Creating TV/Movie Recommendations from Facebook Profile Information

Alejandro Ayala-Hurtado, Yeskendir Kassenov, Nick Yannacone

Motivation

While digital advertising has been pervasive for years, efforts to make it relevant, interesting, and convincing to sites' users are still ongoing. The best ads are those that a user finds compelling enough to take into consideration, rather than merely scroll by -- a reminder of a concert she'd meant to go to but forgotten about, or a nostalgic TV show from childhood. One typical approach to solving the problem of ad relevancy is to use a recommendation system -- broadly, a learning algorithm which decides what items for sale a user is most likely to find interesting, based on some combination of that user's past behavior, other similar users' preferences, and similarities between the sold items themselves. Most of these approaches take into account a user's existing preferences when making new suggestions. For example, the most visible recommendation system on Amazon's website is directly entitled "Customers Who Viewed This Item Also Viewed," directly relating a user's browsing history to other interesting items based on the tastes of other users.

For this project, however, we wanted to build a recommendation system without the aid of a user's existing preferences, at least not in the areas we were hoping to recommend. More specifically, we aimed to provide relevant recommendations for popular TV shows and movies based on other information; in our case we used gender, first language, and other cultural preferences (e.g. sports, music, and books) as this "other information." Our approach could be useful, for example, for a site which specialized in selling music, but also wanted to deliver relevant advertisements from film companies. And the first time a user visits a website, demographic data, like the location of the visiting IP address, is all the recommender has available to deliver high-quality recommendations.

Data Collection

Since our project has a corporate sponsor, Graham Darcey of Screaming Velocity, Inc., and since many other groups in the class are working on this project, we were able to acquire a long list of user interest data scraped from the Facebook profiles of friends of the members of every project group, using a Facebook Graph API scraper written by Graham. The resulting data set includes a list of each user's activities and interests, as reported by the pages they "liked," as well as each user's "locale" (i.e. language) and gender. Unfortunately, it does not include age or hometown, presumably because that information would have made the data set insufficiently anonymized. Nevertheless, we had a lot to work with.

We also received a mapping of popular TV shows and movies to their genres, also courtesy of Graham. With this information in hand, we experimented with a few different methods Naive Bayes, PCA, with Logistic Regression run on the resulting features, and K-Means Clustering for matching users' non-movie/TV interests with for the most part the *genres* of TV shows and movies, rather than the TV shows and movies themselves. We did attempt to use the users' movie and TV show interests to predict specific movies and TV shows but as we describe later it was not very successful.

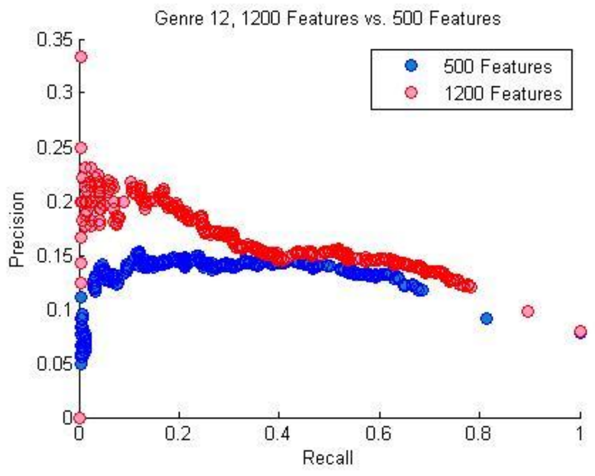
Data Preprocessing

For all the algorithms that we tried, we took the data that had been gathered through the website Graham had set up and isolated the unique strings from their likes/interests description as their "features" -- for example whether they live in the U.S., their gender, etc. Each profile was associated with a vector of binary features representing whether or not the corresponding feature applied to them or not. For example, if somebody liked Harry Potter then that person's profile had a one in the index of the feature vector corresponding to Harry Potter. To get the values we were trying to predict we went through all the profiles and for the ones that had specified their favorite movies or TV shows we figured out the genres of these shows and movies, based on Graham's mapping, and associated these profiles with a set of genres. Since many profiles did not specify a movie or TV show, these profiles were not associated with any genres.

