Movie Grossing Success Prediction with Convolutional Neural Networks on Graphs

Oriol Barbany, Manuel Cherep, Natalia Gullon and Carlos Medina
A Network Tour of Data Science (EE-558), School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
{*oriol.barbanymayor, manuel.cherep, natalia.gullonaltes, carlos.medinatemme*}@*epfl.ch*

Abstract—**Predicting a movie's success before it is released would be a major benefit for movie producers. We investigate this possibility on movie's accessed from The Movie Database (TMDb). Based on the movie's meta-data, e.g. cast or genre, we aim to provide a tool to predict if a movie will generate income or loss. We show that such prediction is possible by applying Graph Convolutional Neural Networks (GCNNs). This GCNN was applied over a graph of features extracted from movies' meta-data. Our results show improved performance over a linear classifier and are equally good as those of a comparable fully-connected neural network.**

Index Terms—**Convolutional Neural Networks, Movies, Graph Processing, TMDb**

I. Introduction

The film industry is generating several billions of dollars each year and this amount has been increasing over the last years[1]. While some movies with relatively low budget, as in the case of *Paranormal Activity*[2], have obtained huge benefits, it is not rare to see multi-million productions failing on the box office and losing huge amounts of money. To prevent this, it would be useful to have a means to predict how successful a movie will be before it is produced. This would enable movie producers to allocate their budget exactly on those factors, which will make their movie successful (e.g. is it worth to spend the budget on a good cast or a good director?).

In this project, we address the problem of predicting movie success from a range of movie meta-data. The original meta-data list was taken from Kaggle's TMDb 5000 Movie Dataset [1], but adjusted as described in section II. Our aim is to demonstrate that movie success prediction is possible by using a Graph Convolutional Neural Network (GCNN) [2]. The associated graph is built over the different movie meta-data fields. Thereby, we simultaneously provide a product prototype enabling users to select certain movie features and letting our model predict, whether this future movie will generate profit or not. We, therefore, reduced the problem to an easy to interpret binary classification task only predicting, whether a movie's revenue is higher than its budget or not. Indeed, our results show, that such prediction is possible.

[1]Statistics found in https://www.the-numbers.com/market/

[2]We highlight the case of this movie as it was the major outlier found in the used dataset. We provide further details in section II-E

II. Data Exploration

The dataset consists of a list of movies with a range of associated meta-data (e.g. budget, runtime, a list of cast or production companies). The original meta-data list was taken from Kaggle's TMDb 5000 Movie Dataset [1]. However, the provided movie meta-data displayed a range of problems (see section II-A) and did not seem suitable for our task. Therefore, we had to replace the data set by our own movie list gathered from the public TMDb API (see section II-B).

A. Original Kaggle Data

The number of movies in Kaggle's TMDb 5000 Movie Dataset [1] is 4,803. Figure 1 shows that many movies have a budget smaller than \$1,000, which seems very unlikely as a movie budget. Two examples that fit this category are *Four Single Fathers* with a budget of \$3 and *El Rey de Najayo* with budget of \$500. Double checking on IMDb shows that *Four Single Fathers* actually has a budget of \$3,000,000 and *El Rey de Najayo* has a budget of \$500,000. Manually checking more movies shows that there does not seem to be a systematic pattern of how to map these small numbers to actual budgets. We can, hence, see that we cannot trust the data in that value range. Excluding those movies, we are left with only 77.3%, i.e. 3,712 movies. Our solution consists in collecting new correct data directly through the public TMDb API.

B. Collected Data

We decided to download data from movies from the same original source (i.e. TMDb) using their API. However, instead of downloading a random amount of movies we design an heuristic to maximize the number of movies with a correct budget and revenue. We use the API to download movies sorted by revenue in descending order, and then select the first 500 pages. The new budget distribution is shown in Figure 2. We observe that now we have an even smaller percentage of movies with the right budget (i.e. 66.5%). However, now we have almost twice the number of suitable movies (i.e 6,651) than we had before. Note that most movies with a budget smaller than \$1,000 have a missing budget, which corresponds to zero.

