International Journal of Innovative Technologies
WWW.IJITECH.ORG
ISSN 2321-8665
Volume.06, Issue.01,
January-June, 2018,
Pages:0105-0110

Item-Based Collaborative Filtering Recommendation Algorithms

KOTHURI SRAVYA SRI POORNIMA[1], N. VIJAYA KUMAR[2]
[1]PG Scholar, Dept of CSE, LITAM, Dhullipalla, Guntur, AP, India.
[2]Assistant Professor, Dept of CSE, LITAM, Dhullipalla, Guntur, AP, India.

Abstract: Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success on the Web. The tremendous growth in the amount of available information and the number of visitors toWeb sites in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations, performing many recommendations per second for millions of users and items and achieving high coverage in the face of data sparsity. In traditional collaborative filtering systems the amount of work increases with the number of participants in the system. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems. To address these issues we have explored item-based collaborative filtering techniques. Item based techniques first analyze the user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users. In this paper we analyze different item-based recommendation generation algorithms. We look into different techniques for computing item-item similarities (e.g., item-item correlation vs. cosine similarities between item vectors) and different techniques for obtaining recommendations from them (e.g., weighted sum vs. regression model). Finally, we experimentally evaluate our results and compare them to the basic k-nearest neighbor approach. Our experiments suggest that item-based algorithms provide dramatically better performance than user-based algorithms, while at the same time providing better quality than the best available user-based algorithms.

Keywords: K-Nearest Neighbor Approach, ACM, Mean Absolute Error (MAE), CF Algorithms.

I. INTRODUCTION

The amount of information in the world is increasing far more quickly than our ability to process it. All of us have known the feeling of being overwhelmed by the number of new books, journal articles, and conference proceedings coming out each year. Technology has dramatically reduced the barriers to publishing and distributing information. Now it is time to create the technologies that can help us sift through all the available information to find that which is most valuable to us. One of the most promising such technologies is collaborative filtering. Collaborative filtering works by building a database of preferences for items by users. A new user, Neo, is matched against the database to discover neighbors, which are other users who have historically had similar taste to Neo. Items that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice, and in both information filtering applications and E-commerce applications. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems. The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual users for whom the site has large amounts of information. For instance, if a site is using browsing patterns as indications of content preference, it may have thousands of data points for its most frequent visitors.

These “long user rows” slow down the number of neighbors that can be searched per second, further reducing scalability. The second challenge is to improve the quality of the recommendations for the users. Users need recommendations they can trust to help them find items they will like. Users will” vote with their feet” by refusing to use recommender systems that are not consistently accurate for them. In some ways these two challenges are in conflict, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simultaneously so the solutions discovered are both useful and practical. In this paper, we address these issues of recommender systems by applying a different approach–item-based algorithms. The bottleneck in conventional collaborative filtering algorithms is the search for neighbors among a large user population of potential neighbors. Item-based algorithms avoid this bottleneck by exploring the relationships between items first, rather than the relationships between users.Recommendations for users are computed by finding items that are similar to other items the user has liked. Because the relationships between items are relatively static, item-based algorithms may be able to provide the same quality as the user-based algorithms with less online computation.

II. RELATED WORK

In this section we briefly present some of the research literature related to collaborative filtering, recommender systems, data mining and personalization. Tapestry is one of the earliest implementations of collaborative filtering-based recommender systems. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. However, recommender system for large communities cannot depend on each person knowing the others. Later, several ratings-based automated recommender systems were developed. The GroupLens research system provides a pseudonymous collaborative filtering solution for Usenet news and movies. Ringo and Video Recommender are email and web-based systems that generate recommendations on music and movies respectively. A special issue of Communications of the ACM presents a number of different recommender systems. Other technologies have also been applied to recommender systems, including Bayesian networks, clustering, and Horting. Bayesian networks create a model based on a training set with a decision tree at each node and edges representing user information. The model can be built off-line over a matter of hours or days. The resulting model is very small, very fast, and essentially as accurate as nearest neighbor methods. Bayesian networks may prove practical for environments in which knowledge of user preferences changes slowly with respect to the time needed to build the model but are not suitable for environments in which user preference models must be updated rapidly or frequently.

