Realistic Movie Recommender

Date: April 19, 2013
Author:Andrew Thompson
Supervisor: Dr. Anthony White
COMP4905 - Honors Project
Carleton University
School of Computer Science
Ottawa, Ontario, Canada
Winter 2013

Abstract

Businesses need to know how to present information to users in order to maximize their traffic, revenue, and satisfaction. Finding an efficient way to obtain and present that information is difficult. Typically recommender applications only have knowledge of a user's previous experience. This document will examine how recommendation applications could be improved by using the user's social networks as well as their previous history. This information is based on studies showing that users trust recommendations made from friends more so than recommendations made by systems [Stokes, 2013]. Using a movie recommender application created on Android for the purpose of investigating this claim, this paper shows how social networks add insight into the recommendation process. Ultimately this information may result in a better recommendations for the user, consisting of more accurate results and a richer selection of movies.

Acknowledgements

I would like to thank Professor Anthony White, first and foremost, for his advice and experience in the world of recommender applications. I would also like to thank my friends for contributing to my social network, and helping test the application after it’s development.

Table of Contents

List of Figures

List of Tables

1. Introduction

Over the course of reading this document I will outline the movie recommender application and the problem that it attempts to solve. First the problem will be detailed and its background will be explored; why it is important to create better techniques for recommender applications, and the motivation behind that.

Then the movie recommender application will be outlined and the development of the application will be discussed. This will include information about the various web services considered during development. The chosen platform will be addressed and various design patterns included in the application will be highlighted. The methods by which data is collected will also be included.

Next I will go over the algorithm that I used to determine the results. This section will include the factors that play into the algorithm such as the pearson coefficient, and the method by which I decided to analyze said results.

Finally the document will discuss the results of the project; what was successful and what was not. Further research will be outlined as well as improvements that could be made to the movie recommender application to make it more robust and able to create better recommendations.

2. Motivation

2.1 Background

In the growing world of e-commerce, online businesses need to be able to market their products to users in an efficient way in order to stay competitive. Many companies such as Amazon and Ebay offer one recommendations based on one's previous purchase history, and items one has previously viewed. In order to do this these companies utilize recommendation systems which parse information generated by the user to recommend which items it computes will be most well received. Typically this process is limited solely to user history but as technology advances social networks are becoming more powerful and contain more information.

Facebook is an example of such a social network. Cited in September 14, 2012 Facebook's user count was estimated to be approximately 1 billion users active on a monthly basis [Zuckerberg, 2012]. In August, Facebook was cited as stating approximately 8.7% of their user base are fake accounts. Based on these measurements we can say that Facebook has approximately 900 million active users as of Sept. 14, 2012. Users in Facebook are able to "like" specific pages which represent products and topics. This information can be utilized to learn not only more about the user, but about the user's social network as well.

2.2 Personal

As a user I am typically involved in various forms of entertainment. Many entertainment platforms that are involved in media and have network connectivity, such as the xbox 360, will tend to recommend movies to the user based on as much information as it can gain. Typically I have found these recommendations to be lacking as the system which they stem from has very little

information about me or my movie tastes.

Part of the motivation for developing the movie recommender application is to demonstrate how user interests and likes are readily available. The application should gather this information from a pre-existing source instead of asking the user to create a user account and import all their liked movies. This simplified process saves the user time, makes the application more convenient to use, and with a wealth of extra information allows for a far more sophisticated recommendation to the user. To me as a user this is extremely appealing.

2.3 Goals

There are multiple goals for the recommender application. The first goal is that the application should require little work on the user's part to work. Part of the application is to circumvent lengthy sign up processes as well as data population that has already been performed by the user at a previous date. The second goal is to create an accurate recommendation based on the data the user has established on favorite movies. The third and final goal is to show that a user's social network can improve and augment a movie recommendation.

3. Methodology

3.1 Platform

3.1.1 Mobile Development

I chose to create the application for mobile development. As the application is a movie recommender it is convenient to have this functionality on the go. The application can be used while deciding what movie to rent or watch. The application is also relatively small which is also ideal for use on a tablet or phone. Typically mobile operating systems make it convenient and quick to load applications so this application is always in reach on a mobile platform.

