Helping Hollywood: A Big Data System to Predict Movie Revenue based on IMDb and YouTube features.

Daphnee Chabal
University of Amsterdam
11200898
daphnee.chabal@student.uva.nl

Savvina Daniil
University of Amsterdam
12253561
savvina.daniil@student.uva.nl

Dimitris Papadimas
University of Amsterdam
12286559
dimitrios.papadimas@student.uva.nl

Anthi Symeonidou
University of Amsterdam
12296082
anthi.symeonidou@student.uva.nl

Dimitris Tsimpoukis
University of Amsterdam
12338281
dimitrios.tsimpoukis@student.uva.nl

ABSTRACT

This project aimed to develop and implement a system that generates predictions of a movie's total Box Office revenue prior to its release in theaters. Data from multiple sources such as IMDb, OMDb, BoxOfficeMojo and Youtube were used in order to predict movie's total Box Office revenue. The architecture of the system is able to handle the big amount of data coming from these sources in different timeframes.

1 INTRODUCTION

Feature films cost between 100 and 150 million dollars on average to production and distribution companies [1]. Yet, around 60% of movies fail to turn a profit, making the multi billion-dollar movie making business a risky and costly one [2].

These big financial loses could however be avoided, or at least reduced. Indeed, companies involved in the creation of a movie spend a minimum of 80% of a budget in the promotion and distribution of a movie, rather than on actually producing it [3]. These post-production costs involve producing copies, marketing and advertising it, giving bonuses to key talents, and actual distribution costs [4]. These activities are often unsuccessful because companies ill-estimate a movie's success. When underestimating a movie, companies overspend on promotion and miss out on ticket sales of theaters which did not obtain a copy. When overestimating a movie, companies mismanage promotional efforts and waste screening rooms which could be allocated to better movies instead. By saving on these post-production costs, the movie industry could therefore cut their losses.

The movie industry therefore needs to gain knowledge of a movie's revenue before a movie's release as they start investing the rest of their budget. With regular predictions of a movie's success, they will be able to take informed decisions during their promotional campaigns. Without audience ratings at their disposal however, these companies need more than an educated guess to infer movie popularity and subsequent revenue.

A source of information, on top of facts about the movie itself, is the expressed and inferred indicators of audience anticipation of a movie. Social media constitute massive databases of such indicators. YouTube is particularly informative to the movie industry because the website hosts the official movie trailers released to the audience, which are then distributed across other social media platforms such as Facebook and Twitter. The audience is also likely to decide whether it is interested in going to see a movie based on its reactions to the trailers, rather than other form of information about the movie [5]. Audience interest are expressed in the form of comments and the sentiment they express, likes, and views.

In this project, we aim to generate regular movie revenue predictions during the promotional phase of a movie prior to its arrival in theaters, by giving information about the movie and data accumulating on YouTube between the release of a trailer and its theatrical release to a machine learning algorithm. Thus, our data question is:

Can accurate movie revenue be predicted based on movie characteristics and pre-release audience reactions to movie trailers on YouTube?

To do so, we want to build a system that can harvest these data continuously to generate predictions for movies prior to their release. However, this system cannot be run locally because of the volume of data, which includes every single user comment posted over a period of over 4 months (i.e. between the trailer and the movie releases), the variety of data types which require different processing pipelines.

Therefore, we need to host our design in a big data environment which will be capable of 1. extracting, storing, and processing a big amount of data from relevant characteristics of a movie at different frequencies 2. aggregating various forms of data from different sources (i.e. IMDb and YouTube), and 3. periodically updating its database to generate new predictions on upcoming movies but also to be able to process and learn from new data automatically. Thus, our research questions are:

How can we efficiently acquire, store, and process large amount of content from YouTube and IMDb in a continuous flow?

How can we combine different sources of large amount of data and train a machine learning algorithm in a big data environment?

Such a system architecture, if functional and reliable, would be an asset movie production and distribution companies. For the Data Science community, such successful system design would also encourage the aggregation of more social media data, for these or other purposes.

2 RELATED WORK

In the recent years, many firms collect data related to their brands and products from social media. User's behavior and opinion or sentiment expression can help firms to better design their promotional campaigns by adjusting their products to users' desires and making higher profit for themselves at the same time [6].

A lot of research has been conducted for predicting the box office revenue the previous year by using metadata about a movie, such as movies genre, MPAA rating, running time, release date, the number of screens on which the movie debuted and actors in the cast [7, 8]. In prior work, polarity of critics' reviews as well as the number of stars given by a critic to the movie has been used to determine domestic box office [9].