Analysis: Naïve Bayes

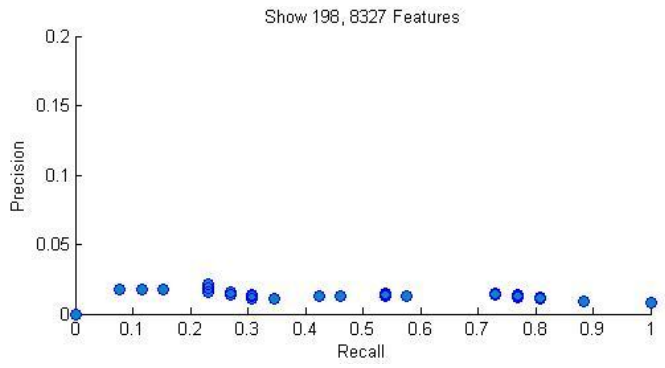
We first decided to model which genres a user would like by creating a binary classifier for each genre. Given a user's interests, such a classifier could predict, for example, whether or not that user liked action films. Once we'd found a user's preferred genres, we could turn them into specific movie and TV show recommendations by creating a list of all the movies and shows that were of the predicted genres. Our first step in this direction was to implement a Naïve Bayes classifier for each genre. We trained each model on 70% of the user profiles, and tested each on the remaining 30%.

With the standard Naïve Bayes approach (predict whichever classification has the highest probability), each genre's classifier tended to predict all 0's -- i.e. that no one was interested in that genre! Given our training data, this was actually a very accurate approach -- even the most popular genre, comedy, was liked only by 790 out of 10271 people, in the sense that only 790 people had "liked" a movie or TV show that we could identify as a comedy. But as this example illustrates, accuracy was not the best measure of our algorithm's success. Instead, we considered precision and recall. We introduced a new "decision boundary" parameter to the Naïve Bayes model that would make it easier for a "1" to be predicted, and we could thereafter generate precision-recall graphs for a given classifier by simply varying the decision boundary.



Precision-recall for different feature set sizes: 500 features (blue) vs. 1200 features (red) on comedy genre.

We started out using only the 500 most common features in our Bayesian classifiers, but soon found that these classifiers had high bias. As a result, we increased the number of input features; using 1200 to 1600 features produced the best (i.e. highest-precision) results. Incidentally, the Naïve Bayes approach also justified our decision to consider genres, rather than individual shows and movies; our classifier

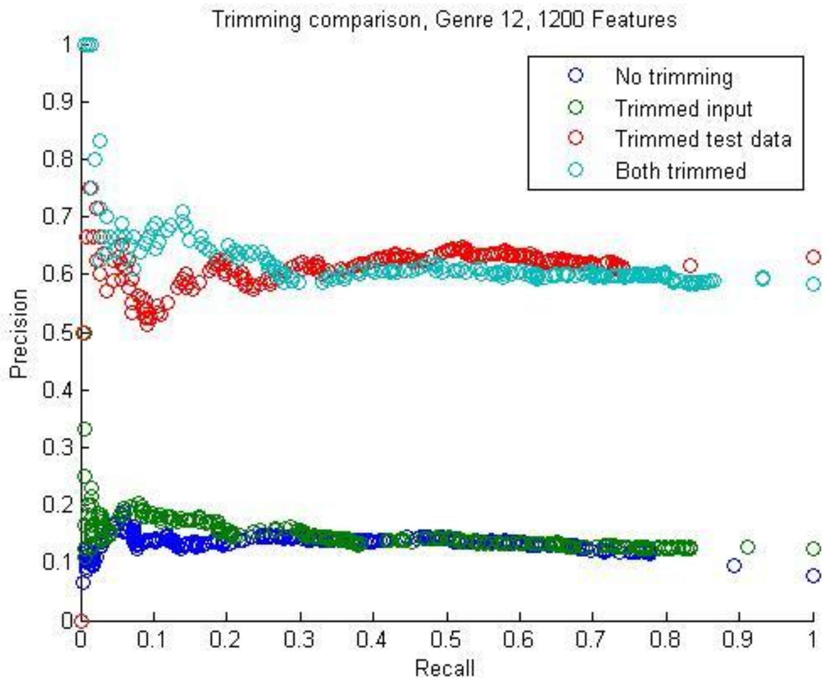


Individual-show classifiers were imprecise.

for even the most popular show (*How I Met Your Mother*) was far less precise than our genre classifiers.

