Name: Gearoid O'Connor

ID: 14203203

Module: COMP 40320 Adaptive Personalization

Title: Recommender System

Contents

Introduction

The World Wide Web was initially designed for the exchange of unstructured data between scientists at "CERN" [1]. It quickly exploded to become a mechanism for e-commerce and social networking with enormous amounts of information being available to the user. In September 2014 a historic milestone was reached, the number of active websites reached one billion and is doubling every nine months [3].

These vast quantities of data being collected and available to the user can be overwhelming. In order to elevate this problem personalization technologies have been widely implemented across multiple domains. The goal is to enhance the users experience by reducing the cognitive load required for user to find the desired information. This is achieved by filtering, sorting and classifying the available information in order to present to the user the personalized items that are relevant and novel. The problem of finding and presenting relevant information has been an issue for a far longer time than the web has been in existence. The classification of information into sets of fixed categories began in 1668 by the English philosopher Jon Wilkins [4].

The area of information retrieval and classification has evolved into topics such as information visualization and recommender systems which are very topical at the moment as research fields. Interest in these topics is due to its challenging open issues [1]. Competitions like the Netflix Prize have also helped make significant progress towards better classification and recommendation systems [2].

In this paper, I document, analyze, implement and compare three various recommender systems against a baseline metric. The baseline in this instance is an overlap scoring function using a max recommender.

The three techniques that will be analyzed are a fixed and personalized threshold weighting (task 4), item based collaborative filtering (task 8) and term frequency weighting using a cosine similarity function (task 9).

The dataset being analyzed contains 1000 users, with 1073 different movies. There is a total of 31288 movie ratings across all genres. The data was extracted from the website "Flixster".

The rest of this paper is organized as follows. Section 2 describes the dataset used in detail providing a deep understanding of the distribution of the data. Section 3 will discuss and analyze the personalized threshold weighting algorithm. Section 4 will discuss and analyze the item based collaborative filtering algorithm. Section 5 will explain and analyze the term frequency weighting using a cosine similarity function.

Dataset Statistics

The graphs in this section give a comprehensive understanding of the distribution of the data being used. The data in this dataset statistics section are taken from the training set data. Figures one and two are skewed right with the vast majority of users rating between zero and thirty movies and a near total majority of movies receiving between zero and fifty ratings. Figure three shows that users are more inclined to give a high rating when reviewing a movie.

Table one displays the ten most liked and disliked movies with five movies receiving a perfect mean rating of five with only 1 receiving a mean rating of 1.Figure four and five are skewed right with majority of directors and actors directing and acting in only one movie. Table two indicates that sci-fi is the lowest rated genre and biography being the highest.

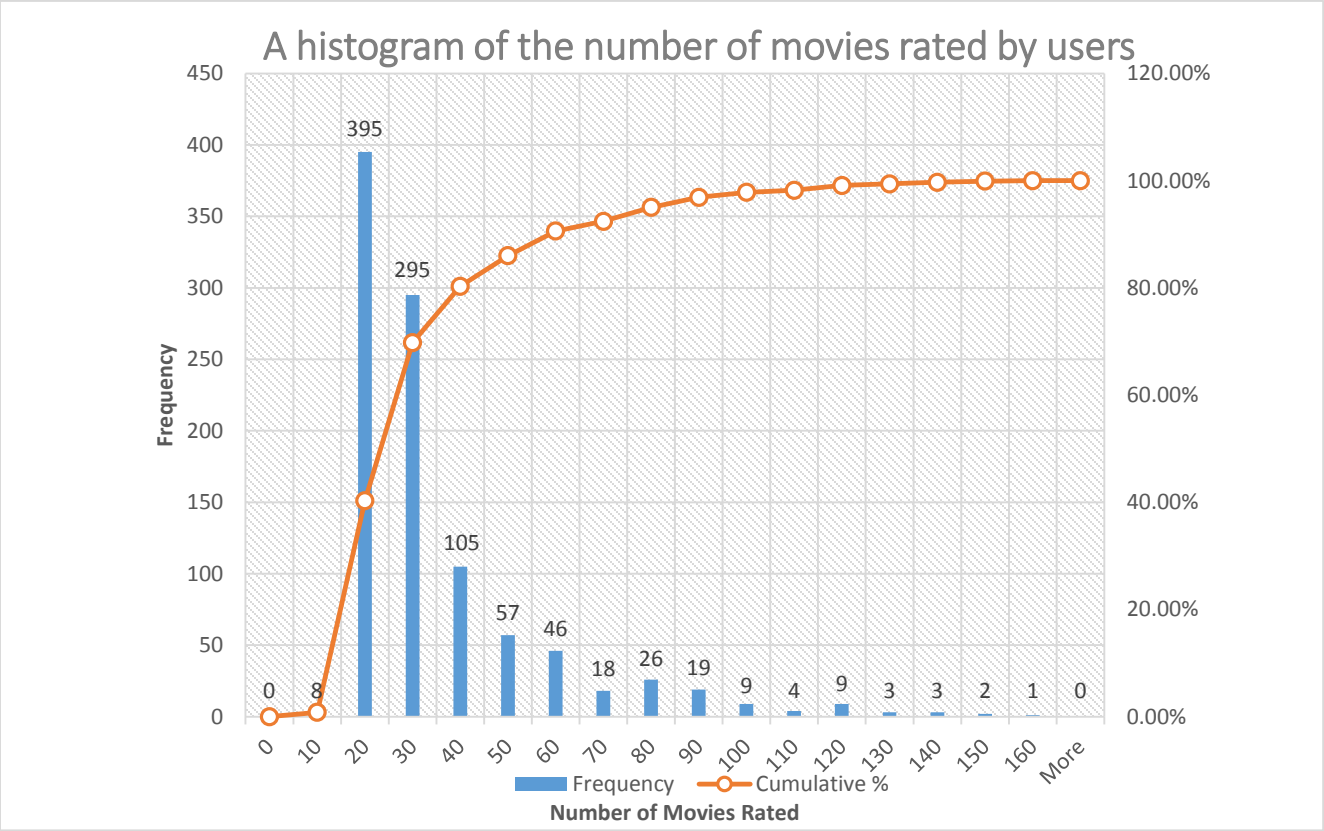

Figure 1: The number of movies each users has in their profiles.

