Project Proposal: Hybrid Model for Movie Recommendation

Bhuvan M S
msbhuvan@uw.edu

Darshan Mehta
darshanm@uw.edu

Yash Kale
yashkale@uw.edu

1 Introduction

We were inspired by the Netflix data challenge explained in the course. We realized that movie recommendation is a highly impactful problem. We build on the dataset by combining it with other forms of data to propose a hybrid model. We explore three papers explained below which inspired us to propose this project.

2 Reaction Papers

2.1 Recommendations Without User Preferences: A Natural Language Processing Approach

This paper discusses how to aptly summarize the plot descriptions of movies, and generate a lower dimension description of the plot. The movies on IMDB generally come with a plot description, and a genre associated with each movie. This paper aims to create topic signatures for each movie, and represent each plot for each movie as a vector representation of the topic signature [1]. Topic Signatures are generally of the format:

(topic, signature), where signature = ((term1, weight), (term2, weight),....., (term3, weight)). Signature consists of all the terms $term_i$, which represent the topic with their respective weights. For example:

Sci-Fi	Horror
1689.4 Alien	1142.5 Vampire
1155.9 Planet	519.2 Blood
1094.1 Space	489.9 Horror
886.3 Scientist	387.1 Monster

The topic signatures for genre Sci-Fi would be: (Sci-Fi, (Alien, 1689.4), (Planet, 1155.9), (Space, 1094.1), (Scientist , 886.3)).

To calculate the weight of each word, we need pre classified sets of relevant and non relevant plot summaries for each genre. Frequency of all the terms in the plot summary is calculated for each genre, across the relevant and non relevant plot summaries. The following assumptions are made: The relevancy of a plot is based on a word: term, if $P(R|term1) = p_1 >> p_2 = P(R|term2)$, where term1 and term2 are different words in the plot summary.

Preprint. Under review.

It seems like a pretty straightforward approach and easy to understand. A stumbling block/weakness of this approach is that we would need pre classified relevant and non relevant plots for each genre, to generate a word list for each topic signature. Once we identify the terms and their respective weights, we use a threshold weight to limit the number of words in the signature of a topic. Once we have the topic signatures for all the genres listed on IMDB, we generate custom genres, and their respective topic signatures to enhance the representation of movie plots. These genre topic signatures now become our basis to check similarity of movies across IMDB. Thus, we create a vector representation of each film, where each feature is the cosine similarity of the film with our base genres. After this step, we take the cosine similarity of these film vectors, to check which films are similar. For example, the representation for the film "Omen," for example, includes the features: Horror=.63, Action= .73, Action/Adventure=.40, Comedy/Romance=.12.

For the structured part of the data, we use simpler steps. We calculate the similarity between the cast members in two films as the ratio of the number of cast members common to both films divided by the total number of cast members in both films. We then represent each film as linear combinations of similarity metrics for each type of information, i.e: plot summaries, the cast, the director, and the writer. The final similarity score would be their weighted average.

For movies with no proper defined genres (approximated genres/custom genres) on IMDB, relying on the relevant words could be problematic, as we may miss out on some words for custom defined genres.

This method of generating topic signatures could be very useful in checking similarities of movies rather than just looking at how similar the words in the plot are or how many actors are shared in common. The vector representations of the movies could be used in such a way that, each feature in the vector could possibly be a shingle in our shingle-movie matrix. Each shingle is basically the genre to which the movie has matched, with its associated weight. If we represent our vector for a movie in the decreasing order of weights of shingles, we could perform Jaccard similarity between these vector representations, to get similar movies.

2.2 Hybrid recommendation model based on incremental collaborative filtering and content-based algorithms

This paper [2] introduces a hybrid recommendation model which combines incremental update item-based collaborative filtering with a content-based algorithm which is based on latent semantic analysis. It is catered towards systems where implicit signals are generated on the fly by the user and updates to recommendations need to be performed at a very fast pace. Most online platforms today that generate recommendations on large scale deal with vast amounts of data. Updating user preferences and generating new recommendations in near real-time requires a lot of optimization. This is especially a challenge for hybrid systems which need to adjust for the similarity of both, the users and the content. This paper [2] proposes a novel approach to optimize both of these in a disjoint fashion for better performance.