In the research of Mishne and Glance [10], sentiment analysis techniques to pre-release and post- release blog posts about movies was applied and showed high correlation between box office revenue and sentiment based metrics. In the research of Joshi *et al*, metadata movie features as well as film critics' reviews from several sources prior to the movie release, haven been used in a linear regression model to predict opening weekend revenue. Their finding showed that review text can improve the prediction [11]. Sharda and Delen faced the prediction problem as a classification problem and used neural networks to classify movies into categories. However, the accuracy they achieved was fairly low [8]. Zhang and Skiena combined news data from online daily newspapers with movie metadata to predict the movie box office using regression and k-nearest neighbor models. Better performance was achieved using this information combination rather than using only movie metadata [12]. Asur and Huberman discovered that the rate at which movie tweets are generated can be used to predict more efficiently the box office revenue than the Hollywood Stock Exchange, the gold standard in the industry. They also discovered that sentiment analysis on tweets after the movie is released, can further improve the revenue prediction [13]. Mestyan *et al*, used data from 535 movies were screened in the United States in 2010, from Box Office Mojo webpage and the corresponding movie Wikipedia pages that had been created a lot earlier than the release date of the movie. Activity features mined from Wikipedia pages such as the number of views of the article page, number of people edited the article and the number of edits, along with features mined from Box Office Mojo, such as the number of theaters that screen the movie, were essential parameters in predicting the movie revenue [14].

Rui and Whinston examined the movie revenue forecasting. They showed that there is valuable information on Twitter that can be incorporated into Business Intelligence(BI) systems and might be used for many business purposes, such as forecasting and monitoring certain variables of interest. Their framework was able to incorporate information from Twitter and reduce the Mean Absolute Percentage Error by 36% for the total movie revenue, while for daily revenue forecasting for a period of 4 weeks after the movie release, the reduction was around 17.5%. They proposed that their BI framework could be easily used for the design of other social-broadcasting-based BI systems [15]. The popularity of social broadcasting networks keeps growing, serving more opportunities to the business world of using these systems.

3 THE DATA

The features required were information on the movie which would form a complete picture of a movie's characteristics as well as YouTube user content which would represent fully the spectrum of reactions from the audience, to enhance the learning process of a machine learning algorithm. information on the audience's reaction to the trailers which would Collecting such data for such a task is a challenge because explicit and implicit information about movie characteristics like budget and box office revenue are scattered all over the web and the sources often contradict each other. Our goal was to have data that is as accurate and complete as possible. Therefore, we recommend to prioritize APIs and websites that carry some integrity and contain movie information in a structured way. In the end, the data to be collected are from four different sources presented below. Both categorical and numerical features are included.

3.1 IMDb

The *IMDb* website is the most recognized source of information when it comes to cinema and TV worldwide. In October of 2018, it was reported that it contains 505,380 movies, with different levels of information completeness [16]. Information from IMDb include, IMDb movie id, IMDb rating, number of votes on IMDb, release date, genre and, runtime, among other information about a movie's cast, producers, and director.

3.2 OMDb

OMDb is an open movie database that can be edited by anyone. However it is credible because it cites sources. It also has the most complete total Box-Office revenue data for movies in various IMDb datasets, and OMDb can therefore be used to obtain this specific feature, in US dollars.

3.3 Box Office Mojo

The *Box Office Mojo* website is an ownership of *IMDb* and tracks box office revenue in a systematic way. It is widely used within the film industry as a source of data. The total number of movies on this website is around 19,000 movie pages. The information available on this website include: movie title, budget, opening weekend revenue, MPAA Rating, distributor, release date, widest release and close date.

3.4 YouTube

Movie trailer information prior to the movie's release can be obtained from the corresponding YouTube videos. Several trailers are often posted per movie, but an official version from the distribution company is always featured on the website. Features that can be extracted include, for every trailer, the like/dislike count, the view count, the favorites count and the number of comments, and most importantly the content of all the comments. For trailers whose movies were already released, it is not possible to distinguish whether views or likes were given prior release or not. However, they are necessary features to incorporate in our model, as these counts are updated instantaneously during the pre-release period, thus reflecting audience anticipation as desired. The trailers themselves are not taken into consideration.

