Uniwersytet Warszawski

Wydział Nauk Ekonomicznych

Andrzej Żernaczuk 407240

Jakub Wujec 420463

# Czy możliwe jest przewidzenie ocen filmów na podstawie analizy sentymentu komentarzy na portalu Reddit

Praca zaliczeniowa na zajęcia

Zastosowania Języka Python oraz Data-Driven Research

prowadzone przez Kristóf Gyódi i Michała Palińskiego.

Warszawa, czerwiec 2021 r.

**SPIS TREŚCI**

## 1. Wstęp

W niniejszej pracy podjęliśmy się zbadania możliwości przewidzenia ocen filmów poprzez analizę sentymentu komentarzy na portalu *Reddit* (dalej zwanym także Redditem). W tym celu posłużyliśmy się bazą danych z portalu *Kaggle* oraz postami z *Twittera* tak, aby wytrenować nasz model w ocenie treści komentarza.

## 2. Dane

Do wytrenowania modelu zostały użyte dane zawierające 1 600 000 tweetów wraz z oznaczeniem sentymentu. Ze względu na złożoność obliczeniową i zasoby potrzebne do trenowania modeli, użyliśmy losowo wybranej połowy zbioru danych.

Do stworzenia bazy filmów których ocenę chcieliśmy przewidzieć posłużyliśmy się API o nazwie *omdbapi*. Wybraliśmy pierwsze 50 filmów (z wyjątkiem następujących produkcji: *Lot nad kukułczym gniazdem*, *Życie jest piękne*, *Dobry, zły i brzydki,* pominiętych z powodu problemu z ich tytułami) z listy top500 portalu *Filmweb*. Stworzony plik JSON wyczyściliśmy z niepotrzebnych nam danych, a następnie skonwertowaliśmy do formatu CSV. Plik zawiera: tytuł, ID z portalu *IMDb*, rok, liczbę ocen oraz, co najważniejsze, oceny produkcji z 3 najważniejszych portali poświęconych ocenianiu filmów: *IMDb*, *Metacritic* oraz *RottenTomatoes*.

## 3. Metodologia

### 3.1. Przygotowanie danych tekstowych do użycia w modelu

Zarówno do uczenia modelu jak i późniejszych predykcji potrzebne jest przetworzenie danych tekstowych do formatu nadającego się dla sieci neuronowej. Pierwszą czynnością jaka została w tym celu wykonana było usunięcie nazw własnych użytkowników oznaczonych w twitterowych postach. Następnie usuwaliśmy znaki takie jak “\n” lub “\r” które służa do do formatowania tekstu oraz wszelkiego rodzaju znaki specjalne. Tak wstępnie przetworzone dane poddawane są procesowi usunięcia *stopwords.* Polega ona na usunięciu słów takich jak ‘i’, ‘me’, ‘you’ i tym podobnych, czyli słów które nie wnoszą nic do treści zdania, a co za tym idzie nie są potrzebne do trenowania modelu. Do tego procesu została użyta biblioteka *nltk*. Następnie następuje proces tokenizacji, do którego użyta została biblioteka *Tensorflow.* Proces ten polega na podziale zdań na pojedyńcze słowa, a następnie przyporządkowaniu każdemu z nich unikalnej cyfry. Dzięki temu otrzymujemy zbiór wszystkich unikalnych słów występujących w naszych danych wraz z przyporządkowaną do

nich unikalna liczbą. Za przykład użyjemy zdanie 'Analiza sentymentu z wykorzystaniem sieci neuronowych'. Jest ona zamieniana w następującą listę:

```
['Analiza', 'sentymentu', 'z', 'wykorzystaniem', 'sieci', 'neuronowych']
```

Następnie każde słowo zamieniane jest na wcześniej przyporządkowaną mu liczbę:

```
[311, 123, 3, 1, 2525, 1311]
```