C. Preprocessing

Several preprocessing steps were taken to clean the data. These steps are summarized in Table I and described in detail in the paragraphs below.

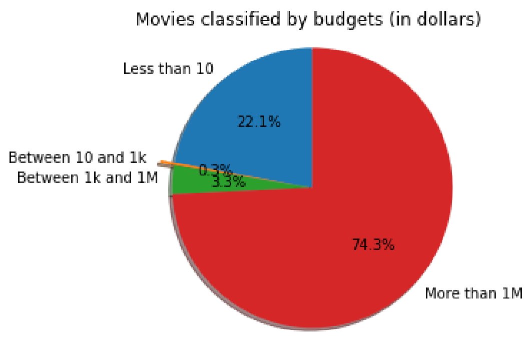

Fig. 1: Kaggle's TMDb 5000 Movie Dataset classified by budget order

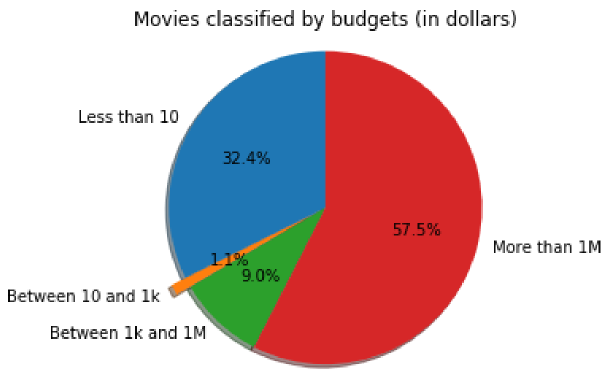

Fig. 2: Improved dataset classified by budget order

As the project is centered in building a product for predicting future movies' success, the inputs to our model should only contain the information a potential movie producer has at hand before starting production. Accordingly, we dropped the vote average, vote count and popularity, as they are only known after production.

We further dropped all movies that were not released yet, or that were missing either the budget or the revenue, since our target (revenue $>$ budget) cannot be computed for them. We further noted, that many budget entries seem to be too low (cf. Section II-B; Figure 1). Thus, we discarded all movies with a budget smaller than \$1,000.

Feature	**Action**
Vote Average	dropped
Vote Count	dropped
Popularity	dropped
Budget	removed movies with missing budget or budget $<$ \$1000
Revenue	removed movies missing a revenue
Status	removed un-released movies

TABLE I: Dropped features and movies.

D. Feature Engineering

We define movie success as the ratio between the revenue and the budget. If the ratio is greater than one, a movie is successful, otherwise it's not. By relying on a ratio rather than absolute numbers we guarantee that even very old and new movies are comparable. Due to inflation, the absolute values will increase. However, the ratio should stay constant. This ratio will be our classification target given a range of movie meta-data features. While some features like *runtime* were already in an adequate numerical format, there were several categorical and textual features as well. Furthermore, fields like *genres* or *cast* were provided as lists of genres or actors, respectively. Accordingly, we had to employ some feature engineering to convert them to numbers. The resulting features are listed in Table II and are described in detail below.

We will first focus on the list-like features. Here we deemed cast, crew, genres and production companies to be potentially important factors. Thus, we selected the main actors and actresses of each production as well as the director from the crew, and we mapped each of them to the average ratio of the movies they have participated in. We did the same for the production companies and for the genres. These features were only computed over the training set, since they implicitly include the target we are predicting and would otherwise lead to data snooping.

For the cast, we took the 3 most important actors or actresses who appear in a movie. However, not all of them appear in the same number of movies. Actors consistently appearing in high ratio movies are likely to be part of the movie's reason for success. For actors appearing in a single movie only, such estimations are less reliable, since many other factors could have decided on movie success. Therefore, we multiply the obtained average ratios by a weighting factor of the count of movie appearances. This weighting factor is a function of the ratio between the number of movies in which the actor appears and the maximum number of movies starred by an actor in our database, inspired by [3]. This factor can also be extended to the other list-like features.