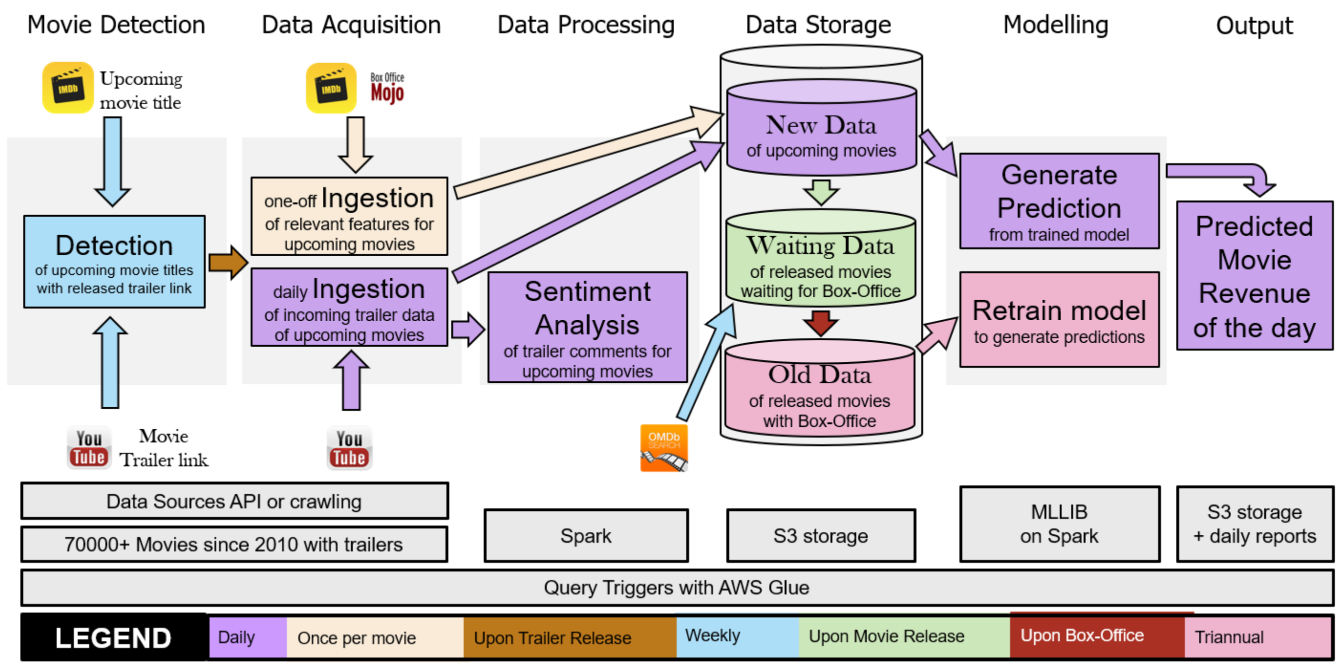

Movie Detection
Data Acquisition
Data Processing
Data Storage
Modelling
Output
Upcoming movie title
Box Office Mojo
Detection of upcoming movie titles with released trailer link
one-off Ingestion of relevant features for upcoming movies
daily Ingestion of incoming trailer data of upcoming movies
Sentiment Analysis of trailer comments for upcoming movies
New Data of upcoming movies
Waiting Data of released movies waiting for Box-Office
Old Data of released movies with Box-Office
Generate Prediction from trained model
Retrain model to generate predictions
Predicted Movie Revenue of the day
Movie Trailer link
OMDb
Data Sources API or crawling
70000+ Movies since 2010 with trailers
Spark
S3 storage
MLLIB on Spark
S3 storage + daily reports
Query Triggers with AWS Glue
LEGEND
Daily
Once per movie
Upon Trailer Release
Weekly
Upon Movie Release
Upon Box-Office
Triannual

Figure 1: Illustration of the system's architecture with proposed implementation tools.

4 SYSTEM DESIGN

The goal of this system is to generate daily movie revenue predictions for every movie released in theaters. To do so, it was important to take into consideration that information about the movie and YouTube trailers are released at different dates. Some data is static as providing facts and details about the movie itself that do not change overtime, and some data is continuously evolving as dependent on incoming reactions from the audience. The system thus needs to extract data from the different sources described in the previous section, combine and employ them at different times and at different frequencies. This will require an architecture capable of triggering certain data processing tasks while simultaneously keeping other data still.

In order to cope with the above mentioned challenges, the system's architecture is divided into six big blocks (see Figure 1), each of which is designed to execute a specific task of the overall work pipeline. These blocks are Movie Detection, Data Acquisition, Data Processing, Data Storage, Modelling and Output.

4.1 Movie Detection

This block is designed to detect the input of our system, that is the upcoming movie titles, as well as their corresponding links on YouTube.

To begin with, upcoming movie detection is accomplished by crawling IMDb's site. More particularly, IMDb has a section dedicated specifically to new movies that are going to be released in the future. It contains information about movie releases up to a year ahead [17]. This particular section is crawled by our system in a weekly basis, in order to update a list of all the movies that are expected to be released.

Another information that is important for our system is the linking of upcoming movies and their corresponding movie trailer on YouTube. However, the announcement of an upcoming movie does not necessarily mean that its trailer is already available. To overcome this issue, our system searches YouTube in a weekly basis, in order to find trailers for movies that it has already detect as upcoming. As soon as a trailer is identified and linked to a movie, the movie is transferred to another list of "upcoming movies with released trailers" where the data acquisition for this movie can begin.

4.2 Data Acquisition

The data acquisition of a movie with a released trailer is divided in two components, one that operates once and another that is executed on a daily basis.

The first component focuses one acquiring all the relevant features for the upcoming movie. To do so, it utilizes the publicly available APIs of IMDb and OMDb and downloads all the relevant information.

