P. M.

Spellcorrector

Implementacja spellcorrector-a dla języka polskiego.

1.Wstęp

Spellcorrector – Narzędzie do poprawy pisowni. Za zadanie ma wygenerowanie jednego lub więcej kandydatów do poprawy błędnego słowa. Powinien działać wystarczająco szybko, aby jego używanie nie było uciążliwe.

2.Rozwinięcie

Mój projekt oparłem rozwiązaniu ze strony http://www.norvig.com/spell-correct.html [1]. Program stara się zaproponować, dla słowa nie znanego, wszystkie znane słowa-kandydaty. Projekt przeznaczony do poprawy pisowni w języku polskim.

Pierwsze program pobiera słownik[2], listę częstotliwości słów [3] oraz listę liter występujących w języku polskim. Do wyboru mamy 3 funkcje check, check1 oraz check2. Check jest główną funkcją, która przyjmuje jako argument, oprócz słowa, ilość zwracanych kandydatów oraz umożliwia wybór używanej funkcji check2 (domyślna) lub check1. Check1/2 są to funkcje, które zwracają listy kandydatów i różnią się „odległością” generowanych kandydatów. Check1 generuje listę kandydatów o tylko 1 edycji, a check2 o 1 i 2 edycjach od słowa pierwotnego. Po wybraniu funkcji pobiera program słowo do sprawdzenia poprawności i jeśli nie znajduję się ono w słowniku to generowana jest lista kandydatów. List kandydatów jest tworzona przez poddanie słowa 4 operacjom – dodanie, zamiana, podstawienie i usunięcie litery. Każda operacja generuje listę kandydatów, i jeśli została wybrana funkcja check2 , każdy z wygenerowanych kandydatów poddawany takiemu samemu procesowi jeszcze raz i lista kandydatów z obu stopni są scalane. Lista wynikowa może mieć od kilkuset do nawet kilkuset tysięcy kandydatów w zależności długości słowa, z takiej listy generujemy listę tylko znanych kandydatów i przypisujemy im częstotliwość, która dla każdego kandydata odległego o 2 edycje jest dzielona przez 100, oraz sortujemy listę według niej. Z tak otrzymanej listy może wybrać max(), aby otrzymać najbardziej prawdopodobnego kandydata na poprawkę.

Przykład wygenerowanych kandydatów

'aszzcoła', 'apszzcoła', 'ąszzcoła', 'ąpszzcoła', 'bszzcoła', 'bpszzcoła', 'cszzcoła', 'cpszzcoła', 'ćszzcoła', 'ćpszzcoła', 'dszzcoła', 'dpszzcoła', 'eszzcoła', 'epszzcoła', 'ęszzcoła', 'ępszzcoła', 'fszzcoła', 'fpszzcoła', 'gszzcoła', 'gpszzcoła', 'hszzcoła', 'hpszzcoła', 'iszzcoła', 'ipszzcoła', 'jszzcoła', 'jpszzcoła', 'kszzcoła', 'kpszzcoła', 'lszzcoła', 'lpszzcoła', 'łszzcoła', 'łpszzcoła', 'mszzcoła', 'mpszzcoła', 'nszzcoła', 'npszzcoła', 'ńszzcoła', 'ńpszzcoła', 'oszzcoła', 'opszzcoła', 'ószzcoła', 'ópszzcoła', 'pszzcoła', 'ppszzcoła', 'qszzcoła', 'qpszzcoła', 'rszzcoła', 'rpszzcoła', 'sszzcoła', 'spszzcoła', 'śszzcoła', 'śpszzcoła', 'tszzcoła', 'tpszzcoła', 'uszzcoła', 'upszzcoła'

Przykład znanych kandydatów

Dla „pszzcoła”

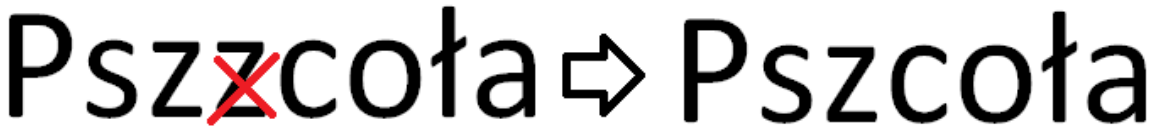
```
{'pszczoła': 1.0}
```

Dla „anglielsky”