3.1.2 Platform Choices

Platforms that were considered were iOS, Blackberry, and Android.To develop on iOS would require more equipment, specifically the Mac OS in order to get access to XCode. Without the MacOS the application could only be developed in Eclipse which would require a physical iOS device that had been previously jailbroken. To develop on Blackberry I would require a blackberry device. Since the latest Blackberry OS can emulate applications from the android market Android becomes the clear choice. I also have an android phone available so development naturally shifted towards Android OS. The SDK is readily available and the language, java, is something I have a foundation in from years of working with it.

Another choice was creating the entire application as a Facebook application. This choice was unappealing as I felt it may limit future choices. Denying the application of other markets was not appealing as I had the intent of making an application with long term appeal. In hindsight a Facebook application may have been more logical as the recommender application's

functionality is already strongly tied to Facebook.

3.2 Web Services

3.2.1 Movielens

Movielens was the first choice when proposing a movie recommender application. Movielens is a project created by Grouplens Research, a research lab in the University of Minnesota. Movielens is academically sound, and offers snapshot of their data from 2011 for free. The downside to basing the application around Movielens is that the dataset does not grow or stay current. Updates to the dataset are given at liberty of Grouplens to aid others in Academic research. This situation generates a problem where as movies become more current the recommender system loses value. Generally new movies get more attention than old movies, and so a large amount of recommendations for new movies would be lost due to the older data set. Ultimately this issue lead to the choice of using existing web services in lieu of Movielens.

3.2.2 Facebook

Facebook is the primary web service that the recommender application uses to determine a user's social network. Facebook is powerful in that it offers a pre-existing wealth of knowledge relating to what movies users like, and what their social network likes. Being community driven the problem with the Facebook application comes in that there are no standards. Users create the pages for movies which can be misspelled and incomplete. This results in genres for movies being missing, and leaves the recommender application with no way of categorizing the movie. Thus for a Movie recommender application, the Facebook web service needs to be augmented with more information.

3.2.3 Rotten Tomatoes

Rotten tomatoes (RT) is a movie database I had planned on investigating at the start of my proposal. RT turned out to be accessible with a wealth of critic reviews and user reviews. Reviews come in two scales for each movie from 0-100% and “fresh or not fresh”. RT also offers a similar movie call which is excellent for finding similar movies that users will enjoy. This is perfect for making a global recommendation, based on what other people outside of the user’s social network like.

The problems that come with RT is that it limits requests to 10 per second per API key and 10,000 per day per API key. This limitation compounds with the problem that the movie search does not return the genres or categories of the movie found in a generic search. In order to get that data, which is crucial to the movie recommender, a second request must be made for the movie specifics. Even as a single user testing the application, with a moderate amount of friends on Facebook, the application would need make over 520 requests in order to get the information it needs for a single user. A minute for the application to start the first time with a single user’s load on the system is unreasonable. Given there are cases of people with 1000’s of friends, and the application could be used by more than one person simultaneously, these limits proved too strict to be the sole database for the application.

3.2.4 IMDB

IMDB provides an API but it is entirely undocumented. It runs on AJAX and gives results back in JSONP format (JSON with Padding). As their API is undocumented and not meant for public use, it can also be changed frequently without documentation updates. IMDB also does not make their limits known. As it provides too many unknowns this API proved to be unusable.

3.2.5 Freebase

Freebase is an internet database containing information on a variety of topics. To query the database Freebase uses it’s own query language Metaweb Query Language (MQL). Freebase

proved to be robust with 100,000 requests per day available per person.

The problems that Freebase encounters is similar to Facebook. Pages are user generated, and make no promise of specific information such as genre, description, or pictures. As a result this database also proved to be unusable.

3.2.6 OMDB Api

The OMDB API is database built around the Freebase engine, and uses IMDB's movie information. It provides very few features, but the features provide all of the information required for the movie recommender to to get information on the movies it finds. OMDB offers two searches, one for IMDB id's, and another for text. Submitting a text query will return a JSON formatted movie matching the name in the text.

3.2.7 Mixture of Web Services

For the movie recommender application a selection of three web services were chosen. Facebook, Rotten Tomatoes, and OMDB.

Facebook is used to get pre-existing user information, including their movie likes and social networks. This information is limited to the names and pictures of movies and the names and pictures of users in their social network.