The second component is dedicated in obtaining all the necessary meta data from all the detected movie trailers. These data are not static. On the contrary, most of them, such as comments, likes, dislikes and number of views can change very frequently. Thus, this feature acquisition from YouTube is performed in a daily basis and all the new data are aggregated and used to update the active movie dataset. However, most of these data are not yet ready to be integrated in the dataset storage but need to be processed fist.

4.3 Data Processing

From all the acquired data of the upcoming movies, those that require processing are the YouTube comments. Sentiment analysis is applied to each of the new downloaded YouTube comments and a sentiment score for each comment is yielded. These score are then aggregated into a new sentiment feature associated with the movie and stored along with the rest of the trailer metadata and movie features in data storage.

4.4 Data Storage

The data storage is designed to store all the movie features that are acquired through the above mentioned pipeline. It is parted of three major components, one for "new data", one for "waiting data" and one for "old data".

To begin with, the "new data" storage component contains all the extracted data and features corresponding to the upcoming movies. These data are used to produce daily prediction of movie revenues and are stored at this component until the actual release of the movie. After the release of a movie, the data corresponding to this movie will be transferred from the "new data" component to the "waiting data" one. This component is actually a waiting queue where released movies are stored until their official box office is announced.

After the actual revenue of a movie is announced, it is downloaded and stored along with the movie's features to the "old data" component. This component contains all the movies that have been detected and processed from this system, along with their acquired and engineered features and their actual box office. This dataset is maintained in order to be used to retrain the predictive model when scheduled.

4.5 Predicting Model

This block is designed towards the generation of a model to predict the box office of the upcoming movies. This model operates in two modes, a training mode and a predicting mode.

The training mode is enabled three times a year, in a scheduled maintenance format. In this stage, the data from the "old data" component of the storage are used in order to retrain the model with a bigger and more contemporary dataset. This mode ensures a constant improvement of the predictive model's performance as it is trained using an increasing in size "up-to-date" dataset.

On the contrary, the predicting mode is used in a daily basis. As mentioned before, every day new data are collected from YouTube, processed through the system and stored in the "new data" component. As soon as all data from this component are updated, they are fed to the model in order to generate daily predictions for all upcoming movies.

4.6 Summary of Time frames

Our system will thus trigger tasks at 7 different frequencies.
Data ingestion of YouTube features occurs daily, with immediate sentiment analysis incorporating new comments to previous ones to generate new scores, as well as immediate aggregation of results into the algorithm to update predictions of movie revenue daily. Details about movies are only crawled once per movie, and immediately combine with the "new data".

Weekly, new movie titles are detected and registered into our system, as well as their corresponding trailer link. Jobs that check whether a movie trailer or a Box-Office figure have been released are also scheduled weekly.

Data about a movie is migrated to a new location upon the theatrical release of a movie, which is then transferred again upon detection and incorporation of a Box-Office figure. A tri-annual trigger sets the resulting "old data" as additional training data for the algorithm to update the examples it learns from.

4.7 Output Delivery

Our system would generate movie predictions updated daily for each movie. To give companies access to these predictions, the revenue forecast for each movie will be integrated into a set-format PDF, with accompanying movie title, automatically sent to a client's preferred e-mail address.

5 IMPLEMENTATION IDEAS

The architecture designed was implemented on a local level. The different tasks that had to be completed in order to simulate the conceptual plan were collected and distributed among the team members. The scripts were created in our separate notebooks and the results were stored in the local disks. The local implementation was as follows.

5.0.1 Data extraction. To extract movie information from IMDb, we used the rich API offered specifically to extract data from the website, although its limits didn't allow us to fully deploy it. Instead, we used daily refreshed data sets included in the website consisting of 5 million titles (including TV episodes). The data sets were combined based on the *IMDb* id of each movie. The merged data set was cleaned and totalled around 70 000 movies after 2010 with trailers. We set 2010 as a time criteria to insure that all movies would have trailers on YouTube. The features included IMDb rating, number of votes on IMDb, release date, genre and, run time. Note that the rating and the number of votes are contingent on a movie's release and the audience's reactions to it, and were collected for research purposes, given that our system aims to generate predictions based on information available only prior to a movie's release.

For OMDb, a great advantage was that its API does not have usage limitations so we used it to get box office revenue information for the movies obtained from the IMDb data set. In order to effectively combine the two sources, we used the IMDb id of each movie, since it was available in the OMDb API. Almost all movies from the IMDb dataset were featured in our extraction of OMDb data. To obtain features from Box Office Mojo, we first extracted the urls to the site pages of the movies available on Box Office Mojo in an alphabetical order. The `urllib.request`[1] and `BeautifulSoup`[2] libraries were used in order to extract the urls for each movie and scraping the required data. In the phase of merging all the data from all the available sources, only the MPAA Rating score, opening weekend revenue and the budget were kept since the other features have also been extracted from the *IMDb* site. Note that the opening weekend revenue, being a post-release variable, was also collected for research purposes.

