

Spell corrector

Kamil Naglik

Maciej Rybiński

Marcelina Stabryła

1. Wstęp

Spell corrector to algorytm komputerowy, którego celem jest automatyczna poprawa błędów pisowni w tekście zwiększając tym samym czytelność i poprawność tekstu. W naszym programie zastosowaliśmy analizę prawdopodobieństwa do wyboru najbardziej prawdopodobnej poprawnej formy słowa.

W raporcie przedstawimy proces tworzenia tego algorytmu, omówimy jego zastosowanie oraz przedstawimy wyniki działania oraz analizę skuteczności naszego programu.

2. Cel

Celem projektu jest stworzenie spell correctora, czyli narzędzia, które poprawia błędy ortograficzne w tekście. Spell corrector ma za zadanie identyfikować słowa napisane błędnie i proponować poprawne wersje na podstawie prawdopodobieństwa ich wystąpienia w danym języku. Celem projektu jest dalsza analiza różnych statystyk które wynikają z poprawiania tekstu.

3. Zakres

Projekt obejmuje analizę tekstu pod kątem błędów ortograficznych i literówek oraz implementację spell correctora przy użyciu modelu probabilistycznego, a następnie analizę otrzymanych wyników, czyli statystyk z działania programu przy różnych danych wejściowych. Kluczowym elementem jest wykorzystanie korpusu tekstu (w tym przypadku pliku 'big.txt') do zbudowania słownika oraz określenia prawdopodobieństw wystąpienia poszczególnych słów. Algorytm opiera się na generowaniu możliwych poprawek dla błędnych słów i wybraniu tej, która ma największe prawdopodobieństwo.

4. Część Teoretyczna

Projekt może być używany do analizy dowolnego tekstu w poszukiwaniu błędów ortograficznych lub literówek oraz dynamicznego poprawiania tekstu. Algorytm oparty jest na modelu probabilistycznym, gdzie słowa są rozpatrywane w kontekście ich prawdopodobieństwa wystąpienia w danym języku. Korzystając z koncepcji statystyki korpusowej, spell corrector stara się maksymalizować prawdopodobieństwo poprawnej korekty.

Analiza prawdopodobieństwa

W naszym podejściu do korekty pisowni kluczową rolę odgrywa analiza prawdopodobieństwa, która opiera się na statystycznej ocenie, jak często poszczególne słowa występują w języku naturalnym. Dla każdego słowa w obliczamy jego prawdopodobieństwo wystąpienia $P(w)$ na podstawie danych uzyskanych z bazy tekstowej ('big.txt'). Wykorzystanie tej informacji pozwala na bardziej precyzyjne określenie, które słowo jest bardziej prawdopodobne w danym kontekście.

Twierdzenie Bayesa w algorytmie korekty

Twierdzenie Bayesa do procesu wyboru korekty pozwala na uwzględnienie kontekstu i relacji między kandydatami na poprawki a prawdopodobieństwem wystąpienia błędu. Wybierając najbardziej prawdopodobną korektę, stosujemy twierdzenie do oceny prawdopodobieństwa, że dana korekta c jest właściwym wyborem w danym kontekście $P(c|w)$. Ostateczne korekty są wybierane na podstawie równania:

$$\operatorname{argmax}_{c \in \text{candidates}} P(c) \cdot P(w | c)$$

Składniki Wyrażenia

- **Mechanizm Wyboru:** argmax - Mechanizm ten pozwala na wybór korekty, której kombinacja prawdopodobieństw jest najwyższa.
- **Model Kandydata:** $\in \text{candidates}$ - Określamy kandydatów na poprawki, biorąc pod uwagę możliwe edycje słowa.
- **Model Języka:** $P(c)$ - Opisuje prawdopodobieństwo, że kandydat c występuje jako słowo w języku angielskim.
- **Model Błędu:** $P(w|c)$ - Określa prawdopodobieństwo, że słowo zostanie wpisane z błędem, gdy miało to być słowo c .

