Marcin Gadamer*, Adrian Horzyk*

AUTOMATYCZNA KONTEKSTOWA KOREKTA TEKSTÓW Z WYKORZYSTANIEM GRAFU LHG

Automatyczna korekta tekstów stanowi ważny problem z punktu widzenia dzisiejszych procesorów i edytorów tekstów. W tym artykule został przedstawiony innowacyjny algorytm służący do automatyzacji kontekstowej korekty tekstów z wykorzystaniem Grafu Przyzwyczajeń Lingwistycznych (LHG), który również opisano w tym artykule. W tym celu zbudowano specjalistycznego pająka internetowego przeszukującego strony internetowe celem skonstruowania Grafu Przyzwyczajeń Lingwistycznych (LHG) na podstawie analizy korpusów tekstów uzyskanych z polskojęzycznych stron internetowych. Otrzymane wyniki korekty tekstu z wykorzystaniem tego algorytmu, bazującego na grafie LHG, zostały porównane z komercyjnymi programami do korekty tekstu takimi jak Microsoft Word 2007, Open Office Writer 3.0 oraz z wyszukiwarką Google. Otrzymane wyniki korekty tekstów okazały się być znacznie lepsze niż w wyżej wymienionych komercyjnych narzędziach.

Słowa kluczowe: *automatyczna korekta tekstów, graf LHG*

AUTOMATIC CONTEXTUAL TEXT CORRECTION USING THE LINGUISTIC HABITS GRAPH LHG

Automatic text correction is an essential problem of today text processors and editors. This paper introduces a novel algorithm for automation of contextual text correction using a Linguistic Habit Graph (LHG) also introduced in this paper. A specialist internet crawler has been constructed for searching through web sites in order to build a Linguistic Habit Graph after text corpuses gathered in polish web sites. The achieved correction results on a basis of this algorithm using this LHG were compared with commercial programs which also enable to make text correction: Microsoft Word 2007, Open Office Writer 3.0 and search engine Google. The achieved results of text correction were much better than correction made by these commercial tools.

Keywords: *automatic text correction, graph LHG*

* Katedra Automatyki, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Akademia Górniczo-Hutnicza w Krakowie, gadamer@agh.edu.pl, horzyk@agh.edu.pl

1. Wstęp

Postęp występuje w każdej dziedzinie życia. Przez niektórych jest on rozumiany jako naturalny proces ewolucyjny, u niektórych natomiast wynika z zaciekawienia światem, możliwości odkrywania nowych zjawisk, bądź odpowiedzi na próbę ułatwiania codziennego życia.

Postęp dotyka wielu sfer życia człowieka, szczególnie tych, które towarzyszą mu w jego codzienności. Można dziś łatwo zauważyć bardzo szybki postęp technologiczny, a także ewolucję, jakiej uległ również sposób komunikacji międzyludzkiej oraz związana z nim przemiana języka. Początkowo ludzie pierwotni komunikowali się przez pojedyncze słowa. Komunikacja na odległość była możliwa jedynie dzięki gestom oraz obrazom. Sposób komunikacji z biegiem czasu uległ jednak zmianie. W dzisiejszych czasach wspomniane wcześniej tradycyjne środki komunikacji odeszły na dalszy plan, oddając miejsce nowoczesnej technologii. Już nikt nie wyobraża sobie komunikacji bez telefonu (stacjonarnego bądź komórkowego) lub bez pośrednictwa Internetu. Koniec XX wieku przyniósł silny rozwój technologiczny, a powstające po dziś dzień coraz to nowe komputery oraz oprogramowanie dostarcza wielu zaawansowanych możliwości pomocy człowiekowi w codziennej pracy.

Użytkownik coraz częściej wykorzystuje drogę elektroniczną w codziennej komunikacji z drugą osobą. W dzisiejszych czasach powstaje bardzo duża liczba elektronicznych dokumentów, w których nieodłącznie występują błędy językowe. Istnieje więc potrzeba stworzenia bardziej inteligentnej i kontekstowej korekty tekstu, który został wprowadzony z różnego rodzaju błędami. Na pojawianie się tych błędów ma wpływ wiele różnorakich czynników. Najczęściej błędy powstają z winy użytkownika a czasami przez program komputerowy. Do najważniejszych błędów użytkownika można zaliczyć:

- 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 teksu (kontekst wyrazów i zdań),
- brak badania wprowadzenia przypadkowo poprawnych / błędnych wyrazów.

Lingwistyka komputerowa bądź inżynieria lingwistyczna, to jedne z wielu nazw tej samej dziedziny, która obejmuje próby zautomatyzowania przetwarzania danych w języku naturalnym. Główne, rozpatrywane aktualnie zadania lingwistyki to: wyszukiwanie, klasyfikacja i selekcja informacji wyrażonych za pomocą języka naturalnego, a także maszynowe tłumaczenie tekstów, z jednego języka na drugi oraz streszczanie i analiza mowy. Związane z tymi zastosowaniami badania dotyczą opisu różnych poziomów wiedzy o języku: morfologii, składni, semantyki i pragmatyki wypowiedzi

w języku naturalnym, budowania zasobów lingwistycznych, słowników jedno- i wielojęzycznych, dużych korpusów tekstów znakowanych różnego typu informacjami oraz tworzenia programów do analizy danych językowych (Mykowiecka 2007).

2. Metody korekcji tekstu

Podczas wprowadzania tekstu często popełniane są błędy. Jednym z zadań niektórych algorytmów i stworzonych na ich podstawie programów jest więc ich poprawa, tak aby zdania wprowadzane z błędami były korygowane i przekształcane na zdania poprawne (w różnym ujęciu, m.in. ortograficznym, gramatycznym, fleksyjnym), z punktu widzenia danego języka naturalnego. Na świecie istnieje kilka sposobów i metod korekcji takich zdań. Do najpopularniejszych i zarazem podstawowych metod można zaliczyć:

