WTUM: Przewidywanie "śmieszności" żartu

Maciej Łukasik, Kajetan Larwa, Anton Yushkevich, Maksim Makaranka

May 2022

Spis treści

1 Wstęp

Wiele portali internetowych wykorzystuje do swojego działania opinie i oceny użytkowników. Analiza opinii jest szczególnie ważna w branżach: rozrywkowej oraz e-commerce, w których ma ona bezpośredni wpływ na odczucia z korzystania przez użytkowników. Dla przykładu, Netflix posiada silnik rekomendacji, który analizuje oceny użytkowników, w celu zapewnienia jak najlepszych rekomendacji filmów i seriali, tak aby były one spersonalizowane i jak najbardziej trafne dla użytkownika.

Dane zebrane od użytkowników można podzielić na dorozumiane i wyraźne.

Do danych dorozumianych zalicza się dane zebrane na podstawie pewnych interakcji użytkownika: obejrzenie określonego filmu, zamówienie pewnego towaru, czas spędzony na stronie internetowej. Zaletą tego typu danych jest ich liczność - w niedługim czasie można otrzymać sporo informacji o interakcjach użytkownika, jednak z drugiej strony takie dane wymagają głębszej analizy, na przykład fakt tego, że użytkownik przeczytał artykuł nie oznacza, że mu się podobał, jedynie może to oznaczać, że sam temat jest dla tego użytkownika interesujący.

Do danych wyraźnych zalicza się dane zebrane bezpośrednio od użytkownika - oceny, komentarze. Jednak ich ilość jest dużo mniejsza w porównaniu do danych dorozumianych (stosunkowo niewielka ilość użytkowników angażuje się w bezpośrednie ocenianie np. przeczytanych artykułów), z drugiej strony takie dane mają dużo większą wiarygodność, ponieważ otrzymujemy rzeczywistą informację zwrotną od samego użytkownika.

W ramach tego projektu skupimy się na ocenach żartów wystawionych przez użytkowników, które to zaliczamy do danych wyraźnych. Spróbujemy jak najtrafniej przewidzieć oceny użytkowników, którzy jeszcze nie ocenili danego dowcipu, mając do dyspozycji bazę dowcipów wraz z ocenami użytkowników.

2 Zestaw danych

Zestaw danych składa się z anonimowych ocen (od -10 do 10) wystawionych łącznie przez 41 000 użytkowników oraz 139 dowcipów. Zbiór treningowy składa się z 1,1 miliona ocen.

Dane przechowywane są w trzech plikach csv:

- Plik train.csv zawiera id użytkowników, id dowcipów oraz wystawione oceny

Variable	Description
id	Unique id
user_id	ID for user
joke_id	ID for joke
Rating	Rating given by the user to the joke

- Plik jokes.csv zawiera treści dowcipów

Variable	Description
joke_id	ID for joke
joke_text	Complete text for the joke

- Plik train.csv zawiera id użytkowników, id dowcipów oraz wystawione oceny

Variable	Description
id	Unique id
user_id	ID for user
joke_id	ID for joke

3 Opracowane metody i modele określające podobeństwo dowcipów

3.1 Pod-modele określające podobeństwo dwóch dowcipów

Skonstruowano następujące pod-modele mające na celu określenie podobieństwa między dwoma żartami:

- Jaccard similarity model
  W tym modelu do wyznaczenia podobieństwa żartów został wykorzystany indeks Jaccarda, który jest zdefiniowany jako iloraz mocy części wspólnej zbiorów i mocy sumy tych zbiorów.

- Cosine Similarity model
  W celu zdefiniowania tej metody ciągi są interpretowane jako wektory w przestrzeni unitarnej, a podobieństwo jest definiowane jako cosinus kąta między nimi, czyli iloraz iloczynu skalarnego wektorów oraz iloczynu ich długości.

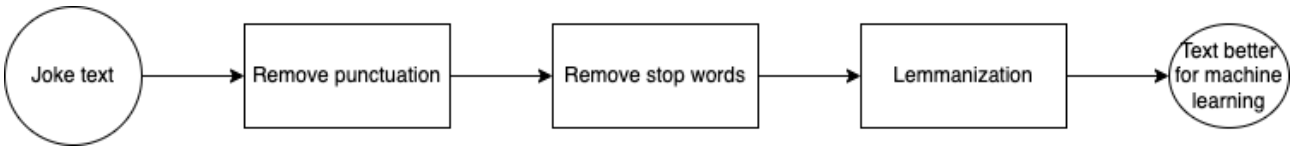
W metodzie Cosine Similarity konieczna była wcześniejsza wektoryzacja tekstu, do czego zostały użyte następujące metody:

- Term Frequency - Inverse Document Frequency
  TF-IDF to miara statystyczna używana do oceny ważności słowa w kontekście dokumentu. Waga słowa jest proporcjonalna do częstotliwości występowania tego słowa w dokumencie i odwrotnie proporcjonalna do częstotliwości występowania tego słowa we wszystkich dokumentach w kolekcji.

