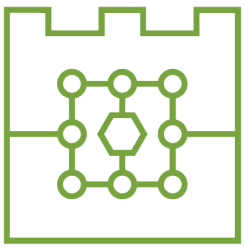

Politechnika Krakowska
Wydział Informatyki
i Telekomunikacji

Politechnika Krakowska
Wydział Informatyki i Telekomunikacji
Spell corrector

Piotr Smach
Gabriela Wielgus

14 maj 2023

1 Wstęp

Korektor ortograficzny to narzędzie używane do identyfikowania i poprawiania błędów ortograficznych w tekście. Mimo że większość nowoczesnych edytorów tekstu ma wbudowane korektory ortograficzne, zrozumienie, jak działają i jak je stworzyć, to ciekawe zadanie z zakresu przetwarzania języka naturalnego (NLP - Natural Language Processing).

Program, który stworzyliśmy, to prosty korektor ortograficzny dla języka polskiego i angielskiego. Używa on podstawowych technik NLP do identyfikowania błędów ortograficznych/literówek i sugerowania poprawek. Mimo że jest to prosta implementacja, dobrze ilustruje podstawowe koncepcje stojące za korektorem ortograficznym.

Nasz korektor ortograficzny działa na zasadzie modelu statystycznego, który ocenia prawdopodobieństwo poszczególnych słów w tekście na podstawie ich częstości występowania w korpusie - dużym zbiorze tekstów w danym języku. Jeśli dane słowo nie występuje w korpusie, program generuje listę kandydatów na poprawki, które są "bliskie"oryginalnemu słowu (tzn. różnią się od niego o jedną lub dwie "edycje", takie jak usunięcie litery, zamiana dwóch liter miejscami, zmiana jednej litery na inną, lub dodanie litery), a następnie wybiera te, które są najbardziej prawdopodobne na podstawie ich częstości występowania w korpusie.

2 Rozwinięcie

Podstawą naszego podejścia do rozwiązania problemu korekcji ortograficznej jest wykorzystanie modelu statystycznego opartego na częstości występowania słów w dużym korpusie tekstów. Wykorzystanie korpusu pozwala nam na ocenę, które słowa są najbardziej prawdopodobne w danym języku.

2.1 Przygotowanie danych

Na początku naszej pracy zaimplementowaliśmy funkcję wczytującą korpus z biblioteki NLTK. Dla języka angielskiego korzystaliśmy z korpusu Brown, natomiast dla języka polskiego skorzystaliśmy z

korpusu 'polish'. Wczytane słowa z korpusu przechowywane są w strukturze danych typu Counter, która pozwala na szybkie zliczanie częstości występowania słów.

2.2 Generowanie kandydatów

Następnie skupiliśmy się na generowaniu kandydatów na poprawki dla słów, które nie występują w korpusie. Wykorzystaliśmy do tego tzw. odległość edycyjną, która mierzy, jak bardzo dwa słowa różnią się od siebie poprzez liczbe operacji edycyjnych potrzebnych do przekształcenia jednego słowa w drugie. Nasz model generuje kandydatów, którzy różnią się od oryginalnego słowa o jedną lub dwie operacje edycyjne.

2.3 Wybór najlepszego kandydata

Po wygenerowaniu listy kandydatów, model ocenia ich prawdopodobieństwo na podstawie ich częstości występowania w korpusie i wybiera najbardziej prawdopodobne słowo jako poprawkę. Jeżeli dla danego słowa nie udało się wygenerować żadnych kandydatów, model zwraca oryginalne słowo bez zmian.

Nasze podejście jest proste, ale skuteczne dla wielu typów błędów ortograficznych. Jest jednak kilka potencjalnych obszarów do dalszego rozwoju i usprawnień, takich jak uwzględnienie kontekstu słowa w zdaniu czy zastosowanie zaawansowanych technik uczenia maszynowego do oceny prawdopodobieństwa kandydatów.