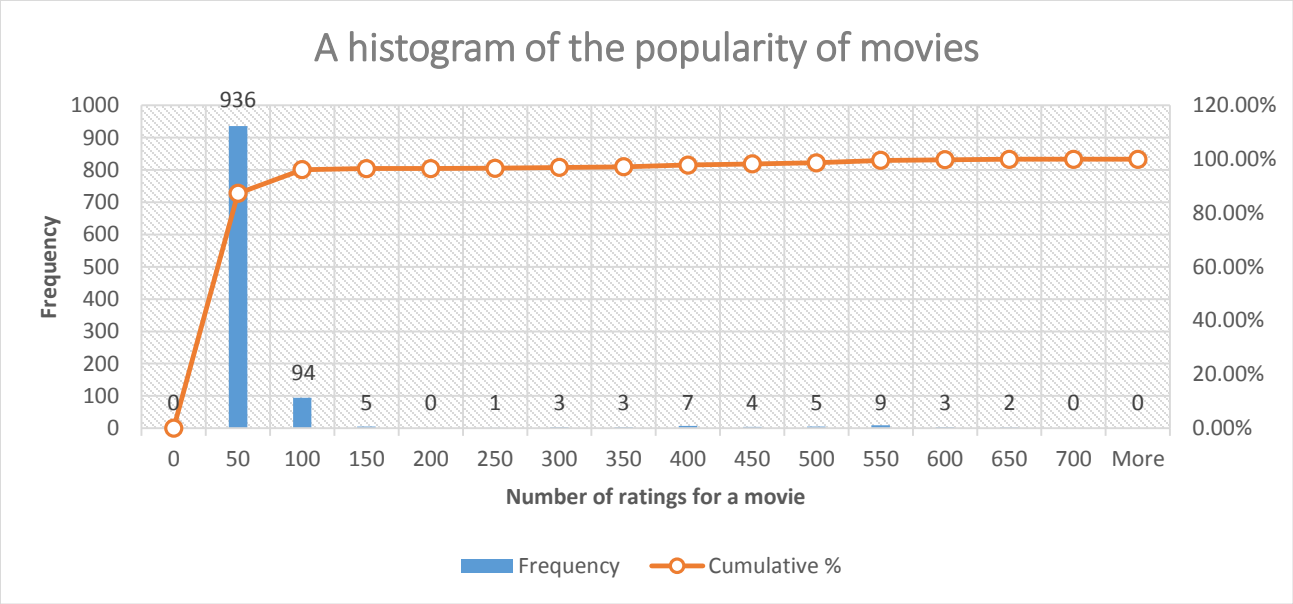

Figure 2: The popularity of movies based on number of ratings

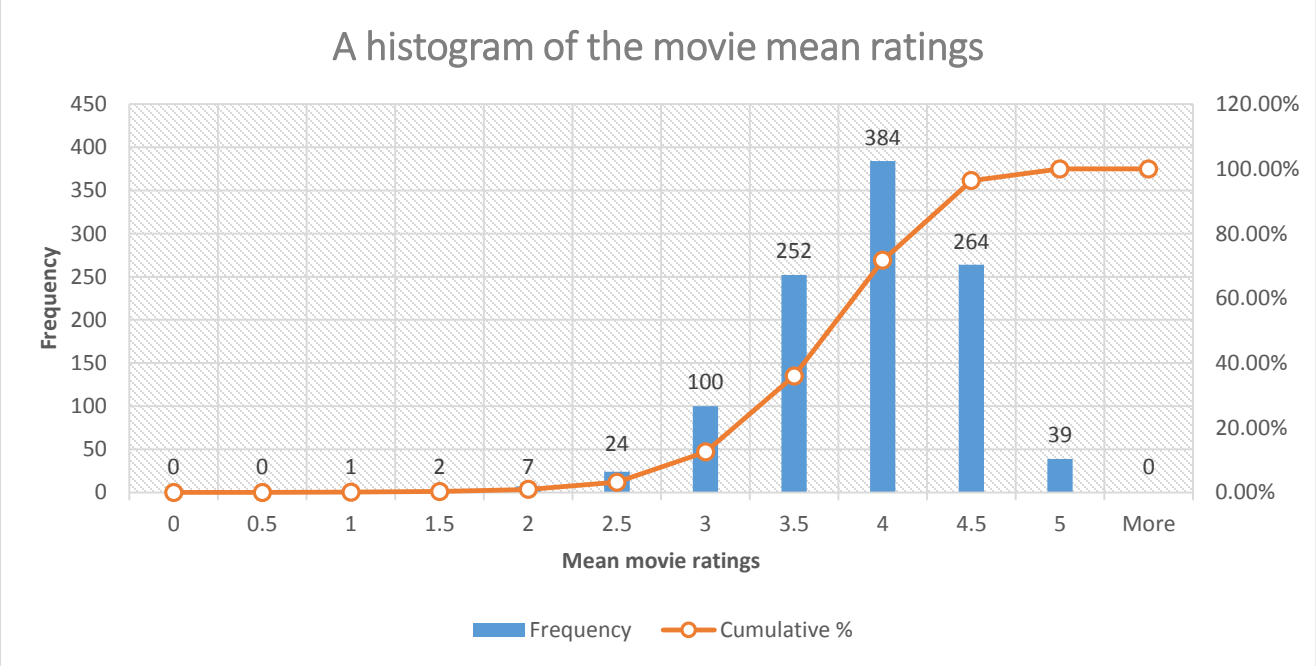

Figure 3: The mean rating of movies given by users.

