POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Inżynieria systemów informatycznych (INS)

Projektowanie systemów z dostępem w języku naturalnym

Opracowanie algorytmu określającego możliwe zamienniki dla błędnych słów w tekstach w języku polskim

AUTOR:
Radosław Taborski - 209347
Piotr Konieczny - 209174

PROWADZĄCY PROJEKT:
dr inż. Dariusz Banasiak

OCENA PROJEKTU:

WROCŁAW 2018

Spis treści

Spis rysunków

Spis tabel

Spis listingów

Rozdział 1

Wstęp

1.1 Cele projektu

Głównym celem projektu jest stworzenie aplikacji, która będzie umożliwiała wpisanie przez użytkownika dowolnego tekstu w języku polskim oraz pozwoli na jego sprawdzenie pod względem użycia poprawnych słów. Następnie użytkownik będzie miał możliwość wstawienia za każde z wychwyconych błędnych wyrazów, innego, jednego z tych, które będą sugerowane przez aplikację, bądź też zignorowanie sugestii. Aplikacja wyposażona będzie w algorytm sprawdzający czy wyraz istnieje w słowniku języka polskiego, gdy takiego nie znajdzie, wyszukiwane będą słowa o podobnej budowie, lecz nie koniecznie pasujące znaczeniowo do kontekstu zdania.

1.2 Założenia projektowe

W projekcie wykorzystany został język programowania C#, z wykorzystaniem technologii WPF, umożliwiającej tworzenie graficznego interfejsu użytkownika w oparciu o składnię XAML.

W projekcie wykorzystany został również słownik języka polskiego dostępny na stronie: *http://sjp.pl/slownik/growy*. Słownik ten zawiera podstawowe formy słów wraz ze wszystkimi możliwymi odmianami. Słownik ten przewiduje jedynie słowa o maksymalnej długości wynoszącej 15 znaków i nie zawiera żadnych nazw własnych tzn. imion, nazw krajów, miast itp.

1.3 Zakres projektu

Zakres projektu dotyczy zaprojektowania i implementacji aplikacji umożliwiającej pisanie własnych tekstów, które na bieżąco są sprawdzane pod względem poprawności ortograficznej.

Umożliwione użytkownikowi zostało również wklejanie gotowych tekstów, w celu ich sprawdzenia.