Algorytm generowania kandydatów

Kluczowym elementem algorytmu jest generowanie kandydatów na poprawki. Ten proces działa na zasadzie uwzględniania znanych słów, błędów edycyjnych oraz ograniczania przestrzeni poszukiwań poprzez uwzględnienie jedynie słów znajdujących się w korpusie.

Funkcja `generate_edits(word)` odpowiada za generowanie kandydatów korekt na podstawie jednej operacji edycyjnej dla danego słowa. Operacje edycyjne obejmują usuwanie litery, zamianę dwóch sąsiednich liter, zamianę litery na inną oraz wstawianie litery. Funkcja ta tworzy zbiór potencjalnych korekt, uwzględniając różne możliwości błędów pisowni.

Funkcja `correction(word, word_counter)` wybiera najbardziej prawdopodobną korektę dla danego słowa. W procesie wyboru korekty, funkcja uwzględnia znane słowa, korekty generowane przez funkcję `generate_edits(word)`, oraz zachowuje oryginalne słowo, jeśli żadne z wcześniejszych warunków nie jest spełniony. Wybór oparty jest na analizie prawdopodobieństw występowania słów w korpusie tekstowym.

Funkcja `count_corrections(words_to_analyze, word_counter)` analizuje listę słów w tekście, korzystając z funkcji `correction`, a następnie zlicza wystąpienia korekt dla każdego słowa. Rezultatem jest `Counter`, który przechowuje informacje o ilości poprawek dla poszczególnych słów.

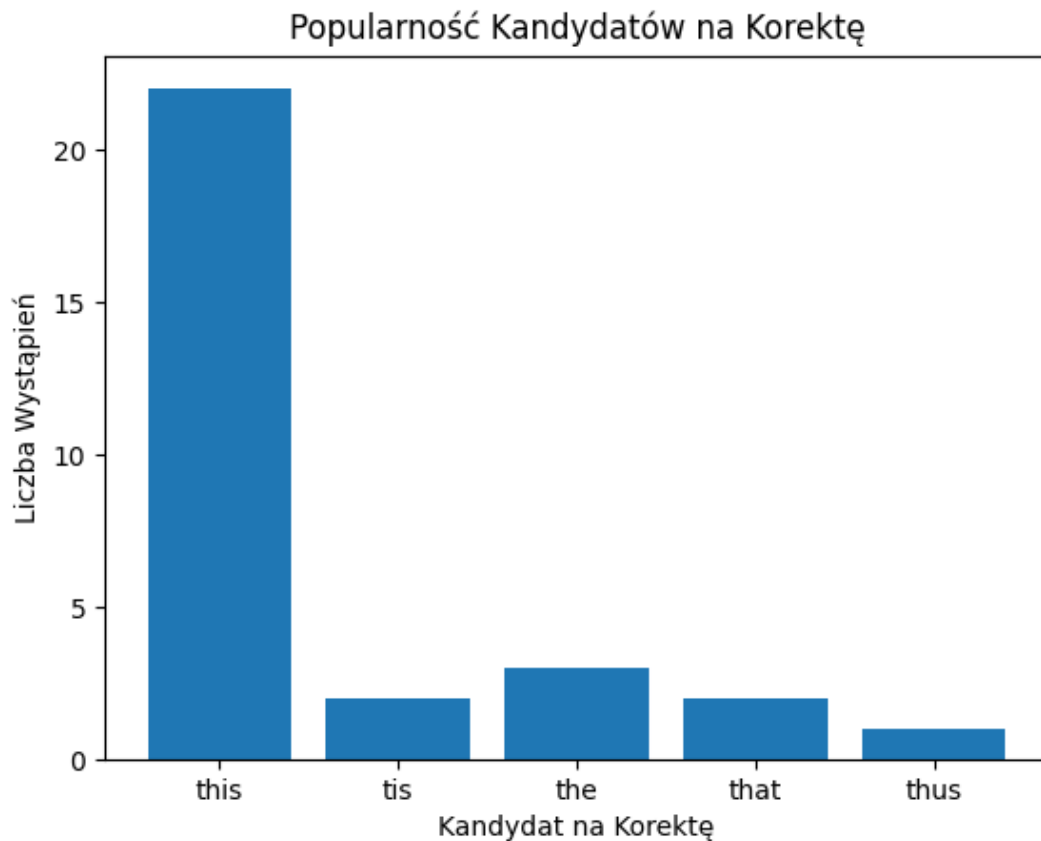
Funkcja `analyze_candidate_popularity(text, word_counter)` analizuje popularność kandydatów na korekty w danym tekście. Wykorzystuje funkcję `correction` do wygenerowania korekt dla poszczególnych słów, a następnie tworzy wykres, prezentując popularność korekt na podstawie ich liczby wystąpień.

