Biased k-NN Similarity Content Based Prediction of Movie Tweets Popularity

Ladislav Peška and Peter Vojtáš

Faculty of Mathematics and Physics
Charles University in Prague
Malostranske namesti 25, Prague, Czech Republic
peska@ksi.mff.cuni.cz, vojtas@ksi.mff.cuni.cz

Abstract. In this paper we describe details of our approach to the RecSys Challenge 2014: User Engagement as Evaluation. The challenge was based on a dataset, which contains tweets that are generated when users rate movies on IMDb (using the iOS app in a smartphone). The challenge for participants is to rank such tweets by expected user interaction, which is expressed in terms of retweet and favorite counts.

During experiments we have tested several current off-the-shelf prediction techniques and proposed a variant of item biased k-NN algorithm, which better reflects user engagement and nature of the movie domain content-based attributes. Our final solution (placed in the third quartile of the challenge leader board) is an aggregation of several runs of this algorithm and some off-the-shelf predictors.

In the paper we will further describe dataset used, data filtration, algorithm details and settings as well as decisions made during the challenge and dead ends we explored.

Keywords: recommender systems, content based similarity, social network, semantic web and linked data, hybrid biased k-NN, ensemble learning, user engagement, RecSys Challenge 2014, SemWexMFF team, data structures for similarity search and indexing

1 INTRODUCTION

Recommending on the web is both an important commercial application and popular research topic. The amount of data on the web grows continuously and it is virtually impossible to process it directly by a human. Various tools ranging from keyword search engines to binary intra e-shop search or product aggregators were adopted to fight against information overload. Although such tools are definitely useful, they can be used only if the user is able to specify in detail what he/she wants. Recommender systems are complementary to this scenario as they are mostly focused on serendipity – showing surprisingly interesting items the user was not aware of and thus couldn't search for them by keywords.

Many recommender systems, algorithms or methods have been presented so far. Initially, the majority of research effort was spent on the collaborative systems and explicit user feedback. Although collaborative recommender systems are generally trusted to be more accurate, they suffer from three well known problems: cold start, new object and new user problem.

New user / object problem is a situation, where recommending algorithm is incapable of making relevant prediction because of insufficient feedback about current user / object. The cold start problem refers to a situation short after deployment of recommender system, where the system cannot provide any relevant recommendation, because of insufficient data generally.

Using attributes of objects and hence content based or hybrid recommender systems can speed up learning curve and reduce the cold start problem. Moreover, content-based recommender systems can compute similarity of a new object based on its features effectively eliminating the new object problem.

Our deep belief is that quality of data used for recommendation are often more important than the algorithm which processes them. In another words rather than designing a brand new algorithm we focus on enhancing our datasets and using state-of-the-art or slightly modified algorithms to improve predictions.

The task of 2014 RecSys Challenge[1] ([17], [20]) was to predict user engagement on Twitter for movie rating tweets automatically posted from IMDb[2] (from users, who connected their IMDb and Twitter accounts). The user engagement of each tweet was defined as a sum of retweets and favorites of this tweet. Other tweet data was also made available for use, especially user rating of the movie, statistics about the user, date and time when the tweet was posted and URL to the IMDb page with the movie (data are available at Github[3]).

The dataset covers the period from February 2013 to March 2014 and contains in total almost 213,000 tweets from 24,000 users about 15,000 movies. The dataset was divided into training, test and validation subsets based on the timestamp when the tweet was created. Evaluation of the task was based on nDCG metric considering top 10 tweets for each user.

The rest of the paper is organized as follows: review of some related work is in section 2. In section 3 we provide some insight on the task and how it affected our solution [14]. Section 4 describes which recommending algorithms were used and their results. Finally section 5 concludes the paper, describes lessons learned during the challenge and points to some future work.