[1]https://docs.python.org/3/library/urllib.request.html
[2]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

YouTube offers a rich API for scraping the website's content but has a daily quota in the number of requests to the site, which limited the amount of available data that could be obtained. We obtained only the three first pages of comments from 1221 trailers. Nonetheless, the API allows both searching for videos, as well as obtaining multiple video attributes in a programmable environment [3]. Having obtained the video titles from the previous data sources (IMDb, OMDb etc.), we used them as search terms for the corresponding movie trailers using the YouTube API. The search was subject to filtering, so that the upload was made from the official YouTube channel as well as some textual filtering to account for the cases where the videos extracted were not primary trailer. All comments were subject to sentiment analysis and some additional features were obtained (see 5). We did not manage to extract likes/dislikes information. In the cases where there were multiple trailers for a single movie, the results for each video were aggregated and averaged. The data from all sources were saved in a `CSV` format for easier manipulation.

Ideally, more complete data could be extracted especially for the sentiment analysis part, since we are extremely limited by the Youtube daily quota in the amount of comments that we can scrape. We applied twice for an extension in daily quota but was rejected both times. In a complete enterprise solution the quota extension could be provided more easily. What could prove very useful would be to include additional data from other more open data sources. Particularly, the *Twitter* API could be used in order to collect more user comments and reactions that could be used in order to create more sentiments features.

5.0.2 Data preprocessing. The data have been preprocessed as is described in the *The DATA* section. Dummy variables were created for all the categorical features in order to be supplied into the regression models. In other words, each separate value of the categorical features was represented as 0 or 1 for each movie.

5.0.3 Data Aggregation. The data from the four different sources had to be aggregated into one file. For this reason, common features (key) were used into a `join` function. The IMDb and OMDb data were aggregated using the "IMDb ID" feature, while the resulted file was merged with the Box Office Mojo and YouTube file by using the "movie title" feature as the key. The obtained dataframe which was used for the regression model had 1226 entries. Since the implementation is a proof-of-concept, these aggregated data were enough to train and test the model.

For future implementation, the "movie title" should not be used as a key to combine the data sets due to possible double title implications. The "IMDb ID" key is far more adequate as it is unique.

5.0.4 Sentiment Analysis. Opinion mining from a given text is a rigorous process and to do that we utilized a specialized tool named VADER (Valence Aware Dictionary and Sentiment Reasoner) [?]. It is supplied through the `NLTK` library and is a sentiment analyzer created specifically for extrating opinions from social media texts. The VADER analyzer when supplied with a text (in our case a comment) yields a score in the range of $[-1, 1]$. For every comment in a YouTube trailer we obtained its sentiment score and separated them into positive and negative comments. The extracted features from the sentiment analysis used for model training were the average positive and negative sentiment scores as well as the total number of positive and negative comments.

[3]https://developers.google.com/youtube/

5.0.5 Missing data. Generally, the CSV lines of the data set that were missing values were dropped. The reasoning behind it was the limit set by the YouTube's API which would not allow us to get trailer information for tens of thousands of movies anyway. The only feature that was filled with generated numbers instead of dropped when missing was the budget, since we considered it a very important feature for the model.

To fill the None values of the budget, the following analysis was performed. The existing budget values were plotted against the opening weekend revenue values and it was noticed that as a general rule, the movies in the data set achieved in revenue 0-100% of their budget during the first weekend. Thus, the budget missing values were completed by multiplying the opening weekend revenue of the corresponding movie by a random number between 0 and 1.

5.0.6 Model training. The prediction of the movie box office is the output of our architecture. Since it is a numerical feature, a regression model is the appropriate model for this prediction. Three different regressors have been used: a basic linear regression model, a support vector regression model and a random forest regression model.

5.0.7 Data Storage. As mentioned, all the consecutive or simultaneous tasks were performed locally. Thus, the outputs were also saved locally. The choice was made due to the fact that the data was not so voluminous at the end and there was no need to store it in some online service. However, future implementations will head to the direction of saving the data on S3.

5.0.8 Prepping for Big Data. Even though for this work a local solution was implemented mostly due to data availability and crawler request limits, in a real world scenario the proposed pipeline would require a cloud implementation due to the increased volume as well as computational power requirements.

With that in mind, we will opt for an AWS cloud solution. Given that the only part of our pipeline that would require ETL operations on a very large dataset is executed tri-annually (machine learning model training) the data would be stored in S3 for its ease of use and low cost. The rest of the workflow such as weekly movie detection, crawling, sentiment analysis and predictions that are performed on a more frequent basis and smaller amounts of data can be executed on on-demand AWS EMR/EC2 instances with appropriate computational resource allocation. The actual data manipulation as well as predictions will be performed on Apache Spark.