Funkcja `display_corrected_text(text, word_counter)` wykorzystuje funkcję `correction` do skorygowania tekstu na podstawie najbardziej prawdopodobnych korekt dla poszczególnych słów. Wyświetla oryginalny tekst oraz skorygowaną wersję, umożliwiając porównanie.

5. Część praktyczna

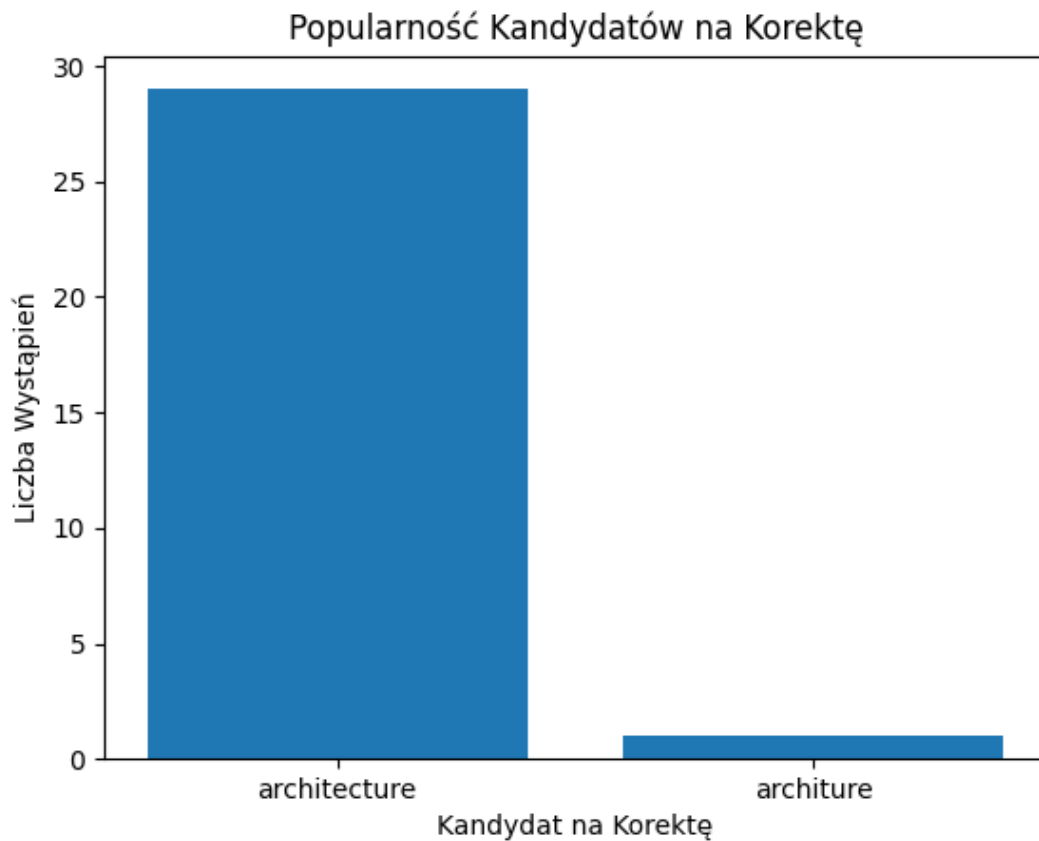
Analiza działania i statystyki

Jako input programu wykorzystamy przygotowany tekst zawierający 30 razy słowo *this* z celowymi błędami. Poniższy wykres przedstawia relację popularności danego kandydata na korektę.



