A Facebook Profile-Based TV Recommender System

Jeff David
Applied Materials
jdavid@stanford.edu

Samir Bajaj
Apple, Inc.
samirb@apple.com

Cherif Jazra
Apple, Inc.
jazracherif@gmail.com

Abstract

We implement and evaluate several algorithms in the context of developing a recommender system based on data gathered from Facebook user profiles. In particular, we look at a Matrix Factorization technique (SVD), a Clustering algorithm (K-Means), two Collaborative Filtering algorithms, a Content-Filtering approach, Latent Semantic Analysis (LSA), Link Prediction, and Naïve Bayes, and compare their performance in terms of standard measures. The algorithms draw from principles and techniques in Machine Learning, Natural Language Processing, Information Retrieval, as well as Graph Theory.

1 Introduction

Recommendation systems are one of the most prominent applications of Machine Learning, and part of everyday life. They empower users to sift through enormous amounts of data and make informed choices. The field of recommender systems has seen a lot of innovation, and research is actively moving in the direction of leveraging social content. In that spirit, our project centers around building a recommendation system for TV shows based on data collected from Facebook profiles of several users.

We start with a brief description of the data set used in the development of the system and the metric we used to measure performance, followed by each of the various approaches and algorithms we implemented.

2 Data Set

The data set used in training and testing the recommendation system was a collection of 11,710 anonymized Facebook profiles contributed by volunteers in the CS229 class, as well as by our friends and family members. Of these, 5,372 users had expressed a liking for at least one TV show, and 4,413 users had liked two or more TV shows in their profiles. There are 5,892 different TV shows across all profiles in the data set. Right away, we can see that the user-TV show matrix is sparse; the ramifications of this fact will be reflected in our findings.

Additionally, Facebook profile data has a unary rating system for movies and TV shows—you either like something, or you don't express any opinion about it. So we used a 1 to indicate a liking for a show, and a 0 (or -1) to denote absence of any association.

3 Evaluation Metric

Traditionally, recommender systems have been evaluated on the basis of RMSE, the Root Mean Squared Error. This is also the metric that was chosen for the famous Netflix Prize. However, this works well in a system where users rate items (TV shows, movies, books, etc.) in a certain discrete range, e.g., on a scale from one to five stars, à la Amazon.com.

In the case of Facebook profile data, ratings are unary: either a user likes an item (thereby making a positive association), or not—and in fact, the absence of a 'like' can be interpreted either as a dislike or ignorance about the item. We therefore chose to use the Precision–Recall–F1 framework instead, because the unary rating scale makes the task of recommending TV shows more like a classification problem. We define *Precision* and *Recall* as follows, where r_i denotes a TV show recommended to the user and L denotes the set of TV shows liked by the user.

$$Precision = \frac{\sum_i \mathbf{1}\{r_i : r_i \in L\}}{\sum_i \mathbf{1}\{r_i\}}$$

$$Recall = \frac{\sum_i \mathbf{1}\{r_i : r_i \in L\}}{|L|}$$

4 Singular Value Decomposition

SVD is a matrix factorization technique that attempts to reduce the ratings space and identify some small number k of "topics" so that the user preferences can be expressed as a combination of the user's interest in a topic and the degree to which the topic is relevant. This is captured succinctly by the following equation:

$$M = U \, \Sigma \, V^T$$

In terms of the data set, $M \in \mathbb{R}^{11710 \times 5892}$, and we chose to factor out $k = 10$ topics—this value of k was chosen based on examination of a sample of the Facebook profile data—so $\Sigma \in \mathbb{R}^{10 \times 10}$.

Testing was done by randomly selecting 30% of the users from the population that had liked at least two TV shows, and then removing half [1] their ratings—i.e., setting to 0—again, in a random fashion. The matrix was then factorized and the predicted ratings were computed by multiplying the user's (modified) vector with ΣV^T and selecting the top positive entries. The result was compared with the original ratings to evaluate the performance.

5 K-Means Clustering

Our next approach was running the K-Means algorithm to identify clusters of similar users in the data. We followed an evaluation procedure similar to the one used in the previous experiment, randomly selecting 30% of the users from the population that had liked at least two TV shows, and removed half their ratings. Using the standard iterative implementation of the algorithm, we were able to partition the data into eight clusters.