Movie-ID	Mean Rating	Standard-Deviation
10 Best Rated Movies		
14671	5	0
14840	5	0
8803	5	0
11199	5	0
15902	5	0
7827	5	0
199	4.833333333	1.47734
2572	4.833333333	0.28868
3210	4.833333333	0.28868
6085	4.833333333	0.28868
10 Most Poorly Rated Movies		
6908	2	0.93541
9518	2	1.83712
1558	2	1.41421
13886	2	1.64317
11850	2	1.58114
1792	1.875	1.55265
8809	1.583333333	1.06849
1105	1.5	1.9685
9780	1.375	1.43614
13638	1	0.5

Table 1: The mean and standard deviation of the ten most liked and dis-liked movies.

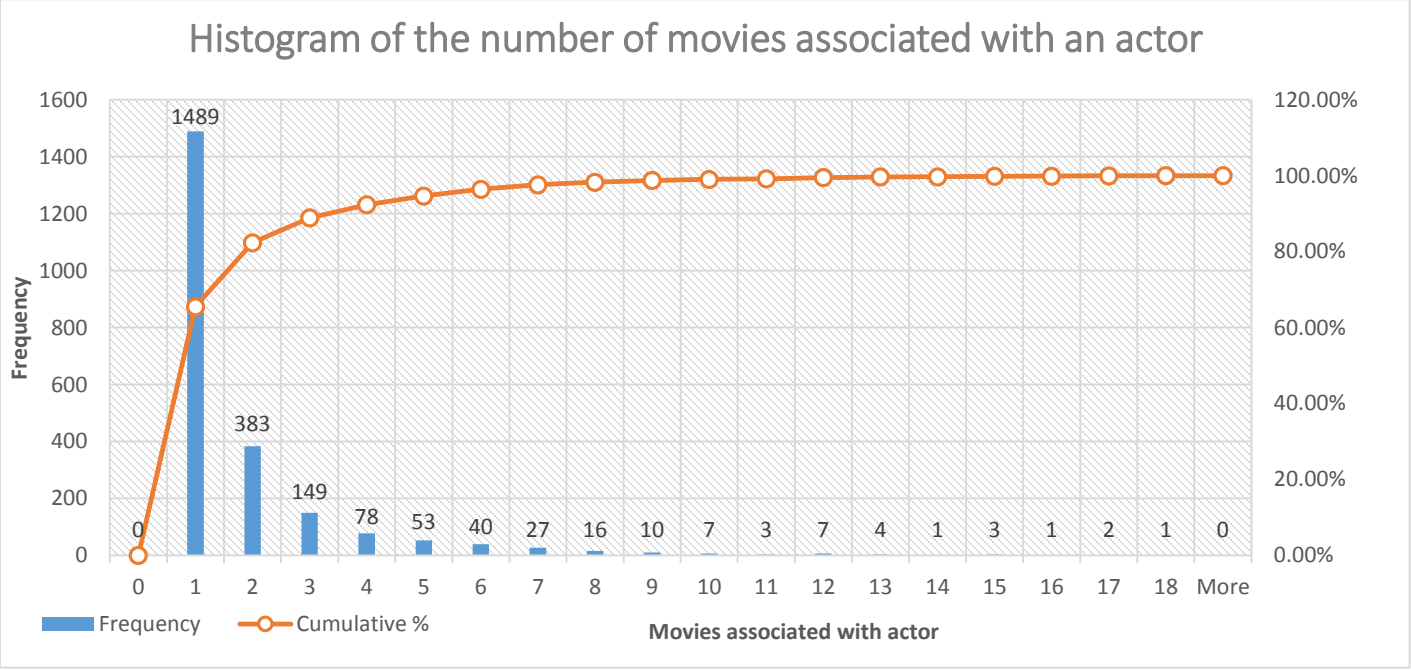

Figure 4: The number of movies each actor is associated with.