2 RELATED WORK

Due to the space reasons, we can provide only a short review of the related work. For general information and introduction to the recommender systems, we suggest

[1] http://2014.recsyschallenge.com/

[2] http://www.imdb.com

[3] https://github.com/sidooms/MovieTweetings/tree/master/recsyschallenge2014

Recommender Systems Handbook [19]. Several state-of-the-art recommending algorithms was used in the experiments namely Factor Wise Matrix Factorization [1], Bi-Polar Slope One [7], Item-based k-NN [8], Decision trees, Support Vector Machines4 etc. Individual results of these methods can be found in Section 4. For the majority of the algorithms we use their implementation in RapidMiner Studio5, or its Recommender extension [9].

We would like to mention also our own previous work, which affected our approach: In [12] we first considered using external semantic content to enhance secondhand bookshop recommender systems. The paper corroborated improvement of success metrics while using DBPedia content and although the following experiments shown that content-based recommending algorithm can be substantially improved, we kept using the item-item similarity method described there. In the following work on the same domain [16] we experimented with Content-boosted Matrix Factorization (CBMF) [4], which outperformed methods from [12]. Similar approach was also used in ESWC RecSys Challenge 2014 [13], however CBMF suffered from too high time complexity with rising number of examples and content attributes which detracts its usability. On the other hand the challenge winning method by Risotski et al. [18] has shown that using relatively simple recommenders combined together may provide surprisingly good results.

Our work is also related to the area of linked data. An inspiration to our previous work was the research by Ostuni et al. [10], whose aim was to develop content-based recommender system for a movie domain based sole on (multiple) LOD datasets and A. Passant [11], who developed dbRec – the music recommender system based on DBPedia dataset. Their point of view is however slightly different as they aim to develop a recommender system based solely on the semantic web datasets, but in our work (both previous and contemporary) we need to integrate external knowledge into the already known structure of the domain, thus our recommending techniques are not based on graph structure of linked data, but we aim to convert LOD into attribute-value structure.

The area of recommending on social networks is currently well covered in the research. We would like to mention e.g. Hannon et al. [6] work on recommending inter-user relationships on twitter or Esparza et al. [5] work on categorizing tweets. Nonetheless due to the specific nature of the challenge task, most of the common social network based research is not applicable.

3 ANALYSIS OF THE TASK

This section aims on discussion and initial analysis of the challenge task, focusing mainly on design choices implied by the nature of the task.

The dataset provided by the challenge organizers consisted of user, tweet and item identification, timestamp when the tweet was scraped, user rating of the movie and all information available from the tweet API excerpt from the text of the tweet (see Fig-

[4] http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/
[5] http://www.rapidminer.com

ure 1). The textual information would be extremely important for prediction of user engagement in other datasets, but automatically generated tweets from IMDb contain only template text and thus are not much relevant. The same reason makes also tweet topic categorization (e.g. CatStream [5]) irrelevant. The tweet API contains e.g. date and time of the tweet posting and statistics about the user (number of friends, followers, tweets etc.).

The user engagement is generally low throughout the dataset. The average user engagement in the training set is 0.216, over 95% of the tweets have zero user engagement and almost 80% of users received zero engagement for all of their tweets. The situation seems to be similar to the number of purchases on an e-commerce site, where most of the users only browse items, but do not buy one. Our experiments on such domain [15] suggested using extended observation of user behavior and content of the items to improve recommendation. As the monitoring of user behavior is not possible in this scenario, we focused on using available content. Another interesting question is how to interpret zero user engagement in situations where no engagement was shown also in other tweets of the same user.

3.1 Content-based Movie Datasets, Collaborative vs. Content-based

Prior to the experiments, we have conducted a small survey of available movie datasets. The IMDb, DBPedia[6], and Freebase[7] were examined concluding that IMDb contains most of the relevant information available in the other two datasets. Due to 100% coverage of items (each tweet was based on single IMDb object) and availability of querying API[8] we decided to use sole IMDb for dataset enhancements. The movie features used in our solution can be distinguished into three classes:

- Attributes describing popularity (average rating, number of awards, IMDb metascore)
- Attributes describing widespread of the movie (number of ratings)
- Attributes describing content (movie name, release date, genre, country, language, director, actors)