Clustering techniques work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each users with partial participation in several clusters. The prediction is then an average across the clusters, weighted by degree of participation. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithms [6]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller. Clustering techniques can also be applied as a "first step" for shrinking the candidate set in a nearest neighbor algorithm or for distributing nearest-neighbor computation across several recommender engines. While dividing the population into clusters may hurt the accuracy or recommendations to users near the fringes of their assigned cluster, pre-clustering may be a worthwhile trade-off between accuracy and throughput. Horting is a graph-based technique in which nodes are users, and edges between nodes indicate degree of similarity between two users. Predictions are produced by walking the graph to nearby nodes and combining the opinions of the nearby users. Horting differs from nearest neighbor as the graph may be walked through other users who have not rated the item in question, thus exploring transitive relationships that nearest neighbor algorithms do not consider. In one study using synthetic data, Horting produced better predictions than a nearest neighbor algorithm.

Schafer et al., present a detailed taxonomy and examples of recommender systems used in E-commerce and how they can provide one-to-one personalization and at the same can capture customer loyalty. Although these systems have been successful in the past, their widespread use has exposed some of their limitations such as the problems of sparsity in the data set, problems associated with high dimensionality and so on. Sparsity problem in recommender system has been addressed. The problems associated with high dimensionality in recommender systems have been discussed, and application of dimensionality reduction techniques to address these issues has been investigated. Our work explores the extent to which item-based recommenders, a new class of recommender algorithms, are able to solve these problems.

III. EXISTING SYSTEM

Recommender systems make use of community opinions to help users identify useful items from a considerably large search space. The technique used by many of these systems is collaborative filtering (CF), which analyzes past community opinions to find correlations of similar users and items to suggest k personalized items (e.g., movies) to a querying user u. Community opinions are expressed through explicit ratings represented by the triple (user, rating, item) that represents a user providing a numeric rating for an item. Myriad applications can produce location-based ratings that embed user and/or item locations. Existing recommendation techniques assume ratings are represented by the (user, rating, item) triple.

Disadvantages of Existing System

- The existing systems are ill-equipped to produce location aware recommendations.
- The existing system provides more expensive operations to maintain the user partitioning structure.
- The existing system does not provide spatial ratings.

IV. PROPOSED SYSTEM

The proposed system address these issues of recommender systems by applying a different approach–item-based algorithms. The bottleneck in conventional collaborative filtering algorithms is the search for neighbors among a large user population of potential neighbors. Item-based algorithms avoid this bottleneck by exploring the relationships between items first, rather than the relationships between users. Recommendations for users are computed by finding items that are similar to other items the user has liked. Because the relationships between items are relatively static, item-based algorithms may be able to provide the same quality as the user-based algorithms with less online computation.

Advantages of Proposed System:

- This system supports a taxonomy of three novel classes of location-based ratings, namely, spatial ratings for non-spatial items, non-spatial ratings for spatial items, and spatial ratings for spatial items.
- System achieves higher locality gain using a better user partitioning data structure and algorithm.

- System exhibits a more flexible trade off between locality and scalability. System provides a more efficient way to maintain the user partitioning structure.

In this section we study a class of item-based recommendation algorithms for producing predictions to users. Unlike the user-based collaborative filtering algorithm discussed in Section 2 the item-based approach looks into the set of items the target user has rated and computes how similar they are to the target item i and then selects k most similar items $fi_1; i_2; : : : ; i_{kg}$. At the same time their corresponding similarities $fs_{i1}; s_{i2}; : : : ; s_{ik}$ g are also computed. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. We describe these two aspects namely, the similarity computation and the prediction generation in details here.

