

# Predicting IMDB movie ratings using Google Trends

Deniz Demir, Olga Kapralova, Hongze Lai

December 15, 2012

Movie ratings are influenced by many factors, so accurate prediction of new movie ratings can be challenging. In recent years, various semantic analysis techniques were successfully applied to analyzing user reviews, which in turn were applied to predict IMDB movie ratings (based on IMDB reviews, youtube movie trailer reviews etc). To the best of our knowledge there has been no research done on predicting IMDB movie ratings using Google search frequencies for movie related information. However, several authors have suggested to use search volume to predict consumer behavior [1, 2].

Our main idea in this project is to characterize each movie by the set of features, and then use Google search frequencies of these features to predict movie popularity. The intuition behind this approach is that for popular movies one should see the higher search volume of queries associated with the movie.

We view this problem as a supervised learning problem, and we used logistic regression, SVM and multi layer perceptron algorithms for our project. We also used dimensionality reduction techniques to select the optimum set of features for one of the experiments that we performed.

In what follows we first describe the dataset that we used for our experiments. After that we give details of the experiments that we performed and discuss their results.

## 1 Dataset

Data that we use for our project comes from two sources: (1) IMDB dataset and (2) Google search frequencies.

**IMDB dataset** IMDB provides rich data about movies, and user ratings for those movies. For our project we selected 400 movies (200 bad with IMDB rating  $\leq 6$ , and 200 good movies with IMDB rating  $> 6$ ) using the pre-processed dataset<sup>1</sup> as well as lists of American movies released in a particular year posted on Wikipedia (we used lists of movies released in 2009, 2010, 2011). Such combination was needed since the above referred preprocessed dataset mostly contains information about the movies released before 2008, and does not cover more recent movies. For each movie in the combined dataset we used the API <http://www.omdbapi.com/> to obtain its IMDB data, which allowed us to collect IMDB data for 400 movies released in the interval from 2004 and 2011. We were not interested in movies released before 2004 as Google search statistics is not available for them. For each movie we recorded the following fields: (1) movie title, (2) movie director, (3) movie actor #1, (4) movie actor #2 (in the order in which actors are listed on the IMDB website), (5) release date.

**Google search frequencies** We tried two different approaches for movie prediction popularity: (1) combining Google Trends and Google AdWords Statistics and (2) using only Google Trends statistics. We now describe our data collection process for both of these experiments.

---

<sup>1</sup>The dataset is released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011) <http://ir.ii.uam.es/hetrec2011> at the 5th ACM Conference on Recommender Systems (RecSys 2011) <http://recsys.acm.org/2011>

**Google trends** provides a search popularity index (SPI) of any particular Google query in a specified time period and geographical region. This index is based on the query share, i.e. the total number of searches for a particular query divided by the total search volume seen in the specified geographical region and within the given time period. Before displaying final trends graph, Google performs the second round of normalization, where it divides each query share value by the highest value of query share within the specified time period. The output is obtained by multiplying this value by 100. Thus Google shows the normalized search statistics, and the normalization constant varies from query to query. Note that this fact does not allow us to compare the popularity of two different queries using the trends data only. We fix this problem using AdWords data as specified below (see section 1, Google AdWords).

**Usage of trends data in our experiments.** In both experiments for each movie we collected the trends data for the above specified fields, i.e. movies title, director, actor#1 and actor#2. The trends query interval was set in the following way (we used the movie release date field recorded):

1. from one month before the movie release date to current moment for first experiment;
2. from 8 months before the movie release date to 4 months after the movie release date for the second experiment.

Such query interval choice for the first experiment is motivated in section (2) and is dictated by the way the Google AdWords data is collected. Such interval length for the second experiment was chosen because we believe that data outside of this range is unlikely to contain any meaningful signal.

**Google AdWords** is the tool aimed at businesses to deliver ads to potential customers. It has a rich functionality for this, but we have used only the keyword tool provided by Google AdWords. The keyword tool returns the search volume for a particular keyword (or keyphrase) in a specified geographical region in the last 12 months. Thus, AdWords data allows us to recover the normalization constant lost in trends data, and compare query popularity. In particular, that is why we set the trends interval end moment to be the current moment for the first experiment (see section 2 for details of this calculation). Note that we are not guaranteed that statistics calculated for trends and AdWords uses the same search database. However, we believe that such approach correctly estimates the order of magnitude of the trends normalization constant.

Let us note that the major problem with using Google AdWords keyword tool is that we were not granted the developer key, since our intended usage was not in compliance with AdWords APIs terms and conditions. Thus, we were not able to automatically collect the AdWords data. However, to try this approach we collected the AdWords data for movie title, movie director, actor1 and actor2 for the set of 120 movies (60 good and 60 bad movies in our 400 movies dataset).