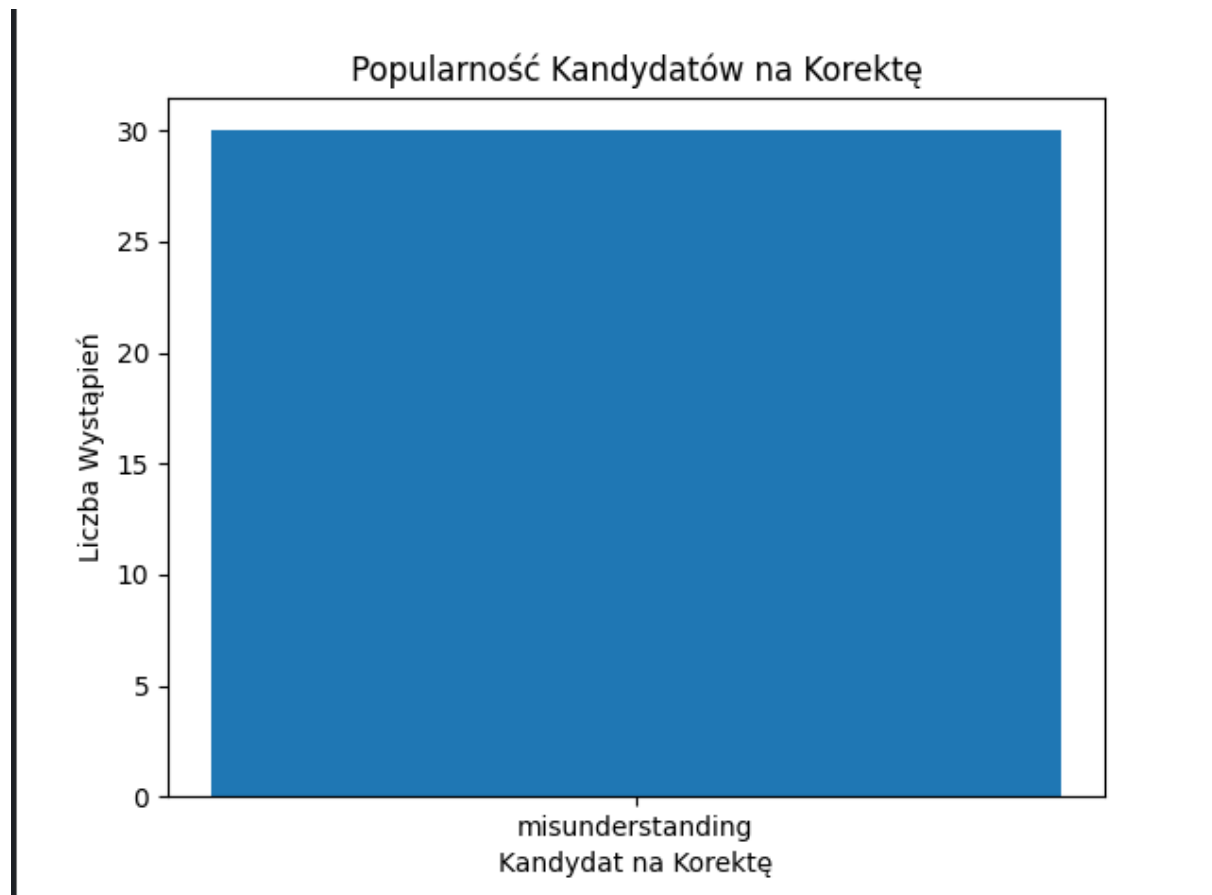
Analizując wykres możemy zauważyć, że w tym przypadku słowo *this* zostało poprawnie zaproponowane ponad 20 razy. Jednakże kilka zaproponowanych poprawek było innymi słowami niż oczekiwane. Wynika to z faktu, że program używając funkcji generujących kandydatów, może znaleźć innych pasujących kandydatów niekoniecznie odpowiadających oczekiwanemu słowu. W zależności od konkretnego przypadku program proponuje słowo o największym prawdopodobieństwie dopasowania. W przypadku krótkich słów program będzie miał więcej możliwości dopasowania innych słów stąd proponuje poprawny wyraz mniejszą ilość razy.

