Predicting Box Office Revenue

Using Twitter

DECEMBER 2018

COMPILED BY:

SOURADEEPTA BISWAS

VISHNU K MENON

CIS 563 INTRODUCTION TO DATA SCIENCE

INSTRUCTOR:

REZA ZAFARANI

Table of Contents

1. Introduction

Twitter has been used for sharing contents and comments on all types of subjects by millions of people on a daily basis. It is clear that businesses have a strong interest in tapping into these huge data sources to extract information that might improve their decision making process. For example, predictive models derived from social media for successful movies may facilitate filmmakers making more profitable decisions[1].
In this work, we try to predict the movie popularity from
sentiment analysis of Twitter data talking about movies. We analysis both the tweets in 2009 and recent tweets in 2012. We manually label tweets to create a training set, and train a classifier to classify the tweets into: positive, negative, neutral, and irrelevant. We further develop a metric to capture the relationship between sentiment analysis and the box office results of movies. Finally we predict the Box Office results by classifying the movie as three categories: Hit, Flop, and Average. Our project also includes investigation on related topics like the relationship between tweet sent time and tweet number[2].

2. Background

By streaming tweets relevant to a movie of our choice straight from twitter we can get an idea for how many people are talking about the movie and what they're saying. By monitoring total twitter engagement and the overall sentiment about the movie we can predict movie performance.[3]. When a studio releases a movie, they don't just put it out there and hope people happen to go see it. They are actively engaged trying to convince the public to go show a movie through various forms of marketing. By tracking twitter engagement and predicting how well a movie will fare over the coming week gives insight into how well people are being reached and how effective current marketing campaigns.

3. Method

3.1 Extracting data from twitter

The first step is to gather the data from Twitter. By giving Twitter a list of keywords we can receive a live stream of all tweets that match those keywords. Before our system can analyze the tweet, it needs to be modified so that we can focus in on the most important features of the tweet. The words are stemmed, punctuation is removed, and we only keep adverbs, adjectives, and verbs because they have been most valuable in determining sentiment.

3.2 Performing Sentiment Analysis

Now that the tweet is ready, it is passed on to a series of machine learning classification algorithms that will each analyze and tell us whether it believes that tweet is speaking positively or negatively. Due to the complexity of spoken languages, especially with words having multiple meanings, these classifiers are not perfectly accurate. In order to improve our results, we use several of them, and each one votes on what it thinks the correct answer is. We then take whichever answer is most popular as the correct answer. That result is then stored for later use.

3.2 Training dataset

Gathering the other part of the data, the box office numbers, is much simpler. Thanks to sites like Box Office Mojo, the box office numbers are readily available. All we need to do is point our program at a webpage for each movie and read the box office data and store it.

3.3 Prediction

Because box office numbers tend to follow a weekly trend of being low at the beginning of the week and high on the weekends, we group all of our data into weeks to reduce the effect that has on our tests. Then for each week we calculate key features. The features we currently gather are: whether the total twitter engagement went up or down since last week, the percentage difference of engagement between the two weeks, the overall sentiment this week (positive, negative, or neutral), and how much did the sentiment change from last week to this week.

Those features for each week are passed through a classifier called a decision tree. The decision tree was built on the features of past weeks that we had gathered. The tree picks out patterns in the features and uses that to make a prediction. In this case our prediction comes in the form of the percentage change we should see from this week's box office numbers to the next week's numbers. Decision trees are limited on the number of results they can return, so in this case the percentage is a value rounded to the nearest 10%.

4. Implementation

4.1 Using the sentiment Corpora

We have used the movie review polarity Corpora from Cornell[4] which already has pre-classified tweets to process them before feeding into the classifiers. For this we first tokenizer the reviews find the frequency distribution.
word_features, which contains the top 3,000 most common words. Next, we're going to build a quick function that will find these top 3,000 words in our positive and negative documents, marking their presence as either positive or negative.
fairly popular text classification task is to identify a body of text as either spam or not spam, for things like email filters. In our case, we're going to try to create a sentiment analysis algorithm.

To do this, we're going to start by trying to use the movie reviews database that is part of the NLTK corpus. From there we'll try to use words as "features" which are a part of either a positive or negative movie review. The NLTK corpus movie_reviews data set has the reviews, and they are labeled already as positive or negative. This means we can train and test with this data. First, let's wrangle our data. Basically, in plain English, the above code is translated to: In each category (we have pos or neg), take all of the file IDs (each review has its own ID), then store the word_tokenized version (a list of words) for the file ID, followed by the positive or negative label in one big list.