The average ratio computed for a particular actor i can be expressed as:

$$r_i = \left(\frac{1}{N_i}\sum_{n=1}^{N_i} r_{n,i}\right) \cdot \left(\frac{N_i}{\max_i N_i}\right)^{\beta} \tag{1}$$

where N_i is the number of movies where the actor i appears, $r_{n,i}$ is the ratio of the movies n starred by the actor i and β is a coefficient that adjusts the importance of the number of movies factor. In particular, we chose a value of $\beta = 0.3$ after checking that the cast with higher ratio is coherent given their popularity and number of successful movies. As a result, the feature *cast*, is the average over the individual actors revenue/budget ratios (r_i's).

In the case of the production companies, there also exists an ordering by importance (role and investment). Therefore, we weighted the ratio of every production company in a

Feature	Action
Cast	Mean over top 3 actors' revenue/budget ratio
Production companies	Mean production companies' revenue/budget ratio
Genres	Revenue/budget ratio
Director	Revenue/budget ratio
Release year	—
Release month	—
Run time	—
Homepage	Does the movie have a homepage or not?
Tagline	Does the movie have a tagline or not?
Original Language	English or other language?
Individual genres	One dummy variable for each of the 20 genres.

TABLE II: All used features after feature engineering.

movie depending on its importance and performed the same calculation as for actors before.

The numerical value computed for production companies for a particular movie i can be expressed as:

$$R_{prod,i} = \frac{1}{N_i}\sum_{k=1}^{N_i} e^{-k\alpha} r_i \tag{2}$$

with N_i being the number of production companies of movie i and α a positive constant below 1 that is aimed to weight the importance of each production company on the average ratio due to production companies of movie i. We chose a value of $\alpha = 0.1$ after checking that the production companies are given a proper decreasing factor and the results are coherent.

For genres the above procedure was performed, only without weighting. Thus, a movie's *genres* field contains the average revenue/budget ratio over all its genres' revenue/budget ratios.

Apart from these list-like features, there are some more features we engineered. We additionally included the release year and month, since selecting the right time for releasing a movie might be important. Furthermore, we added some dummy variables for the following relations: Does the movie have a homepage, tagline or not? Is the movie in English or some other language? For each individual genre, is the movie of that genre? Thus we used a total of 30 features.

E. Outliers

Another problem we had to deal with was the presence of outliers. Some movies show extreme ratios (cf. Figure 3), e.g. *Paranormal Activity* has a revenue/ratio of 12,890. It generated a revenue of 12,890 times its budget. We determined the outliers following the inner fence approach described in [4]. This approach is based in determining the inter-quartile range IQR of the data separated by features, i.e. the difference between the third and first quartiles, and then setting expected minimum and maximum values from that. All the data points that lay outside this range are considered outliers. Then, the inner fence defines the following limits:

$$\min = Q_1 - 3IQR \quad ; \quad \max = Q_3 + 3IQ \tag{3}$$

After removing the outliers, the histogram of ratios revenue/budget looks more sensible. The number of movies decreases as the ratio increases, as shown in Figure 4.

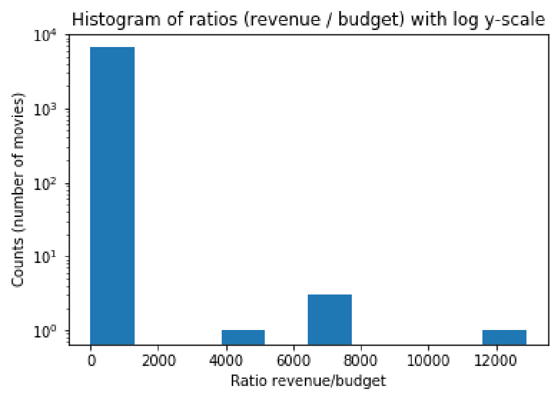

Fig. 3: Histogram of movie ratios with outliers.

