Implementing Relevance Feedback in the Bayesian Network Retrieval Model

Luis M. de Campos
Departamento de Ciencias de la Computación e I.A., E.T.S.I. Informática, Universidad de Granada, 18071, Granada, Spain. E-mail: lci@decsai.ugr.es

Juan M. Fernández-Luna
Departamento de Informática, Escuela Politécnica Superior, Universidad de Jaén, 23071, Jaén, Spain. E-mail: jmfluna@ujaen.es

Juan F. Huete
Departamento de Ciencias de la Computación e I.A., E.T.S.I. Informática, Universidad de Granada, 18071, Granada, Spain. E-mail: jhg@decsai.ugr.es

Relevance Feedback consists in automatically formulating a new query according to the relevance judgments provided by the user after evaluating a set of retrieved documents. In this article, we introduce several relevance feedback methods for the *Bayesian Network Retrieval Model.* The theoretical frame on which our methods are based uses the concept of partial evidences, which summarize the new pieces of information gathered after evaluating the results obtained by the original query. These partial evidences are inserted into the underlying Bayesian network and a new inference process (probabilities propagation) is run to compute the posterior relevance probabilities of the documents in the collection given the new query. The quality of the proposed methods is tested using a preliminary experimentation with different standard document collections.

Introduction

Relevance Feedback is one of the most useful *Query Modification* techniques in the field of *Information Retrieval* (IR). This method is put into practice when the user needs to improve the query formulated to the *IR system,* because the documents initially retrieved do not completely fulfill the user's information need. Relevance feedback works in the following way: a user submits a query representing his/her information need to the IR system, which then ranks the documents according to their corresponding degrees of relevance to the query (with the documents most closely matching the query ranked first). The user then inspects this list,[1] and determines which documents are relevant and which are not relevant to his/her information need (the *relevance judgments*). Using this information, the IR system updates the initial query, modifying the importance of the terms it contains[2] (*term reweighting*), and adding new terms that are considered useful to retrieve more relevant documents (*query expansion*). This process is repeated until the user is completely satisfied with the set of retrieved relevant documents. Relevance feedback has been successfully applied in a great variety of IR models. A good review of this technique can be found in (Spink & Losee, 1996).

Probabilistic models constitute an important kind of IR models, which have been widely used for a long time (Jones, Walker, & Robertson, 2000a, 2002b; Maron & Kuhns, 1960; Robertson & Jones, 1976). They offer a principled way of managing the uncertainty that naturally appears in many elements within this field. For example, queries are only approximate representations of the user's internal information needs, and document representations, in the form of term-weight vectors or whatever, cannot completely capture the contents of documents. Nowadays, the dominant approach to managing probability within the field of Artificial Intelligence is based on the use of Bayesian networks (Pearl, 1988). They have also been successfully applied in the IR environment as an extension/modification of probabilistic IR models, giving rise to the *Infer-*

[1] Actually he/she only inspects the top b documents in this list, for a given value of b.

[2] Giving more importance to the terms appearing in relevant documents, and weakening the strength of those that belong to nonrelevant documents.

ence Network (Turtle, 1990; Turtle & Croft, 1990, 1991), *Belief Network* (Silva, 2000; Ribeiro-Neto & Muntz, 1996), and *Bayesian Network Retrieval* (de Campos, Fernández-Luna, & Huete, 2000, 2002a, 2002b; Fernández-Luna, 2001) models, as well as other approaches (Fung & Favero, 1995; Ghazfan, Indrawan, & Srinivasan, 1996).

In this article, we will introduce an approach for relevance feedback in the *Bayesian Network Retrieval* (BNR) model. Our proposal is based on the propagation of partial evidences in the Bayesian network, representing the new information obtained from the user's relevance judgments to compute the posterior relevance probabilities of the documents.

The remainder of the article is organized as follows: The next section describes the BNR model, showing its most important features. Then we deal with a theoretical overview of the proposed relevance feedback method for the BNR model. Next, we explain how we have put the ideas developed in the previous section into practice, which have resulted in several feedback techniques, followed by showing the experimental results obtained by our methods when they were applied to several well-known standard test collections. Finally, we discuss the conclusions of this work and introduce new research lines in this area.

Description of the Bayesian Network Retrieval Model

In this section, we will briefly describe the *Bayesian network retrieval model* to have the knowledge needed to understand the relevance feedback methods that will be proposed later. This model is explained in more detail in Fernández-Luna (2001), and de Campos et al. (2002a).