Next, we use random to shuffle our documents. This is because we're going to be training and testing. If we left them in order, chances are we'd train on all of the negatives, some positives, and then test only against positives. We don't want that, so we shuffle the data.

4.2 Training Naive Bayes classifier

Now it is time to choose an algorithm, separate our data into training and testing sets, and press go! The algorithm that we're going to use first is the Naive Bayes classifier. This is a pretty popular algorithm used in text classification, so it is only fitting that we try it out first. Before we can train and test our algorithm, however, we need to go ahead and split up the data into a training set and a testing set.

You could train and test on the same dataset, but this would present you with some serious bias issues, so you should never train and test against the exact same data. To do this, since we've shuffled our data set, we'll assign the first 1,900 shuffled reviews, consisting of both positive and negative reviews, as the training set. Then, we can test against the last 100 to see how accurate we are.

4.3 Pickling data for speed

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script. This saves us time and helps us not run the training models again and again[6].

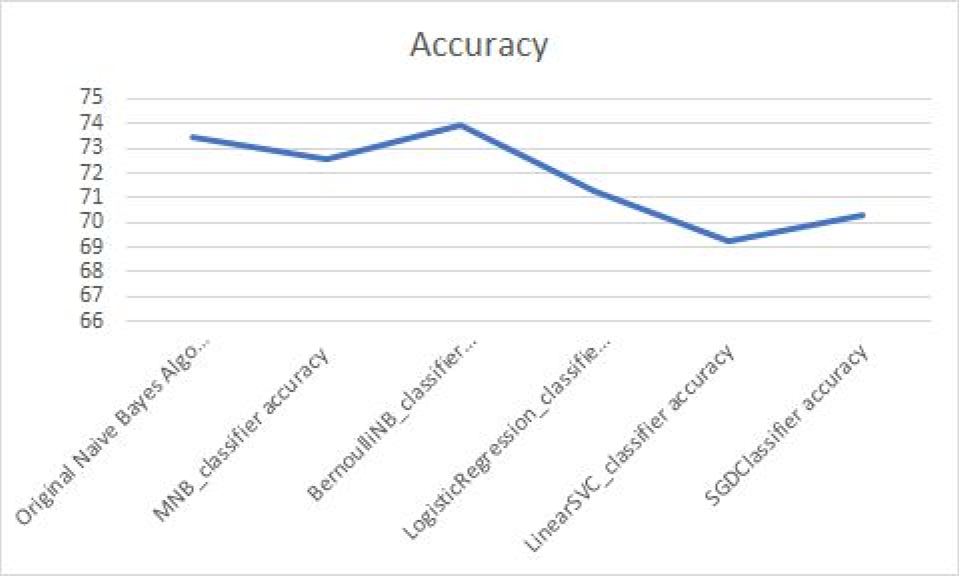
So after we have trained our model then, we use pickle.dump() to dump the data. The first parameter to pickle.dump() is what are you dumping, the second parameter is where are you dumping it.

After that, we close the file as we're supposed to, and that is that, we now have a pickled, or serialized, object saved in our script's directory![5]

4.4 Training Additional Classification models

Now to make sure we get the best accuracy percentage we have decided to run a few more models and create a Invoking the fit method on the VotingClassifier which will fit clones of those original estimators that will be stored in a class attribute[7].n estimator can be set to None using set_params. The output which we see from running our multiple models is below:

```
Original Naive Bayes Algo accuracy percent: 73.49397590361446
MNB_classifier accuracy percent: 72.59036144578313
BernoulliNB_classifier accuracy percent: 73.94578313253012
LogisticRegression_classifier accuracy percent: 71.23493975903614
LinearSVC_classifier accuracy percent: 69.27710843373494
SGDClassifier accuracy percent: 70.33132530120481
```



4.5 Getting tweets

We get tweets from twitter using twitterstream function of tweepy to gather tweets and then run our pre trained models on those tweets to process them and assign negative and positive sentiment values along with confidence on them. This data we write into a MySQL database. This lets us the ability of resuming the timeout that we faced from twitter disconnecting us repeatedly after 10-30 minute intervals and also to decide how to create the dataset as in movies to be selected. Indie movie had very low mentions and most of the sentiment on twitter output is negative because of cant and because of using expletives.