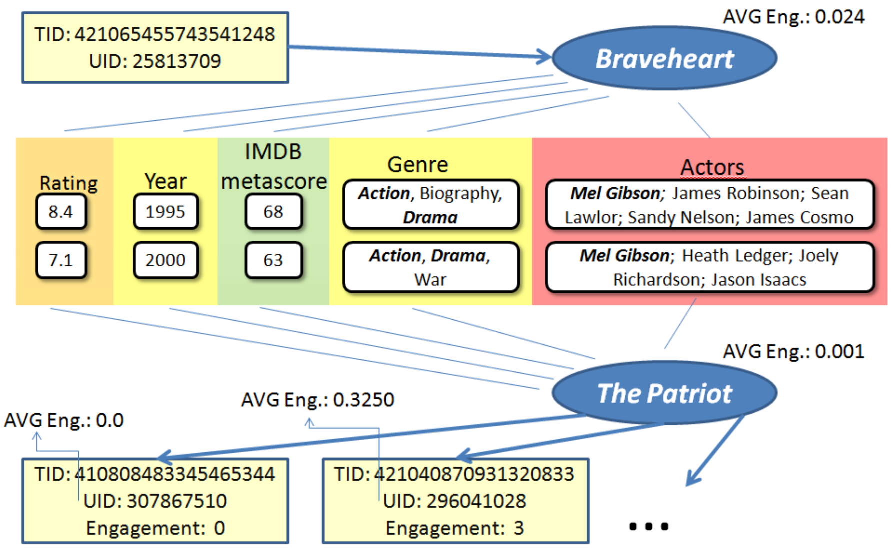
There is however some space left for further improvements especially by employing DBPedia features like dcterms:subject or widespread metrics like ingoing / outgoing links or number of Wikipedia language editions.

The test dataset contains large number of new movies unseen in the training data, so we expect that purely collaborative recommenders will not provide very good predictions. Another possible limitation is large number of zero user engagement. This caused problems to some algorithms (e.g. decision trees) as they almost constantly predicted zero for all tweets. The problem can be bypassed e.g. by filtering out (some) examples with zero engagement or by copying other tweets. The task is also not well

[6] http://www.dbpedia.org
[7] http://www.freebase.com
[8] http://www.omdbapi.com

suited for the purely content-based recommenders as there are new users in the test dataset and also for many users we have only a few tweets available.

Fig. 1. In this figure we illustrate two tweets in training data (evaluating The Patriot) and one tweet in testing data (evaluating Braveheart). Automatically generated tweets could look like "I rated The Patriot 9/10 http://www.imdb.com/title/tt0187393/#IMDb" and "I rated Braveheart 8/10 http://www.imdb.com/ title/tt0112573/#IMDb". We download additional data about the movie and we calculate average engagement for the user and also for the movie. The task is to predict engagement for tweet in testing data.



3.2 What Does User Engagement Depend on?

We are deeply convinced that crucial for any recommending task is to estimate on which variables the final success may depend. In the current case each tweet contains almost the same text except for the name of the movie and the user rating, so we do not expect that the tweet itself can affect user engagement.

Important variable determining user engagement is probably composition of user friends and followers. Unfortunately the dataset contains only total numbers of friends and followers for each user, not the variables describing them, but we can at least employ user bias defining average engagement for each user.

Another important component is, according to our assumption, features of the movie that the tweet refers to. The sole movie ID may not be enough as there are numerous new movies in the test set and some movies are not rated with enough users. Thus we need to define content-based similarity between movies under assumption that similar movies will be treated similarly.

Also the date when the tweet was posted may be interesting since the structure of friends or followers might change over time and also popularity of the movie may evolve, but we expect that relation of previous components should be stronger and so we did not pursue this direction and leave it for the future work. The same applies also for the dependence between user rating and user engagement.

4 RECOMMENDING ALGORITHMS

In our approach we worked with two main hypotheses:

- Engagement of similar movies should be similar.
- Engagement depends on neighborhood of the current user.

In order to define inter-movie similarity, we used IMDb querying API to generate content-based attributes. We also considered using DBPedia or Freebase, but IMDb contains most of the relevant information and furthermore offers guaranteed 100% item coverage. Three types of attributes were downloaded: attributes describing popularity (average rating, number of awards, IMDb metascore), attributes describing widespread of the movie (number of ratings), attributes describing content (movie name, release date, genre, country, language, director, actors).