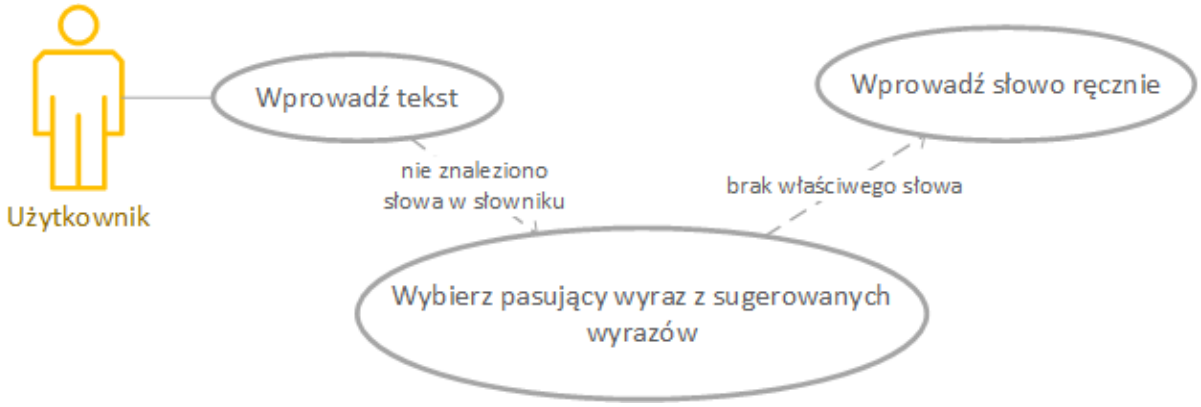
Rozdział 2

Opis użytych algorytmów

Rozdział 3

Projekt i implementacja aplikacji

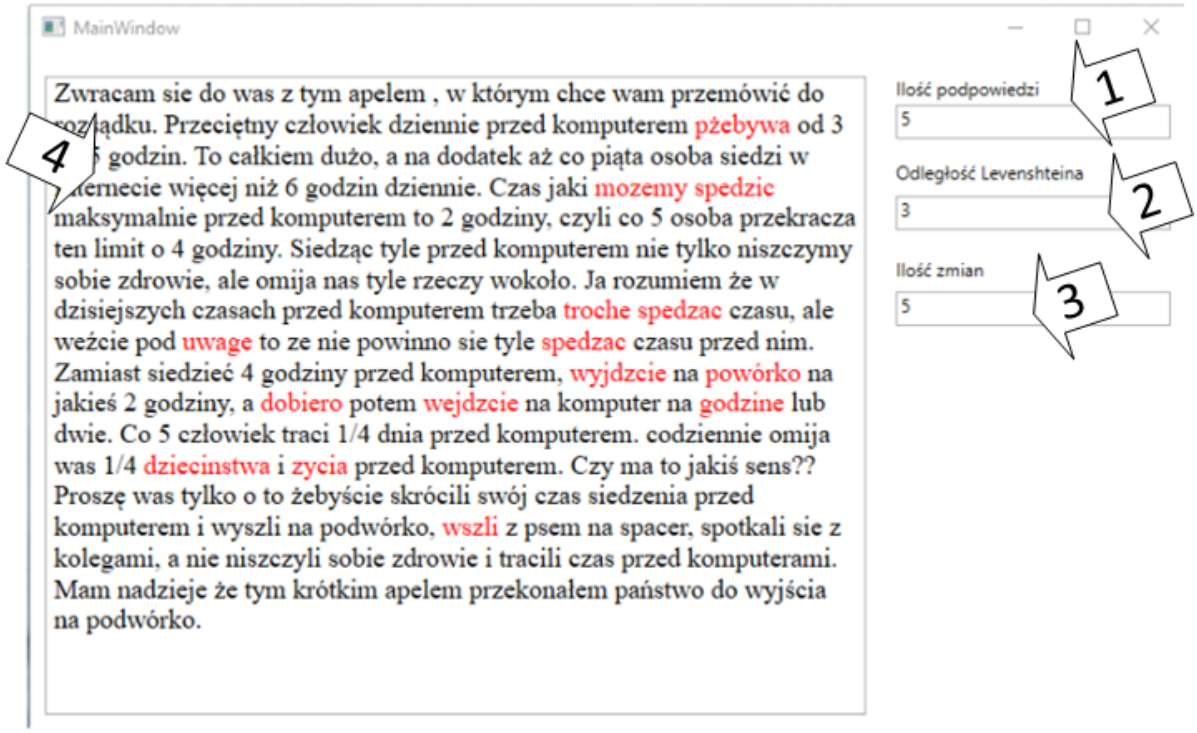
3.1 Funkcje aplikacji - diagram przypadków użycia



Rysunek 3.1 Diagram przypadków użycia

Głównym zadaniem aplikacji jest wyszukiwanie błędnych wyrazów w języku polskim oraz ich korekcja. Aby tego dokonać aplikacja porównuje wyrazy znajdujące się w tekście z wyrazami znajdującymi się w słowniku. Jeśli danego słowa nie znaleziono zostaje ono uznane za błędne. Aby dokonać korekcji błędnych wyrazów stosowane są algorytmy opisane w poprzednich rozdziałach. Dla optymalizacji działania aplikacji wyszukiwanie sugestii odbywa się w osobnych wątkach co znacząco przyspiesza działanie aplikacji.