- *Metodę odległości edycyjnej* – metoda odległości edycyjnej jest jedną z najbardziej popularnych metod korekcji tekstów. Odległość edycyjną (później od nazwiska autora nazwaną metodą odległości Levensteina) wprowadził w 1966 r. V. I. Levenshtein. Metoda ta polega na porównaniu dwóch słów i próbie przekształcenia jednego z nich w drugie, wykorzystując przy tym dostępne operacje edycyjne (wstawienie bądź usunięcie litery, jak również zastąpienie litery inną). Każdej takiej operacji przypisywany jest pewien koszt i wyszukiwane jest takie przekształcenie wyrazu uznanego za błędny na poprawny, dla którego koszt jest najmniejszy. Ów najmniejszy koszt jest odległością edycyjną danych dwóch słów (Miró et al. 2004).
- *Metodę N-gramów oraz łańcuchów Markowa* – wnioskowanie statystyczne to dział statystyki zajmujący się problemami uogólniania wyników badania próby losowej na całą populację oraz szacowania błędów wynikających z takiego uogólnienia. Przykładem wykorzystania takiego wnioskowania może być rozwiązanie problemu przewidywania następnego słowa, dla danych wcześniejszych słów. Metoda N-gramu (część teorii łańcuchów Markowa) bazuje na wnioskowaniu statycznym. Metoda ta jest związana z modelem probabilistycznym, pozwalającym obliczyć prawdopodobieństwo występowania wyrazów w zdaniu (Gawrysiak 2006).
  W metodzie N-gramu wykorzystywana jest historia poprzedzających słów danego badanego wyrazu. Biorąc pod uwagę jedną historię, nie można jednak odgadnąć następnego słowa. Dane zdanie może mieć bowiem podobny początek, ale całkiem inny koniec (Statistical Inference 2009). Zatem do sprawdzania historii możemy wykorzystywać:
  - bigramy – badanie tylko poprzedzającego wyrazu,
  - trigramy – badanie dwóch poprzedzających wyrazów,
  - tetragramy – badanie trzech poprzedzających wyrazów.

  Zwiększając liczbę istotnych słów otrzymuje się więcej informacji o kontekście. Istnieje jednak ryzyko, że w danej klasie nie będzie żadnych danych lub wystąpi bardzo mała ich ilość, co może wpłynąć na wiarygodność estymacji.

- *Analizę statyczną – prawo Zipfa* – metoda ta bazuje na frekwencyjnym słowniku dla badanego języka naturalnego. Słownik ten zawiera słowa zgodne z zasadą ortografii wraz z liczbą (częstością) ich występowania w języku naturalnym. Aby sformułować prawo Zipfa, należy najpierw określić, czym jest częstość występowania słowa oraz czym jest pozycja rankingowa (Dębowski 2005).
  - częstość to liczba wystąpień słowa w tekście (korpusie).
  - lista rankingowa słów – to lista słów posortowana malejąco względem liczby wystąpień w tekście (korpusie).
  - ranga słowa jest to zatem liczba porządkowa słowa na liście rankingowej. Najczęściej występujące słowo ma rangę równą 1, drugie co do częstości ma rangę równą 2, trzecie ma rangę 3, itd.

  Prawo Zipfa można sformułować następująco:

  Twierdzenie 1 (Prawo Zipfa). *Częstość występowania słów w języku naturalnym jest odwrotnie proporcjonalna do ich pozycji w rankingu.*

- *Metoda odległości na klawiaturze* – zauważono, że część błędów we wprowadzanym tekście wynika z błędnych naciśnięć klawiszy, które znajdują się w niewielkiej odległości od siebie na klawiaturze (tzw. literówki). Użytkownik wprowadzając tekst naciska nie ten klawisz, który zamierzał, przez co generowany tekst zawiera błędne lub przestawione znaki. We wprowadzanych zdaniach pojawiają się wyrazy, których nie ma nawet w słowniku danego języka. Błędy te nie wynikają jednak z nieznajomości poprawnej konstrukcji wyrazu, lecz są zdeterminowane zdolnościami manualnymi osoby wprowadzającej tekst i wynikają często z pośpiechu piszącego, jego niewystarczającego skupienia się itp.
  Metoda ta bada wprowadzane słowa z wykorzystaniem układu liter na klawiaturze. W chwili napotkania błędnego wyrazu próbuje dopasować wyraz najbardziej bliski danemu zamieniając litery, wykorzystując przy tym informacje, jakie litery stoją najbliżej rozważanych, które brane są jako błędnie wprowadzone. Literom stojącym bliżej błędnie naciśniętej nadaje się mniejsze wagi błędów, niż literom, które są dalej od litery wyjściowej.

Do najbardziej popularnych edytorów tekstu na polskim rynku można z pewnością zaliczyć:

- *Microsoft Word* – edytor tekstu wchodzący w skład pakietu biurowego Microsoft Office. Najnowsza wersja tego programu – Microsoft Word 2007 – ułatwia tworzenie i udostępnianie profesjonalnie wyglądającej zawartości, dzięki połączeniu kompletnego zestawu narzędzi do pisania i łatwego w użyciu interfejsu użytkownika Microsoft Office (Microsoft Word 2009)
- *Writer* – edytor tekstu wchodzący w skład pakietu biurowego OpenOffice.org. OpenOffice.org to pakiet biurowy, działający w wielu systemach operacyjnych i środowiskach, z otwartym dostępem do kodu źródłowego. Pakiet jest obecnie wyposażony w polski słownik ortograficzny, słownik synonimów, zasady podziału wyrazów, polską dokumentację i pomoc, zasady autokorekty, a także korektor

gramatyczny. Od 27 czerwca 2007 OpenOffice.org należy do Koalicji na Rzecz Otwartych Standardów (KROS) (OpenOffice.org Writer 2009).

Już w poprzednich wersjach edytora Word pojawiło się narzędzie, wykonujące automatyczną korektę wpisywanych wyrazów. Jego zadaniem było poprawianie błędnych wyrazów podczas ich wpisywania. Narzędzie to działa dobrze, ale przy pisaniu tekstów językiem ściśle technicznym niezbędne bywają ręczne poprawki. Writer również posiada autokorektę, ale działa ona mniej radykalnie od tej w Wordzie. Zawiera on jednak ciekawą opcję podpowiedzi słów używanych w danym dokumencie. Gdy program znajduje słowo, które pasuje do wpisywanych początkowych liter, podpowiada je, wyświetlając w negatywie. Wystarczy nacisnąć klawisz Enter, by zaakceptować propozycję. Niestety nie działa ona dla wszystkich wyrazów w języku polskim, gdyż nie uwzględnia ona kontekstowo polskiej fleksji. Ważną opcją OpenOffice'a jest moduł sprawdzania pisowni w języku polskim. Chociaż jest odczuwalnie wolniejszy od Worda 2007, działa bardzo dobrze. Nie odstaje od swojego komercyjnego odpowiednika. Można spotkać się również ze stwierdzeniami, że jest od niego nawet lepszy. Moduł ortograficzny Worda, podczas sprawdzania pisowni w języku polskim, nie uwzględnia wielu nazw geograficznych. Przykładem może być miejscowość Redmond, która jest siedzibą Microsoftu. Najmocniejszą stroną Writera, w tej dziedzinie, jest bardzo ważny dodatek - LanguageTool. Narzędzie to integruje się z Writerem, umożliwiając kontrolę poprawności gramatycznej oraz stylistycznej także w języku polskim. LanguageTool sprawdza wiele reguł (ponad 800), wliczając bardzo zaawansowane (błędy odmiany, pleonazmy, błędy leksykalne). W odróżnieniu od kontroli pisowni, sprawdzanie we Writerze nie odbywa się podczas pisania, lecz należy ją włączyć, po napisaniu tekstu. Word 2007 ma podobne narzędzie, które może działać także w trakcie pisania, ale jego skuteczność pozostawia wiele do życzenia. LanguageTool podaje ponadto propozycje zmian (na przykład umieszczenie brakującego przecinka), ale nie zawsze potrafi to zrobić prawidłowo i czasami wybiera dwa razy podobne fragmenty zdania. Mimo niedociągnięć, LanguageTool jest jednym z poważnych argumentów za używaniem Writera, gdyż ten dodatek bardzo pomaga przy pisaniu i korygowaniu tekstów (Marciniak 2000).