Bayesian networks are graphical models that are capable of efficiently representing and manipulating n-dimensional probability distributions (Pearl, 1988). A Bayesian network uses two components to codify qualitative and quantitative knowledge: (a) A *Directed Acyclic Graph* (DAG), $G = (V, E)$, where the nodes in V represent the random variables from the problem we want to solve, and the topology of the graph (the arcs in E) encodes conditional (in)dependence relationships among the variables (by means of the presence or absence of direct connections between pairs of variables); (b) a set of conditional probability distributions drawn from the graph structure: For each variable $X_i \in V$ we have a family of conditional probability distributions $P(X_i|pa(X_i))$, where $pa(X_i)$ represents any combination of the values of the variables in $Pa(X_i)$, and $Pa(X_i)$ is the parent set of X_i in G. From these conditional distributions we can recover the joint distribution over V:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} (P(X_i|pa(X_i)) \qquad (1)$$

This decomposition of the joint distribution results in important savings in storage requirements. It also allows probabilistic inference (propagation) to be performed in many cases, i.e., the posterior probability to be computed for any variable given some evidence about the values of other variables in the graph (Jensen, 1996; Pearl, 1988).

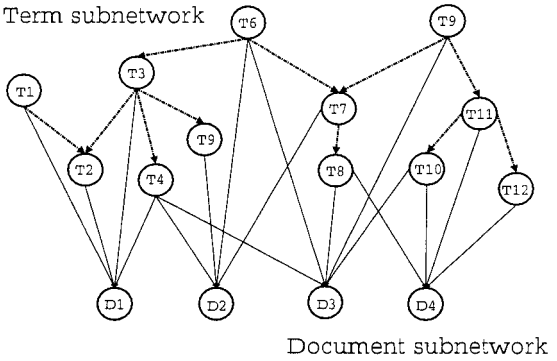

FIG. 1. Network structure in the BNR model.

The set V in the Bayesian Network Retrieval model is composed of two different sets of variables, $V = \mathcal{T} \cup \mathcal{D}$: The set $\mathcal{T} = \{T_1, \ldots, T_M\}$ of the M terms in the glossary from a given collection, and the set $\mathcal{D} = \{D_1, \ldots, D_N\}$ of the N documents that comprise the collection.[3] Each document variable, D_j, is a binary random variable taking values in the set $\{\bar{d}_j, d_j\}$, where $\bar{d}_j$ stands for "the document D_j is not relevant for a given query," and d_j represents "the document D_j is relevant for a given query." We say that a document is relevant for a given query if it satisfies the user's information need expressed using this query. Similarly, a variable referring to a term T_i has its domain in the set $\{\bar{t}_i, t_i\}$, where in this case, $\bar{t}_i$ and t_i respectively mean "the term T_i is not relevant," and "the term T_i is relevant." In this case, we talk about the relevance of a term in the sense that the user believes that this term will appear in relevant documents (hence, he/she will explicitly use it when formulating the query). Similarly, a term is not relevant when users believe that the relevant documents do not contain it: they are not interested in documents containing this term (and, therefore, they could also include it in the query, but in this case in a negative form).

Focusing on the structure of the network in the BNR model (see Fig. 1), we can distinguish two subsets of arcs: (1) the first includes arcs between the term and document nodes. For each term indexing a document, there is an arc from the term node to the node associated with the document it belongs to. (2) The second one only includes arcs connecting pairs of term nodes. These arcs represent dependence relationships between terms, and their inclusion improves the performance of the IR system (de Campos et al., 2002a; Fernández-Luna, 2001). Note that the network does not contain arcs linking document nodes[4] (the dependence

[3] We will use the notation T_i (D_j, respectively) to refer to the term (document, respectively) and also to its associated variable and node.

[4] This restriction is removed in the BNR model described in (de Campos et al., 2002b).

relationships between documents are not direct, they always depend on the terms included in these documents). To capture the relationships between terms in the collection, a Bayesian network learning algorithm is used to construct the term subnetwork. Taking into account efficiency considerations in the learning and inference stages, some restrictions on the structure of the learned graph have been imposed. In particular, the relationships between terms are represented by means of a polytree (a DAG in which there is no more than one undirected path connecting each pair of nodes). The used polytree learning algorithm is described in detail in (de Campos, Fernández, & Huete, 1998).

