A Collaborative Method to Reduce the Running Time and Accelerate the k-Nearest Neighbors Search

Alexandre Costa
Federal University of Campina Grande
Campina Grande, Brazil
antonioalexandre@copin.ufcg.edu.br

Reudismam Rolim
Federal University of Campina Grande
Campina Grande, Brazil
reudismam@copin.ufcg.edu.br

Felipe Barbosa
Federal University of Campina Grande
Campina Grande, Brazil
feliperamos@copin.ufcg.edu.br

Gustavo Soares
Federal University of Campina Grande
Campina Grande, Brazil
gsoares@computacao.ufcg.edu.br

Hyggo Almeida
Federal University of Campina Grande
Campina Grande, Brazil
hyggo@dsc.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande
Campina Grande, Brazil
perkusic@dee.ufcg.edu.br

***Abstract*—Recommendation systems are software tools and techniques that provide customized content to users. The collaborative filtering is one of the most prominent approaches in the recommendation area. Among the collaborative algorithms, one of the most popular is the k-Nearest Neighbors (kNN) which is an instance-based learning method. The kNN generates recommendations based on the ratings of the most similar users (nearest neighbors) to the target one. Despite being quite effective, the algorithm performance drops while running on large datasets. We propose a method, called Restricted Space kNN that is based on the restriction of the neighbors search space through a fast and efficient heuristic. The heuristic builds the new search space from the most active users. As a result, we found that using only 15% of the original search space the proposed method generated recommendations almost as accurate as the standard kNN, but with almost 58% less running time.**

***Keywords*—knn, recommendation systems, collaborative filtering, space restriction.**

I. Introduction

The emergence of Web 2.0 brought a significant increase in the volume of information available on the Internet, contributing to the information overload problem [10], that overwhelm the user with useless (in most cases) choices. Hence, recommendation systems (RS) [15], [1], [17] gained prominence and became very attractive for both the production sector and academic field. These systems bring together computational techniques in order to provide custom items (movies, music, books, etc.) to users, thus facilitating their choices. In the recommendation area, a prominent approaches is the collaborative filtering (CF) [22] that recommends items based on ratings of users with common interests to the target user. The state-of-the-art in recommendation field is formed by latent factor models [18], where some of the most successful implementations are based on Matrix Factorization (MF) [12]. In its basic form, the MF characterizes users and items with vectors of latent factors inferred from the pattern of the rated items. The latent factors represent aspects of physical reality, but we can not specify which aspects are these, therefore it is impossible to justify the provided recommendation.

Considered state-of-the-art, before the emergence of latent factor models, the k-Nearest-Neighbors (kNN) [11], [20], [21] is an instance-based learning algorithm. In the recommendation area this method is widely used as a collaborative technique for rating prediction. In contrast to latent factor techniques, the kNN recommendations can be justified, since they are generated from the nearest neighbors data. The main limitation of kNN is that its performance is inversely proportional to the size of the dataset. As the number of users and items grows, the computational cost to apply the method rises quickly, which decreases its time performance.

In this paper, we propose a collaborative method to reduce the running time and accelerate the nearest neighbor search, called Restricted Space kNN (RS kNN). This method restricts the search space to a percentage p of the original one, using a heuristic based on selecting the most active users. As a result, we found that with only 15% of the original space it is possible to generate high accuracy recommendations, but with almost 58% less running time.

This paper is organized as follows: In Section II, we present a background of the recommendation area. In Section III, we describe the related work. In Section IV, the proposed method is presented. Section V contains a description of the experiment conducted and its results. Section 6 refers to the conclusion of the work.

II. Background

In this Section, we present basic concepts of the recommendation area, that will allow to understand the proposed method.

A. Collaborative Filtering

Collaborative Filtering is one of the most recurring approaches in the recommendation area. Its techniques recommend items based on the ratings of other users. The ratings can be implicit, when they are measured based on user's behavior (e.g, viewing time, number of hits, etc.), or explicit, when users clearly express their interest on items through numerical grades, for example. The idea behind CF is that users with similar rating pattern tend to rate new items in a

DOI reference number: 10.18293/SEKE2015-153

similar manner. In CF, an instance is represented by a user feature vector that records which items were evaluated and which not. An advantage of collaborative techniques is the object representation independence, since CF techniques use only user ratings, which enables to work even with items in which the content extraction can be complex, such as audio and video. Another advantage refers to the recommendations diversity, since CF can suggest items different from those the user showed interest in the past.

