analizowanych tekstach.

$$\begin{aligned} P(w_i|w_1 w_2 ... w_{i-1}) &\approx P(w_i|w_{i-1}) \\ P(w_i|w_1) &= \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \\ P(S) &= \prod_{i=1}^{l+1} P(w_i|w_{i-1}) \end{aligned} \tag{25}$$

W Grafie Przyzwyczajeń Lingwistycznych w łatwy sposób można sprawdzić istnienie określonych bigramów. W tym celu należy bowiem zbadać jedynie częstotliwość połączeń asocjacyjnych ASEQ, występujących pomiędzy odpowiednimi neuronami słownymi. Z drugiej strony jednak, jak zostało opisane w rozdziale 3, modele n-gramowe (unigramy, bigramy itd.) są bardzo często niewystarczającymi modelami do modelowania języka. Przeprowadzone badania wykazały, że przy użyciu modeli n-gramów nie osiągnięto należytej korekty tekstu. Niemniej jednak opracowano nowe metody, które dają szybszy wynik korekty tekstu niż przy użyciu metod modeli n-gramowych.

Pierwsza zaproponowana metoda operuje na asocjacjach sekwencyjnych, wykorzystując wszystkie możliwe ścieżki ASEQ między badanymi słowami. Początkowo, w tym przypadku, wyznaczana jest odległość ścieżki między słowami z wykorzystaniem otoczenia (ewentualnych innych neuronów słownych). Domyślnie algorytm sprawdza odległość ścieżki maksymalnie między dodatkowymi dwoma najbliższymi sąsiadami dla każdego węzła. Pod uwagę brane są tylko ścieżki o maksymalnej długości 3. Następnym krokiem jest wyznaczenie długości ścieżki dla każdego słowa. Algorytm zatem sprawdza następujące połączenia:

- liczność bezpośrednich połączeń między badanymi neuronami. Jest to liczba bezpośrednich ścieżek od zadanego początkowego neuronu słownego do zadanego końcowego neuronu słownego (startWordNeuron → endWordNeuron),
- liczność połączeń między badanymi neuronami z dodatkowym krokiem. Jest to liczba ścieżek przechodzących od neuronu początkowego do neuronu końcowego, z uwzględnieniem jednego neuronu sąsiadującego (startWordNeuron → $wordNeuron_1$ → endWordNeuron),
- liczność połączeń między badanymi neuronami z dwoma dodatkowymi krokami. Jest to liczba ścieżek przechodzących od neuronu początkowego do neuronu końcowego, z uwzględnieniem dwóch neuronów sąsiadujących (startWordNeuron → $wordNeuron_1$ → $wordNeuron_2$ → endWordNeuron).

Po określeniu liczności dla każdego słowa kolejnym działaniem algorytmu jest obliczenie dla każdej ze znalezionych ścieżek pewnego „priorytetu", dzięki któremu mogą zostać aktywowane przejścia w danym kontekście. Algorytm ten pozwala określić, które słowa są w poprawnej kolejności, a które w niewłaściwej i muszą zostać poprawione przez inne zaproponowane algorytmy. Priorytet dla wskazanego neuronu słownego na danej pozycji jest obliczany jako suma jego liczby wystąpień na wskazanej pozycji (o_{nb}) pomnożonej przez różnicę w odległości między wskazanymi neuronami słownymi. Równanie to zostało przedstawione poniżej:

$$w_p(i) = \sum_{n=i-2}^{i+2} o_{nb}(i) * (\frac{1}{2})^{|(i-n)|} \tag{26}$$

Kolejnym zaproponowanym algorytmem wykorzystującym informację o bezpośrednich połączeniach ASEQ jest metoda, która pomaga w ustaleniu prawidłowej kolejności słów w danym zdaniu. Algorytm działa bardzo dobrze przy wykrywaniu następujących rodzajów błędów:

- gdy słowo zostało pominięte w zdaniu;
- gdy słowo zostało nadmiernie wprowadzone do zdania;
- gdy dwa słowa zostały połączone w jedno.

Algorytm bada wprowadzony tekst pod względem wystąpienia w Grafie Przyzwyczajeń Lingwistycznych asocjacji sekwencyjnych ASEQ pomiędzy kolejnymi wyrazami. Gdy połączenie pomiędzy parą (w_i, w_{i+1}), gdzie i oznacza i-tą pozycję słowa w zdaniu, nie wystąpi, to automatycznie badane jest sąsiedztwo słowa w_i. Pod uwagę brane są następujące otoczenia (badanie jest, czy w grafie występuje ścieżka połączeń ASEQ pomiędzy parami):

- (w_{i-1}, w_{i+1}),
- (w_i, w_{i+2}),
- (w_{i-1}, w_{i+2}),
- (w_i, w_{i+3}).

Jeśli jedna lub więcej z tych ścieżek wystąpi, to otrzymana zostanie informacja o tym, jaką korektę tekstu należy wprowadzić.

Metoda wykorzystująca pobudzenia asocjacyjne

Inny nowo zaproponowany algorytm służący semi-automatycznej korekcie tekstu został wprowadzony na podstawie inspiracji zaczerpniętej z zachowań neuronów w mózgu człowieka. Graf Przyzwyczajeń Lingwistycznych jest znakomitym modelem, w którym można zasymulować proces, w jaki neurony stymulują kolejne połączone neurony w mózgu.

Funkcją neuronów jest odbieranie, przetwarzanie i przekazywanie sygnałów. Jest to możliwe jednak pod pewnymi warunkami: ilość sygnałów pobudzających neuron musi być odpowiednia, a sygnały hamujące powinny być słabsze od tych pobudzających, by neuron mógł osiągnąć próg aktywacji i wysłać sygnał dalej do połączonych z nim neuronów [96]. Oznacza to, że jeśli potencjał elektryczny przychodzącego sygnału jest większy niż potencjał progowy pojedynczego neuronu to sygnał zostanie przekazany. W przeciwnym razie, żaden sygnał nie zostanie przekazany do kolejnych neuronów. Czynność ta wykonywana jest na podstawie wyznaczenia funkcji progowej zaprezentowanej na równaniach poniżej:

$$y_k = \varphi \sum_{j=0}^{m} w_{kj} x_j \tag{27}$$

$$y_k = \begin{cases} 1 \text{ jesli } \theta_k \geq 0 \\ 0 \text{ jesli } \theta_k < 0 \end{cases} \tag{28}$$

Każdy sygnał wejściowy (dendryt) ma przypisaną wagę w_{kj} do wartości wejściowej x_j. Następnie suma wszystkich danych wejściowych mnożona jest przez funkcję przenoszenia φ. Wynik tej operacji porównywany jest z wartością progową θ_k. Przepuszczony sygnał jest następnie

przekazywany do kolejnego neuronu wchodzącego w skład układu nerwowego przez synapsę lub grupę synaps [97].

W grafie LHG proces korekcji tekstu dla opisywanego algorytmu rozpoczyna się zawsze od aktywacji neuronu specjalnego, który wyznacza początek zdania. Następnie aktywowane są kolejne neurony dla poprawianego zdania, w celu zbudowania zdania poprawnego. Aktywacja ta odbywa się w odpowiedniej kolejności, zgodnie z siłą wag między neuronami słownymi. Kontekst poprzednio aktywowanych neuronów słownych pobudza kolejne neurony słowne, które zwykle pojawiają się we wskazanym kontekście z wykorzystaniem asocjacji sekwencyjnych ASEQ oraz asocjacji kontekstowych ACON. Połączenia te są dodatkowo wzmacniane przez ich częstotliwości, które są obliczane podczas budowy sieci grafu LHG. Aktywowane neurony stymulują kolejne połączone z nimi neurony, biorąc pod uwagę wagi obliczone w zależności od częstotliwości takich sekwencji słownych. Stymulacja neuronu jest definiowana przy użyciu kilku stymulacji:

- stymulacji zewnętrznej extIn(t),
- stymulacji wewnętrznej intIn(t),
- poprzedniego stanu neuronu w kroku czasowym (t – 1).

Waga w_i określona jest poprzez częstotliwość połączenia (f_i) między neuronami słownymi, która dodatkowo jest znormalizowana przez sumę częstotliwości wszystkich innych połączeń wejściowych danego neuronu.

Szczegółowe równania określające ten algorytm zapisano poniżej. Wyznaczenie wagi odbywa się na podstawie obliczenia równania:

$$w_i = \frac{f_i}{\sum_{j=0}^{J} f_j} \tag{29}$$

Wyznaczenie stymulacji dla neuronu odbywa się natomiast poprzez obliczenie równania:

$$x(t) = \frac{x(t-1)}{2} + \mathrm{extIn}(t) + \sum_{j=1}^{J} w_j * \mathrm{intIn}_j(t) \tag{30}$$

Jak można zauważyć, w rzeczywistości połączenia ASEQ i ACON są aktywowane z badanego neuronu za każdym razem. Założono dodatkowo, że na każdym etapie poprzednie pobudzenie neuronalne wybranego neuronu słownego będzie pomniejszane o połowę. W ten sposób połączenia asocjacji kontekstowych ACON wyższych rzędów mają znacznie mniejszy wpływ na aktywację następnego neuronu słownego. Wraz z rosnącym poziomem asocjacji kontekstowej

pobudzenie jest mniejsze. Innymi słowy można stwierdzić, iż im bliższy kontekst, tym większy ma on wpływ na poziom wzbudzenia następnych neuronów słownych. Jeśli wskazana aktywacja przejścia do kolejnego neuronu słownego nie jest możliwa, wówczas sugerowana jest korekta do najbardziej podobnego lub najbardziej częstego kolejnego słowa. Jeśli po sugerowanym neuronie słownym następuje neuron słowny, który reprezentuje następne słowo w poprawionym zdaniu, to sugestia ta jest jeszcze silniejsza. Zastosowanie opisanego powyżej algorytmu semi-automatycznej korekty tekstu daje bardzo dobre wyniki. Jedynym jego ograniczeniem jest potrzeba wcześniejszego zbudowania sieci Grafu Przyzwyczajeń Lingwistycznych dla możliwie dużej ilości korpusów tekstowych.

Podsumowując, sprawdzanie pisowni jest jednym z najczęstszych zadań dotyczących przetwarzania języka naturalnego. Ma ono szeroki zakres zastosowań. Wykorzystywane jest podczas wyszukiwania informacji, korekty tekstu, itp. Obecnie w wielu aplikacjach NLP używany jest moduł do sprawdzania pisowni, który analizuje tekst pod kątem wystąpienia głównie błędów ortograficznych. Mechanizmy te najlepiej radzą sobie z przypadkiem, gdy błędny wyraz nie występuje w słowniku danego języka. Bazując na metodzie słownikowej, algorytmy mają ściśle określone słowa, z którymi mogą porównać każde wprowadzane słowo. Jeśli wprowadzonego słowa nie znajdą w słowniku, wyraz ten automatycznie uznawany jest za błędny. Oczywiście jest to poprawne zachowanie. Niemniej jednak, we wpisywanym tekście mogą wystąpić wszelkiego rodzaju inne błędy, dla których mechanizmy te nie zadziałają. Jednym z kluczowych obszarów jest znajomość kontekstu wypowiedzi. Znając kontekst, można bowiem stwierdzić, że choć dane słowa występują w słowniku językowym, to użycie ich w danym kontekście jest błędne. Opracowane metody semi-automatycznej korekty tekstu na bazie Grafu Przyzwyczajeń Lingwistycznych umożliwiają ten problem rozwiązać. Opracowane algorytmy bazują bowiem, podczas swojego działania, na asocjacjach sekwencyjnych ASEQ oraz asocjacjach kontekstowych ACON. Połączenia te są tak naprawdę odzwierciedleniem kontekstu wypowiedzi, który został zapisany w grafie. Może on bowiem sugerować różne opcje korekty w kontekście innych słów pojawiających się we wcześniej przeczytanych zdaniach. Ponadto różne sugerowane opcje są ważone przez częstotliwość ich używania przez innych ludzi w przeszłości oraz przez określony kontekst innych słów w analizowanych zdaniach. Warto również zwrócić uwagę na to, że algorytmy działające na bazie sieci grafowej, są bardzo szybkie. Spowodowane jest to skończoną i ograniczoną ilością potrzebnych przejść. Nie trzeba w tym przypadku przeszukiwać wszystkie możliwe ścieżki czy neurony. Dla zaprojektowanych metod wystarczy przeszukać zazwyczaj kilkanaście połączeń. Dodatkowo, w wyniku połączeń asocjacyjnych w grafie, proces sprawdzania i określania podgrupy najbardziej prawdopodobnych poprawek jest zawsze dostępny w stałym czasie.

Rozdział 5

Rodzaje opracowanych metod do korekty tekstu

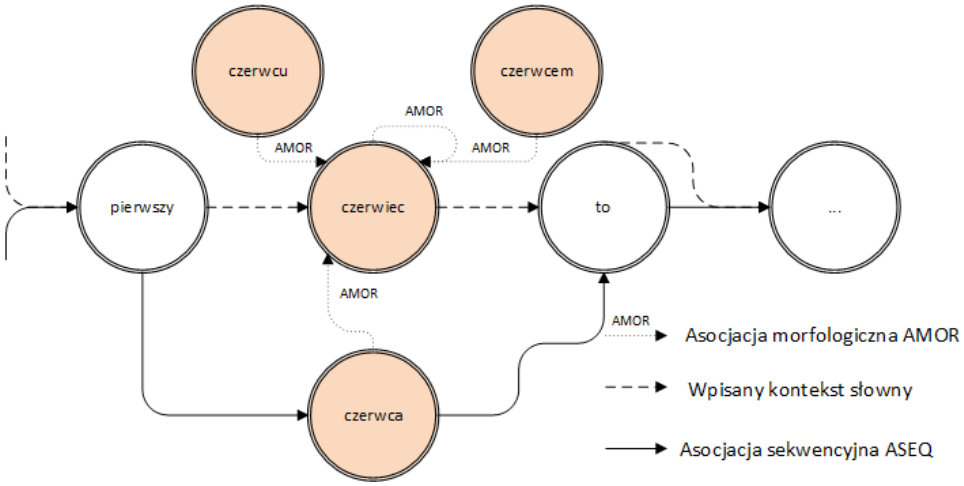
Jak zostało omówione w poprzednim rozdziale, na bazie opracowanego Grafu Przyzwyczajeń Lingwistycznych zostało utworzonych kilka metod semi-automatycznej analizy i korekty tekstu. Do metod tych można zaliczyć:

1. Statyczne sprawdzanie poprawności poszczególnych wyrazów w słownikach językowych.

Ponieważ opracowane metody przetestowano dla zdań zarówno w języku polskim jak i angielskim, dlatego skorzystano ze słowników odpowiednich dla tych języków. Dla języka polskiego wykorzystano Słownik Języka Polskiego PWN (https://sjp.pl) oraz słownik morfologiczny (https://github.com/morfologik), natomiast dla języka angielskiego skorzystano ze Słownika Języka Angielskiego (http://app.aspell.net/create z wykorzystaniem *Spell Checking Oriented Word Lists*).

Metody z wykorzystaniem słownika poprawnej pisowni są znane i powszechnie stosowane również w innych programach do korekty tekstu. Działanie ich polega na sprawdzeniu, czy dane słowo wpisane przez użytkownika występuje w słowniku. Jeśli nie występuje, to poszukiwane są słowa występujące w słowniku, którymi można zastąpić błędnie wpisane słowo.

Dla języka polskiego skorzystano dodatkowo podczas tworzenia grafu ze słownika morfologicznego. Słownik ten zawiera informacje dotyczące fleksji języka. Za jego pomocą możliwe jest m. in. określenie formy podstawowej (słowa bazowego) dla danego wyrazu. Informacja ta jest wykorzystywana podczas sprawdzania pisowni. Podczas badań zauważono, że często występujące błędy cechują się użyciem niewłaściwej odmiany słowa. Dzięki wykorzystaniu informacji o formie podstawowej wszystkie słowa oznaczające to samo, a użyte w innej formie leksykalnej, posiadają tą samą formę podstawową. Podczas konstrukcji grafu połączone są one specjalną krawędzią – asocjacją morfologiczną AMOR, opisaną wcześniej w pracy. Podczas korekty tekstu, jeśli słowo uznane jest za błędne, możliwe jest łatwe wyznaczenie wszystkich jego odmian, a następnie sprawdzenie, czy słowo w innej formie fleksyjnej dla słowa błędnie wprowadzonego, nie występuje w zadanym kontekście. Schemat działania takiej korekty został zaprezentowany na rysunku 21.



Rysunek 21: Przykład połączenia AMOR występującego w grafie LHG

Jak zaobserwowano, wykorzystanie słownika morfologicznego do korekty tekstu nie jest powszechnie stosowane. Dla języka polskiego występują aplikacje, które wykorzystują takie słowniki, niemniej jednak działają one osobno. Nie są powiązane z innymi metodami korekty tekstu i służą głównie do podania formy podstawowej słowa. Przykładem takiej aplikacji może być analizator morfologiczny Morfeusz (http://morfeusz.sgjp.pl/doc/about/). Program Morfeusz wykonuje analizę morfologiczną słów dla języka polskiego. Na rysunku 22. zaprezentowany został przykład otrzymanych wyników działania programu dla tekstu „Mam próbkę analizy morfologicznej.”.

Segment	Forma ortograficzna	Lemat	Znacznik
0-1	Mam	mama	subst:pl:gen:f
		mamić	impt:sg:sec:imperf
		mieć	fin:sg:pri:imperf
1-2	próbkę	próbka	subst:sg:acc:f
2-3	analizy	analiza	subst:sg:gen:f
			subst:pl:nom.acc.voc:f
3-4	morfologicznej	morfologiczny	adj:sg:gen.dat.loc:f:pos
4-5	.	.	interp

Rysunek 22: Przykład otrzymanych wyników działania programu Morfeusz dla tekstu „Mam próbkę analizy morfologicznej.”

Cytując autorów *„Tekst wejściowy został podzielony na słowa (w szczególności kropka została oddzielona od napisu „morfologicznej”). Na prawo od słów podano odpowiadające im lematy (formy hasłowe), a w następnej kolumnie — znaczniki opisujące wartości kategorii gramatycznych charakteryzujące poszczególne formy. Słowu „mam” zostały przypisane trzy*

interpretacje: jako forma liczby mnogiej rzeczownika „mama", jako forma trybu rozkazującego czasownika „mamić" i wreszcie jako forma czasu teraźniejszego czasownika „mieć". Słowo „analizy" zostało jednoznacznie przypisane do lematu „analiza", może ono jednak być interpretowane zarówno jako forma liczby pojedynczej jak i mnogiej — w różnych przypadkach." [98].

Jak można przeczytać, dla wspomnianego słowa „mam" zostały przypisane trzy interpretacje. W grafie będą to trzy połączenia AMOR do słów podstawowych – *mama*, *mamić* oraz *mieć*. Stanowi to niezwykle cenną właściwość grafu. Podczas wykorzystania w grafie metody opierającej się na słowniku morfologicznym, zostanie pobudzonych kilka neuronów słownych – odpowiednio połączonych asocjacjami AMOR dla trzech słów podstawowych i sprawdzone zostanie ewentualne wystąpienie kontekstu słownego dla trzech grup wyrazów, które pozornie są od siebie zupełnie różne i w żaden sposób nie powiązane.