3.2 Interfejs aplikacji



Rysunek 3.2 Interfejs aplikacji

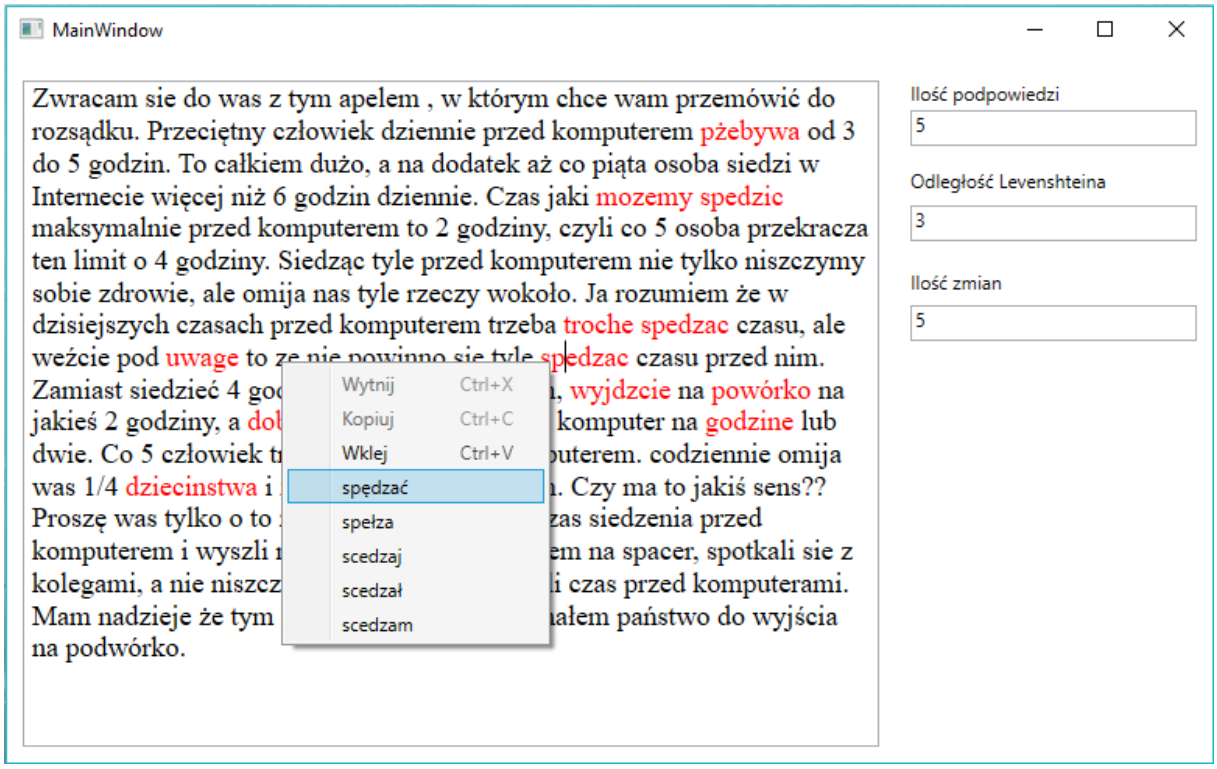
1. Ilość podpowiedzi - pozwala na ustawienie maksymalnej ilości podpowiedzi jakie mają się pojawić po kliknięciu prawym przyciskiem myszy na błędnie napisane słowo.

2. Odległość Levenhteina - pozwala na ustawienie odległości jaka ma być brana pod uwagę w przypadku algorytmu Levenhstaina. Im większa liczba tym więcej podpowiedzi ale jednocześnie zmniejsza to szybkość działania aplikacji ze względu na dodatkowe obliczenia które muszą zostać wykonane.

