Concept-Based Document Recommendations for CiteSeer Authors[*]

Kannan Chandrasekaran[1], Susan Gauch[2], Praveen Lakkaraju[1], Hiep Phuc Luong[2]

[1]University of Kansas, EECS Department
[2]University of Arkansas, CSCE Department
kannanc@ku.edu, sgauch@uark.edu, lakkaraju.praveen@gmail.com, hluong@uark.edu

Abstract. The information explosion in today's electronic world has created the need for information filtering techniques that help users filter out extraneous content to identify the right information they need to make important decisions. Recommender systems are one approach to this problem, based on presenting potential items of interest to a user rather than requiring the user to go looking for them. In this paper, we propose a recommender system that recommends research papers of potential interest to authors known to the CiteSeer database. For each author participating in the study, we create a user profile based on their previously published papers. Based on similarities between the user profile and profiles for documents in the collection, additional papers are recommended to the author. We introduce a novel way of representing the user profiles as trees of concepts and an algorithm for computing the similarity between the user profiles and document profiles using a tree-edit distance measure. Experiments with a group of volunteers show that our concept-based algorithm provides better recommendations than a traditional vector-space model based technique.

Keywords: Recommender System, CiteSeer, Digital Library, Conceptual Recommender

1 Introduction

The web has grown tremendously since its inception. Traditional search engines gave the same results to all the users without considering their specific user needs. However, the nature of information available on the web, its applications, and its user base has diversified significantly. In addition, a user's ability to locate relevant content would be based on their ability to construct good queries. This has lead to the development of systems that identify the needs of individual users and provide them with very specific information to satisfy their requirements. "Recommender systems" which recommend items to the users by capturing their interests and needs, are one approach to implementing personalized information filtering systems [19].

[*] This research was supported in part by the National Science Foundation grant number 0454121: CRI: Collaborative: Next Generation CiteSeer.

Recommender systems have been used to recommend many different types of items. For example, websites such as Amazon.com use recommendation engines to make personalized recommendations of the products to its users, and digital libraries like CiteSeer [22] make recommendations of technical papers to its users. Most existing recommender systems use a form of recommendation called collaborative filtering [21]. In this approach, every user in the system has a neighborhood of similar users who share many of the current user's interests. The recommendations provided for the current user are provided as a function of ratings provided by the users in their neighborhood. However, even when there are a large numbers of users to provide recommendations and large numbers of items to be recommended; only a small portion of items receive a sufficient number of ratings to form the neighborhood. Consequently, the recommendations are isolated to only a subset of the available items. Also, when a new item is introduced, there are no ratings available for its recommendation. These problems can be avoided if the recommendation is based on the content of the item.

Previous research has shown that recommendation is a very valuable service to the users of digital libraries [20]. The large amount of textual information in the library collections, such as CiteSeer, can be exploited to provide content-based recommendations. Traditional content-based recommender systems [23] have used the tf*idf [3] similarity measure to compute the similarity between documents. In this model, the documents are modeled as keyword vector and similarity is computed using a distance metric such as cosine similarity measure. However, this model relies heavily on the exact keyword match and does not consider ambiguities present in natural language such as synonymy and polysemy. In this work, we propose a content-based recommender system that represents documents and the user profiles as trees of concepts and computes the similarity between the documents and user profile using a version of the tree-edit distance algorithm. We demonstrate that this approach outperforms a traditional keyword vector-based recommender system.

2 Related Work

In the section, we present sample recommender systems with more emphasis given to recommendations of textual data such as book recommendations and digital library recommendations since these are directly related to our work. [8], [10] and [17] use content-based recommendations whereas [11] and [15] use collaborative recommendations to recommend different items. [2] and [5] take a hybrid approach by combining both content and collaborative recommendations.