Porównanie to dotyczy jedynie modułu korekcji tekstu w tych dwóch edytorach. Można zauważyć, że lepszym modułem do sprawdzania poprawności tekstów powinien dysponować darmowy Writer.

Coraz częściej widać dbałość o formę przekazu treści. Wystarczy zwrócić uwagę na popełnianą ilość błędów ortograficznych, zamian liter w wyrazie lub ich skrótów. Powstaje więc podstawowa trudność – *jak zrozumieć i zinterpretować przekaz drugiej osoby?* O ile człowiek, w dość prosty sposób, może zrozumieć wypowiedź drugiej osoby, tak zrozumienie tego samego tekstu przez komputer, okazuje się bardzo trudnym i złożonym zadaniem. W związku z tym na świecie pojawiają się liczne próby stworzenia algorytmów, które byłyby w stanie „zrozumieć sens" (semantykę) wpisanego zdania i o ile zajdzie taka potrzeba, będą w stanie poprawić to zdanie tak, by było ono poprawne, zarówno ortograficznie, gramatycznie jak i stylistycznie.

Dotychczas opisywane i wykorzystywane metody, w dużej mierze, bazowały na przekształceniu pojedynczych słów, w oderwaniu ich od kontekstu wypowiedzi. Podczas przeprowadzonych badań korekty tekstu z wykorzystaniem istniejących mechanizmów, została zaproponował nowa metoda do analizy i korekcji tekstu. Polega ona na badaniu, w jakim kontekście oraz w jakiej formie fleksyjnej użyte są wyrazy w obrębie zdania. Dodatkowo dokonuje ona sprawdzenia, czy możliwe jest, aby dane słowo występowało właśnie w takim kontekście wyrazów stojących bezpośrednio przed jak i za nim. Badanie zdań za pomocą takiego algorytmu w znaczny sposób poprawia możliwości korekty błędnych wyrazów w zdaniu. Dodatkowo sprawdzanie również przez algorytm formy fleksyjnej słów w znaczący sposób rozszerza kontekst tego badania.

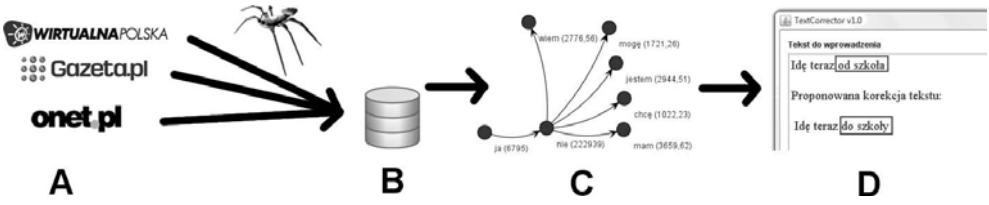
Celem, jaki został więc postawiony, było opracowanie i zaimplementowanie innowacyjnego mechanizmu korekcji tekstów oraz późniejsze jego porównanie z dotychczas stosowanymi metodami.

3. Algorytm kontekstowej korekcji tekstów

Stworzony algorytm do automatycznej kontekstowej korekty tekstu składał się z kilku części. Ich schemat został przedstawiony na rysunku 1.

Na rysunku 1 można wyróżnić trzy główne etapy:

1. Specjalistyczny pająk internetowy zbiera informacje o zdaniach ze stron internetowych i zapisuje je w bazie danych.
2. Dzięki analizie zebranych korpusów tekstów w bazie danych tworzony jest Graf Przyzwyczajeń Lingwistycznych (LHG) dla języka polskiego.
3. Na podstawie grafu LHG innowacyjny algorytm dokonuje automatycznej kontekstowej korekty wprowadzonych tekstów, bazując na kontekście i częstotliwości jego występowania oraz podobieństwie słów występujących w tym kontekście do słów błędnych.



Rys. 1. Schemat działania algorytmu

W tym celu skonstruowano specjalistycznego robota (pająka) internetowego przeszukującego strony internetowe, napisane w różnych językach skryptowych oraz formatach tekstowych, celem zbudowania maksymalnie rozbudowanego i ogólnego Grafu Przyzwyczajeń Lingwistycznych (LHG) ludzi dla wybranego języka, na podstawie analizy tych testów.

Pierwszą czynnością, jaką wykonywał pająk, to pobranie z bazy danych adresu WWW strony, która zostanie poddana analizie. Po pobraniu odnośnika z bazy, pająk pobierał całą zawartość strony (jej źródło), w celu dalszej obróbki. Następie wyszukiwane były wszystkie odnośniki innych stron WWW, do których można przejść z danej strony w dalszej kolejności. Każdy adres sprawdzany był w bazie danych, czy nie został on już wcześniej dodany przez pająka. Jeśli adres nie znajdował się wcześniej w bazie odnośników, pająk dodawał ten adres do lisy adresów stron, na jakie ma wejść, jak również do spisu wszystkich adresów.

Kolejną czynnością, jaką wykonywał pająk, było sprawdzenie, czy strona internetowa zawiera informacje w języku polskim. Gdy pająk stwierdził, że tak jest, przystępował do ekstrakcji jak największej ilości zdań, aby następnie móc analizować kontekst wypowiedzi poszczególnych słów. Dla każdego ze zdań pobierany był do bazy danych jego kontekst słowny. Działanie algorytmu polegało na zapisaniu do bazy, każdej trójki występujących po sobie słów w zdaniu. Jeśli zdanie składa się z dwóch lub jednego wyrazu, słowa takie są pomijane. Przykład obrazujący pobieranie kontekstu słów ze zdania został przedstawiony na rysunku 2.