Recommendations made to a test user were those that others in the same cluster had collectively liked the most.

6 Collaborative Filtering

The theme of this category of algorithms revolves around establishing similarities between different users based on the TV shows listed in their Facebook profiles, and between the items (the TV shows) based on user interest. This allowed us to recommend TV shows to users on the basis of the viewing preferences of other similar users. Likewise, using an item-based similarity, we recommend shows based on their preferences for certain other shows.

6.1 User-Based Collaborative Filtering

In this implementation[2], we used the classic *kNN* algorithm to define a neighborhood of users similar to the user who will be recommended TV shows of interest. The neighborhood is defined on the basis of the **cosine similarity** measure, which is defined as follows:

$$cos(\overrightarrow{u_1}, \overrightarrow{u_2}) = \frac{\overrightarrow{u_1} \cdot \overrightarrow{u_2}}{\|\overrightarrow{u_1}\| \ \ \|\overrightarrow{u_2}\|}$$

The vectors $\overrightarrow{u_1}$ and $\overrightarrow{u_2}$ represent the user's likes and dislikes in the unary-based system. A higher value of similarity indicates that the two users are closer in their tastes for TV shows.

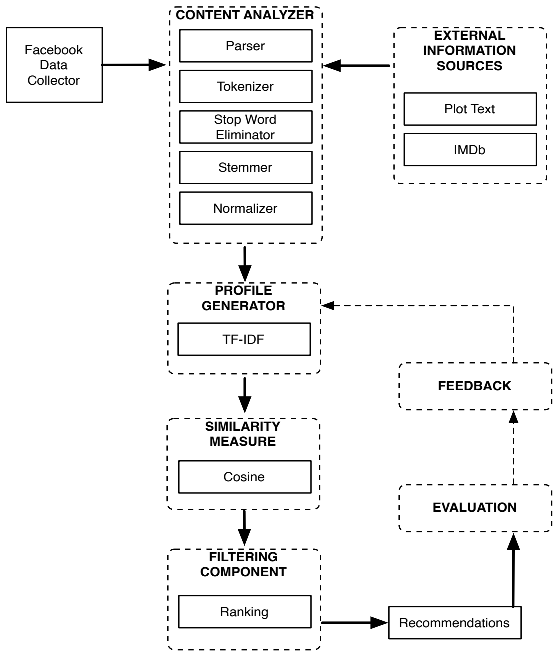
6.2 Item-Based Collaborative Filtering

This algorithm is similar in spirit to the User-based version, with the users and the items (TV shows) switching roles: Rather than using similarities between users' likes to predict preferences, item-based collaborative filtering uses similarities between rating patterns of different TV shows. If two TV shows have the same set of users like or dislike them, then they are similar; users are expected to have similar preferences for similar items. Our implementation used the same cosine similarity measure, and much of the same code to evaluate this approach.

We performed 10-fold cross-validation on each of the above approaches in order to evaluate performance.

7 Content-Based Filtering

Having examined collaborative filtering techniques to build a recommender system, we next turned to content-based methods, incorporating techniques from Information Retrieval (IR) as well as Natural Language Processing (NLP).



As a first step, the following transformations were applied to all input data—the Facebook profiles as well as the metadata on TV shows—in order to be able to treat all data uniformly:

1. **Parsing and Tokenization**: All text was split into tokens with whitespace as the delimiting set of characters. Further, punctuation and other non-alphabetic characters were dropped.

[1]Admittedly, withholding less than half the ratings would deliver better precision and recall, so the decision was rather arbitrary.
[2]This part of the project was coded in Python. For the complete source code, see [1].

2. **Stop-Word Elimination**: A set of 129 most common English words were used as the stopword dictionary; all occurrences of these words were eliminated.
3. **Stemmer**: A standard Porter stemmer was applied.
4. **Normalizer**: All text was converted to lowercase as part of the preprocessing task, and encoded as UTF-8.

Subsequent to the aforementioned preprocessing, user profiles were generated based on their Facebook data as well as the TV programming metadata available to us. Next, **tf-idf** scores were computed for the various features and **cosine similarity** was used in the **ranking** function to compute users most similar to a test user. Recommendations were based on ratings of users in the test subject's neighborhood.