Collaborative methods can be grouped as memory-based or model-based algorithms. In memory-based algorithms the dataset is loaded at the moment in which the recommendations are generated. They are easier to implement and can better adapt to changes of user interests. In contrast, model-based algorithms generate recommendations using a model previously constructed from the dataset. They can provide more accurate recommendations, but the model construction is an expensive step.

A common scenario in collaborative filtering is the rating prediction, where a user rating to a given item is inferred. In this scenario, it is assumed that items with higher values are more interesting. Ratings can be represented in different scales, usually in 1-5 stars. The quality of the prediction is normally measured through error-based metrics, calculated from the difference between the predicted value and the real user rating. A common metric for this purpose is the Mean Absolute Error (MAE) represented by Equation 1, where $r(u,i)$ is the rating of a user u to an item i, $r'(u,i)$ corresponds to the prediction generated for u about i and $|I_u|$ is the size of the item set evaluated by u.

$$MAE = \frac{1}{|I_u|} \cdot \sum_{i \in I_u} |r'(u,i) - r(u,i)| \tag{1}$$

B. k-Nearest Neighbors

The kNN is a memory-based method, thus it is necessary to have all the training set stored to do the recommendation process. In larger datasets the kNN computational cost can grow quickly. This occurs because as the number of users and items grows the kNN demands more memory to store the data, more similarity calculations and more time to perform the neighbors selection (because the search space becomes larger).

Recommendations are based on the k nearest neighbors ratings, where a similarity measure to select the nearest neighbors must be defined. This measure has a direct impact on the kNN results, because it is used to determine how close two users are. The similarity between two users is calculated from the items they have rated simultaneously. A popular similarity measure in the recommendation field is the Pearson correlation, represented by the Equation 2, where $|I_{au}|$ is the size of the item set simultaneously evaluated by users u and a, $x = r(u,i)$ and $y = r(a,i)$. The Equation 2 differs from the traditional form, because it is adapted to perform faster calculations.

$$sim(a,u) = \frac{|I_{au}| \sum_{i \in I_{au}} xy - \sum_{i \in I_{au}} x \cdot \sum_{i \in I_{au}} y}{\sqrt{[|I_{au}| \sum_{i \in I_{au}} x^2 - (\sum_{i \in I_{au}} x)^2] \cdot [|I_{au}| \sum_{i \in I_{au}} y^2 - (\sum_{i \in I_{au}} y)^2]}} \tag{2}$$

The recommendation process consists in making a prediction for the set of items not evaluated by the user. One of the most common ways to accomplish this goal is through the Equation 3, where U_a is the set of nearest neighbors of the target user a and $\overline{r(a)}$ corresponds to the average ratings of the user a.

$$r'(a,i) = \overline{r(a)} + \frac{\sum_{u \in U_a} sim(a,u) \cdot (r(u,i) - \overline{r(u)})}{\sum_{u \in U_a} sim(a,u)} \tag{3}$$

III. Related Work

Boumaza and Brun [3] proposed a method based on the restriction of the nearest neighbors search space. In traditional search, it is necessary to check the similarity among the target user with each other user in the dataset, and then select the k nearest neighbors. On large datasets, this task becomes expensive. In their work, Boumaza and Brun used a stochastic search algorithm, called Iterated Local Search (ILS) [13]. The ILS returns a subset of users, Global Neighbors (GN), in which the neighbors are chosen. The idea is to accelerate the neighborhood formation by seeking the neighbors in a smaller subset, rather than search in the entire dataset. As a result, the proposed method can reduce the search space to 16% of the original while maintaining the accuracy of recommendations near to those achieved by traditional search. We adapt their work by introducing a new heuristic to perform a faster and less expensive GN selection, instead of using the ILS algorithm.

Friedman et al. [5] proposed an optimization to k-Dimensional Tree (kd tree). Originally proposed by Bentley [2], the kd tree is an algorithm for storing data to be retrieved by associative searches. The kd tree structure provides an efficient search mechanism to examine only those points closer to the point of interest, which can reduce the nearest neighbors search time from $O(N)$ to $O(logN)$. In Friedman's work, the goal was to minimize the number of points examined during the search, that resulted in a faster search. Gother et al. [8] developed a method which represents a slight variation of the work done by Friedman et al. [5]. In the kd tree construction each node is divided into two, using the median of the dimensions with greater variance between the points in the subtree, and so on. As a result, the method got 25% faster in the classification step. The kd tree based methods differ from ours, because they accelerate the neighbor selection using a non-linear search by partitioning the whole space into non-overlapping regions.