To estimate the quantitative component of the Bayesian network, i.e., the probability distributions stored in each node, we distinguish three different cases:

1. Term nodes having no parent (root nodes): in this case we store marginal distributions, estimated as follows: $p(t_i) = 1/M$ and $p(\bar{t}_i) = (M - 1)/M$, where M is the number of terms in the collection.
2. Term nodes with parents: for each node, we need to store a set of conditional probability distributions $p(T_i|pa(T_i))$, one for each possible configuration of the parent set $Pa(T_i)$. These distributions are simply estimated using frequencies of cooccurrence of terms in the documents in the collection.
3. Document nodes: in this case, the number of conditional probabilities that we need to estimate and store for each D_j grows exponentially with the number of parents of D_j: for example, if D_j has been indexed by 30 terms (and this is not uncommon), we need 2^{30} (approx. 1.07×10^9) probability distributions, which is obviously unfeasible. So, instead of explicitly computing and storing these probabilities, the BNR model uses a *probability function* (also called a canonical model of multicausal interaction (Pearl, 1988)), which returns a conditional probability value when it is called during the propagation stage, whenever a conditional probability is required. In this article, we use the following probability function, based on the cosine measure (Salton & McGill, 1983): for any configuration $pa(D_j)$ of $Pa(D_j)$ (i.e., any assignment of values to all the term variables in D_j), the probability of relevance of document D_j is

$$p(d_j|pa(D_j)) = \alpha_j \sum_{\substack{T_i \in D_j \\ t_i \in pa(D_j)}} tf_{ij} idf_i^2 \qquad (2)$$

where tf_{ij} is the frequency of the term T_i in the document D_j, idf_i is its inverse document frequency,

$$\alpha_j = \frac{1}{\left(\alpha \sqrt{\sum_{T_i \in D_j} tf_{ij} idf_i^2} \right)}$$

and α is a normalizing constant. The expression $t_i \in pa(D_j)$ in Equation 2 means that we only include in the sum those weights such that the value assigned to the corresponding term T_i in the configuration $pa(D_j)$ is t_i. So, the higher the number of relevant terms in $pa(D_j)$, the greater the probability of relevance of D_j.

Given a query Q submitted to our system, the retrieval process begins by placing the evidences in the term subnetwork: the value of each term variable T_k belonging to Q is fixed to relevant,[5] $[T_k = t_k]$. The inference process is then run, obtaining the probability of relevance of each document D_j, given that the terms in the query are also relevant, $p(d_j|Q)$. Finally, the documents are sorted in decreasing order of probability to perform the evaluation process, and these will be shown to the user.

Taking into account the size and topology of the BNR model, and due to efficiency considerations, general-purpose inference algorithms cannot be applied, even for small document collections. To solve this problem, a specific inference process comprising two steps has been designed: *propagation + evaluation.* It is important to note that this process ensures that the results are the same as those obtained using exact propagation in the entire network (de Campos et al., 2002a; Fernández-Luna, 2001): First, we propagate the query only in the term subnetwork using Pearl's Exact Propagation Algorithm in Polytrees (Pearl, 1988), and compute the probability of relevance for each term given the query, $p(t_i|Q)$. In the second step, we compute the probability of relevance of each document given the query by means of the evaluation of the following formula:

$$p(d_j|Q) = \alpha_j \sum_{T_i \in D_j} tf_{ij} idf_i^2 p(t_i|Q) \qquad (3)$$

In this way, we are performing exact probabilistic inference in a complex network very efficiently. Note that as $p(t_i|Q) = 1 \ \forall T_i \in Q$, then Equation 3 becomes

$$p(d_j|Q) = \alpha_j \left(\sum_{T_i \in D_j \cap Q} tf_{ij} idf_i^2 + \sum_{T_i \in D_j \setminus Q} tf_{ij} idf_i^2 p(t_i|Q) \right) \qquad (4)$$

Related Work

In this section, we will briefly describe the two main retrieval models based on Bayesian networks, comparing them with our model and establishing the main differences.

