Gabriela Lewandowska
Paweł Prusałowicz

01.04.2019

Prowadzący: Łukasz Neumann

Dokumentacja wstępna

Heurystyczne naprawianie atrybutów tekstowych

Zaawansowane Programowanie w C++

Zadanie projektowe

Zadanie projektowe polega na stworzeniu biblioteki z interfejsem w języku Python, która pozwoli na czyszczenie danych rzeczywistych na podstawie słownika.

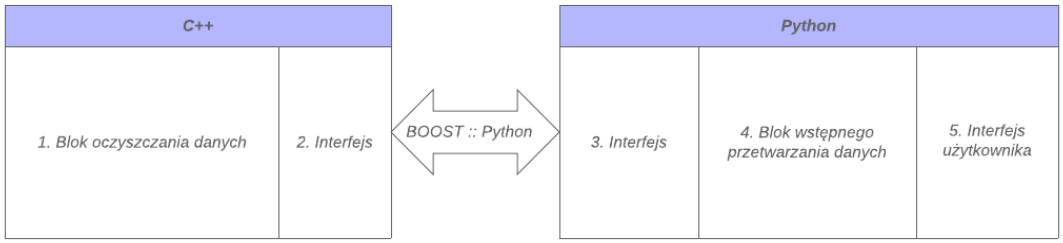
Opis funkcjonalności

Biblioteka dostarczy użytkownikowi narzędzia do implementacji we własnym programie funkcji do czyszczenia danych rzeczywistych. Otrzymane oczyszczone dane mogą potem zostać poddane przez niego dalszej analizie.

Podstawowymi funkcjami biblioteki są:

- możliwość załadowania własnego słownika, na podstawie którego może zostać naprawiona dowolna ilość zestawów danych;
- możliwość załadowania własnego zestawu danych w postaci jednowymiarowego numpy.array, pandas.Series lub Pythonowej listy, gdzie typem elementu jest string;
- wykonanie czyszczenia danych.

Struktura rozwiązania



1. Blok dokonujący właściwego czyszczenia danych według zadanego algorytmu i przechowujący załadowany słownik. Podczas ładowania słownika dokonywane jest także uporządkowanie znajdujących się w nim słów.
2. Interfejs do języka Python. Obejmuje funkcje zapewniające poprawną komunikację z kodem w języku C++. Zaimplementowane funkcje obejmą ładowanie i reset słownika, załadowanie danych i zwracanie informacji na temat stanu obiektu przetwarzającego.
3. Interfejs do języka C++.
4. Blok wstępnego przetwarzania danych.
5. Interfejs dla użytkownika. Dostarcza podstawowe funkcjonalności opisane w poprzedniej sekcji. Czyszczenie danych będzie możliwe tylko po uprzednim załadowaniu słownika.

Na algorytm czyszczenia danych rzeczywistych składają się:

- rzutowanie danych słownikowych na dane rzeczywiste; ta część algorytmu ma na celu znalezienie w słowniku pozycji o odległości Levenshteina mniejszej niż założona. Ze względu na mnogość dostępnych rozwiązań, ostateczna metoda zostanie wybrana w toku prac nad projektem, a kryteriami wyboru algorytmu będą: znalezienie poprawnego podstawienia, szybkość działania, prostota implementacji.

  Proponowane podejścia:
  1. Algorytm oparty na drzewie Trie
  Bazuje on na jednym z podstawowych algorytmów do obliczania odległości Levenshteina między dwoma słowami - algorytmu Wagnera-Fishera, w którym wyrazy porównywane są w tablicy liczb. Algorytm trie

wykorzystuje je, jednak zamiast tworzenia nowej tablicy dla każdego porównania, jedynie modyfikuje tablicę dla kolejnych, bliskich alfabetycznie wyrazów.

2. Algorytm Norviga
Polega na generowaniu wszystkich możliwych kombinacji wyrazu w zadanej odległości Levenshteina, a następnie wyszukiwaniu ich w słowniku.

3. Algorytm SymSpell
Polega na generowaniu możliwych kombinacji wyrazów w słowniku z użyciem tylko usuwania znaków, a następnie wyszukaniu wśród nich kombinacji (także tylko z użyciem usuwania) słowa wejściowego. Modyfikacja słownika następuje tylko raz, w czasie jego ładowania i może być wykorzystywana do czyszczenia danych wejściowych wielokrotnie.
Algorytm wykorzystuje symetrię odległości Levenshteina. Jeśli pierwszy wyraz jest w danej odległości od drugiego, to drugi jest w tej samej od pierwszego.

- punktowanie; determinuje, w jakiej kolejności ustawione są wyniki, które mógł mieć na myśli autor. Metoda zostanie wybrana spośród: korzystania z listy słów najpopularniejszych w zbiorze danych, wprowadzenia faworyzowania wybranych modyfikacji słów (na podstawie danych opensource).

Proponowane technologie

Biblioteka zaimplementowana zostanie w języku Python, jednak z uwagi na relatywnie dużą ilość danych, które mają być oczyszczone, silnik napisany zostanie w języku niskopoziomowym - C++.

Interfejs między tymi językami zostanie zapewniony przez zastosowanie biblioteki Boost.Python, która pozwala na wołanie funkcji zaimplementowanych w C++ w Pythonowych skryptach.

Do budowania rozwiązania zostanie wykorzystane narzędzie SCons.

Z biblioteki będzie można korzystać zarówno w systemie Linux, jak i Windows.

System kontroli wersji

W procesie tworzenia biblioteki używany będzie system kontroli wersji Github, gdzie znajdować się będzie aktualna wersja projektu.
link: https://github.com/PawelPrusalowicz/Heurystyczne-naprawianie-atrybutow-tekstowych