PREDICTING MOVIE SUCCESS FROM SCRIPT ANALYSIS USING NLTK TOOLKIT

Antwerp

Diana, Tije, Andrew, Adedoyin

Table of Contents

[INTRODUCTION 2](#_Toc535507617)

[BACKGROUND REVIEW 3](#_Toc535507618)

[PROJECT SPECIFICATION 4](#_Toc535507619)

[SOLUTION DESIGN/METHODOLOGY 4](#_Toc535507620)

[RESULTS 5](#_Toc535507621)

[PREDICTIONS 6](#_Toc535507622)

[CONCLUSIONS 7](#_Toc535507623)

[FURTHER APPLICATIONS 7](#_Toc535507624)

[REFERENCES 7](#_Toc535507625)

# INTRODUCTION

Every movie enthusiast has favourite movies. There are certain criterium that makes movies likable. This criterium is often based on individual bias. Movies are often rated as “outstanding” or “boring” depending on personal criterium. At sources like IMDB, Movie enthusiasts rate movies with scores ranging from one to ten. These scores vary. For example, popular movies like “The Shawshank Redemption” receive a huge number of votes between 9 and 10 while poor movies have ratings between 1 and 3. From this voting we can conclude that there are some general conditions that determine how successful a movie will do. These conditions may be independent of subjective human bias

In the 21st century where data is considered the sexiest job in the century, data analysis could prove useful in understanding why some movies are received better than others. In movie ratings, movie enthusiasts often talk about camera angles, acting performances, and image quality. However, these qualities are not enough. In addition, it is difficult to find patterns in the features mentioned above. For instance, experienced actors could argue that acting makes a film great, while photographers find the image and video quality more appealing. In order to concretize movie quality, the analysis of movie scripts could prove useful. Writing and storytelling are important parts of a movie and are often mentioned in user reviews.

Script analysis could provide us with useful information. Patterns and features that could help and explain the difference between a good and a bad movie. Programming languages like Python allow us to analyse data sets quickly, with our current technologies. Looking for patterns in bad/good movies is something that hasn’t been done, or at least documented on the internet. This reason, and our love for movies, made us decide to pick this topic for our group project. Features like lexical diversity are useful considerations and can be calculated with Python. We chose the topic from the perspective of movie enthusiasts, as well as starting Python programmers.

# BACKGROUND REVIEW

Hunter III, et al., 2016 used Network Text analysis – a way of mapping interconnected contexts of words to predict how well a film does in the box office. This involved an intricate network which is complex.

Dursun and Ramesh, 2009 used an information fused approach. They employed artificial neural networks, decision tress and vector machines to predict the financial performance of a movie based on box office receipts. This does not necessarily take into consideration script analysis.

Jehoshua et al., 2014, also developed a methodology to predict box office performance. They analysed scripts focusing on textual features such as genre and content, semantics and bag of words. Production budget was also considered in their analysis.

# PROJECT SPECIFICATION

We follow in the foot-steps of Jehoshua et al., 2014. However, we took a different approach. We introduced new features such as lexical diversity, speech tagging, the average length per sentence and the amount of ratings. Our project also focused on only script analysis in relation to the ratings of audience because there are movies with huge budgets that performs low in the box

We used the NLTK toolkit in Python, as well as the website of the Internet Movie Script Database (IMSDB) as tools for our research. The IMSDB website (<https://www.imsdb.com>) has an enormous amount of movie scripts ready to be downloaded. In order to get a general idea about the movies in the database, we calculated the Lexical Diversity from the entire database of scripts. In total, 1092 scripts were scanned, and a lexical diversity score was calculated. This was done with the stop-words function, the word\_tokenize function and the Porterstemmer function. The words in every script were tokenized as individual words. All words were “stemmed” to their base words. For example; walking becomes walk, sleeping becomes sleep e.t.c. The number of individual stems were then divided by the amount of unique words to generate a diversity score. The higher this score, the more diverse the vocabulary of the movie script.

Next up, we scanned every word and added a tag to it. The abbreviations of the tags can be looked up on this website: [link](http://cl.indiana.edu/~obscrivn/docs/POStagsVegan.pdf). A few examples of frequent occurring tags are: Nouns, Verbs, Adverbs and Adjectives. Every word was tagged for each of the scripts. With these tags, a percentage was calculated which showed the composition of each of the scripts. This way we could find out if a script was noun-heavy or verb-heavy for example.