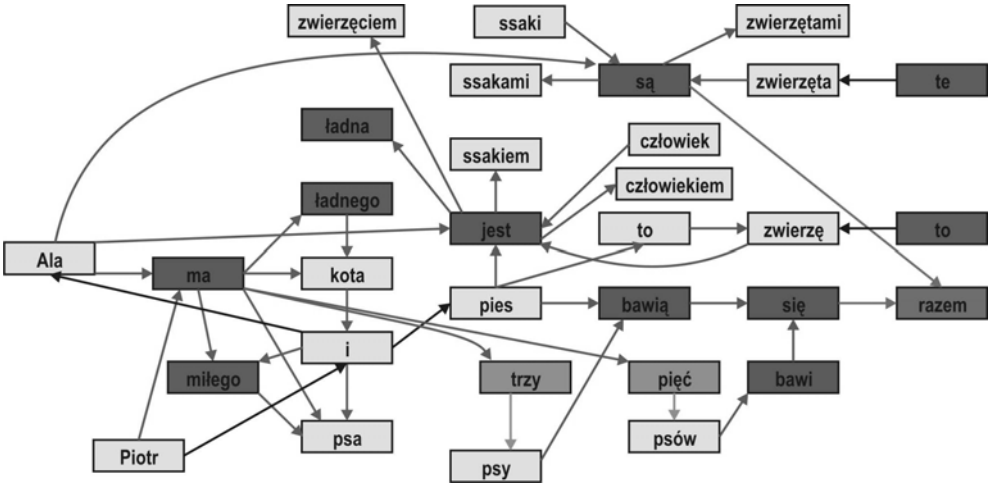
Rys. 2. Przykładowy kontekst trzech słów, pobierany przez pająka

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

- 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.

Ostatnią czynnością jest dopisanie do listy stron odwiedzonych aktualnej strony internetowej oraz wykasowanie jej z listy stron odwiedzonych i przeanalizowanych. Następnie pająk pobiera kolejny adres strony WWW, znajdującej się na liście stron do odwiedzenia i rozpoczyna cały etap jej analizy od początku, stopniowo rozbudowując bazę trójek.

Kolejnym etapem pracy było zbudowanie maksymalnie rozbudowanego i ogólnego Grafu Przyzwyczajeń Lingwistycznych (LHG) dla języka polskiego (rys. 3), na podstawie uzyskanego kontekstu słów.



Rys. 3. Przykład grafu LHG

Graf LHG umożliwiający badanie fleksyjno-częstotliwościowego kontekstu słownego, zaproponowany przez współautora tej pracy – Adriana Horzyka, jest grafem skierowanym i składa się z węzłów, które symbolizują słowa w określonych formach fleksyjnych tak, że każda forma fleksyjna występuje w grafie dokładnie raz, oraz krawędzi skierowanych, które symbolizują istniejące następstwo występowania połączonych ze sobą słów w określonych formach fleksyjnych, przy czym krawędzie są dodatkowo etykietowane ilością wystąpień słów w takim właśnie kontekście słownym – co daje dodatkową bardzo ważną informację o częstotliwości występowania takiego kontekstu w danym języku – czyli mówi o przyzwyczajeniach językowych stosujących go ludzi.

Wynikiem działania pająka internetowego było zbudowanie bazy kilkudziesięciu milionów trójek-wyrazów, które tworzą pewien kontekst wypowiedzi dla języka polskiego. Zaimplementowany graf LHG zawiera jednak jedynie informacje o kontekście wypowiedzi, czyli o określonym porządku słów mogącym wystąpić w zdaniach. Niewątpliwą zaletą stosowania grafów LHG jest oprócz możliwości odtworzenia wszystkich zapisanych w bazie zdań, utworzenie wielu nowych, które wcześniej nie zostały zapisane, a mogą być poprawne. Z taką agregacją kontekstu wypowiedzi wiąże się również poważna wada. Budując graf dla poprawnych językowo zdań, można byłoby odtworzyć zdania niepoprawne. Wynika to z faktu posiadania w bazie kontekstu jedynie dla trzech najbliższych słów. Gdy całe zdanie składa się z kilku wyrazów, to może się zdarzyć, że jego początek sztucznie wygenerowanego zdania z grafu LHG nie będzie semantycznie poprawny z jego końcem. Grafy LHG jednak nie służą do budowania logicznych zdań, ale do ich analizy i sprawdzania z uwzględnieniem form fleksyjnych występujących w nim następników i poprzedników słów z określoną częstotliwością. 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 do automatycznej korekty tekstów są w stanie automatycznie eliminować różnego rodzaju błędy językowe z „przeczytanych" 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, która podczas korekty tekstu nie „przebije się" ponad kontekst poprawnych językowo słów. 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.

Wierzchołkami grafu są kolejne nowe wyrazy, które zostały znalezione przez pająka internetowego. W pierwszej fazie budowy grafu, gdy wierzchołków jest zaledwie kilka, następuje bardzo duży przyrost liczby nowych wierzchołków. Dla kilku początkowo przeczytanych zdań, „znanych wyrazów", które zostały umieszczone w grafie, jest niewiele. Stąd każde nowe zdanie będzie posiadać wiele nowych słów, które zostaną umieszczone w grafie jako kolejne nowe wierzchołki. W trakcie czytania kolejnych zdań przez pająka, struktura grafu powiększa się w coraz mniejszym stopniu i równocześnie aktualizowane są wagi połączeń w grafie świadczące o najczęstszych kontekstach słów, które występują w tekstach czytanych przez pająka. Znaczy to, iż zwiększane są jedynie wagi krawędzi grafu LHG, reprezentujące liczbę wystąpień połączeń pomiędzy znanymi już wyrazami. Jest to kolejna ważna zaleta proponowanego rozwiązania polegająca na zapisie kontekstu słów przy pomocy tego grafu. Graf LHG może być ciągle rozbudowywany w miarę czytania kolejnych tekstów przez pająka, zaś informacje o nowych kontekstach słów będą na bieżąco aktualizowane w grafie, przy czym nowe wierzchołki do grafu będą dodawane coraz rzadziej.

Następnie zaproponowano i wykorzystano innowacyjny algorytm służący kontekstowej korekcji tekstów. W obecnych czasach nie jest dostępny żaden edytor, który potrafiłby poprawiać wprowadzony tekst na podstawie jego kontekstu słowno-fleksyjnego dla języka polskiego. Najczęściej wykorzystywanymi edytorami tekstu, dla przeciętnego użytkownika, jest Microsoft Word, który wchodzi w skład pakietu Microsoft Office oraz jego darmowy odpowiednik Writer, wchodzący w skład pakietu OpenOffice.org. W obydwu tych programach zostały co prawda zaimplementowane mechanizmy korekcji tekstu, lecz dotyczą one generalnie korekcji pojedynczego wyrazu, zasadniczo nie uwzględniając szerszego kontekstu, w jakim został on użyty.

W zbudowanej przez pająka internetowego bazie danych znajdują się miliony rekordów z trójkami słów, które posłużyły do zbudowania grafu LHG, który dzięki temu może dla dowolnych wyrazów określić kontekst słowny i fleksyjny poprzedzających i następnych słów. Dla każdego słowa istnieje możliwość wyróżnienia trzech typów możliwych korekt:

1) badane słowo jest poprzednikiem dwóch pozostałych występujących w bazie trójek oraz grafie LHG,
2) badane słowo znajduje się pomiędzy dwoma znanymi wyrazami występującymi w bazie trójek oraz grafie LHG,
3) badane słowo znajduje się jako następnik dwóch pozostałych występujących w bazie trójek oraz grafie LHG.