The second hypothesis reflects our expectation that composition of user friends and followers would greatly affect observed engagement. The twitter API contains only aggregated information (total numbers of friends and followers for each user), so we decided to use simple user bias instead of machine learning over user's friends.

Table **1.** Results of the off-the-shelf algorithms. The best achieved result for each algorithm is shown.

Method	nDCG@10
Random predictions	*0.7482*
Bi-Polar Slope One [7]	0.7652
Slope One [7]	0.7557
Factor Wise Matrix Factorization [1]	0.7556
Item-Item K Nearest Neighbors [8]	0.7604
Decision Tree	0.7494
AdaBoost with Decision Stump	0.7505
Support Vector Machines (SVM)	**0.8057**

Prior to the design of our own recommending algorithm we evaluated several off-the-shelf algorithms using RapidMiner and its Recommender extension. As expected, results of both collaborative-filtering and standard machine learning algorithms were except for SVM rather unsatisfying. Table 1 contains the best achieved results for each algorithm over different settings. Some dataset transformations (e.g. omitting records with zero engagement from the training set, transformation of user engagement in training set etc.) were also examined, but they did not significantly improve the results.

4.1 Hybrid Biased k-NN

According to the assumptions and hypothesis formulated in Section 3, we decided to pursue especially content-based movie similarity. We implemented a variant of well known k-nearest-neighbors as our main individual recommender (see Algorithm 1), where similarity of tweets is determined as content-based similarity of the respective movies. The similarity is defined as average of attributes similarities. Attribute similarity is defined according to attribute type. Similarity of *numeric* attributes (*average rating*, *number of ratings*, *number of awards*, *IMDb metascore* and *release date*) is defined as their difference normalized by maximal allowed distance (1).

$$sim_{x,y,maxDist} = \max\left(0, \frac{maxDist - |x - y|}{maxDist}\right) \tag{1}$$

For string attributes (movie name) the similarity is defined as inverse of relative Levenshtein distance (2). This allows us to define as similar e.g. movie series.

$$sim_{x,y} = 1 - \left(\frac{levenshtein(x,y)}{\max(lenght(x),\ lenght(y))}\right) \tag{2}$$

Finally, similarity of set attributes (genre, country, director and actors) is defined as Jaccard similarity (3). Note that nominal attributes can be dealt as sets of size 1.

$$sim_{\mathbf{x,y}} = |(\mathbf{x} \cap \mathbf{y})| \,/\, |(\mathbf{x} \cup \mathbf{y})\,| \tag{3}$$

Differences between audiences of users will be considered in the form of user bias (average value of engagement per user). The bias of current user is not important, as the evaluation is on per user base, however the bias of other users should be considered within the k-NN algorithm.

Algorithm 1: Hybrid biased k-NN algorithm: for tweet tIDte from the test set, its movie mIDte and fixed k, the algorithm first computes similarities to other movies in training set and selects k most similar movies. Then for each tweet about a similar movie the predicted ranking $\hat{e}$ is increased according to similarity s, user engagement e and bias of the tweeting user. The bias of the current movie is added in the final $\hat{e}$ prediction too (see Figure 1).

```
function HybridBiasedKNN($tID_{te}$ ∊ $TestSet$, $k$){ $\hat{e}$ = 0; extract
$mID_{te}$ from tID$_{te}$, extract uID$_{te}$ , extract $mID_{te}$ content from IMDb
  /*compute similarity for all movies */
foreach($mID$ ∊ $TrainSet$){
    S[$mID$] = similarity($mID_{te}$, $mID$);      }
/*get k most similar movies */
$\bar{S}$ = getKMostSimilar($mID_{te}$,S,k);
/*get all tweets about movies in $\bar{S}$ */
foreach({uID, mID, e, s}:
{uID, mID, e} ∊ TrainSet && $\bar{S}$[mID]=s){ê += s * e / bias($uID_{te}$);}
ê = bias($mID_{te}$) + (ê / sum(s))
return $tID_{te}$, ê;                                   }
```

Several meta modeling techniques were used to derive final predictions based on hybrid k-NN and off-the-shelf algorithms predictions. We have experimented with stacking with random trees, linear regression in cross-validation like setting and also tried several variants of averaging selected predictions (omitting portion of highest and lowest predictions for each tweet).