The first model was developed by Croft and Turtle (Turtle, 1990; Turtle & Croft, 1990, 1991), the *Inference Network Model,* which in its simplified form comprises two networks: the document, and query networks. The former represents the document collection and contains two kinds of nodes: the document nodes, representing the documents, and the concept nodes, symbolizing the index terms contained in the documents. The arcs go from each document node to each concept node used to index it. The document network is fixed for a given collection. However, the query

[5] Although the BNR model can also deal with queries including terms in negative form, we will assume here the more common situation where this does not happen.

network is dynamic in that it is specific for each query, and comprises three types of nodes: The Information Need node (inn), which represents the user's generic information need; a set of intermediate query nodes, used when there are multiple query representations; and finally, the query concept nodes (in simplified form, these are only the concept nodes in the document network, and represent the connection between the two networks). The arcs in the query network go from query concept nodes to query nodes, and from query nodes to the Information Need node. Each type of node stores a probability matrix, called the link matrix in their notation, which in certain cases, depends on the type of query being formulated (Boolean or probabilistic). The retrieval is performed by instantiating a single document node D_j each time, and computing the probability that the Information Need is satisfied given that this document has been observed, $p(inn|d_j)$. In reality, Turtle and Croft precompute the intermediate probabilities $p(t_i|d_j)$ in the document network, and, later use closed-form expressions to evaluate $p(inn|d_j)$ as a function of the probabilities $p(t_i|d_j)$, for those terms T_i appearing in the query submitted by the user.

A first difference between this approach and ours is that we do not have a query network. A second distinction, also topologic, is that the arcs in our model are directed in the opposite direction (from term nodes to document nodes). We consider it to be more intuitive to speak about the probability of a document being relevant given a query than the contrary. Our choice therefore implies instantiating the query, or specifically, the terms it contains, and propagating towards the document nodes. This means that we only have to propagate once, unlike Turtle and Croft's model in which they have to run one propagation per document.

Ribeiro and Silva's model (Ribeiro-Neto & Muntz, 1996; Silva, 2000; Silva, Ribeiro-Neto, Calado, Moura, & Ziviani, 2000) is designed to simulate the Vector Space, Boolean, and Probabilistic models. Their network comprises three types of nodes: document nodes, concept nodes, and the query node. The arcs go from the concept nodes to the document nodes where they occur, and from the concept nodes (appearing in the query) to the query node. In this model, the probabilities of interest are $p(d_j|Q)$, which could be computed as

$$p(d_j|Q) = \alpha^{-1} \sum_{\tau} p(d_j|\tau)p(Q|\tau)p(\tau), \qquad (5)$$

where τ represents any of the 2^M assignments of values to all the terms in the collection. This computation is obviously unfeasible. Depending on the model to be simulated, the probabilities $p(Q|\tau)$ and $p(\tau)$ are therefore defined so that all the terms in the previous addition except one (corresponding to a given configuration τ_Q) are always equal to zero. Thus, the computation in Equation 5 becomes straightforward: the inference is reduced to evaluate a function ($p(d_j|\tau_Q)$) in the only nonzero configuration.

The main differences appear in the conditional probability distributions considered. In our case, these distributions are not "degenerated" and do not depend on the query, and we do truly perform probability propagation.

Another important difference between these two models and ours is that we include direct relationships between the terms, thereby obtaining a more expressive model.

A Methodology for Relevance Feedback in the BNR Model

The proposed methodology for relevance feedback in the BNR model is based on the idea that by evaluating a set of retrieved documents obtained as a consequence of running a query Q, the user obtains new pieces of evidence that may help to discriminate which documents are relevant to his/her information need. After inserting these new evidences into the network, we can build a new query Q_1, rerun the inference process, and obtain the new posterior probabilities of relevance, $p(d_j|Q_1)$. The main advantage of this methodology is that we can include the evidences in the model without adding new nodes nor reestimating the probability distributions stored in the network (as the Inference Network model does; Haines & Croft, 1993).

The key technical question is how the new evidential information is entered in the Bayesian network. The natural approach would be to instantiate each evaluated document to its corresponding relevance value. So, if we have judged b documents, the set $\{D_{k_1}, \ldots, D_{k_h}\}$ contains the documents that are relevant and $\{D_{k_{h+1}}, \ldots, D_{k_b}\}$ those that are not relevant, the new query Q_1 would be:

$$Q_1 = Q \wedge [D_{k_1} = d_{k_1}, \ldots, D_{k_h} = d_{k_h}, D_{k_{h+1}} = \bar{d}_{k_{h+1}}, \ldots, D_{k_b} = \bar{d}_{k_b}].$$

However, this approach is unfeasible because as soon as we instantiate a document node, we cannot use the propagation + evaluation algorithm and we would have to resort to the general-purpose, time-consuming propagation algorithms. It is therefore necessary to develop a new two-step method to efficiently approximate the desired probabilities.