Podczas badań okazało się, że najlepsza weryfikacja oraz poprawa słów występuje wówczas, gdy wykorzystany jest trzeci sposób badania, tj. sprawdzane jest, czy badane słowo występuje jako następnik dwóch poprzedzających go wyrazów. Analizując budowę LHG stwierdzono, że znacznie częściej można znaleźć badane słowo jako następnik niż jako poprzednik określonych wyrazów w zdaniu. W konstrukcji wyrażeń o wiele częściej można znaleźć takie same początki wielu zdań, z różnymi ich zakończeniami, niż takie same zakończenia z różnymi początkami. Ponadto błędy językowe w zdaniu najczęściej są popełniane w jego środku lub na jego końcu niż na jego początku. Budowa algorytmu kontekstowej korekty tekstów została zatem oparta głównie o analizę następników. Może się również zdarzyć, że we wprowadzonym tekście pierwsze słowa są błędne. Do analizy takiego przypadku wykorzystano sprawdzanie kontekstu wstecznego. Opiera się ono na badaniu, czy możliwe jest wystąpienie danych wyrazów jako słów w środku kontekstu oraz jako poprzedników dwóch następujących po nim wyrazów. Dodatkowo badanie kontekstu wstecznego wykorzystano podczas korekcji błędnych wyrazów. Dla każdego słowa, uznanego przez algorytm za błędne, wyszukiwane są wyrazy występujące w różnych kontekstach, mogące go zamienić.

4. Porównanie z najpopularniejszymi edytorami tekstu

W trakcie badań wykonano porównania zaproponowanego w tym artykule algorytmu korekty tekstów z wykorzystywanymi mechanizmami do korekty tekstów zaimplementowanych w popularnych edytorach tekstów, tj. Microsoft Word oraz OpenOffice Writer. Również w popularnej wyszukiwarce Google zostały zaimplementowane pewne sposoby korekty tekstów, które użytkownik błędnie wprowadził. W dalszej części tego rozdziału zostaną porównane wyniki korekcji tekstów w tych właśnie aplikacjach z zaimplementowanym algorytmem, bazującym na LHG, wykorzystując kilka przykładowych zdań. Do testów zostały wykorzystane najpopularniejsze edytory biurowe we wskazanych poniżej wersjach:

- Microsoft Word 2007 wchodzący w skład pakietu Microsoft Office 2007,
- Writer wchodzący w skład pakietu OpenOffice.org 3.0.1,
- wyszukiwarka internetowa Google (http://www.google.pl).

Algorytm kontekstowej korekty tekstu został oparty na LHG zbudowanym w oparciu o bazę, która zawierała:

- 273 11 unikalnych słów wraz z określeniem częstości ich występowania, na podstawie Słownika Frekwencyjnego Języka polskiego (KGLK 2009),

- 30 636 215 trójek słów w określonej formie fleksyjnej w ich rzadkim kontekście słowno-fleksyjnym w obrębie zdań.

4.1. Moduł do konstrukcji zdań

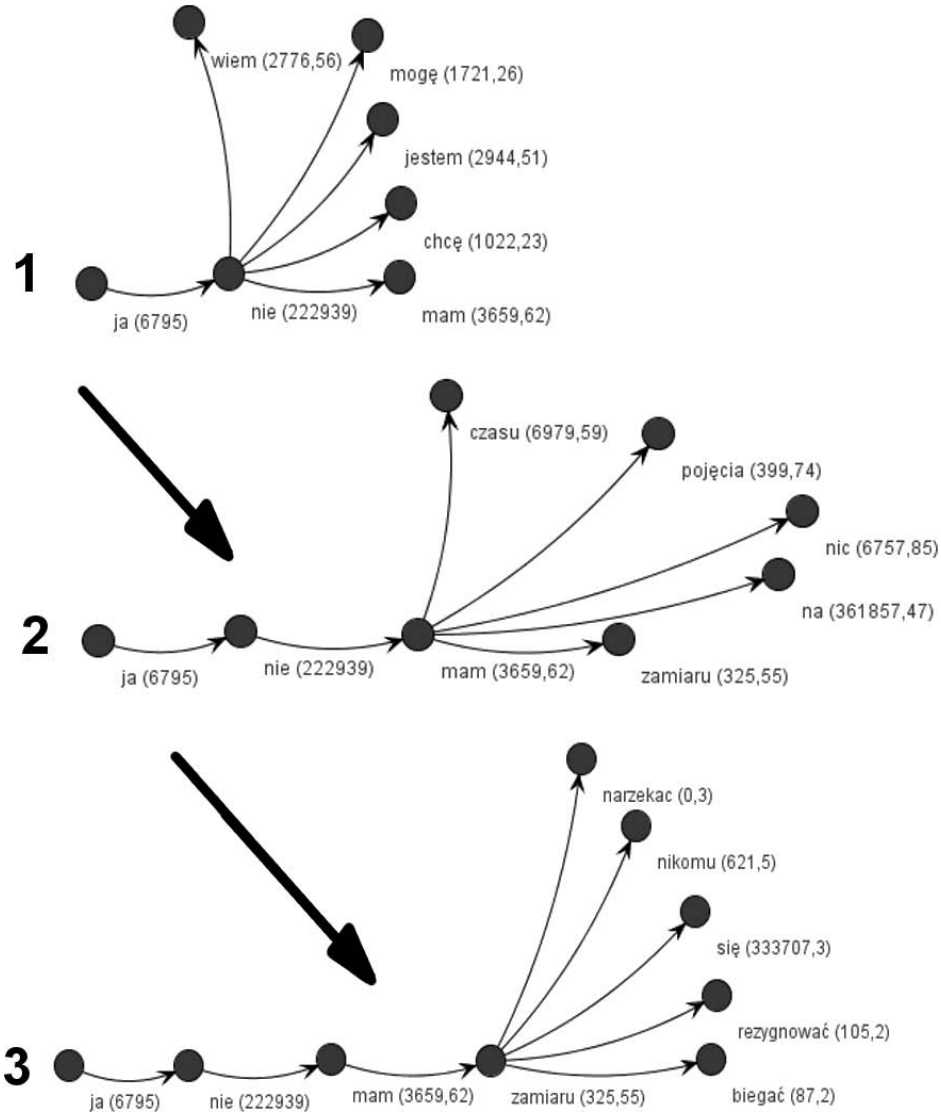
Wiele razy użytkownik wpisując początek zdania może mieć wątpliwości, jakiego należy użyć następnego wyrazu lub w jakiej formie fleksyjnej. W programie Writer obecny jest moduł do „podpowiadania" końcówki wyrazu, jednak w tym porównaniu podpowiadany ma być cały następny wyraz. Niestety żaden z pośród dwóch wspomnianych edytorów tekstu, nie dostarcza takiej funkcjonalności. Częściowo moduł do podpowiadania (uzupełniania) zdań występuje w wyszukiwarce Google, jednak ogranicza się on jedynie do najczęściej wpisywanych fraz w tej wyszukiwarce, które nie muszą być reprezentatywne dla rozważanego języka ani dla kontekstu wpisywanego przez użytkownika.

