Predicting Movie and TV Preferences from Facebook Profiles

Johan Lindell
Stanford University
johli295@stanford.edu

Anders Haponen
Stanford University
haponen@stanford.edu

December 14, 2012

1 Introduction

The project we selected was taken off the list of suggested projects on the course website. The project idea was presented by Head TA Andrew Maas in cooperation with Graham Darcey and Wayne Yurstin of Screaming Velocity Inc [1]. It is a project that uses data collected from Facebook and tries to map it to data found on IMDb to suggest a TV-show or movie that the algorithm thinks that the user will like. Due to using Facebook we will also have access to users likes. These can be used both in training purposes and can be held out during testing to approximate accuracy of our suggestions. We chose this project for several reasons.

Firstly, Facebook and IMDb are good pages to work with when it comes to machine learning. Not only are they incredibly popular, but they both provide relatively simple APIs to work with. This means that we have to spend less time working on how to obtain the data and can spend more time selecting features and improving algorithms. We also like the idea behind the project, having computer generated suggestions (such as purchase tips on Amazon or friend recommendations on Facebook) is something we face daily and it would be interesting to see for ourselves one of the ways it can be done. Finally we are both avid watchers of various TV-series, something that will make the project more fun to implement.

2 Goal

The purpose of our project is to implement an algorithm that will take a users Facebook data as input. Based on the input it will then suggest a TV-series or movie, where the data concerning the media will come from IMDb's database or the dataset provided to us at the start of the project. The goal is to get an algorithm who's suggestions relate closely to the actual Facebook likes as these are the closest approximation to the truth that we can obtain easily.

3 Methods

There are two major tasks within this project, one is gathering and parsing of data and the other one is using machine learning to come up with relevant suggestions. The first part is described in section 3.1. For recommending TV-series we used two different approaches, described in section 3.2 and section 3.3 respectively.

3.1 Facebook data

The first problem we faced was the issue of gathering data. Facebook prevents users from accessing friends lists of your friends when using their Graph API. This strictly limits users from gathering large amounts of data single-handedly. This means that it would be hard for us get enough data by ourselves. Luckily we received tremendous help from Graham Darcay. He created a website that with the permission of the visitor collects the users and his friends' Facebook data [2]. Posting the link to this website on Piazza gave us a fair amount of data, at the moment of writhing this we have access to an XML-file with 16k users. We created a python script to read the XML-data and output a matrix indicating which series each user liked as well as other information such as gender, locale and other likes.

3.2 TV-series correlation approach

Our first approach to get an algorithm up and running was to use only the existing TV-series likes from each user. With that data we tried to come up with an algorithm that would capture the correlation between different TV-series, similar to Naive Bayes with Laplace smoothing [3]. The first step was training a large matrix ϕ, which elements would contain the probability that you like series i, given that you like series j. The i, j:th element of this matrix is shown in equation 1:

$$\phi_{i,j} = \frac{\sum_{k=1}^{m} 1\{x_i^k = 1 \wedge x_j^k = 1\} + 1}{\sum_{k=1}^{m} 1\{x_j^k = 1\} + N} \tag{1}$$

Where $x_i^k = 1$ means that user k liked series i, N is the total number of TV-series liked by all users and m is the total number of users.

Once this matrix has been trained a measure of the probability that a user likes the series l could be calculated using the users likes as follows:

$$p(l) \propto \prod_{\{\forall k | x_k = 1\}} \phi_{l,k}$$

Due to the low values of ϕ the following measure is more feasible:

$$log(p(l)) \propto \sum_{k=1}^{N} 1\{x_k = 1\} log(\phi_{l,k}) \tag{2}$$

To improve our performance we tried weighting these probabilities with other probabilities given other features, such as gender and locale. These probabilities were estimated similarly to the correlation matrix, e.g. for calculating the probability that a male liked a particular series we simply calculated the fraction between the number of likes from males and the total number of likes that series had. The locale information we had access to in the Facebook data was location and language settings.

3.3 User and series clustering approach

The next step we took was to do keyword analysis on the series. We did this in two parts. First we used the more concise descriptions of shows provided by Graham and Wayne. We then downloaded the keywords database from IMDb. We proceeded to write a Python script that went through the database extracting the keywords for each show and reformatting. After this we checked this data against the shows we obtained though our Facebook data to obtain keywords for more of them. With this we could run K-means on all the shows, using the keyword as features, to group the shows together in clusters. The reasoning behind this was to determine which cluster the user was most in favor of and recommend a TV-show from that cluster. To determine which cluster a user belongs to we averaged the features of the users TV-series likes and found the nearest cluster center retrieved form K-means. The features we then used as input to our algorithm was user locale, gender as well as a tokenization of all words retrieved from the facebook profile. The algorithm was then trained and tested using a SVM.

4 Evaluation metric

One difficulty of this project is measuring the quality of your recommendations. The reason for this is that it is difficult to exactly match the likes of a user when limiting yourself to a small number of recommendations. One approach would be to cluster all TV-shows (e.g. by genre) and if the recommendations are within the same cluster as the actual preferences of the evaluated person it would be counted as a successful labeling. Another approach to evaluating performance is to compare the algorithm against always recommending the most popular TV-shows, e.g. by recommend a larger number of TV-shows (~ 10) and count how many of those the person actually liked. After doing this for each test user the fraction of correct likes generated by the algorithm and the most popular shows will indicate the performance of the algorithm.

5 Result

5.1 TV-series correlation approach

Using the evaluation metric described in chapter 4 we got approximately 12% improvement compared to just recommending the most popular series. Table 1 lists the testing error when weighting with different additional features.