A. Item Similarity Computation

One critical step in the item-based collaborative filtering algorithm is to compute the similarity between items and then to select the most similar items. The basic idea in similarity computation between two items i and j is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity $s_{i;j}$. Fig1 illustrates this process, here the matrix rows represent users and the columns represent items. There are a number of different ways to compute the similarity between items. Here we present three such methods. These are cosine-based similarity, correlation-based similarity and adjusted-cosine similarity.

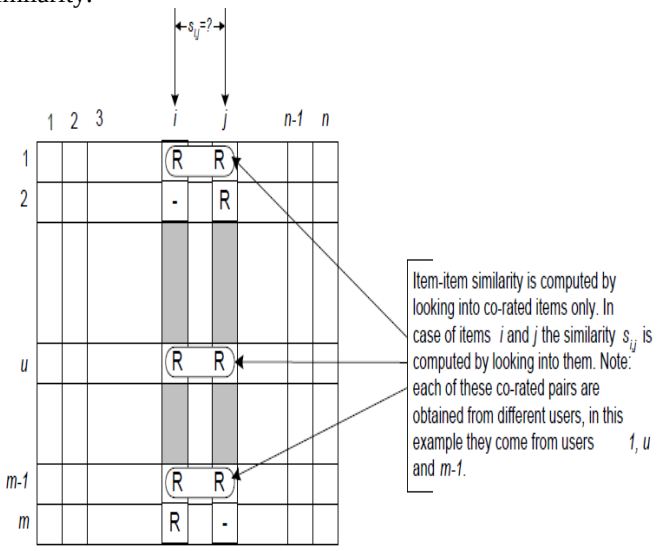

Fig1. Isolation of the co-rated items and similarity computation.

B. Adjusted Cosine Similarity

One fundamental difference between the similarity computation in user-based CF and item-based CF is that in case of user-based CF the similarity is computed along the rows of the matrix but in case of the item-based CF the similarity is computed along the columns i.e., each pair in the co-rated set corresponds to a different user (Figure 2). Computing similarity using basic cosine measure in item-based case has one important drawback–the difference in rating scale between different users are not taken into account.

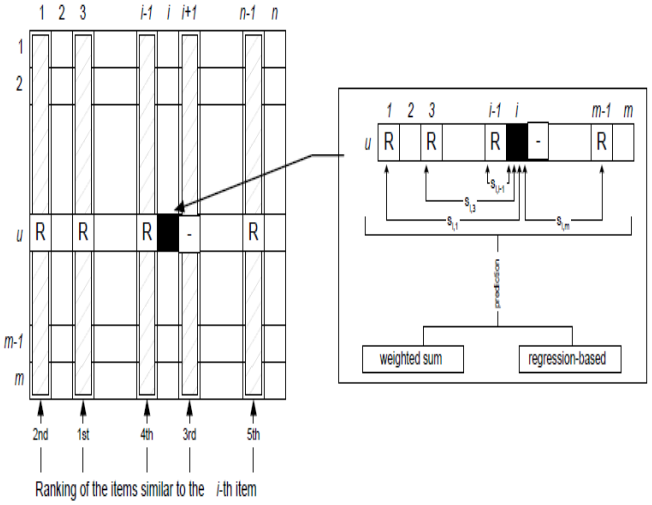

Fig2. Item-based collaborative filtering algorithm.

The adjusted cosine similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair.

V. EXPERIMENTAL PROCEDURE

A. Experimental steps

We started our experiments by first dividing the data set into a training and a test portion. Before starting full experimental evaluation of different algorithms we determined the sensitivity of different parameters to different algorithms and from the sensitivity plots we fixed the optimum values of these parameters and used them for the rest of the experiments. To determine the parameter sensitivity, we work only with the training data and further subdivide it into a training and test portion and carried on our experiments on them. For conducted a 10-fold cross validation of our experiments by randomly choosing different training and test sets each time and taking the average of the MAE values.