Our third feature consisted of the average length of each sentence. Longer sentences tend to be more complex than simple ones, and this could be reflected into the user ratings. This is why we also included this feature. The last feature was the amount of votes each movie received.

# SOLUTION DESIGN/METHODOLOGY

Methodology and library used: Initial set of movie scripts was downloaded from https://figshare.com/projects/imsdb\_movie\_scripts/18907 which were fetched from imsdb.com (internet movie script database) as of January 2017. The files were unzipped to give 1092 movie scripts as plain .txt files. Containing movie titles file names were used as identifiers throughout the project.

The major libraries used in the project were:

**NLTK** - text processing and analysis. Was used to generate features for the regression models;

**IMDBPY** - a library specialized for data retrieval from imdb.com. It was used to get movie ratings served as our model response and one additional feature, the number of ratings left under each movie;

**SCIKIT-learn** and **Numpy** were used for machine learning: they were used to build the regression models and perform predictions.

**Time and datetime** - used to measure the time spent processing several functions

**MAIN.py**

This was used to combine all the work into one user friendly package, from there the user has access to all the combiner, textProcessing and ml functions (called with combineIt textProcessing and ml respectively). The main loop runs until the user quits (q or quit), the operations can be performed independently or all in one go, depending on the user's request.

**COMBINER.py**

Using imdbpy, movies were found by their titles at imdb.com and their imdb identifiers were determined. Based on these ids and using another imdb database as a tsv file, ratings and number of votes were found and added to the dataset file.

**TEXTEDIT.py**

First, some initial work with scripts was performed: lexical diversity was calculated. With nltk the movie scripts were split into sentences and then into words. Stemming of words was performed and then the ratio of original stems to the total number of words was taken as lexical text diversity.

This script also carries out the analysis of parts of speech + average sentence length. Both were performed using nltk. Frequencies of each parts of speech occurrences were reported as part of the total word count. All new features were added to the dataset.

**ML.py**

First, files were converted into a final dataset by removing unwanted elements. It was further divided into testing and training models. Linear regression models were used to train and test the models.

# RESULTS

38 features were extracted for 1000 movie scripts: 32 frequencies of speech parts, lexical diversity, average sentence length and number of votes on imdb.com. The rating of the movie was taken as a response. The dataset was split either in halves or randomly in various ratios into the training and the testing sets. Various regression algorithms were tested in order to find the model and use it for predictions.

The linear regression model is characterized by an adjusted R-squared value of 0.392 showing relatively low correlation between the features and the response. The most significant features in this model were determined: number of votes on the imdb.com, lexical diversity, sentence length and parts of speech.

**TYPICAL OUTPUT FOR A RANDOMIZED 50/50 DATASET**

TYPICAL OUTPUT FOR A RANDOMIZED 50/50 DATASET

Mean squared error: 0.73

Variance score: 0.23

OLS Regression Results

==============================================================================

Dep. Variable: y R-squared: 0.445

Model: OLS Adj. R-squared: 0.392

Method: Least Squares F-statistic: 8.338

Date: Sun, 13 Jan 2019 Prob (F-statistic): 1.22e-31

Time: 17:59:28 Log-Likelihood: -472.38

No. Observations: 434 AIC: 1023.

Df Residuals: 395 BIC: 1182.

Df Model: 38

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const 2.3063 1.843 1.251 0.212 -1.318 5.930