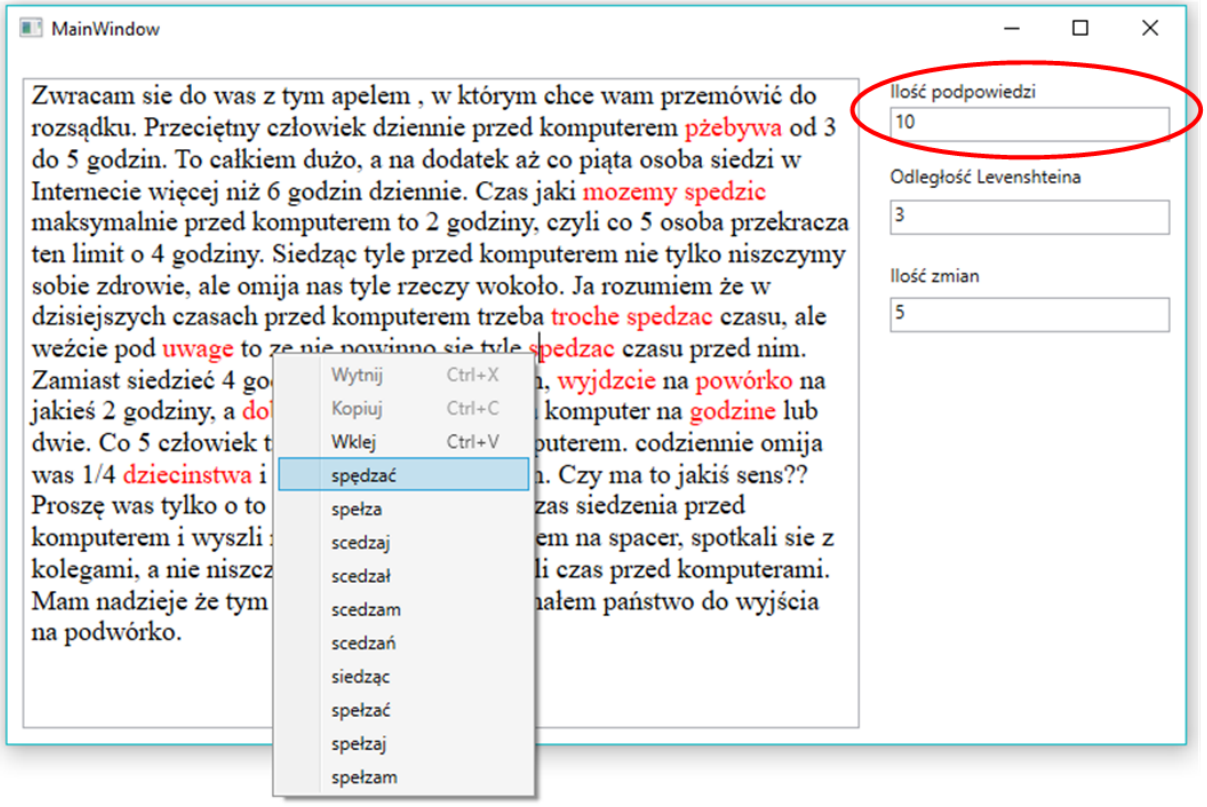
3. Ilość zmian - pozwala na ustawienie parametru ilości zmian dla algorytmu podmieniającego znaki diakrytyczne. Im większa liczba tym więcej podpowiedzi oraz dłuższy czas wykonywania algorytmu.

4. Edytor tekstu - pozwala na edycję tekstu. Wyszukiwanie błędów działa w czasie rzeczywistym (funkcja sprawdzająca poprawność uruchamia się z każdym kliknięciem spacji). Jeśli dane słowo zostało uznane za błędne zostaje zaznaczone kolorem czerwonym. Aby je poprawić należy kliknąć na nie prawym przyciskiem myszki. Zostanie wyświetlone menu kontekstowe zawierające możliwe zamienniki. Aby podmienić słowo wystarczy kliknąć na zamiennik. Możliwe jest również poprawienie tekstu ręcznie wpisując w miejscu błędnego słowa poprawne.