Basu et al [8] model the task of assigning technical papers to conference reviewers as a problem of recommending technical papers to the authors based on their interests and background. Using WHIRL [9], they analyze the effect of combining different information sources about papers and reviewers in providing recommendations. Pazzani et al [10], describe a content-based recommender system for recommending news items for users of handheld devices such as PDA's and cell phones. Implicit information about the user is collected based on the activities such as selecting or skipping a news item and is modeled as a profile describing the user's interest. The content-based machine learning algorithm then learns the user's short-term and long-term interests and provides recommendations for future news items. Zhang et al [17], developed a content-based recommender system that extends information filtering systems by recommending papers that are not just relevant, but also novel. They use set theory, cosine similarity measure and Kullback-Leibler measure to propose different models that assign a redundancy score to the new items based on previously seen items.

Si and Jin [11], propose a probabilistic model for collaborative filtering in which they extend the existing partitioning algorithms used for collaborative filtering by clustering both the users and the items simultaneously. They use a modified version of the EM algorithm to predict the ratings for the unseen items for individual users based on their past ratings. Rather than using clustering to address the problem of limited data, Sarwar et al [15] apply dimensionality reduction techniques to the sparsely-populated user-product matrix. Then they perform latent semantic indexing [16] before using the cosine similarity measure to select items to recommend.

In work related specifically to digital libraries, Torres et al [2] use a combination of content-based and collaborative algorithms to build a recommender system for digital libraries. The content-based algorithms find similar papers based on the text of the current paper using cosine similarity measure while collaborative algorithms use the standard K-nearest neighbor algorithm on the input list of citations to output an ordered list of citations as recommendations. These content and collaborative algorithms are then combined together to generate the hybrid recommendation algorithms. They find that the hybrid algorithms perform better than the individual algorithms. Huang et al [5] also take a hybrid approach to recommendations from a digital library. They employ a two stage approach to build a graph based recommender system for a Chinese book store. Books, customers and the purchase information are modeled as a two layered graph. Once the model is set up the task of recommendation reduces to a graph search problem.

3 Approach

The system consists of three main modules, i.e., the classifier module, the profiler module, and the recommender module. The architectural diagram for our system is shown in Fig. 1.

3.1 Classifier Module

This module classifies the documents in the CiteSeer database into a predefined set of concepts, in particular, ACM's Computing Classification System (or CCS). A vector-space classifier was trained the categories of this taxonomy using published papers labeled with the CCS category codes and evaluated as part of prior work [26]. The output from the module is a concept vector for each document in the CiteSeer collection. Since these vectors are sparse, they are stored in the database as lists of (*conceptId*, *wt*) pairs in which the *wt* represents the degree of association between the document and the concept.

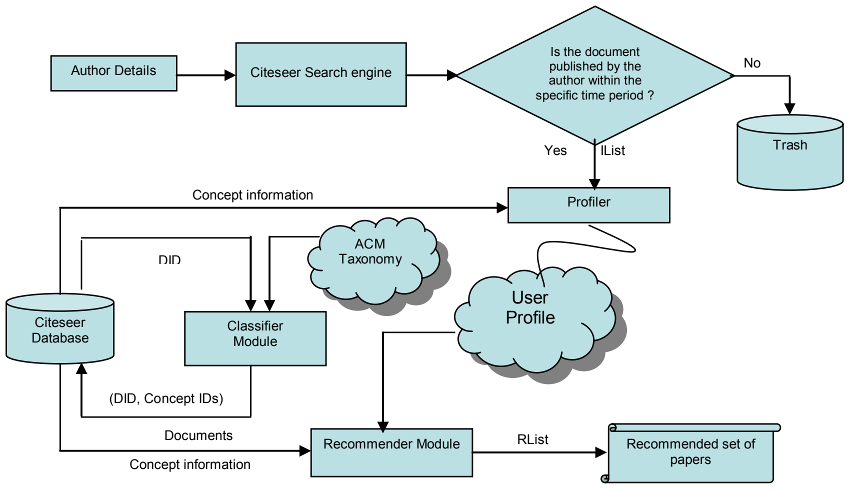

Fig. 1. Author Recommender System for CiteSeer

3.2 Profiler Module

The main objective of the profiler module is to create a conceptual user profile for the author for whom we are trying to recommend papers. The input to the profiler module is a list of documents from the CiteSeer database that were published by the author, called the *IList*. For authors in our study, we create this list by manually querying the CiteSeer search engine with the author's first, last or other common names provided by as part of the registration process. For each document in the *IList*, we retrieve the set of associated concepts and sort them in decreasing order by their weights. They are then added together to create a weighted concept vector representing user’s interests.