Additional features	Improvement from recommending popular series
None	12.01 %
Gender	12.80 %
Locale (location and language)	11.57 %
Gender and locale	12.09 %

Table 1: Improvements in recommendations

5.2 User and series clustering approach

Using this method, clustering the series into ten clusters, we could predict facebook users cluster belongings with accuracy only slightly above the the fraction of occurrence of the most popular cluster. Using these cluster belongings to recommend series turned out to perform worse than just recommending the most popular series.

6 Discussion

Using a simple approach we managed to achieve approximately 12% improvement when recommending series in comparison to just recommending the most popular series. These relatively poor results were caused by many factors, some of which are described in the following subchapter. Figure 1 shows a learning curve for the correlation approach, which says quite a few things about the algorithm. Since both the training curve and the test curve seems to have converged when training on all our data we would not have much use for more data of the type we had. It also seems like the two curves will not converge to the same performance, indicating that our model is slightly over fitting to our training data.

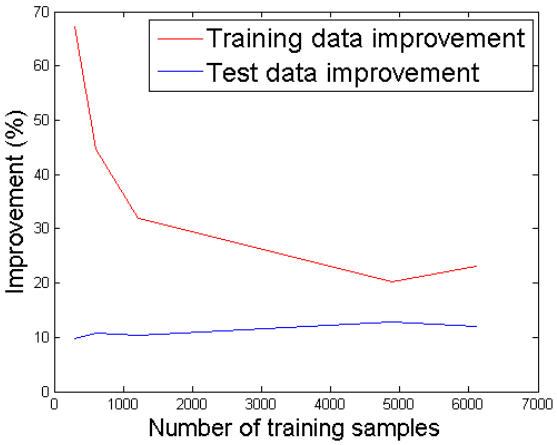

Figure 1: Learning curve.

6.1 Problems

6.1.1 Sparse data

A surprisingly high degree of Facebook profiles where just empty shells containing no data at all. This meant that a large portion of our data set added nothing but parsing time to our algorithm. There were also a large amount of low data profiles, containing only the locale and gender of a user. As we couldn't find good correlation between these metrics and TV-shows (see next subchapter) this further reduced the amount of useful data. Even the profiles with the highest amount of data were quite sparse, as we did not have access to for example their friends or their comments.

6.1.2 Low correlation

After experimenting with combinations of features in the TV-series correlation approach we saw that for the data we had there was no clear improvement in the result after adding either gender, location or a combination of both to the algorithm. In fact the only thing we could see improving the result beyond the simple approach of just recommending the most popular TV-show was taking the other likes of TV-shows into consideration. It is possible that this could be due to the sparsity of the data and that we don't have enough meaningful data for the correlations to appear.

6.1.3 Facebook pages naming convention

In matching the likes from the Facebook data to the tokens from the IMDb data we ran into the problem of the inconsistent naming convention of the often fan made pages on Facebook. Partial names, forgotten punctuation or abbreviations create problems in connecting the pages with their corresponding IMDb tokens. We tried to minimize

the damage with some basic string conversions, going all lower case, removing non alphabetic chars and such but there could still be a significant alteration of the results if a very popular TV-show has a poorly named Facebook page.

6.1.4 Execution time vs data set

When working with the TV-show correlation approach which gave us the best result we ran in to the problem of long execution times. Training our correlation matrix at the largest data set took upwards of 70 minutes. Naturally it is not supposed to be a process that is repeated too many times but it can still be frustrating when you wish to try small changes. As shown in figure 1 we are not reaching a convergence of training and test data at a much earlier point of data size so reducing the data set is not a viable solution. This can likely again be attributed to the sparsity of the data, a higher percentage of the profiles having likes and a higher average of likes per profile would allow us to reduce the size of the training set while retaining performance. Likewise a higher correlation with other parameters than Facebook likes could also allow us to draw more precise conclusions from a smaller data set.

6.2 Future improvements

The following subchapters discuss some improvements that could help performance and development.

6.2.1 Collecting other data

The biggest problem faced when trying to suggest series was the sparsity of each individual facebook profile. The additional information we believe would have helped the most would be the profiles of all the users friends, the users comments as well as age. This would allow better categorization of each user which could lead to better suggestions.

6.2.2 More refined data

One way of improving performance without slowing down training time would be to attempt to recruit quality data from the Facebook pages of TV-series enthusiasts. These would in all likelihood have a much higher average of TV-series likes making it easier to see correlations between shows. A possible argument against this would be that this is tainting the algorithm with a user base that is not representative of the end user (who is just an average Facebook user and not an above average TV-series viewer) which would cause the recommendations to become poor. A counter point can be made in that recommendations based on people who have a large experience with TV-shows and can draw better comparisons is actually worth more when suggesting new shows to an average user due to the nature of their expertise.

6.2.3 Better linking

If we could improve the linking between Facebook pages and the corresponding IMDb title we could get a very cheap performance boost with regards to execution time. This could of course be done with better string comparisons, there are really advanced algorithms for detecting similarities between two strings and we could continue doing this to an arbitrarily advanced level. A cheaper and quicker variant could just be to take a look at the X% most popular Facebook pages and see if we have any of them being uncounted for due to poor naming. This way we could also account for duplicates which in the current implementation can cause some TV-shows to be quite underrepresented if both pages cannot be accurately accounted for. Considering the amount of TV-shows we've seen from our data set so far it doesn't seem impossible to make manually ensure that the most popular series are correctly parsed.

7 References

References

[1] Screaming Velocity, Inc. Website: http://screamingvelocity.com/

[2] Face profile scraper. Online: http://screamingvelocity.com/Stanford-CS-229/index.html

[3] CS229 course lecture notes two. Online: http://cs229.stanford.edu/notes/cs229-notes2.pdf