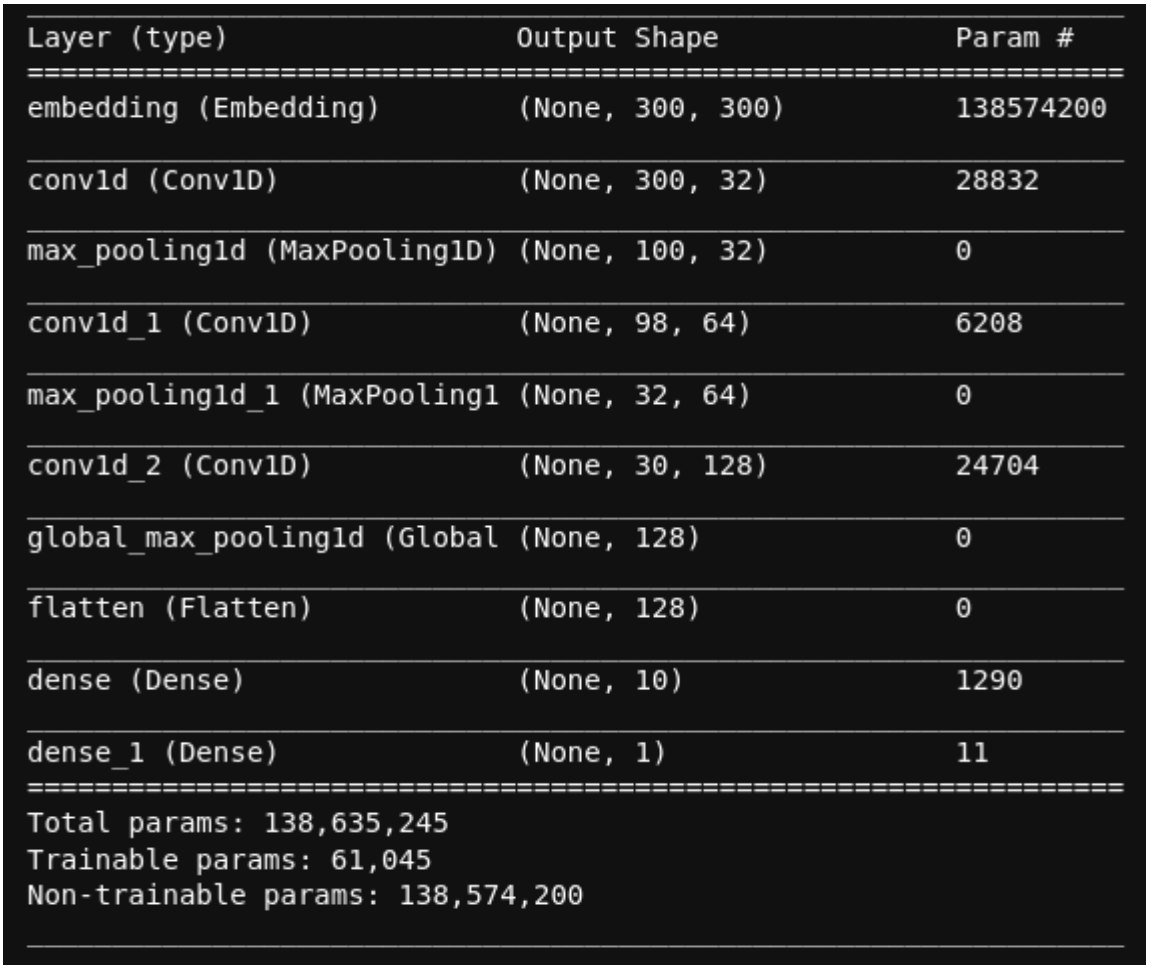
Tak przygotowane dane poddajemy procesowi *Paddingu.* Jest to procedura konieczna ze względu na różne długości tekstów na których będziemy pracować. Dzieje się tak, ponieważ model musi mieć stały kształt danych wejściowych. *Padding* polega na normalizacji długości każdej listy ze ztokenizowanymi danymi. Przykładowo, jeżeli chcemy żeby długość naszych danych wejściowych wynosiła 10, to biorąc pod uwagę, że nasze zdanie miało tylko 6 słów, w puste pola zostaną wpisane cyfry 0:

```
[0, 0, 0, 0, 311, 123, 3, 1, 2525, 1311]
```

Jeżeli przykładowy tekst byłby dłuższy od wybranej przez nas długości danych wejściowych, to zdanie zostanie skrócone.