In order to generate accurate results in this setup, we used only those users who had liked at least one show for which we had some metadata, collected from an external source like Screaming Velocity [7] or IMDb [8]. A 70–30 split of this reduced set of users was used to carry out cross-validation. The results surpassed those we had seen thus far.

8 Latent Semantic Analysis

The user–feature matrix generated as part of developing the Content-Based Filtering system was next used to investigate LSA by finding a low-rank approximation to the large feature space. The user–feature matrix $M \in \mathbb{R}^{11710\times 26263}$ was factorized using SVD to $U\Sigma V^T$, with $\Sigma \in \mathbb{R}^{10\times 10}$. The test users' neighborhoods were subsequently computed in this semantic subspace and recommendations were generated accordingly.

Note that the matrix being factorized in LSA is the user–feature matrix, which is different from the one used in the very first approach where every user was represented as a vector in the (also very high-dimensional) space defined by the various TV shows, but without any Facebook metadata.

9 Link Prediction

From the perspective of Graph Theory, we can look at the set of users U, TV shows T, and the associations between them, L, as a bipartite graph $G = (U, T, L)$ where $U \bigcap T$ is empty, and all edges $e \in L$ connect vertices in U with those in T. The edges are unweighted and the graph represents the complete information of the input data. We can now apply graph algorithms to gain insight into this network. Concretely, we transform the problem to that of link prediction, where a recommendation is a link that is likely to appear as the network evolves over time.

There are several linkage measures that can be employed to predict connections; we used Dijkstra's shortest path algorithm on the bipartite graph, which, for this analysis, includes users as well as TV shows, and hence has dimensions $\mathbb{R}^{17602\times 17602}$. Recommended shows were ranked by distance; those that were closer to a user appeared at the top of the list. This algorithm performed fairly well in comparison to others that also operated only in the domain of users and TV shows.

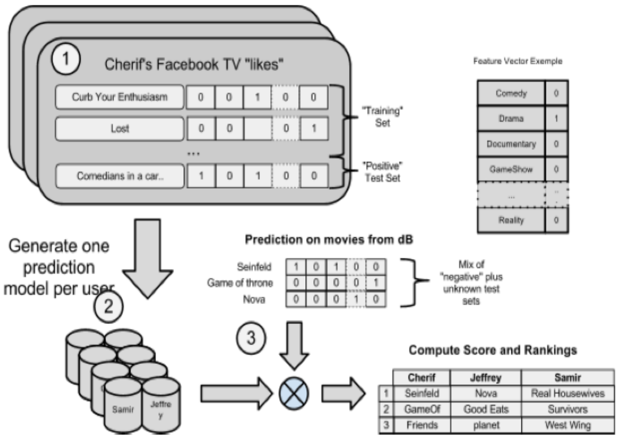
10 Naïve Bayes I

The first method aims at learning what genre of TV shows each user likes based on attributes such as comedy, drama, reality, documentary etc. Facebook profiles with more than 5 liked TV shows that are listed in our database are considered.

1. Training phase: We determine, for each user, the probability distribution of each attribute given a TV show is liked. Each user has his own separate prediction model. The list of liked shows is taken from Facebook profile and is divided into a Training set (70%) and a Positive test set (30%). An additional Negative set of the same size as the positive set is used to test TV shows for negative cases since negative true labels from the users are not available. The Negative set is taken from the overall TV database and doesnt intersect with the positive set.
2. Prediction phase: After training our user models, we rank all TV shows from our database by computing a score for each show. The score is the sum of probabilities of the users attribute (denoted by a below) distribution for only the attributes that are found in each show.

   $$Score(user\ i, TV show\ j) = \sum_a [p_i(a) * 1\{a \in TV show\ j\}]$$

   k-Cross Validation is performed by repeating the training/testing phases and shuffling the 80/20 ratio of the training and positive set in order to better gauge how the learning algorithm is performing.

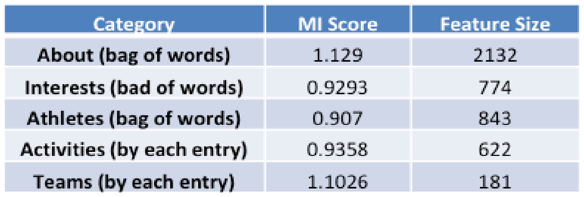


If the score of a TV show from the positive set is in the top 20 scores, it is considered a good recommendation and thus a successful prediction (since the show belongs to the users Facebook tv list). If the score of a TV show from the negative set is outside the top 20 scores, it is considered a successful (not liked) prediction since we assume it is not liked.

