Embedding-Based Bias Reduction Approach to Movie Review Data

Rohan Bais [1], Travis Chen [1], Rifath Rashid [1]
[1] Department of Computer Science, Stanford University
https://github.com/travis14/cs224w-final-project

1 Introduction

The explosive growth of platforms enabling users to review products, movies, or experiences has drastically transformed the spending behavior of our world. In fact, Bazaarvoice, a digital marketing company, published a 2012 comprehensive study which found that more than 51% of Americans trust user-generated reviews over any other information available on a website. User-rating platforms like Yelp and Google reviews have grown tremendously since 2012, motivating these businesses and independent organizations alike to conduct research on how to most effectively aggregate user reviews to report some notion of 'objective quality.'

Traditional rating aggregation services use a simple mean of all user ratings as their aggregation method. However, this makes the assumption that all deviations from 'objective quality' can be interpreted as statistical noise rather than systematic bias, but in reality ratings are subject to many situational and subjective preferences. In particular, individuals may have systematic biases along specific dimensions of products, for examples liberals rating conservative news publications or vice versa.

In particular, merchants often have nuanced information about their products but little to no information about the preferences of users aside from their interaction via ratings with products. We will attempt to accomplish 2 tasks in our paper: to generate movie embeddings that define semantically meaningful features based on user and tagging interaction graphs, and to develop a bias-detection algorithm to help define bias vectors for individual users. We will combine the two to uncover "objective ratings" for each movie and understand how this compares to top critic ratings.

2 Related Work

2.1 Constant User Bias (Mishra)

Mishra cites bias as being problematic in the context of rating aggregation. An object recommendation might not be considered because some users are biased against it even though it is inherently good. Mishra decided to capture bias information as a constant shift per user to understand what adjusted ratings would tell him about the dataset. Mishra [5] models user biases $b_i \in [-1, 1]$, meaning that a user is either inherently more positive or more negative in their ratings. Through this assumption, he defines an iterative algorithm for updating the true entity ratings r_j and user biases b_i. For notation, let us assume a model with m users and n entities, where w_{ij} is the observed rating given by user i to entity j. Let R be the set of all observed ratings. Our objective is to uncover the true ratings of each entity, r_j, having observed the set of ratings R.

$$r_j^{t+1} = \frac{1}{\sum_{i=1}^{m} \mathbb{1}[w_{ij} \in R]} \sum_{i=1}^{m} \max(0, \min(1, w_{ij} - \alpha b_i^t)) \mathbb{1}[w_{ij} \in R] \tag{1}$$

$$b_i^{t+1} = \frac{1}{\sum_{j=1}^{n} \mathbb{1}[w_{ij} \in R]} \sum_{j=1}^{n} (w_{ij} - r_j^{t+1}) \mathbb{1}[w_{ij} \in R] \tag{2}$$

$\alpha \in [0, 1]$ is a parameter that weights how much the biases are factored into the updated entity ratings. This iterative updating guarantees that the difference between b_i^t and b_i^{t+1} goes to zero, at which point the algorithm converges (see [5] for full proof). However, no guarantee is provided that the error decreases from iteration to iteration. Note that this bears a huge similarity of the recursive defintion of HITS, since this is the idea of how Mishra defined bias. Bias depends on the ratings and ratings true ratings depend on the aggregation of ratings minus each individual user's biases. Both bias and true ratings depend on each other the same way hubs and authorities do. One issue with Mishra's results is that he defines bias as a scalar that applies across all movies for a given user. This is an issue because a user may have many particular biases and preferences (across genres, casts, etc.), and this low-dimensional scalar cannot capture more subtle shades of bias. Thus we will extend it by defining a bias vector that works with a movie embedding to properly capture more information of specific biases in a high dimensional space.

2.2 Tagging-Based Embeddings (Guan et al.)

Guan et. al's work generates embeddings of users, resources and tags in the same k-dimensional hyperspace and uses these embeddings to create a recommender system called MIOE [4]. MIOE is used to recommend movies to users by finding which movie embeddings are most similar to user embeddings and recommending the closest one. One of the issues mentioned in the paper is that while Collaborative Filtering is a good method to recommend objects to certain users, it suffers from the fact that without explicit ratings the algorithm will serve mediocre recommendations and that it ignores tag data. Tags are much more finely-grained than ratings and showcase the user's comprehension of the resource when they tag it. Sparsity also may be fixed with inclusion of tags. To counteract this issue, Guan used spectral clustering to generate embeddings for each object in the same space. MIOE was able to outperform Collaborative Filtering and SVD on user-document-tag datasets. However, one issue with the algorithm is that while it does consider more information than the Collaborative Filtering, it also will take much longer. It considers all users, tags, and documents. We are interested in understanding movies in relationship with both users and tags, so we shall use this but we will extend it by only picking the most popular tags rather than using all of them, since we suspect that this algorithm without modification is prone to overfitting. Once we get these embeddings, we would directly use them as input into computing the user bias vectors in the next step.