W stworzonej aplikacji, która wykorzystuje graf LHG, należy wpisać dwa początkowe słowa, dla których mają pojawić się propozycje kolejnych wyrazów. Po ich wpisaniu użytkownikowi zostanie zaprezentowana część grafu, która powiązana jest z wierzchołkami reprezentującymi te słowa (rys. 4). Obok nich, w nawiasie, zostaną podane częstości występowania danego słowa (rys. 4) w Słowniku Frekwencyjnym, zbudowanym dla języka polskiego (KGLK 2009). Gdy początkowe wyrazy zostaną wpisane błędnie, algorytm ich nie odnajdzie w swojej bazie i użytkownik będzie musiał wpisać je raz jeszcze, tym razem poprawnie.

Po ukazaniu się początkowych słów, użytkownik ma możliwość wpisać wyraz w grafie, dla którego mają być podane możliwe do wystąpienia następne słowa. Poczynając od trzeciego słowa, w nawiasie obok jego nazwy podawane są dwie liczby. Pierwsza z nich to wspominana już częstość występowania danego słowa w słowniku, natomiast druga liczba to częstość występowania danego wyrazu w kontekście dwóch poprzedzających go słów.

Na rysunku 4 zostały przedstawione początkowe etapy konstrukcji zdania z wykorzystaniem LHG.

Dla przeprowadzonych testów znacząco lepsze propozycje wyrazów uzyskano z wykorzystaniem skonstruowanego grafu LHG. Podczas porównania kolejne wyrazy były proponowane zarówno przez zaimplementowany algorytm, jak i przez wyszukiwarkę Google. W wyszukiwarce internetowej podane są one niejednokrotnie bez ogonków polskich znaków diakrytycznych, co może sugerować, iż użytkownicy częściej wpisują wyrazy bez polskich „ogonków". Takie wyrazy wystąpiły również jako propozycje w utworzonej bazie danych dla skonstruowanego i zaproponowanego w tym artykule algorytmu. Zaimplementowany pająk internetowy przeszukiwał bowiem wszystkie strony internetowe, więc natrafił również na liczne blogi, na których twórcy nie zawsze używają poprawnej konstrukcji wyrazów i budowy zdań. Jednak można przyjąć, że dla tak dużej bazy danych (ponad 30 milionów rekordów) wyrazy uznane za błędne w poprawnej pisowni wystąpią dużo rzadziej niż wyrazy poprawne.



Rys. 4. Kolejne etapy konstrukcji nowego zdania z wykorzystaniem grafu LHG

Ponieważ żaden z dwóch wiodących w popularności na rynku edytorów tekstu nie posiadał modułu do konstrukcji nowych zdań, testy zostały przeprowadzone jedynie dla stworzonego programu. Dodatkowo zostało sprawdzone, czy generowane kolejne słowa są poprawne językowo. W bardzo dużej mierze wyrazy były poprawne i zgodne z zasadami pisowni. Kilka z nich wystąpiło jednak w formach, np. „bez ogonków".

4.2. Sprawdzanie wprowadzanych wyrazów i podpowiadanie ich zakończeń

W rozdziale tym zostaną zaprezentowane i porównane dwa zaimplementowane mechanizmy:

1) automatycznego sprawdzania poprawności wpisywanych słów,
2) podpowiadania aktualnie wpisywanego słowa.

Z podobnymi algorytmami, występującymi w programach Microsoft Word 2007 oraz Open Office Writer 3.0.

Bardzo ważną i istotną rolę podczas wprowadzania tekstu pełni automatyczne sprawdzanie wpisywanych wyrazów. Dzięki temu użytkownik zaraz po napisaniu wyrazu jest informowany o ewentualnym wystąpieniu w nim błędu. Moduły do automatycznego sprawdzania poprawności wyrazów dostępne są bez konieczności dodatkowej ich instalacji we wspomnianych wcześniej dwóch edytorach. Użytkownik, zaraz po zainstalowaniu programu, ma możliwość sprawdzania pisowni. Moduły te bazują głównie na wspomnianej wcześniej odległości edycyjnej Levensteina.

Zaimplementowany algorytm oprócz sprawdzania, czy słowo występuje w stworzonym Słowniku Frekwencyjnym Języka Polskiego (KGLK 2009), sprawdza również, czy dany wyraz został użyty w poprawnym kontekście. Takiego mechanizmu sprawdzania wprowadzonych wyrazów nie oferują inne pogramy.

Programy użyte do testów badają jedynie, czy wpisany przez użytkownika wyraz występuje w słowniku. Jeśli występuje, programy uważają go za poprawny. Tak więc użytkownik podczas wpisywania tekstu może popełnić błąd polegający na zmianie jednego wyrazu w drugi (również występujący w słowniku) lub napisanie słowa w niepoprawnej formie fleksyjnej - programy te nie będą w stanie takich błędów rozpoznać. Dla edytorów bowiem wyraz występujący w słowniku jest zawsze poprawny.

Drugim innowacyjnym mechanizmem jest dopełnianie aktualnie wprowadzonego wyrazu. Program Writer 3.0 oferuje podobną możliwość. Niestety automatyczne podpowiadanie wprowadzanego wyrazu występuje bardzo rzadko. Użytkownik, jeśli chce uzyskać informację o wprowadzanym wyrazie (w programach takich jak Microsoft Word 2007 oraz Writer 3.0), musi po wprowadzeniu jego początku kliknąć na nim prawym przyciskiem myszy i z pojawiających się wyrazów podpowiedzi wybrać pasujący wyraz. Nie jest to ściśle dokończeniem wyrazu, lecz znajdywaniem wyrazu najbliższego wprowadzanemu. Zaproponowany w tym artykule algorytm dysponuje mechanizmem automatycznej podpowiedzi dla wprowadzonego początku wyrazu. Dodatkowo sprawdza on kontekst początku wyrazu, więc prezentowane wyrazy nie tylko rozpoczynają się od wprowadzonych liter, ale również są poprawne w określonym kontekście z uwzględnieniem ich fleksji.