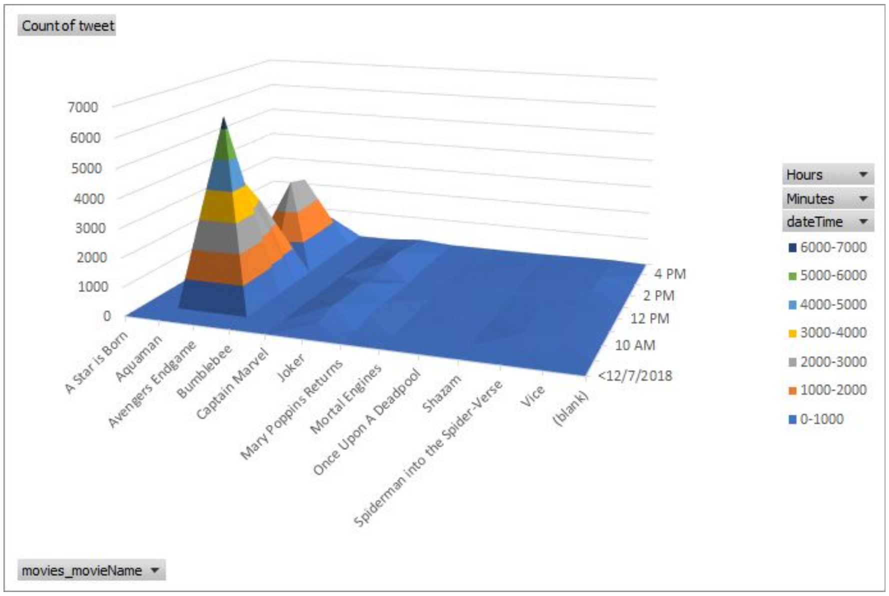
4.6 Train vs Test data set

For our training dataset we selected movies released a few months back and we had pulled the box office revenue for those movies from BoxOfficeMojo website. We gathered these tweets and performed sentiment analysis on them and added further features such as IMDB starmeter value for the highest values for 2 persons (which were not the actor and the actress as we had assumed. Also Black Panther making a ton of money had stars with really low ratings which made us realise that this was not a very accurate feature for our prediction) and widest release in theaters. We performed linear regression on our model and trained the dataset. And then we compared the values of box office revenue with the values we had pulled from the site. But our error was very very high!

4.7 Model fit

Eventually we ran the trained model through our dataset to perdict the box office revenue of movies which were not released(Note: we have created this list of movies based on coming soon movies from IMDB) and we faced huge errors because of various factors which would be discussed in the results.

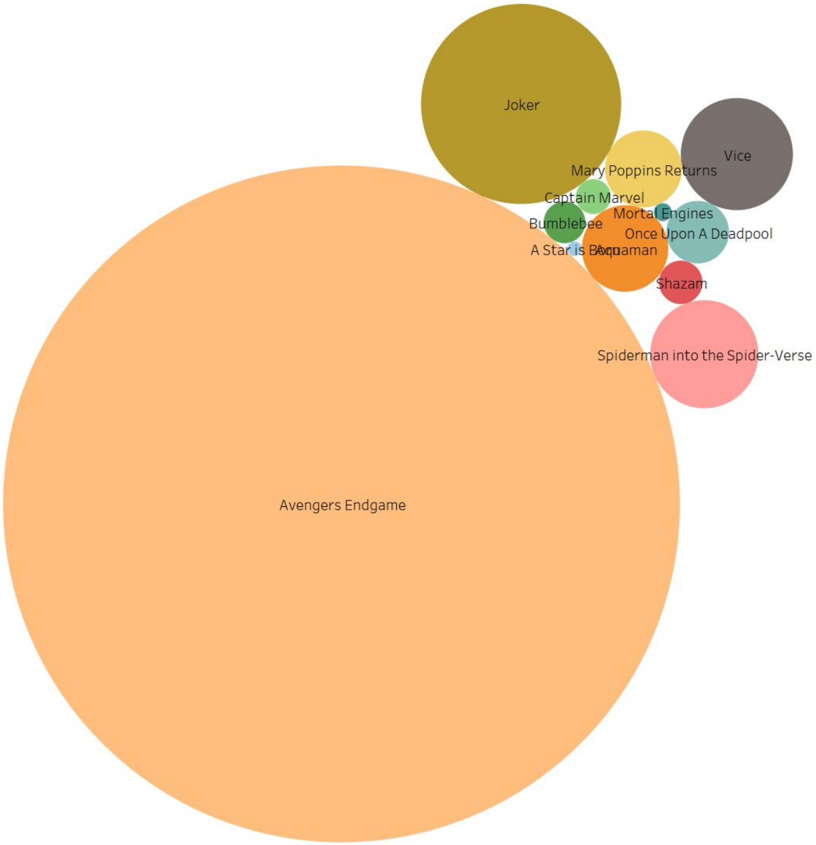
We see below the amount of tweets pulled only on the last day compared to the other movies.