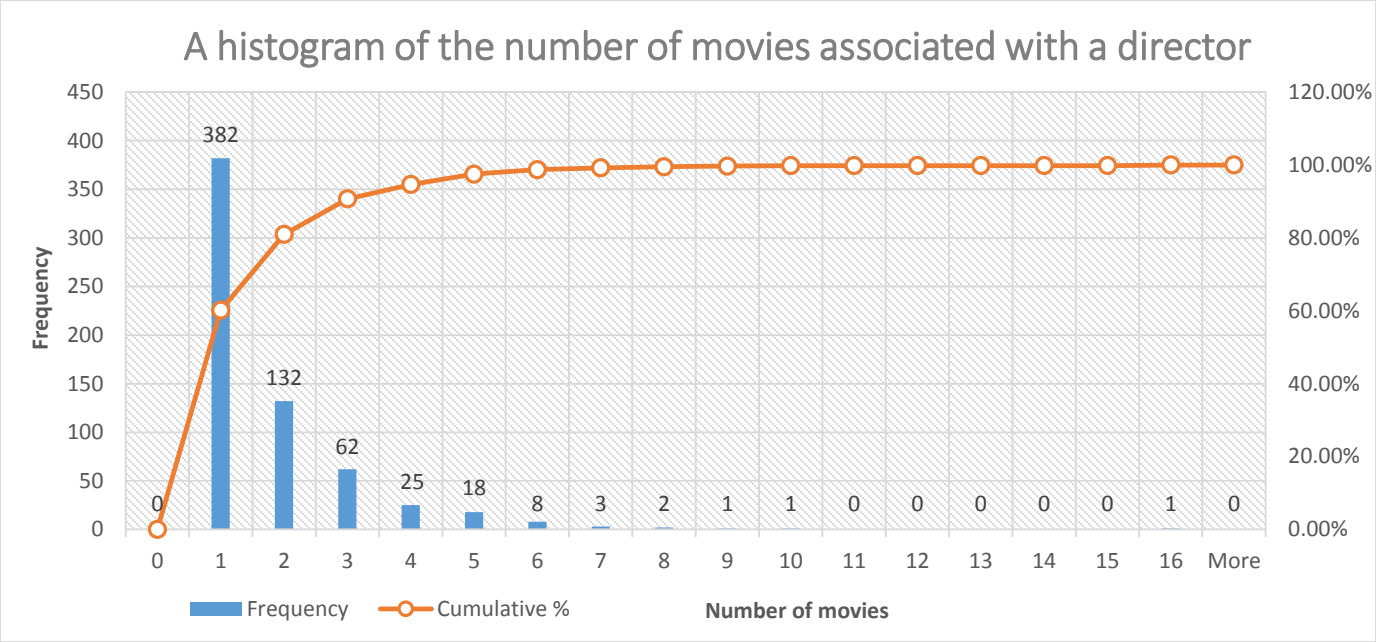

Figure 5: The number of movies each director is associated with.

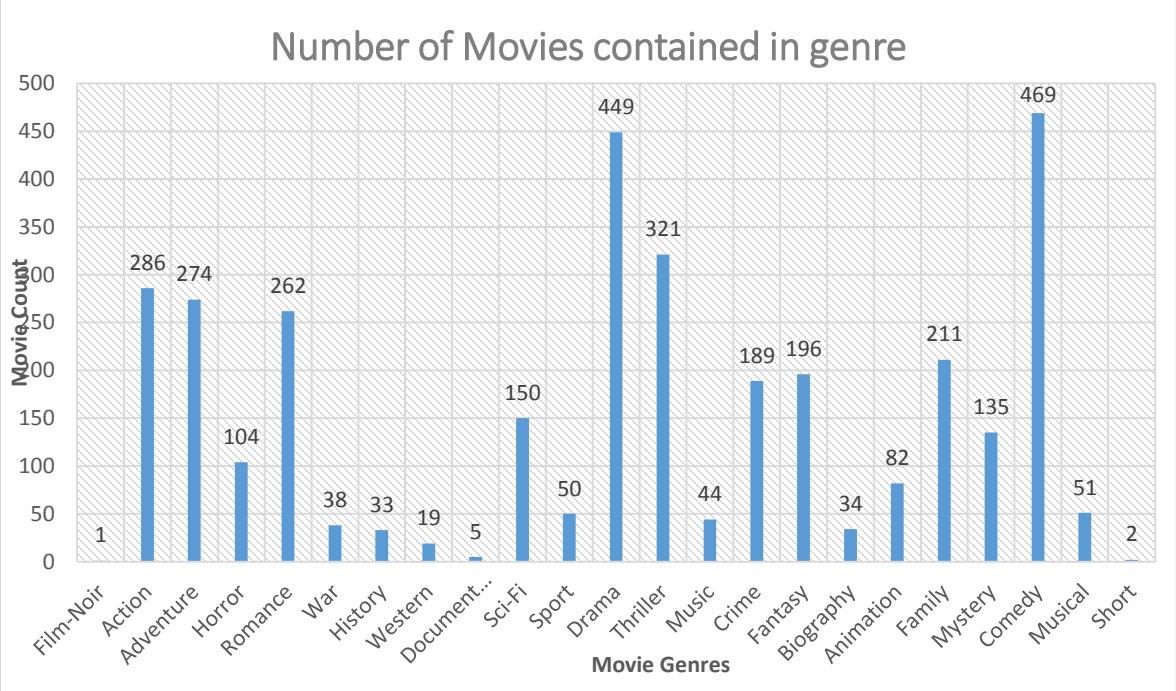

Figure 6: Number of movies per genre.

Genre	Mean	Standard Devation
Film-Noir	3.833333333	0
Action	3.528691874	0.577369114
Adventure	3.593519901	0.546198187
Horror	3.506410877	0.54816855
Romance	3.67442788	0.486486542
War	3.932212628	0.567939267
History	3.800772185	0.608938114
Western	3.823001484	0.436756077
Documentary	3.610042544	0.842739417
Sci-Fi	3.454401961	0.58157431
Sport	3.745302702	0.519150176
Drama	3.817459321	0.513441857
Thriller	3.602531712	0.590296902
Music	3.662274782	0.594700765
Crime	3.754417126	0.590700415
Fantasy	3.632574364	0.503776771
Biography	4.067228423	0.349954274
Animation	3.819376083	0.442285425
Family	3.613159404	0.540391877
Mystery	3.709851472	0.557887352
Comedy	3.60244102	0.559842844
Musical	3.804624795	0.412617667
Short	3.963541667	0.119791667

Table 2: Mean and standard deviation of movie ratings for each genre.