Podobnie jak z automatycznym sprawdzaniem wpisywanych wyrazów, również i z podpowiadaniem aktualnie wpisywanego wyrazu opracowany program, bazujący na opisanym w tym artykule i zaimplementowanym algorytmie oraz grafie LHG, radzi sobie znacznie lepiej niż programy Word czy Writer. W programach tych nie ma gotowych modułów służących do tego celu, a byłyby one bardzo przydatne. Stworzony

mechanizm dodatkowo analizuje wpisywany początek zdania i proponuje użytkownikowi jedynie te wyrazy, które występują w określonym kontekście wyrazów stojących bezpośrednio przed nim. Dla dłuższego zdania może się zdarzyć, że zaproponowane podpowiedzi nie będą pasowały kontekstowo do całego zdania. Sprawdzany jest bowiem kontekst trzech najbliższych wyrazów dla wpisywanego słowa. Program wyświetla wyniki właśnie dla takiego kontekstu. Analiza kontekstu całego zdania wymagałaby zapisywania znacznie szerszego kontekstu wypowiedzi w bazie, co niewątpliwie miałoby znaczenie m.in. dla szybkości działania całej aplikacji. Stąd została zawarta jedynie informacje o kontekście dla trzech najbliższych słów.

4.3. Automatyczna kontekstowa korekta

Podczas wprowadzania błędnego tekstu powstaje podstawowa trudność – jak zrozumieć i zinterpretować przekaz drugiej osoby? Człowiek w dość prosty sposób może dokonać pewnych przekształceń i „niejako odszyfrować" zaburzoną treść nadaną przez drugiego człowieka. Niestety, zrozumienie tego samego tekstu przez komputer okazuje się bardzo trudnym i złożonym zadaniem. Stąd też powstała potrzeba skonstruowania całego mechanizmu do automatycznej korekty wpisanego przez użytkownika tekstu, który zawiera różnorakie błędy. Niewątpliwie najważniejszym zadaniem było opracowanie specjalnego algorytmu do analizy już wprowadzonego tekstu. Algorytm ten miał za zadanie tak zmienić wpisany pierwotnie tekst, aby usunąć z niego wszystkie możliwe błędy zachowując kontekst zdania.

Na rysunku 5 oraz w tabelach 1 i 2 zaprezentowano wprowadzone błędne zdania oraz ich sugerowaną korektę przez:

- Microsoft Word 2007,
- Open.Office Writer 3.0,
- Wyszukiwarkę Google,
- Stworzoną aplikację wykorzystującą opisany w tym artykule algorytm oraz LHG.

Tabela 1
Proponowana korekta zdania „Oni mają bardo madre dzieic"

Wykorzystywane narzędzie	Zauważone błędy i proponowana korekta dla zdania „*Oni mają bardo madre dzieic*"
Microsoft Word 2007	**madre** – sugerowana poprawa: Madre, **dzieic** – sugerowana poprawa: dzieci, Dzielic, dziewic, Dzibic.
OpenOffice.org Writer 3.0	**bardo** – sugestia prawdopodobnej literówki, **madre** – sugerowana poprawa: mądre, mader, Mare made, **dzieic** – sugerowana poprawa: dzieci, dziec, dziewic.
Wyszukiwarka Google	**bardo** – sugerowana poprawa: bardzo, **dzieic** – sugerowana poprawa: dzieci.
Stworzona aplikacja	**bardo** – sugerowana poprawa: bardzo, **madre** – sugerowana poprawa: mądre, **dzieic** – sugerowana poprawa: dzieci.

Rys. 5. Proponowana korekta zdania „oin maj bardzo dobre wynyk”

Tabela 2
Proponowana korekta zdania „Idę teraz od szkoła”

Wykorzystywane narzędzie	Zauważone błędy i proponowana korekta dla zdania „*Idę teraz od szkoła*”
Microsoft Word 2007	Brak sugestii
OpenOffice.org Writer 3.0	**szkoła** – przyimek wymaga dopełniacza
Wyszukiwarka Google	Brak sugestii
Stworzona aplikacja	**od** – sugerowana poprawa: do, **szkoła** – sugerowana poprawa: szkoły.

5. Podsumowanie

Jednym z największych wyzwań obecnych czasów jest próba automatycznego przetworzenia i zrozumienia przez maszyny dostępnych informacji pochodzących z różnych źródeł. Występuje więc znaczne zapotrzebowanie na algorytmy i programy, które będą w stanie przetworzyć i skorygować wprowadzany przez użytkownika tekst. Ich zadaniem jest dokonanie takich operacji, aby nie zmieniając sensu całego tekstu przetworzyć go tak, aby był poprawny w danym języku naturalnym oraz zrozumiały dla drugiego człowieka. Niestety w Polsce nie ma jeszcze tak wyspecjalizowanych programów, które mogłyby to umożliwić. W tej pracy został położony szczególny nacisk na stworzenie takiego właśnie algorytmu oraz dodatkowo na porównanie jego działania z możliwościami korekty tekstu przez dwa wiodące edytory tekstu. Zaproponowany innowacyjny algorytm korekty tekstów bazuje na zaimplementowanym Grafie Przyzwyczajeń Lingwistycznych. Jak zostało wykazane, skonstruowany mechanizm kontekstowej korekty okazał się być znacznie bardziej skuteczny od konkurencji w rozpoznawaniu miejsc, w których wystąpiły błędy oraz w ich korekcie.

Na potrzeby przeprowadzonych badań powstały dwa programy:

1) **Pająk internetowy**, który przegląda strony internetowe w poszukiwaniu odnośników do dalszych stron oraz indeksuje i zapisuje w bazie danych wyszukane zdania w języku polskim. Pająk, dzięki zastosowaniu bazy danych, może działać z przerwami – stąd możliwe jest jego czasowe wyłączenie i ponowne włączenie. Dzięki temu baza słów może być nieustannie poszerzana o kolejne słowa w ich unikalnym kontekście oraz LHG może być w razie potrzeby powiększany.
2) **Program do automatycznej kontekstowej korekcji tekstu**, który służy do poprawy błędnie wprowadzonych zdań. Korekta tekstu możliwa jest dzięki wykorzystaniu Grafu Przyzwyczajeń Lingwistycznych dla języka polskiego. Program ten składa się z trzech modułów:
   - modułu do tworzenia i przeglądania Grafu Przyzwyczajeń Lingwistycznych,
   - modułu do automatycznego sprawdzania poprawności wprowadzonych wyrazów,
   - modułu do podpowiadania końca słów, jak i do korekcji całego wprowadzonego tekstu.

Do zalet wyróżniających zaproponowany algorytm pośród innych, należą:

- moduł do tworzenia nowych zdań,
- innowacyjny sposób wykrywania potencjalnych błędów w tekście,
- inteligentne dopełnianie wyrazów,
- automatyczna kontekstowa korekta tekstów.