The biggest challenge in our implementation is the automated workflow scheduling, given that different parts of the pipeline need to be executed in different timeframes (monthly, weekly, daily). A solution to the above problem would be to make use of AWS Glue which offers a serverless environment that allows scheduled crawling and ETL operations on data stored in S3 directly without specifically assigning computational resources [18]. Instead, it allocates the necessary resources in a dynamic way which would not require us to generate specific EC2/EMR instances as shown above. Even though it is subject to some limitations (all operations

Table 1: List of Features used for model training

	Features
Prior To Release	Movie Runtime Budget Genre (Categorical) MPAA Rating (Categorical) Trailer Like/Dislike Ratio Trailer View Count Trailer Comment Sentiment Positive Score Trailer Comment Sentiment Negative Score Trailer No of Positive Comments Trailer No of Negative Comments
After Release	IMDb Rating IMDb No. of Votes Opening Weekend Box Office

must be executed in pure Python or Spark) it could prove an easily deployable and totally scalable solution. In the current implementation we are not limited by the above limitations, but in the case that we do (e.g need for GPU instance for Neural Network training) we can always opt for the AWS Data Pipeline service which gives total resource control in fully automated scheduled pipelines [19].

6 RESULTS AND ANALYSIS

The final dataset which was used in order to train and test the different regression models had the sets of features as they are shown on Figure 2. More information about the data preprocessing and aggregation has been mentioned in the corresponding subsections of 5.

6.1 Basic Linear Regression model

The `sklearn.linear_model` library was used to train a simple Linear Regression model on 80% of the data, chosen randomly. For this training, the *IMDb* ratings information was deployed as well. Then it was tested on the remaining 20%. The mean absolute error (MAE) was calculated to be 28 million, meaning that for every movie the prediction for the box office was off by 28 million dollars on average.

6.2 Support Vector Regression model

As for the Support Vector Regression (SVR) model, the `sklearn.svm` library was used for training. The *IMDb* ratings information was included here too. The difference from the basic Linear Regression is that it tries to fit the error of the regression within a certain threshold instead of minimizing it, using the margin of error idea of the Support Vector Machines. Different configurations of the hyperparameters were attempted but the ones that performed the best were the epsilon (margin of error) set to 1 and the normalization parameter set to 1000. Once again, the data set was randomly divided 80-20% to train and test. At the end, the MAE was around 19 million dollars.

6.3 Random Forest Regression model

The `sklearn.model_selection`[4] and the `sklearn.ensemble.RandomForestRegressor`[5] libraries were selected in order to split the data into train and test set, as well as, to train the model. The `Numpy` library were also used in order to transform the data into numpy arrays to feed them into the model.

The `n-estimators` parameter were set to 1000 trees, while the data were split into 75% training and 25% test data. The feature importance was also calculated. This is an indicator of which features participate more in the tree formation and therefore, how important they are in order to predict the selected label.

The MAE was the metric used in order to compare the performance of all aforementioned models. The Random Forest Regressor was the model that performed best with MAE equal to 16,468,317 millions dollars when only the features prior to movie release were used (Table 2).

Additionally, the training and box office revenue prediction was executed by using features related to post release date period of a movie, such as the "number of votes", the "average rating" and "Opening Weekend Box Office" features in order to inspect their importance and examine model's performance. In this case the MAE was lower (12,482,499 millions dollars when "number of votes" and "average rating" features were used) and as it is shown on the Figure 2a, the "number of votes", which indicates the number of people voted for the movie on *IMDb* is the master predictor of the box office revenue (Figure 2b). This information could be used in order to create other features that mimic the "number of votes", such as the ratio of likes and dislikes of a movie trailer on YouTube, and predict more accurate the box office revenue.

When the "Opening Weekend Box Office" feature also included to the model, it was the most important feature for the movie revenue prediction and the MAE apparently dropped to 7,785,853 millions dollars (Figure 2c).

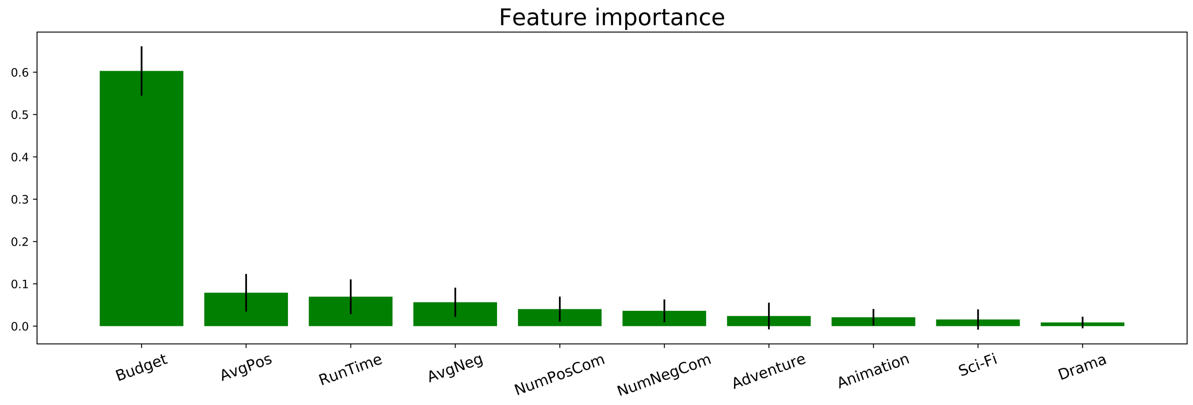
Table 2: Mean Absolute Error

	Mean Absolute Error (dollars)
With features prior to release	16,468,317
With features prior and after the release(only IMDb Rating and IMDb No.of Votes	12,482,499
With features prior and after the release(including Opening Weekend Box Office	7,785,853