4.2 Results and Discussion

Table 2 contains results of several variants of hybrid k-NN algorithm as well as best aggregated predictions (for the sake of clarity we show only a fraction of results expressing different aspects of the data). Generally spoken, the best performing individual recommender was SVM followed by several variants of hybrid k-NN. Almost all experimented settings of hybrid k-NN outperformed other standard machine learning methods (Table 1). Surprisingly, stacking based ensemble did not predict well, probably due to dependence of the results on user, which is hard to express with decision trees. Also linear regression did not improve results, but averaging results of selected algorithms provided a significant improvement over the best individual recommenders.

While evaluating Hybrid Biased k-NN we focused mainly on the utility of each attribute, using of user bias and also ways to combine results from multiple algorithm settings. Only a fraction of our results can be shown due to the space reasons. We can state that most of the attributes used as sole similarity measure provided good results (especially IMDb metascore, director, country and language) – see Table 2.

Table 2. Results (nDCG@10) of Hybrid biased k-NN algorithm using only single content-based attribute to compute similarity.

Avg rating	0.7918	Movie name	0.7947	Language	0.8005
Awards	0.7652	Date	0.7962	Director	0.8029
Metascore	**0.8057**	Genre	0.7919	Actors	0.7930
# of ratings	0.7964	Country	0.7984		

One of our research questions was which value to use as user engagement e. The experiments showed that if using directly sum of retweets and favorites, the algorithm is highly dependent on using user bias. Another option was to use rank of the tweet in the list of current user's tweets ordered by user engagement (rank of the tweet should better reflect considered success metric). Under this setting was algorithm less dependent on using bias, however overall results were slightly worse.

Another question was how to interpret if all tweets of a user have constantly zero user engagement. Omitting those users from the training set however resulted into the decrease of performance so we suppose that even these tweets carries some negative evidence. Comparing with off-the-shelf algorithms, almost all variants of Hybrid k-NN achieved better results.

The neighborhood size k between 50 and 100 provided good results. We also tried numerous variants of combining attribute similarities within the Hybrid k-NN algorithm (omitting some attributes, weighting schemas) and ensemble methods (stacking, linear regression), but so far the best results was achieved by simple average of single attribute predictions omitting single top and bottom result – see Table 3.

Table 3. Results of Hybrid biased k-NN algorithm. *No bias* stands for omitting user and item bias from the algorithm.

Method	nDCG
Hybrid k-nn (Metascore, Language, Director, Country, Date, # ratings)	0.7927
Hybrid k-nn(Metascore, Language, Director, Country, Date, # ratings),no bias	0.7792
Linear Regression (Metascore, Language, Director, Country, Date, # ratings)	0.7913
AVG (Metascore, Language, Director, Country, Date, # of ratings), omit best and worst prediction	**0.8134**

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented our solution to the RecSys Challenge 2014. After analysis of the task, available data and current prediction techniques, we proposed a variant of k-NN algorithm leveraging content-based similarity of movies. The algorithm performed comparably with the best examined prediction techniques and the best results were achieved after averaging results of multiple runs of hybrid k-NN and SVM. Our solution was placed ninth in the challenge leader board. Some of our ideas didn't work as we expected, namely using more advanced ensemble techniques and using ranks of the tweet instead of its user engagement resulted in worse predictions.

There are several directions of the future work. In our research so far we did not pursue temporal dependence at all which could also affect user engagement. The defined movie similarity should be also examined and tuned. We could also try to employ tweet similarity instead of movie similarity (we didn't so far for the sake of computation effectivity). Also other procedures to aggregate results from multiple algorithms should be examined. Last but not least enhancing current dataset with e.g. DBPedia popularity measures should be considered.

Concerning data structures for similarity search and indexing – the query object is usually multimodal ([2]). Our objects have simple attributes and metrics is easy to compute. Our query is initiated by the whole user's history, in contrast with [21], [3]. Moreover the metrics is dynamically changing because of bias is changing. It is a challenge to consider index structure for fast k-NN for online usage.