OMDB API is used to enhance the information gained through Facebook. Each movie that is found across the user and all the user's friends is looked up in the OMDB API to gain information on the movie's genres, year, and IDs for IMDB and RT.

Rotten Tomatoes is used to create generic recommendations by looking up similar movies to high rated (or globally recommended) movies. The API limit for this function is never reached as information on the movies has already been obtained.

3.3 Software Design

3.3.1 User interface

The user interface is done with Fragments. The Android platform is generating more and more hardware variations with different screen sizes. Tablets running the android OS are becoming more popular [Singh, 2013]. Fragments offer more control over the user interface and how information is displayed, and so they are the logical choice when creating a modern android application.

As such, the application uses a single fragment activity. Each view is loaded immediately when the application is started. As the application needs to load movies and gather data anyway, this extra step adds a negligible amount of time to the application's startup. Following that, showing and hiding fragments is extremely quick and fluid. When fragments are shown they are added to a backstack so the user can navigate to previously viewed fragments.

3.3.2 MainActivity

For the purposes of this application the fragment activity acts as the controller, and each individual fragment acts as a view. Users interact with the view which informs the controller of changes and user actions. The controller then delivers the action to its destination. The work is done in a separate thread, and the controller is notified when the work is done. The result of this design is an entirely asynchronous application. Each component of the application is informed when work is completed, and then processes that work for the user. This design is very beneficial to working with web services.

3.3.3 WebServiceManager

The web service manager is responsible for handling all the web service calls. Each call is processed in a separate thread, though only one at a time. These threads are handled with a thread pool in order to avoid rebuilding a new thread and destroying it when the thread is terminated. The reason for this is that Rotten Tomatoes will block requests that are made too quickly, resulting in a significant number of dropped requests.

3.3.4 SettingsManager

The settings manager is responsible for saving settings, and storing settings for other aspects of the application. The settings can be modified in the settings fragment. When saving settings they are saved to a properties file on the phone's external storage so they can be accessed by the user. Other aspects of the application can request settings from the settings manager which acts as a singleton.

3.3.5 MovieManager

The movie manager is responsible for saving and loading the movies. In order for the user to import their "liked" movies from Facebook the application must request these movies of a more detailed webservice. In one test case where the user had over 250 friends this Facebook request resulted in over 700 unique movie requests. To minimize these requests at application startup the application caches and saves any successful movie requests. This process is done through the movie manager.

When the movie manager loads movies it loads them from a JSON file. I chose to use a JSON file as opposed to an SQL database because I am more familiar with JSON files. Since the database does not grow too large JSON files worked fine for the purpose of this application. The file is saved in external storage so it can be retrieved and backed up.

3.3.6 FacebookManager

The Facebook manager is responsible for handling the user and user's friend's data. Communication with Facebook always goes through the Facebook manager. The Facebook manager is also responsible for communicating with the other web service managers in order to create social and global recommendations. Social and global recommendations both operate through the Rotten Tomatoes web service.

3.4 Data Collection

3.4.1 User Comparisons

In order to compare users to each other the application computes a Pearson coefficient for each friend on the user's friend list. A Pearson coefficient is a number between -1 and 1, and shows a correlation either positive or negative between two variables. This value is calculated based on the genres that the user and the user's friend like.

To do this a genre map is created for both users mapping the number of times the genre appears in their liked movies to the genre name. The genre map is then normalized so that users with more movies than other users don't skew the results. This calculation is done by dividing the number of times the genre appears by the total number of genres giving us a percentage of total genres.

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

	Genre	**User 1 - X**	**User 2 - Y**	**xy**	**x^2**	**y^2**
	Crime	14.29	14.29	204.08	204.08	204.08
	Action	25.00	25.00	625.00	625.00	625.00
	Adventure	14.29	14.29	204.08	204.08	204.08
	Fantasy	3.57	3.57	12.76	12.76	12.76
	Animation	3.57	3.57	12.76	12.76	12.76
	War	0.00	0.00	0.00	0.00	0.00
	Sci-Fi	14.29	14.29	204.08	204.08	204.08
	Comedy	3.57	3.57	12.76	12.76	12.76
	Thriller	14.29	14.29	204.08	204.08	204.08
	Drama	7.14	7.14	51.02	51.02	51.02
	SUM	100.00	100.00	1,530.61	1,530.61	1,530.61
	N	10.00				
	Pearson Coefficient	1				

Table 1 - Genre Map between two Identical Users

Once both maps are normalized the coefficient is calculated based on the standard equation as seen in Figure X. The coefficient between two users gives the application a means of comparing them. Users who have a high negative coefficient should be negatively correlated, meaning that when one user dislikes one genre the other user should like a separate genre. If the correlation held, increasing the like of one genre would also increase the like of the other genre. A coefficient close to 0 means there is no correlation between the users.

