December 1998

A Simulation Model of Document Information Retrieval System with Relevance Feedback

Praveen Pathak
University of Michigan

A Simulation Model of Document Information Retrieval System with Relevance Feedback[1]

Praveen Pathak
Department of Computer & Information Systems
University of Michigan

Abstract

Information retrieval systems are very complex in nature due to the complex interaction of document, query, and matching subsystems involved in the process. Researchers working on developing new techniques to improve system performance typically use standardized document collections to test their techniques. This paper has a twofold objective. First, it presents a simulation model of the document information retrieval system. A simulation model can serve as a test-bed document collection that can be used by researchers in the area to test their techniques. A few experiments are run on this simulated collection to demonstrate the effect of varying queries, documents, and user requirements. Second, relevance feedback is employed in the model whereby the matching function used by the system, to match documents with queries, is progressively adapted to the preference function used by the user to judge relevance of the documents.

Introduction

A document based information retrieval (IR) system typically consists of various subsystems as shown in figure 1. There are users with varying information requirements both in terms of breadth and depth of topics they are interested in. A document collection consists of documents describing various different topics of interests. Users express their information requirements in terms of queries issued to the system. A system matching function matches the information in queries with that in the documents and calculates a score 'retrieval status value' (*rsv*). Typically the documents are presented to the user in decreasing order of rsv. The user rates these documents as either relevant or non-relevant to his/her information needs. While rating these documents the user implicitly compares his/her implicit preference function with that of the system. Various system performance criteria like precision and recall have been used to gauge the effectiveness of the system. Relevance ranking feedback is typically used by the system to improve document descriptions or queries.

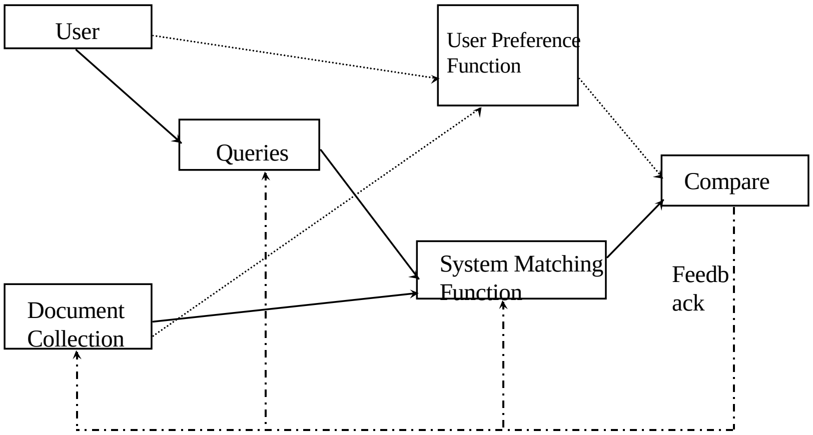

Figure 1

IR researchers have focussed on developing new techniques to improve the overall system performance. Testing these new techniques have been typically done on either the standardized document collections available (e.g. Cranfield, TREC etc.) or by running experiments with actual users. The goal of this paper is two-folds. First, we developed a simulation model of the complete document IR system. A simulation model will help IR researchers in pre-testing their techniques. It allows for carrying out exploratory research which might be difficult (if not impossible) in real life settings. The second goal of the paper is to see how relevance feedback can be used to modify the system matching function. Previous studies about employing relevance feedback (Gordon 88, Salton 90) have typically focussed on document (and document clusters) and query redescription. Very little research has been done to utilize feedback to adapt the system matching function. In this paper we treat the system matching function as a weighted combination of four different measures of association used in literature (Cosine, Jaccard, Dice, and Overlap). The system matching function is adapted to the user's preference function by adapting the weights associated with each measure of association.

[1] Details of the model and results of detailed experiments are available at http://www.umich.edu/~praveen/research/irmodel.html.

The next section describes how modeling was done for each subsystem and also talks about model validity and verification. Following this, we describe results of a few experiments that were carried out. The paper concludes with possible future directions for research in this area.

Modeling Subsystems

Documents: The collection was assumed to have documents relevant to four different broad topic areas. Index terms were used to describe each area. A document was described as a collection of index terms (presence or absence of them). A document was modeled as describing one primary topic, and one secondary topic. For a document about 60% to 80% (chosen randomly) of the terms describing primary topic were marked present, about 30% to 50% of the secondary topic and 10% to 30% of the remaining topics were marked as present.

Users: Users are categorized as either having 'narrow' interest or having 'broad' interests. While 'narrow' users were interested in only one topic out of four, the 'broad' users were interested in a primary topic and also in a secondary topic, modeled using weights for each topic. Primary weights were uniformly distributed between 50% and 70% (meaning 50% to 70% of the terms in user's interests came from the primary topic) and the remaining terms were allocated for secondary topic.

Apart from differing over the scope of their interests, the users were also modeled to differ about the number of retrieved documents they were willing to look at. This number varied between 20 and 30.

User Preference Function: 'Cosine' function (cosine between the document and query vector), a popular function used in IR systems (van Rijsbergen 1979), has been used to model user's preference function.