- SentenceBERT
  BERT to model językowy opracowany przez Google, który został wytrenowany za pomocą 3,3 mln angielskich słów. Jest to oparta na transformatorach technika uczenia maszynowego do przetwarzania języka naturalnego. Dużą zaletą tego modelu jest fakt, że BERT „rozumie" kontekst, w którym zostało użyte dane słowo.

Warto zaznaczyć, że w przypadku TF-IDF należy wcześniej odpowiednio przygotować przetwarzany tekst. Z kolei model BERT tego nie wymaga. Przetwarzanie zrobiono za pośrednictwem następującego "pipeline-u":

Rysunek 1: Wstępne przetwarzanie tesktu dla TTF-IDF



3.2 Modele wykorzystujące opracowane metryki

Zbudowano i przetestowano następujące modele:

3.2.1 Popularity based filtering

Model popularnościowy jest modelem podstawowym opartym na popularności i ocenie dowcipu przez użytkownika.

Pseudokod:

- Uczenie modeli:
  avg = {}
  Dla każdego żartu:
    avg[indeks żartu] = suma(oceny żartu) / liczba ocen żartu
- Przewidywanie oceny:
  Zwróć avg[indeks żartu]

Rysunek 2: Wyniki uzyskane dla modelu popularnościowego

```
Explained variance score: 0.09554334235659334
Mean absolute error: 4.04297249507049
Mean squared error: 24.82726319260841
Median absolute error: 3.5145840819086263
R2 coefficient: 0.09554313723334729
```

3.2.2 Collaborative filtering

Jest to metoda, która wykorzystuje znane preferencje grupy użytkowników do przewidywania nieznanych ocen innego użytkownika. Główne założenie tej metody jest następujące: użytkownicy, którzy w przeszłości oceniali dane dowcipy w ten sam sposób, będą mieli tendencję do wystawiania podobnych ocen innym dowcipom w przyszłości.
Pseudokod:

- Uczenie modeli:
  Pominięte z powodu złożoności $O(n^2)$, n - liczba wszystkich użytkownków
- Przewidywanie oceny:
  user - użytkwnik którego ocenę przewidujemy
  joke - żart do którego przewidujemy ocenę
  ocena = 0
  podobieństwa = [ ]
  Dla każdego użytkownika, który ocenił **joke** oraz przynajmniej jeden żart, który ocenił **user**:
    ocena = ocena + ocena innego użytkownika * podobieństwo oceniania żartów
    podobieństwa.Add(podobieństwo oceniania żartów)
  Zwróć $\frac{ocena}{sum(\text{podobieństwa})}$

Wykorzystując tą metodę otrymaliśmy średni błąd równy 3.774.

Rysunek 3: Wyniki uzyskane dla collaborative filtering

```
Explained variance score: 0.13461457401839272
Mean absolute error: 3.873382278716714
Mean squared error: 23.52309007901426
Median absolute error: 3.2861883327026877
R2 coefficient: 0.1308095527890245
```

3.2.3 Category based

Model ten został oparty na pomyśle, aby każdemu z żartów przydzielić kategorię i założeniu, że użytkownik będzie podobnie oceniać żarty z tej samej kategorii. Przydział kategori został zrealizowany przy pomocy spectral clusteringu. Niestety wpsomniane założenie okazało się błędne i model otrzymał bardzo słaby wynik.

Rysunek 4: Wyniki uzyskane dla modelu category based

```
Explained variance score: 0.013571324725132783
Mean absolute error: 4.239056488836024
Mean squared error: 26.96707686340645
Median absolute error: 3.7867023484674136
R2 coefficient: 0.013570339018883493
```

3.2.4 Content based

Filtrowanie oparte na treści wykorzystuje wcześniejsze oceny danego użytkownika w celu stwierdzenia w jakim stopniu dany żart, którego jeszcze nie oceniał, może mu się spodobać.

Pseudokod:

- Uczenie modeli:
  Dla każdej pary żartów oblicz podobieństwo pomiędzy nimi za pomocą jednej z w.w. metryki i zapisz do tablicy.