Filtered Threshold weightings

Filtering threshold weights involves excluding movies from a user's profile which are below a certain level. Two filtering methods were implemented. The first was a fixed threshold weight. Using values of 4, 3.5 and 3 to filter the movies from a user's profile. The second method was to use personalized weights. The personalized weights were based on the mean (μ) and standard deviation (σ) of the user's profile. The formula used to generate the threshold was $\mu + \omega\sigma$. Where ω is a weight used to vary the importance of the standard deviation. 0.1, 0.3, 0.4 were the three different values used for ω.

I hypothesize that the predicted accuracy will decrease. I believe the accuracy will decrease slightly because user's profiles will contain only their favorite movies as their lower rated movies will have been removed. This seems logical because if you were to ask a user for two sets of movies to make recommendations from, the first set containing a user's highly like movies along with ones they rated poorly or just movies they rated highly, you would prefer the set with both liked and disliked movies. Movies in a user's profile with low ratings are very useful for knowing what sets of movies not to recommend to a user.

Fixed Weight	3.0	3.5	4.0
Removed ratings	6061	10342	14115
Percent Removed	19.37%	33.05%	45.11%

Table 3: Number of ratings outside the fixed threshold levels

ω	0.1	0.3	0.5
Average Threshold	3.75	3.97	4.15
Removed	14206	16110	17720
Percent Removed	45%	51.48%	56%
Users Removed	25	46	75

Table 4: Users removed and average user weight for personalized filter threshold

Figure 7 and 8 show the obtained results. It is clear from these figures that these methods did not improve on the baseline and also that the personalized filter performed worse than the fixed threshold. From analyzing the data I have realized this was for two reasons.

The first is that the user profiles have a vastly reduced number of movies ratings to make recommendations on. Comparing figure 7 and 8 with table 3 and 4 it is clear to see how the percentage of ratings removed affects the ability to accurately recommended movies. There is a clear correlation between the size of user profiles and the precision and recall score.

In table 4, the average threshold weight was calculated in order to gain insight into what threshold levels the algorithm was using. Also the number of users removed was calculated which equates to the number of users who had no ratings below the $\mu + \omega\sigma$ value.

Secondly, movies which a user dislikes and rates poorly are as useful for making predictions as highly rated movies as they help us in understanding which sets of neighbors a user will not like. This verifies the hypothesis made in the second paragraph that this implementation will decrease the precision and recall compared with the baseline.

The reasoning for the personalized threshold performing worse than fixed threshold is clear once the average threshold applied in table 4 is analyzed. The average threshold applied is higher than the fixed threshold in table 3 for all three cases. Resulting in a smaller user profile in turn reducing the accuracy of the predictions.

The results from the filtered thresholds tests inform us that both user's lower ratings and higher ratings are needed in the prediction of recommendations for a user as the removal of either reduces the precision and recall. They also show us that there is a correlation between profile sizes and the ability to make recommendations as smaller profile sizes produce lower precision and recall.

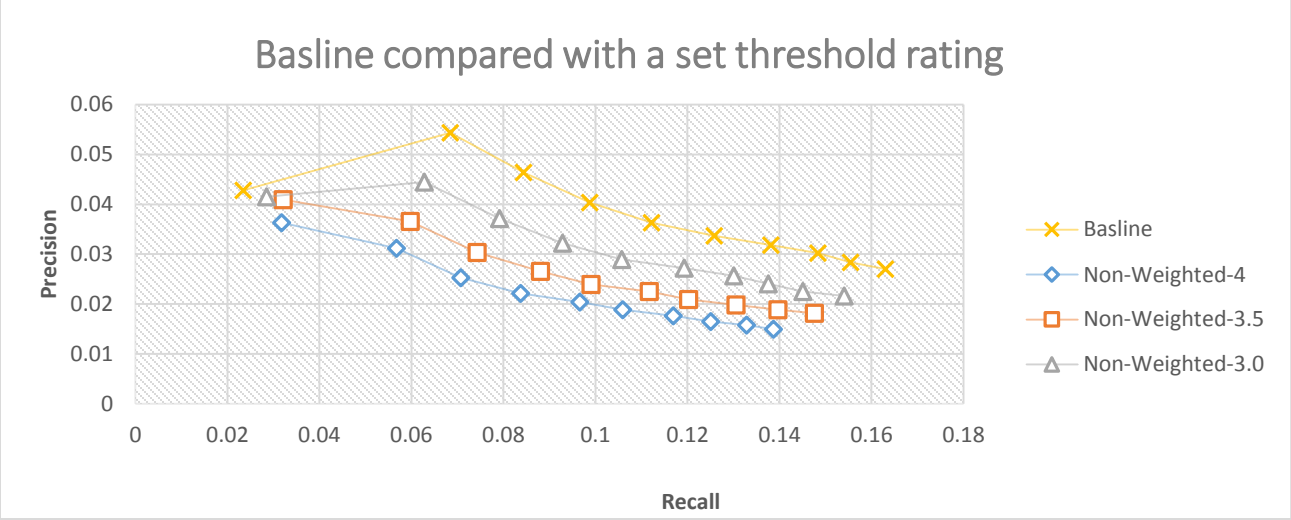

Figure 7: A fixed rating threshold of 3.0, 3.5 and 4.0 compared with the baseline case.