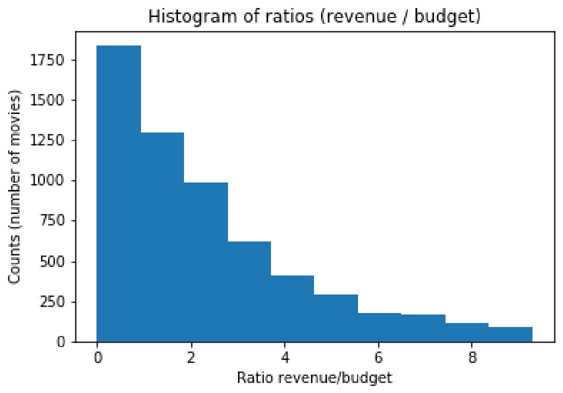

Fig. 4: Histogram of movie ratios after removing outliers.

III. Graph

A. Creation

We built a graph, where the nodes represent the individual movies. The edges are determined by distances between movies as calculated based on the movie meta-data features. The metric we employed to compute the distance and create the adjacency matrix is the Euclidean. The graph weights were computed with a Gaussian Kernel which expression is shown below.

$$\mathbf{W}[i,j] = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{\sigma^2})$$

,

where σ, i.e. the kernel width, was set to the mean over all distances. The graph nodes represent the individual movies, so we have 5,589 nodes. To avoid having a connected graph, we force sparsity in the adjacency matrix by setting a threshold of 0.90. After dropping the edges with lowest weight, we end up by finally having 25,519 edges, which is much lower than with the previously connected graph, where we had

$$\frac{N_{\text{nodes}}(N_{\text{nodes}} - 1)}{2} \sim 15 \textit{ millions of edges}$$

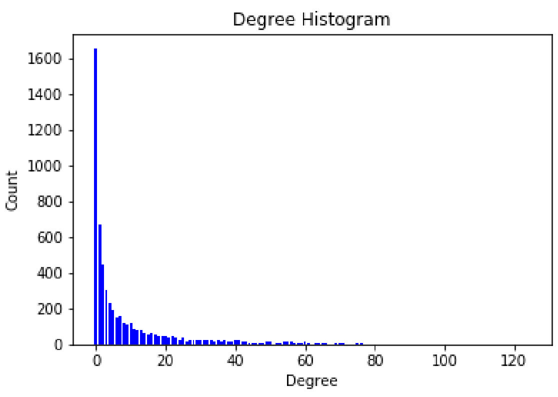

Fig. 5: Degree distribution of our graph.

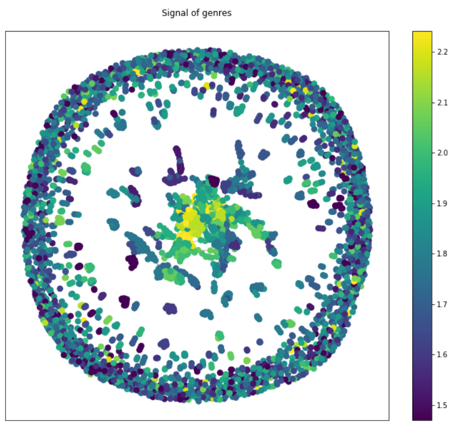

Fig. 6: Genres feature over the graph.

Additionally, our graph is un-directed since the relation between movies is symmetric, i.e., if a movie A is related with another movie B, then the movie B is related in the same way with movie A.

B. Properties

In order to characterize our graph, we analyzed some general properties to determine which model it fits better to. We first plotted the degree distribution of our graph and realized that it follows a power law distribution, i.e. the count rapidly decreases as the degree increases, as shown in Figure 5. This shape is common for scale free networks. After it, we computed the average clustering coefficient, which takes a value of 0.343. This value indicates the tendency of the nodes to cluster together in our graph and is substantially bigger than in scale free networks, which was already highlighted in previous explorations of this same movie data set and better modeled in [5].