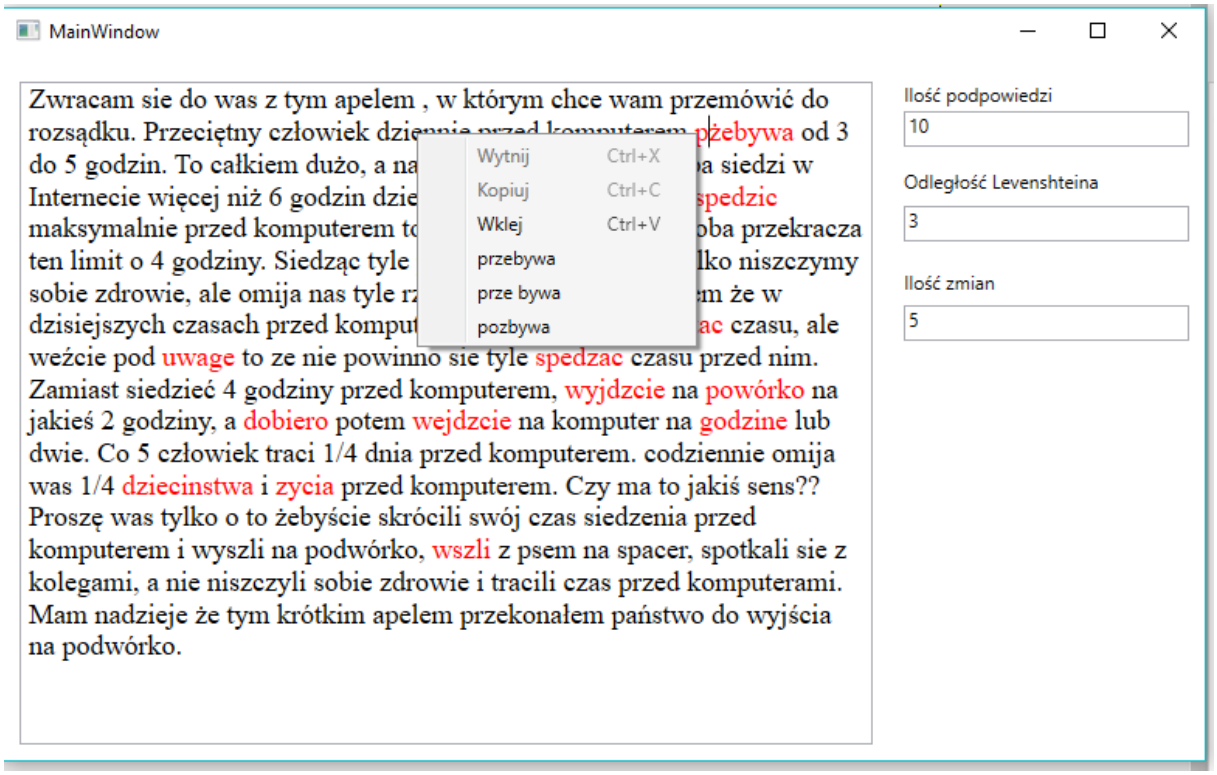
3.3 Wpływ parametrów na wyniki wyszukiwania