### 3.2. Przygotowanie modeli.

W naszym badaniu testowaliśmy dwa typy sieci neuronowych. Pierwszą z nich jest CNN czyli *Convolutional Neural Network*. Jest to sieć neuronowa zawierająca warstwy konwolucyjne. W dużym skrócie, konwolucja jest przekształceniem macierzowym, fragmentu zdjęcia, tekstu lub innych danych, mające na celu wyciągnięcie informacji o konkretnych jego cechach. Po warstwie konwolucyjnej używa się tzw. *poolingu*, służącego do zmniejszenia wymiarów cech konwolucyjnych, które zostały wyznaczone w poprzedniej warstwie, zachowując najważniejsze cechy. Drugim rodzajem sieci jest RNN czyli *Recurrent neural network*. Jest to rodzaj sieci umożliwiający wykorzystanie danych sekwencyjnych. Zawierają one tak zwane *komórki pamięci* umożliwiające sieci zachowywanie informacji z poprzednich kroków czasowych. Obydwa te modele mają jednak część wspólną - pierwszą ich warstwą jest *embedding*. Do jej stworzenia został użyty algorytm *word2vec* z wykorzystaniem biblioteki *gensim*. Proces ten polega na zamianie każdej cyfry reprezentującej ztokenizowane słowo na wektor o wymiarach przez nas wybranych. Wektory dobrane są tak, że słowa o podobnych znaczeniu znajdują się w przestrzeni blisko siebie. Model CNN wygląda w następujący sposób:



| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding (Embedding) | (None, 300, 300) | 138574200 |
| conv1d (Conv1D) | (None, 300, 32) | 28832 |
| max_pooling1d (MaxPooling1D) | (None, 100, 32) | 0 |
| conv1d_1 (Conv1D) | (None, 98, 64) | 6208 |
| max_pooling1d_1 (MaxPooling1 | (None, 32, 64) | 0 |
| conv1d_2 (Conv1D) | (None, 30, 128) | 24704 |
| global_max_pooling1d (Global | (None, 128) | 0 |
| flatten (Flatten) | (None, 128) | 0 |
| dense (Dense) | (None, 10) | 1290 |
| dense_1 (Dense) | (None, 1) | 11 |

Total params: 138,635,245
Trainable params: 61,045
Non-trainable params: 138,574,200

Składa się on z trzech warstw konwolucyjnych, po każdej z nich następuje warstwa *Max Poolingu.* Wyjątkiem jest tutaj ostatnia warstwa, która zamiast klasycznego *Max Poolingu* używa *Global Max Poolingu*. W modelu tym udało się osiągnąć około 0.73 accuracy.