- Przewidywanie oceny:
  ocena = 0
  podobieństwa = [ ]
  Dla każdego innego żartu:
  ocena = ocena + ocena innego żartu * podobieństwo żartów
  podobieństwa.Add(podobieństwo żartów)
  Zwróć $\frac{ocena}{sum(\text{podobieństwa})}$

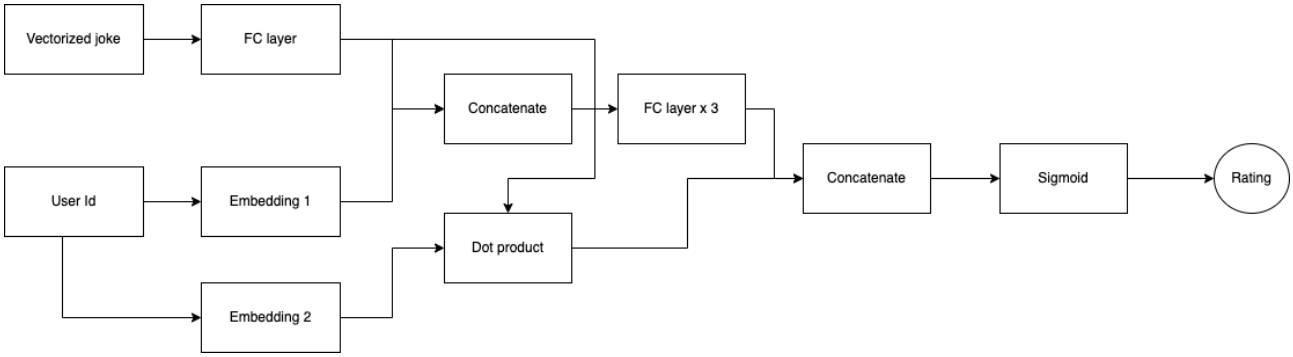
Rysunek 5: Wyniki uzyskane dla modelu content based

```
Explained variance score: 0.18371622140417865
Mean absolute error: 3.5097232397245515
Mean squared error: 22.409155959619465
Median absolute error: 2.5940000000000003
R2 coefficient: 0.17761894197234673
```

3.2.5 Neural Collaborative Filtering

Jest to model którego działanie w założeniach jest podobne do zwykłego collaborative filtering, jednkaże wykorzystuje on sieć neuronową. Architektura sieci jest oparta na artykule (https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401) i dodatkowo została nieco zmodyfikowana. Klasyczna wersja tego modelu nie korzysta z zawartości dowcipów, ale wejście to (w postaci zwektoryzowanego tesktu) zostało przez nas dodane w celu poprawy wydajności modelu. Architektura finalnej sieci wygląda następująco:

Rysunek 6: Architektura sieci neurownoej użytej w modelu NCF



Rysunek 7: Wyniki uzyskane dla modelu NCF

```
Explained variance score: 0.34378391865573044
Mean absolute error: 3.130479928668112
Mean squared error: 17.913342655281546
Median absolute error: 2.309259143829346
R2 coefficient: 0.3435478725907911
```

Model, którego wyniki tutaj przedstawiono, został nauczony przez 30 epok i przy pakowaniu danych po 256. Szczegóły implementacyjne i wartości inncyh parametrów znajdują się w repozytorium.

3.2.6 Podsumowanie

Model	Uzyskany średni błąd
Popularity based	4.04
Collaborative filtering	3.87
Category based	4.27
Content based	3.51
NCF	3.13

4 Wnioski

Jak możemy zobaczyć w statystykach dla poszczególnych modeli, mają one średni błąd około 3.5, co w ramach zadania nie jest najlepszym wynikiem. Zdecydowanie najlepsze rezultaty otrzymano dla Neural Collaborative Filtering, ale błąd na poziomie 3.13 wciąż nie jest zadowalający. Z wyników otrzymanych za pomocą tego modelu możemy jednakże wywnioskować, że stosowanie sieci neuronowych w silnikach systemów rekomendujących znacznie zwiększa ich wydajność.

Dużym rozczarowaniem okazał się model collaborative filtering, którego średni błąd jest nie dużo mniejszy niż dla modelu popularnościowego - podczas pracy traktowaliśmy ten model jako poziom bazowy.

Modele opierające się na zawartości żartów dostają na wejście tekst w postaci zwektoryzowanej - a więc wymagają przeprowadzenia wektoryzacji. Oba skonstruowane modele pełniące tą rolę dawały podobne rezultaty przy zastosowaniu z content based i NCF.

Niewątpliwie możliwe jest skontruowanie lepszych modeli dla rozpatrywanego problemu - jednkże wymagało by to głębszej wiedzy, więcej czasu i prób, a także w miarę możliwości więcej danych treningowych. Dużego ogrnaiczenia można dopatrywać się w niewielkim rozmiarze zbioru dowcipów - dla modeli wykorzystujących treść powiększenie puli mogłoby znacząco zwiększych dokładność.

5 Źródła

https://datahack.analyticsvidhya.com/contest/jester-practice-problem/
https://www.statisticshowto.com/jaccard-index/
https://www.sciencedirect.com/topics/computer-science/cosine-similarity
https://www.miquido.com/blog/perks-of-recommendation-systems-in-business/