ACKNOWLEDGMENTS

The work was supported by the grant SVV-2014-260100, GAUK-126313 and P46. Our source code is available on http://www.ksi.mff.cuni.cz/~peska/hybrid_knn.zip.

6 REFERENCES

1. Bell, R.; Koren, Y. & Volinsky Ch.: Modeling relationships at multiple scales to improve accuracy of large recommender systems. *In KDD '07*, ACM, 2007, 95-104
2. Budíková P. Towards Large-Scale Multi-Modal Image Search, Doctoral thesis Masaryk University, 2013
3. Alan Eckhardt, Tomás Skopal, Peter Vojtás: On Fuzzy vs. Metric Similarity Search in Complex Databases. In FQAS 2009, Springer LNCS, Volume 5822, (2009) 64-75
4. Forbes, P. & Zhu, M. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. *In RecSys 2011, ACM,* 2011, 261-264
5. Esparza, S. G.; O'Mahony, M. P. & Smyth, B. CatStream: Categorizing Tweets for User Profiling and Stream Filtering. *In IUI 2013, ACM,* 2013, 25-36
6. Hannon, J.; Bennett, M. & Smyth, B. Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches. *In RecSys 2010, ACM,* 2010, 199-206
7. Lemire, D. & Maclachlan, A.: Slope One Predictors for Online Rating-Based Collaborative Filtering. *In SIAM Data Mining (SDM 2005)*
8. Linden, G.; Smith, B. & York, J.: Amazon.com recommendations: item-to-item collaborative filtering, *Internet Computing, IEEE*, 2003, 7, 76 - 80
9. Mihelčić, M., Antulov-Fantulin, N., Bošnjak, M., Šmuc, T., Extending RapidMiner with recommender systems algorithms, *In RCM 2012*, Budapest, Hungary, 2012
10. Ostuni, V. C.; Di Noia, T.; Di Sciascio, E. & Mirizzi, R. Top-N recommendations from implicit feedback leveraging linked open data, *In RecSys 2013, ACM,* 2013, 85-92
11. Passant, A. dbrec - Music Recommendations Using DBpedia *In ISWC 2010, Springer, LNCS,* 2010, 209-224
12. Peska, L.; Vojtas, P.: Enhancing Recommender Systems with Linked Open Data. *In FQAS 2013, Springer, LNCS* 8132, 2013, 483-494
13. Peska, L.; Vojtas, P.: Hybrid Recommending Exploiting Multiple DBPedia Language Editions, *In ESWC 2014 Linked Open Data-enabled Recommender Systems Challenge*, 2014
14. Peska L., Vojtas P. Hybrid Biased k-NN to Predict Movie Tweets Popularity, poster, http://2014.recsyschallenge.com/program/SemWexMFF_short_09-21.pdf
15. Peska, L. & Vojtás, P.: Recommending for Disloyal Customers with Low Consumption Rate. *In SOFSEM 2014, Springer, LNCS 8327,* 2014, 455-465
16. Peska, L.; Vojtas, P.: Using Linked Open Data to Improve Recommending on E-Commerce. *In SerSy Worlshop at RecSys 2013, Hong Kong*
17. RecSys Challenge 2014: User Engagement as Evaluation. Complete dataset. https://github.com/sidooms/ MovieTweetings/tree/master/recsyschallenge2014
18. Ristoski, P.; Mencia, E.L. & Paulheim, H.: A Hybrid Multi-Strategy Recommender System Using Linked Open Data, *In ESWC 2014*, 2014
19. Ricci F.; Rokach L.; Shapira B. & Kantor P.B., editors, Recommender Systems Handbook, Springer Science + Business Media, LLC 2011
20. Said A., Dooms S., Loni B., Tikk D. Proceedings of the 2014 Recommender Systems Challenge, http://dl.acm.org/citation.cfm?id=2668067
21. Skopal T., Bustos B.: On nonmetric similarity search problems in complex domains. ACM Comput. Surv. 43(4): 34 (2011)