Jak twierdzą autorzy programu Morfeusz „*Za bardziej interesującą uważamy możliwość użycia biblioteki Morfeusz we własnym programie, np. napisanym, w którymś z języków skryptowych. Daje to dużo większą elastyczność wykorzystania wyników analizatora (np. można łatwo ograniczyć analizę do samego hasłowania)*" [98]. Tak więc raz jeszcze zostało podkreślone, że użycie słownika morfologicznego jest cenne, natomiast powinno zostać zaimplementowane w osobnych programach.

2. Algorytmy operujące na asocjacjach sekwencyjnych oraz kontekstowych dla wprowadzonego tekstu.

Główną zaletą posiadania grafowej struktury jest możliwość wykorzystania jej do łatwego zapisu, a następnie analizy i przeprowadzenia korekty wypowiedzi. Nowatorskie zdolności korekcyjne zostały osiągnięte właśnie dzięki zastosowaniu opisanego wcześniej, aktywnego mechanizmu asocjacyjnego, zaimplementowanego w strukturze grafowej. Badania wykazały, że bezpośrednie połączenia między powiązanymi obiektami, takimi jak litery i słowa, mają ogromne znaczenie i powinny być wykorzystywane do konstruowania jeszcze bardziej efektywnych algorytmów. Strategia ta jest najprawdopodobniej również stosowana w naturalnych sieciach neuronowych, w naszych mózgach i właśnie ona pozwala nam tak szybko kojarzyć dane. Eksperymenty wykazały, że korekta może być lepsza, jeśli użyje się do jej przeprowadzenia kontekstu słów, występujących w bliskim otoczeniu badanego słowa. Kontekstu, tak potrzebnego dla Grafu Przyzwyczajeń Lingwistycznych, można nauczyć się za pomocą ogromnej ilości korpusów tekstowych. Poniżej przedstawiono szczegółowy opis korekty tekstu w grafie LHG dla przykładu zdania wprowadzonego z błędami.

Korekta tekstu za pomocą zaimplementowanej metody pobudzeń asocjacyjnych

Po zbudowaniu Grafu Przyzwyczajeń Lingwistycznych dla korpusów tekstów dla języka angielskiego przeprowadzono korektę tekstu dla błędnego zdania. W tym celu wprowadzono zdanie „She was afraid of taking to Joh.”. Z kontekstu tego zdania wynika, że występują w nim dwa błędy:

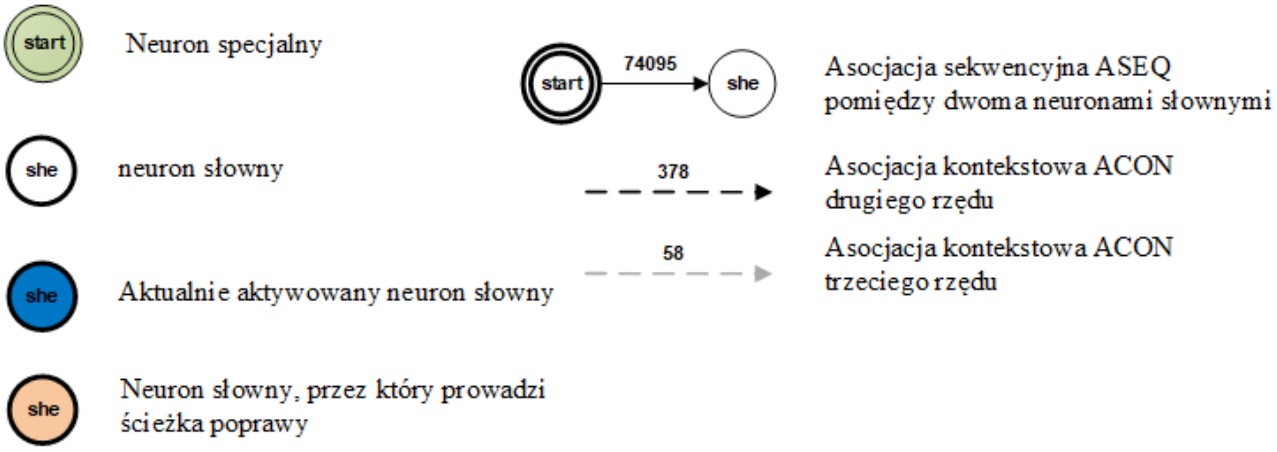
1. Błędnie zostało użyte słowo taking. W wyrażeniu tym, należało bowiem użyć słowa talking. Błąd ten mógł się pojawić w skutek pominięcia jednej litery „l”. Warto zwrócić uwagę, że zarówno słowo taking jak i talking występują w języku angielskim. Słowo *taking* może być formą ciągłą dla słowa *take* i w tym kontekście można go przetłumaczyć jako *brać, zabierać*. Dodatkowo wyraz taking może być traktowany jako przymiotnik i oznacza wówczas *uroczy (o osobie), ujmujący, zachwycający* lub w innym kontekście może być przetłumaczony jako *chwytliwy, zaraźliwy*.

   Słowo *talking* jest natomiast formą ciągłą dla słowa *talk* oznaczającego *rozmawiać, mówić opowiadać*. Może być również tłumaczone jako rzeczownik *rozmawianie, mówienie* lub jako przymiotnik *rozmowny, gadający (np. ptak)*.

2. Drugim błędem jest użycie słowa Joh. Słowo to nie występuje w słowniku języka angielskiego. W przypadku tego błędu nie ma jednoznacznej poprawnej korekty. Wiadomo, że słowo Joh jest niepoprawne, ale istnieje kilka możliwości jego poprawy. Można znaleźć bowiem kilka słów podobnych, np. John, Josh, Joe. Z samego kontekstu i struktury zdania można wywnioskować, że każde imię może być tutaj poprawne. Nic nie stoi bowiem na przeszkodzie poprawy zdania na „She was afraid of talking to Paul.”. Niemniej wykorzystując dodatkowy algorytm, np. badanie odległości edycyjnej, gdzie wszystkie operacje dodania, usunięcia i zamiany są równe, można stwierdzić, że najbardziej prawdopodobnym będzie użycie zamiast słowa Joh słów, których odległość edycyjna jest równa jeden. Będą to więc słowa takie jak John, Josh lub Joe.

Na rysunkach 24. oraz 25. przedstawione zostały poszczególne kroki przeprowadzania kontekstowej korekty dla podanego tekstu. Dla lepszego zrozumienia i łatwiejszego przedstawienia, na rysunkach tych zawarte zostały jedynie pewne wycinki grafu LHG dla każdego z kroków. Nie uwzględniono na nich również wszystkich ścieżek występujących w grafie (ukryte zostały niektóre połączenia ASEQ oraz ACON), ukryte zostały również neurony literowe. Na rysunku 23. omówione zostały symbole użyte podczas korekty tekstu. Warto zwrócić uwagę, że każda asocjacja

ma swój indywidualny licznik, który określa, ile razy dane połączenie wystąpiło pomiędzy dwoma neuronami.



Rysunek 23: Legenda dla mechanizmu kontekstowej korekty tekstu

Po wpisaniu zdania z możliwymi błędami mechanizm automatycznej kontekstowej korekty tekstu dokonuje wyodrębnienia poszczególnych tokenów, przeprowadza również proces ich ujednolicania (np. słowo Joh zostaje znormalizowane do formy „joh"). Gdy etap normalizacji słów zostanie skończony, rozpoczyna się proces sprawdzania i ew. korekty tekstu.

Pierwszym krokiem algorytmu jest odszukanie w Grafie LHG neuronu specjalnego, który wskazuje początek zdania. Następnie dla tego neuronu sprawdzane jest, czy istnieje asocjacja sekwencyjna dla tokenu słownego uznanego jako pierwszy. Jak można zauważyć, z neuronu specjalnego „start" będzie wychodziło bardzo dużo połączeń ASEQ. Stąd, algorytm nie sprawdza wszystkich ścieżek po kolei. Bada on wystąpienie jednej konkretnie określonej ścieżki. Jak zostało przedstawione to na ilustracji, połączenie takie występuje. Co więcej, algorytm posiada informację, że wyraz „she" rozpoczynał zdanie w 74 065 przypadkach.

Drugim krokiem algorytmu jest przejście do wskazanego, kolejnego słowa, a więc do wyrazu „she" i aktywowanie wszystkich asocjacji wychodzących z danego wierzchołka. Jak zostało opisane wcześniej, w każdym kroku podczas przechodzenia pomiędzy neuronami słownymi, każde wejście pewnego potencjału do neuronu słownego aktywuje ten neuron pod warunkiem osiągnięcia wartości wyższej lub równiej dla progu przejścia. Jeśli neuron słowny zostaje aktywowany, to w kolejnym kroku aktywowane są wszystkie jego asocjacje wychodzące. W chwili obecnej dla algorytmu najważniejsze są asocjacje sekwencyjne ASEQ, niemniej warto już teraz pamiętać o tym, że stymulacja neuronu jest definiowana jako suma stymulacji zewnętrznej, wewnętrznej oraz poprzedniego stanu neuronu słownego w poprzednim kroku czasowym – opisanych wzorem 30. Jak

zaprezentowane zostało na rysunku, istnieje asocjacja sekwencyjna pomiędzy dwójką neuronów słownych (she, was), tak więc można uznać, że początkowy kontekst zdania „She was" jest poprawny i można przejść do aktywacji kolejnego neuronu na ścieżce, a więc do neuronu „was".

Trzeci krok to wspomniana już wcześniej aktywacja neuronu słownego „was". Podobnie jak poprzednio, ten neuron słowny przekroczył próg pobudzenia, więc aktywuje wszystkie asocjacje wychodzące. Dodatkowo po ich aktywacji, badane są progi pobudzenia dla wszystkich neuronów aktualnie stymulowanych. Jeśli dla jakiegoś neuronu słownego próg pobudzenia jest dostatecznie wysoki, następuje jego wzbudzenie. W aktualnym kroku dochodzi do aktywacji kolejnego neuronu na ścieżce, zgodnego z wprowadzonym tekstem, a więc neuronu „afraid".

Czwartym krokiem jest aktywacja słowa „afraid". W tym kroku można zauważyć, że na rysunku zostało zaprezentowane pobudzenie asocjacji sekwencyjnych ASEQ dla m. in. słów „to", „of", „he" oraz „brother" jak i pobudzenie asocjacji kontekstowych ACON. Połączeniami ACON drugiego rzędu pobudzane są neurony słowne dla m. in. takich wyrazów jak „you", „talking", „speaking" oraz „took". Warto zwrócić uwagę także, że słowo „taking" nie zostało dotychczas pobudzone przez żadną asocjację. Co więcej, słowo takie nie zostało dodane do grafu w procesie uczenia.

Piąty krok związany jest z aktywacją słowa „of". Słowo to bezpośrednio aktywuje asocjacja sekwencyjna wyrazów „that", „talking", „from" oraz „such". Jak można zauważyć, na liście aktywowanych słów nie występuje słowo „taking", które zostało podane jako kolejne w zdaniu poddanym korekcie. Również takie połączenie kontekstowe dla tej dwójki słów nie występuje. Neurony słowne „pamiętają" przez pewien czas swój stan pobudzenia, który z każdym krokiem jest zmniejszany. W dalszym ciągu aktywne są pobudzenia z poprzedniego kroku, a więc pobudzenia kontekstowe od słowa „afraid" dla słów „you", „talking", „speaking" oraz „took". Zaproponowany algorytm sprawdza wszystkie neurony, które są aktualnie pobudzone i aktywuje neuron, który posiada najwyższe pobudzenie. Jak można zwrócić uwagę, wyraz „talking" jest pobudzony przez:

- asocjację sekwencyjną pochodzącą od słowa of,
- asocjację kontekstową pochodzącą od słowa afraid.

Ten krok pokazuje przewagę opracowanego algorytmu. Potrafi on wyznaczyć w jasny sposób kolejny neuron słowny, który powinien zostać wybrany podczas etapu korekty tekstu i jakim słowem powinno zostać zastąpione słowo błędnie wprowadzone.

Szósty krok dotyczy aktywacji neuronu słownego, który został wybrany w poprzednim kroku. Po raz pierwszy nie jest to neuron słowny pochodzący ze zdania wpisanego przez

użytkownika. Jest to bowiem neuron, który został wyznaczony w procesie kontekstowej korekty tekstu z zastosowaniem omawianej metody. Dla tego neuronu podobnie wyznaczane i obliczane są kolejne asocjacje pochodzące z aktywowanego w tym kroku neuronu, jak również z neuronów aktywowanych wcześniej.

Krok siódmy dotyczy aktywacji neuronu słownego „to". Krok ten jest identyczny w swoim działaniu jak poprzednie etapy. Warto zwrócić uwagę jednak, że połączeniem kontekstowym drugiego rzędu pobudzony zostaje neuron specjalny określający zakończenie zdania. W kroku tym dokonuje się również druga korekta dla wyrazu błędnie wprowadzonego. Dotychczas nie odbyło się pobudzenie błędnego neuronu słownego „joh". Algorytm na tym etapie wyznaczył kilka możliwych zamian. Kolejnymi możliwymi słowami będzie m. in. wyraz *me* (częstość występowania tego słowa to 8987 razy), *him* (częstość występowania 2530), *john* (częstość występowania 167). Ponieważ istniej kilka możliwych kolejnych przejść z taką samą wagą aktywacji, algorytm bierze pod uwagę częstość występowania danego słowa. W tym wypadku poprawnym słowem będzie słowo *me*, ponieważ wystąpiło najczęściej w badanym kontekście.

Warto nadmienić, że choć korekta tekstu może wyglądać na niepoprawną, opisywany jest w tym miejscu jedynie pojedynczy algorytm. W kompleksowym rozwiązaniu, gdy do korekty tekstu zostanie zastosowanych kilka algorytmów, w kroku tym opisywany algorytm zasugeruje kilka możliwych wyrazów (me, him, john, itd.). Następnie algorytm wyznaczający odległość edycyjną pomiędzy dwójką słów i określi, że bardziej prawdopodobnym do poprawy będzie słowo *john* niż *me*, gdyż odległość edycyjna pierwszego słowa jest mniejsza niż drugiego.

Ostatnim krokiem jest sprawdzenie, czy dla aktywnego neuronu „me" występuje ścieżka prowadząca do neuronu specjalnego oznaczającego zakończenie zdania. Okazuje się, że ścieżka taka występuje i istnieje połączenie dla dwójki (me, koniec_zdania).

Po przeprowadzeniu tych operacji algorytm asocjacyjnej korekty tekstu wyznaczył błędne słowa i zdołał poprawić jedno z nich. Ostatecznie błędnie wprowadzone zdanie „She was afraid of taking to Joh." zostało zmienione na zdanie uznane przez algorytm za poprawne „She was afraid of talking to me.'.

Następnie wykonano eksperyment, którego celem było sprawdzenie słuszności zaimplementowania połączeń kontekstowych ACON wraz z połączeniami ACON_PREV. Jak wskazano we wcześniejszym fragmencie pracy, dysponując jedynie połączeniami ACON (bez połączeń ACON_PREV) nie jest możliwe odtworzenie pełnego kontekstu wypowiedzi we

wszystkich przypadkach. Przykładem może być wystąpienie niejednoznaczności dla pierwszych wyrazów w zdaniu. Podobnie przy zastosowaniu jedynie połączeń asocjacji kontekstowej „w tył" (bez połączeń asocjacji kontekstowej „w przód") niemożliwym jest wyeliminowanie niejednoznaczności występującej w końcowych wyrazach zdania. W celu wykazania słuszności zaimplementowania dwóch typów połączeń kontekstowych wraz z dwoma niezależnymi algorytmami do korekty tekstu wykonano następujące badanie: w 20 zdaniach dla języka angielskiego, które zawierały min. 5 wyrazów, wprowadzono manualnie następujące błędy:

- dla pięciu pierwszych zdań wykonano zamianę dwóch liter w pierwszym wyrazie zdania;
- dla pięciu następnych zdań wykonano zamianę dwóch liter w drugim wyrazie zdania;
- dla pięciu następnych zdań wykonano zamianę dwóch liter w przedostatnim wyrazie zdania;
- dla pięciu ostatnich zdań wykonano zamianę dwóch liter w ostatnim wyrazie zdania.

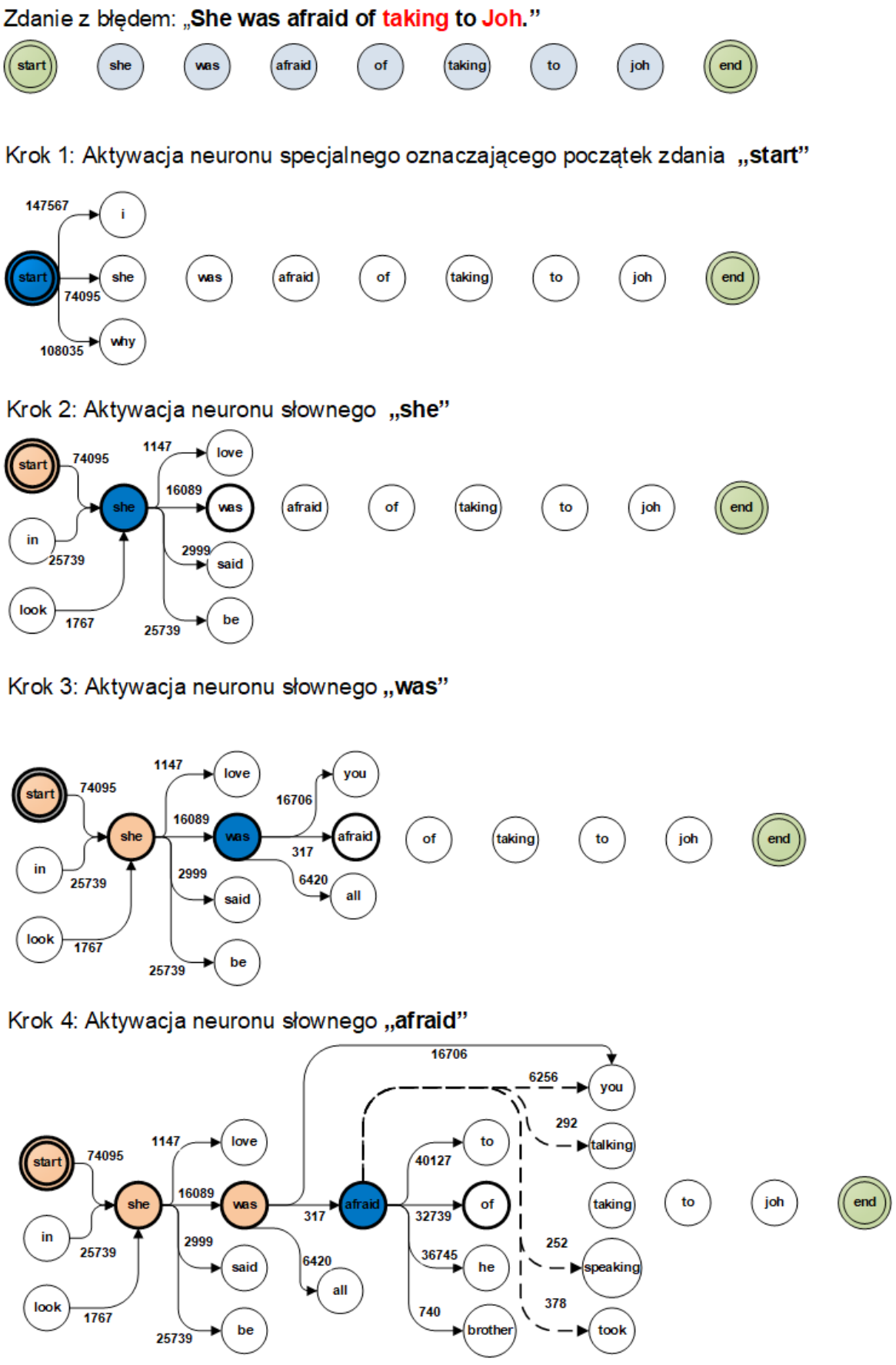
Następnie uruchomiono semi-automatyczną korektę tekstu wykorzystując:

- jedynie algorytm asocjacyjnej korekty tekstu „w przód";
- jedynie algorytm asocjacyjnej korekty tekstu „w tył";
- algorytmy asocjacyjnej korekty tekstu „w przód" oraz „w tył".