Model RNN wygląda w następujący sposób:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 300)          138574200
_________________________________________________________________
dropout (Dropout)            (None, 300, 300)          0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               160400
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 138,734,701
Trainable params: 160,501
Non-trainable params: 138,574,200
_________________________________________________________________
```

Użyliśmy specjalnego wariantu sieci RNN, czyli LSTM, z angielskiego *Long short-term memory.* W celu walki z *overfittingiem*, czyli zjawiskiem nadmiernego dopasowania do danych treningowych zastosowany został *dropout.* W obydwu modelach funkcją służącą do optymalizacji jest *Adam*, a funkcją straty *binary cross-entropy*, wyrażona następującym wzorem:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

gdzie:

N - rozmiar danych wyjściowych

W tym modelu udało się osiągnąć accuracy na poziomie 0.789

W związku z zauważalnie lepszymi wynikami dla modelu LSTM to on zostanie wykorzystany w dalszej pracy

### 3.3. Pobieranie komentarzy z portalu Reddit.

Do pobierania komentarzy z portalu *Reddit* używamy biblioteki *psaw*, zawierającej API do wcześniej wspomnianego portalu. Następnie pobieramy 1000 komentarzy i dla każdego z nich przeprowadzamy analizę sentymentu i liczymy średnia ważoną gdzie wagami są liczby punktów przy danym komentarzu.

### 3.4. Przygotowanie danych do porównania predykcji z właściwymi wartościami

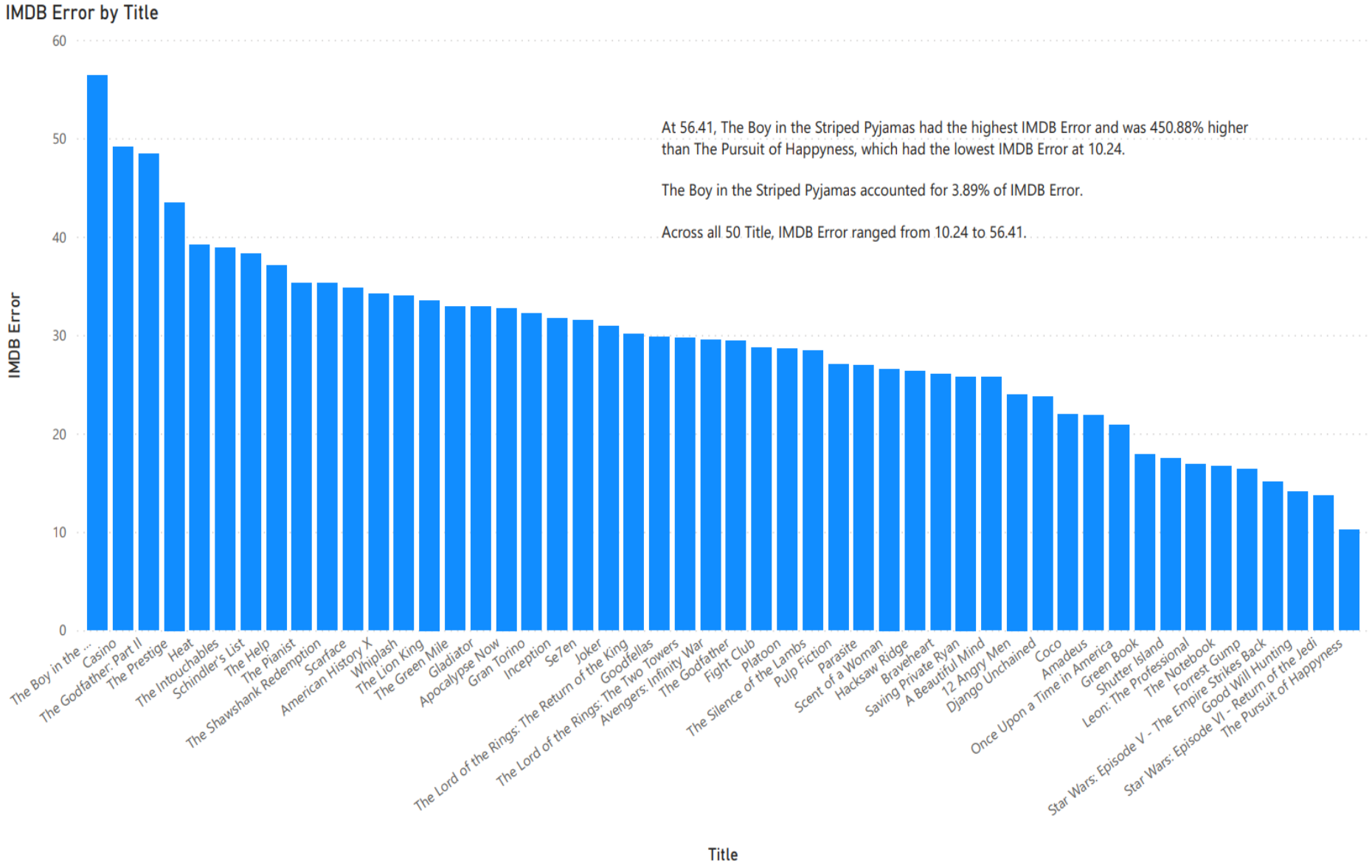
Do obróbki i analizy używamy Data Frame'ów czyli dwuwymiarowych rodzajów przechowywania danych na wzór tablicy, o zmiennym rozmiarze i potencjale do przechowywania różnorodnych typów danych. Pierwszy z Data Frame'ów tworzony jest z pliku *Filmweb_top50.csv* w którym zawarte są dane dotyczące pięćdziesięciu pierwszych filmów z rankingu portalu *Filmweb*. Następnie konwertujemy oceny to spójnego formatu, w przypadku