B. Benchmark user-based system

To compare the performance of item-based prediction we also entered the training ratings set into a collaborative filtering recommendation engine that employs the Pearson nearest neighbor algorithm (user-user). For this purpose we implemented a flexible prediction engine that implements user-based CF algorithms. We tuned the algorithm to use the best published Pearson nearest neighbor algorithm and configured it to deliver the highest quality prediction without concern for performance (i.e., it considered every possible neighbor to form optimal neighborhoods).

C. Experimental platform

All our experiments were implemented using C and compiled using optimization flag=06. We ran all our experiments on a linux based PC with Intel Pentium III processor having a speed of 600 MHz and 2GB of RAM.

VI. EXPERIMENTAL RESULTS

In this section we present our experimental results of applying item-based collaborative filtering techniques for generating predictions. Our results are mainly divided into two parts–quality results and performance results. In assessing the quality of recommendations, we first determined the sensitivity of some parameters before running the main experiment. These parameters include the neighborhood size, the value of the training/test ratio x, and effects of different similarity measures. For determining the sensitivity of various parameters, we focused only on the training data set and further divided it into a training and a test portion and used them to learn the parameters.

A. Effect of Similarity Algorithms

We implemented three different similarity algorithms basic cosine, adjusted cosine and correlation as described in Section and tested them on our data sets. For each similarity algorithms, we implemented the algorithm to compute the neighborhood and used weighted sum algorithm to generate the prediction. We ran these experiments on our training data and used test set to compute Mean Absolute Error (MAE). Figure 3 shows the experimental results. It can be observed from the results that offsetting the user-average for cosine similarity computation has clear advantage, as the MAE is significantly lower in this case. Hence, we select the adjusted cosine similarity for the rest of our experiments.

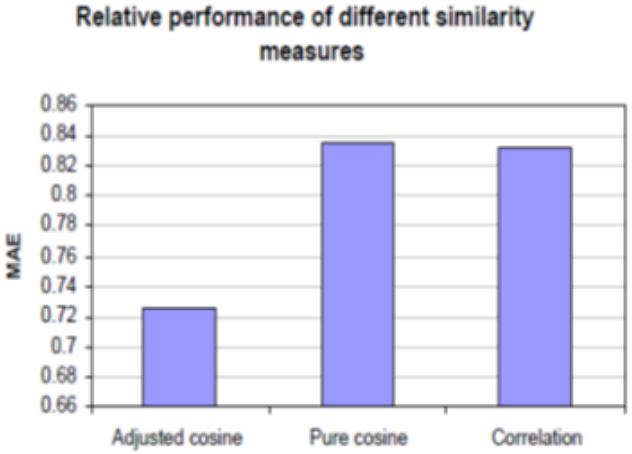

Fig3. Impact of the similarity computation measure on item-based collaborative filtering algorithm.

B. Sensitivity of Training/Test Ratio

To determine the sensitivity of density of the data set we carried out an experiment where we varied the value of x from 0.2 to 0.9 in an increment of 0.1. For each of these training/test ratio values we ran our experiments using the two prediction generation techniques–basic weighted sum and regression based approach. Our results are shown in Figure 4. We observe that the quality of prediction increase as we increase x. The regression-based approach shows better results than the basic scheme for low values of x but as we increase x the quality tends to fall below the basic scheme. From the curves, we select xD0.8 as an optimum value for our subsequent experiments.