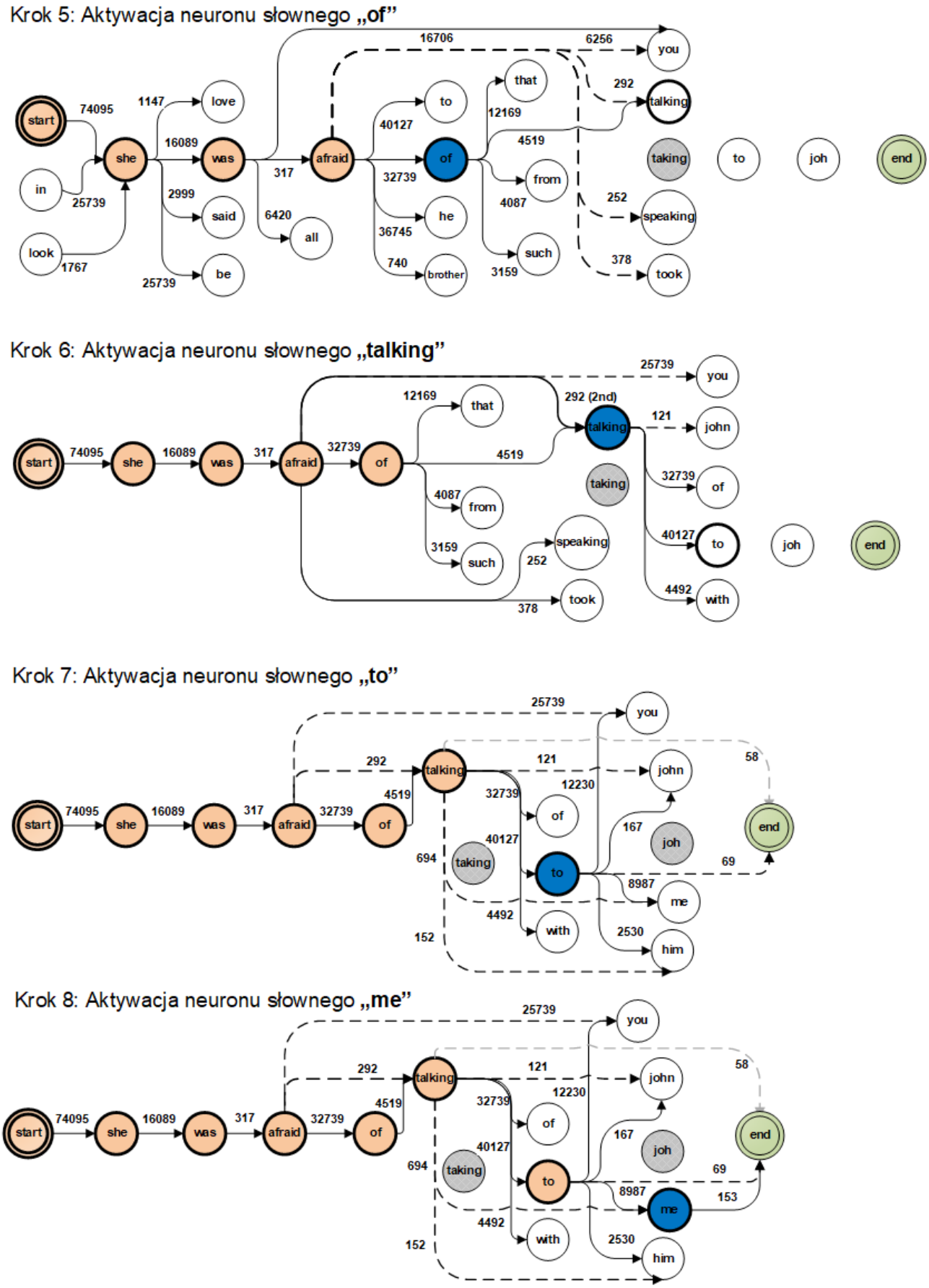
W tabeli poniżej przedstawiono wyniki korekty zdań wprowadzonych z błędami.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów
Algorytm asocjacyjnej korekty tekstu „w przód"	13	65%
Algorytm asocjacyjnej korekty tekstu „w tył"	12	60%
Łącznie algorytmy asocjacyjnej korekty tekstu „w przód" oraz „w tył"	20	100%

Jak można zauważyć, gdy błędy występują na początku lub na końcu zdania wymagane jest działanie dwóch algorytmów poprawy tekstu. Wówczas wykrywane są wszystkie błędy i możliwa jest poprawna korekta tekstu.



Rysunek 24: Początkowe kroki opracowanego algorytmu do asocjacyjnej korekty tekstu



Rysunek 25: Końcowe kroki opracowanego algorytmu do asocjacyjnej korekty tekstu

Porównanie korekty tekstu z istniejącymi aplikacjami dla języka angielskiego

Wykonane metody semi-automatycznej korekty tekstu przetestowano i porównano z ogólnie dostępnymi narzędziami, które również służą do korekty tekstów. Aplikacje te można podzielić na dwie zasadnicze grupy:

1. Pierwsza grupa dotyczy aplikacji, które występują w postaci stron internetowych. Na stronach tych należy wpisać proponowany tekst, a kilka chwil później otrzymywany zostaje tekst poprawiony.
2. Drugą grupę stanowią „dodatkowe moduły" dostępne dla edytorów tekstów. Jednymi z najpopularniejszych edytorów tekstów są Microsoft Word, wchodzący w skład oprogramowania Microsoft Office oraz jego darmowe alternatywy, np. program Writer, wchodzący w skład darmowego oprogramowania LibreOffice.

Największym problemem w narzędziach korekcji tekstu jest to, że większość z nich nie rozwiązuje problemu błędnego kontekstu wypowiedzi. W odróżnieniu od nich, głównym celem budowy grafu LHG jest właśnie przechowywanie połączeń ASEQ i ACON dwóch typów oraz, co za tym idzie, odbudowanie za ich pomocą poprawnego kontekstu zdań. Wykonane algorytmy półautomatycznej korekty działają dobrze przy założeniu, że graf LHG został skonstruowany na możliwie dużym korpusie tekstu szkoleniowego. Ogólnie, kontekst poprzednio aktywowanych neuronów słownych pobudza następne neurony słowne, które zwykle pojawiają się w danym kontekście.

W obecnych czasach powstaje coraz więcej narzędzi wchodzących w skład pierwszej grupy. Każda z aplikacji cechuje się możliwością poprawy tekstu. Jak będzie można zauważyć w dalszych eksperymentach, każda z nich działa inaczej, oferując różną jakość w korekcie tekstu. Nie ma jednej, „najlepszej" aplikacji. Każda z dostępnych aplikacji będzie sprawdzała się lepiej w określonych zadaniach, a podczas innego trybu pracy nie będzie w stanie poprawić wszystkich błędów.

Graf Przyzwyczajeń Lingwistycznych może zostać zbudowany dla korpusów tekstów różnych języków. Podczas badań wykorzystywano tą strukturę do korekty tekstu zarówno dla zdań wprowadzonych w języku angielskim, jak i polskim. Dlatego też w dalszej części tego rozdziału porównano aplikacje, służące do korekty tekstu dla tych dwóch języków.

Aplikacji do korekty tekstu w języku angielskim jest znacznie więcej niż dla języka polskiego. Wynika to z faktu, że języka angielskiego używa dużo więcej osób na świecie niż języka polskiego. Z danych zamieszczonych na portalu https://www.ethnologue.com/ wynika, że języka angielskiego używa ponad 1,5 biliona osób (1500 miliona osób), natomiast języka polskiego używa ok. 40 milionów osób. Tak więc język angielski jest o ponad 37 razy częściej używany niż język polski.

W Internecie istnieje kilka rankingów, które podają najlepsze aplikacje do korekty tekstu w języku angielskim. Rankingi takie można spotkać na stronach https://www.lifewire.com/best-spelling-and-grammar-check-apps-4176088, https://www.makeuseof.com/tag/best-grammar-checker/ lub https://masterblogging.com/online-grammar-checker-tools/.

Najczęściej występujące aplikacje wspomniane w rankingach zostały opisane poniżej. Warto zwrócić uwagę, że opis każdej z aplikacji, w głównej mierze, pochodzi z jej własnej strony internetowej. Opisy te są wobec tego subiektywnymi opiniami autorów aplikacji i przedstawiają ich własne przeświadczenia.

- Grammarly (https://app.grammarly.com) – jedna z najbardziej znanych aplikacji służąca do korekty tekstu. Aplikacja jest dostępna online za pośrednictwem strony internetowej, jak również dostępna jest jako plugin do przeglądarek lub jako osobna aplikacja możliwa do zainstalowania na komputerach z systemem operacyjnym MacOS lub Windows. Sprawdza ona tekst w języku angielskim pod kątem błędów gramatycznych, ortograficznych i interpunkcyjnych.

  Jak podają twórcy aplikacji, oprócz sprawdzania błędów, poprawia ona często mylone słowa, gdy są używane w niewłaściwym kontekście. Tak więc, również i w tym przypadku mamy do czynienia z aplikacją, która działa w oparciu o kontekst wypowiedzi.

  Aplikacja ta występuje w dwóch wersjach – darmowej i płatnej. W wersji płatnej dostępne są dodatkowe sugestie dotyczące ulepszenia słownictwa, sprawdzanie stylu. Dostępny jest również detektor plagiatu, który sprawdza ponad 16 miliardów stron internetowych. Możliwe jest także wyłapywanie błędów kontekstowych oraz gramatycznych. Jak podają twórcy ponad 20 milionów ludzi założyło dotychczas konta w aplikacji (https://www.grammarly.com/better-writing).

- Ginger (https://www.gingersoftware.com/grammarcheck) - to aplikacja pozwalająca na sprawdzanie gramatyki, poprawę zdań, jak również na proponowanie następnych słów. Aplikacja dostępna jest w dwóch wersjach - darmowej i płatnej. Wersja płatna - premium

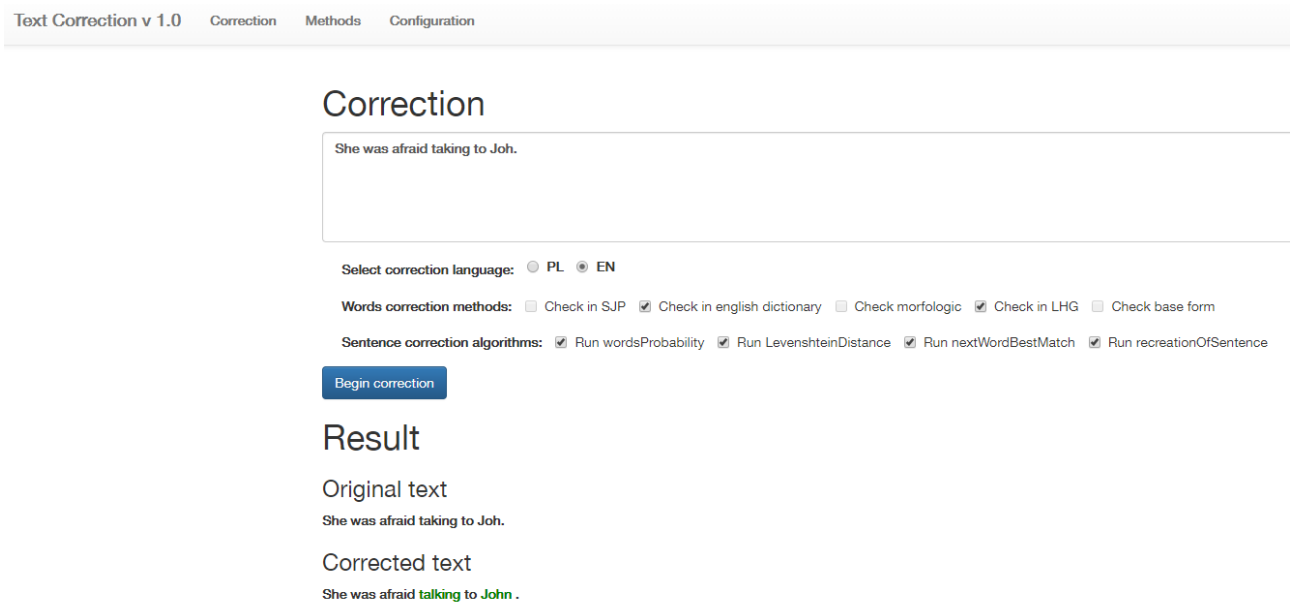
pozwala na nieograniczoną liczbę sprawdzeń gramatycznych, definicji, korekty zdań oraz analizy błędów. Jak zapewniają twórcy, aplikacja oferuje bardzo dobrą korektę tekstów już w wersji bezpłatnej.

- ProWritingAid (https://prowritingaid.com/) - jest kolejną aplikacją do sprawdzania gramatyki. Cechą charakterystyczną tej aplikacji jest, oprócz przeprowadzania korekty tekstu, możliwość wygenerowania raportu dla autora o stylu jego pracy. Aplikacja sprawdza bowiem tekst dodatkowo, np. pod względem powtórzeń. Zawiera ona informacje o ponad 5000 różnych ulepszeniach, które można wprowadzić, aby poprawić czytelność tekstu. Największym minusem jest fakt, że do korzystania z oprogramowania wymagana jest subskrypcja. Dostępna jest jednak dwutygodniowa, bezpłatna wersja próbna (z której skorzystano przy okazji przeprowadzanych porównań w tej pracy). Następnie wymagany jest zakup rocznej licencji.
- WhiteSmoke (https://www.whitesmoke.com) – na stronie internetowej można przeczytać, że jest to najbardziej zaawansowane narzędzie do poprawy gramatyki, ortografii, interpunkcji oraz stylu, które jest dostępne online. Niestety aplikacja ta występuje jedynie w wersji płatnej.
- Scribens (https://www.scribens.com/) - zgodnie z informacją na stronie internetowej aplikacja koryguje ponad 250 rodzajów typowych błędów gramatycznych i ortograficznych, w tym czasowniki, rzeczowniki, zaimki, przyimki, homonimy, interpunkcje, typografie oraz inne. Poprawia 10 razy więcej błędów niż jest w stanie poprawić Microsoft Word. Oferując zaawansowane oprogramowanie korekcyjne, jest ona w stanie znacznie polepszyć jakość pisania.
- Portal GrammarCheck (https://www.grammarcheck.net/editor/) - to kolejna aplikacja do korekty tekstu. Jest ona w stanie poprawić błędy ortograficzne, zasugerować zmianę stylu lub poprawić gramatykę. Aplikacja dostępna jest w dwóch wersjach: bezpłatnej i płatnej, w której jest w stanie odszukać jeszcze trudniejsze do wykrycia błędy, nawet do dziesięciu razy więcej niż popularne edytory tekstu.
- Portal OnlineCorrection (https://www.onlineCorrection.com) - narzędzie przeznaczone do wyszukiwania podstawowych błędów gramatycznych i stylistycznych w tekstach angielskich. Aplikacja jest całkowicie darmowa.

Zbadano zatem możliwości poprawy zdania „She was afraid of taking to Joh.” przez wskazane powyżej aplikacje. Dla porównania zaimplementowany algorytm kontekstowej korekty

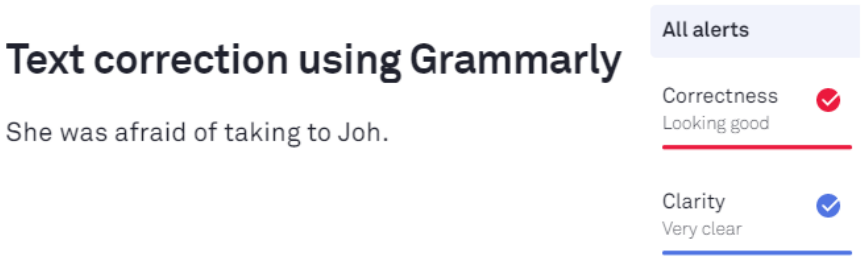
tekstu wykorzystujący wszystkie dostępne metody zauważył dwa błędy i poprawił błędny tekst na zdanie poprawne „She was afraid of talking to John.”.

Na rysunku 26. przedstawiono wynik poprawy w wykonanej aplikacji służącej kontekstowej korekcie tekstu.



Rysunek 26: Wynik poprawy tekstu w wykonanej aplikacji

W pierwszej kolejności przetestowana została korekta tekstu z wykorzystaniem aplikacji Grammarly. Na rysunku 27. przedstawiono wynik korekty.