With the aim of seeing the similarities in our graph, we plotted some signals over it to detect patterns. We realized that, in particular, with the genres feature, which consists of the average ratio of the combination of genres of every movie, we find some similarities around neighboring nodes as depicted in Figure 6. To explode this relation, we then ran the k-means algorithm to detect these clusters. Indeed, as there are 19 different genres, we found as much clusters and then studied the similarities among them. As expected, the movies grouped into the same cluster turned out to have a very similar value for the genres and also for the binary variables indicating the presence of a genre. This last remark was quite surprising given that the presence or absence of a genre is related to its average ratio of the mixture of all the genres of a movie, but this mapping is obviously not one to one.

Layer	Filters	Polynomial Orders	Nodes	Params
Conv1	10	20	320	200
FC1	0	0	50	16000
FC2	0	0	50	250
output	0	0	1	100

TABLE III: GCNN structure.

IV. Data Exploitation

A. Features Graph

We built a graph over the features, i.e. with 30 nodes, one for each feature. For each of the nodes edges to the three nearest neighbors (Euclidean distance) were added. The corresponding weights were calculated using a Gaussian kernel.

B. Graph Convolutional Neural Network

Since there were more successful (67%) than unsuccessful movies, we randomly up-sampled the unsuccessful movies to match the number of successful movies. Now predicting always the same class would result in a classification accuracy of 50%. As a model to predict movie success we employed a Graph Convolutional Neural Network (GCNN) [2]. In particular, we have a classification task determining whether a movie would make more money than invested or not using our selected features. Our GCNN is summarised in Table III. It was trained using the code from [2]. The training was for 200 epochs with a batch size of 100, a learning rate of 0.25, a decay rate of 0.95, momentum of 0.9 and a regularisation parameter of $5e-10$. All other parameters (pooling, Chebyshev filters, ReLUs) were set as for the best performing models in [2]. The training was performed in a Google Colab using a Tesla K80 GPU.

V. Results

Table IV shows the results from comparing our GCNN to a comparable fully connected neural network and standard logistic regression. The CGNN outperforms a logistic regression classifier and performs equally to a comparable fully-connected neural network. Comparable here refers to the fact, that it is similar in both number of layers, number of parameters and has all other hyperparameters, e.g. learning rate, set to exactly the same value.

Model	Training Accuracy	Test Accuracy
GCNN	99.85 ± .07	85.75 ± .64
FC-NN	99.72 ± .05	85.52 ± .76
Logistic Regression	77.56	74.41

TABLE IV: Results.

VI. Discussion

There are a couple of possible extensions to our current work. For example, the movie overviews could have been used for sentiment analysis. Maybe happy movies tend to be more successful. Furthermore, it is not clear how well our model generalises for edge cases as the previous discussed *Paranormal Activity*, which had a very small budget, but a huge revenue.

VII. Conclusion

In the present paper we demonstrate that a future movie's success can be predicted from its meta-data, e.g. cast or genre. This was achieved through a graph Convolutional Neural Netowrk trained on a graph of movie meta-data. This meta-data was feature engineered based on a cleaned version of a movie list, downloaded through the TMDb public API. Our results suggest GCNNs as a valuable tool to forecast movie's success already before production starts.

References

[1] "Tmdb 5000 movie dataset," https://www.kaggle.com/tmdb/tmdb-movie-metadata/home, accessed: 2018-09-28.

[2] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 3844–3852.

[3] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[4] J. Tukey, *Exploratory Data Analysis*, ser. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977.

[5] K. Klemm and V. M. Eguiluz, "Highly clustered scale-free networks," *Physical Review E*, vol. 65, no. 3, p. 036123, 2002.