3.3 Recommender Module

The recommender module uses the user profile to create the list of recommended papers, or *RList*. For each non-zero concept *x* in the user profile, the recommender module searches the CiteSeer database for documents which have non-zero values for *x* in their concept vectors. These documents are then added to the *RList*. The number of concepts () in the user profile used by the recommender module is provided as a parameter and only the highest weighted concepts are used.

After processing the concepts in the user profile, the *RList* holds the list of documents that are associated with concepts in the author’s profile. The final step is to rank order these documents in decreasing order of their likely interest to the author. To begin this ranking, the recommender module retrieves the concept vectors for each document in the *RList*. Traditionally, recommender systems would calculate the cosine similarity measure between the document and user keyword vectors in order to rank them by similarity. We extend this approach by calculating the similarity between the user and the documents using concept, rather than keyword vectors.

The cosine similarity measure assumes that the elements of the vectors being compared are independent, which is not true. In order to exploit the relationships between concepts in a hierarchical concept space such as the CCS, we next convert the concept vectors for the user profile and the documents in the *RList* into weighted concept trees. This conversion is performed by creating a full taxonomy of all CCS concepts with zero weights for each concept. Then, for each non-zero concept in the original vector, we add the concept's weight to the tree. Finally, we recursively propagate weights up the tree until the root is reached. A tuning parameter, α, is introduced to control the proportion of weight that is propagated to a parent by its child.

Once the user profile and the documents are represented as trees, the problem of computing the distance between them is reduced to finding the distance between the two trees. Based on previous research [26], we use the tree-edit distance measure to calculate the cost of transforming one tree into another with the minimum number of operations. By matching each pair of nodes in two trees, we have three kinds of operations as following:

1. insertion: The cost of inserting a new node into the tree
2. deletion : The cost of deleting a existing node from the tree
3. substitution: The cost of transforming the one node into another

The cost of deletion or insertion of a node is equal to the weight associated with the node and the cost of substitution is equal to the difference between weights of the substituted nodes.

This algorithm calculates the cost of modifying the document profile to match the user profile. The closer the two profiles, the lower the cost of the required modifications. It then sorts the documents in the *RList* in increasing order so that the closest documents appear first and the most distant documents appear last. The closest 10 documents are then displayed to the author as the set of recommended papers.

4 Evaluation and Results

We used the ACM's Computing Classification taxonomy to classify the documents in the CiteSeer database into a predefined set of concepts. This taxonomy consists of 368 categories and is three levels deep. After the classification, each document had three concepts associated with it. We used the documents in CiteSeer database published between 1994-2005 as dataset for building the profile and generating the recommendations. In order to establish truth for the recommendations, we conducted a user study involving 8 published professors from the computer science and computer engineering departments from various universities.

During registration, the professors entered basic information such as their First Name, Last Name, Email Address and any common names that they used in their published papers. This information is used to manually create the queries provided to the CiteSeer search engine to generate the *IList* for each subject.

4.1 Baseline Method

CiteSeer has a built-in recommender system that can compute the similarity between documents using different semantic features [1]. In order to compare our conceptual approach to a keyword approach, we used the tf*idf scheme implemented by CiteSeer as our baseline algorithm. In order to identify the most similar documents for a registered author, for each document in the author's *IList*, we used CiteSeer to retrieve a rank-ordered list of the most similar documents based on tf*idf similarity. These lists, one list per document in the *IList*, are then combined to create the final list for the author. If a document occurred in more than one list, then the weights were accumulated to produce the total weight for that document. The top 10 documents from the final list are then treated as the final set of recommendations produced by the baseline method and is then presented to the author for evaluation.