Rysunek 27: Wynik poprawy tekstu przez aplikację Grammarly

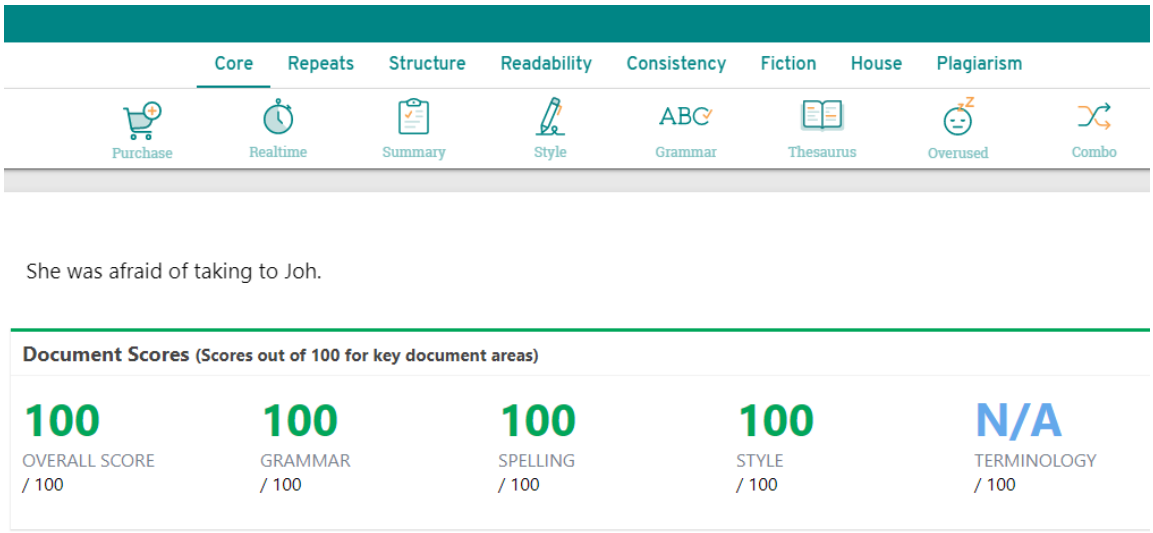
Jak można zauważyć, aplikacja ta nie znalazła żadnego błędu – ani dla słowa taking, ani dla słowa Joh.

Następnie przeprowadzono korektę tego samego tekstu przez aplikację Ginger. Na rysunku 28. przedstawiono wynik korekty.

Rysunek 28: Wynik poprawy tekstu przez aplikację Ginger

Jak można zauważyć, aplikacja ta prawidłowo zauważyła błąd w słowie taking. Zaproponowała ona również poprawną korektę tego wyrazu na wyraz talking. Niestety dla wyrazu Joh nie została zaproponowana korekta.

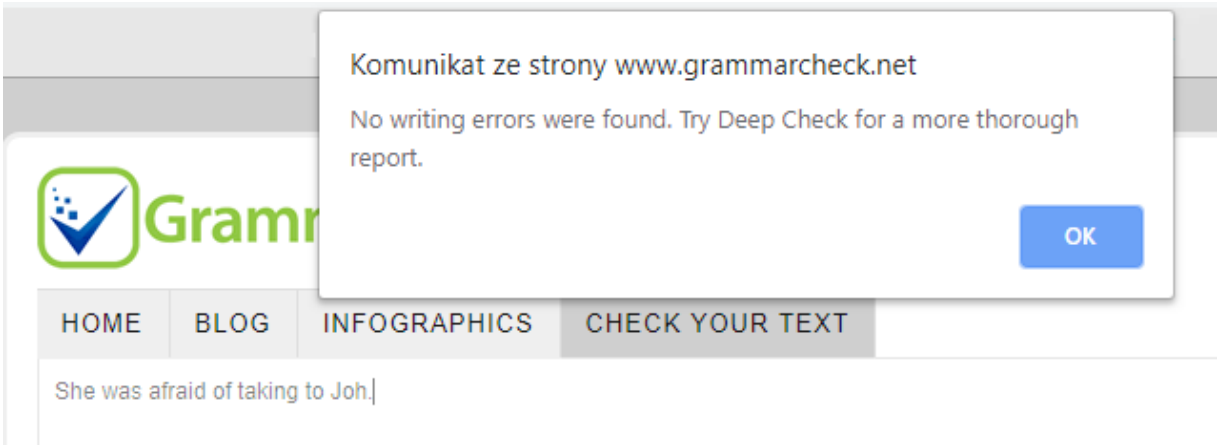
Kolejną aplikacją, której zadaniem była korekta niepoprawnego zdania była aplikacja ProWritingAid. Na rysunku 29. przedstawiono report po analizie i korekcie tekstu.



Rysunek 29: Wynik poprawy tekstu przez aplikację ProWritingAid

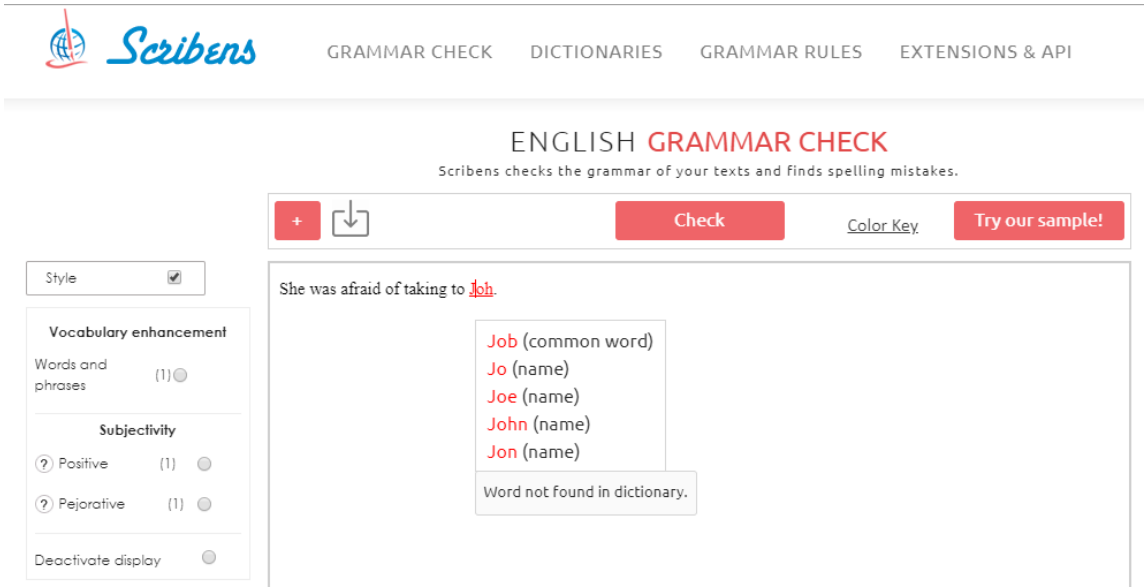
Również i ta aplikacja nie zauważyła żadnego błędu we wprowadzonym zdaniu, oceniając jego gramatykę, pisownię i styl na 100%.

Taki sam wynik jak dla aplikacji ProWritingAid otrzymano wprowadzając ten sam tekst do aplikacji znajdującej się na stronie https://www.grammarcheck.net Na rysunku 30. przedstawiono wynik korekty tekstu.



Rysunek 30: Wynik poprawy tekstu przez portal GrammarCheck

Dwie kolejne aplikacje – narzędzie Scribens oraz portal OnlineCorrection.com poprawnie rozpoznały błędy w drugim słowie – Joh. Niemniej zauważyć można, że również obie te aplikacje proponują zmianę słowa, np. na słowo job (pol. *praca*), które nie jest odpowiednie, jeśli chodzi o poprawność kontekstu wypowiedzi. Na rysunkach 31. oraz 32. przedstawiono wynik działania obu aplikacji.



Rysunek 31: Wynik poprawy tekstu przez aplikację Scribens

Rysunek 32: Wynik poprawy tekstu przez portal OnlineCorrection.com

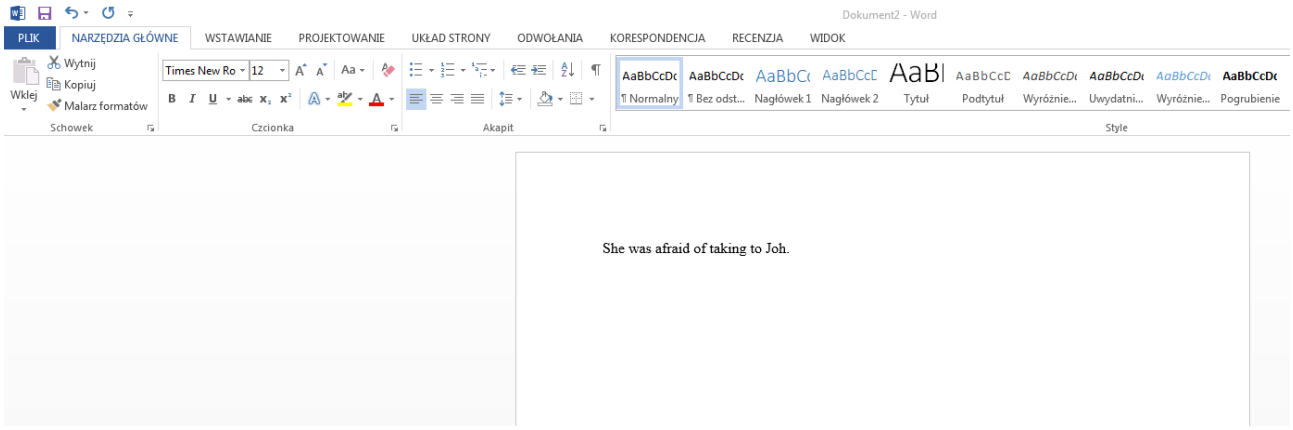
Wyniki działania wszystkich aplikacji zebrano w poniższej tabeli.

	Zauważony błąd w słowie „taking”	Zauważony błąd w słowie „Joh”
Algorytmy oparte na grafie LHG	TAK	TAK
Aplikacja Grammarly	NIE	NIE
Aplikacja Ginger	TAK	NIE
Aplikacja ProWritingAid	NIE	NIE
Portal GrammarCheck	NIE	NIE
Aplikacja Scribens	NIE	TAK
Portal OnlineCorrection.com	NIE	TAK

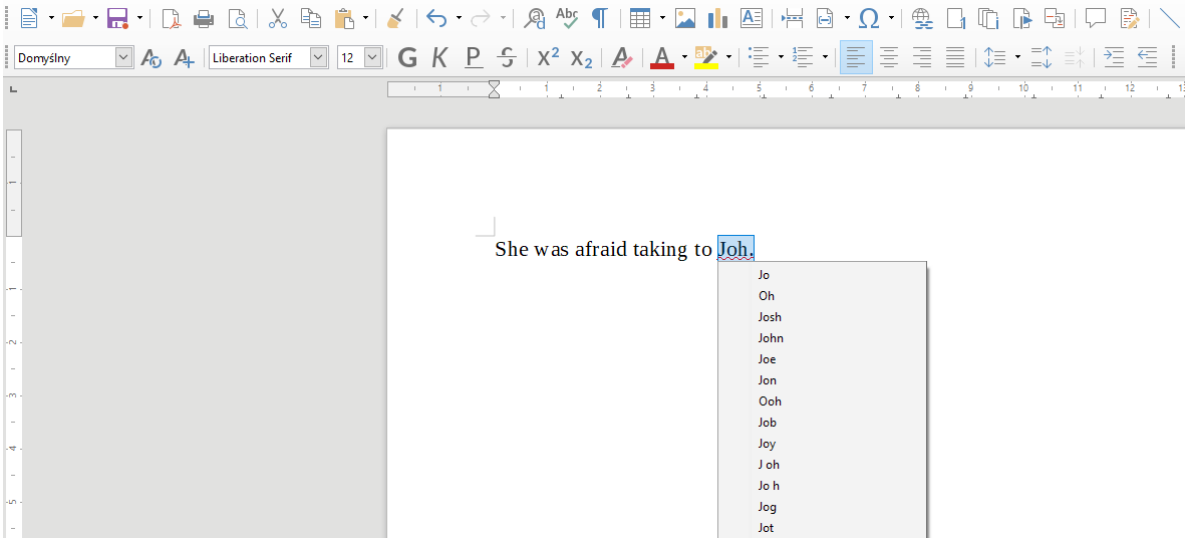
Jak można zauważyć, iż jedynie algorytm oparty na Grafie Przyzwyczajeń Lingwistycznych był w stanie znaleźć i zasugerować poprawę dla obu błędnych wyrazów. Pierwszy błąd, ze słowem taking, został zauważony jedynie przez aplikację Ginger. Drugi błąd, ze słowem Joh, został odszukany przez dwie aplikacje – Scribens oraz portal OnlineCorrection.com.

Testy te przeprowadzano na aplikacjach w wersjach bezpłatnych. Nie ma jednak pewności, czy, w wersjach płatnych aplikacje te byłyby w stanie odszukać i poprawić wszystkie błędy.

Kolejną częścią testu było sprawdzenie poprawy początkowego zdania przez edytory tekstu – Microsoft Word oraz LibreOffice Writer. Na rysunkach 33. oraz 34. przedstawiono otrzymane rezultaty.



Rysunek 33: Brak propozycji korekty tekstu dla aplikacji Microsoft Word 2013



Rysunek 34: Wynik poprawy tekstu przez edytor LibreOffice Writer

Jak można zauważyć aplikacja Microsoft Word 2013 nie znalazła żadnego błędu w tekście. Aplikacja LibreOffice Writer znalazła jeden błąd dla niepoprawnego słowa „Joh", jednak tylko niektóre zaproponowane sugestie są w stanie poprawnie skorygować zdanie.

Po przeprowadzeniu tych eksperymentów dokonano kolejnej weryfikacji możliwej korekty tekstów dla zdań, które zostały wprowadzone z różnego rodzaju błędami. W tych eksperymentach sprawdzono możliwość korekty tekstu w następujących sytuacjach:

- użycie małej litery w słowie, które występuje jedynie dużą literą,
- użycie błędnego przedimka dla języka angielskiego,
- pominięcie znaku końca zdania,
- pominięcie przecinka w zdaniu,
- sklejenie dwóch słów w jedno (pominięcie spacji),
- pominięcie słowa w zdaniu,
- błąd w słowie.

Tak jak poprzednio porównywano wpisany tekst z tekstem, na jaki powinien on zostać poprawiony, dla algorytmów opartych na Grafie Przyzwyczajeń Lingwistycznych, jak również dla popularnych aplikacji. Metody oparte na wykonanej sieci asocjacyjnej wykorzystywały bazę, w której znajdowało się ponad 3 miliony wszystkich typów neuronów oraz ponad 56 milionów połączeń ASEQ oraz ACON różnych typów i poziomów (łącznie). Wyniki te zostały zebrane w tabelach poniżej.

Zdanie wprowadzone z błędem: *'What a curious feeling!' said **alice**.*

Poprawne zdanie po korekcie tekstu: *'What a curious feeling!' said **Alice**.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	'What a curious feeling!' said **Alice**.	TAK
Aplikacja Grammarly	'What a curious feeling!' said **Alice**.	TAK
Aplikacja Ginger	'What a curious feeling!' said **Alice**.	TAK
Aplikacja ProWritingAid	Zaproponowana korekta:	NIE

	alive, alike, **Alice**, slice	
Portal GrammarCheck	Zaproponowana korekta: **Alice**, alive, alike	TAK/NIE – gdyż poprawna korekta znajduje się na pierwszej pozycji
Aplikacja Scribens	Zauważone błędne słowa **said** oraz **Alice**	NIE
Portal OnlineCorrection.com	Zaproponowana korekta: **Alice**, alive, alike, Aline, a lice	TAK/NIE – gdyż poprawna korekta znajduje się na pierwszej pozycji

Zdanie wprowadzone z błędem: *Oh, Joe, you're **a** angel.*

Poprawne zdanie po korekcie tekstu: *Oh, Joe, you're **an** angel.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	Oh, Joe, you're **an** angel.	TAK
Aplikacja Grammarly	Oh, Joe, you're **an** angel.	TAK
Aplikacja Ginger	Oh, Joe, you're **an** angel.	TAK
Aplikacja ProWritingAid	Oh, Joe, you're **an** angel.	TAK
Portal GrammarCheck	Oh, Joe, you're **an** angel.	TAK
Aplikacja Scribens	Oh, Joe, you're **an** angel.	TAK
Portal OnlineCorrection.com	Oh, Joe, you're **an** angel.	TAK

Zdanie wprowadzone z błędem: *What does this mean*

Poprawne zdanie po korekcie tekstu: *What does this mean**?***

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	What does this mean**?**	TAK
Aplikacja Grammarly	Brak informacji o błędzie	NIE

Aplikacja Ginger	What does this mean**?**	TAK
Aplikacja ProWritingAid	What does this mean**?**	TAK
Portal GrammarCheck	Brak informacji o błędzie	NIE
Aplikacja Scribens	What does this mean**?**	TAK
Portal OnlineCorrection.com	Brak informacji o błędzie	NIE

Zdanie wprowadzone z błędem: *Alice looked all round the table but there was nothing on it.*

Poprawne zdanie po korekcie tekstu: *Alice looked all round the table**,** but there was nothing on it.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	Alice looked all round the table**,** but there was nothing on it.	TAK
Aplikacja Grammarly	Brak informacji o błędzie	NIE
Aplikacja Ginger	Alice looked all round the table**,** but there was nothing on it.	TAK
Aplikacja ProWritingAid	Alice looked all round the table**,** but there was nothing on it.	TAK
Portal GrammarCheck	Brak informacji o błędzie	NIE
Aplikacja Scribens	Brak informacji o błędzie	NIE
Portal OnlineCorrection.com	Brak informacji o błędzie	NIE

Zdanie wprowadzone z błędem: *Well, **iknow** that.*

Poprawne zdanie po korekcie tekstu: *Well, **I know** that.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	Well, **I know** that.	TAK
Aplikacja Grammarly	Zaproponowana korekta: **I know**, know	TAK/ NIE
Aplikacja Ginger	Well, **I know** that.	TAK
Aplikacja ProWritingAid	Zaproponowana korekta:	NIE

	arguing, know, again	
Portal GrammarCheck	Zaproponowana korekta: **I know**, know	TAK/ NIE
Aplikacja Scribens	Błąd zauważony, brak poprawy	NIE
Portal OnlineCorrection.com	Zaproponowana korekta: know, **I know**	NIE/TAK

Zdanie wprowadzone z błędem: *Alice had no idea what do.*

Poprawne zdanie po korekcie tekstu: *Alice had no idea what **to** do.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	Alice had no idea what **to** do.	TAK
Aplikacja Grammarly	Alice had no idea what **to** do.	TAK
Aplikacja Ginger	Alice had no idea what **to** do.	TAK
Aplikacja ProWritingAid	Alice **did not understand** what do.	NIE
Portal GrammarCheck	Brak informacji o błędzie	NIE
Aplikacja Scribens	Brak informacji o błędzie	NIE
Portal OnlineCorrection.com	Brak informacji o błędzie	NIE

Zdanie wprowadzone z błędem: ***Luckil** for Alice.*

Poprawne zdanie po korekcie tekstu: ***Luckily** for Alice.*

	Zdanie po poprawie	Czy tekst został poprawiony?
Algorytmy oparte na grafie LHG	***Luckily** for Alice.*	TAK
Aplikacja Grammarly	***Luckily** for Alice.*	TAK
Aplikacja Ginger	***Luckily** for Alice.*	TAK
Aplikacja ProWritingAid	Zaproponowana korekta: **Luckily,** Largely, Local	NIE
Portal GrammarCheck	Zaproponowana korekta:	NIE

	Lucky, **Luckily,** Luck	
Aplikacja Scribens	***Luckily*** *for Alice.*	TAK
Portal OnlineCorrection.com	***Luckily*** *for Alice.*	TAK

Powyżej przedstawiono otrzymaną korektę tekstu dla siedmiu różnych rodzajów błędów językowych. Dla poszczególnych aplikacji uzyskano następujący wynik poprawy:

- Algorytmy oparte na grafie LHG – 7/7 = 100%,
- Aplikacja Grammarly – 4.5/7 = 64 %,
- Aplikacja Ginger – 7/7 = 100%,
- Aplikacja ProWritingAid – 3/7 = 43%,
- Portal GrammarCheck – 2/7 = 29%,
- Aplikacja Scribens – 3/7 = 42%,
- Portal OnlineCorrection.com – 3/7 = 42%.