To maintain the computational advantage of using the propagation + evaluation algorithm, our proposal is to translate the *definite* evidences about the relevance of the evaluated documents to *partial* evidences about the relevance of the terms indexing these documents. Nevertheless, this information must be combined with the previous evidence supported by the original query to formulate a new query to be processed by the IR system. For example, whenever a term T_i indexes a document judged relevant by the user, perhaps we should increase the belief supporting T_i's relevance; similarly, if a term T_i only appears in documents which are not relevant, we should decrease its relevance degree.

To better understand the way in which we are going to perform the inclusion of partial evidence in the term nodes, it is useful to briefly explain how Pearl's Exact Propagation Algorithm for Polytrees works: The key property of a poly-

tree is that the variables that are ancestors and those that are descendants of a given variable X, are conditionally independent given X. This fact allows an efficient inference scheme to be designed in which, given some evidence ev, each uninstantiated node X receives a vector message $\pi_X(Z)$ from each of its parent nodes Z, representing predictive evidence from all the nondescendants of X mediated by X's parents; X also receives a vector message $\lambda_Y(X)$ from each of its child nodes Y, representing the diagnostic or retrospective support from X's descendants. The λ_Y messages are combined in order to obtain a single vector $\lambda(X) = \Pi_Y \lambda_Y(X)$; all the π_X messages are also combined with the conditional probabilities $p(X|pa(X))$ to obtain a vector $\pi(X)$. The product $\lambda(X)\pi(X)$, after normalization, contains the posterior probability $p(X|ev)$. Moreover, each instantiated node X also receives a $\lambda_o(X)$ vector from a dummy node, an imaginary child of X, codifying the observed value of the variable X. In the case of a binary variable, either $\lambda_o(X) = (1, 0)$ or $\lambda_o(X) = (0, 1)$, this represents the observations $[X = \bar{x}]$ and $[X = x]$, respectively. If the available evidence about a variable X is a partial rather than a definite observation, then the λ_o vector is interpreted as conveying likelihood information, $\lambda_o(X) = (p(Obs|\bar{x}), p(Obs|x))$, where *Obs* stands for *Observation.* In this case, only the likelihood ratio $p(Obs|\bar{x})/p(Obs|x)$ is important, so, for instance, the vectors $(p(Obs|\bar{x}), p(Obs|x))$ and $(p(Obs|\bar{x})/p(Obs|x), 1)$ are equivalent. Note that a value $p(Obs|\bar{x})/p(Obs|x) < 1$ implies evidence in favor of x: the smaller this value is, the more evidence there is supporting the assignment $[X = x]$. This λ_o message will be combined with the information that node X obtains from its parents (π vector) and children (λ vector) in the network, to obtain the posterior probability $p(X|ev) \propto \lambda_o(X)\lambda(X)\pi(X)$.

Remark: the use of λ_o vectors can be extended for the rest of the nodes in the network, so that we can assume that every node will receive a λ_o vector: for uninstantiated nodes that do not receive any evidence, we use the vector $\lambda_o(X) = (1, 1)$.

Now, taking into account these comments about Pearl's Propagation Algorithm, it is clear that the design of a relevance feedback method for the BNR model, within the proposed methodology, is reduced to deciding the λ_o messages that the term nodes will receive. Obviously, only those terms indexing the observed documents will receive λ_o vectors which are different from (1, 1): the terms that do not appear in any retrieved document will remain unaltered.

We shall now introduce some notation that will be used in the rest of the paper: R_Q is the set of documents retrieved and evaluated for a given query Q, and $|R_Q|$ is the number of such documents; n_r and $n_{\bar{r}}$ represent the number of retrieved relevant and nonrelevant documents, respectively; n_{t_i} and $n_{\bar{t}_i}$ denote the number of retrieved documents where the term T_i has been observed and not observed, respectively; n_{rt_i} is the number of retrieved relevant documents in which the term T_i appears; $n_{\bar{r}t_i}$ is the number of retrieved nonrelevant documents including the term T_i; $n_{r\bar{t}_i}$ is the number of retrieved relevant documents that do not include the term T_i; $n_{\bar{r}\bar{t}_i}$ is the number of retrieved nonrelevant documents that do not include the term T_i. All these numbers form a contingency table (see Table 1), which lists the relevance of the documents where the term T_i appears.