4.2 Conceptual Recommendation Method

As discussed in the previous section, there are 2 input parameters to our algorithm, i.e., α (the propagation factor) and ▯ (the number of concepts in the user's concept vector to be used to find documents for the *RList*). We tested with four different values for α: 0, 0.33, 0.67 and 1.0. When $\alpha = 0$, no weight is propagated from the child concept to the parent concept during the concept vector to concept tree conversion. Since we are not exploiting the hierarchical relationships of the concepts, this is essentially a concept vector approach. When $\alpha > 0$, weight is propagated from child concepts to their parents and we consider these variations of a concept tree approach. In order to evaluate whether authors are more interested in receiving recommendations for papers on all their interests, or just their major ones, we also evaluated the three values of ▯: 5, 10 and 15.

We compared the concept tree, concept vector and the baseline algorithm with each other to test the hypothesis that the algorithm computing the similarity using the concept tree ($\alpha > 0$) is better than the algorithm computing the similarity using concept vector ($\alpha = 0$) which is in turn better than the algorithm computing the similarity using keyword vector.

For each value of α, we varied ▯ to obtained three different outputs of the concept-based algorithm. Thus, for each author we obtained 12 different lists, each containing 10 recommended papers. To reduce the work of authors we merged these lists and removed duplicates. To receive unbiased judgments, this list was randomized before being presented to the subjects for evaluation. Once the recommended papers were identified, the author was emailed to notify them that they have papers to review. For easier and more efficient interactions, a web interface was provided for rating the documents. For each recommended document, the title, abstract and the link to the original document were provided and the author was asked to rate the documents using a scale of 1-4 with 1 representing the most relevant and 4 representing the least relevant documents.

4.3 Results

Our hypothesis is that a recommender system based on the concept tree algorithm would be more accurate than the one using concept vectors which in turn should be more accurate than a recommender system based on keyword vectors.

We compared the approaches using the number of correct recommendations within the top 10. Unlike simple "relevant-not relevant" judgments, we had judgments on a scale of 1-4. We report the results when only documents with a user judgment of 1 are considered good recommendations, a very strict definition of correct. The middle ground occurs when we consider documents with ratings of 1 or 2 as correct. Finally, we report results we considering documents with any rating except 4 as being correct.

The first experiment was designed to identify the best performing concept tree algorithm, i.e., algorithms that propagate of α to create a connected tree. Thus, we compared results for the three non-zero values of the propagation factor α, i.e., 0.33, 0.67, and 1.0. For each value of α, we evaluated three values of β, the number of concepts in the user profile used to create the *RList*. Thus, for each value of α we identified the value of β that gave the highest accuracy overall. Fig. 2 displays the results with the best-performing β for each non-zero value of α.

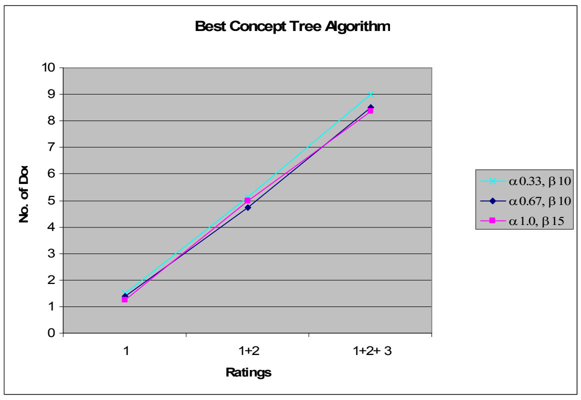

Fig. 2. Best Concept Tree Algorithm for each Value of α

With a high propagation factor, the best results were obtained when 15 concepts from the user profile were used. However, for the other two propagation factors, slightly better results were obtained when only 10 concepts were used. The best overall results occurred with α = 0.33 and β = 10. From this figure, we observe that this version provided the best performance for all definitions of "correct." However, we also observe that the algorithms provide very similar performance, so that the results do not seem to be particularly sensitive to the choice of α or β. Only one of the recommended documents, on average, would satisfy the strictest definition of correct, i.e., be judged 1. However, when we consider a reasonable definition of "correct", i.e., documents judged 1 or 2, roughly half of the 10 documents presented to the authors were of interest to them. With the loosest definition of correct, documents judged 3 or above, 9 of the 10 documents are of interest.