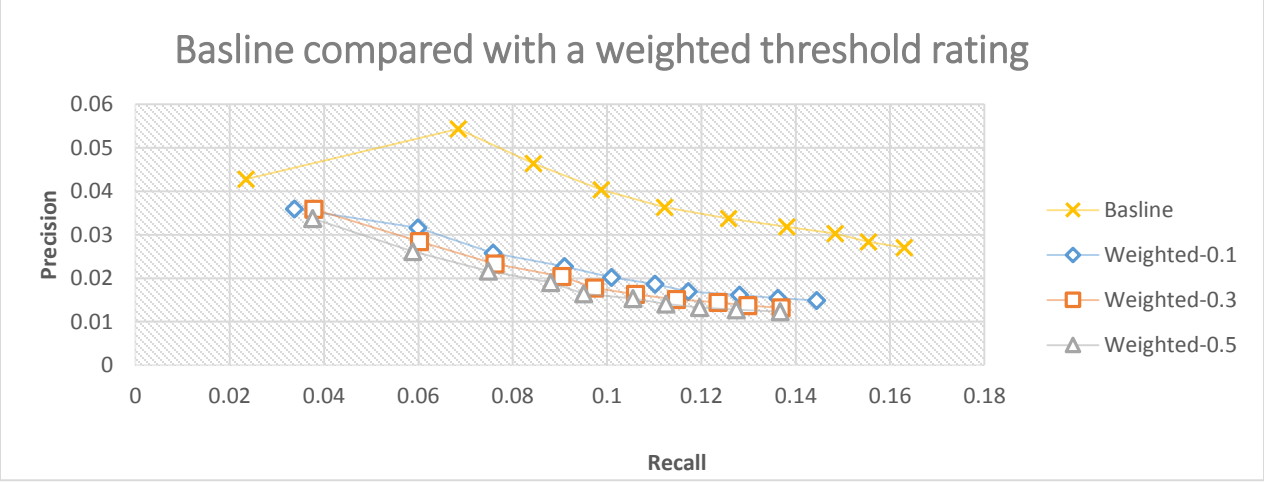

Figure 8: A personalized weighted threshold with ω values of 0.1, 0.3, and 0.5 plotted against the baseline.

Item based Collaborative Filtering

The approach used for this task was an item based collaborative filtering method. Collaborative filtering is currently one of the most popular and widely used personalization techniques for commercial products [7]. Collaborative filtering works on the basis that people who rated movies with a similar score in the past will also make similar recommendations in the future [8]. Collaborative filtering uses user ratings to discover similarity patterns between users and generates the prediction for an item by weighting the ratings of similar users on this item.

Item-based collaborative filtering which is a model based algorithm analyses the set of items rated by a user and computes how similar they are to the target item. It then selects the K most similar items. The Adjusted Cosine Similarity metric was used to find the similarity between items. A weighted average of the target user's ratings on most similar items is taken as the prediction.

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}$$

Equation 1: Adjusted Cosine Similarity

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$$

Equation 2: Weighted Sum

I anticipated that using the Adjusted Cosine Similarity function with weighted sum would produce a result higher than that of the baseline.

The reason for using the Adjusted Cosine Similarity is that people often rate movies on different scales. A movie critic my rarely ever give a 5 star rating where as a casual movie watcher my give it often to a lot of movies. The Adjusted Cosine Similarity handles this problem by subtracting the corresponding user's average rating from each rating.

It is imperative that when using the weighted sum function that a user's rating of a movie is normalized to between [-1, 1]. Once a prediction is made it can be demoralized to the [1, 5] rating.

The results obtained from this algorithm provided extremely low results when compared to the baseline. With some of results boarding a precision of zero. This is the exact opposite of the improvement I had predicated when I was implementing the algorithm. In order to understand the reason for obtaining such low results we must analyze the short comings of using collaborative filtering.

Despite collaborative filtering's wide-spread adoption it has two major short comings. The first relating to its sparsity and the second is related to its scalability [9]. The sparsity of test data set is 96.36%. This means that we have very little co-occurring rated items which results in very poor similarity results in the Adjusted Cosine Similarity metric being zero a high percentage of the time. The sparsity of this matrix is however on pair with other datasets used to measure recommendation algorithms. The Movie-Lens 1M dataset has a sparsity of 95.53% and the Movie-Lens 10M has a sparsity of 98.65% sparsity [10]. In this system 69.78% of users rated less than 4% of all available movies.

Another issue with the provided solution is scalability as the computation grows with the number of user's and items. It is clear with a run-time of over four minutes that this is a very computationally heavy approach as the given dataset is tiny compared with the amount of information handle by the companies such as Amazon or Google .

One solution to deal with this issue is too use dimensionality reduction as a pre-processing step [11]. This could take the form of principal component analysis (PCA) or canonical correlation analysis.

In conclusion item-based collaborative filtering is not an applicable approach for the given dataset.