Other works use the approximate nearest neighbors (ANN) concept to deal with the kNN search problem. The ANN methods do not guarantee to return the exact nearest neighbors in every case, but in the other hand, it improves speed or memory savings. An algorithm that supports the ANN search is locality-sensitive hashing (LSH). According to Haghani et al. [9], the main idea behind the LSH is to map, with high probability, similar objects in the same hash bucket. Nearby objects are more likely to have the same hash value than those further away. Indexing is performed from hash functions and from the construction of several hash tables to increase the probability of collision between the nearest points. Gionis et al. [7] develop a LSH method to improve the neighbors search for objects represented by the points of dimension d in a Hamming space $\{0,1\}^d$. Their method was able to overcome in terms of speed the space partitioning tree methods, when data are stored on disk. Datar et al. [4] proposed

a new version of the LSH algorithm that deals directly with points in a Euclidean space. To evaluate the proposed solution experiments were carried out with synthetic datasets, sparse vectors with high dimension ($20 \leq d \leq 500$). As a result, they obtained performance of up to 40 times faster than kd tree. The ANN methods are related to ours, because they aim to accelerate the neighbor search by giving up accuracy in exchange for time reduction.

IV. Restricted Space kNN

In standard kNN, the nearest neighbors are selected by checking the similarity of the target user with each other user in the dataset. For every user, a distinct neighborhood has to be formed. When the number of users and items in the training set grows, the kNN running time decreases, because its computational cost rises quickly. To minimize this problem we proposed a collaborative method derived from the k-Nearest Neighbors for rating prediction. Our method is based on the restriction of the neighbor search space. In Figure 1, we can see the main idea of the proposed method in contrast to the standard approach. The search space is reduced to a percentage p of the original one, which is accomplished by selecting a subset of users capable to offer ratings to generate accurate recommendations. Then, the neighbor search is performed in the new space, which allows it to be faster, since the space becomes smaller. Finally, the most similar users to the target one are chosen to form his neighborhood.

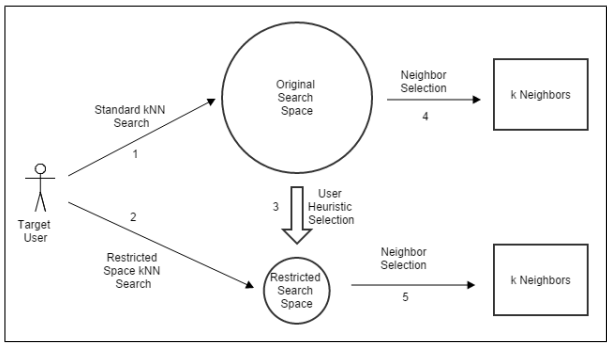

Fig. 1: Standard search and restricted Search

The RS kNN was inspired by the work presented by Boumaza and Brun [3]. Their method was able to achieve accurate recommendations with a reduced search space that corresponds to 16% of the original one. A stochastic search algorithm called Iterated Local Search does the user selection. The ILS is an efficient algorithm for optimization problems, but requires a considerable running time and expensive operations to achieve its results. Given this limitation, we saw an opportunity to improve Boumaza and Brun's work [3]. Therefore, instead of using the ILS algorithm, we propose a faster and accurate heuristic to select the percentage p of users who will compose the new search space.

Our approach aims to reduce computational cost and improve running time, but it is susceptible to lose recommendation accuracy, since it works with a considerably smaller amount of data. In order to minimize such loss, it is necessary to define an efficient heuristic for the user selection. In our work, we investigated the following heuristics[1]:

- **Similarity Average:** users are selected according to their similarity average. For each user, we calculate his similarity with each other that remains in the dataset and then get the average. Those with the highest averages are chosen;
- **Number of Connections:** select users according to their ability to be neighbors. Each time a user shows a positive similarity with another he receives one point. In the end, those with the highest scores are selected;
- **Neighbors Recurring:** users are scored according to the number of times that arise between the k nearest neighbors of a target user. For example, we check the k nearest neighbors of a target user and then assign one point for each neighbor. This process is repeated with all users in the dataset and at the end, we have the list of the most common neighbors;
- **Most Active Users:** it selects users according to the number of items rated. Those with the largest quantities are chosen;
- **Distinct Items:** it corresponds to a variation of the previous heuristic, with the aim of bringing together users with the greatest possible number of distinct items. It selects users that, together, offers the most distinct set of items.