Jako kolejny przykład proponujemy słowo *architecture*. Przygotowane dane wejściowe zawierają tekst z 30 wariacjami słowa “architecture” z błędami i literówkami. Poniżej zaprezentowano wykres popularności kandydatów na korektę:



Jak możemy zobaczyć na wykresie poprawnie zaproponowano słowo *architecture* 29 razy. 1 raz zostało zaproponowane słowo “architure”. Takie zachowanie wynika z faktu, że w przypadku dłuższych słów prawdopodobieństwo wygenerowania “odpowiedniego” kandydata jest zdecydowanie większe niż w przypadku krótkich słów.

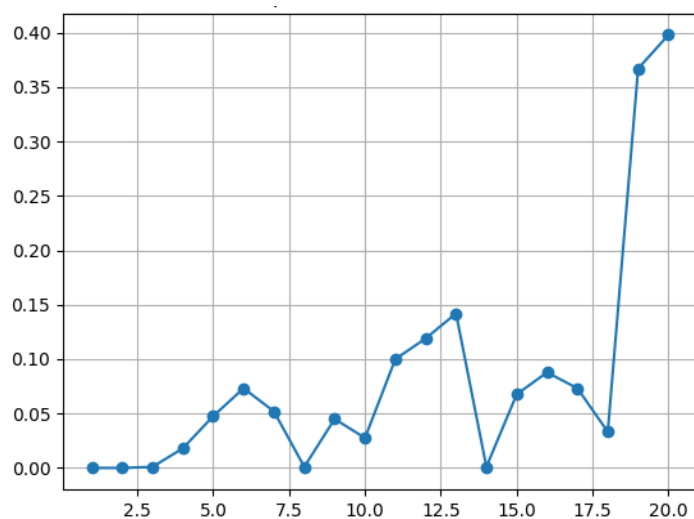
Jako ostatni przykład proponujemy tekst z 30 wariacjami błędnego słowa *misunderstanding*:



W tym przypadku poprawne słowo *misunderstanding* zostało zaproponowane za każdym razem. Wniosek wynikający z tych obserwacji jest taki, że im dłuższe słowo, tym większe prawdopodobieństwo dopasowania odpowiedniego słowa, co za tym idzie lepsza skuteczność spell correctora (o ile słowo jest dostępne w korpusie). W krótszym słowie w momencie popełnienia błędu np literówki możemy otrzymać inne słowo, które będzie równie poprawne w danym języku.

Pomiary czasu w zależności od długości słowa

Dla długości słowa od 1 do 20 liter zmierzone zostały czasy, w jakich spell corrector zwróci propozycję poprawionego słowa. Wykres czasów prezentuje się następująco (czas w sekundach):



Analizując powyższy wykres możemy zauważyć tendencję do wprost proporcjonalnej relacji długość słowa - czas na propozycję kandydata. Czas znacznie wzrasta dla szczególnie długich słów. Jednakże wahania na wykresie mogą wynikać z różnicy w popularności danego słowa w danym języku.

6. Podsumowanie

Projekt spell correctora może być z łatwością dostosowany do różnych języków oraz korpusów tekstu. Optymalizacje, takie jak uwzględnienie kontekstu zdania czy wykorzystanie bardziej zaawansowanych metod uczenia maszynowego, mogą poprawić skuteczność korekty, zwłaszcza w przypadku specyficznych kontekstów czy rzadkich słów.

7. Bibliografia

[1] <https://www.norvig.com/spell-correct.html>

[2] https://pl.wikipedia.org/wiki/Twierdzenie_Bayesa