Najlepszą poprawą błędów wykazały się wykonane semi-automatyczne metody analizy i kontekstowej korekty tekstu oparte na Grafie Przyzwyczajeń Lingwistycznych oraz aplikacja Ginger. Obie poprawiły teksty w 100% przypadków. Aplikacjami, dla których otrzymano również zadowalające wyniki, są Grammarly oraz nieco niżej ProWritingAid oraz Scribens wraz z portalem OnlineCorrection.com. Jak zostało wspomniane powyżej, są to wyniki uzyskane dla bezpłatnych wersji aplikacji. Uwzględniając ten aspekt, wynik dla wersji komercyjnych aplikacji może być lepszy.

Porównanie korekty tekstu z istniejącymi aplikacjami dla języka polskiego

Podobnie jak dla języka angielskiego, porównanie efektywności korekty tekstu przeprowadzono również dla języka polskiego. W tym celu do wykorzystano aplikacje wymienione poniżej. Warto zwrócić uwagę, że ich opisy są również subiektywnymi opiniami autorów aplikacji i przedstawiają ich własne przeświadczenia.

- Language Tool (https://languagetool.org/) - aplikacja ta wykrywa ponad 1000 błędów w polskich tekstach. Może posłużyć ona do korekty stylistycznej, gramatycznej, ortograficznej

oraz typograficznej. Wskazuje ona literówki kontekstowe, które są niewidoczne dla standardowych korektorów pisowni.

- iKorektor (https://ikorektor.pl) - internetowe narzędzie, które sprawdza tekst w języku polskim pod kątem błędów językowych i automatycznie poprawia ewidentne błędy ortograficzne, interpunkcyjne, typograficzne, literówki, uzupełnia polskie znaki diakrytyczne itp.
- KorektorOnline (https://www.korektoronline.pl) – kolejny serwis, który wyszukuje i zaznacza błędy pisowni, jak również wiele podstawowych błędów gramatycznych i stylistycznych w tekstach napisanych w języku polskim. Aplikacja sugeruje także możliwe poprawki w tekście. Warto wspomnieć, że autorzy zaznaczają, iż w swojej aplikacji wykorzystują narzędzia Hunspell, DICTION i LanguageTool. Ostatnie z nich stanowi osobną aplikację, która została opisana w pierwszym punkcie tej listy.
- Ortograf (https://www.ortograf.pl) - jest to system sprawdzający podany tekst pod względem poprawności ortograficznej, stylistycznej, gramatycznej i typograficznej. Dla języka polskiego ortografia i interpunkcja sprawdzane są pod kątem ponad 1000 reguł. Autorzy zaznaczają jednak, że znalezione błędy i propozycje poprawek należy traktować jedynie jako sugestie. Zastosowany algorytm nie znajduje wszystkich błędów i w wielu przypadkach nie jest w stanie zaproponować poprawnej pisowni. Użytkownik korzystając z witryny, robi to na własne ryzyko.

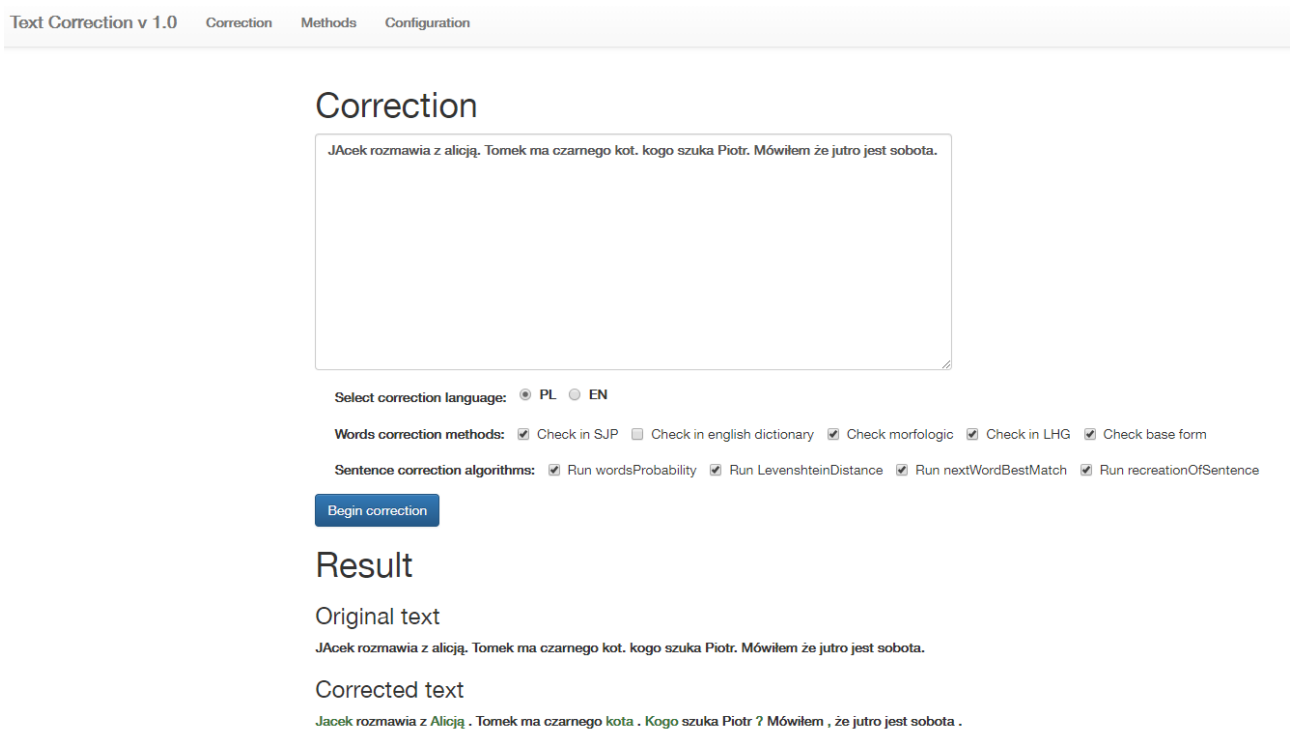
Podobnie jak dla języka angielskiego, w języku polskim wprowadzono tekst z różnymi typami błędów, celem sprawdzenia poprawności korekty tekstu. Tekst, który został poddany ocenie, brzmiał „JAcek rozmawia z alicją. Tomek ma czarnego kot. kogo szuka Piotr. Mówiłem że jutro jest sobota.”. Można w nim rozpoznać następujące błędy:

- JAcek – słowo znajduje się w słowniku j. polskiego, lecz zostało błędnie wpisane. Poprawnie powinno brzmieć „Jacek”;
- alicją – to słowo podobnie jak słowo Jacek, zostało źle wprowadzone. Powinno rozpoczynać się od dużej litery. Poprawnie powinno brzmieć „Alicją”;
- czarnego kot – w tym przypadku słowo „kot” powinno zostać odmienione i poprawnie powinno zostać zapisane „czarnego kota”;
- kogo – wyraz ten rozpoczyna zdanie, a więc powinien rozpoczynać się dużą literą;

- zdanie „kogo szuka Piotr.” jest zdaniem pytającym, a więc zamiast kropki powinno kończyć się znakiem zapytania „?”;
- Mówiłem że – przed słowem „że” stawiamy przecinek. Wobec tego poprawny początek tego zdania powinien zostać zapisany w następujący sposób „Mówiłem, że”.

Wyniki poprawy tekstu poszczególnych aplikacji zostały zaprezentowane poniżej.

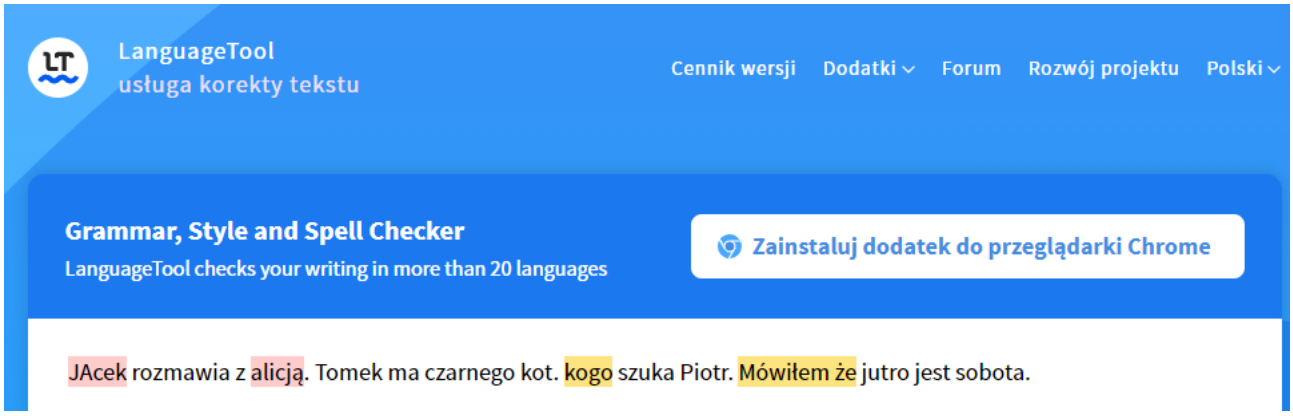
Wynik poprawy tekstu za pomocą wykonanej aplikacji przedstawiono na rysunku 35.



Rysunek 35: Wynik poprawy tekstu w języku polskim przeprowadzony przez wykonaną aplikację

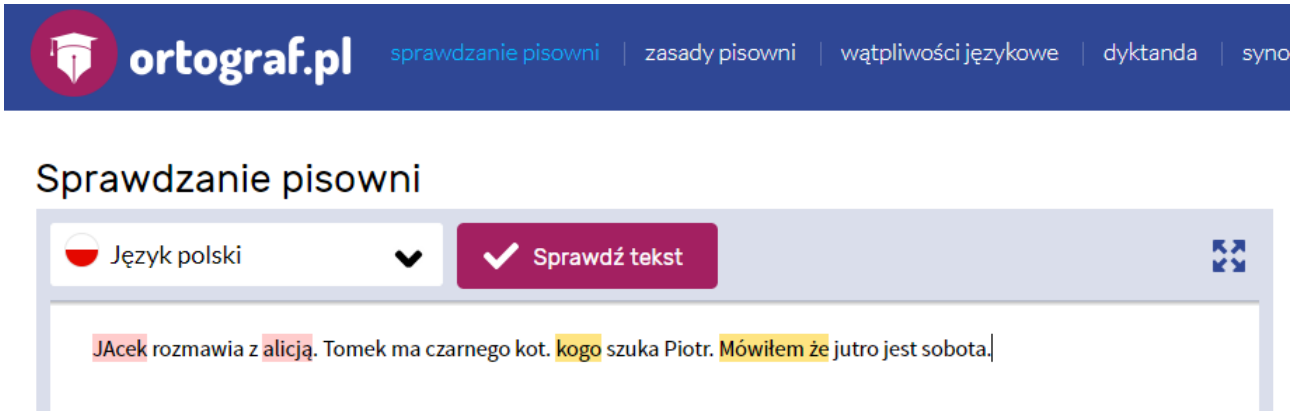
Jak można zauważyć, zaproponowana aplikacja poprawiła wszystkie błędy występujące w tekście.

Na kolejnych rysunkach 36., 37. oraz 38. przedstawiono wyniki korekty tekstu dla aplikacji LanguageTool, KorektorOnline.pl oraz portalu Ortograf.pl.



Rysunek 36: Znalezione błędy w tekście przez apllikację LanguageTool

Rysunek 37: Wynik korekty tekstu przez portal KorektorOnline.com

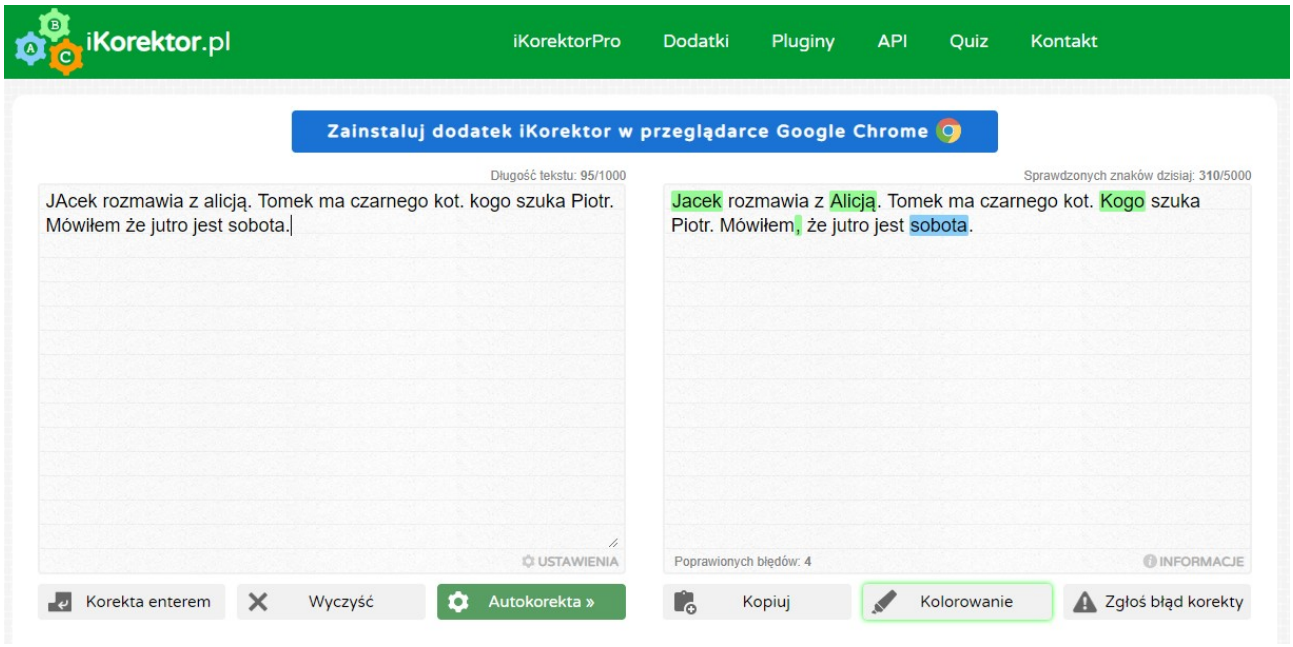


Rysunek 38: Uzyskany rezultat korekty tekstu w portalu ortograf.pl

Można również dostrzec, że zaprezentowana została ta sama korekta tekstu dla wszystkich trzech aplikacji. Sposób wyświetlania błędów jest również bardzo podobny. Może to być związane

z tym, iż aplikacja KorektorOnline.pl wprost wskazuje, że do swojej korekty tekstu wykorzystuje aplikację LanguageTool. Niestety aplikacje te zaznaczają jedynie błędne słowa, nie wprowadzając automatycznej korekty (choć po kliknięciu w słowo prezentują jedynie jedną opcję poprawy).

Ostatnią aplikacją, dla której przeprowadzono sprawdzenie korekty tekstu, był portal iKorektor.pl. Wynik działania aplikacji został zaprezentowany na rysunku 39. Program ten, w odróżnieniu od poprzednich wprowadza częściową automatyczną korektę tekstu. Niestety jako jedyny zaznacza jako błędne słowo „sobota", które jest poprawne.



Rysunek 39: Przeprowadzona korekta tekstu przez portal iKorektor.pl

W tabeli 3. zebrano szczegółowe informacje odnośnie wykrytych błędów oraz możliwości ich poprawy.

Tabela 3: Porównanie otrzymanych wyników korekty tekstu dla języka polskiego.

	Algorytmy oparte na grafie LHG	Aplikacje LanguageTool, KorektorOnline.pl, Ortograf.pl	Aplikacja iKorektor.pl
JAcek	**TAK**	**Jacek**, Jajek, Jarek,...	**TAK**
alicją	**TAK**	Alicja, **Alicją**, Alicjom	**TAK**
czarnego kot	**TAK**	NIE	NIE
kogo	**TAK**	**TAK**	**TAK**
Pytajnik na końcu zdania	**TAK**	NIE	NIE
Przecinek dla „mówiłem że"	**TAK**	**TAK**	**TAK**

sobota	**Nie wymaga poprawy**	**Nie wymaga poprawy**	sobotą

Dla poszczególnych aplikacji uzyskano następujący wynik poprawy:

- Algorytmy oparte na grafie LHG – 6/6 = 100%,
- Aplikacje LanguageTool, KorektorOnline.pl, Ortograf.pl – 3/6 = 50%,
- Aplikacja iKorektor.pl 4/6 = 66%, niemniej wykryty został dodatkowo niepoprawny błąd w słowie sobota. Tak więc można określić, że wynik poprawy to 4/7 = 57%.

Wyniki wskazują, że aplikacje służące poprawie tekstu dla języka polskiego działają nieco gorzej niż aplikacje służące korekcie tekstu dla języka angielskiego. Powodów takiego działania może być kilka. Jak zostało przedstawione wcześniej, język polski jest mniej powszechny niż język angielski. Dodatkowo uważa się, że język polski jest jednym z najtrudniejszych języków na świecie. UNESCO szacuje, że nawet 77 proc. Polaków ma problem ze zrozumieniem czytanego po polsku tekstu. Dużą trudność może sprawiać odmiana rzeczowników oraz deklinacja. Problematyczna jest też odmiana czasowników przez osoby czy wyrażanie czasu przeszłego, teraźniejszego i przyszłego. Nade wszystko trudne są zaimki i przyimki, zwłaszcza gdy osoba, która uczy się języka polskiego w swoim rodzimym języku, nie odmienia słów, a jedynie dodaje końcówkę, czy też nie używa odmian przez osoby. W języku polskim występuje też dużo synonimów, czyli wyrazów bliskoznacznych, ale również siostrzanych słów, które mimo że brzmią tak samo i zapisywane są tak samo, to w różnym kontekście mają różne znaczenie, np. komórka - telefon albo miejsce do przechowywania przedmiotów w domu [99].

Skonstruowany Graf Przyzwyczajeń Lingwistycznych wraz z metodami kontekstowej korekty tekstu, jak wykazano, jest w stanie działać efektywnie dla kilku języków. Zmiana języka nie wpływa znacząco na sposób konstrukcji grafu, czy też na opracowane metody. Są to duże zalety stosowania takiego rozwiązania do korekty tekstów.