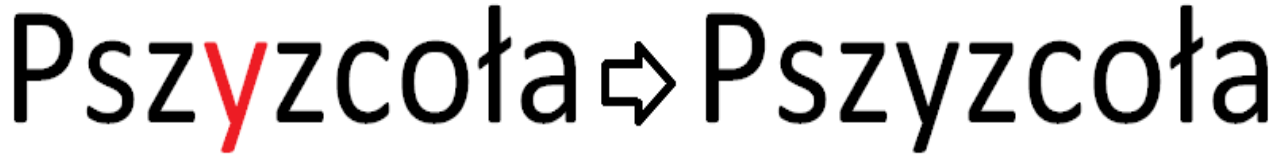
{'angielski': 0.07, 'angielscy': 0.01, 'angielska': 0.01, 'angielską': 0.01, 'angielsko': 0.01, 'angielsku': 0.01}

Rodzaje operacji na przykładzie „Pszzcoła”

1. Usuń

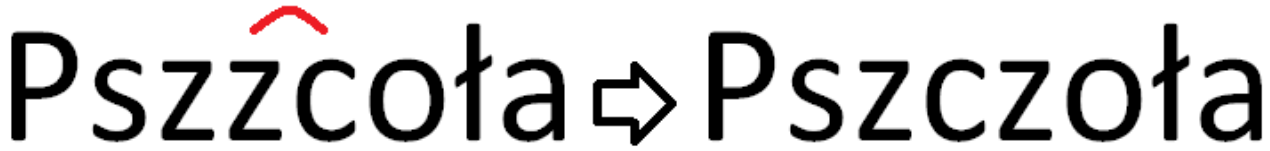


2. Dodaj



3. Podmień

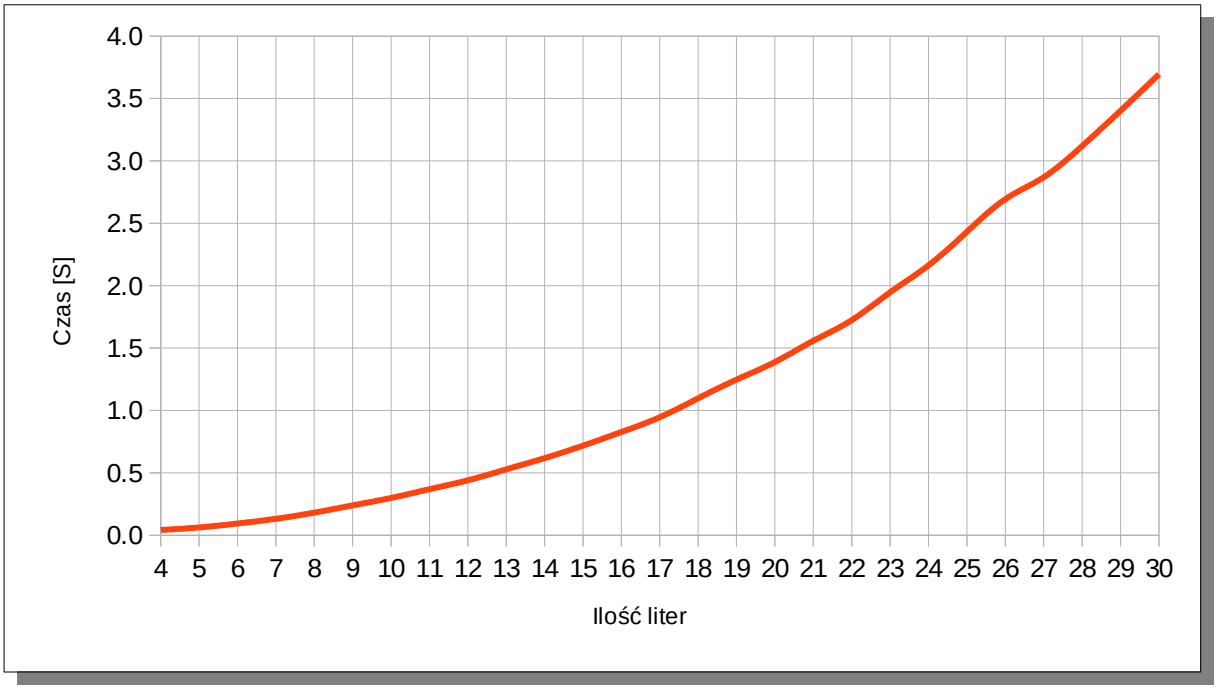
4. Zamień



3. Podsumowanie

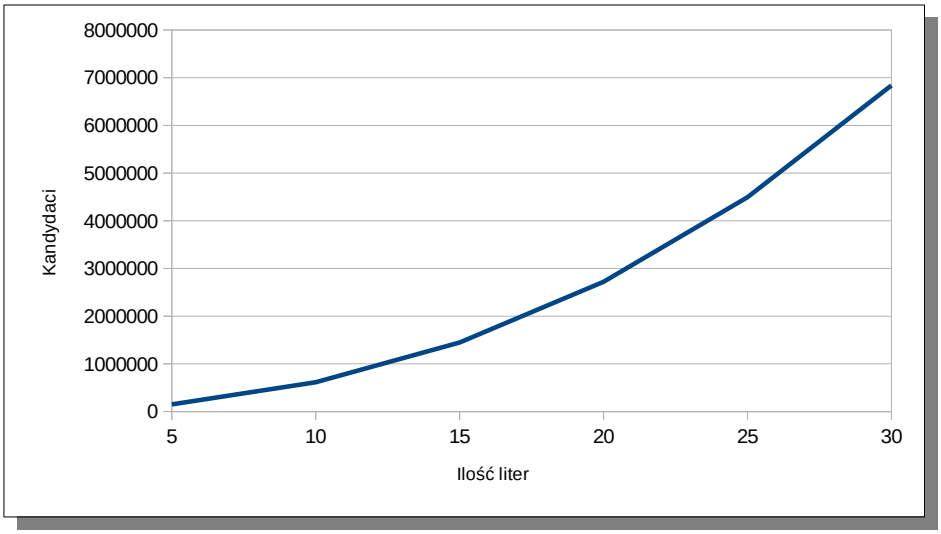
A) Pomiary

Czas wykonywania w zależności od długości



Ilość kandydatów w zależności od ilości liter

Ilość liter	Kandydaci
5	150 365
10	613 880
15	1 449 795
20	2 721 110
25	4 494 575
30	6 840 690



Wszystkie testy były wykonywane w notebooku Jupytera. Dla długich słów czas znajdywania znanych kandydatów jest duży i w przypadku dużej liczby długich słów do sprawdzenia czas wykonywania programu będzie bardzo długi. Dla słów o średniej wielkości ~5 liter czas szukania to około 50~70mS. Z wykresu można zauważyć nie liniową zależność, która zaczyna bardzo szybko rosnąc od około 20 znakowego słowa. Takich słów w języku polskim jest stosunkowo mało, więc wpływ na działanie w większości wypadków będzie minimalny.

B) Możliwe ulepszenia

- **Analizowanie sąsiednich słów –** W celu zwiększenia jakości zaproponowanych kandydatów
- **Implementacja stemmingu –** Aby poprawić jakość znajdywania oraz szybkość. Aktualny słownik ze wszystkimi odmianami ma ponad 3 miliony słów.
- **Własny korpus oraz lista frekwencyjna wyrazów –** Umożliwi lepszy wybór znanych kandydatów. Aktualnie dla wielu słów nie mam częstości występowania, więc i sugestie stają się mniej dokładne
- **Dalsze optymalizacje algorytmu –** Poprawić operacje dodaj, usuń, zamień i podstaw aby czas wykonywania był bliższy liniowemu.
- **Implementacji wielowątkowości**

C) Stopień realizacji – Projekt udało się w pełni zrealizować

4.Bibliografia

[1] http://www.norvig.com/spell-correct.html

[2] https://sjp.pl/sl/growy/

[3]https://zasobynauki.pl/zasoby/listy-frekwencyjne-z-korpusow-tekstu,18459/