TABLE 1. Contingency table for term T_i.

	$T_i = \bar{t}_i$	$T_i = t_i$	
$\bar{r}$	$n_{\bar{r}\bar{t}_i}$	$n_{\bar{r}t_i}$	$n_{\bar{r}}$
r	$n_{r\bar{t}_i}$	n_{rt_i}	n_r
	$n_{\bar{t}_i}$	n_{t_i}	$\|R_Q\|$

Relevance Feedback Methods for the BNR Model

It is useful to classify the terms indexing retrieved documents in the three following categories: terms that only occur in relevant documents (*positive terms,* $\mathcal{T}^+$), those that only occur in nonrelevant documents (*negative terms,* $\mathcal{T}^-$), and those that occur in both types of documents (*neutral terms,* $\mathcal{T}^\pm$). Moreover, we will also distinguish between terms indexing retrieved documents that appear in the original query and those which do not. The first type, *query terms* ($\mathcal{T}^q$), will be used for term reweighting, and the second, *expansion terms,* ($\mathcal{T}^e$), for query expansion.

Query Term Reweighting

The terms in $\mathcal{T}^q$, belonging to the original query, were instantiated as relevant (i.e., they received a message $\lambda_o(T) = (0, 1)$). However, those query terms that only appear in nonrelevant documents ($T_i \in \mathcal{T}^q \cap \mathcal{T}^-$) have not performed well. Consequently, they should be penalized by decreasing the belief supporting their relevance. An extreme option would be to use a message $\lambda_o(T_i) = (1, 1)$, thus considering them to be nonquery terms. Nevertheless, these terms have been explicitly selected by the user, so that a more conservative option is to use a vector $\lambda_o(T_i) = (\gamma_{t_i}, 1)$, with $0 < \gamma_{t_i} < 1$. The method proposed in this paper considers a γ_{t_i} value sensible to the number of nonrelevant documents in which the term T_i occurs, $n_{\bar{r}t_i}$: the higher this number, the greater γ_{t_i} will be (thus rendering the λ_o vector close to the uninformative vector (1, 1)). The following expression satisfies this requirement, assessing γ_{t_i} values verifying $0.5 \leq \gamma_{t_i} < 1$:

$$\lambda_o(T_i) = \left(1 - \frac{1}{n_{\bar{r}t_i} + 1}, 1\right) \tag{6}$$

We have also tested a different approach for determining the value γ_{t_i}: using a fixed value γ for all the negative query terms. However, some preliminary experimentation showed that the method in Equation 6 performed almost as well as the method using a fixed γ value, for the best choice of γ, with the main advantage that we do not have to tune this parameter.

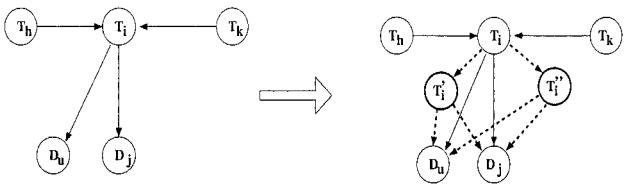

FIG. 2. Replicating three times the query term node T_i.

On the other hand, the terms in $\mathcal{T}^q \cap \mathcal{T}^+$ and to a lesser extent, those in $\mathcal{T}^q \cap \mathcal{T}^\pm$, have performed excellently, and have been useful when it comes to retrieving relevant documents. Therefore, rather than being penalised, these terms should maintain their relevance status or, even better, be rewarded. However, in principle, we cannot increase the relevance of the positive or neutral query terms, because they are already completely relevant. So, the first approach to deal with these terms is simply to maintain them as normal query terms, i.e., they will all receive a vector $\lambda_o(T_i) = (0, 1)$. This method will be denoted *tr-ins.*

Another approach, which aims to increase the importance of each of these terms T_i in the new query, is to *replicate* the corresponding term node in the network as many times as the number of relevant documents where it appears, n_{rt_i}, and instantiate all the duplicate nodes as relevant. We call this approach *tr-rep.* To change the structure of the term subnetwork at least as possible, we only connect the duplicate nodes as children of the original term node T_i and as parents of the documents that contain T_i. Figure 2 displays an example where a node T_i appears in three relevant documents.