IMDb z X.Y/10 na XY, RottenTomatoes z XY% na XY i dla portalu Metacritic z XY/100 na XY. Te ujednolicenie formatowania pozwala nam porównać wartość przewidzianą z wartościami właściwymi. Różnicę między ocenami obliczaliśmy za pomocą prostego odejmowania między predykcją a oceną na danym portalu, której wynik zaokrąglaliśmy do dwóch miejsc po przecinku. Dla każdego z filmów zapisywana jest do DataFrame o nazwie *data_df*: przewidywana ocena oraz wielkość różnic wraz z pozostałą resztą danych. Na potrzeby zabezpieczenia się przed utratą danych w przypadku błędu lub awarii sprzętu co 5 filmów dane zapisywane były do pliku *top_50_preds.csv*. Po wykonaniu się całego kodu wszystkie wyniki znajdują się w wyżej wymienionym pliku.

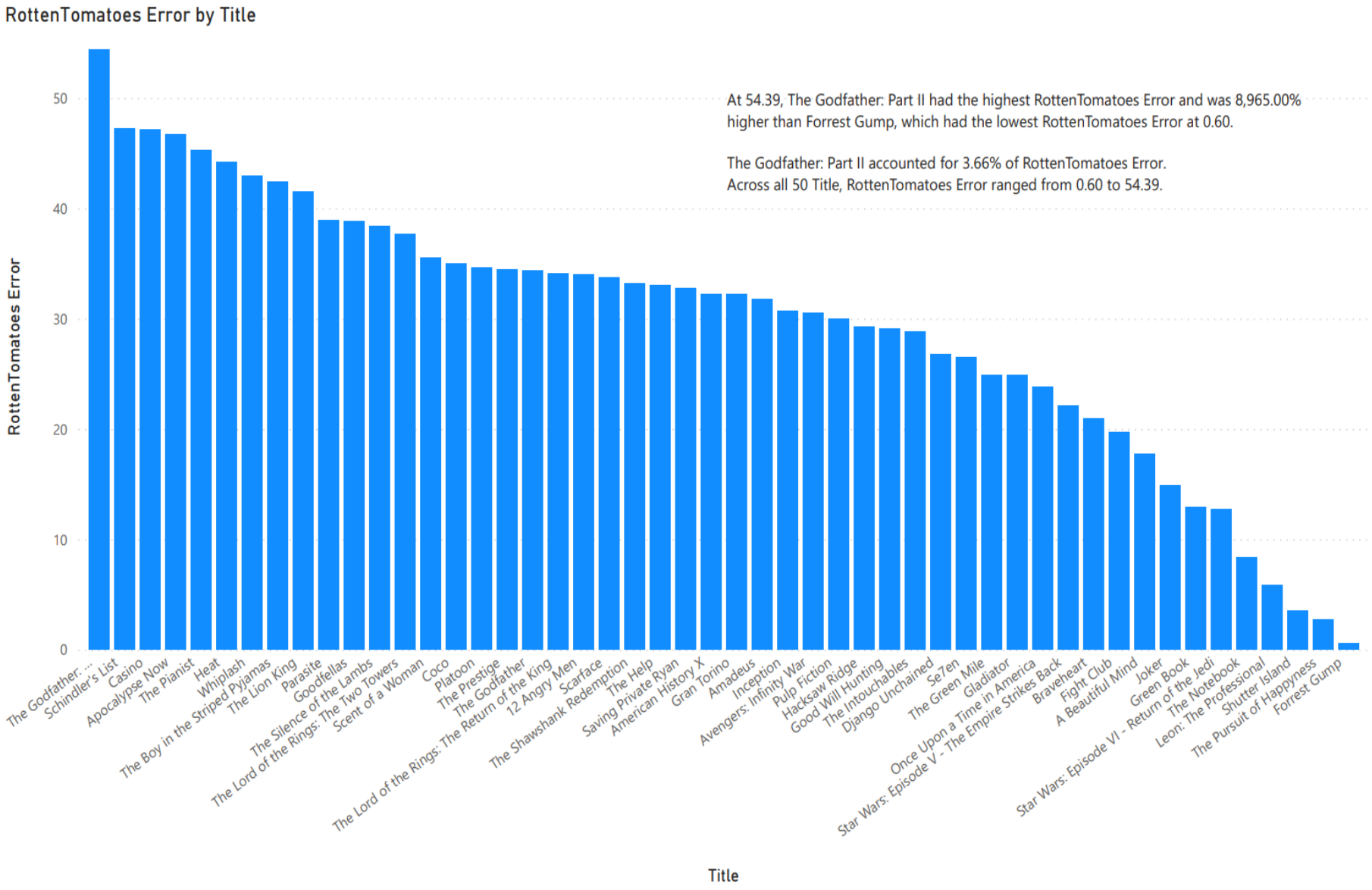
## 4. Wyniki

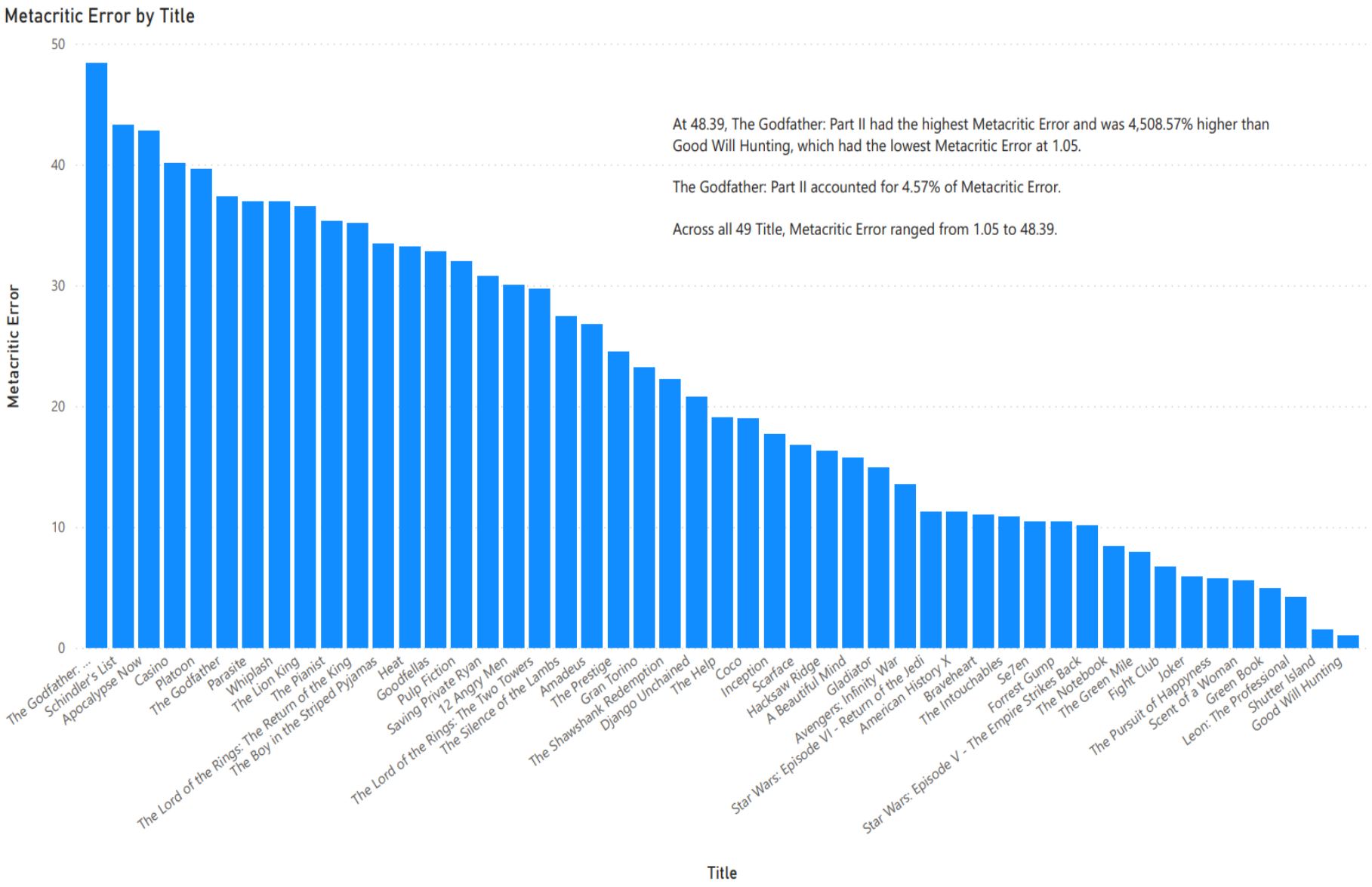
Nasze wyniki sprawdzaliśmy na wcześniej wspomnianym zbiorze. Średni błąd dla każdego z nich wyglądał następująco: *IMDb*: 29,03, *Metacritic*: 29,01, *RottenTomatoes*: 20,5.