Porównanie korekty tekstu dla dzieł literackich

W celu dokładniejszego sprawdzenia efektów korekty tekstu dla języka angielskiego i polskiego wykonano porównania dla kilku fragmentów tekstów pochodzących z powszechnie znanych książek, dostępnych w portalach https://www.gutenberg.org/ oraz https://wolnelektury.pl . W tekstach tych wprowadzono manualnie błędy, a następnie taki tekst poddano korekcie aplikacjom, które uzyskały najlepsze wyniki we wcześniejszych porównaniach. Wyniki tych

eksperymentów zaprezentowano w poniższych tabelach, na rysunkach natomiast zobrazowano dokonane w tekście zmiany.

Fragment angielskiej książki „Przygody Sherlocka Holmesa", autorstwa Arthura Conana Doylea. Fragment liczył 682 słowa, w których wykonano 45 błędów.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów
Algorytmy oparte na grafie LHG	45	100%
Aplikacja Grammarly	39	86%
Aplikacja Ginger	35	77%
Aplikacja ProWritingAid	37	82%
Aplikacja Scribens	37	82%

Na rysunku 40 zaznaczono błędy jakie naniesiono w tym fragmencie książki. Kolorem czerwonym zostały zaznaczone wszystkie znaki, które zostały usunięte z oryginalnego tekstu, natomiast kolorem zielonym zostały zaznaczone znaki, które zostały dodane w sprawdzanym tekście.

To Sherlock Holmes she is always the woman.

I have seldom heard him mention her under any other name.

In his eyes she eclipses and predominates the whole of her sex.

It was not that he felt any emotion akin to love for Irene Adler.

All emotions, and that one particularly, were abhorrent to his cold, precise but admirably balanced mind?.

He was, I take it, the most perfect reasoning and observing machine that the Wworld has seen, but as a lover he would have placed himself in a false position.

hHe never spoke of the softer passions, save with a gibe and a sneer.

tThey were admirable things for the observer-excellent for drawing the veil from mens motives and actions.

But for the trained reasoner to admit such intrusions into his own delicate and finely adjusted temperament was to introduce a distracting factor which might throw a doubt upon all his mental results.
Grit in a sensitive instrument, or a crack in one of his own high-power lenses, would not be more disturbing than an strong emotion in a nature such ass his.

And yet there was but one woman to him, and that woman was the late Irene Adler, of dubious and questionable memory.

I had seen little of hHolmes lately.

My Mmarriage HHad drifted us away from each other.

My own complete happiness, and the home-centred interests which rise up around the man who first finds himself master of his own establishment, were sufficient to absorb all my attention, while Holmes, who loathed every form of society with his whole Bohemian soul, remained in our lodgings in Baker Street, buried among his old books, and alternating from week to week between cocaine and ambition, the drowsiness of the drug, and the fierce energy of his own keen nature.
He wereas still, as ever, deeply attracted by the study of crime, and occupied his immense faculties and extraordinary powers of observation in following out those clues, and clearing up those mysteries which had been abandoned as hopeless by the official police.
fFrom time to time I heard some vague account of his doings: of his summons to oOdessa in the case of the Trepoff murder, of his clearing up of the singular tragedy of the aAtkinson brothers at Trincomalee, and finally of the mission which he head accomplished so delicately and successfully for the reigning family off h Holland.
Beyond these signs of his activity, however, which I merely shared with all the readers of the daily press, I knew little of my former friend and companion.
One night - it was on the twentieth of mMarch, 1888 - I was returning from a journey to a patient (for I had now returned to civil practice), when my way led me through Baker sStreet.
As I passed the well-remembered door, which must always be associated in my mind with my wooing, and with the dark incidents of the sStudy in Scarlet, I was seized with a keen desire to see Holmes again, and to know how he was employing his extraordinary powers.

His rooms were brillantly lit, and, even as I looked up, I saw his tall, spare figure pass twice in a dark silhouette against the blind?.

He was pacing the room swiftly, eagerly, with his head sunk upon his chest and his hands clasped behind him.

tTo me, who knew his every mood and habit, his attitude and manner told their own story.

He was at works again.

He haad risen out of his drug-created dreams and was hot upon the scent of some new problem.

I rang the bell and was shown up to the chamber which had formerly been in part my own.

His manner was not effusive.

It seldom was; but he wereas glad, I think, to see me.

With hardly a word spoken, but with a kindly eye, he waved me to an armchair, threw across his case of cigares, and indicated a spirit case and a gasogene in the corner.

Then hee stood before the fire and looked me over in his singular introspective fashion.

Rysunek 40: Zaznaczone błędy jakie naniesiono we fragmencie książki „Przygody Sherlocka Holmesa”

Fragment angielskiej książki „Alicja w krainie czarów”, autorstwa Lewisa Carrolla. Fragment liczył 764 słowa, w których wykonano 50 błędów.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów
Algorytmy oparte na grafie LHG	50	100%
Aplikacja Grammarly	44	88%
Aplikacja Ginger	33	66%
Aplikacja ProWritingAid	57	114%
Aplikacja Scribens	26	52%

Jak można zauważyć w tabeli powyżej dla aplikacji ProWritingAid uzyskano wynik 114%. Jest to efektem założeń dotyczących oceny poprawności korekty. W fragmencie książki „Alicja w krainie czarów” zostało dokonanych 50 błędów. Wobec czego, jeśli aplikacja wszystkie błędy zaznaczyła poprawnie to uzyskała wynik 100%. Aplikacja ProWritingAid zaznaczyła do poprawy 57 błędów. Jest to wynik o 7 błędów wyższy, niż liczba błędów faktycznie poprawionych. Narzędzie to w tekście oryginalnym zaznaczyło do poprawy już kilka błędów, wobec czego końcowy wynik jest tak wysoki. Autorzy aplikacji ProWritingAid zaznaczają, że oprócz korekty tekstu zawiera ona informacje o ulepszeniach, które można wprowadzić, aby poprawić czytelność tekstu.

Na rysunku 41 zaznaczono jakie błędy naniesiono w tym fragmencie książki. Kolorem czerwonym zostały zaznaczone wszystkie znaki, które zostały usunięte z oryginalnego tekstu, natomiast kolorem zielonym zostały zaznaczone znaki, które zostały dodane w sprawdzanym tekście.

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversations?'
So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.
There was nothing so very remarkable in that; nor did Alice think it so very much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually took a watch out of its waistcoat-pocket, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.
Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next.
First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs.
She took down a jar from one of the shelves as she passed; it was labelled 'ORANGE MARMALADE', but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.
'Well!' thought Alice to herself, 'after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home! Why, I wouldn't say anything about it, even if I fell off the top of the house!' (Which was very likely true.)

Down, down, down.

Would the fall never come to an end! 'I wonder how many miles I've fallen by this time?' she said aloud.

'I must be getting somewhere near the centre of the earth.

Let me see: that would be four thousand miles down, I think—' (for, you see, Alice had learnt several things of this sort in her lessons in the schoolroom, and though this was not a very good opportunity for showing off her knowledge, as there was no one to listen to her, still it was good practice to say it over) '—yes, that's about the right distance—but then I wonder what Latitude or Longitude I've got to?' (Alice had no idea what Latitude was, or Longitude either, but thought they were nice grand words to say.)

Presently she began again.

'I wonder if I shall fall right through the earth! How funny it'll seem to come out among the people that walk with their heads downward! The Antipathies, I think—' (she was rather glad there was no one listening, this time, as it didn't sound at all the right word) '—but I shall have to ask them what the name of the country is, you know.
Please, Ma'am, is this New Zealand or Australia?' (and she tried to curtsey as she spoke—fancy curtseying as you're falling through the air! Do you think you could manage it?) 'And what an ignorant little girl she'll think me for asking! No, it'll never do to ask: perhaps I shall see it written up somewhere.'

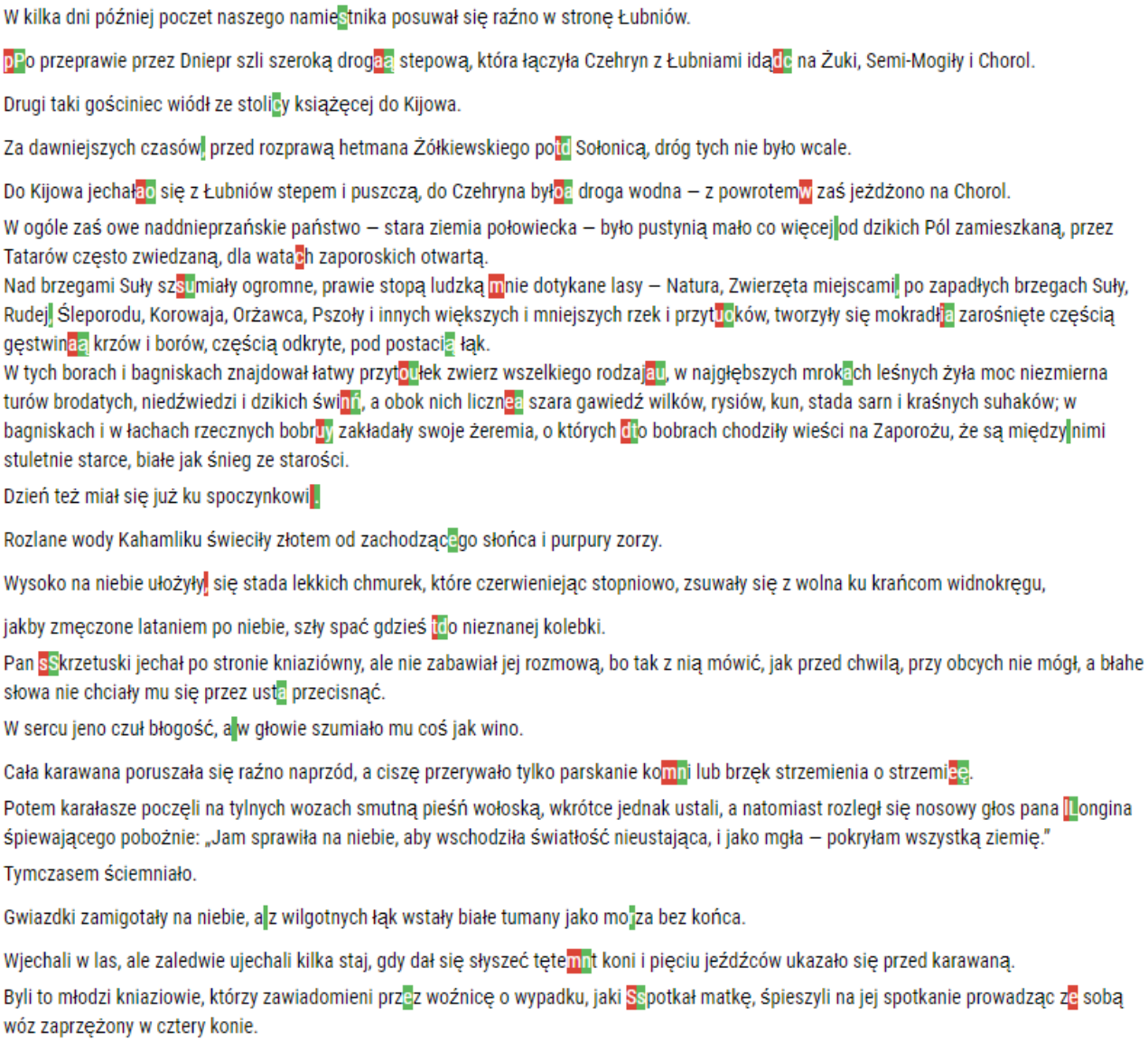
Rysunek 41: Zaznaczone błędy jakie naniesiono we fragmencie książki „Alicja w krainie czarów"

Fragment powieści „Ogniem i Mieczem", autorstwa Henryka Sienkiewicza. Fragment ten liczył 422 słowa, w którym wprowadzono 45 błędów.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów
Algorytmy oparte na grafie LHG	45	100%
Aplikacja LanguageTool	28	62%
Portal iKorektor.pl	22	48%
Aplikacja MS Word 2013	22	48%

Aplikacja LifreOffice Writer	21	46%

Na rysunku 42 zaznaczono jakie błędy naniesiono w tym fragmencie powieści. Kolorem czerwonym zostały zaznaczone wszystkie znaki, które zostały usunięte z oryginalnego tekstu, natomiast kolorem zielonym zostały zaznaczone znaki, które zostały dodane w sprawdzanym tekście.


W kilka dni później poczet naszego namiestnika posuwał się raźno w stronę Łubniów.

pPo przeprawie przez Dniepr szli szeroką drogaą stepową, która łączyła Czehryn z Łubniami idąc na Żuki, Semi-Mogiły i Chorol.

Drugi taki gościniec wiódł ze stolicy książęcej do Kijowa.

Za dawniejszych czasów przed rozprawą hetmana Żółkiewskiego pod Sołonicą, dróg tych nie było wcale.

Do Kijowa jechałao się z Łubniów stepem i puszczą, do Czehryna byłoa droga wodna — z powrotemw zaś jeżdżono na Chorol.

W ogóle zaś owe naddnieprzańskie państwo — stara ziemia połowiecka — było pustynią mało co więcej od dzikich Pól zamieszkaną, przez Tatarów często zwiedzaną, dla watach zaporoskich otwartą.
Nad brzegami Suły szumiały ogromne, prawie stopą ludzką nie dotykane lasy — Natura, Zwierzęta miejscami po zapadłych brzegach Suły, Rudej, Śleporodu, Korowaja, Orżawca, Pszoły i innych większych i mniejszych rzek i przytuków, tworzyły się mokradła zarośnięte częścią gęstwinaą krzów i borów, częścią odkryte, pod postacią łąk.
W tych borach i bagniskach znajdował łatwy przytułek zwierz wszelkiego rodzajau, w najgłębszych mrokach leśnych żyła moc niezmierna turów brodatych, niedźwiedzi i dzikich świń, a obok nich licznea szara gawiedź wilków, rysiów, kun, stada sarn i kraśnych suhaków; w bagniskach i w łachach rzecznych bobry zakładały swoje żeremia, o których dto bobrach chodziły wieści na Zaporożu, że są między nimi stuletnie starce, białe jak śnieg ze starości.

Dzień też miał się już ku spoczynkowi.

Rozlane wody Kahamliku świeciły złotem od zachodzącego słońca i purpury zorzy.

Wysoko na niebie ułożyły się stada lekkich chmurek, które czerwieniejąc stopniowo, zsuwały się z wolna ku krańcom widnokręgu,

jakby zmęczone lataniem po niebie, szły spać gdzieś do nieznanej kolebki.

Pan sSkrzetuski jechał po stronie kniaziówny, ale nie zabawiał jej rozmową, bo tak z nią mówić, jak przed chwilą, przy obcych nie mógł, a błahe słowa nie chciały mu się przez usta przecisnąć.

W sercu jeno czuł błogość, a w głowie szumiało mu coś jak wino.

Cała karawana poruszała się raźno naprzód, a ciszę przerywało tylko parskanie komni lub brzęk strzemienia o strzemię.

Potem karałasze poczęli na tylnych wozach smutną pieśń wołoską, wkrótce jednak ustali, a natomiast rozległ się nosowy głos pana Longina śpiewającego pobożnie: „Jam sprawiła na niebie, aby wschodziła światłość nieustająca, i jako mgła — pokryłam wszystką ziemię."

Tymczasem ściemniało.

Gwiazdki zamigotały na niebie, a z wilgotnych łąk wstały białe tumany jako morza bez końca.

Wjechali w las, ale zaledwie ujechali kilka staj, gdy dał się słyszeć tętent koni i pięciu jeźdźców ukazało się przed karawaną.

Byli to młodzi kniaziowie, którzy zawiadomieni przez woźnicę o wypadku, jaki Sspotkał matkę, śpieszyli na jej spotkanie prowadząc ze sobą wóz zaprzężony w cztery konie.

Rysunek 42: Zaznaczone błędy jakie naniesiono we fragmencie powieści „Ogniem i Mieczem"

Fragment powieści „Syzyfowe prace", autorstwa Stefana Żeromskiego. Fragment ten liczył 459 słów, w którym wprowadzono 50 błędów.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów

Algorytmy oparte na grafie LHG	50	100%
Aplikacja LanguageTool	36	72%
Portal iKorektor.pl	29	58%
Aplikacja MS Word 2013	31	62%
Aplikacja LifreOffice Writer	24	48%

Na rysunku 43 zaznaczono jakie błędy naniesiono w tym fragmencie powieści. Kolorem czerwonym zostały zaznaczone wszystkie znaki, które zostały usunięte z oryginalnego tekstu, natomiast kolorem zielonym zostały zaznaczone znaki, które zostały dodane w sprawdzanym tekście.


Termin odstawienia Marcina do szkoły przypadł na dzień czwarty Sstycznia.

Obydwoje państwo bBorowiczowie postanowili odwieźć jedynaka na miejsce.

Zaprzężono konie do malowanych i kutych sanek, główne siedzenie wysłanao barwnym, strzyżonym dywanem, który zazwyczaj wisiał nad łóżkiem pani, i około pierwszej z południa wśród powszechnego płaczu wyruszonao.

Dzień był wietrzny i mroźny?.

Mimo to jednak, że szczyty wzgórz kurzyły się nieustannie od przelatującej zadymki na rozległych dolinach, między lasami, zmarznięte pustkowia leżałoy w spokoju i prawie w ciszy.

Szedł tylko tamtędy zimny przeciąg, wiejąc szypki śnieg niby lotną plewę.

Gdzieniegdzie wałęsały się nad zaspami smugi najdrobniejszego Pyłu jak pyłek przyduszonego Ppaleniska.

Chłopak siedzący na koźle, podobny do głowy cukru opakowanej szarą bibułą, w swym spiczastym baszłyku, który w tamtych okolicach od dawien dawna uległ nostryfikacji i otrzymał swojską nazwę maślocha, i w brunatnej sukmanie – mocno trzymał lejce garściami, ukrytymi w niezmiernych rękawicach wełnianych o jednym wielkim palcu.
Konie były wypoczęte, nie chodziły bowiem od pewnego już czasu do żadnej ciężkiej roboty, toteż pomykały, parskając, ostrego kłusa po ledwo przetartej, a już znowu na pół zadętej drożynie, i sucho, jednostajnie trzaskały podkowami o nadmarzniętą zwierzchnią skorupę Śniegu.
Pan wWalenty Borowicz ćmił fajkę na krótkim cybuszku, wychylał się com kilka minut na bok i przyglądał uważnie już to sanicom, już migającym kopytom.

Wiatr go chłostał pod zaczerwienionej twarzy i on to zapewne wyciskał zowe łzy, które szlachcic ukradkiem ocierał.

pPani Borowiczowa nie siliła się wcale na maskowanie wzruszenia.

Łzy stały bez przerwy w jej oczach skierowanych Na syna.

Twarz ta, niegdyś piękna, a w owej chwili wyniszczona już bardzo przez troski i chorobę piersiową, miała niezwykły wyraz namysłu czyś jakiejś głębokiej a gorzkiej rozwagi.

Malec siedział „w nogach", tyłem do koni.

bBył to duży, tęgi i muskularny chłopak ośmioletni, z twarzą nie tylae piękną, ile rozumną i miłą.