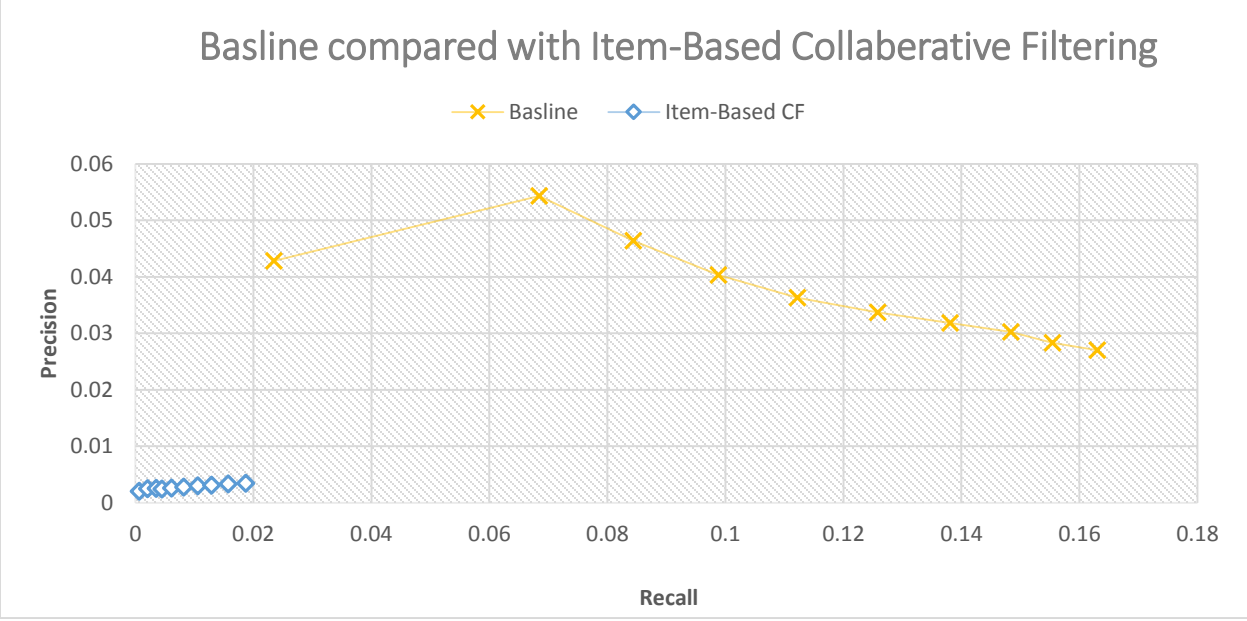

Figure 9: Graph comparing an item-based collaborative filtering approach plotted against the baseline case.

Term Frequency weighting:

The approaches analyzed so far have been based on case-based and collaborative filtering recommendation techniques. They have not utilized the unstructured text recommendations provide with each users review of a movie. This technique which is a content based approach merges all the reviews for a certain movie into a single document. This document can then be represented as a term-vector, which is used in calculating document-document similarity. The document-document similarity is calculated using the Cosine Similarity. The Cosine Similarity is then used as an extra feature in the overall assessment of the similarity between two items.

$$\text{Cosine}(d_i, d_j) = \frac{\sum_{t \in d_i \cap d_j} w_{t,d_i} \times w_{t,d_j}}{\sqrt{\sum_{k \in d_i} (w_{k,d_i})^2} \sqrt{\sum_{k \in d_j} (w_{k,d_j})^2}}$$

Equation 3: Cosine Similarity

Converting the document into a term-vector consists of two steps. The first step is to "clean" the document which is a process of removing non-alphabetic characters, stemming each word and removing stop words. The second is to "weight" the importance of a particular term in a particular document. Two weighting functions are compared for this technique, binary term frequency weighting and TF-IDF (Term Frequency-Inverse Document Frequency).

$$\text{TF-IDF}(t_i, d_j) = \frac{f(t_i, d_j)}{\max\{f(w, d_j) : w \in d_j\}} \times \log \frac{|D|}{1 + |\{d \in D : t_j \in d\}|}$$

Equation 4: TF-IDF Weighting

The aim of stemming and stop word removal is to remove words which do not help in identify similar movies reducing the term-document size which in turn should reduce the runtime. The stop word list contained 547 words and porters stemming algorithm was used. The selection of stop words is critical as failing to reduce common words which don't distinguish documents will reduce precision. The word "good" is a critical stop word. When not removed it reduces the average F1 score across all the top N recommendations by .99% for TF-IDF weighting.

15909 was the number of terms in the term-doc matrix prior to removing the stop words and there were 6838 left after removing them. This gives use a compression of 42% which verifies the typical compression rate of 40% stated by Ricardo Baeza-Yates [5]. There is however concern that a high number of stop words used might reduce recall.

The purpose of maintaining an extra matrix (MovieWordList) with an array containing every word in a document was to reduce the time required for finding the co-occurring words. Comparing arrays should reduce the run-time compared with looping through the term-document matrix. This reduce run-time by a phenomenal amount. Prior to implementing this it took over thirty minutes to run and after it took just under four minutes.

The hypothesis for using the TF-IDF weighting metric was that terms which appear often in a lot of documents should get a low weighting resulting in these terms being less important in calculating the similarity between documents. TF-IDF reflects this so in turn should produce higher precision and recall.

The TF-IDF performs better for the top 5, 10 and 15 predictions. TF-IDF performs 44% better than the baseline when comparing F1 scores. This is a substantial increase. The binary method performs better for the remanding values. The top 5, 10 and 15 are the most important to a user so TF-IDF performs well in the critical areas. Both binary and TF-IDF perform better than the baseline. This was expected as you are adding a new feature which allows us identify similar movies which was not possible before. For instance "Light" "hearted" are two words which appear in a lot of similar movie reviews that is not reflected in genres so improves the precision of the recommender.

TF-IDF outperforming on the top end of the scale is also expected. I had not expected binary to outperform TF-IDF at the lower end but logically it makes sense. Using the inverse term document frequency we give terms which appear in a small collection of documents higher weighting resulting in documents which contain rare co-occurring terms getting a high similarity. In our collection the most defining terms for a document will probably occur in at most 15 documents. So outside this range the TF-IDF will perform poorly as selecting terms which are of relevance to a document becomes hard.