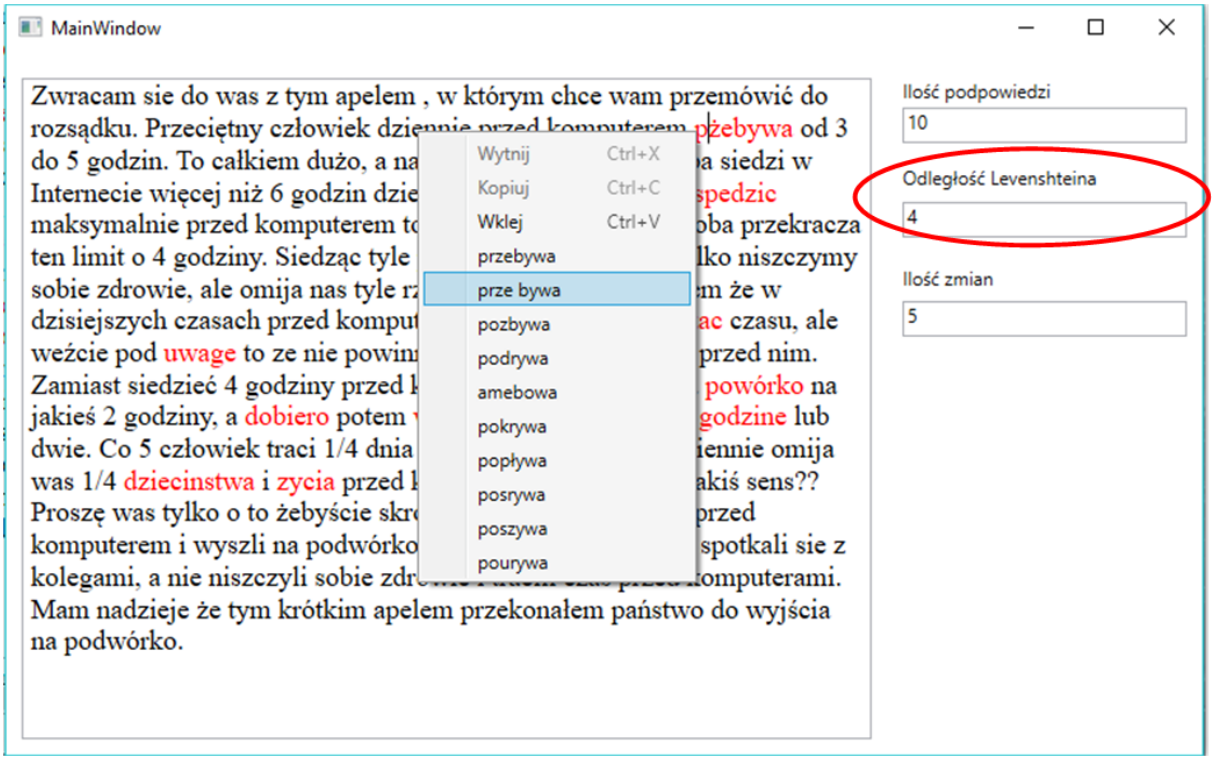
Rysunek 3.3 Przykładowy wynik wyszukiwania sugestii



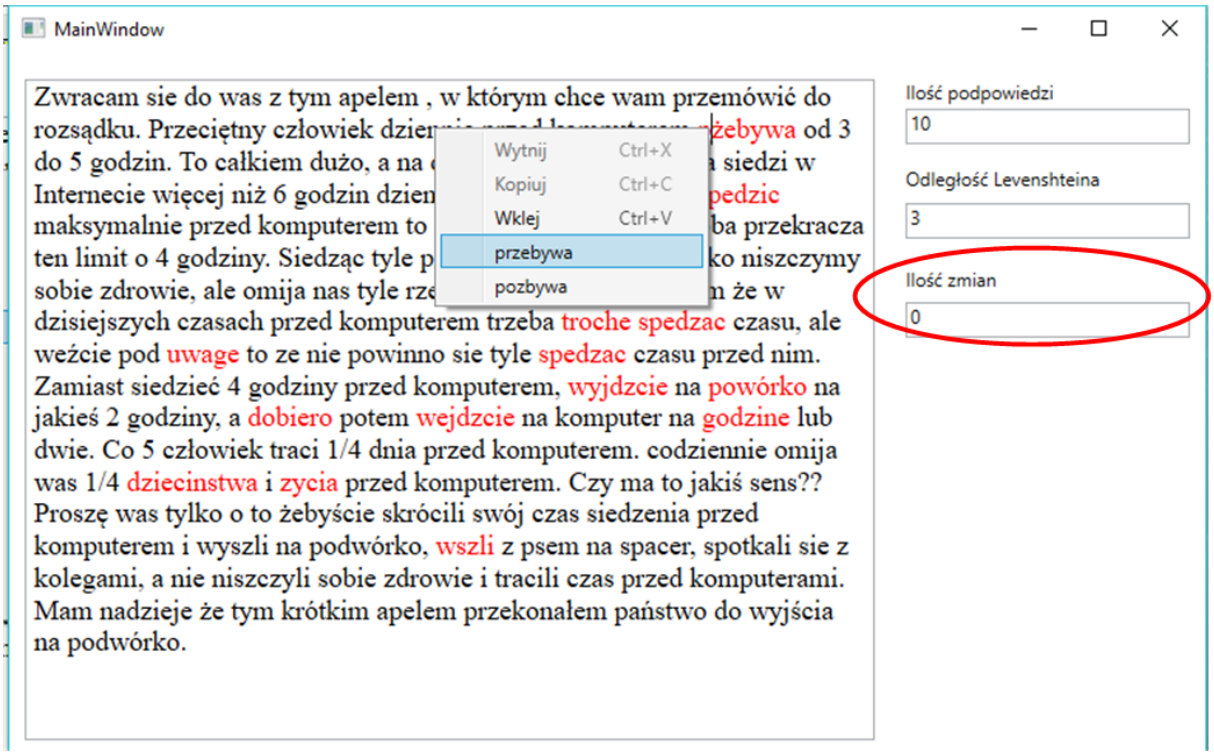
Rysunek 3.4 Zwiększenie ilości wyświetlanych sugestii