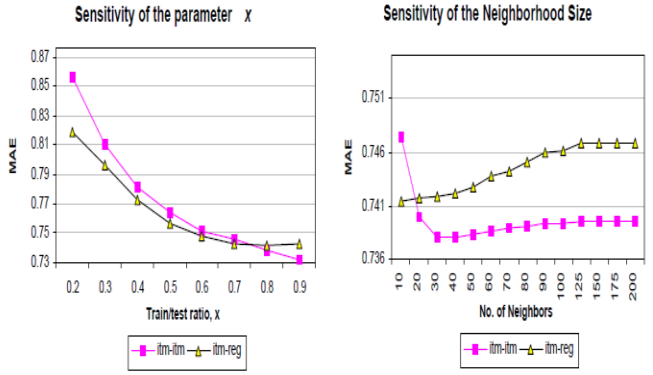

Fig4. Sensitivity of the parameter x on the neighborhood size.

C. Experiments with neighborhood size

The size of the neighborhood has significant impact on the prediction quality. To determine the sensitivity of this parameter, we performed an experiment where we varied the number of neighbors to be used and computed MAE. Our results are shown in Figure 4. We can observe that the size of neighborhood does affect the quality of prediction. But the two methods show different types of sensitivity. The basic item-item algorithm improves as we increase the neighborhood size from 10 to 30, after that the rate of increase diminishes and the curve tends to be flat. On the other hand, the regression-based algorithm shows decrease in prediction quality with increased number of neighbors. Considering both trends we select 30 as our optimal choice of neighborhood size.

D. Quality Experiments

Once we obtain the optimal values of the parameters, we compare both of our item-based approaches with the benchmark user-based algorithm. We present the results in Figure 5. It can be observed from the charts that the basic item-item algorithm out performs the user based algorithm at all values of x (neighborhood size = 30 and all values of neighborhood size (x = 0.8). For example, at x = 0.5 user-user scheme has an MAE of 0.755 and item-item scheme shows an MAE of 0.749. Similarly at a neighborhood size of 60 user-user and item-item schemes show MAE of 0.732 and 0.726 respectively. The regression-based algorithm, however, shows interesting behavior. At low values of x and at low neighborhood size it out performs the other two algorithms but as the density of the data set is increased or as we add more neighbors it performs worse, even compared to the user-based algorithm. We also compared our algorithms against the naive nonpersonalized algorithm described. We draw two conclusions from these results. First, item-based algorithms provide better quality than the user-based algorithms at all sparsity levels. Second, regression-based algorithms perform better with very sparse data set, but as we add more data the quality goes down. We believe this happens as the regression model suffers from data overfiting at high density levels.

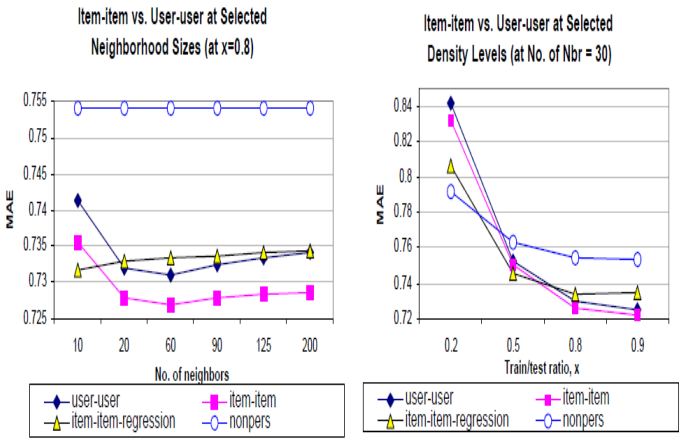

Fig5. Comparison of prediction quality of item-item and user-user collaborative filtering algorithms. We compare prediction qualities at x = 0.2; 0.5; 0.8 and 0.9.

E. Performance Results

Having clearly established the superior quality of item-based algorithms over the user-based ones, we focus on the scalability challenges. As we discussed earlier, item-based similarity is more static and allows us to precompute the item neighborhood. This precomputation of the model has certain performance benefits. To make the system even more scalable we looked into the sensitivity of the model size and then looked into the impact of model size on the response time and throughput.