One approach that did work well was cleaning the input. Our major challenge in this problem was dealing with the fact that most Facebook users don't share all their interests online, so that many profiles listed no, or only a few, interests for us to use. We also attempted to build classifiers based on **trimmed input**, wherein we only trained the classifier on users whose profiles gave us at least three features. With this modification, the precision readings from our Bayesian classifiers were more stable over the full range of recall values.

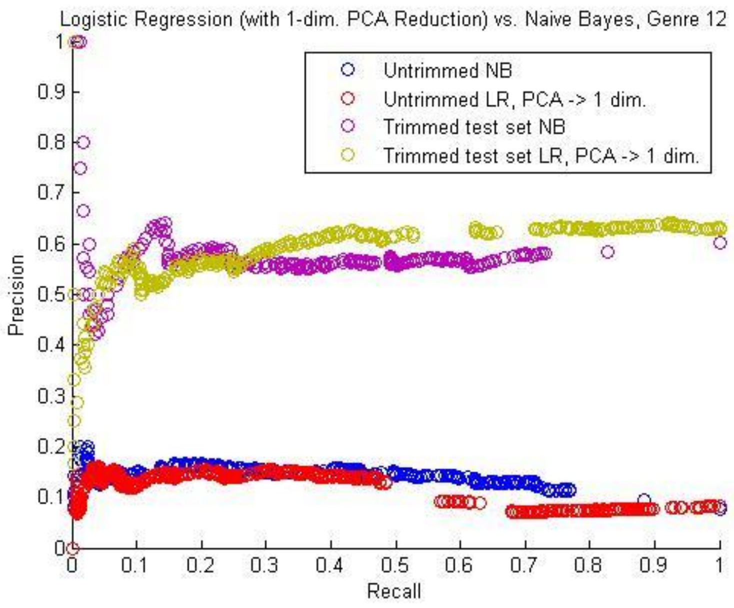
In this same vein, we noted that many users did not list any movies or TV shows in our show-genre map as interests, meaning that we couldn't compute genres for them, and represented them in our data as if they didn't like any genres. Since virtually everyone does like *some* genre, even if he/she might not talk about it on Facebook, we also ran some algorithms over a **trimmed test set**, consisting of our usual 30% test set without these users. Over this test set, Naïve Bayes performed (comparatively) extraordinarily well, reaching over 60% precision on the comedy genre. In practice, of course, we'd have to give recommendations to all users, regardless of how much we knew about them. However, for users who listed a non-trivial amount of interests (i.e. would be in the trimmed input) and who liked at least *some* genres of film (i.e. would be in the trimmed test set, at least if their Facebook interests accurately reflected their real interests), our model would perform fairly well.



Trimming test data (light blue, red) greatly increased precision, while trimming input increased stability.

Analysis: Logistic Regression

Given that predicting genres worked better than predicting individual shows and movies, we wondered whether the input features might also benefit from some summarization. As a result, we decided to try running PCA on these features, then building and testing a logistic regression model over the resulting lower-dimensional feature set. The approach itself performed well; it was actually better than the Naïve Bayes model for trimmed test sets. More interesting, however, is just how low-dimensional an adequate representation of our feature space could be! Even reduction to a single dimension (from the otherwise 8327-dimensional feature space) produced quite good results, though reduction to 10 dimensions did better. Presumably, the data is so sparse that the input feature vectors are determined largely by the presence or absence of the most common features -- living in the US, being male, being female, liking Harry Potter, and so on. As a result, the apparently high-dimensional feature space is quite well approximated by very low-dimensional subspaces, and so PCA works very well.



Logistic regression (red, gold) operated over a wider range of recalls than Naïve Bayes, and performed better with a trimmed test set, even after PCA to just one dimension.