Oczy miał czoarne, połyskliwe, w cieniu gęstych brwi ukryte.

Włosy krótko przystrzyżone „na jeża" okrywała barankowa czapka wciśnięta aż na uszy.

Strój mMiał na sobie zgrabną bekieszę z futrzanym kołnierzem i wełniane rękawiczki.

Włożono nań ten strój odświętny, za którym tak przepadał, ale za to wieziono go do szkoły.

Z niemego smutku matki, z Mminy ojca udającego dobry chumor wnioskował doskonale, że w owej szkole, którą mu tak zachwalano, przyobiecanych rozkoszy będzie nie tak znowu dużo.
Znajomy widok wioski rodzinnej znikł mu prędko z oczu; nagie wierzchołki lip stojących przed dworem, schyliły się nza brzeg lasu obwieszonego kiściami śniegu... Najbliższa góra poczęła wykręcać się, zmieniać, jakby krzywić i dziwaczniae garbić.
Wypadały teraz przed jego oczy smugi zarośli, jakich jeszcze nigdy nie widział, płoty z sękatych, nieociosanych zzerdzi, na których wisioały przedziwne, niezmiernie długie sople lodu, wynurzały się pewne obszary puste, gdzieniegdzie okryte lodami o barwie sinawej, zimnej, i dzikiej.

Niekiedy las z nagła podbiegał ku drodze i odkrywał przed zdumionymi oczyma chłopca posępne swoje głębie.

Rysunek 43: Zaznaczone błędy jakie naniesiono we fragmencie powieści „Syzyfowe prace"

Po przeprowadzeniu porównania korekty tekstu dla fragmentów książek wykonano badanie korekty tekstu dla dłuższego tekstu. W eksperymentach przedstawionych powyżej wykazano, że lepszą skuteczność poprawy błędów w tekstach posiadają aplikacje dla języka angielskiego. Wobec czego do badania poprawności korekty tekstu wybrano książkę Marka Twaina pt. „Przygody Tomka Sawyera". Badany tekst liczył 5037 wyrazów. We fragmencie tym wprowadzono losowo łącznie 285 następujących błędów:

- 150 razy przestawiono dwie litery w słowie;

- 50 razy dokonano zamiany dużej litery na małą;
- 25 razy dokonano zamiany małej litery na dużą;
- 25 razy połączono dwa wyrazy w jeden;
- 25 razy usunięto przecinek;
- 10 razy zamieniono kropkę jako znak końca zdania na znak zapytania.

Wynik działania aplikacji przedstawiono w tabeli poniżej.

	Liczba znalezionych błędów	Procentowa liczba znalezionych błędów
Algorytmy oparte na grafie LHG	273	96%
Aplikacja Grammarly	251	88%
Aplikacja Ginger	238	84%
Aplikacja ProWritingAid	232	81%
Aplikacja Scribens	215	75%

W powyższym przykładzie zaimplementowane algorytmy oparte na grafie LHG znalazły 273 błędy spośród 285 wszystkich błędów. Wśród dwunastu nieznalezionych błędów:

- 2 razy nie rozpoznano przestawienia dwóch liter w słowie;
- 6 razy nie rozpoznano usunięcia przecinka;
- 3 razy nie rozpoznano zamiany dużej litery na małą;
- 2 razy nie rozpoznano zamiany małej litery na dużą;
- 2 nie rozpoznano zamiany kropki na znak zapytania.

Pomimo wyniku 96% algorytmy oparte na Grafie Przyzwyczajeń Lingwistycznych dały bardzo dobry wynik, gdyż pozostałe algorytmy znalazły mniejszą liczbę błędów.

Przykłady opisane powyżej oraz zaproponowane poprawy zdań, raz jeszcze pokazują, że korekta tekstu nie jest zadaniem prostym. W większości przypadków jej wynik zależy w bezpośredni sposób od zastosowanych wzorców i ich ilości. Jakość przeprowadzanej korekty wynika bowiem z liczby reguł, do których będzie pasować wprowadzone zdanie. Moduły sprawdzające tekst bazują w bardzo ograniczony sposób na samej gramatyce rozumianej jako poprawna struktura zdań. Składnia badanych języków jest niezwykle skomplikowana. Nie ma również gotowego zestawu programowalnych reguł dla tych języków, które mogłyby zagwarantować poprawność działania w każdym przypadku. Graf Przyzwyczajeń Lingwistycznych

potrafi w satysfakcjonującym stopniu odtworzyć kontekst zdań, które zostały wcześniej w nim zapisane. Dzięki wykorzystaniu połączeń asocjacyjnych sekwencyjnych oraz kontekstowych (zarówno „w przód", jak i „w tył") możliwa jest poprawa wyrazów występujących zarówno na początku zdania, w jego środku jak i na końcu.

Ludzie rozumieją język i gramatykę dzięki regułom i wzorom, których uczą się zarówno pośrednio (np. poprzez środowisko i przykłady, które zapamiętują), jak i jawnie (np. w szkole). To dzięki swojemu doświadczeniu są oni w stanie połączyć reguły języka z wiedzą o tym, jak działają rzeczy w świecie. Aplikacje, usiłujące przeprowadzić poprawną korektę tekstu, próbują działać w podobny sposób. Dysponują bowiem również zbiorem reguł i zasad, brak im jednak wiedzy o semantyce i zrozumieniu. W związku z tym sprawdzanie gramatyki dla aplikacji polega w dużej mierze właśnie na zrozumieniu reguł i wyjątków od nich. Na wstępie tworzony jest model języka, który najczęściej jest probabilistycznym modelem opartym na zdaniach wprowadzanych do systemu komputerowego. Istnieją również modele opierające swoje działanie na regułach. Dostępnych jest wiele sposobów konstrukcji takich modeli, niemniej jednak główną ideą jest zasilenie ich przez dobrze ustrukturyzowany oraz poprawny gramatycznie tekst.

Oprócz dużej liczby błędów ortograficznych, które można dość prosto wykryć, istnieje niezliczona liczba błędów wynikających z kontekstu zdań. Metody oparte na Grafie Przyzwyczajeń Lingwistycznych również nie działają w 100% przypadku. „Wiedza" dotycząca kontekstów słownych i ich użycia wynika z bezpośredniego przeanalizowania i dodania danego kontekstu słownego w przeszłości. Jeśli dany kontekst lub bardzo podobny nie wystąpił wcześniej, kilka z zaproponowanych algorytmów nie będzie poprawnie działać. Skuteczność działania aplikacji, w gruncie rzeczy, zależy również od ilości opracowanych metod i sposobu ich połączenia. Często bowiem okazywało się, że dopiero analizując poszczególne wyniki z każdej metod i wybierając sugestię, która pojawiała się najczęściej, dokonywano poprawnej korekty tekstu.

Ponieważ liczba możliwych błędów w pisowni, użyciu i składni dla języka jest nieskończona, zbliżenie się do ich całościowego zidentyfikowania i rozwiązania poprzez moduły korekcji tekstów jest zadaniem niezwykle trudnym. Wiele programów sprawdzających gramatykę oferuje pomoc w korekcie tekstu bądź w poprawie stylu poprzez wskazywanie kilku możliwych rozwiązań. Natomiast to do autora należy zazwyczaj ostateczna decyzja, jak zdanie powinno finalnie wyglądać. Warto zwrócić uwagę, że każdy użytkownik będzie wprowadzał tekst charakterystyczny dla siebie, a więc będą istnieć różne błędy językowe. Każdy bowiem człowiek używa właściwego dla siebie zestawu wyrazów [100].

Rozdział 6

Podsumowanie

Do przeprowadzenia badań wykorzystano teksty pochodzące z różnych źródeł, na ich podstawie uzyskując konteksty wypowiedzi na różne tematy. Podczas analizy dostępu do korpusów słownych zauważono, że dostępne są serwisy, które posiadają darmowe cyfrowe edycje książek znanych autorów. Jednym z takich serwisów jest Projekt Gutenberg (https://www.gutenberg.org/) oferujący ponad 59 tysięcy darmowych eBooków. Teksty pochodzące z takiego źródła posiadają kilka zalet – są one powszechnie znane, są dobrze napisane (nie powinny w nich wystąpić błędy), co jest niezmierne ważne podczas „uczenia" grafu poprzez zasilanie go nowymi tekstami, dodatkowo istnieje zazwyczaj wiele tekstów jednego autora. Dzięki tak dużej liczbie książek możliwe było uzyskanie bardzo bogatego i różnorodnego kontekstu wypowiedzi.

W kolejnym kroku przeanalizowano, a następnie przetworzono pozyskane teksty i zapisano relacje pomiędzy kolejnymi słowami w taki sposób, aby „zrozumieć" czytane zdanie w danym kontekście słownym. Dodatkowo opracowano budowę specjalistycznych grafów lingwistycznych, które zostały nazwane Grafami Przyzwyczajeń Lingwistycznych. W celu zapisu tekstów do grafu, opracowano metody, które umożliwiły automatyczne pozyskiwanie tekstów z plików różnego formatu. Podstawową zaletą skorzystania z grafów LHG jest posiadanie informacji o kontekście wypowiedzi, poprzez zastosowanie specjalistycznych połączeń asocjacyjnych oraz o ilości wystąpień poszczególnych słów w danym kontekście.

Zastosowanie Grafu Przyzwyczajeń Lingwistycznych umożliwia badanie fleksyjno-częstotliwościowego kontekstu słownego. Jak zauważono wykorzystanie teorii grafów dynamicznie wzrasta w dziedzinach takich, jak chemia, lingwistyka, geografia, architektura. Ponadto grafy są potężnym narzędziem, jeśli chodzi o możliwość wizualizacji danych w nich zebranych.

Jedną z kluczowych inspiracji podczas tworzenia modelu języka opartego na strukturze grafowej był model ludzkiego mózgu, który dzięki swojej konstrukcji jest niezwykle wydajny. Podobnie jak dla ludzkiego mózgu proces uczenia, tak dodawanie nowych kontekstów do grafu jest operacją bardzo czasochłonną. Podczas czytania tekstów dla każdego zdania, tworzona jest najpierw tablica ze słowami oraz ich cechami. Każdy element tablicy jest dodawany do grafu. Słowo traktowane jest jako nowy wierzchołek (neuron słowny), a znaki interpunkcyjne stanowią o jego właściwościach. Dla każdego wierzchołka tworzona jest krawędź (asocjacja sekwencyjna) do następnego wierzchołka (neuronu słownego). Jeśli dla jakiegokolwiek wierzchołka będącego w grafie występuje niejednoznaczność (z wierzchołka wychodzą lub do wierzchołka wchodzą dwie

lub więcej krawędzi danego typu asocjacji) to tworzona jest krawędź asocjacji kontekstowej wyższego rzędu. Podczas analizowania kolejnych zdań zdarza się, że dodane krawędzie powodują powstanie niejednoznaczności dla poprzednio dodanych zdań. W związku z tym algorytm po analizie ostatnio dodanego zdania może kilka razy przeanalizować raz jeszcze niektóre z poprzednich zdań w poszukiwaniu kolejnych niejednoznaczności. Tak więc, gdy w grafie zapisywanych jest coraz więcej zdań, coraz częściej zdarza się wystąpienie niejednoznaczności, która w późniejszym etapie powinna zostać poprawiona. Gdy graf zostanie zasilony odpowiednią ilością tekstów, można w bardzo szybki sposób z niego skorzystać. Wystarczy wyszukać odpowiednie słowo (które jest dostępne w stałym czasie), a następnie poruszać się po poszczególnych połączeniach w grafie, których liczba (wchodząca lub wychodząca z określonego neuronu) jest skończona. Podobną właściwość można zauważyć w ludzkim mózgu, którego połączenia pomiędzy neuronami są wielokrotnie aktywowane w ciągu sekundy. Kolejnym podobieństwem do mózgu jest fakt, że Graf Przyzwyczajeń Lingwistycznych nie jest strukturą zamkniętą, która raz utworzona nie może być rozwijana w przyszłości. W każdym momencie graf LHG można rozszerzyć o kolejne zdania, a jego rozbudowa automatycznie poprawia wyniki korekty tekstów. W tym celu wystarczy uruchomić procedurę odpowiedzialną za przetworzenie kolejnych korpusów tekstu. Z metody tej skorzystano podczas korekty tekstu. Gdy użytkownik wprowadzi zdanie, które zostanie wstępnie poprawione poprzez zasugerowanie kilku opcji do wyboru, a użytkownik wybierze jedną z propozycji (tym samym uzna, że zaproponowana korekta jest prawidłowa) zdanie takie zostaje dodane do grafu jako kolejny prawidłowy kontekst. Daje to niesamowitą zdolność dodawania nowych zdań, które nie zostały do tej pory przeanalizowane, a są poprawne i mogą zasilić opracowany model.

Dzięki zastosowaniu wyspecjalizowanej struktury do przechowywania zdań, możliwe było przeprowadzenie w dalszym etapie semi-automatycznej kontekstowej korekty różnych tekstów z zastosowaniem wiedzy zebranej i powiązanej wcześniej. Jeden z algorytmów wykorzystuje równoczesne „pobudzenia” wielu wierzchołków poprzez wspomniane połączenia asocjacyjne. Każde z połączeń pobudza połączony z nim wierzchołek na taki czas, jakiego rzędu ono jest, z siłą odwrotnie proporcjonalną do jego rzędu. Na bazie cech słowa zapisanego w grafie został opracowany kolejny algorytm służący poprawie interpunkcji. Głównym celem tego algorytmu jest sprawdzanie, czy dane słowo może zaczynać się od małej bądź dużej litery, czy po danym wyrazie powinien wystąpić przecinek lub inny znak interpunkcyjny. Dodatkowo algorytm ten sprawdza, czy pierwsze słowo z zdaniu zaczyna się od dużej litery, a kończy się kropką, wykrzyknikiem lub pytajnikiem. Następnym algorytmem, jaki został zaprojektowany i zaimplementowany jest algorytm służący do weryfikacji poprawności i kolejności występowania słów. W celu oceniania

poprawności słów posłużono się badaniem częstości występowania słowa w zaproponowanym grafie, jak również informacjami pochodzącymi ze słownika danego języka. Dzięki temu, podczas wprowadzania tekstu możliwe jest oznaczanie słów, które występują w grafie oraz zostały odnalezione w słowniku jako poprawne, a słowa które nie zostały znalezione oznaczane są jako potencjalnie błędne.

W przeprowadzonych porównaniach z innymi aplikacjami, służącymi korekcie tekstu, wykazano, że metody oparte na Grafie Przyzwyczajeń Lingwistycznych dają bardzo dobre rezultaty dla różnych typów błędów. Zauważono również, że korekta tekstu jest zadaniem bardzo trudnym, z uwagi na stały rozwój języka oraz mnogość form, w jakich mogą wystąpić poprawne zdania, dlatego też Graf Przyzwyczajeń Lingwistycznych bazuje na różnych metodach, których połączenie daje lepsze wyniki niż zastosowanie poszczególnych algorytmów osobno.

W pracy wykazano, że możliwym jest zbudowanie efektywnej, innowacyjnej struktury reprezentującej model języka. Taką strukturę osiągnięto dzięki zastosowaniu modelu asocjacyjnego. Dodatkowo na podstawie korpusów tekstów oraz algorytmów ich efektywnej analizy, jak również z zastosowaniem wiedzy zebranej w Grafie Przyzwyczajeń Lingwistycznych, zostało opracowanych kilka metod, służących przeprowadzeniu semi-automatycznej kontekstowej korekty różnych tekstów. Porównanie ich efektywności działania z komercyjnie dostępnymi aplikacjami przyniosło zadowalające efekty.

Przeprowadzone badania i eksperymenty potwierdziły tezę, iż możliwe jest takie zbudowanie specjalistycznych grafów lingwistycznych na podstawie korpusów tekstów oraz algorytmów ich efektywnej analizy, żeby było możliwe przeprowadzenie semi-automatycznej kontekstowej korekty różnych tekstów z zastosowaniem wiedzy zebranej w tych grafach wykorzystując w trakcie ich tworzenia liczbę wystąpień poszczególnych słów w kontekście korygowanego tekstu.

Algorytmy wraz z zaproponowaną strukturą służącą zapisowi korpusów tekstu posłużyły do automatycznej korekty wprowadzanego tekstu, co obrano za cel podejmowanych w pracy badań.

Oryginalnymi osiągnięciami przeprowadzonych badań są:

- efektywny model grafowy służący do zapisu wyrazów oraz znaków interpunkcyjnych występujących w badanym tekście;
- opracowane metody umożliwiające pozyskanie tekstów z kilku źródeł w celu zbudowania podanego grafu;
- opracowane i wdrożone niezależne metody służące do analizy i kontekstowej korekty tekstu;

- wykonana aplikacja webowa (serwis internetowy), która umożliwia skorzystanie z zaimplementowanych algorytmów.

Problem korekty tekstu jest obecnie szeroko znanym i ważnym problemem w świecie. Zaproponowana aplikacja stara się przeprowadzić korektę tekstu w sposób automatyczny lub semi-automatyczny. Wszystko po to, aby móc wyłączyć człowieka z manualnego sprawdzania dużej ilości tekstu, oferując wstępnie sprawdzony tekst i zaznaczając jedynie miejsca, w których należy wprowadzić pewne poprawki. Niemniej jednak, zagadnienie badania języka, próba jego zrozumienia i zamiana wiedzy uzyskanej w procesie uczenia na jednoznaczne reguły i procedury, które będą wykorzystywane przez algorytmy, są bardzo trudne, a czasami wręcz niemożliwe. Dlatego też choć prace badawcze nad poprawą korekty tekstu nadal trwają, to w chwili obecnej jedynie człowiek jest w stanie poprawić wszystkie błędy językowe.

Otwarte spostrzeżenia badawcze

Podczas prac nad strukturą reprezentującą model języka w postaci grafu LHG oraz podczas implementacji algorytmów służących kontekstowej korekcie tekstu, natrafiono na kilka problemów i spostrzeżeń, które warto przeanalizować w przyszłości. Głównie dotyczą one poprawy efektywności zaproponowanych algorytmów, jak również możliwości opracowania kolejnych. Do uzupełnień tych można zaliczyć:

- Zbudowanie Grafu Przyzwyczajeń Lingwistycznych w oparciu o dzieła jednego autora. Tym samym otrzymamy graf dla konkretnej osoby i będzie można go użyć, aby zweryfikować, czy nowy tekst wprowadzony przez użytkownika został napisany przez tę samą osobę, która napisała te dzieła.
- Grupowanie słów podobnych. Jak zauważono w grafie występuje wiele wyrazów określających tą samą rzecz lub będących pewnym uszczegółowieniem. Interesująca więc wydaje się możliwość wprowadzenia dodatkowych relacji uogólniających.
- Zastosowanie dodatkowych bibliotek w celu oznaczania części mowy (POS tagging) dla zdań. Dodatkowa informacja o rodzajach części mowy występujących we wprowadzonym zdaniu mogłaby posłużyć do zastosowania kolejnego moduł, który na podstawie wzorców mógłby określić, czy zdanie jest poprawne.
- Podczas konstrukcji grafu dodawane są krawędzie asocjacyjne ACON oraz ACON_PREV, tych samych rzędów, pomiędzy tymi samymi neuronami słownymi. Można zoptymalizować

liczbę tworzonych nowych połączeń w grafie dodając zamiast dwóch połączeń kontekstowych jedno, specjalnie etykietowane połączenie, które pozwoli na uzyskanie tych samych właściwości. Po zastosowaniu takiej optymalizacji funkcjonalność korzystania z grafu pozostanie niezmienna, a dodatkowo zmniejszy się liczba krawędzi.