Wyszukiwanie informacji w grafie LHG nie zajmuje wiele czasu. Sprawdzanie, czy słowo występuje w słowniku języka polskiego oraz czy jest w określonym kontekście, zajmuje średnio ponad jedną sekundę – co jest w pełni akceptowalne z praktycznego punktu widzenia. Proces uzyskiwania podpowiedzi końcówek do wprowadzonego początku słowa trwa od jednej do dwóch sekund, natomiast etap automatycznej kontekstowej korekcji tekstu składającego się trzech zdań trwa średnio od trzech do pięciu sekund na przeciętnym komputerze PC. Można zatem stwierdzić, że zostały spełnione wymagania czasowe stawiane przez potencjalnego użytkownika.

Nie było możliwe uzyskanie bardzo dobrej automatycznej korekcji tekstu dla całych zdań, które występują rzadko. Algorytm korzysta jedynie z wiedzy zapisanej w bazie danych. Jeśli więc wpisane przez użytkownika wyrazy nie wystąpiły w bazie, program nie będzie w stanie ich poprawić ani zaproponować ich uzupełnienia. Jeśli natomiast określony fragment zdania występował w bazie bardzo rzadko, to użytkownikowi zostanie zaprezentowana poprawa właśnie w kontekstach występujących w bazie.

Istnieje szereg możliwość rozbudowania tego algorytmu i programu tak, aby jeszcze lepiej potrafił dokonywać korekty tekstu dla większej ilości zdań. Do możliwości przyszłego rozwoju, powstałego już algorytmu i aplikacji, można zaliczyć:

- **Poszerzenie bazy wiedzy o nowe wyrazy**. Najprostszym rozwiązaniem wydaje się być ciągłe uzupełnianie bazy wiedzy o analizę i przetworzenie nowych zdań i dalsze rozbudowywanie grafu LHG. Dzięki temu, możliwe będzie prowadzenie poprawnej korekty większej ilości tekstów. Niestety takie rozwiązanie pociąga za sobą ciągłe jej rozbudowywanie, co powoduje zwiększenie zajętości dysku przez graf oraz przede wszystkim zwiększa czas wyszukiwania wyrazów, przez co program może wolniej działać. Dzięki wykorzystaniu zaproponowanego grafu LHG informacje o trójkach słownych są w dużej mierze kompresowane, co prowadzi do dużej oszczędności pamięci do przechowywania tych danych w porównaniu do konieczności przechowywania wszystkich trójek słownych w bazie danych.
- **Opracowanie lepszej metody automatycznej korekty**. Dotychczasowo algorytm wykorzystywał informację o częstości pojawiania się słów w określonym kontekście, ich odległość edycyjną oraz długość słowa. Możliwe jest zmodyfikowanie mechanizmu w przyszłości tak, aby podczas etapu korekty korzystał z wielu metod algorytmów na raz w odpowiedniej kolejności.
- **Analizę możliwych wystąpień części mowy w zdaniu**. Na podstawie istniejącej już bazy danych można każdemu słowu przypisać jego część mowy. Na tej podstawie można stwierdzić, czy możliwa jest konstrukcja zdania, w któ-

rym występowałyby, np. odpowiednio dane części mowy rzeczownik, przymiotnik, czasownik itp. Wynika z tego, że wprowadzenie takiego rozwiązania dałoby możliwość analizy i poprawnej korekty zdania, którego wyrazy nie wystąpiły dotychczas w bazie kontekstów słownych.

- **Wprowadzenie i analiza reguł budowy zdań (zastosowanie sztucznej inteligencji, sieci semantycznej)**. Rozwiązaniem, które powinno wprowadzić najlepszą możliwą automatyczną kontekstową korektę tekstu, będzie opracowanie i stworzenie reguł korekty. Będą one utworzone na podstawie danych zawartych w bazie wiedzy o określonym języku. Dzięki nim będzie możliwe określenie zasad, które są wykorzystywane podczas tworzenia zdań. Technologie takie, jak Semantic Web, dają możliwość łączenia i wnioskowania na podstawie rozproszonych źródeł danych, będą więc stanowiły podstawę dalszego rozwoju algorytmu i aplikacji. Również zaimplementowana w niej sztuczna inteligencja, będzie w stanie na bieżąco uczyć się możliwych konstrukcji i coraz lepiej i bardziej automatycznie poprawiać błędy.

Zaproponowany Graf Przyzwyczajeń Lingwistycznych oraz późniejsze jego wykorzystanie dla zaimplementowanego innowacyjnego algorytmu poprawy tekstu dałoby możliwość skonstruowania aplikacji, która wykorzystując kilka mechanizmów, będzie znacząco lepiej i bardziej automatycznie dokonywać korekty błędnych tekstów napisanych w języku polskim, dzięki wykorzystaniu kontekstu słów w określonych formach fleksyjnych. Obecnie na rynku dostępnych jest kilka edytorów tekstu dla języka polskiego, które posiadają zaimplementowane różne mechanizmy automatycznej jego korekty. Niestety żaden z nich nie potrafi dokonywać kontekstowej korekty wprowadzonego tekstu. Mechanizmem korekty błędnie wprowadzonego tekstu (również kontekstowo) dysponuje wyszukiwarka Google. Nie jest jednak możliwe stwierdzenie, że może ona być użyta jako edytor tekstu lub do sprawdzania dłuższych tekstów. Dodatkowo w wielu miejscach mechanizm w niej zaimplementowany nie wykrywa błędów.

Analizując rysunek 5 oraz tabele 1 i 2 można zauważyć, że najlepsze wyniki korekty tekstu dla zdań przeczytanych przez stworzonego wcześniej pająka, osiągnięto przez skonstruowany w tej pracy algorytm. Wobec powyższego można wyprowadzić następujący wniosek, iż jeżeliby pająk internetowy jeszcze dłużej czytał teksty, konstruując przy tym bardziej rozbudowany graf LHG, wtedy opracowane mechanizmy byłyby w stanie jeszcze lepiej i bardziej automatycznie korygować teksty zastępując lub uzupełniając obecnie stosowane rozwiązania.

Literatura

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

[2] Miró J., Rosselló F.: *Czy w Unii Europejskiej mówiono po polsku?*. Magazyn Delta, 05, 2004

[3] Gawrysiak P.: *Modelowanie języka*. Politechnika Warszawska, 2006

[4] Statistical Inference: n-gram Models over Sparse Data: `http://mi007.wikispaces.com/file/view/rozdzial6.pdf`, 2009

[5] Dębowski Ł.: *Prawo Zipfa – próby objaśnień*. Instytut Podstaw Informatyki PAN, 2005

[6] Microsoft Office Word 2007 2009: *Opis programu Word*. `http://office.microsoft.com/pl-pl/word/HA101650321045.aspx`

[7] OpenOffice.org Writer 2009: *Opis programu*. `http://pl.openoffice.org/`

[8] Marciniak M.: *MS Office kontra OpenOffice*. PC Word 2000

[9] KGLK Krakowska Grupa Lingwistyki Komputerowej: *Słownik Frekwencyjny Języka Polskiego*, 2009