Analysis: k-Means Clustering

For our last approach, we divided the set of users as **active** – the users that have liked at least one movie on Facebook, i.e. those in our trimmed test sets -- and **passive** – the rest of the users. More than 80% of users in our data set happened to be passive and incorporating unsupervised learning would help us utilize this big amount of data.

We ran K-means clustering on the data set, ignoring TV/movie likes as before. To predict the liked genres for a new user, we would identify the cluster he/she would belong to and if the genre was among most popular P genres in that cluster, as identified by that cluster's active users. We used 70% of active users and all passive users for training and tested the results on 30% of the active users. The precision-recall curve is generated by varying the parameters K and P.

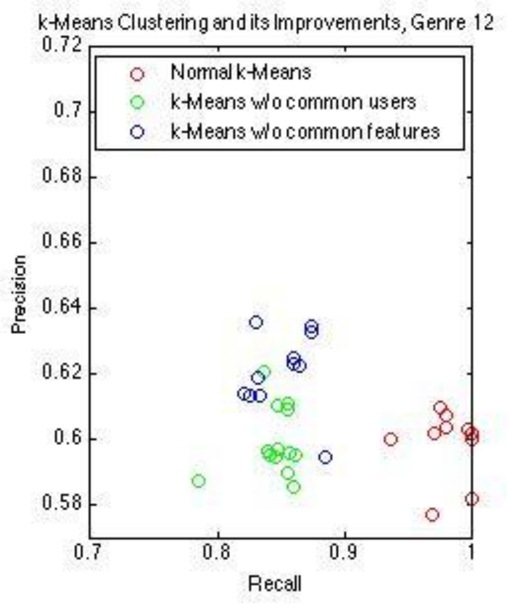
The intuition behind the model is that by running K-means we find a group for each user to which they belong based on similarities of their features. And the active members of those groups represent every other member of their group, including the passive ones.

Our first implementation was the straightforward use of the algorithm above. We achieved high recall, but the algorithm gave a lot of false positives which led to low precision. The reason was that only two clusters -- U.S. females and U.S. males -- represented most of the users. Most of the users in the test set would be assigned to one of those big clusters and get a positive result most of the time, since the chances of finding active users in that cluster who have liked any popular genre would be high. Therefore the main challenge was to reduce the sizes of clusters thereby reducing the number of false positive results, while maintaining most of the relationships within the cluster.

We came up with two related solutions to this problem.

We found that a third of all users had ones in their feature vectors only at positions representing en_US and/or Male/Female and called those users **common** users. Not surprisingly the majority of common users were also passive.

In our second attempt we removed common passive users from our training set. This would mean that the mean vectors of clusters of mostly common users were less pulled toward the common users and would give more freedom to other features to 'pull' the means. Also we excluded the common active users from the clustering and created their own two 'clusters' (males from US and females from US) and we would treat them as other clusters.
This attempt had fewer false positives and therefore improved the precision; however still about half of the users in the training set were in the 'big clusters', whose means were close to those of the two big clusters of our attempt.



k-Means worked best once we reduced the impact of the features – residence in the US and gender – common to most users.

Therefore, our last attempt was a slight improvement of our second attempt. That is, after removing all the common users and before the clustering process we nulled out the features of all the users in the training set (i.e. making an assumption that whether a person was from US or the user's gender were not useful in identifying whether the given genre is liked or not), thereby letting the other features impact the clustering. This last approach had better precision than the second one at the same recall levels.

Conclusion

After trying out these different approaches it became clear that predicting movie and TV show interests based on information in Facebook profiles is a very difficult task. While the supervised and unsupervised learning algorithms did do moderately well, the highest precision we got was still around 60%. The sparsity of the data, notably the fact that most profiles did not include movie and TV show interests, made it more difficult for our supervised learning algorithms to predict that a particular profile would like the genre that we were trying to predict. A similar problem occurred with our K-Means Clustering algorithm; in this case the lack of similarities between profiles caused very few large clusters to form. For both of these types of algorithms, we saw that testing on profiles that have at least one movie or TV show interest increased their precision.