Rysunek 3.5 Przykładowy wynik wyszukiwania sugestii dla małej ilości podpowiedzi



Rysunek 3.6 Zwiększenie odległości Levenhstaina

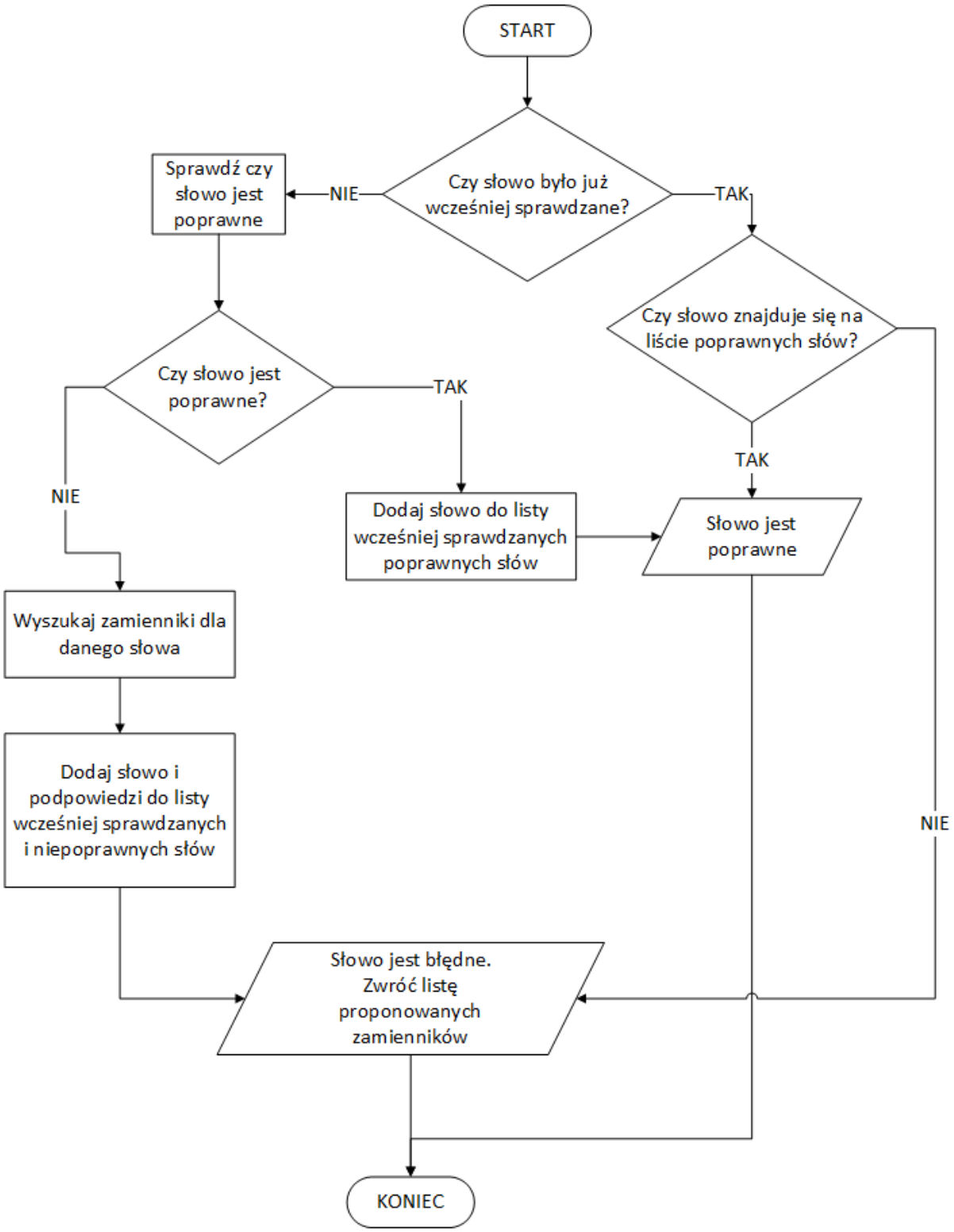


Rysunek 3.7 Zwiększenie ilości zmian

Odpowiednio zmieniając parametry znajdujące się w prawej części okna aplikacji można dostosować wyszukiwanie do potrzeb użytkownika. Rysunek 3.3 przedstawia przykładowe wyniki wyszukiwania podpowiedzi. Jeśli ilość wyświetlanych wyników nie jest odpowiednia można zmienić ich ilość tak aby uzyskać satysfakcjonujący wynik. Rysunek 3.4 przedstawia wpływ zwiększenia ilości wyświetlanych podpowiedzi. Jednocześnie ilość podpowiedzi nie ma wpływu na szybkość działania aplikacji.

Pozostałe parametry wpływają na podobieństwo sugestii do wprowadzonego błędnie słowa. Na rysunku 3.5 został przedstawiony przykładowy wynik wyszukiwania który przy danych parametrach nie wypełnia całej listy podpowiedzi. Na rysunku 3.6 zwiększono odległość Levenhstaina tak aby uzyskać więcej sugestii. Natomiast na rysunku 3.7 całkowicie zminimalizowano wpływ algorytmu podmieniającego znaki tak aby pozbyć się niechcianych podpowiedzi zawierających spacje.

3.4 Realizacja wybranych funkcjonalności



Rysunek 3.8 Schemat korekcji tekstu z poziomu interfejsu

Główną funkcjonalnością aplikacji jest wyszukiwanie a następnie korekcja błędów. Rysunek 3.8 przedstawia schemat działania algorytmu odpowiedzialnego za powyższe zadanie.