3 Implementacja detali

3.1 Funkcje klasy SpellCorrector

Szczegółowe wyjaśnienie funkcji zawartych w klasie SpellCorrector:

`P(self, word)`: Ta funkcja zwraca "prawdopodobieństwo"danego słowa. Jest to zdefiniowane jako częstotliwość występowania danego słowa w korpusie dzielona przez całkowitą liczbę słów. W praktyce, jest to prosta metoda estymacji prawdopodobieństwa danego słowa w języku.

`correction(self, word)`: Ta funkcja zwraca poprawione słowo. Jeżeli słowo jest znane (tj. występuje w korpusie), jest zwracane bez zmian. W przeciwnym razie, funkcja generuje listę możliwych kandydatów i zwraca słowo z najwyższym prawdopodobieństwem.

`candidates(self, word)`: Ta funkcja generuje listę potencjalnych kandydatów na poprawkę dla danego słowa. Najpierw sprawdza, czy dane słowo jest znane. Jeżeli tak, zwraca to słowo. W przeciwnym razie, generuje listę słów, które mogą być uzyskane przez wykonanie jednej lub dwóch operacji edycyjnych na oryginalnym słowie, i zwraca te z nich, które są znane.

`known(self, words)`: Ta funkcja sprawdza, które z podanych słów są znane, tj. występują w korpusie. Zwraca zbiór tych słów.

`edits1(self, word)`: Ta funkcja generuje listę słów, które mogą być uzyskane przez wykonanie jednej operacji edycyjnej na oryginalnym słowie. Operacje edycyjne to: usunięcie litery, zamiana dwóch sąsiednich liter, zamiana litery na inną oraz wstawienie litery.

`edits2(self, word)`: Ta funkcja generuje listę słów, które mogą być uzyskane przez wykonanie dwóch operacji edycyjnych na oryginalnym słowie. Robi to poprzez wygenerowanie wszystkich słów z edits1(word), a następnie dla każdego z nich generuje wszystkie słowa z edits1. Jest to znacznie szerszy zbiór kandydatów, ale również bardziej czasochłonny do generowania.

3.2 Kolejność wywoływania funkcji

1. Kiedy użytkownik wprowadza słowo do korekcji, główna funkcja correction jest wywoływana.
2. Wewnątrz funkcji correction, najpierw wywoływana jest funkcja candidates dla danego słowa.
3. Funkcja candidates najpierw próbuje znaleźć słowo w korpusie, wywołując funkcję known na samym słowie. Jeśli słowo jest znane, zostaje zwrócone.

4. Jeśli słowo nie jest znane, funkcja candidates wywołuje funkcję edits1 na słowie, aby wygenerować wszystkie możliwe słowa, które mogą być uzyskane przez jedną operację edycyjną. Potem wywołuje funkcję known na tych słowach, aby sprawdzić, które z nich są znane. Jeśli jakiekolwiek są znane, zwraca te słowa.

5. Jeżeli żadne z tych słów nie są znane, funkcja candidates wywołuje funkcję edits2 na słowie, aby wygenerować wszystkie możliwe słowa, które mogą być uzyskane przez dwie operacje edycyjne. Potem wywołuje funkcję known na tych słowach, aby sprawdzić, które z nich są znane. Jeśli jakiekolwiek są znane, zwraca te słowa.

6. Jeżeli żadne z tych słów nie są znane, funkcja candidates zwraca oryginalne słowo.

7. Następnie, funkcja correction wywołuje funkcję P na każdym ze słów zwróconych przez candidates, aby obliczyć ich prawdopodobieństwa. Wybiera słowo z najwyższym prawdopodobieństwem i zwraca je jako poprawione słowo.

3.3 Funkcje edycji

Nasza implementacja korzysta z czterech podstawowych operacji edycyjnych:

1. Usuwanie (deletes): usuwa każdą literę z oryginalnego słowa, tworząc nowe słowo.

2. Zamiana (transposes): zamienia miejscami każde dwie sąsiednie litery w oryginalnym słowie.

3. Zmiana (replaces): zmienia każdą literę w oryginalnym słowie na inną literę.

4. Wstawianie (inserts): wstawia każdą możliwą literę na każdą możliwą pozycję w oryginalnym słowie.

Te operacje są wykonywane na wszystkich słowach, które nie są obecne w korpusie, w celu generowania listy potencjalnych kandydatów na poprawki.