[1]The Similarity Average and Number of Connections were already introduced in [3]. The others were proposed in our work.

V. Experiment

We conducted an experiment to evaluate the proposed method. The experiment was divided into two stages. The first aims to evaluate the heuristics described in Section IV, in order to choose the most suitable to compose our method. The second corresponds to the evaluation of the proposed method by comparing it with implementations developed to accelerate the kNN search. In the experiment, we focused on measuring time performance (in seconds) and accuracy (error rate of the predictions, measured by the Mean Absolute Error metric).

The experiment was executed on a machine with Core i7 2300K (3.4 GHz) processor, 8GB of DDR3 RAM and Windows 7 64-bit. We used the Java language and Eclipse[2] (Kepler) in its 64-bit version for developers. We also used two external libraries, the MyMediaLite [6], which is specialized in recommendation systems methods, and the Java Statistical Analysis Tool (JSAT) [16], which offers dozens of machine learning algorithms.

A. Dataset

We choose two popular datasets that contain real data from movie domain. The first one is the MovieLens 100K, which has 943 users, 1,682 movies and 100,000 ratings. The second has 6,040 users, 3,952 movies and 1,000,209 ratings. Both have ratings on a scale of 1 to 5 stars that represent the level of the user interest to an item.

The data were segmented following the 10-fold cross-validation pattern, which consists in the separation of 90% of the data for training and 10% for testing. This process was repeated five times to provide a final amount of 50 samples for execution.

B. Setting Parameters

Before going on with the experiment we had to set some parameters. Thus, we used the standard kNN algorithm for rating prediction, whose implementation was based on the source-code available in MyMediaLite library. We performed 10 execution on the MovieLens 100K dataset, focusing in accuracy. The parameters and their values are listed below:

- **Similarity measure:** we compared the Pearson correlation with another popular measure in the recommendation area, the Cosine similarity [19]. As we can see in Table I, the Pearson correlation provides a lower MAE, which means more accurate recommendations. Therefore, we chose the Pearson correlation as similarity measure for our approach.

[2]www.eclipse.org

- **Number of nearest Neighbors (k):** choosing the optimal value for k is not a trivial task, because it can vary according to the data. In Figure 2, we can see that for the MovieLens 100K the best k is 30, because it provided the lowest error rate. We used the same k for the larger dataset to maintain the speed gains. In addition, a greater k would increase the running time.
- **Percentage of the search space (p):** we used the Most Active Users heuristic to find the optimal value for p. A greater p would give better MAE, but at the expense of the running time. Thus, our choice was given by a trade-off between the MAE and the running time. According to the Figure 3 when p reaches 15%, it seems to be the best trade-off. Besides it, we thought that should be important to choose a p closer to the one in Boumaza and Brun [3], since our work is inspired by theirs.