```
Błąd predykcji w przypadku ocen z IMDB wynosi:  29.03
Błąd predykcji w przypadku ocen z RottenTomatoes wynosi:  20.5
Błąd predykcji w przypadku ocen z Metacritic wynosi:  29.01
```

Nasz model osiągnął najlepszy rezultat dla portalu *RottenTomatoes*, w przypadku pozostałych dwóch portali średni wynik był praktycznie taki sam, jednak różnica w zakresie wielkości błędów stoi na korzyść *IMDb*. Średnia niedokładność wśród trzech portali wyniosi 26,18 punktów procentowych. Na następnych trzech stronach zaprezentowane zostały wykresy pokazujące różnice pomiędzy naszą predykcją, a faktycznymi ocenami na poprzednio wspomnianych portalach, wyrażone w punktach procentowych. Zostały one wykonane za pomoca rozwiązania *PowerBI* dostarczonego przez firmę *Microsoft*.


IMDB Error by Title
60
50
40
30
20
10
0
IMDB Error
At 56.41, The Boy in the Striped Pyjamas had the highest IMDB Error and was 450.88% higher than The Pursuit of Happyness, which had the lowest IMDB Error at 10.24.
The Boy in the Striped Pyjamas accounted for 3.89% of IMDB Error.
Across all 50 Title, IMDB Error ranged from 10.24 to 56.41.
The Boy in the ...
Casino
The Godfather: Part II
The Prestige
Heat
The Intouchables
Schindler's List
The Help
The Pianist
The Shawshank Redemption
Scarface
American History X
Whiplash
The Lion King
The Green Mile
Gladiator
Apocalypse Now
Gran Torino
Inception
Se7en
Joker
The Lord of the Rings: The Return of the King
Goodfellas
The Lord of the Rings: The Two Towers
Avengers: Infinity War
The Godfather
Fight Club
Platoon
The Silence of the Lambs
Pulp Fiction
Parasite
Scent of a Woman
Hacksaw Ridge
Braveheart
Saving Private Ryan
A Beautiful Mind
12 Angry Men
Django Unchained
Coco
Amadeus
Once Upon a Time in America
Green Book
Shutter Island
Leon: The Professional
The Notebook
Forrest Gump
Star Wars: Episode V - The Empire Strikes Back
Good Will Hunting
Star Wars: Episode VI - Return of the Jedi
The Pursuit of Happyness
Title


RottenTomatoes Error by Title
At 54.39, The Godfather: Part II had the highest RottenTomatoes Error and was 8,965.00% higher than Forrest Gump, which had the lowest RottenTomatoes Error at 0.60.
The Godfather: Part II accounted for 3.66% of RottenTomatoes Error.
Across all 50 Title, RottenTomatoes Error ranged from 0.60 to 54.39.
RottenTomatoes Error
0
10
20
30
40
50
The Godfather: ...
Schindler's List
Casino
Apocalypse Now
The Pianist
Heat
Whiplash
The Boy in the Striped Pyjamas
The Lion King
Parasite
Goodfellas
The Silence of the Lambs
The Lord of the Rings: The Two Towers
Scent of a Woman
Coco
Platoon
The Prestige
The Godfather
The Lord of the Rings: The Return of the King
12 Angry Men
Scarface
The Shawshank Redemption
The Help
Saving Private Ryan
American History X
Gran Torino
Amadeus
Inception
Avengers: Infinity War
Pulp Fiction
Hacksaw Ridge
Good Will Hunting
The Intouchables
Django Unchained
Se7en
The Green Mile
Gladiator
Once Upon a Time in America