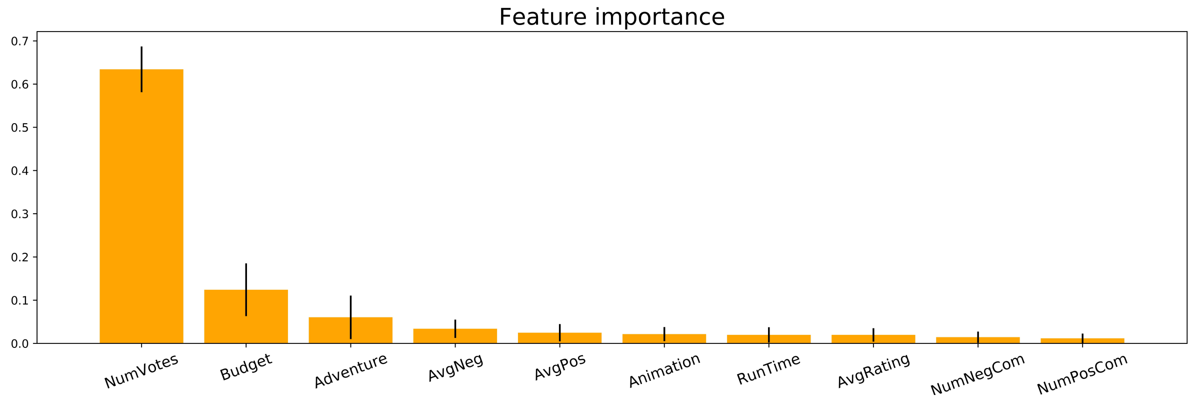
7 SCALABILITY

With a fully operational system, that is, one without YouTube data accessibility issues and that is implemented on AWS Glue, we can expect an improvement of these results, in three foreseeable ways. Firstly,as previously mentioned, other algorithmic models, neural networks,can improve the predicting performance but only with more data to train with [**?**]. Secondly, some features that are not available prior to a movie's release but proved informative for our regression models could also be mimicked by processing available information. For example, the ratings can be replaced by some computations of a likes to dislikes ratio of incoming audience responses

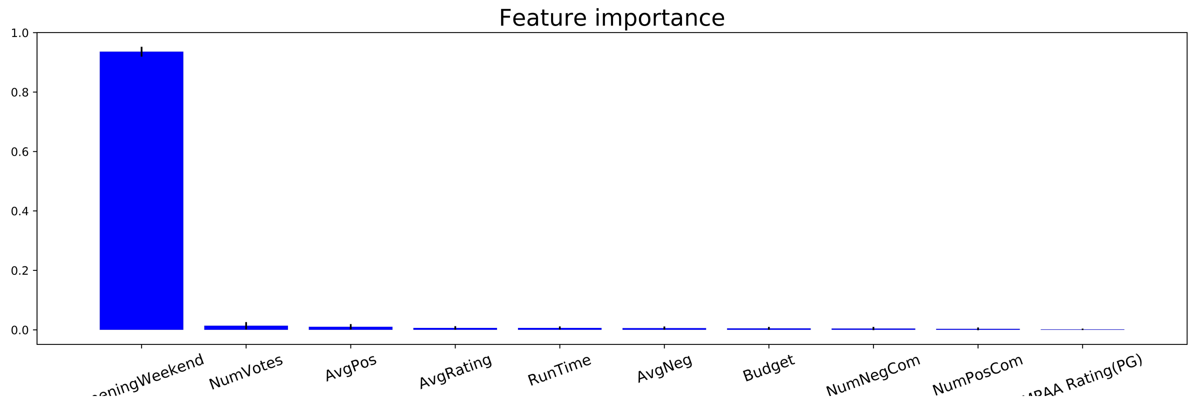
[4]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[5]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html



(a) Feature importance prior to movie release



(b) Feature importance after the movie release



(c) Feature importance after the movie release with financial data

Figure 2: Feature Importance

to YouTube trailers. Finally, more information about movies is contained in the movie plots available on IMDb and Wikipedia. It is possible to extract these plots, process them via a topic analysis, and aggregate the resulting scores in the "new data". Such supplementary analysis would further rely on a big data infrastructure to host our architecture.