Finally, a high positive correlation should mean that both users have similar tastes. As

one user likes a genre the other use should like the same genre. For the purpose of this application only users who have high positive correlations are explored. I believe it is possible that users with high negative correlation could also shed some insight on the user's likes and dislikes, though that will not be explored in this document.

3.4.2 Global Recommendation

After Pearson coefficients have been generated the application runs all the data through a series of steps in order to determine the movie recommendation. First the users favorite genres are determined and the top three are selected. Following that the recommender selects 6 movies from the user that share the user's favorite genres. Given that the user has stated they liked these movies the application then requests 5 similar movies to these movies from Rotten Tomatoes. The resulting 30 movies becomes the user's global recommendation.

3.4.3 Social Recommendation

Similar to the global recommendation the social recommendation performs all the above steps. In order to augment the social recommendation the application starts by determining which of the user's friends are "good" friends to listen to. All friends who beat the user's set minimum Pearson coefficient qualify as "good" friends. The top three of these "good" friends are then selected as "best" friends in order to be recommenders. From each of these "best" friends to listen to, the top genre is selected and from that genre the application selects the highest rated movie.

The recommender then uses three of the user's top movies, and the selected top movies from the user's friends to create a recommendation of 30 movies from Rotten Tomatoes. The movies that are liked by the user's friends are flagged with a thumbs up, to show that they are liked by a friend who is over the Pearson threshold.

3.4.4 Survey

Once the application has been run the user is presented with a button that tells the application to save the results to a comma separated value file (.csv). From there I, as the developer, can import their data into a Google document and analyze the data. Using the csv files I created a survey which I had participants fill out after running the application. The surveys created follow the same format as Tables X and Y (seen below).

Fox Recommendation						
	Movie Name	**Rating**	**Flag**	**Haven't Seen It**	**Did Not Like It**	**Liked It**
	X2: X-Men United	87	FALSE			1
	Troy	54	FALSE	1		
	The Matrix Revolutions	36	FALSE		1	
	The Matrix Reloaded	73	FALSE		1	
	The Lord of the Rings: The Return of the King	94	FALSE			1
	The Last Castle	52	FALSE	1		
	The Green Mile	85	TRUE	1		

Table 2 - Social Recommendation Survey Example

Owl Recommendation						
	Movie Name	**Rating**	**Flag**	**Haven't Seen It**	**Did Not Like It**	**Liked It**
	Kiki's Delivery Service	100	FALSE	1		
	Spartacus	96	FALSE	1		
	Castle in the Sky	94	FALSE	1		
	The Lord of the Rings: The Return of the King	94	FALSE			1
	Marvel's The Avengers	93	FALSE			1
	Ponyo	92	FALSE			1
	Aladdin	92	FALSE			1

Table 3 - Global Recommendation Survey Example

Social and Global recommendations are coded as Fox and Owl recommendations respectively to avoid having participants skew their results towards the bias of the survey. Surveys contain approximately 25 to 30 movies for global and social recommendations. A participant is informed that they should mark “Liked it” with a 1 if they are willing to recommend the movie to a friend. If the participant has not seen the movie they are informed to mark the “Haven’t Seen It” column with a 1. Otherwise the participant should mark the “Did not like it” field with a 1.

4. Results

4.1 User Interaction

The goal of making the application simple to use was a complete success. The user is given a simple login button and after logging into Facebook, and no extra work is required. The application assumes that the user has selected movies to “Like” on facebook previously. If the user has less than 6 movie likes upon logging in they are informed that the application requires at least 6 movie likes to be effective.