Star Wars: Episode V - The Empire Strikes Back
Braveheart
Fight Club
A Beautiful Mind
Joker
Green Book
Star Wars: Episode VI - Return of the Jedi
The Notebook
Leon: The Professional
Shutter Island
The Pursuit of Happyness
Forrest Gump
Title


Metacritic Error by Title
Metacritic Error
50
40
30
20
10
0
At 48.39, The Godfather: Part II had the highest Metacritic Error and was 4,508.57% higher than Good Will Hunting, which had the lowest Metacritic Error at 1.05.
The Godfather: Part II accounted for 4.57% of Metacritic Error.
Across all 49 Title, Metacritic Error ranged from 1.05 to 48.39.
The Godfather: ...
Schindler's List
Apocalypse Now
Casino
Platoon
The Godfather
Parasite
Whiplash
The Lion King
The Pianist
The Lord of the Rings: The Return of the King
The Boy in the Striped Pyjamas
Heat
Goodfellas
Pulp Fiction
Saving Private Ryan
12 Angry Men
The Lord of the Rings: The Two Towers
The Silence of the Lambs
Amadeus
The Prestige
Gran Torino
The Shawshank Redemption
Django Unchained
The Help
Coco
Inception
Scarface
Hacksaw Ridge
A Beautiful Mind
Gladiator
Avengers: Infinity War
Star Wars: Episode VI - Return of the Jedi
American History X
Braveheart
The Intouchables
Se7en
Forrest Gump
Star Wars: Episode V - The Empire Strikes Back
The Notebook
The Green Mile
Fight Club
Joker
The Pursuit of Happyness
Scent of a Woman
Green Book
Leon: The Professional
Shutter Island
Good Will Hunting
Title

## 5. Podsumowanie

W niniejszej pracy podjęliśmy się zbadania możliwości przewidywania ocen filmów na podstawie komentarzy użytkowników *Reddita*. Nasze wyniki wskazują że jesteśmy w stanie to zrobić z średnim błędem mieszczącym się w zakresie 20-30 punktów procentowych. Nie jest to jednak satysfakcjonująca dokładność. Możliwym wytłumaczeniem tego jest inny charakter portalu *Reddit* od typowego przedstawiciela Social Mediów. Analizując komentarze z subreddita poświęconego danemu tematowi możemy spodziewać się przeważnie pozytywnych ocen, w końcu ta część forum będzie tworzona przez ludzi pozytywnie nastawionych do niego. Subreddity często opisywane są jako bańki informacyjne, gdzie pewny zbiór poglądów może być uznawany za "jedyny słuszny". W części *Reddita* poświęconej kinematografii wypowiadają się ludzie, którzy są zainteresowani kinem, przy czym na portalach poświęconym stricte filmom, częściej zabierają głos także "niedzielni" kinomaniacy. W naszej ocenie te powody mogą wpływać na osiągnięty przez nas wynik.

## 6. Bibliografia

https://www.reddit.com/
http://www.omdbapi.com/
https://www.filmweb.pl/ranking/film
https://www.imdb.com/
https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/
https://www.kaggle.com/paoloripamonti/twitter-sentiment-analysis
https://towardsdatascience.com/cnn-sentiment-analysis-9b1771e7cdd6
https://www.tensorflow.org/text/tutorials/text_classification_rnn
https://www.researchgate.net/figure/The-architecture-of-CNN-based-text-sentiment-classification_fig2_328431365