TABLE I: Comparison of Pearson Correlation and Cosine Similarity

Similarity measure	MAE
Pearson	0.6813
Cosine	0.7100

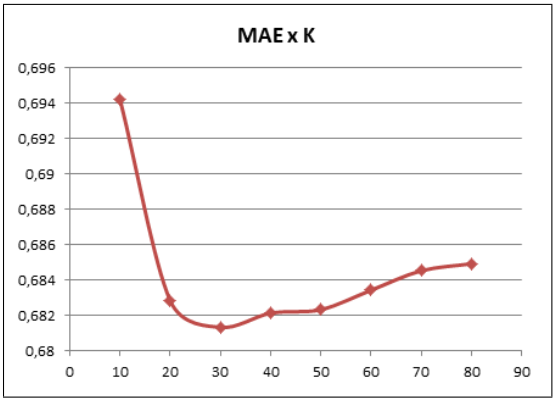

Fig. 2: Variation of k and its respective error rates

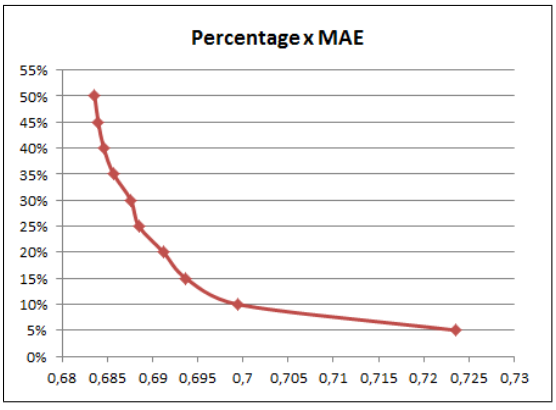

Fig. 3: Variation of p and its respective error rates

C. Heuristic Evaluation

The heuristic evaluation process was based on the time needed to select the users to compose the new search space and the accuracy that they can provide. Tables II and III contain the results of each heuristic. Similarity calculations are expensive, thus the heuristics *Similarity Average*, *Number of Connections* and *Neighbors Recurring* demanded the largest time to select their users. The others showed significantly shorter times, because they use less expensive operations, since there is no similarity computing. Regarding the accuracy, the results were more homogeneous, there was even a decrease in error rate in the larger dataset.

The results showed the heuristic *Most Active Users* as the best option to compose the proposed method. It provided the smallest error rate, resulting in more accurate recommendations. In addition, the selection time was the best among the heuristics and remained almost constant in the tests with the larger dataset.

TABLE II: Results of the heuristic evaluation on MovieLens 100K

Heuristic	Time (s)	Error (MAE)
Most Active Users	**0.001**	**0.6937**
Distinct Items	0.072	0.6954
Similarity Average	1.604	0.7053
Number of Connections	1.591	0.7046
Neighbors Recurring	1.631	0.6994

TABLE III: Results of the heuristic evaluation on MovieLens 1M

Heuristic	Time (s)	Error (MAE)
Most Active Users	**0.002**	**0.6546**
Distinct Items	1.097	0.6549
Similarity Average	92.713	0.6588
Number of Connections	92.652	0.6579
Neighbors Recurring	100.390	0.6553

D. Method Evaluation

To evaluate our method, we chose four approaches that correspond to implementations derived from the kNN method. They were developed to accelerate the nearest neighbor selection. Their performance were measured under accuracy and running time and the results are presented in Tables IV and V. The compared approaches are:

- **k-dimensional tree (kd tree):** implemented in JSAT tool. Corresponds to the traditional form [2];
- **Locality Sensitive Hash (LSH):** implemented in JSAT tool. It was based on the algorithm presented by Dating et al. [4];
- **Standard k-Nearest Neighbors (kNN):** corresponds to the kNN for rating prediction. We implemented it based on the source code of the MyMediaLite library;
- **Iterated Local Search (ILS) kNN:** we implemented this method based on the paper presented by Boumaza and Brun [3].

Three of them (standard kNN, ILS and the RS kNN) are collaborative techniques from the recommendation field that were developed focusing on the rating prediction task. They usually deal with sparse vectors and try to "fill" each missing value of the vectors. Regarding the running time, our method achieved the best values, being almost 58% faster (in MovieLens 1M) than the standard kNN, which took second place. The running time of ILS was much greater than the others, because its fitness function needs to check the errors provided by the subsets of users build at each iteration of the algorithm. Furthermore, as a non-deterministic method, it is impossible to predict the number of iterations needed to find the final subset. As expected, the accuracy of the recommendations generated by our method was a little lower than standard kNN, but considering the running time gain, the results were very promising.

The kd tree and LSH methods were originally implemented for the classification task. They generally work with dense vectors and

aim to label an unknown instance. In the proper domain, they are capable to reduce the search time from linear to logarithmic level, although in this experiment this behavior was not evidenced. The classification methods presented poor performances in running time and accuracy. The kd tree tends to lose great performance with high dimension vectors ($d \geq 10$) [7], [5], a problem known as the curse of dimensionality [14], which contributes to the low performance, since the datasets are composed of high dimension sparse vectors. The LSH prioritizes time instead of accuracy, since it returns the approximated nearest neighbors. This reason justifies the high error rate of the LSH. The running time performance of the LSH was unexpected, because this algorithm is designed to be fast even with high dimension data. We believe the data sparsity was responsible, because it reduces the probability of collisions, making difficult to group users in the same hash bucket and consequently it increases the search time.