4.2 Global Recommendation

	Global Recommendation	Haven't Seen It	Did Not Like It	Liked It	Total
	Participant 1	7	1	17	
	Participant 2	1	5	19	
	Participant 3	11	5	9	
	Participant 4	9	3	13	
	Total	28	14	58	100

Table 4 - Global Recommendation Combined Data

The global recommendation was quite successful. If the user had seen the movie already there was approximately a 79% chance that they liked the movie based on the results from four participants. If one assumes that movies the user has not watched yet will follow similar trends then the application will recommend a movie that the user will like 79% of the time.

This information aside the sample pool of the movies is relatively small. These results also vary from user to user. In my case, I liked approximately 94% of the movies recommended.

In another participants case it was 64%. In that participants case I knew him to be a more critical movie watcher.

4.3 Social Recommendation

	Social Recommendation	Haven't Seen It	Did Not Like It	Liked It	Total
	Participant 1	12	4	14	
	Participant 2	2	2	21	
	Participant 3	13	3	9	
	Participant 4	11	6	8	
	Total:	38	15	52	105

Table 5 - Social Recommendation Combined Data

The social recommendation was successful, though interesting, and this will be discussed in the next section. Once again across four participants the social recommendation showed a high degree of accuracy. The sample pool however is small, and as such the results will vary from one user to the next. The average chance a movie would be liked by a user in this case is 75% if they had already watched the movie.

A second aspect of the recommendation however, is to flag movies that are specifically chosen and liked by by friends of the user who are already deemed similar to them (based upon the Pearson coefficient). Across all 4 participants a total of 21 movies were flagged. Of these 21 movies, 4 had not been watched, 1 was not liked, and 16 were liked. Ignoring the unwatched movies 94% of the movies were liked by the user when enjoyed by a similar user.

4.4 Comparisons

The original hypothesis was that the social recommendation would make a significant improvement on the global recommendation. To prove this hypothesis I used a chi-square test to determine if the difference between the social recommendation and the global recommendation were significant.

Using two columns for “Liked” and “Did Not Like”, and two rows for “Global” and “Social”, I created a table (See Table 6, below) to determine the chi-square value of the data for each participant and the combination of all participants for a total of 5 tests.

Chi-Square Test		Did Not like	Liked	Total
	Global	15	52	67
	Social	1	16	17
	Total:	16	68	84
	Expected:	12.76	54.24	
		3.24	13.76	
	Results:	0.39	0.092	
		1.55	0.36	2.40
			Crit Value:	3.84
				DO NOT REJECT NULL HYPOTHESIS

Table 6 - Chi-Square test for data combination

In no case of any participant, or in the combination of data, did the chi-square value beat the critical value. In no case was I able to reject the null hypothesis. Thus given my data I am unable to show that the social recommendation has a significant improvement over the global recommendation.

Looking at the data this conclusion is logical and sound. In the case of participant 1, the global recommendation recommended 17 movies of which 16 were liked. At these low values,

this does not leave much room for improvement.

In the case of other participants such as participant 4, the social recommendation made a worse recommendation than the global recommendation. Even though the recommendation is worse a chi-square test still shows that the results are not significant.

4.5 Conclusion

The global and social recommendations seem effective, although they lack a proper control group to compare their results to. The question I pose is: given an entirely random movie what is the chance the user will like it? As I lacked the resources to do a full investigation into the statistics of this question, I could not get a large participant pool to watch entirely random movies. The participant pool I did have access too had not watched enough movies for me to achieve accurate results with a survey of random movies.

Using the results and knowledge we do have access to, the user's movie history, nearly 80% accuracy in recommending movies seems fairly successful. Further, 94% accuracy when only selecting from movies chosen from the user's similar friends also seems very successful.

As for the significance of the results, a flaw with the chi-square test is that it is difficult to show significance with low values. If the sample size were to be doubled and the current trends to proceed at a perfectly linear rate, the results would be significant to 1 degree of freedom.

Finally, I believe results may be improved by factoring dislikes into the algorithm. A problem with using Facebook's movie likes and social network is they do not have a section for dislikes. Thus you are limited to what a user likes, but what a user dislikes is also very important. I believe factoring in these values could add significant substance to the application.

5. Final Thoughts

5.1 Learned Concepts

Over the course of developing the application I learned a lot about web services. The web service of note being Facebook, which the majority of the application is built around. The Android Facebook SDK is very well documented, offers some excellent tutorials, and has a strong developer community. Getting answers to questions on sites such as Stack Overflow is possible.