One of the strengths of this paper is that they minimize the amount of computation done at runtime for updating recommendations. The major updates to recommendation and preferences happen periodically and incremental updates only happen when the user interacts with a certain item. During this interaction, only the similarity of this article with the ones the user has clicked on before is updated. There are a few drawbacks to the approach proposed by the authors. Firstly, the user to item behavior matrix is binary, i.e., the paper assumes that if a user clicks on an article, they always like it. In real life, it is more sophisticated than that. They would have to account for implicit signals that the user drops such as exiting before completing the article, sharing the article, etc. Also, the user only updates the similarity for articles that are being clicked on. Ideally, they should also update the similarity of articles that are similar to the articles being clicked on in terms of content. Another drawback is that the authors propose to use SVD. Although this technique has impeccable performance, it generates extremely dense matrices. Such a technique is less suitable for problems where the original matrix is quite sparse (such as a movie rating database).

This idea proposed by this paper can be quite useful in many more domains other than just recommending news articles. Consider the problem where a new user arrives on a platform. Asking the user explicitly for preferences might not always be the best choice for every user. Some people don't like to be asked so many questions when they are just browsing superficially. A better idea would be

to start off with the best guess of items possibly say, recommending based on demography and then update these recommendations as the user interacts with the platform. As suggested by the authors of the paper, we need to make sure that these updates are minimal to ensure near real-time updates. Here we could hypothetically use implicit signals dropped by the user to update their preferences instead of asking them upfront. We watch how the user interacts with the first item and then recommend similar items if we measure that the user liked the item or we recommend a completely different item if the user did not like the item. We could also consider using alternatives to SVD such as CUR or Coreset based algorithm which are more suitable for sparse matrices.

2.3 Dimensionality Reduction of Massive Sparse Datasets Using Coresets

In real-world most of the high dimensional data is sparse, like the recommendations data or the features computed from natural language. We have learnt that the dimensionality reduction could be very useful in processing massive datasets having very high dimensionality. We explored Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and CUR decomposition in the course. The paper [3] from highly recognized conference (NeurIPS 2016) proposes a new algorithm for dimensionality reduction of sparse data that uses a weighted subset of the data, and is independent of both the size and dimensionality of the data. The authors claim that this algorithm provably approximates original data using a small subset and a small subspace of dimensions. They also claim that the algorithm is an embarrassingly parallel enabling it to be easily scaled up in the cloud using Hadoop/Spark.

The paper suggests a "Coreset" C which is a weighted subset of rows of original matrix A (n x d), such that the sum of squared distances from any k-dimensional ($k > 0$ and $k < d$) subspace to the rows of A is approximately the same as the sum of squared weighted distances to the rows in C. Such a Coreset C represented as (e, k)-Coreset (epsilon e is the allowed error). Since A is sparse, the weights corresponding to each row would also be sparse (containing many zeros) due to the algorithm in which it is determined, leading to the reduced cardinality of C, meaning we would require to process less than n rows where n is really large. This also results in reduced dimensionality as k is much smaller than original dimensions d.

The main advantage of this algorithm is that the results is a weighted subset of original rows which makes it very interpretable. In contrast, in SVD and PCA the rows gets represented along the principal components which makes the transformed data less interpretable and the singular vectors are dense. The authors give a technically sound proof showing that the size of coreset C, is independent of the input dimensions and is a subset of original input rows (C only depends on k and epsilon e). The authors prove that use the coreset helps in efficiently computing the low-rank approximation (reduced SVD).

On the other hand, there are few drawbacks with this paper. The initial step of the algorithm depends on SVD which might take significant amount of time in computation. Also, the authors claim claim that distributing across M machines will reduce construction time of C by a factor of M, but this fact is not clearly described in the paper. The term "merge-reduce" is used instead of "map-reduce" which is not clear, and it is mentioned that the algorithm is embarrassingly parallel, but this is also not clear from the explanation. Also, the authors use synthetic data as ground truth to show that their algorithm provides a good approximation to the original matrix, but they do not show it on real data.

Further research could be to experiment this algorithm on a real-world data and by proposing a map-reduce or spark implementation of the algorithm to prove that it actually scales up. More importantly, this needs to be compared with CUR decomposition in terms of reconstruction error and the computational performance. That would help in determining an appropriate method which can be used on a real-world data which is massive in size and also high dimensional and sparse.

After exploring the above three papers, we see methods to summarize a text document based on small topic signatures. This is reducing dimensions in a way. Also, we explore efficient methods to reduce dimensions in massive sparse datasets. This inspired us to pursue text data (natural language) which generally has high dimensions and is sparse. Over the web, there are many huge datasets of text available which can help us learn these topics by applying them onto such datasets. Also, we learnt how to perform incremental updated to collaborative filtering method to get a hybrid recommendation systems along with content-based recommendation methods. From these papers, we plan to propose the project explained in the next section.