We next compared the best concept tree algorithm to the best concept vector algorithm and the keyword vector baseline. To find the best concept vector algorithm, we evaluated the algorithm with no propagation, i.e., $\alpha = 0$, for the same three values of ▯. The best concept vector algorithm occurred when 10 concepts in the user profile were used. To find the keyword vector algorithm, we merely evaluated the results on the 10 documents recommended by combining the results from CiteSeer's built-in tf*idf recommender system as described in Section 4.1.

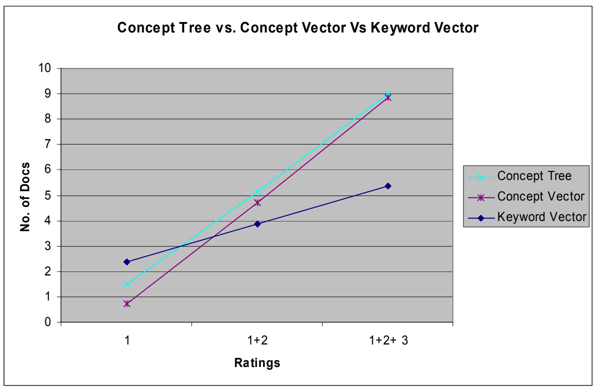

Fig. 3. Concept Tree vs. Concept Vector vs. Baseline

Fig. 3 shows the results comparing the best performing concept tree and concept vector algorithms with the baseline. We also conducted a two-tailed t-test of significance. We found that, for all definitions of correct, the concept tree algorithm outperforms the concept vector algorithm, but this difference was not statistically significant, perhaps because of the small number of authors in the study. The keyword-based algorithm outperforms the concept-based algorithms for the strictest definition of correct only. Although it was only statistically significantly better than the concept vector approach (p=0.005), it was not significantly better than the concept tree approach. The keyword vector approach would provide the author with approximately 1 more highly relevant document, 2.5 vs 1.5 judged 1, in the list of 10. In contrast, the concept tree approach would provide an additional document judged 1 or 2 (5 vs 4). The biggest different occurs when we consider documents judged 3 or above. On average, only 5 documents provided by the keyword approach meet this criteria, meaning that roughly half of the recommended documents are totally irrelevant. However, with the conceptual approaches, only 1 of the 10 documents is not at least somewhat relevant. For documents judged 2 or above and 3 or above, both methods based on a conceptual representation of the documents outperform the keyword-based algorithm and these results are statistically significant (p= 0.001).

5 Conclusions

In this work, we presented a novel way recommending technical papers to the users of the CiteSeer. We represent the user profiles and the documents as content tree and used a tree matching algorithm to compute the similarity between them. To evaluate our system we conducted a user study involving 8 authors with published papers in the CiteSeer collection. We compared recommendations provided by CiteSeer's built-in tf*idf keyword vector representation, a concept vector-based matching algorithm, and a concept-tree based algorithm based on the tree-edit distance measure. We obtained the best results when the propagation factor used to convert the concept vector into concept tree was 0.33 and we used the highest weighted 10 concepts in the user profile to create the unranked set of candidate documents to recommend.

We conclude the following from our results:

1. The concept tree matching algorithm performed much better than the traditional algorithm based on keywords for providing recommendations. The result was found to be statistically significant. We found an improvement of 8% and 31% on average for the documents judged 2 and above and 3 and above, respectively.
2. The concept tree method performed better than the concept vector method. We found an improvement of 6% to 9% on the average. However, this result was not statistically significant.

The list of documents used to build the user profile for an author was created by manually querying the CiteSeer system. Future work will focus on automating this process so that this improved recommender system can be deployed on the CiteSeer site. In addition, we want to explore the effect of a hybrid approach combining keyword and conceptual matches to see if we can get improvements for highly relevant documents.