5. Results

5.1 Avengers Endgame bias

Our results are skewed because we had assumed that the TwitterStream from tweepy library would allow us to pull data from current time till past for our movie tweet collection. But unfortunately it pulls data in from the current time and since the release of Avengers Endgame trailer we were facing a huge bias on the amount of data being pulled in. Below is the graph which shows the amount of tweets pulled for the upcoming movies. This bias and the inability to get past tweets had lead to a very skewed dataset.



5.2 Seasonal pull of twitter data

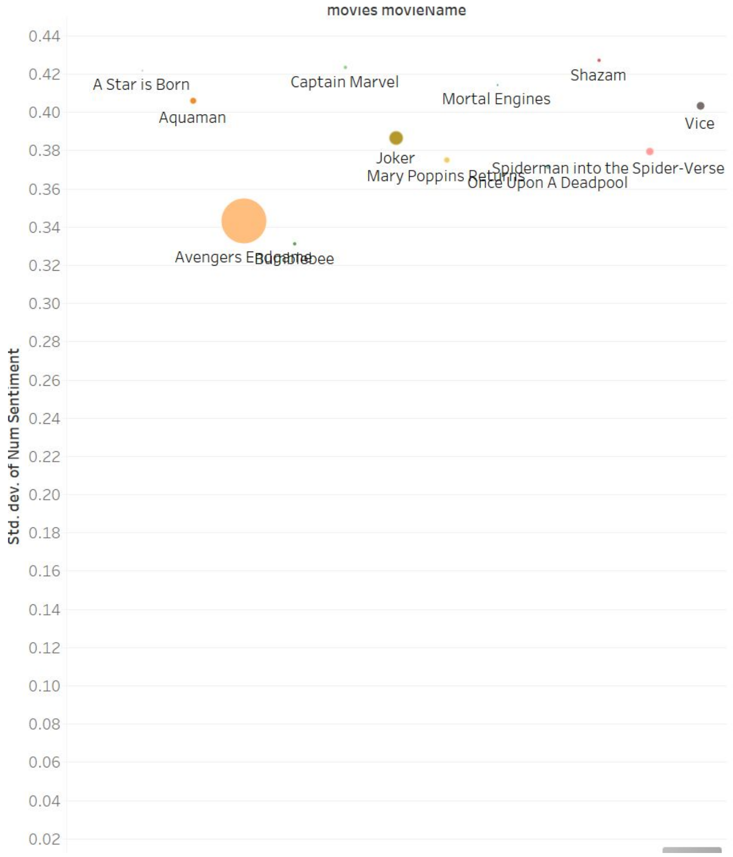
Also the number of large production movies releasing in December were significantly less compared to movies releasing in Summer or in Fall. Because of this even though we had movies with popular actors (high star meter value on IMDB) for an actor, we were seeing unusually low tweets and overall sentiments captured.

5.3 Lack of features

Since we were unable to pull monthly gradient of movie tweets to create gradient features like monthly sentiment changes or monthly tweet counts, we tired to create new features as mentioned before

6. Conclusion

We have created our model with features such as standard deviation and mean of the sentiment of tweets based on the confidence of the tweets. While our output is error laden we have learned the shortcoming in the various features which we had assumed would work by legacy standard and the limitation of the twitter API



7. Libraries And Resources Used

- This project was written in Python
- Python Natural Language Toolkit
- scikit-learn machine learning toolkit for Python
- Tweepy Twitter API library for Python
- Beautiful Soup HTML parser
- Python Programming Tutorials for NLTK

8. References

[1] Prediction of Movies Box Office Performance Using
Social Media - Krushikanth R. Apala, Merin Jose, Supreme Motnam, C.-C. Chan, Kathy J. Liszka, and Federico de Gregorio

[2] Prediction of Movie Success using Sentiment Analysis of Tweets Vasu Jain Department of Computer Science University of Southern California Los Angeles, CA, 90007 vasujain@usc.edu

[3] http://0jadams.github.io/TwitterMovieTracker/

[4] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment Classification using Machine Learning Techniques, Proceedings of EMNLP 2002

[5] https://pythonprogramming.net/pickle-data-analysis-python-pandas-tutorial/

[6] https://www.geeksforgeeks.org/understanding-python-pickling-example/

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier