# Customized Reviews for Small User-Databases using Iterative SVD and Content Based Filtering

Jon Gregg
School of Computational Science and Engineering
Georgia Tech
Atlanta, Georgia 30332
jgregg6@gatech.edu

Nitin Jain
School of Computational Science and Engineering
Georgia Tech
Atlanta, Georgia 30332
njain63@gatech.edu

## ABSTRACT

Recommender systems have proven to be a valuable tool for web companies like Amazon and Netflix for attracting and maintaining a large user base. However, in situations when user data is more scarce (e.g., for mid-sized companies, or an online retailer testing a new ratings system) algorithms tailored to smaller datasets can be used to further increase accuracy. This paper explores the potential of combining collaborative and content-based (using user comments) filtering algorithms using Yelp.com[1] data from a single city. We present the method to combine two approaches, and find that the MSE for predicting a user's new rating can be reduced from a baseline MSE of 1.744 to 0.934 given just 2500 rated items in our real-world dataset.

## Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Applications**]: Data mining

## General Terms

Algorithms, Experimentation, Theory

## Keywords

Machine Learning, Recommender System, Dimensionality Reduction, Content-Based Filtering

## 1. INTRODUCTION

Recommender systems provide users with customized recommendations based on what the user has previously purchased/liked. Pandora Radio, for example, has used its Music Genome Project[17] recommender system to build a 50+ million user base. Larger websites like Amazon, YouTube, and Netflix use custom recommendations to increase sales and/or the time users spend on their websites. Netflix famously held an open competition called the Netflix Prize[4] in which it awarded one million dollars to the team that improved Netflix's prediction algorithm by over 10%. Most large-scale systems use a hybrid approach combining collaborative and content-based filtering.

These websites have implemented ensemble algorithms built for big data, but there still exists a need for novel and intuitive customized recommender systems tailored to more commonly-sized customer ratings datasets. This paper borrows the popular methods from big data recommendation and modifies them to increase accuracy and allow for a semantic analysis component.

The most common method of obtaining customized recommendations is through compression. Various modifications of Singular Value Decomposition or Principal Components were used by the top Netflix finishers. Each method takes the full ratings dataset and compresses it to a much smaller rank, which is vaguely similar to the idea of taking points in $\mathbb{R}^2$ and 'compressing' them onto a least squares regression line, except these versions achieve compression over multiple dimensions.

This paper also incorporates comment data that were not available in the Netflix prize but are often available on ratings or retail websites. In the case of a restaurant recommendations website, semantic meanings within the user comments reveal factors that affect how a person values a restaurant. The term Opinion Mining is generally used to describe this method, and its importance has increased in recent years as more retailers have developed an online presence that allows users to post reviews or comments.

The recommendation system in this paper is a hybrid system that uses both of these methods. The collaborative filtering algorithm is based on iterative SVD, and the content-based filtering algorithm uses opinion mining to extract information from user comments.

Section 2 explains the data used in this paper and related work in the field. In Section 3, the Iterative SVD and content-based filtering approaches are explained. Section 4 describes the experimental results, with references following.

## 2. DATA AND RELATED WORK

Yelp - the internet's largest business directory service that is most famous for its users' restaurant reviews - does not offer customized recommendations. Regardless of a user's preferences (that might be gathered from a user's previous ratings, comments, or clicks), each user sees only the aggre-

[1]http://www.yelp.com/


*SNAKDD 2013* Chicago, Illinois USA

gated user rating for each restaurant. Yelp ratings and user comments for every restaurant in Atlanta were obtained for use in this project.

When reviewing a restaurant, each user supplies a rating from 1-5 stars in 1-star increments, along with a comment. For the purposes of testing the recommender system, only users with 9+ recommendations were retained for prediction (allowing for user ratings to be split into a train/test set), resulting in a dataset of 1538 restaurants, 2386 users, and 78,136 comments. This allowed for the creation of a 'ratings matrix' of size 1538 restaurants x 2386 users that contained the user's rating for that restaurant if it exists, or a 0 if the user had not yet rated the restaurant. Futher, a 'restaurant-comments matrix' of size 1538 x 78136 and a 'user-comments matrix' of size 2386 x 78136 were used in the content-based filtering.

These datasets are of reasonable size for a business that has generated a user base and is experimenting with the benefits of customized recommendations. These businesses could be in either of the following situations:

- a startup web company that is using customized recommendations to differentiate itself from its competitors
- an online retailer with a ratings system in place, and would like to test customized recommendations in a certain region before rolling the system out globally.

There are two 'state of the art' groups of restaurant recommendation services. The first group consists of the large websites - Yelp, Urbanspoon, Zagat - that have hundreds of ratings for popular restaurants, and show a user the aggregated rating (often the rounded mean) of each restaurant. The drawback with these services is that, when a user requests a list of restaurants in the nearby area, the returned list will not factor in that user's tastes, and as a result the user will have to look harder (or elsewhere) to find the less-popular restaurants that fit his/her tastes. For example, if a user loves vegetarian food, she will have to search for that term herself instead of getting those recommendations automatically. More importantly, because there are relatively few popular vegetarian restaurants in any city, she will likely be shown a list of less-popular restaurants, and fewer reviews means higher variance / harder to trust. Note that, a customized recommender that has knowledge of which users gave a certain review to a less-popular restaurant (and if some of those users have tastes that match this particular user) this variance can be reduced.

The second group is led by Ness, the current standard in smartphone apps, which uses machine learning to lower the error rate of user recommendations. The drawback of Ness is that its goal is to reduce the error rate, and in a city like Atlanta where only a couple dozen restaurants have elite ratings (with a high number of ratings), a lower error rate can often be achieved by recommending those top restaurants regardless of user taste. As a result, unless a user supplies Ness's recommender with a large number of ratings, Ness's results are similar to Yelp's.

## 2.1 Methodology

The algorithms chosen for this paper have their roots in previous research, but have been modified for use in smaller datasets. For example, Simon Funk's iterative SVD algorithm[6] was used by the winning Netflix Prize team (along with Funk himself, who placed top-10 at the Progress Prize stage), but is tailored to big data in that features are trained one at a time, resulting in better scalability but lower accuracy. This paper's recommender system incorporates the computational efficiency benefits of the iterative algorithm, but trains the features together for even quicker runtime speeds and greater accuracy. Also, while opinion mining is a common task in the field of semantics, this system uses an intuitive approach that generates a surprisingly strong prediction accuracy while also serving as a proof of concept for more complex, non-intuitive models that stress pure prediction. We combine user-comment opinion mining and variable reduction techniques to estimate a user's preferences on a variety of restaurant characteristics.

## 2.2 Iterative SVD

Simon Funk's iterative singular value decomposition algorithm created the foundation for this paper's collaborative filtering algorithm. On his blog, Funk explained the purpose of using an SVD[7] in prediction by saying: "If meaningful generalities can help you represent your data with fewer numbers, finding a way to represent your data in fewer numbers can often help you find meaningful generalities. Compression is akin to understanding." The theory behind Singular Value Decomposition is not covered in this paper (see Sarwar (2000)[14] for early research into the benefits of using SVD over standard collaborative filtering in a recommendation engine, which can include faster performance), but below is some basic intuition behind Funk's algorithm.

Given an original matrix of size (a x b) and a given rank k of the compressed matrix, Funk's algorithm generates one matrix of size (a x k) and another of size (k x b) that, when multiplied, results in the lowest-MSE estimate of the original matrix. That is, the two matrices that result from Funk's algorithm represent modificiations of the U and V matrices of an SVD that have information from the square S matrix built in. In the Yelp dataset, with input ratings matrix of size 1538 restaurants x 2386 users, Funk's algorithm with k = 20 will result in one matrix of size 1538 x 20, and another of size 20 x 2386. This method provides several benefits for creating customized recommendations:

- The modified U matrix of restaurants x 20 features allows for the calculation of a similarity rating between restaurants, as the matrix values can be thought of as representing how a restaurant is rated by a certain 'dimension' of the user base.

  Typically, a Cosine similarity is used in an SVD because each successive feature accounts for a smaller amount of variance in the model. As a result, in a 2-feature example, two restaurants with a similar ratio of the first feature to the second are likely more similar than two restaurants that have a closer Euclidean distance (which can result from having similar coefficients for the second, less-important feature that is incorrectly given equal importance in a typical distance calculation). These similarity ratings can be used for clustering purposes, and one potential benefit of this similarity index is described in the data visualization of the Results section.

- Similarly, the modified V matrix of users x 20 features allows us to calculate a similarity rating between users.

- Given a user's feature values for the 20 features (trained from previously-made comments), predicting how user $i$ will like any restaurant $j$ in the dataset simply requires multiplying $\mathbf{U_i V_j'}$

An iterative method like Funk's must be used in place of an actual SVD (which could be run on a dataset of this size, albeit at a higher runtime) for two reasons. First, the '0' entries in the ratings matrix correspond to an unknown rating, and not a rating of 0. As a result, an SVD would take two users' 0 ratings for the same restaurant as evidence of a similarity, when in reality two 0 ratings tells us nothing about their similarity. Second, the factored matrices of an SVD, when multiplied, represent the k-rank matrix with the smallest possible mean squared error when compared to the original matrix. But in this example, the '0' values should be ignored from the MSE minimization problem. This can be done easily with a iterative SVD that ignores '0' in its cost and gradient functions.

The theory behind the iterative SVD follows. Given $i$ restaurants and $j$ users, the resulting ratings matrix is denoted by $R \in \mathbb{R}^{ixj}$, the restaurant-feature matrix is $F^{(r)} \in \mathbb{R}^{ixk}$, and the user-feature matrix $F^{(u)} \in \mathbb{R}^{kxj}$, where k is the rank of the feature matrices produced by the iterative SVD. The first feature vector in both feature matrices begin with default values of random numbers between 0 and 1.

The product $F^{(r)}F^{(u)} \in \mathbb{R}^{ixj}$ is the set of 'predictions' for the ratings matrix. At first, these are just random predictions, but with each iteration the MSE of (actual ratings minus predicted ratings) decreases until a minimum is reached. The overall cost of these predicted values is

$$C = \sum_i \sum_j (R_{i,j} - F^{(r)}F^{(u)})^2.$$

Therefore, for each rating $(i, j)$, the change in the cost with respect to one feature value $f \leq k$ for one user is the partial derivative

$$\frac{\partial C}{\partial F^{(u)}_{f,i}} = -(2C)F^{(r)}_{j,f}$$

Similarly, the change in the cost with respect to one feature value $f < k$ for one restaurant is the partial derivative:

$$\frac{\partial C}{\partial F^{(r)}_{j,f}} = -(2C)F^{(u)}_{f,i}$$

An optimum can then be found through gradient descent with the update equations shown below (with $\alpha$ denoting the learning rate).

$$F^{(u)}_{f,i} \leftarrow F^{(u)}_{f,i} + \alpha * (2C)F^{(r)}_{j,f}$$

$$F^{(r)}_{j,f} \leftarrow F^{(r)}_{j,f} + \alpha * (2C)F^{(u)}_{f,i}$$

Note that these updates should be made simultaneously for each non-zero rating in the ratings matrix. Therefore, it is important when coding the algorithm to save $F^{(u)}_{f,i}$ into a temporary variable before making the first update, and then using that temporary value in the $F^{(r)}_{j,f}$ update equation.

Once every non-zero rating is iterated through, the entire iteration process is repeated in a while loop until the change in cost is less than some pre-specified threshold. Once a minimum is reached, the feature values are saved. From there, another feature of random values is then added, and the entire iterative process is repeated until that feature is trained. This process continues until all k features have been trained.

Funk's method of 'greedily' training one vector at a time was devised for use on a dataset of 8 billion entries, and allowed for the entire recommender system to run in around two hours. However, in the same way that greedy training of a decision tree results in a 'good' but often non-optimal set of splits, greedy training will often achieve a lower prediction accuracy than training all the features at once - a process that can easily complete on a dataset of this project's size.

As a result, this paper proposes starting the training process by filling all values of $F^{(r)}$ and $F^{(u)}$ with random numbers, then updating all features at each MSE-lowering step. This has led to an average reduction in prediction MSE of around 3% when compared to the greedy algorithm in tests on this dataset.

Further, the improved feature value estimates allow for the creation of a more accurate and intuitive similarity index for each user. When the greedy algorithm is used, each feature will be more important than any feature that follows it, and it is often difficult to determine how much more important. The result is that Cosine Similarity, which merely compares ratios of feature values, is typically used to compare features, meaning some of the distance information is ignored. But when every feature is trained simultaneously the resulting features are of equal importance, and therefore a simple L2 norm of the U matrix can be used to find similarity in a more intuitive (and often more accurate) way.

## 2.3 Content-based Filtering

Recommenders that use content-based filtering examine how a user has previously rated items with similar content. For example, if a user has shown a preference for action movies starring a certain actor, then a good recommendation system should show the user all other actions movies starring that actor, along with similar movies in that genre. More formally, while a collaborative filtering system examines the correlation between two users with similar preferences, a content-based system will examine the correlation between an item's content and a user's rating on items with similar content. [16]

It should be noted that "content" can come in many forms. In the Yelp dataset and many others, content can be found in a business's characteristics and in user comments, and that information can be linked with a user's rating. The music streaming service Pandora simply uses 'like' and 'dislike' buttons to gather information from users and finds common characteristics between liked and disliked songs. Two early works that combined a collaborative filtering element with content based filtering include Fab (Balabanovic, et al.)[2], a web recommendation engine from the 1990s that generated recommendations using content from each page (as part of a hybrid system) and trained the algorithm further by asking users to rate each recommendations; and a reading recommender by Woodruff et al. (2000) [18] that combined text and bibliography information with documents a user has already chosen to read to recommend other documents to the user.

### 2.3.1 Sentiment Analysis

For this problem, the greatest source of information about

the content of restaurants on Yelp is the user reviews section. Sentiment analysis has been explored [11] primarily with respect to text classification and categorization for documents like news articles Argamon-Engelson et al.[1] or movie reviews Pang et al. [12]. Most of the research is focused towards a binary classification of textual information with respect to people's opinion (positive or negative), which suggests the use of Naive-Bayes classifier and Latent Semantic Analysis approaches. [13, 9, 8]

### 2.3.2 Problem Definition

The problem of rating restaurants is an ordinal regression problem where ratings are predicted for restaurants on a scale of 1-5. A similar problem for movie reviews was explored by Basilico and Hofmann[3], where a joint perceptron ranking model was used based on the PRank algorithm by Crammer and Singer[5]. An implementation by Snyder and Barzilay[15] to predict restaurant recommendations extended this joint ranking model by attempting to capture contrastive differences among a single user comment and introduced an agreement model in addition to the JRank model.

In the context of this problem, unigrams were extracted from user comments on the Yelp website and employed as features for a gradient descent algorithm. (As observed in the study by Basilico and Hofmann [3], incorporating bigrams in the model does not seem to make a considerable difference in the results.) The features were selected based on a combination of the word's popularity, the authors' intuition for their importance in defining not only the characteristics of a restaurant, and a user's preferences for what she values in a restaurant. Secondary interpretations of a word's meaning based on context in a single user comment were not accounted for, although the number of possible occurrences were minimized by the words chosen. Further, the features that are selected seldom appear together in a single comment.

Table 1 shows the top 15 words that were extracted from the user reviews and chosen as features to train the prediction function in the gradient descent algorithm, along with their frequency of occurence in all of the comments. The intuition behind including the word 'burger' for example, is that a restaurant might not be known as a burger restaurant, but if the average rating for comments including the word 'burger' was noticeably high, then users who enjoyed burger restaurants should know about it (and apart from obtaining a dataset of each restaurant's menus, the authors know of no other way to get this information outside of user comments). Similarly, a restaurant that rates highly in 'atmosphere' should be recommended to users who place a high value on a restaurant's atmosphere.

### 2.3.3 Gradient Descent

Least-squares gradient descent [10] was used to generate a prediction function. The cost function is below.

$$\frac{1}{2}\sum_{z:r=1}((\Theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(\sum_{i=1}^{n_m}(x_k^{(i)})^2 + \sum_{i=1}^{n_u}(\Theta_k^{(j)})^2)$$

where $\Theta$ is updated according to the following equation:

$$\Theta_k^{(j)} = \Theta_k^{(j)} - \alpha\sum_{i:r=1}(((\Theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\Theta_k^{(j)})$$

Here, $z$ is an entry $(i, j)$ and $r$ is a vector such that $r(i, j) = 1$ when a user has rated a restaurant, and 0 otherwise. For this problem, the $x$ restaurant-ratings matrix is of dimension (numRestaurants x numFeatures) and the $\Theta$ user-ratings matrix is of dimension (numUsers x numFeatures).

**Table 1: Frequency of top 15 selected features**

| | | | | | |
|---|---|---|---|---|---|
| service | 22362 | atmosphere | 8580 | price | 3344 |
| chicken | 13373 | friends | 7936 | spicy | 2494 |
| night | 11295 | parking | 7644 | quality | 1873 |
| burger | 10386 | sushi | 5446 | dessert | 1865 |
| fresh | 9840 | wine | 5098 | patio | 1513 |

Bayesian averages were used to fill each element of the $x$ and $\Theta$ matrices to reduce the effect of a small but extreme number of votes for one particular feature. For example, if a user has mentioned 'sushi' three times and gave a 5-star rating each time, that certainly does not mean the user likes all restaurants that offer sushi. Instead, this system interprets Yelp ratings to be m a sample from the population of a user's possible ratings for all restaurants, and so the Bayesian average shows merely that a user prefers sushi restaurants instead of assuming that user will give a perfect 5-star rating for anything sushi-related.

Singular value decomposition was used to compress the number of features, as many of the feature words are intuitively similar ('ambiance' and 'atmosphere', for example). The resulting compressed feature matrix therefore describes a restaurant (or user's preferences for the $\Theta$ matrix) on several different categories instead of single words.

## 3. RESULTS AND CONCLUSIONS

Each of Yelp's restaurant pages displays an average rating for all user ratings for that restaurant. While this is hardly a prediction of a future rating, it can serve as a lower bound baseline MSE for comparison purposes. Using these average ratings, the 'prediction MSE' on a test dataset was 1.744 stars$^2$. In comparison, the test MSE for the iterative SVD was 0.993 stars$^2$ (learning rate $\alpha = .0001$, k = 20 features), and the test MSE for the content-based filtering was a similar 0.998 stars$^2$ (learning rate $\alpha = 0.0001$, regularization term $\lambda = 90$). When combined using a coefficient for each of the two prediction matrices and trained via a simple grid search algorithm, the test MSE reduced to 0.934 stars$^2$, a significant improvement over either method and especially over the lower bound baseline despite having only 2386 users in the model, with ratings scattered over 1538 restaurants.

For an additional proof of concept, a recommendation visualization comparing recommended restaurants and similar user's ratings for those restaurants was extracted from the output of this recommender system, and is shown in Figure 1 on the following page. A dataset with a larger ratio of users to restaurants would allow for a more telling visual for all recommendations, but for this problem, a heatmap visualization still provided strong results for the most popular restaurants. Even with scarce data, this can provide insight into what's possible beyond just a simple recommendation.

Adding a visualization like this to a website or an app, and linking the similar users' ratings directly to that user's comments, allows for customers to interact with the recommender system, as opposed to just viewing a suggestion.

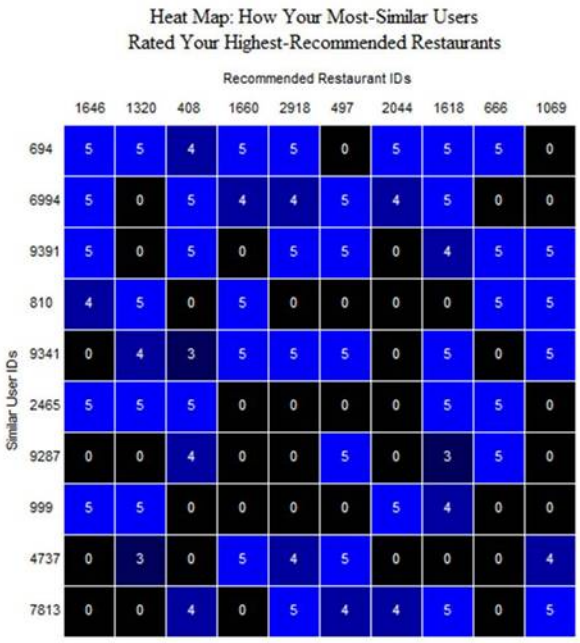


**Figure 1: (Referenced in the results and conclusions section on the previous page.) Heat map showing most-similar users on the vertical axis, highest-recommended restaurants for a specific user on the horizontal axis, and the ratings by similar users for those recommended restaurants within each cell.**

Note that the similar users (listed on the y-axis) were found using the user-ratings matrix from the iterative SVD algorithm. The adjustment made to Funk's iterative algorithm - training every dimension simultaneously - allowed for a simple L2 norm to be used to find user-similarity.

## 4. ACKNOWLEDGMENTS

The authors would like to thank Dr. Polo Chau of the Georgia Institute of Technology for his help in editing this text.

## 5. REFERENCES

[1] S. Argamon-Engelson, M. Koppel, and G. Avneri. Style-based text categorization: What newspaper am i reading. In *Proc. of the AAAI Workshop on Text Categorization*, pages 1–4, 1998.

[2] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *CACM*, 40(3):66–72, 1997.

[3] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9. ACM, 2004.

[4] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[5] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[6] S. Funk. Netflix update: Try this at home, 2006. *URL http://sifter. org/˜ simon/journal/20061211.html*, 2011.

[7] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHUP, 2012.

[8] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[9] K. Lang. Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning*. Citeseer, 1995.

[10] A. Ng. Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

[11] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[12] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86. Association for Computational Linguistics, 2002.

[13] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems, a case study. *ACM WebKDD Workshop*, 2000.

[15] B. Snyder and R. Barzilay. Multiple aspect ranking using the good grief algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL)*, pages 300–307, 2007.

[16] R. Van Meteren and M. Van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, 2000.

[17] T. Westergren. The music genome project. *Online: pandora.com/about/mgp*, 4(25):07, 2007.

[18] A. Woodruff, R. Gossweiler, J. Pitkow, E. H. Chi, and S. K. Card. Enhancing a digital book with a reading recommender. In *Proc. of ACM CHI 2000 Conference on Human Factors in Computing Systems*, pages 153–160, 580. ACM, 2000.