- Jak zostało stwierdzone wcześniej, algorytmy kontekstowej korekty tekstu oparte na grafie LHG są w stanie tym lepiej działać im więcej kontekstów słownych znajduje się w grafie. Stąd też zasadnym wydaje się ciągłe analizowanie i dodawanie poprawnych kontekstów słownych do grafu. Do realizacji takiego zadania trzeba jednak dysponować dużą pamięcią oraz serwerami odpowiedniej mocy obliczeniowej.
- Podczas poprawy tekstów można zauważyć, że najpopularniejsze wyrazy będą się powtarzać w wielu wprowadzonych tekstach. Dlatego też można zbudować lokalny podgraf dla grafu LHG, w którym będą przechowywane najczęściej występujące słowa i połączenia. Powstanie takiej dodatkowej struktury zapewni szybsze działanie algorytmów, jak również będzie miejscem wprowadzania ewentualnych optymalizacji, które nie wpłyną w istotny sposób na główny Graf Przyzwyczajeń Lingwistycznych.

Bibliografia

[1] Konieczny P., „Historia komunikacji: od mowy do Internetu”. [Online] https://histmag.org/Komunikacja-od-mowy-do-Internetu-744/1. [Dostęp: 24-sie-2019].

[2] Dance F. E. X., *The “Concept” of Communication*, Journal of Communication, t. 20, nr 2, s. 201–210, 1970.

[3] O’Reilly T., *What is Web 2.0*. O’Reilly Media, Inc., 2009.

[4] Główny Urząd Statystyczny, „Społeczeństwo informacyjne w Polsce w 2018 roku”, *stat.gov.pl*. [Online] https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2018-roku,2,8.html. [Dostęp: 21-sie-2019].

[5] Norris S., Maier C. D., *Interactions, Images and Texts: A Reader in Multimodality*. Walter de Gruyter GmbH & Co KG, 2014.

[6] Mykowiecka A., *Inżynieria lingwistyczna: komputerowe przetwarzanie tekstów w języku naturalnym*. Wydawnictwo Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych, 2007.

[7] Kukich K., *Techniques for Automatically Correcting Words in Text*, ACM Comput. Surv., t. 24, nr 4, s. 377–439, grudz. 1992.

[8] Peters J., Matusov E., „Automatic Text Correction”, US20070299664A1, 27-grudz-2007.

[9] Curran J. R., Clark S., Bos J., „Linguistically Motivated Large-scale NLP with C&C and Boxer”, Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Stroudsburg, PA, USA, 2007, s. 33–36.

[10] Màrquez L., Rodríguez H., „Part-of-speech tagging using decision trees”, Machine Learning: ECML-98, 1998, s. 25–36.

[11] Hasan F. M., UzZaman N., Khan M., „Comparison of different POS Tagging Techniques (n-gram, HMM and Brill’s tagger) for Bangla”, Advances and Innovations in Systems, Computing Sciences and Software Engineering, 2007, s. 121–126.

[12] Google Inc., „Google Zeitgeist 2012: A Year in Search”. [Online] https://archive.google.com/zeitgeist/2012/. [Dostęp: 21-sie-2019].

[13] Google Inc., „Google Search Statistics - Internet Live Stats”. [Online] https://www.internetlivestats.com/google-search-statistics/. [Dostęp: 21-sie-2019].

[14] Smith C., „365 Interesting Google Search Statistics and Much More (2019)”. [Online] https://expandedramblings.com/index.php/by-the-numbers-a-gigantic-list-of-google-stats-and-facts/. [Dostęp: 21-sie-2019].

[15] Macioch M., Czajka R., „Nie ma Cię w internecie? Znaczy, że nie istniejesz - Fundacja Firmy Rodzinne”. [Online] https://ffr.pl/pl/nie-ma-cie-w-internecie-znaczy-ze-nie-istniejesz/. [Dostęp: 24-sie-2019].
[16] „Jak działa wyszukiwarka Google | Pobieranie i indeksowanie”, [Online] https://www.google.com/intl/pl/search/howsearchworks/crawling-indexing/. [Dostęp: 24-sie-2019].
[17] Abiteboul S., Hull R., Vianu V., *Foundations of Databases*. Addison-Wesley, 1995.
[18] Swanson A. K., „Development And Management Of A Computer-Centered Data Base: Part 4: A Computer-Centered Data Base Serving Usaf Personnel Managers”, System Development Corp Santa Monica CA, TM-1456/004/00, lis. 1963.
[19] „Baza danych”, *Wikipedia, wolna encyklopedia*. 24-maj-2019.
[20] Codd E. F., *Relational Database: A Practical Foundation for Productivity, Readings in Artificial Intelligence and Databases*, (pod. red.: Mylopolous J., Brodie M.) San Francisco (CA): Morgan Kaufmann, 1989, s. 60–68.
[21] „Podstawy MySQL dla technika informatyka 351203”, [Online] http://www.glowacki.p9.pl/nowa_strona/strony/niedatowane/kurs_mysql/k_2_2_2.php. [Dostęp: 24-sie-2019].
[22] Moniruzzaman A. B. M., Hossain S. A., *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison*, arXiv:1307.0191 [cs], cze. 2013.
[23] Cattell R., *Scalable SQL and NoSQL Data Stores*, SIGMOD Rec., t. 39, nr 4, s. 12–27, maj 2011.
[24] Tudorica B. G., Bucur C., „A comparison between several NoSQL databases with comments and notes”, 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research, 2011, s. 1–5.
[25] „Exploring the Different Types of NoSQL Databases Part II”, *3Pillar Global*, 07-paź-2013. [Online] https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases. [Dostęp: 24-sie-2019].
[26] „Czym jest NoSQL, jak wykorzystać nierelacyjne bazy danych”, *ITwiz*, 09-cze-2017. [Online] https://itwiz.pl/czym-jest-nosql-jak-wykorzystac-nierelacyjne-bazy-danych/. [Dostęp: 24-sie-2019].
[27] „Which database system(s) does Twitter use? - Quora”, [Online] https://www.quora.com/Which-database-system-s-does-Twitter-use. [Dostęp: 24-sie-2019].
[28] „What Database Does Twitter Use? - A Deep Dive”, *8bitmen.com*, 30-maj-2019. .
[29] Russell M. A., *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O’Reilly Media, Inc., 2013.

[30] Mayer-Schönberger V., Cukier K., *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. Houghton Mifflin Harcourt, 2013.

[31] Ramsay J. O., *Functional Data Analysis*, *Encyclopedia of Statistical Sciences*, American Cancer Society, 2006.

[32] Martin J. H., Jurafsky D., *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.

[33] Nadkarni P. M., Ohno-Machado L., Chapman W. W., *Natural language processing: an introduction*, Journal of the American Medical Informatics Association, t. 18, nr 5, s. 544–551, wrz. 2011.

[34] „History of the punch card - Reference from WhatIs.com”, *WhatIs.com*. [Online] https://whatis.techtarget.com/reference/History-of-the-punch-card. [Dostęp: 24-sie-2019].

[35] Gonfalonieri A., „How Amazon Alexa works? Your guide to Natural Language Processing (AI)”, *Medium*, 31-grudz-2018. [Online] https://towardsdatascience.com/how-amazon-alexa-works-your-guide-to-natural-language-processing-ai-7506004709d3. [Dostęp: 24-sie-2019].

[36] Dale R., Moisl H., Somers H., *Handbook of Natural Language Processing*. CRC Press, 2000.

[37] Cambria E., Schuller B., Xia Y., Havasi C., *New Avenues in Opinion Mining and Sentiment Analysis*, IEEE Intelligent Systems, t. 28, nr 2, s. 15–21, mar. 2013.

[38] Lebart L., „Classification Problems in Text Analysis and Information Retrieval”, Advances in Data Science and Classification, 1998, s. 465–472.

[39] „Przetwarzanie języka naturalnego”, *Wikipedia, wolna encyklopedia*. 17-luty-2019.

[40] Potapenko A., Zobnin A., Kozlova A., Yudin S., Zimovnov A., „Natural Language Processing - National Research University Higher School of Economics”, *Coursera*. [Online] https://www.coursera.org/learn/language-processing/home/info. [Dostęp: 24-sie-2019].

[41] Zipf G. K., *Human behavior and the principle of least effort*. Oxford, England: Addison-Wesley Press, 1949.

[42] Dębowski Ł., „Prawo Zipfa - próby objaśnień.”, prezentowano w Instytut Podstaw Informatyki PAN.

[43] Kurcz I., *Słownik frekwencyjny polszczyzny współczesnej*. Polska Akademia Nauk, Instytut Języka Polskiego, 1990.

[44] Szydłowski M., Tambor P., *Emergentny i uniwersalny charakter prawa rozkładu Zipfa w nauce*, Humanistyka i Przyrodoznawstwo, nr 21, s. 61–78, sie. 2018.

[45] Sołdacki P., „Application of shallow text processing methods for Polish documents analysis”, 2008.

[46] Gajęcki M., „Automatic Text Clustering in the Polish Language”, Intelligent Information Processing and Web Mining, 2004, s. 419–423.
[47] Branny E., Gajęcki M., *Text summarizing in Polish*, Computer Science, t. Vol. 7, s. 31–48, 2005.
[48] Ingersoll G. S., Morton T. S., Farris A. L., *Taming Text: How to Find, Organize, and Manipulate It*. Greenwich, CT, USA: Manning Publications Co., 2013.
[49] Geitgey A., „Natural Language Processing is Fun!”, *Medium*, 02-sty-2019. [Online] https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e. [Dostęp: 25-sie-2019].
[50] Walkowska J., „NLP w pigułce”. [Online] http://namiekko.pl/2017/04/10/nlp-w-pigulce/. [Dostęp: 21-sie-2019].
[51] „Normalizacja tekstu”, *Wikipedia, wolna encyklopedia*. 13-luty-2019.
[52] Chomsky N., *Aspects of the Theory of Syntax*. MIT Press, 2014.
[53] Friedl J. E. F., *Mastering Regular Expressions*. O’Reilly Media, Inc., 2006.
[54] Boros T., Dumitrescu S., Pipa S., „Fast and Accurate Decision Trees for Natural Language Processing Tasks”, 2017, s. 103–110.
[55] Zhang K., Shasha D., *Simple Fast Algorithms for the Editing Distance between Trees and Related Problems*, SIAM Journal on Computing, t. 18, nr 6, s. 1245–1262, grudz. 1989.
[56] Jurafsky D., Manning C. D., „Natural Language Processing - Lecture Slides from the Stanford Coursera course”. [Online] https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html. [Dostęp: 25-sie-2019].
[57] Jurafsky D., „Spelling Correction and the Noisy Channel”, *https://web.stanford.edu/class/cs124*. [Online] https://web.stanford.edu/class/cs124/lec/spelling.pdf. [Dostęp: 04-wrz-2019].
[58] „What are some algorithms of spelling correction that are used by search engines? For example, when I used Google to search «Google imeges», it prompted me, «Did you mean: Google images?». - Quora”, [Online] https://www.quora.com/What-are-some-algorithms-of-spelling-correction-that-are-used-by-search-engines-For-example-when-I-used-Google-to-search-Google-imeges-it-prompted-me-Did-you-mean-Google-images. [Dostęp: 24-sie-2019].
[59] „How to Write a Spelling Corrector”, [Online] http://norvig.com/spell-correct.html. [Dostęp: 25-sie-2019].
[60] Garbe W., „1000x Faster Spelling Correction algorithm (2012)”, *Medium*, 14-maj-2018. [Online] https://medium.com/@wolfgarbe/1000x-faster-spelling-correction-algorithm-2012-8701fcd87a5f. [Dostęp: 25-sie-2019].

[61] Lubaszewski W., *Słowniki komputerowe i automatyczna ekstrakcja informacji z tekstu*. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, 2009.
[62] Blunsom P., *Hidden markov models*, Lecture notes, August, t. 15, nr 18–19, s. 48, 2004.
[63] „Soundex System”, *National Archives*, 15-sie-2016. [Online] https://www.archives.gov/research/census/soundex.html. [Dostęp: 25-sie-2019].
[64] Czoska A., *Klasykacja operatorów metatekstowych i częstość ich występowania w krótkich tekstach naukowych w języku polskim*, Investigationes Linguisticae, t. 23, s. 1–33, cze. 2011.
[65] Manning C. D., Manning C. D., Schütze H., *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
[66] Masłowska I., *Natural Language Processing*, s. 48.
[67] Mamchenkov A. L., „10,000 most common English words”, *Blog of Leonid Mamchenkov*, 18-sty-2017. .
[68] Brants T., Popat A. C., Xu P., Och F. J., Dean J., „Large Language Models in Machine Translation”, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 2007, s. 858–867.
[69] Gadamer M., *Automatyczna kontekstowa korekta tekstów na podstawie Grafu Przyzwyczajen Lingwistycznych (LHG) zbudowanego przez robota internetowego dla języka polskiego.*, Praca magisterska, AGH, 2009.
[70] Young T., Hazarika D., Poria S., Cambria E., *Recent Trends in Deep Learning Based Natural Language Processing*, arXiv:1708.02709 [cs], sie. 2017.
[71] Saravia E., „Deep Learning for NLP: An Overview of Recent Trends”, *Medium*, 29-paź-2018. [Online] https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d. [Dostęp: 25-sie-2019].
[72] Mikolov T., Sutskever I., Chen K., Corrado G., Dean J., *Distributed Representations of Words and Phrases and their Compositionality*, arXiv:1310.4546 [cs, stat], paź. 2013.
[73] Collobert R., Weston J., „A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Proceedings of the 25th International Conference on Machine Learning, New York, NY, USA, 2008, s. 160–167.
[74] Horzyk A., *Innovative Prediction Technology in Automatic Speech Recognition*.
[75] Ramachandran A., „NLP Guide: Identifying Part of Speech Tags using Conditional Random Fields”, *Medium*, 05-paź-2018. [Online] https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31. [Dostęp: 24-sie-2019].

[76] „Comparison of Treebank Statistics”, [Online] https://universaldependencies.org/treebanks/pl-comparison.html. [Dostęp: 24-sie-2019].

[77] Kasprzak W., *Rozpoznawanie obrazów i sygnałów mowy (Image and Speech Recognition)*. 2009.

[78] Nadeau D., Sekine S., *A survey of named entity recognition and classification*, Lingvisticæ Investigationes, t. 30, nr 1, s. 3–26, sty. 2007.

[79] Sołdacki P., „Zastosowanie metod płytkiej analizy tekstu do przetwarzania dokumentów w języku polskim”, 2008.

[80] Blunsom P., „Maximum Entropy Markov Models for Semantic Role Labelling”, Proceedings of the Australasian Language Technology Workshop 2004, Sydney, Australia, 2004, s. 109–116.

[81] Sutton C., McCallum A., *An Introduction to Conditional Random Fields*, arXiv:1011.4088 [stat], lis. 2010.

[82] Ciosek M., „Teoria Grafów”. [Online] http://www.student.krk.pl/026-Ciosek-Grybow/wprowadzenie.html. [Dostęp: 24-sie-2019].

[83] Rivest R. L., Leiserson C. E., *Introduction to Algorithms*. New York, NY, USA: McGraw-Hill, Inc., 1990.

[84] Wilson R. J., *Wprowadzenie do teorii grafów*. Wydawnictwo Naukowe PWN, 2007.

[85] „Zagadnienie mostów królewieckich”, *Wikipedia, wolna encyklopedia*. 24-cze-2019.

[86] Głowacki T., *Zastosowanie metod opartych na teorii grafów do rozwiązywania wybranych problemów analizy sekwencji nukleotydowych i aminokwasowych*, mar. 2013.

[87] „Graf (matematyka)”, *Wikipedia, wolna encyklopedia*. 24-cze-2019.

[88] Gadamer M., Horzyk A., *Text Analysis and Correction Using Specialized Linguistic Habit Graphs LHG*, Image Processing & Communications, t. 17, sty. 2012.

[89] Gadamer M., Horzyk A., *Automatyczna kontekstowa korekta tekstów z wykorzystaniem grafu LHG*, Computer Science, t. Vol. 10, s. 37–55, 2009.

[90] Gadamer M., Horzyk A., „Biologically Inspired Linguistic Habit Graph Networks Used for Text Correction”, 2018, s. 304–314.

[91] Horzyk A., Gadamer M., „Associative Text Representation and Correction”, 2013, t. 7894, s. 76–87.

[92] Bressler D., „Building a convolutional neural network for natural language processing”, *Medium*, 11-grudz-2018. [Online] https://towardsdatascience.com/how-to-build-a-gated-convolutional-neural-network-gcnn-for-natural-language-processing-nlp-5ba3ee730bfb. [Dostęp: 09-wrz-2019].

[93] Markowski A., *Kultura języka polskiego: Teoria. Zagadnienia leksykalne*. Wydawnictwo Naukowe PWN, 2005.

[94] Doroszewski W., Kurkowska H., *Slownik poprawnej polszczyzny PWN*, 1973.

[95] Gadamer M., „Linguistic Habit Graphs Used for Text Representation and Correction", 2017, s. 233–242.

[96] „Jak działają neurony? | ChangeMaker", [Online] https://blog.krolartur.com/jak-dzialaja-neurony/. [Dostęp: 15-wrz-2019].

[97] „How do brain neurons work? - Quora", [Online] https://www.quora.com/%20How-do-brain-neurons-work. [Dostęp: 24-sie-2019].

[98] „O Morfeuszu – Morfeusz 2", [Online] http://morfeusz.sgjp.pl/doc/about/. [Dostęp: 27-sie-2019].

[99] „Język polski - jeden z najtrudniejszych języków świata", *Onet Podróże*, 17-wrz-2018. [Online] https://podroze.onet.pl/porady/jezyk-polski-jeden-z-najtrudniejszych-jezykow-swiata/dl12eyk. [Dostęp: 07-wrz-2019].

[100] „How does Grammarly (the grammar checker) work? - Quora", [Online] https://www.quora.com/How-does-Grammarly-the-grammar-checker-work. [Dostęp: 24-sie-2019].

[101] Géron A., *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. .

[102] Socher R., Lin C., Ng A., Manning C., „Parsing Natural Scenes and Natural Language with Recursive Neural Networks", prezentowano w Proceedings of the 28th International Conference on Machine Learning, ICML 2011, 2011, s. 129–136.

Oświadczenie

Oświadczam, że wykonałem pracę samodzielnie oraz nie korzystałem z innych źródeł, niż te, które są wymienione w niniejszej rozprawie.

Marcin A. Gadamer