Index Terms: An index term is modeled to be describing a primary topic and remaining secondary topics. Although the primary topic is the main topic described by the term, we allow for other topics as well to accommodate varying interpretations of the terms based on the context in which it occurs in the document. This feature is achieved using a primary weight (distributed normally with mean 0.65 and standard deviation of 0.05) and the remaining weights for the other topics.

Queries: Queries have been created in terms of the index terms used. A query is binary in nature i.e. in a query the term is either present or absent. A number of queries are created for users. Queries have been divided into those catering to 'narrow' users (described earlier) and those to 'wide' users. Query length (the number of terms present in the query) is normally distributed with mean 12 and sd 1.5 thus having 96% of the queries with length between 9 and 15 terms. To select the terms in the query first a user is selected at random. For narrow users 85% of terms come from the primary topic of the user while for wide users 50% to 70% of the terms come from primary interests and the remaining terms come from secondary and remaining interests.

System Matching Function: The system matching function has been modeled as a weighted sum of scores produced by Cosine, Dice, Jaccard, and Overlap matching functions. Thus the overall score is calculated based on: (a1*fn1 + a2 * fn2 + a3 * fn3 + a4 * fn4), where a1, a2, a3, and a4 are the weights and fn1, fn2, fn3, and fn4 are the scores produced by individual matching functions (matching document vector to the query vector). The overall score is used to rank the documents in the collection. A parameter called DCV (document cut-off value) is used to present the user with the top DCV number of documents (stated in the user's preference about number of documents the user is willing to look at) based on the overall score.

Relevance of a Document: The top DCV documents (from the results of system matching function) are presented to the user as a result of his/her search request. A document is considered relevant to a user if it exists in the top DCV number of documents produced by the user's preference function.

Adaptation of the System Matching Function: Adaptation is done by training the weights a1 to a4 in such a way (a kind of hill-climbing) that the precision score (ratio of number of retrieved relevant documents to the total number of retrieved documents) is maximized. For a starting combination of the weights, average precision over all queries is calculated. To start the process, a weight is chosen and changed at random. If the average precision increases then the new weight is kept otherwise the previous value of weight is reinstated. This process continues till the average precision does not change for four consecutive runs or till the maximum number of runs is reached. We allow for no change in average precision for four consecutive runs as this may allow the system to come out of the local maxima and search other maxima.

Model Implementation: Implementation was done using the C language. Model verification techniques suggested by Sargent (1987) were carried out. Internal validity and face validity was done for the model.

Experiments and Results

The parameter set used in the experiments was as follows: Number of terms varied from 80 to 140; number of documents varied from 50 to 125; DCV (document cutoff value) varied form 10 to 25. Queries were of three types: narrow (all queries narrow as explained earlier), wide (all queries wide), and mid (1/2 the queries were narrow and remaining wide). Increment/decrement value for training weights varied from 0.05 to 0.25.

For various parameter values a set of users, documents, terms, and queries were created. The process of issuing queries, retrieving documents, and ranking them as relevant/non-relevant was carried out. Adaptation of weights was performed to progressively get higher average precision values. Maximum precision value reached varied between 0.65 and 1.0. Each experiment was run using seven different starting seed values for the random number generator. Performance was measured in terms of the maximum average precision value reached and the median number of runs required to reach maximum precision value. Statistical analysis showed no significant differences for various values of number of terms used, and for different query types used. However, significant differences were found between increment values of 0.05 and other values (0.1, 0.15, 0.2, and 0.25). This can be explained as follows: when the increment value is small adaptation tends to find local maxima and gets stuck at it, while with higher increment values the search for even higher maxima continues. Significant differences were also found for DCV values of 20 and 25.

A graphical analysis was done to look for differences. It was found that narrow query types yielded higher average precision values than other query types. As the increment values for weights were increased, the maximum average precision values also increased.

Future Directions

The model presented here is a simple model. To make it more realistic we will need to incorporate following changes: 1. Rather than having query terms as binary (present or absent), weights can be used for terms to signify the strength of the need for information. 2. Document terms could also be weighted. 3. While adapting the weights for individual functions in the system matching process, the increment values of the weights can be made to depend on observed differences between current and previous precision values. 4. More sophisticated learning algorithms like Genetic Algorithms can be employed to learn these weights.

Conclusions

The paper developed a simulation model that can serve as a test-bed on which IR researchers can carry out exploratory research (for example, to test new algorithms). The paper also discussed a method of adapting the system matching function to the implicit preference function used by the person issuing queries.

References

Gordon, M.D. “Probabilistic and genetic algorithms for document retrieval”, Communications of the ACM, 31(10), 1988, pp: 1208-1218

Salton, G. & Buckley C. “Improving retrieval performance by relevance feedback”, Journal of the American Society for Information Science, 41(4), 1990, pp: 288-297

Sargent, R.G. “A tutorial on validation and verification of simulation models”, Proceedings of the 1988 Winter Simulation Conference, 1988, pp: 33-39

van Rijsbergen, C.J. “Information Retrieval”, Butterworth, 1979