

Politechnika Krakowska

Quora Question Pairs

Projekt z przedmiotu Przetwarzanie Języka Naturalnego

Szymon Salamon

Adrian Szlag

Andrzej Sasak

2023

Spis treści

Wstęp.....	3
Cel projektu.....	3
Technologie.....	3
Opis działania algorytmu.....	4
Wizualizacja wyników.....	8
Podsumowanie.....	9

Wstęp

Cel projektu

Celem projektu było napisanie programu, który będzie potrafił stwierdzić, czy dwa pytania na platformie Quora są takie same. Quora to serwis, na którym użytkownicy mogą zadawać pytania i brać udział w dyskusjach. Program stanowiący przedmiot niniejszego projektu może być użytecznym narzędziem dla tego serwisu internetowego w celu wyfiltrowania duplikatów pytań i zwiększenia jakości zawartości na platformie.

Technologie

Ze względu na dostępność wielu bibliotek oraz narzędzi do przetwarzania tekstu oraz języka naturalnego zastosowano język Python. Język ten pozwala na łatwą integrację z bibliotekami do analizy i przetwarzania danych. W projekcie została użyta wersja Pythona 3.9.

Biblioteki użyte w kodzie projektu:

1. Pandas - biblioteka do analizy danych

Pandas to biblioteka, która dostarcza klasy i narzędzie do łatwej manipulacji danymi w formie tabelarycznej. Główny obiekt wykorzystany w projekcie to `DataFrame`, który umożliwia efektywną organizację i analizę danych w tabeli.

W projekcie Pandas zostało użyte do wczytywania danych z pliku csv oraz przygotowania danych do analizy.

2. Scikit-learn - biblioteka do uczenia maszynowego

Scikit-learn to biblioteka, która zawiera liczne narzędzia do uczenia maszynowego, która zawiera wiele przydatnych modułów w użytecznych w kontekście przetwarzania języka naturalnego.

Użyte w projekcie moduły to m.in.:

- `sklearn.feature_extraction.text`
- `sklearn.model_selection`
- `sklearn.metrics`

3. NLTK (Natural Language Toolkit) - biblioteka do przetwarzania języka naturalnego

NLTK jest biblioteką dostarczającą narzędzia do analizy tekstów w języku naturalnym. W projekcie z NLTK został użyty moduł `nltk.stem` oraz klasa `PorterStemmer` używana w celu normalizacji tekstu.

Opis działania algorytmu

Krok 1: Wczytywanie danych

Pierwszym krokiem jest wczytanie zbioru danych zawierającego pary pytań z Quora. Używamy do tego biblioteki `pandas`, która jest standardowym narzędziem do analizy danych w Pythonie. Kod do wczytania danych wygląda następująco:

```
import pandas as pd
df = pd.read_csv('train.csv', header=0)
```

Zestaw danych `train.csv` jest wczytywany do struktury `DataFrame` (`df`) która pozwala na łatwe manipulowanie i analizowanie danych. Parametr `header=0` wskazuje, że pierwszy wiersz pliku CSV zawiera nagłówki kolumn.

W celu zapewnienia spójności danych, usuwane są wiersze zawierające brakujące wartości. Dzieje się to poprzez wywołanie metody `dropna()` na zestawie danych. Parametr `axis=0` określa, że operacja usuwania będzie przeprowadzana wzdłuż wierszy.

```
df = df.dropna(axis=0)
```

Krok 2: Moduł Oczyszczania Tekstu

Oczyszczanie tekstu jest niezbędne w analizie semantycznej i przetwarzaniu języka naturalnego. Obejmuje ono kilka kluczowych kroków, zaimplementowanych w następujący sposób:

Usuwanie nadmiarowych spacji

Do usuwania nadmiarowych spacji wykorzystywane są standardowe metody Pythona, które pozwalają na skuteczne usunięcie zbędnych białych znaków z tekstu.

```
sentence = ' '.join(decontracted)
```

Usuwanie słów stopu

Słowa, takie jak "and", "or", "but", są usuwane, ponieważ nie wnoszą znaczącej wartości semantycznej. Usuwanie tych słów pozwala na skupienie się na bardziej istotnych elementach tekstu, które mają większe znaczenie dla analizy i procesu uczenia maszynowego. Zestaw stop-słów jest załadowany z modułu sklearn a następnie odfiltrowywanie.