F. Sensitivity of the Model size

To experimentally determine the impact of the model size on the quality of the prediction, we selectively varied the number of items to be used for similarity computation from 25 to 200 in an increment of 25. A model size of l means that we only considered l best similarity values for model building and later on used k of them for the prediction generation process, where k < l. Using the training data set we precomputed the item similarity using different model sizes and then used only the weighted sum prediction generation technique to provide the predictions.

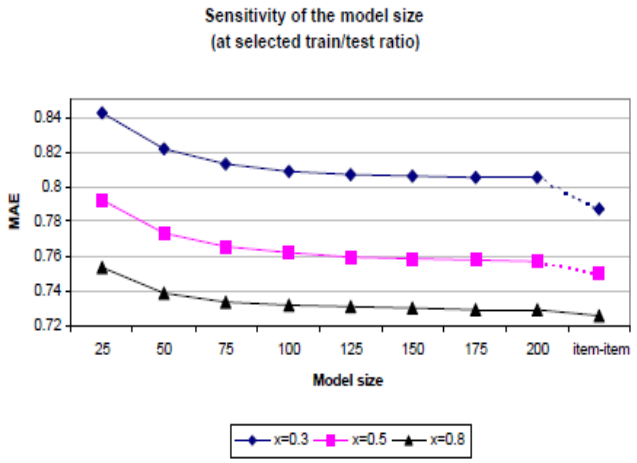

Fig6. Sensitivity of the model size on item-based collaborative filtering algorithm.

We then used the test data set to compute MAE and plotted the values. To compare with the full model size (i.e., model size = no. of items) we also ran the same test considering all similarity values and picked best k for prediction generation. We repeated the entire process for three different x values (training/test ratios). Figure 6 shows the plots at different x values. It can be observed from the plots that the MAE values get better as we increase the model size and the improvements are drastic at the beginning, but gradually slows down as we increase the model size. The most important observation from these plots is the high accuracy can be achieved using only a fraction of items. For example, at x = 0.3 the full item-item scheme provided an MAE of 0.7873, but using a model size of only 25, we were able to achieve an MAE value of 0.842. At x = 0.8 these numbers are even more appealing–for the full item-item we had an MAE of 0.726 but using a model size of only 25 we were able to obtain an MAE of 0.754, and using a model size of 50 the MAE was 0.738. In other words, at x = 0.8 we were within 96% and 98:3% of the full item-item scheme's accuracy using only 1.9% and 3% of the items respectively. This model size sensitivity has important performance implications. It appears from the plots that it is useful to precompute the item similarities using only a fraction of items and yet possible to obtain good prediction quality.

VII. CONCLUSION

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of user data in existing corporate databases, and will be stressed even more by the increasing volume of user data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems. In this paper we presented and experimentally evaluated a new algorithm for CF-based recommender systems. Our results show that item-based techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high-quality recommendations.

VIII. REFERENCES

[1] Aggarwal, C. C., Wolf, J. L., Wu K., and Yu, P. S. (1999). Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In Proceedings of the ACM KDD'99 Conference. San Diego, CA. pp. 201-212.

[2] Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as Classification: Using Social and Content-based Information in Recommendation. In Recommender System Workshop '98. pp. 11-15.

[3] Berry, M. W., Dumais, S. T., and O'Brian, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. SIAM Review, 37(4), pp. 573-595.

[4] Billsus, D., and Pazzani, M. J. (1998). Learning Collaborative Information Filters. In Proceedings of ICML '98. pp. 46-53.

[5] Brachman, R., J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., and Simoudis, E. 1996. Mining Business Databases. Communications of the ACM, 39(11), pp. 42-48, November.

[6] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43-52.

[7] Cureton, E. E., and D'Agostino, R. B. (1983). Factor Analysis: An Applied Approach. Lawrence Erlbaum associates pubs. Hillsdale, NJ.

[8] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41(6), pp. 391-407.

[9] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., Eds. (1996). Advances in Knowledge Discovery and Data Mining. AAAI press/MIT press.