3 Project proposal

3.1 Problem Statement

We choose a common problem of movie recommendation based on multiple data attributes. Though this is a common problem, it is highly impactful in the real-world since it is used in some of the most popular services like Netflix, Amazon Prime, Hulu etc. We plan to take a hybrid approach to this problem by not only using ratings and content-based information like the metadata of the movie, but also to include the unstructured text data from the producers i.e, the published plot and synopsis of the movie, and the subjective reviews of the users as described in the data description section below. In short, we aim to recommend 20 movies to each user, which have the highest probability (ratings) of liking by the user. We will try to extend the the problem by using the poster cover of the movie also as another indicative in the extension of the work.

3.2 Data Description

For this project, we aim to combine multiple datasets on movies to get a rich set of features on movies. A few datasets that we aim to explore and combine are the Netflix Prize dataset [4], the MovieLens dataset [5], and the IMDB movies dataset [6]. The Netflix dataset [4] contains CustomerID, MovieID, and Rating. Such a dataset in itself is quite primitive in content. It does not contain enough data which could ideally help in judging the similarity of movies and user preference. It could be the case that the user only watches action movies in a certain language or which has a certain cast. Such features are quite important in understanding the user's preference and are lacking in the Netflix dataset [4]. We aim to improve this dataset with the information obtained from the other datasets. The IMDB dataset [6] provides us metadata on the movie such as the year it was released, genre, number of awards won, IMDB rating, duration of the movie, etc. The MovieLens dataset [5] provides us with even more detailed metadata such as the list of the cast and the crew, budget, revenue, languages spoken in the movie, link to the imdb poster, IMDB ID, and the plot synopsis. These datasets when combined together provide us much more features to measure the similarity of the movies on leading to better recommendations. We also plan to gather the features extracted from the poster and the plot of the movies (and potentially the reviews) and use them to compute similarity as well.

3.3 Proposed Methods

We plan to use the IMDB dataset, which contains features like actors, writers, directors and plots, using content based filtering approach, to come up with top 20 recommendations of movies. We would then perform dimensionality reduction on the features of the IMDB dataset using the NLP based method mentioned previously. The plots and reviews of the movies would be replaced with topic signatures for each movie in the IMDB dataset.

We aim to use the Netflix dataset to generate top 20 recommendations of movies using item-item based collaborative filtering. Moreover, another way to represent the plots and reviews of the movies is also through the Coreset based dimensionality reduction method. Both the NLP and the Coreset-based dimensionality reduction techniques would be explored and experimented upon, to see which fares well for the given dataset and the given problem statement.

Once we have the recommendations from both the methods (item-item collaborative filtering and content-based filtering), we then take a weighted average/majority vote of the two methods to give out our final recommendations.

3.4 Evaluation

We will consider an unseen test test which would be held-out at the beginning which is a subset of ratings data containing 20% of the total users. The ground truth will contain 'true' (to represent whether the user actually watches it or not) for the top 20 movies based on the ratings provided by the users, and 'false' for all other movies. The recommendation method proposed would find top 20 movies, and these ten movies would have 'true' and the rest would have 'false'. We shall use the the area under ROC (Receiver Operating Characteristic) curve as the primary metric. As a secondary

metric, we would consider to use the Spearman correlation (rank correlation) on the predicted top 20 movies versus actual top 20 movies.

4 Conclusion

We aim to implement the proposed hybrid method with highly computationally intensive parts on Spark, and get an evaluation score for the primary and secondary evaluation metric explained above. We plan to compare our results with the simple item-item collaborative recommendation method and the content-based recommendation method, and the latent-factor recommendation method.

References

[1] Fleischman, Michael, and Eduard Hovy. "Recommendations without user preferences: a natural language processing approach." IUI. Vol. 3. 2003.

[2] Wang, Haiming, et al. "Hybrid recommendation model based on incremental collaborative filtering and content-based algorithms." 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2017.

[3] Feldman, Dan, Mikhail Volkov, and Daniela Rus. "Dimensionality reduction of massive sparse datasets using coresets." Advances in Neural Information Processing Systems. 2016.

[4] Bennett, James, and Stan Lanning. "The netflix prize." Proceedings of KDD cup and workshop. Vol. 2007. 2007.

[5] Banik, Rounak. The Movies Dataset. 10 Nov. 2017, www.kaggle.com/rounakbanik/the-movies-dataset. Accessed 25 Apr. 2019.

[6] Leka, Orges. IMDB Movies Dataset. 15 Nov. 2016, www.kaggle.com/orgesleka/imdbmovies. Accessed 25 Apr. 2019.