11 Naïve Bayes II

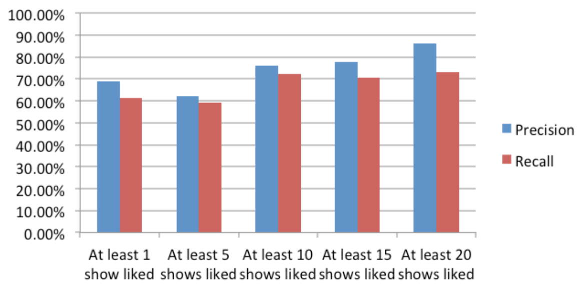
The following data preparation and implementation steps were taken for this approach to the problem:

1. Two types of dictionaries were created for each of the 12 features. The first type of dictionary was a bag-of-words approach (this type of dictionary is more appropriate for the about feature). The second type of dictionary tokenized each feature entry found (more appropriate for the teams feature).
2. The label for the classification models was TV shows. A TV show dictionary was created by compiling a list of all TV shows that were liked in all Facebook samples. The top 1000 TV shows were used for initial data processing, and the top 505 TV shows were used for the final classification model (505 TV shows, including N/A, occurred 10 or more times in all Facebook samples).
3. Mutual Information was performed to determine the features with the highest relevance. Based on these results, 5 features were selected to use for classification. Each feature dictionary was truncated to include only tokens/words that occurred 5 or more times in all samples combined. A table showing the selected features, MI score, and feature size for each of the feature categories is given below:



Category	MI Score	Feature Size
About (bag of words)	1.129	2132
Interests (bad of words)	0.9293	774
Athletes (bag of words)	0.907	843
Activities (by each entry)	0.9358	622
Teams (by each entry)	1.1026	181

4. For each Facebook profile sample, a record was created for each TV show liked in that Facebook sample. The label for each of those records was the liked TV show. So if a Facebook user liked 10 TV shows, 10 records were created for that sample. The binary feature vector for each record was of size 4552, which identified which features were present in the record. 31166 records were generated. Thus the feature matrix was 31166x4552, with a corresponding label matrix of 31166x1.
5. Using this feature and label matrix, SVM was implemented using the publicly available LibSVM [5]. Results were poor, 3% accuracy. A closer inspection of the recommendation vector revealed that the most popular TV show was always recommended. A grid approach for trying different values of C (Cost) and gamma was attempted, as explained in the LibSVM guide. Prediction accuracy did not improve. LibLinear was also implemented but this did not work as well.
6. Naïve Bayes was implemented with success. Various parameters were tried with Matlabs [6] built-in Naïve Bayes function. It was found that multinomial distribution mn worked the best. A 70%/30% partition was used for training/testing the dataset. To weight it fairly, the number of TV shows recommended for each profile was equivalent to the number of TV shows liked in each Profile sample. This is important when calculating precision and recall for this case, as the precision when calculated like this is found to increase as we only consider cases where a minimum number of likes are observed (e.g. 20 likes). Results for this implementation were as follows: Precision = 43.82% and Recall = 34.49%.
7. Intuition suggests that if a Facebook profile is sparse, then it becomes difficult to make a TV show recommendation regardless of algorithm robustness. To quantify this intuition, the best trained classification model - Naïve Bayes multinomial distribution - was tested on data that did not contain any N/As in the selected features (About, Interests, Athletes, Activities, Teams). There were a total of 6662 records out of 31166 that contained non-N/A values for all of these features. The data was again partitioned into 70% training and 30% testing. Results improved significantly to Precision = 69.05%, and Recall = 61.39%. 1392 correct predictions were made out of 2016 TV show likes for Facebook profiles that contained no N/A values for any of the 5 features. It is worth noting that Facebook users that had all five of the features completed averaged 9.8 TV show likes each. This data validates the intuition that if a Facebook profile contains lots of information in the relevant features, then the probability of making a correct TV show recommendation increases significantly. On the other end of the spectrum, if a user has no profile information, then it is virtually impossible to make a TV show recommendation. Future work might include determining the probability of a correct TV show recommendation based on the sparsity of the data in a Facebook profile.
8. We now analyze results when only a minimum number of likes are reported in the Facebook profile. Results are shown below.