3 A Feature-Based Debiasing Rating Aggregation Approach

3.1 A Motivating Synthetic Example - Political Debiasing

Assume we run a political news site, and have a set of articles along the political spectrum from liberal to conservative, as expressed by a uniformly random score $\rho \in [-1, 1]$, where one end is liberal and the other conservative. Independently, the article has a veracity score r, a measure of how objectively "true" or "good" the article is without partisanship.

Our users (as characteristic of America) are often polarized, as conveyed by a political bias score $b \in [-0.5, 0.5]$ where again one end is liberal and the other conservative. In order to capture the fact that this distribution of bias is non-uniform and in fact polarized, we model bias of a user i as drawn from a beta distribution $b_i \sim \beta(0.5, 0.5) - 0.5$, as shown in Figure 1.

The user i rates an article j with independent probability $p = 0.2$, and its rating w_{ij} of the article is distributed according to:

$$w_{ij} = min(max(r_j + \rho_j b_i + \epsilon, 0), 1)$$

where r_j is the articles "true" rating, ρ_j is its political bias score, b_i is the user's political bias, and $\epsilon \in N(0, -0.1)$ is random noise.

The intuition here is that a user will overrate an article that lies on the same side of the political bias spectrum (and hence $\rho_j b_i$ will be positive if both ρ_j and b_i are positive, i.e. both conservative, or both

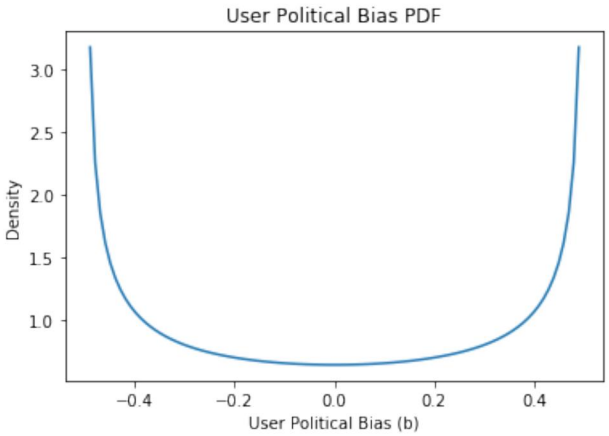

Figure 1: Distribution of User Political Bias. We see the that the probability mass of user bias is far higher along the extremes of the spectrum as a simulation assumption.

negative, i.e. both liberal), and conversely underrate if they lie on opposite ends of the political spectrum, and rate close to true veracity (with noise) if moderate.

Our task is to recover the true veracity r of the articles as closely as possible, given that we know the political feature of the article but not the users.

3.2 A Multi-Dimensional Extension

As a more general case, consider when a user has feature-specific biases. For example, if an article can be defined by political leaning along 3 dimensions: fiscal policy, social policy, foreign policy. A Stanford undergraduate student who claims to be "fiscally conservative but socially liberal" may have a negative bias against an article with a conservative score in. Formally, we can define an article's political bias feature vector as n-dimensional:

$$\rho \in R^n, \rho_i \sim unif(-1, 1)$$

Likewise a user's bias feature vector, and ratings are respexctively drawn from:

$$b \in R^n, b_i \sim \beta(-0.5, 0.5) - 0.5$$

$$w_{ij} = min(max(r_j + \rho_j \cdot b_i + \epsilon, 0), 1)$$

3.3 Approach

As mentioned in the related work section, Mishra's algorithm does not account for the case where users are not uniformly biased positively or negatively, but rather where user bias is different depending on the features of the entity being rated. Concretely, we extend Mishra's algorithm to function for the case where each $b_i \in R_k$, $b_{ij} \in [-1, 1]$, a bias of user i towards entity feature j.

In order to compute b_{ij}, we introduce a feature matrix S of the entities, where S_j is a k-dimensional vector, S_{jk} encodes the value that movie j has in the the k^{th} feature direction. The ratings are computed identically to Equation (1), with the bias per feature as a vector instead:

$$r_j^{t+1} = \frac{1}{\sum_{i=1}^{m} \mathbb{1}[w_{ij} \in R]} \sum_{i=1}^{m} \max(0, \min(1, w_{ij} - \alpha b_i^t S_j)) \mathbb{1}[w_{ij} \in R] \tag{3}$$

3.4 Results

We generate a synthetic dataset using the aforementioned data generation assumptions, with 1000 users and 200 articles. A user rates an article with probability p=0.2 is randomly assigned a 3-dimensional bias vector where each entry is drawn from $\beta(0.5, 0.5) - 0.5$. Each article is assigned a random 3-dimensional feature vector where each entry is drawn from $unif(-1, 1)$.

Figure 2 shows the MSE (compared to the synthetic "ground truth" veracity ratings) of our iterative algorithm compared to the baselines of mean and median aggregation.

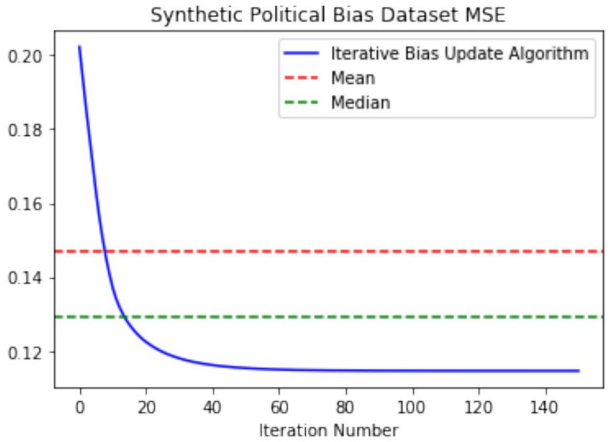

Figure 2: MSE of our Iterative Bias Update Algorithm: Taking the mean of all ratings yields an MSE of 0.147, Median yields 0.129, while our algorithm gives 0.115 as the MSE.

We see that our Iterative Bias Algorithm intelligently identifies the user feature-specific biases, and thus converges to a better MSE optimum than a simple Mean which will average all ratings without removing bias. We also notice that within the first few iterations, our algorithm converges quickly outperforms Mean MSE. On some runs, it takes a sharper downward slope, likely because there may be extremely sharp biases for some of the users such that given sparse data, the updates occur unstably. We add a momentum parameter that interpolates biases from iteration to iteration to produce smoother interpolation.

4 A Spectral Approach to Movie Embeddings

Our bias-detection and adjustment algorithm warrants the need for "movie embeddings", that is a mapping of movies into a k-dimensional space such that similar movies are close together. With the vanilla bias-detection algorithm, we define bias as a scalar rather than a multi-dimensional vector, but this can be problematic as bias might be defined over multiple dimensions rather than a singular dimension (scalar) and to do so, we need to modify the formula that uses a movie embedding rather than a scalar. As an example, suppose we defined the movie embedding as a vector of genres where 1 indicates if the movie is in that genre and 0 indicates otherwise. This is important because it will capture the user specific biases across genres in movies, if we use the altered multi-dimensional bias formula. However, genre isn't enough, there may be a relation across directors, cast, plot, and user preferences themselves so our desire is to make a movie embedding that captures more than just genre relationships so that our bias algorithm can incorporate those embeddings to define biases across different "factors" properly.

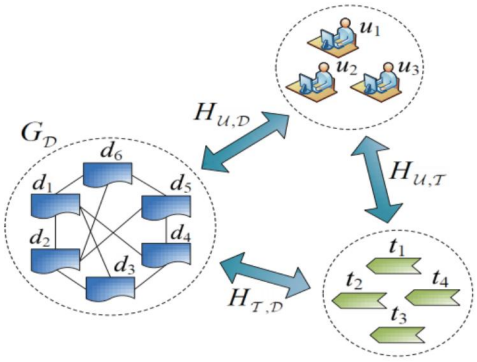

Figure 3: Original figure of Guan et. al's work that portray user, tags and movies as a circular relationship. $H_{U,D}$, $H_{T,D}$, $H_{U,T}$ are the adjacency matrices of the bipartite graphs of the user-movie, tag-movie, and user-tags respectively. G_D is the similarity matrix comparing each of the movies to themselves

4.1 Dataset for Movie Embeddings

For this problem, we used the dataset in the MovieLens paper used by GroupLens, a research lab in the Department of Computer Science at the University of Minnesota Twin Cities. The dataset consists of triplets of user, movie, and tag where it describes a user tagging a movie with some tag. There are 2113 users, 10197 movies, and 13222 tags, and 855598 ratings (which we normalize to be between 0 and 1). Also the sheer amount of tags we have may not be helpful for the algorithm so we will use the top 100 most popular tags instead to include more than just genre info but not too much as to make the clustering algorithm slow (as an extension of the Guan et. al algorithm).

Moreover, we join this data with Rotten Tomato Top Critic Ratings from another GroupLens Dataset, choosing those movies with at least 5 top critic ratings to use as a surrogate for ground truth unbiased ratings. The merits of this approach for generating evaluation data will be discussed later.

4.2 Spectral Clustering on User, Movies and Tags

We will perform spectral clustering on three types of matrices, one adjacency matrix on user to tags used across all movies, an adjacency matrix from users to movie tagged, and an adjacency matrix from movies to tags given to that movie and let's call them R^{ut}, R^{um}, and R^{tm} all of size $|U| \times |T|, |U| \times |M|$, and $|T| \times |M|$ respectively. We also define a $|M| \times |M|$ matrix that contains similarities between movies based on Jaccard similarity of genres. Including this in our cost function will prevent the cold-start issue based for recommendation services. Our cost function Q is defined as in terms of the matrices R^{ut}, R^{um}, R^{tm} and W, and let's define a $|U| \times k$, $|T| \times k$, and $|M| \times k$ matrices where $k = 1$: f, g, p. Say f_i will be the 1-dimensional embedding for user u_i, g_i the embedding for t_i, and p_i the embedding for m_i (here let's say $k = 1$ for math derivation and then generalize later). Our cost will be Q will be:

$$Q(\mathbf{f}, \mathbf{g}, \mathbf{p}) = \alpha \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} R_{ij}^{ut} (f_i - g_j)^2 + \beta \sum_{i=1}^{|T|} \sum_{j=1}^{|M|} R_{ij}^{tm} (g_i - p_j)^2 + \gamma \sum_{i=1}^{|U|} \sum_{j=1}^{|M|} R_{ij}^{um} (f_i - p_j)^2 + \frac{1}{2} \eta \sum_{i,j=1}^{|M|} W_{ij} (p_i - p_j)^2$$

Where $\alpha + \beta + \gamma + \eta = 1$. These represent sort of importance factors. Notice that these look very similar to the normal spectral clustering algorithm except it works on all graph types: users to tags, tags to movies, and user to movies. It also works on movies to movies. The reason we sum all of these up is because for us when generating thifs space when embedding users and movies and tags in the same space, for user to user similarity we need to consider how its tag usage compares with other user's tag usage and

how its movie-watching compares with other user's movie watching. Same idea with movies and tags, only movies has an additional affinity matrix W that also tries to make it such that movies that share a high Jaccard similarity on genre are embedded close together in the space, since we really care about the movie embeddings. This is more informed than just rating usage. Now the α, β, γ, and η tell us how important one particular relationship is. A high α dictates that we really care about embedding users and tags close together if they were used in the same context, and a high β dictates we care about embedding tags and movies close together if they appeared together in a user-tagging. That additional W constraint also constrains the fact that the movie embeddings should be somewhat similar based on genre.

We can simplify the math by defining D_{ut}, D_{um} and D_{tm} as a degree matrix of users in the user/tag graph, degree matrix of users in the user/movie graph, and degree matrix of tag in the tag/movie graph respectively. Similarly, D_{tu}, D_{mu} and D_{mt} as degree matrix of tags in the user-tag graph, degree matrix of the movies in the user-movie graph, and the degree matrix of the movies in the movie-tag graph. Also for the affinity matrix let D be the degree matrix of all the complete graph of movies (where connections are jaccard similarity on genre from one movie to another) and let $L = D - W$ be the Laplacian. We can simplify the term $\alpha \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} R_{ij}^{ut}(f_i - g_j)^2$ in terms of the degree matrix as follows:

$$\sum_{i=1}^{|U|} \sum_{j=1}^{|T|} R_{ij}^{ut}(f_i - g_j)^2$$

$$= \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} R_{ij}^{ut}(f_i^2 - 2f_i g_i + g_j^2)$$

$$= \sum_{i=1}^{|U|} D_{ii}^{ut} f_i^2 + \sum_{j=1}^{|T|} D_{jj}^{tu} g_j^2 - 2 \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} R_{ij}^{ut} f_i g_j$$

$$= \mathbf{f}^T \mathbf{D}^{ut} \mathbf{f} + \mathbf{g}^T \mathbf{D}^{tu} \mathbf{g} - 2\mathbf{f}^T (R^{ut}) \mathbf{g}$$

Similarly you can show that

$$\sum_{i=1}^{|T|} \sum_{j=1}^{|M|} R_{ij}^{tm}(g_i - p_j)^2 = \mathbf{g}^T \mathbf{D}^{tm} \mathbf{g} + \mathbf{p}^T \mathbf{D}^{mt} \mathbf{p} - 2\mathbf{g}^T (R^{tm}) \mathbf{p}$$

$$\sum_{i=1}^{|U|} \sum_{j=1}^{|M|} R_{ij}^{um}(f_i - p_j) = \mathbf{f}^T \mathbf{D}^{um} \mathbf{f} + \mathbf{p}^T \mathbf{D}^{mu} \mathbf{p} - 2\mathbf{f}^T (R^{um}) \mathbf{p}$$

$$\frac{1}{2} \sum_{i,j=1}^{|M|} W_{ij}(p_i - p_j)^2 = \mathbf{p}^T \mathbf{L} \mathbf{p}$$

Our $Q(\mathbf{f}, \mathbf{g}, \mathbf{p})$ will be:

$$\alpha(\mathbf{g}^T \mathbf{D}^{tu} \mathbf{g} - 2\mathbf{f}^T (R^{ut}) \mathbf{g}) + \beta(\mathbf{p}^T \mathbf{D}^{mt} \mathbf{p} - 2\mathbf{g}^T (R^{tm}) \mathbf{p}) + \gamma(\mathbf{f}^T \mathbf{D}^{um} \mathbf{f} + \mathbf{p}^T \mathbf{D}^{mu} \mathbf{p} - 2\mathbf{f}^T (R^{um}) \mathbf{p}) + \eta \mathbf{p}^T \mathbf{L} \mathbf{p}$$

We can divide by the sum of the squared L2-norms to eliminate scaling factor and if we define $\mathbf{h} = [\mathbf{f}^T \mathbf{g}^T \mathbf{p}^T]^T$ we can simplify it as the spectral clustering optimization problem of:
:

$$min_{\mathbf{f},\mathbf{g},\mathbf{p}} \frac{Q(\mathbf{f}, \mathbf{g}, \mathbf{p})}{\mathbf{f}^T \mathbf{f} + \mathbf{g}^T \mathbf{g} + \mathbf{p}^T \mathbf{p}} = min_{\mathbf{h}} \frac{\mathbf{h}^T \tilde{\mathbf{L}} \mathbf{h}}{\mathbf{h}^T \mathbf{h}} \;\; s.t. \, \mathbf{h}^T \mathbf{e} = 0$$

Where we define $\tilde{L}$ as the following:

$$\tilde{L} = \begin{bmatrix} \alpha\mathbf{D}^{ut} + \gamma\mathbf{D}^{um} & -\alpha\mathbf{R}^{ut} & -\gamma\mathbf{R}^{um} \\ -\alpha(\mathbf{R}^{ut})^T & \alpha\mathbf{D}^{tu} + \beta\mathbf{D}^{tm} & -\beta\mathbf{R}^{tm} \\ -\gamma(\mathbf{R}^{um})^T & -\beta(\mathbf{R}^{tm})^T & \beta\mathbf{D}^{mt} + \gamma\mathbf{D}^{mu} + \eta\mathbf{L} \end{bmatrix}$$

Now that we did it for $k = 1$ dimension we can do it for a general k. We will have a $\mathbf{F} = [\mathbf{f}_1\mathbf{f}_2...\mathbf{f}_k]$ where $\mathbf{f}_i$ is a column containing all the users coordinates in the i^{th} of the embedding. Same construction goes for $\mathbf{G}$ and $\mathbf{P}$. Our $\mathbf{H}$ will be defined as $\mathbf{H} = [\mathbf{h}_1\mathbf{h}_2...\mathbf{h}_k]$ where $\mathbf{h}_i = [\mathbf{f}_i^T\mathbf{g}_i^T\mathbf{p}_i^T]^T$. If we formulate all over again with a generic k according to Guan et. al we get optimization problem:

$$min_{\mathbf{H}} \frac{tr(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H})}{tr(\mathbf{H}^T\tilde{\mathbf{D}}\mathbf{H})} \;\; s.t. \; \mathbf{h}_i^T e = 0 \; \forall i$$

Where we define $\tilde{D}$ as a diagonal matrix of the diagonal entries of $\tilde{L}$. To solve this spectral clustering problem, we find the first k eigenvectors of $\tilde{L}$ and we set those as the k columns of H. So say we have a users, b movies, and c tags, that will mean the first a rows of $\mathbf{H}$ will be embeddings for the users, the next b rows of $\mathbf{H}$ will be embeddings for the tags and the last c rows will be the embeddings for the movies.

5 Movie Embedding Results

5.1 Movie Embeddings

Our bias-detection algorithm depends on how well our movie embeddings capture similarity and differences of movies across the new hyperspace we defined. We would like similar movies to be clustered together ideally and this similarity should be based on types of users who looked at it and how similar are those users, and types of tags assigned to it and how similar are those tags. This is why the spectral clustering similarity problem was long, it depends on this circular definition of similarity across these 3 attributes. Furthermore, there has to be some genre similarity across the movies, hence the inclusion of the affinity matrix in the spectral clustering algorithm, but it shouldn't be something that dictates the entire similarity algorithm since genres are too inclusive and could relate movies in the same genre that are only mildly similar.

To see how our spectral clustering algorithm worked, we compared visually seeing how it groups similar movies together and what the clusters represent, and how does it relate to and correlate with clustering movies based on Jaccard similarity by genre. We generated $k = 50$ dimensional embeddings using the spectral clustering algorithm above and we reduced it to a 2-dimensional space using TSNE, since PCA had less than admirable results due to non-linearity of the problem. We achieved the results shown in Figure 4.

We can see that there exist clusters but it appears more finely-grained than just by genre, since there are many more clusters than the 6 to 7 genres present in our dataset. After analyzing the labeled examples specifically, we can see they share tag similarities: Dawn of the Dead and its remake are tagged with "Romero" (director) and "Zombies" and Castle in the Sky and My Neighbor Totoro are tagged with "Hayao Miyazaki" (director) and "anime" and "Japan". Our Embedding is thus able to pick up these finer details and cluster based on tag data and some user data rather than solely genre. It can sometimes detect beyond genre as we can see in Figure 5.

Some of the titles are indeed in the same series like the Alien titles or the Dawn of the Dead series, but others are not so related but yet intuitively make sense. Doctor Zhivago and Vertigo are not related plot wise but have tags of "Classics" or "IMDB 250" that place them in that same category of classic movies that are similar to each other in ways other than plot and genre. Harry Potter was most related to The Vampire Chronicles even though there are 0 shared genres because they have tags like "magic", "fantasy", and "based on a book". Ratatouille and Finding Nemo have a shared genre, but have tags of "Pixar" which is why they are so similar to each other. This and probably the fact that similar types of users (anime fans,

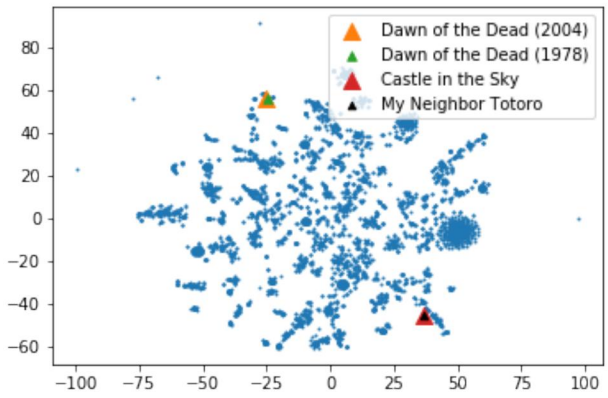

Figure 4: TSNE Reduction of embeddings onto 2-dimensional space. Clusters are formed and are more finely grained than number of genres

Movie	**Most Similar Movie**	**Shared Genres**
Dawn of the Dead (1978)	Dawn of the Dead (2004)	Action, Horror
Aliens (1986)	Alien: Resurrection (1997)	Action
The Vampire Chronicles	Harry Potter and the Philosopher's Stone	None
Castle in the Sky	My Neighbor Totoro	Animation, Children
Doctor Zhivago	Vertigo	Drama
Ten Commandments	The Last Temptation of Christ	Drama
Ratatouille	Finding Nemo	Children

Figure 5: Table showcasing Movie and its most similar counterpart. Sometimes the most similar movies are in the same series or not related at all but are remotely

horror fans, etc.) also watch both of these contribute to the quality of the clustering algorithm.

We also want to test how the algorithm works compared to genre similarity and if there is a correlation between the two, which would serve as a good metric for whether or not our clustering algorithm works when the genre-similarity algorithm works.

Figure 6 shows us performing some hyperparameter searching on values of k and β from out spectral clustering formula, which control dimensionality of embedding and how closely strictly clustered do document and tag embeddings need to be respectively. Our results were actually somewhat surprising. With low values of k we can see that there was no correlation with genre similarity at all, which means it underfitted and didn't capture enough information about the user-tag-movie graphs. However with higher k, around $k = 50, 200$ we see some improved correlation especially high k having no less than a Pearson Correlation Coefficient of 0.27 with the genre similarity and reaching a high of 0.4. What's surprising is that we were able to reach a somewhat high correlation coefficient of 0.392 with $\beta = 0.1, k = 50$. We assumed that a low β would somehow ruin the embeddings of movies since it doesn't care about embedding movies and tags close together; however, this is not the case. We can infer that having a low β wouldn't necessarily hurt the algorithm since now it will rely more on the α and γ variables now which weigh user-movies and user-tag relationships. It is possible that movies can be closely embedded together by seeing which users liked the same movie and related users based on which tags were used by the same users and used that to infer similarity of movies. That said, there is probably a sweet spot somewhere in here, that we intend to do in future work, but having this much information of the 3 bipartite graphs show why β value may not necessarily matter as much as we think.

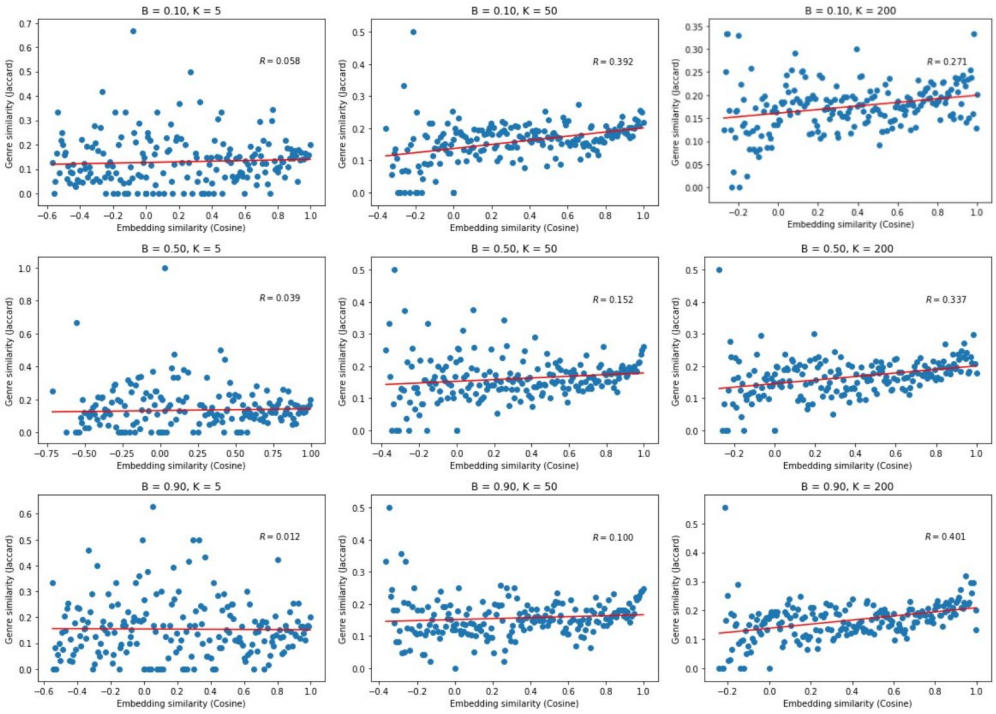

Figure 6: Test values of $\beta = 0.1, 0.5, 0.9$ and $k = 5, 50, 200$, High k performs better than low k, but it looks like β wasn't as influential since we significant correlation on each value

5.2 Embedding-Based Bias Reduction Results

We also assess the quality of our embeddings by seeing how well it performs as a set of features to our feature-specific bias-reduction algorithm. We take same approach as 3.3, notably feature-specific bias reduction, with the embedding K-dimensional vector for each movie as our set of features, and a corresponding dimension K for the user bias vectors. We compare how well our embedding did at very values of K versus simply defining our feature vector as a sparse genre vector, against the baselines of mean and median. Figure 7 shows a visualization of the results of our algorithm.

We see that as expected, when $k = 50$ our algorithm takes longer to converge, given that the dimension of the bias vector is larger. However, it converges to about the same MSE as $k = 20$, perhaps suggesting that the additional semantic information obtained by higher dimensionality is offset by sparsity issues in the dataset, particularly given that the number of ratings per user follows a roughly power law distribution, so many users have very few ratings. This means that for many users it is hard to uncover the biases along each dimension. However, it is clear that both embeddings perform noticeably better than using a sparse genre feature vector, confirming to us that our embeddings indeed do communicate more semantic information than genre alone.

Unfortunately, none of our feature sets as applied to the iterative algorithm perform better than taking the mean alone. There are several possible reasons. Firstly, what constitutes "ground truth rating" may itself be biased, given that even top critics may suffer from preferring certain styles or people in film more than others. Secondly, it is possible that there exists causal correlation between top critic and average user ratings, that each one influences the other in practice given that top critics come to conclusions presumably taking audience reaction into mind. Finally, there may be significant noise in the data not captured by our embeddings, from user interface issues to unaccounted relationships in the data such as hidden correlations between frequency and scores of ratings, which an average approach does a better job accounting for.

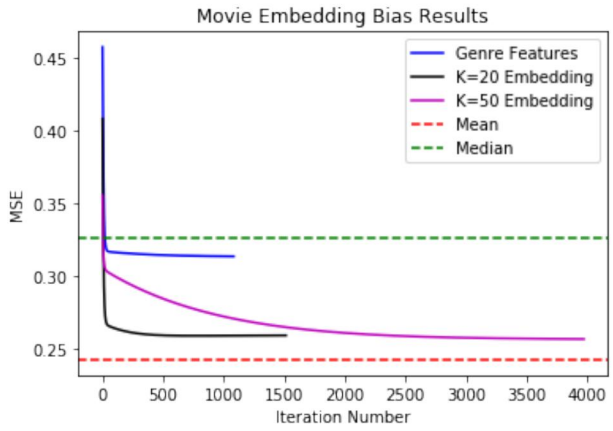

Figure 7: Predicted True Movie Ratings Compared to Top Critic Ratings by Feature Set. Genre Features' MSE was 0.313, K=20 Embedding's MSE was 0.259, K=50 Embedding's MSE was 0.256

6 Conclusion

Our paper models the fact that we are human and have subjective preferences, and challenge supposedly aggregation systems that depend on users ratings that could always hold some inherent bias or the other. Many users may dislike an object purely based on their own principles when it might be not as bad as others might think (e.g. a science film rated by an extremely religious person would show some bias if it conflicts with their beliefs). Thus to eliminate bias we pull from both Guan et. al's and Mishra's work to first find a way to create movie embeddings to cluster similar movies together and then to use those movie embeddings with Mishra's work to generate user bias-vectors rather than a single bias-scalar to see how user biases are mapped across in a high-dimensional space. To uncover true rating, we take the initial rating and subtract the bias vector dotted with the movie embedding and use the relative ordering of the user's true ratings as a list and compare them to critic list and see how many of them are in the same spot (since we consider critic as an unbiased authority). We obtained results that show that the mean of the true ratings doesn't perform better than the mean of the initial ratings, and that could be because we compare it to the critic ratings order which also could be biased itself (critics might prefer snobbier art-films by Darren Aronofsky over generic action-packed films whereas the general audience might like the opposite), in addition to the other reasons mentioned in the aforementioned section.

For future direction, there are two possible routes, extending the bias algorithm or improving the movie embeddings. The movie embeddings right now work well since the spectral clustering algorithm considers the user-movies, user-tags, and tags-movie bipartite graphs which give much more information than normal collaborative-filtering. However, there are so many hyperparameters to choose from, such as the weighing coefficients $\alpha, \beta, \gamma, and \eta$. Right now we only experimented with β and k (dimension of embedding) a hyperparameter search over all of these hyperparameters might prove useful in finding the sweet spot for finding best spectral clustering algorithm for best movie similarities. Moreover, robust outlier detection and removal algorithms should be considered, as for example a manual inspection of the data suggests that there are many users who only give perfect ratings to movies.

6.1 Acknowledgments

Of note, Mishra's algorithm was implemented for a similarly motivated final project for CS269I, which focused more on various different algorithms for aggregation. All the content regarding Movie Embedding creation, feature similarity, and the application of User-Product Feature specific aggregation was created for CS224W.

References

[1] Grouplens research. https://grouplens.org/datasets/movielens/. Accessed: 2018-12-9.

[2] A normal-distribution based reputation model. http://folk.uio.no/josang/papers/AXJ2014-TrustBus.pdf. Accessed: 2018-10-28.

[3] Joel Grover and Amy Corral. Don't fall for fake reviews online. www.nbclosangeles.com/news/local/Fake-Reviews-on-Yelp-Facebook-Google-447796103.html, 2017.

[4] Ziyu Guan, Can Wang, Jiajun Bu, Chun Chen, Kun Yang, Deng Cai, and Xiaofei He. Document recommendation in social tagging services. http://people.cs.uchicago.edu/~xiaofei/www2010-guan.pdf, 2010.

[5] Abhinav Mishra. An approach towards debiasing user ratings, 2016.