I also learned more about Android development. Specifically I learned how to handle external and internal storage, and how to work with Fragments and Fragment activities. It is clear that Google is expanding on the Fragment design as they add more and more Fragment types that mirror activity types. Given the low cost of loading them, versus an activity, and the ease of switching between Fragments, I will be incorporating them into all of my future applications.

Finally, I learned a lot about real world applications of statistics. It was a challenge to determine exactly what test I should use on my data to determine its significance.

5.2 Web Service Limitations

Web service limitations were a challenge that came up frequently while creating the recommender application. Many popular web services are willing to offer their service for free, but the limitations can be quite staggering. In the case of Rotten Tomatoes there is a limit of 10 requests per second and 10,000 requests per day per API key. When a single user is using the application they can run into problems if a large number of requests need to be done upon initialization. If multiple users are factored in this limitation is hit extremely quickly.

Finding a way to both circumvent these limitations and have an efficient application is a challenge that any developer will face. Ultimately if a developer wants to create a serious

application that utilizes pre-existing web services they are going to have to make a choice. Either the developer will have to pay the licensing fee that the service is offering, or they will have to create their own web service.

5.3 Future Work

There is a significant amount of work that can be done to explore this field of social networks and recommender applications. More research could be done into the method of comparing two users. In my case I used a Pearson coefficient to compare users by genre likes which are extrapolated from their movie likes. I don't believe that is an unreasonable way of measuring two users, but my method is limited in that I lack a scale for each genre and movie.

Using the Facebook "like" system results in a flawed measurement. As a user you may enjoy something, but enjoy something else much more. Liking is also subjective. Two users stating they "like" the same thing could have vastly different interpretations. The system also lacks dislikes which is another rich source of information. Based on what a user dislikes one may be able to recommend other items the user does like or at least avoid recommending the things they dislike.

5.4 Redesigned Recommender

If I were to redesign the recommender application there are a few changes I would make. The first change would be to decouple it from Facebook more. In my own case I had added 69 movies to my Facebook account for purposes of testing the application. With 55 friends on Facebook I returned a total of 236 movies. Not counting my movies this result means that on average a user will have three movies liked on their account. This means that the genre map used to compare people that is extrapolated from these users is going to be extremely small and not very accurate. Given that the data on these movies from Facebook is also flawed and inaccurate, and requires augmentation from a separate web service, this makes the value of

Facebook much lower than originally estimated.

Due to the restrictions on API calls and the weakness in Facebook's social network for movie likes I believe it may be better to create a new web service and use the application as an extension of that web service. While it would compromise the first goal of allowing the user to use the application without sign up or data submission, a dedicated web service would grant the developer much more control over the application and its data source.

Finally, with a new web service and no API restrictions to worry about, I would have everything done through the application. Users would be able to log in, add movies they like, remove movies they don't like, get recommendations, add and remove friends, etc. Then the whole application could hook into an existing movie service that offers the actual movies and not just reviews. When users get recommendations they would be able to navigate directly to the movie and queue it up to watch. I believe this would make a much more useful and satisfying application as the user could go from discovering a movie that interests them to purchasing and watching the movie in relatively few steps.

References

Geoffrey, Fowler. "Facebook: One Billion and Counting."*The Wall Street Journal*. 4 Oct 2012. Web. 19 Apr 2013. http://online.wsj.com/article/SB10000872396390443635404578036164027386112.html.

Golbeck, J. (20XX). *Generating predictive movie recommendations from trust in social networks*. Unpublished raw data, University of Maryland, College Park, Maryland.

Singh, Sameer. "Tablet Market Share Trends: Android Tablets Overtake IPad, Maybe For Good." *Tech Thoughts*. 5 Feb 2013. Web. 19 Apr 2013. <http://www.tech-thoughts.net/2013/02/android-tablets-overtake-ipad-market-share.html>

Stokes, Tracy. "Majority of consumers trust brand recommendations from friends." *CMO*. , 27 Mar 2013. Web. 19 Apr 2013. <http://www.cmo.com.au/article/457399/majority_consumers_trust_brand_recommendations_from_friends/>.