x1 2.19e-06 1.6e-07 13.711 0.000 1.88e-06 2.5e-06

x2 -3.6616 2.105 -1.740 0.083 -7.799 0.476

x3 28.5565 23.140 1.234 0.218 -16.936 74.049

x4 48.8130 24.922 1.959 0.051 -0.184 97.810

x5 45.7409 41.848 1.093 0.275 -36.532 128.014

x6 26.9791 61.544 0.438 0.661 -94.016 147.975

x7 -17.6602 10.587 -1.668 0.096 -38.473 3.153

x8 -7.6554 9.712 -0.788 0.431 -26.749 11.438

x9 -0.0879 8.802 -0.010 0.992 -17.392 17.217

x10 -26.6006 16.467 -1.615 0.107 -58.974 5.773

x11 24.0727 6.032 3.991 0.000 12.214 35.931

x12 -16.0172 7.119 -2.250 0.025 -30.013 -2.022

x13 23.3205 9.390 2.483 0.013 4.859 41.782

x14 810.6948 880.824 0.920 0.358 -920.995 2542.384

x15 199.5895 111.860 1.784 0.075 -20.326 419.505

x16 -27.2954 14.452 -1.889 0.060 -55.707 1.116

x17 7.3603 87.542 0.084 0.933 -164.746 179.466

x18 7.1944 10.166 0.708 0.480 -12.793 27.181

x19 -28.2152 182.857 -0.154 0.877 -387.709 331.279

x20 188.6613 454.525 0.415 0.678 -704.929 1082.252

x21 -4.8249 16.746 -0.288 0.773 -37.747 28.098

x25 -61.3896 106.323 -0.577 0.564 -270.419 147.640

x26 -15.5728 150.616 -0.103 0.918 -311.682 280.536

x27 204.6259 194.262 1.053 0.293 -177.291 586.542

x28 -18.0820 22.256 -0.812 0.417 -61.836 25.672

x29 6.6691 21.029 0.317 0.751 -34.673 48.011

x30 33.0199 62.270 0.530 0.596 -89.402 155.442

x31 5.3331 2.795 1.908 0.057 -0.162 10.828

x32 96.7838 85.821 1.128 0.260 -71.940 265.507

x33 5.0469 10.150 0.497 0.619 -14.908 25.002

x34 -589.6085 280.644 -2.101 0.036 -1141.351 -37.866

x35 14.5846 9.949 1.466 0.143 -4.975 34.144

x36 6.4065 5.455 1.174 0.241 -4.318 17.131

x37 14.4948 18.907 0.767 0.444 -22.677 51.666

x38 0.0370 0.019 1.906 0.057 -0.001 0.075

==============================================================================

Omnibus: 62.573 Durbin-Watson: 2.133

Prob(Omnibus): 0.000 Jarque-Bera (JB): 128.379

Skew: -0.791 Prob(JB): 1.33e-28

Kurtosis: 5.143 Cond. No. 7.47e+09

==============================================================================

# PREDICTIONS

R-squared value for predictions varied significantly depending on the regression model and training/test set ratio. Naturally, the highest levels close to the above R2 of the model itself were achieved for small test sets (15% and less), up to 0.38-0.39. Bayesian ridge regression was always the best performing model. Of the data pre-processing methods, Dimensionality reduction (PCA) was the most successful. Coupled with simple linear regression, it delivered on average much better results than the full-dimensional modelling. This can be attributed to high number of features, many of the marginal or non-significant, that add some noise to the model.

# CONCLUSIONS

The carried-out analysis showed that textual content of a movie (script analysis) cannot completely account for success among the audience as we hoped for. Nevertheless, some correlations were observed, despite its moderate values.

Besides features like actors' and director' talents, production budget etcetera, possible reasons such as dialogs and scenery descriptions were not separated during script analysis; relatively low response variance (scripts are available only for the most significant movies which have ratings 5-9 out of 10). Nevertheless, movies have a unique advantage of popularity criteria, the ratings from imdb.com, being easily extractable. This sharply contrasts with other purely textual artworks, such as novels, where much more correlation between success and the text analysis could be expected, but the popularity criterium is much harder to mine.

Finally, other features like Actor’s input, Budget could be added to make the code more robust.

# FURTHER APPLICATIONS

If the code becomes more robust in the future, film makers could use it to assess their script before producing a movie. This would save a lot of money and help manage the budget. It could also lead to better movies, and thus more revenue. Not only could it prove beneficial in the movie industry, but also in the book industry for example. The code could compare different texts in order to find a relation between variables that are given. The range of variables that could apply is very broad.

# REFERENCES

Hunter III, David S; Smith, Susan; Singh, Saba, 2016. Predicting box office from the screen play: A text analytical approach. Journal of Screenwriting, Volume 7, Number 2. 1 June 2016, pp. 135-154(20).

Dursun, Delen and Ramesh, Sharde, 2009. Predicting the financial success of Hollywood movies using an information fusion approach.

Jehoshua, Eliashberg; Sam K. Hui; Z. John Zhang, 2014. Assessing Box office performance using movie scripts: A kernel-based approach.