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS as stop_words

sentence = [word for word in sentence.split() if not word in stop_words]
```

Zmiana skrótów na pełne formy

Zastosowanie mapowania skrótów na pełne formy słów jest realizowane za pomocą słownika skrótów i ich rozszerzeń.

```
contractions = {
    "ain't": "am not / are not / is not / has not / have not",
    "aren't": "are not / am not", ...}

text = ' '.join([contractions.get(word, word) for word in text.split()])
```

Usuwanie interpunkcji

Do usunięcia interpunkcji wykorzystywana jest metoda translate z Pythona, która pozwala na skuteczne usunięcie znaków interpunkcyjnych z tekstu.

```
sentence = sentence.translate(str.maketrans(' ', '', string.punctuation))
```

Stemming

Do stemmingu używana jest klasa PorterStemmer z modułu nltk. Pozwala ona na redukcję słów do ich formy podstawowej.

```
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()

sentence = ' '.join([stemmer.stem(w).strip("'") for w in sentence.split()])
```

Krok 3: Przekształcanie tekstu i przygotowanie cech

W trzecim etapie projektu, przetworzony tekst zostaje przekształcony w numeryczne wektory cech, co jest niezbędne do wykorzystania danych w modelach uczenia maszynowego. Proces ten realizowany jest poprzez następujące działania:

Importowanie niezbędnych bibliotek

Do przekształcania tekstu i przygotowania cech wykorzystywane są specjalne biblioteki z pakietu scikit-learn. Pozwalają one na efektywne wektoryzowanie tekstu oraz podział danych na zbiory treningowe i testowe.

```
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.model_selection import train_test_split
```

Wektoryzacja tekstu

Wektoryzacja to proces konwersji tekstu na numeryczne wektory cech. W projekcie wykorzystywany jest HashingVectorizer, który przekształca kolekcję dokumentów tekstowych na macierz wystąpień tokenów.

```
vectorizer = HashingVectorizer(n_features=2**10, alternate_sign=False)
q1_transformed = vectorizer.transform(df['question1'])
q2_transformed = vectorizer.transform(df['question2'])
```

Kombinacja cech

Po wektoryzacji, cechy obu zestawów pytań są łączone w celu utworzenia nowego zestawu cech, który będzie używany do trenowania modelu. W tym projekcie, cechy są kombinowane poprzez obliczanie różnicy elementów macierzy.

```
X = q1_transformed - q2_transformed
y = df['is_duplicate']
```

Operacja ta tworzy zestaw cech X, który reprezentuje różnicę wektorów cech obu pytań. Dodatkowo wyznaczamy zestaw etykiet y, który zawiera informacje o tym, czy pary pytań są duplikatami.

Podział danych na zbiory treningowe i testowe

Ostatnim krokiem w tym etapie jest podział danych na zbiory treningowe i testowe, co umożliwia późniejszy trening oraz walidację modelu.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=0.2, random_state=42)
```

Parametr `test_size=0.2` określa to, że 20% danych zostanie użytych jako zbiór testowy, a pozostałe 80% jako zbiór treningowy. `random_state` zapewnia powtarzalność podziału przy każdym uruchomieniu kodu.

Krok 4: Budowanie i trenowanie modelu XGBoost

W czwartym kroku projektu, skupiamy się na budowaniu i trenowaniu modelu klasyfikacyjnego przy użyciu biblioteki XGBoost.

Importowanie Biblioteki XGBoost

XGBoost (eXtreme Gradient Boosting) to implementacja modelu uczenia maszynowego, który jest szeroko stosowany w zadaniach klasyfikacyjnych ze względu na swoją wydajność i skuteczność.

```
import xgboost as xgb
```

Inicjalizacja modelu XGBoost

Model XGBoost jest inicjalizowany z domyślnymi parametrami.

```
model = xgb.XGBClassifier()
```

Trenowanie modelu

Model jest trenowany na zbiorze treningowym. Proces ten polega na dopasowaniu modelu do danych, co umożliwia mu naukę wzorców i zależności.

```
model.fit(X_train, y_train)
```

Tutaj `X_train` i `y_train` to odpowiednio cechy i etykiety ze zbioru treningowego, które zostały przygotowane w poprzednich krokach.