## 2 Methodology

**First Experiment** In this section we describe the way of defining features for experiment 1. First, due to the manual data collection process, we were able to collect data for 120 movies. Thus, to keep the ratio  $m \approx 20n$  (where as usual,  $m$  denotes the number of points in the training set, and  $n$  is the number of features) we limit the set of features to be the 4-dimensional vector that contains the total number of search volume of the movie title, director, actor1 and actor2 around the movie release date. We estimate this search volume in the time interval starting 1 month before the release date ending 4 months after the release date. In particular, we calculate the values of the features in the following way:

- Let  $A$  be the search volume provided by AdWords for a given keyword in the last 12 months in the US

- let  $t_1 =$  one month before the release date
- let  $t_2 =$  current moment (note that the length of the interval  $[t_1, t_2]$  is always more than 1 year, as we work with movies released before 2012)
- Let  $X_t$ , where  $t_1 < t < t_2$ , be the trends data for the same keyword in the US
- Calculate  $G = \sum_t X_t$ , where  $t$  ranges over the subintervals in the last 12 month
- Set  $N_\tau = A/G$ , calculate  $F = N_\tau \sum_t X_t$ , where  $t$  ranges over the subintervals around the release date, i.e. in the interval (-1 month, +4 month).

Note, that due to the very small dimension of the feature space we did not apply any feature selection or dimensionality reduction techniques in this experiment.

**Results** We approach the problem of predicting movie rating as a two-class classification supervised learning problem, i.e. is the problem of determining if movie rating is high or low. In particular, we assign all the movies with IMDB rating less than 6 to class 0, and all movies with rating  $\geq 6$  to class 1. In this experiment we compared the performance of the logistic classifier,  $\ell_1$  regularized SVM (with Gaussian kernel, i.e.  $\exp(|u - v|^2)$ ) and multilayer perceptron. Note that due to the small size of the dataset we used 5-fold cross validation to estimate the empirical error, where the dataset was divided randomly into 5 parts. Table 1 shows the algorithms performance:

Correctly classified	Overall:	High rated movies:	Low rated movies:
Logistic	49.07%	44%	54%
$\ell_1$ -SVM	54.63%	11.26%	98%
Multilayer perceptron	51.85%	69%	35%

Table 1: Distribution of movie ratings in the first experiment

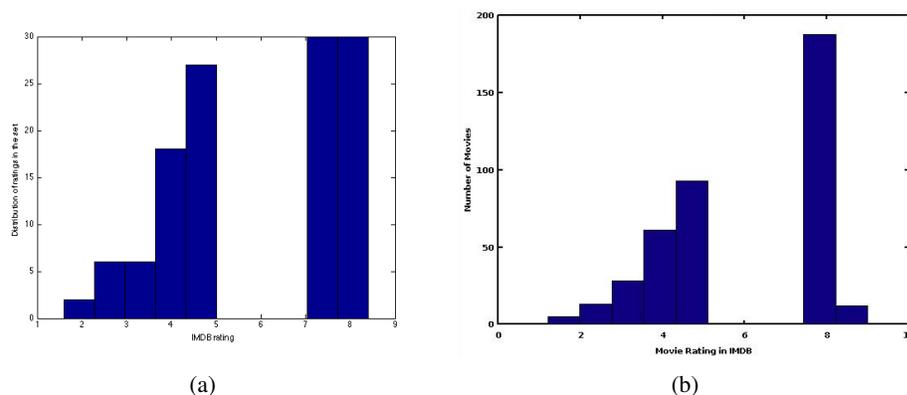


Figure 1: (a): Distribution of movie ratings in the dataset (experiment 1) (b): Distribution for experiment 2.

Fig. 1(a) shows the distribution of movie ratings in the generated dataset. Note that in our dataset 50% of the movies have high rank, and 50% have low rank, so the movie database is balanced.

**Discussion** First, as the above table shows all three classifiers perform very badly. In fact logistic regression classifier performs similarly to a fair coin toss: around 50% of high and low rated movies are classified correctly. On the other hand,  $\ell_1$  SVM appears to always classify the movies as a low rated. Finally, the multi layer perceptron

performs to a coin toss with one appearing with probability 0.65. Thus, surprisingly we essentially obtain no signal using this setup. We next describe our second experiment which gives nontrivial performance.

### 3 Second Experiment

**Defining features** In our second experiment we did not use the Google AdWords data, and relied only on the trends data. We briefly described the data collection process in section 1 (see Google Search Frequencies). Let us now describe it in more detail. Fig. 2(a) shows an example of time series data returned by Google trends.

Usually, the data had a sharp peak around the movie release date, and it was decaying around it. We would also like to note that the fast increase in popularity usually occurred one week before the release date, and that 4 months after the release the number of searches was very small. So, we concentrate on the data contained in the interval starting 1 month before the release and ending 4 months after the release, as we assume that this 5-month interval contains most information. We then further split these 5-month interval into the following segments: (1) one month before the release and (2) after the release (we first create 6 intervals 2 weeks each (see Fig. 2(a)), and we define the remaining part to be the 7th interval).