This design, if functional offers a lot of promises. For the movie industry, it is possible to move on to additional social media sources, such as facebook and twitter to gather more insights on the audience anticipation of a movie which could improve accuracy. Our hope is also to demonstrate the extent to which audience reactions on social media can be informative and predictive of behavior, and that our current approach can be easily transferable to fields where public opinion provides valuable insights, such as businesses in general, public governance, and policy making.

8 CONCLUSION AND FUTURE WORK

This project aimed to help movie production and distribution companies in their effort to maximize profit for each movie they release. To do so, we set to generate daily predictions of a movie's Box-Office revenue during the period leading up to the theatrical release. Input of our prediction model were information from IMDb, OMDb, and Box Office Mojo, all three of which we recommend for implementation, and continuously processed Youtube data, such as comments and view counts, to quantify the audience anticipation, and thus conversion likelihood, of a movie.

The system architecture proposed here enables such goal. Information from different sources are ingested at different times and different rates, to generate revenue predictions, from a model which is re-trained periodically. Thus, this system is self-sufficient, automated, and designed for a smooth workflow. The system integrates tasks of data detection, data ingestion and processing, data storage, modelling and prediction generation, at 7 different time frequencies. We demonstrated that it was possible to design a system which efficiently takes in and processes a large amount of data which is continuously integrated.

By implementing our system locally, we were able to generate predictions, but with limited accuracy. This was due to the shortage of training examples, due to data crawling quotas, and to the fact that we could not extract all the features we wanted to (i.e. likes/dislikes) because of programming difficulties. We discuss several solutions selected to resolve this. The next step of this project is to enable the processing of large amount of incoming data for all movies worldwide, as published on IMDb, and thus on a cloud platform. AWS Glue[6] is proposed as the service which enables the workflow design with automated job triggers. Future implementations of our project should also consider different prediction models and some new features which can be computed.

In conclusion, we recommend to the movie industry to implement a system architecture which aggregates data from different websites and performs extraction, processing, storing, and predicting tasks at different timeframes and different frequencies, to generate regular movie revenue predictions, in the hope of cutting their losses, and achieve what all businesses want: a profit.

REFERENCES

[1] Feature film budget breakdown. http://parlaystudios.com/blog/feature-film-budget-breakdown/. Accessed: 2019-02-30.
[2] Do hollywood movies make a profit? https://stephenfollows.com/hollywood-movies-make-a-profit/. Accessed: 2019-02-30.
[3] https://towardsdatascience.com/deep-neural-networks-for-regression-problems-81321897ca33. Accessed: 2019-02-30.
[4] How movie distribution works. https://entertainment.howstuffworks.com/movie-distribution.htm. Accessed: 2019-02-30.
[5] Do youtube movie trailer searches correlate to box office success? https://tubularinsights.com/do-youtube-movie-trailer-searches-correlate-to-box-office/. Accessed: 2019-02-30.
[6] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.
[7] Jeffrey S. Simonoff and Ilana R. Sparrow. Predicting movie grosses: Winners and losers, blockbusters and sleepers. *CHANCE*, 13(3):15–24, 2000.
[8] Ramesh Sharda and Dursun Delen. Predicting box-office success of motion pictures with neural networks. *Expert Systems with Applications*, 30(2):243 – 254, 2006.
[9] N Terry, M Butler, and D De'Armond. The determinants of domestic box office performance in the motion picture industry. *Southwestern Economic Review*, 32:137–148, 01 2005.
[10] Gilad Mishne and Natalie Glance. Predicting movie sales from blogger sentiment. pages 155–158, 01 2006.
[11] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith. Movie reviews and revenues: An experiment in text regression. pages 293–296, 06 2010.
[12] W. Zhang and S. Skiena. Improving movie gross prediction through news analysis. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 301–304, Sep. 2009.
[13] S. Asur and B. A. Huberman. Predicting the future with social media. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 492–499, Aug 2010.
[14] MÃČÂąrton MestyÃČÂąn, Taha Yasseri, and JÃČÂąnos KertÃČÂĺsz. Early prediction of movie box office success based on wikipedia activity big data. *PLOS ONE*, 8(8):1–8, 08 2013.
[15] Huaxia Rui and Andrew Whinston. Designing a social-broadcasting-based business intelligence system. *ACM Trans. Manage. Inf. Syst.*, 2(4):22:1–22:19, January 2012.
[16] Imdb statistics.
[17] IMDB: New Movies Coming Soon. https://www.imdb.com/movies-coming-soon/2019-12/. Accessed: 2019-02-30.
[18] Amazon Web Services. AWS Glue. Simple, flexible, and cost-effective ETL. https://aws.amazon.com/glue.
[19] Amazon Web Services. AWS Data Pipeline. Easily automate the movement and transformation of data. https://aws.amazon.com/datapipeline.

A TEAM MEMBERS CONTRIBUTIONS

All 5 members contributed equally to this project. Task division was decided as a group and everyone delivered their tasks in time and in a useful manner.

[6]https://aws.amazon.com/glue/