4 Obszary do dalszego rozwoju

Kontekstowe sprawdzanie pisowni: Opisany powyżej korektor pisowni analizuje tylko pojedyncze słowa bez uwzględniania ich kontekstu. Jednakże poprawna pisownia słowa często zależy od kontekstu. Na przykład słowo "read"mogłoby być błędem pisowni słowa "red"w zdaniu "I like the read car"(Lubię czerwony samochód), ale mogłoby być poprawne w zdaniu "I like to read books"(Lubię czytać książki). Kontekstowe sprawdzanie pisowni polega na użyciu kontekstu słowa (tj. słów wokół niego) do określenia najprawdopodobniej poprawnej pisowni. Można to zrobić za pomocą technik takich jak n-gramy, ukryte modele Markova, czy rekurencyjne sieci neuronowe.

Modele Deep Learning: Modele Deep Learning, w szczególności rekurencyjne sieci neuronowe (RNN), transformery (takie jak BERT, GPT) i ich warianty, odniosły duży sukces w wielu zadaniach NLP, w tym w korekcji pisowni. Modele te mogą uchwycić skomplikowane wzorce i zależności w danych, co czyni je bardziej dokładnymi niż prostsze metody. Mogą być one używane zarówno do sprawdzania pisowni bez kontekstu, jak i kontekstowego.

Modelowanie języka: Model języka to model, który może przewidzieć prawdopodobieństwo sekwencji słów. Może to być używane do oceny różnych możliwych poprawek słowa lub zdania na podstawie ich prawdopodobieństw. Modele języka mogą być tworzone za pomocą różnych technik, w tym n-gramów, RNN i transformerów.

Algorytmy fonetyczne: Algorytmy fonetyczne, takie jak Soundex, Metaphone, czy Double Metaphone, kodują słowa na podstawie ich fonetycznego podobieństwa. Mogą one być używane do sugerowania poprawek, które brzmią podobnie do błędnie napisanego słowa, co może być szczególnie przydatne przy poprawianiu literówek.

Modele sekwencja-do-sekwencji: Modele sekwencja-do-sekwencji (seq2seq) są typem modelu, który może przekształcić jedną sekwencję (np. zdanie z błędami pisowni) w inną sekwencję (np. to samo zdanie z poprawkami). Te modele, które zazwyczaj są implementowane jako RNN lub transformery, mogą być trenowane do poprawiania błędów pisowni w bardziej elastyczny i potężny sposób niż metody, które patrzą na każde słowo indywidualnie.

Model błędów: Model błędów to model typów błędów, które ludzie popełniają podczas pisania. W oparciu o model błędów można opracować algorytmy poprawiające tekst, które uwzględniają najczęstsze błędy popełniane przez użytkowników. W ten sposób narzędzia te mogą pomóc w poprawie jakości tekstu oraz zwiększeniu efektywności procesu pisania i edycji.

5 Podsumowanie

Nasza implementacja korektora ortograficznego oparta jest na statystycznym modelu języka, który ocenia prawdopodobieństwo poszczególnych słów na podstawie ich częstości występowania w korpusie. Nasze podejście jest skuteczne dla wielu typów błędów ortograficznych, ale ma też swoje ograniczenia.

Nie uwzględnia kontekstu słowa w zdaniu, co może prowadzić do niepoprawnej korekty w przypadku słów, które są ortograficznie poprawne, ale użyte są w niewłaściwym kontekście. Ponadto, nasza implementacja opiera się na relatywnie prostych technikach NLP i nie wykorzystuje zaawansowanych technik uczenia maszynowego, które mogą zwiększyć precyzję korekty.

Mimo tych ograniczeń, nasz korektor ortograficzny jest dobrym punktem wyjścia do dalszego rozwoju i eksploracji w dziedzinie przetwarzania języka naturalnego. Przez rozszerzenie naszego modelu o kontekstualne sprawdzanie pisowni lub zastosowanie zaawansowanych technik uczenia maszynowego, moglibyśmy stworzyć jeszcze bardziej precyzyjny i skuteczny korektor ortograficzny.