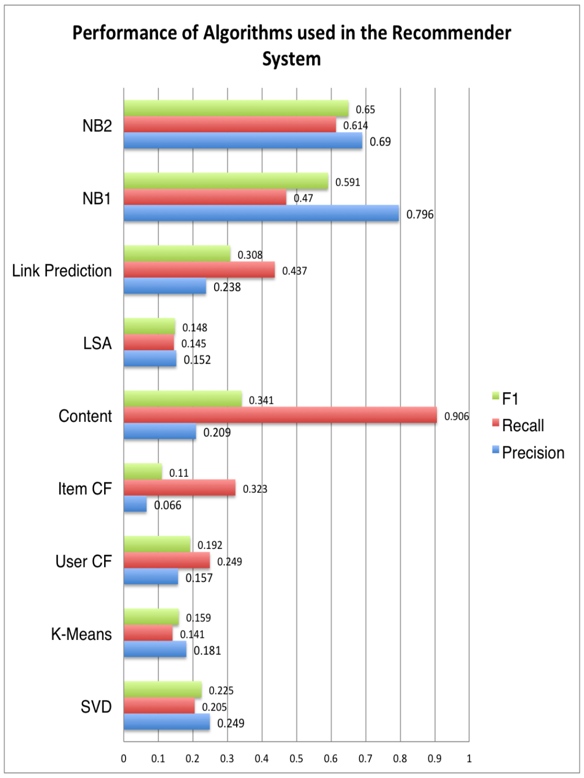
It can be seen that precision increases when the minimum number of likes increases. This tells us that our Naïve Bayes recommender is very good at matching the top 20 likes with recommendations, but may not necessarily be in the same sequence. For example, if only one TV show is recommended, in order for precision to count that case as true, that TV show must be the only one mentioned (and thus also the favorite) TV show of the user. Even if that TV show was the second favorite TV show, that recommendation will be counted as false. When considering 20 TV shows per user, the criteria becomes less stringent and the % of true matches of recommendations to likes goes up significantly.

12 Summary

We covered a lot of territory in the course of working on this project, and gained useful insight in the process of implementing the different algorithms.

First, a real recommender system is a complex structure, and takes a lot of effort to build and fine-tune. Even for a student research project like the one we built, it took an immense amount of labor to collect, process, filter, and organize the data before we could run any algorithms. Second, there is real value in metadata, if it is available. The algorithms that used external data and attributes from a user's Facebook profile performed consistently better than those that didn't avail any metadata. In discussions with engineers who have built large-scale commercial recommendation engines, we learned that production systems typically use a blend of several dozen algorithms, combining the best of what each one has to offer.

The following table lists the precision/recall results for the various algorithms we implemented.



We also spent a substantial amount of time investigating a solution using an SVM-based classifier. The idea was to treat the recommendation of a TV show to a user as a classification problem because of its similarity to the Facebook 'like' feature: we predict $+1$ for a TV show that we think the user would like, and -1 otherwise. Unfortunately, the results from this approach were poor, indicating that it wasn't a viable technique.

13 Acknowledgments

We thank Andrew Maas for his help in getting started, and for guidance along the way.

We would also like to acknowledge the support and data provided by Graham Darcey and Wayne Yurtin of Screaming Velocity, Inc., for our project.

14 References

1. Bajaj, Samir, Source Code available at GitHub: `samirbajaj/cs229-project`
2. Lops, P., de Gemmis, M., & Semeraro, G. (2011) Content-based Recommender Systems: State of the Art and Trends, In F. Ricci et al. (eds.), *Recommender Systems Handbook.* Springer Science+Business Media.
3. Ng, Andrew, *CS229 Lecture Notes*, 2012, Stanford University.
4. Turney, Peter D. & Pantel, Patrick (2010) From Frequency to Meaning: Vector Space Models of Semantics. In *Journal of Artificial Intelligence Research 37*, pp. 141-188.
5. Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines.* *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm
6. http://www.mathworks.com/help/stats/naivebayesclass.html
7. Screaming Velocity, Inc., http://www.screamingvelocity.com
8. IMDb, The Internet Movie Database, http://www.imdb.com