We define a feature as the area under the curve in each of these time segments. Note that since the search activity before the release is small, we do not split the the area under the curve before the release into smaller sub-segments and treat it as a single feature. Thus, we generate 8-feature vector from each trends query.

For each movie we request trends data for the following four fields: (1) title of the movie, (2) directory of the movie, (3) actor #1, (4) actor #2. Thus, we have  $4 \cdot 8 = 32$  features total.

**Experiment description** The following Fig. 1(b) shows the distribution of the movie ratings in 400-movie dataset. Note, that our dataset is balanced again, since it contains same number of high and low rated movies.

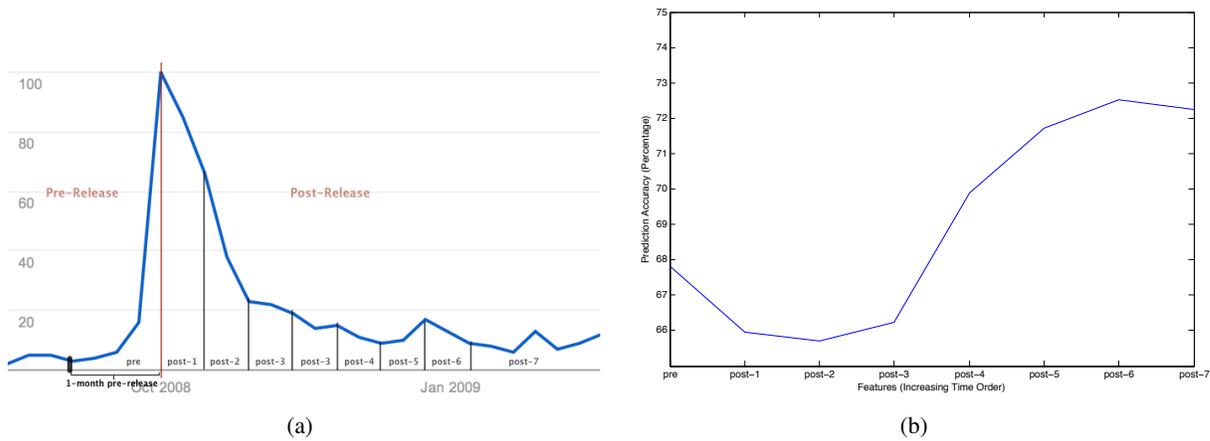


Figure 2: (a): Feature definition for the second experiment (b): Marginal prediction value

**Results** Table below shows the classification performance of the same set of algorithms as we used in our first experiment (in this case, we chose 8-fold cross validation):

**Discussion** As the above figure shows the same algorithms perform much better in this experiment. In particular,  $\ell_1$ -SVM correctly predicts in 72% of cases, and is consistent within both classes. However, logistic classifier and

Correctly classified	Overall:	High rated movies:	Low rated movies:
Logistic	64.13%	63.53%	64.64%
$\ell_1$ -SVM	72.25%	72.22%	72.27%
MLP (3 hidden layers)	68.32%	67.37%	69.23%

Table 2: Distribution of movie ratings in the second experiment

MLP perform a bit worse. We think that such behavior is explained by the presence of outliers in the dataset. In particular, we saw several movies that have very low rank, but that are very popular (e.g. the movie Justin Bieber: Never say never which IMDB rank is 1.6), i.e. these movies behave in terms of popularity closer to good movies, and thus trends data decays slowly.

## 4 Feature Selection

Next, we consider how  $\ell_1$  SVM performance changes when we use different set of features. Fig. 2(b) shows the prediction accuracy if we add features in the increasing time order, using more and more area under the curve in each run of the experiment. Thus, this figure shows the marginal discriminative value of using search activity in additional time period.

Note that adding first two post release features in fact decreases the performance of the classifiers. When we run the experiment with only pre feature (using only the area before the release), we get better result (67.80%) than when we use “pre”, “post-1” and “post-2” features (65.70%). Note that using pre, post-1 and post-2 features corresponds to analyzing search frequencies from one month before the release to 4 weeks after the release. When we use only post-1 and post-2 features (only 4 weeks of search activity after the release), we achieve 64.92% accuracy. Therefore, short term post-release search activities have less predictive power than those of pre-release and longer term post-release.

We also analyzed which movies characteristic (i.e. title, director, actor1 and actor2) has the highest impact on prediction accuracy. The results show that the search activities for the title, director, and the first actor of the movies have very close discriminative values, while those of the second actor have significantly less discriminative value. This is consistent with the results of the feature selection algorithms we have run.

We ran CFS Subset Evaluation with forward, backward and bi-directional search, and all have eliminated the features related to search activity for the second actor. The selected features give close prediction performance (SVM): 70.68%, compared with the result that we have seen with using full feature set.

## References

- [1] H. Choi and H. Varian. Predicting the present with google trends. 2011.
- [2] S. Goel, J. M. Hofman, S. Lahaie, D. M. Pennock, and D. J. Watts. Predicting consumer behavior with web search. *PNAS*, 2010.