Krok 5: Ocena Modelu

Po wytrenowaniu modelu, jego wydajność jest oceniana na zbiorze testowym. Pozwala to na weryfikację, jak dobrze model radzi sobie na nowych, nieznanych danych.

```
y_pred = model.predict(X_test)
```

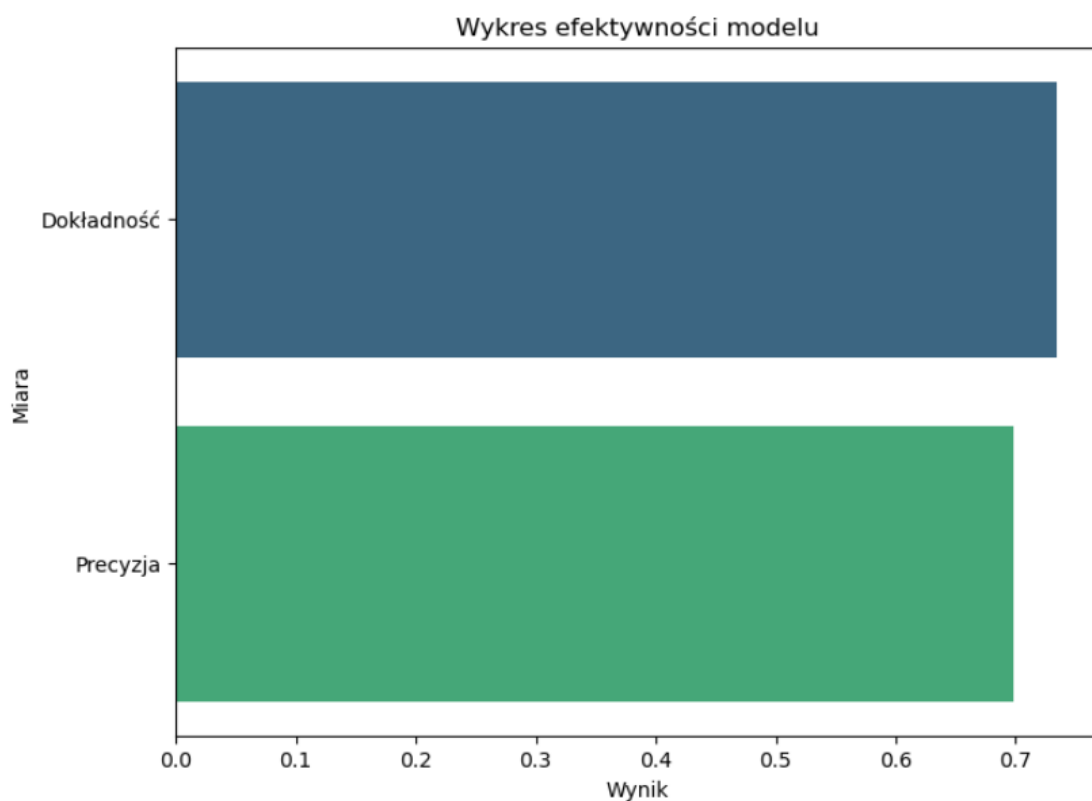
Dokładność modelu jest obliczana jako procent poprawnych przewidywań w stosunku do całkowitej liczby testów.

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
```

Wizualizacja wyników

Poniższe wykres oparty jest na wynikach w przypadku zastosowania podziału zestawu danych wejściowych na dane uczące i testowe w proporcji odpowiednio 80/20.



1. Dokładność modelu wyniosła 73%.

Dokładność to stosunek poprawnie sklasyfikowanych przypadków do wszystkich przypadków. Mierzy ogólną poprawność. Wynik na poziomie 73% jest akceptowalny w kontekście rozpoznawania duplikatów pytań

2. Precyzja modelu wyniosła 70%.

Precyzja mierzy stosunek poprawnie sklasyfikowanych pozytywnych przypadków do wszystkich przypadków, które model sklasyfikował jako pozytywne. W kontekście rozpoznawania duplikatów na platformie takiej jak Quora, precyzja modelu jest ważna, ponieważ nie powinno dopuszczać się do sytuacji, w której model błędnie oznacza pytania użytkowników które nie są duplikatami, jako duplikaty.

Podsumowanie

Model XGB osiągnął akceptowalną dokładność na poziomie 73%. Precyzja modelu natomiast w przypadku rozpoznawania duplikatów pytań na platformie Quora jest szczególnie ważną metryką, ponieważ oznaczanie pytań jako duplikaty w przypadku 30% pytań mogłoby być uciążliwe dla użytkowników platformy. W kontekście projektu można byłoby zastanowić się jak poprawić wyniki precyzji, poprzez manipulację hiperparametrami modelu XGB lub zastosowanie innych algorytmów klasyfikacji.