TABLE IV: Results on MovieLens 100K

Method	Running Time (s)	MAE
kd tree	22.24	0.8333
LSH	11.13	0.7528
ILS	1233.15	0.7083
kNN	4.35	**0.6826**
RS kNN	**2.45**	**0.6937**

TABLE V: Results on MovieLens 1M

Method	Running Time (s)	MAE
kd tree	1576.01	0,7875
LSH	454.42	0,7014
ILS	7892.7	0,6591
kNN	242.13	**0,6500**
RS kNN	**100.87**	**0,6546**

VI. Conclusion

In this paper, we presented the Restricted Space kNN, a collaborative method to reduce the running time and accelerate the k-Nearest Neighbors search. The RS kNN focuses on the idea of restricting the nearest neighbors search space using a fast and efficient heuristic for user selection. The method was capable to perform up to 58% faster than standard kNN. The Most Active Users heuristic was quick in reducing the search space, getting together a set of users able to provide accurate recommendations. Using only 15% of the original search space we have achieved an error rate just 0.7% higher than the standard method.

The main limitation of our method refers to the validation process. We evaluated it in only one domain, which makes difficult to generalize the results achieved. Our method showed great performance gains in exchange for a small reduction in accuracy, however, we cannot guarantee that the error rate will remain constant in other domains.

As future work, we intend to investigate the effects of the proposed method in a larger dataset, because we noticed that the accuracy gap between our method and the standard kNN became smaller when the data increased. In the MovieLens 100K, we obtained an absolute difference of 0.0111, whereas with MovieLens 1M, the difference was reduced to 0.0046. In addition, we also intend to investigate the proposed method in the item predicting scenario with implicit feedback.

References

[1] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 121–128, New York, USA, 2013. ACM.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[3] A. Boumaza and A. Brun. Stochastic search for global neighbors selection in collaborative filtering. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 232–237, New York, NY, USA, 2012. ACM.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, New York, USA, 2004. ACM.

[5] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, Sept. 1977.

[6] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 305–308, New York, USA, 2011. ACM.

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] P. Grother, G. T. Candela, and J. L. Blue. Fast implementations of nearest neighbor classifiers. *Pattern Recognition*, 30(3):459–465, 1997.

[9] P. C.-M. K. A. P. Haghani. Lsh at large – distributed knn search in high dimensions. 2008.

[10] R. Janssen and H. de Poot. Information overload: Why some people seem to suffer more than others. In *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles*, NordiCHI '06, pages 397–400, New York, USA, 2006. ACM.

[11] L. Jiang, Z. Cai, D. Wang, and S. Jiang. Survey of improving k-nearest-neighbor for classification. In *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 01*, FSKD '07, pages 679–683, Washington, USA, 2007. IEEE Computer Society.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[13] H. R. Lourenco, O. C. Martin, and T. Stutzle. Iterated local search. In *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, 2002.

[14] F. Murtagh, J.-L. Starck, and M. W. Berry. Overcoming the curse of dimensionality in clustering by means of the wavelet transform. *The Computer Journal*, 43(2):107–120, 2000.

[15] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *Expert Syst. Appl.*, 39(11):10059–10072, Sept. 2012.

[16] E. Raff. Java Statistical Analysis Tool. https://code.google.com/p/java-statistical-analysis-tool/, Sept. 2013.

[17] R. Rolim, F. Barbosa, A. Costa, G. Calheiros, H. Almeida, A. Perkusich, and A. Martins. A recommendation approach for digital tv systems based on multimodal features. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 289–291, New York, NY, USA, 2014. ACM.

[18] M. Rossetti, F. Stella, and M. Zanker. Towards explaining latent factors with topic models in collaborative recommender systems. In *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*, pages 162–167, Aug 2013.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, USA, 2001. ACM.

[20] B. Wang, Q. Liao, and C. Zhang. Weight based knn recommender system. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on*, volume 2, pages 449–452, Aug 2013.

[21] L. Xiong, Y. Xiang, Q. Zhang, and L. Lin. A novel nearest neighborhood algorithm for recommender systems. In *Intelligent Systems (GCIS), 2012 Third Global Congress on*, pages 156–159, Nov 2012.

[22] J. Zhou and T. Luo. Towards an introduction to collaborative filtering. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 4, pages 576–581, Aug 2009.