References

1. Bollacker, K., Lawrence, S., Giles, C.L.: Citeseer: an autonomous web agent for automatic retrieval and identification of interesting publications. Agents'98, 2nd International ACM Conference On Autonomous Agents, pp. 116-123 (1998).
2. Torres, R., McNee, S.M., Abel, M., Konstan, J.A., Riedl, J.: Enhancing digital libraries with techlens. In: Digital Libraries. Proceedings of the Joint ACM/IEEE Conference, pp. 228 – 236. June (2004).
3. Salton, G., Buckley, C.: Term weighting approaches in Automatic Text Retrieval. Information Processing and Management, 24(5), pp. 513-523 (1988).
4. Burke, R.: Hybrid recommender systems: survey and experiments. User Modeling and User-adapted Interaction, 12(4), pp. 331-370 (2002).
5. Huang, Z., Chung, W., Ong, T-H., Chen, H.: A graph-based recommender system for digital library. In: Proc. Joint Conf. on Digital Libraries, pp. 65-73. Portland (2002).
6. Chen, H., Ng, T.: An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): symbolic branch-and-bound search vs. connectionist hopfield net activation. Journal of the American Society for Information Science, 46 (5), pp. 348-369 (1995).

7. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation, Communications of the ACM, v.40 n.3, pp.66-72, March (1997).
8. Basu, C., Hirsh, H., Cohen, W., Nevill-Manning, C.: Technical paper recommendation: a study in combining multiple information sources. Journal of Artificial Intelligence Research, (14) pp. 231-252. (2001).
9. Cohen, W.: The WHIRL approach to information integration. In Trends and Controversies, Marti Hearst ed., IEEE Intelligent Systems, pp.20-23. October (1998).
10. Billsus, D., Pazzani, M.J., Chen, J.: A learning agent for wireless news access. Proc. of the International Conference on Intelligent User Interfaces, pp. 33-36. (2000).
11. Si, L., Jin, R.: Flexible mixture model for collaborative filtering. Proc. 20th Int'l Conf. Machine Learning, pp. 704-711. August (2003).
12. Hofmann, T., Puzicha, J.: Latent class models for collaborative filtering. In the Proceedings of IJCAI, pp. 688-693. (1999).
13. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 39(1):1–38, 1977.
14. Hofmann, T., Puzicha, J.: Statistical models for co-occurrence data (Technical Report). Artificial Intelligence Laboratory Memo 1625, M.I.T. (1998).
15. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender systems-a case study. In ACM WebKDD Wk.shop (2000).
16. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), pp. 391-407. (1990).
17. Zhang, Y., Callan, J., Minka, T. : Novelty and redundancy detection in adaptive filtering. Proc. ACM SIGIR 2002, pp.81-88. (2002).
18. http://en.wikipedia.org/wiki/Kullback-Leibler_divergence
19. Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A.: User Profiles for Personalized Information Access. Brusilovsky P., Kobsa A., and Nejdl W.(Eds.): The adaptive web, LNCS 4321, pp. 54 – 89. (2007).
20. Geisler, G., McArthur, D., Giersch, S.: Developing recommendation services for a digital library with uncertain and changing data. In Proceedings of the 1st ACM/IEEE-CS . JCDL '01. ACM Press, pp. 199-200. New York, (2001).
21. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems. Brusilovsky P., Kobsa A., and Nejdl W.(Eds.): The adaptive web, LNCS 4321, pp. 291 – 324, 2007
22. CiteSeer.IST Scientific Literature Digital Library, http://citeseer.ist.psu.edu/
23. Pazzani, M.J., Billsus, D.: Content-Based Recommendation Systems. Brusilovsky P., Kobsa A., and Nejdl W.(Eds.): The adaptive web, LNCS 4321, pp. 325 – 341, 2007
24. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. On Knowledge and Data Engineering 17 (6), pp. 734-749. June (2005).
25. Breese, J.S., Heckerman, D. Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. Proc. 14th Conf. Uncertainty in AI, pp. 43-52. July (1998).
26. Lakkaraju, P., Gauch, S., Speretta, M.: Document Similarity Based on Concept Tree Distance. 19th International Conference on Hypertext and Hypermedia (Hypertext 2008), Pittsburgh, PA, June 19-21, 2008, (to appear).
27. The ACM Computing Classification System, http://acm.org/class/1998/