Początkowo algorytm sprawdza czy dane słowo było już wcześniej wyszukiwane. Jeśli tak to sprawdza czy jest ono na liście poprawnych słów. Jeśli tak to słowo jest poprawne i algorytm kończy pracę. Jeśli nie to sprawdza on listę błędnie wprowadzonych słów i zwraca on podpowiedzi. Jeśli podpowiedzi zostały już raz wyszukane dla danego słowa to zostają one zapisane. Ma to na celu przyspieszenie działania algorytmu ponieważ najwięcej czasu pochłania wyszukiwanie sugestii.

Jeśli słowo jest wyszukiwane po raz pierwszy następuje sprawdzenie jego poprawności porównując je ze słowami znajdującymi się w słowniku. Jeśli jest poprawne to zostaje ono dodane do listy poprawnych sprawdzanych wcześniej słów i algorytm kończy pracę. Natomiast jeśli słowo jest błędne następuje wyszukiwanie sugestii. Po ich wyszukaniu zostają on zapisane wraz z niepoprawnym wyrazem na liście błędnych słów. Algorytm zwraca listę sugestii i kończy pracę.

Algorytm ten działa dla każdego słowa w osobnym wątku. Ma to za zadnie przyspieszenie działania aplikacji.

Rozdział 4

Optymalizacja

Rozdział 5

Wdrożenie i testowanie aplikacji

Rozdział 6

Podsumowanie

```
1 przyklad
2 listingu
3 pamietaj zmieniac nazwy labelek!!! :D
```

Listing 6.1 Przykład listingu

Literatura

[1] Anonim G. i in., *Kronika polska*, T.1: *Kronika i czyny książąt czyli władców polskich*, Gniezno 1115