The key findings from this experiment are that a combination of case-based and content-based recommendation performs much better than case-based on its own for the critical values. The main limitations of this approach is the assumption made by the cosine measure which is that a document length has no impact on relevance which according to TREC data is not true [6].

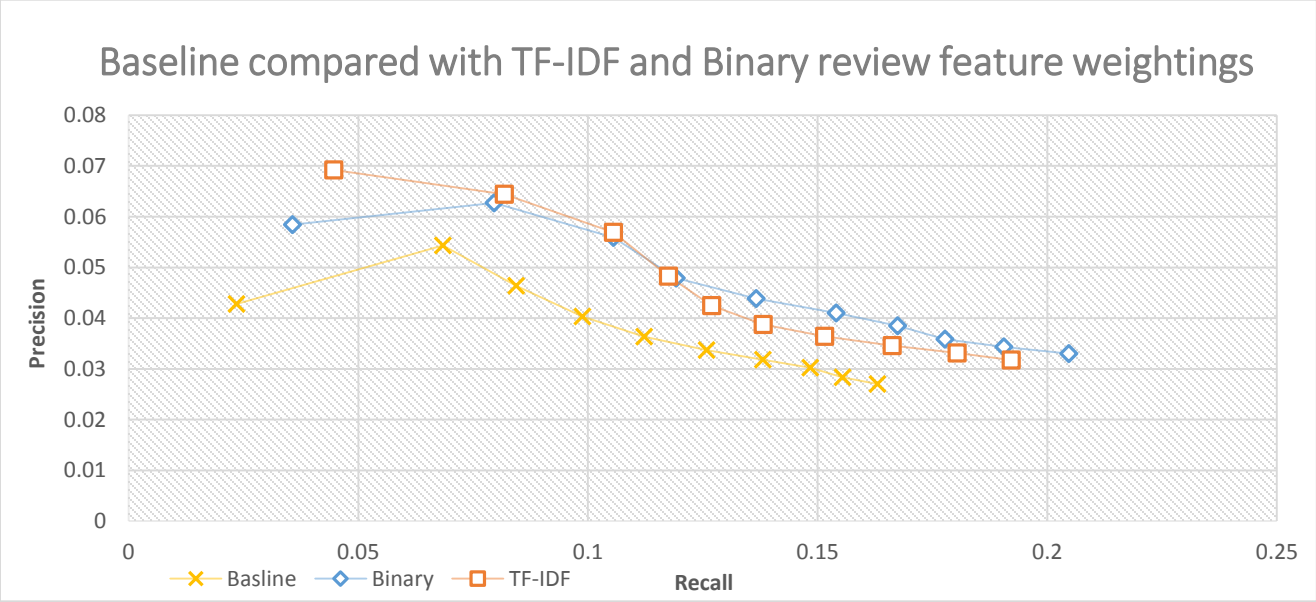

Figure 10: TF-IDF and Binary weighting function for content based retrieval plotted against the baseline case.

Conclusion

In this paper we analyzed the given dataset. Reviewed and implemented work in case-based retrieval, collaborative filtering and content based retrieval accessing the effectiveness of each on the dataset. A meticulously detailing the purpose of performing every task identifying the hypostasis we hope to achieve and comparing this hypostasis to the gathered results.

We obtained a high degree of variance between the results of each implementation. We have shown that that due to the sparsity of the matrix collaborative filtering is not an ideal approach. In comparison the inclusion of term frequency weighting for the reviews provided greatly increase the accuracy of the predictions.

Meticulous implementation and careful testing of a given recommender system is required to access the real world value of a given implementation. It is critical that the system is not over fitted to the data. K-Fold Cross Validation should be applied to multiple datasets to insure the results are accurate and reliable. The results presented in this paper have only been tested on a single dataset using a single training and test set. Further testing would be required to verify the finds of each implementation.

The greatest improvement on the baseline came from the term-frequency weighting therefore a further research direction of particular interest could be develop this implementation into a genetic algorithm to dynamically select the appropriate weight for each term. I believe that dynamically selecting the term weights could produce more accurate recommendations. The fitness function for the genetic algorithm could be based on a relevance assessment of the top x documents retrieved.

References

[1] J. Teevan and R. W. White, "Slow Search: Information Retrieval without Time Constraints," Association for Computing Machinery, 2010.

[2] "Netflix Prize," 2 October 2006. [Online]. Available: http://www.netflixprize.com/rules.

[3] "NetCraft," 10 October 2014. [Online]. Available: http://news.netcraft.com/archives/2014/10/10/october-2014-web-server-survey.html.

[4] J. L. Subbiondo, John Wilkins and the 17thCentury British Linguistics, John Benjamins Publishing Company, 1992.

[5] R. Baeza-Yates, Modern Information Retrieval, Addison Wesley, 1999, p. 167.

[6] D. A. Grossman, Information Retrival, Kluwer Academic Publishers, 1998.

[7] G. Uchyigit, Personalization Techniques And Recommender Systems, World Scientific Publishing Co, 2008.

[8] U. Shardanand, "Social Information Filtering: Algorithms for Automating "Word of Mouth"," 1995. [Online]. Available: http://www.sigchi.org/chi95/proceedings/papers/us_bdy.htm.

[9] J. K. a. J. R. J. Schafer, "Recommender systems in e-commerce," in *ACM E-Commerce*, 1999.

[10] "MovieLens," [Online]. Available: http://www.recsyswiki.com/wiki/MovieLens.

[11] P. Cunningham, "Dimension Reduction," UCD-CSI, 2007.