The effect of this process is that the weight associated to each of the terms $T_i \in \mathcal{T}^q \cap (\mathcal{T}^+ \cup \mathcal{T}^\pm)$, in the basic propagation formula in Equation 4, changes from $tf_{ij}idf_i^2$ to $n_{rt_i}tf_{ij}idf_i^2$, thus increasing n_{rt_i} times the importance of T_i. Therefore, the replication of the term nodes is only *virtual,* we do not actually have to change the network structure.

It is worth mentioning that query reweighting is not a process of *belief updating,* because we are not only adding new evidence to old and repropagating it in the network. Instead, we are performing a process of *belief revision,* because we are also using the new information to retract previous evidence that is no longer supported.

Query Expansion

We now want to determine the impact of adding new terms to the original query. The terms in $\mathcal{T}^e$, as in the case of $\mathcal{T}^q$, may be positive ($\mathcal{T}^e \cap \mathcal{T}^+$), negative ($\mathcal{T}^e \cap \mathcal{T}^-$), or neutral ($\mathcal{T}^e \cap \mathcal{T}^\pm$).

1. All the negative expansion terms will be directly instantiated to nonrelevant, i.e., they will receive a vector $\lambda_o(T_i) = (1, 0)$: we consider that they are not at all useful for relevant document retrieval. However, it may be useful to use these terms, in negative form, to prevent nonrelevant documents that contain them from appearing in the ranking before other relevant documents.
2. With regard to neutral expansion terms, we use the vector $\lambda_o(T_i) = (1, 1)$ (which is equivalent to not considering them). We have also experimented with other alternatives, taking into account the number of relevant and irrelevant documents where each term appears, but we did not obtain better results.
3. Positive expansion terms are quite promising: these are able to clearly distinguish between relevant and irrelevant documents, but in a way unanticipated by the user, who did not select them as query terms. The naive approach to including these terms in the new query would be to instantiate all of them to relevant, using a vector $\lambda_o(T_i) = (0, 1)$. This method for expanding positive terms will be denoted by *qe-n*. Preliminary experimental results demonstrated that this is not a good choice, as we shall see in the next section. Therefore, we will include the positive expansion terms in the new query as we did with negative query terms, using soft evidence (likelihood values, $\lambda_o(T_i) = (\gamma_{t_i}, 1)$, $\gamma_{t_i} = p(Obs|\bar{t}_i)/p(Obs|t_i)$). Although the process is similar in form, conceptually it is quite different: for negative query terms we try to *decrease* their probability of relevance by increasing the value γ_{t_i}, thus changing the initial λ_o vector from (0, 1) to (γ_{t_i}, 1). However, for positive expansion terms we try to *increase* their probability of relevance by decreasing the value γ_{t_i}, in this case changing the initial λ_o vector from (1, 1) to (γ_{t_i}, 1).

We have studied different approaches to carry out positive query expansion. They will be grouped into two classes. The first uses the available *global* information about each term T_i, obtained from the previously introduced contingency table (Table 1) to build the corresponding $\lambda_o(T_i)$ vector. The second class uses the *local* information provided by each relevant document D_j to elaborate a specific local vector, $\lambda_{D_j}(T_i)$, and then combines all these vectors to obtain the final message $\lambda_o(T_i)$. These two types of methods will be discussed separately in the following subsections.

Positive Query Expansion Using Global Information

We can see the probability of obtaining a relevant document when we consider that the term T_i is relevant, i.e., $p(r|t_i)$ (where r stands for relevant document), as an evidence favoring the relevance of T_i. Analogously, the evidence supporting the non-relevance of T_i could be measured by the probability of obtaining a relevant document given that the term is nonrelevant, $p(r|\bar{t}_i)$. These two values constitute the λ_o vector for each positive expansion term $T_i \in \mathcal{T}^e \cap \mathcal{T}^+$:

$$\lambda_o(T_i) = (p(r|\bar{t}_i), p(r|t_i)) \quad \text{or, equivalently,}$$

$$\lambda_o(T_i) = \left(\frac{p(r|\bar{t}_i)}{p(r|t_i)}, 1\right) \quad (7)$$

These probabilities will be estimated, for each term T_i, from the information obtained by evaluating the retrieved docu-

ments, which is globally summarized in Table 1.[6] Different ways of estimating these probabilities give rise to different positive query expansion methods:

1. *qe-gmle:* Using a maximum likelihood estimator (frequency counts):

$$p(r|t_i) = \frac{n_{rt_i}}{n_{t_i}} \quad \text{and} \quad p(r|\bar{t}_i) = \frac{n_{r\bar{t}_i}}{n_{\bar{t}_i}} \tag{8}$$

2. *qe-gbe:* Using a Bayesian estimator (Cestnik, 1990):

$$p(r|t_i) = \frac{n_{rt_i} + s_{t_i}\dfrac{n_r}{|R_Q|}}{n_{t_i} + s_{t_i}} \quad \text{and} \quad p(r|\bar{t}_i) = \frac{n_{r\bar{t}_i} + s_{\bar{t}_i}\dfrac{n_r}{|R_Q|}}{n_{\bar{t}_i} + s_{\bar{t}_i}} \tag{9}$$

   where the parameters s_{t_i} and $s_{\bar{t}_i}$ represent the *equivalent sample size.* The reason for proposing this method is that the quality of the maximum likelihood estimation may be low, because we are dealing with few data (only the $|R_Q|$ retrieved documents). We, therefore, need some alternatives to fix these parameters s_{t_i} and $s_{\bar{t}_i}$.
3. *qe-gbe1:* The first option is obtained by setting them to a given fixed value. In this article, we have used $s_{t_i} = s_{\bar{t}_i} = n_r$.
4. *qe-gbe2:* Another alternative for computing s_{t_i} and $s_{\bar{t}_i}$ may be obtained by taking into account the total number of documents in which T_i occurs and does not occur in the whole set of documents in the collection (as opposed to the set of retrieved documents). For instance, if a term appears in three documents, and these three documents have been retrieved, s_{t_i} should be equal to 0, because we are using all the possible data to estimate the probability. The opposite case occurs when T_i indexes a large number of documents, for example, 100 documents, and only one is retrieved: s_{t_i} should be high. This behavior can be obtained by computing the ratio between the frequency, tf_{T_i}, of the term T_i in the collection and n_{t_i}, in the following way (N is the number of documents in the collection):

$$s_{t_i} = \frac{\log(tf_{T_i} + 1)}{\log(n_{t_i} + 1)} + 1 \quad \text{and}$$
$$s_{\bar{t}_i} = \frac{\log(N - tf_{T_i} + 1)}{\log(n_{\bar{t}_i} + 1)} + 1 \tag{10}$$

   We use logarithms because in many cases the values computed for the parameters could be extremely high and have to be smoothed.

Positive Query Expansion Using Local Information

In this approach, each term $T_i \in \mathcal{T}^e \cap \mathcal{T}^+$ will receive a local message, $\lambda_{D_j}(T_i)$, from each document D_j that has been evaluated as relevant and contains T_i:

$$\lambda_{D_j}(T_i) = \left(\frac{p(Obs_j|\bar{t}_i)}{p(Obs_j|t_i)},\ 1\right) \tag{11}$$

where Obs_j represents the observation focused on D_j. This vector codifies the likelihood about the relevance or the irrelevance of T_i derived from the knowledge that D_j is a relevant document. All these vectors are then multiplied to obtain the combined vector $\prod_{j=1}^{n_{rt_i}} \lambda_{D_j}(T_i)$. Moreover, to obtain the final λ_o vector, we also want to take into account the quality of the original query, measured as the number of relevant documents retrieved, n_r: when this number is high, then the original query has performed well. Because we do not therefore want to modify it much, we add new terms but in such a way that these terms do not have a strong impact on the original query. On the contrary, if we retrieve a small number of relevant documents, it may be convenient to modify the original query more markedly. In this case, the terms indexing the relevant documents should be added to the query using a weight greater than in the previous case. We have implemented this idea using a convex combination, as follows:

$$\lambda_o(T_i) = \frac{n_r}{|R_Q|}(1,\ 1) + \left(1 - \frac{n_r}{|R_Q|}\right)\left(\prod_{j=1}^{n_{rt_i}} \lambda_{D_j}(T_i)\right) \tag{12}$$

We must now specify how to compute the vectors $\lambda_{D_j}(T_i)$. We have developed different options:

1. First, we will use the likelihood of the term T_i due to the fact that D_j is a relevant document, but taking into account the information already provided by the original query:

$$\lambda_{D_j}(T_i) = \left(\frac{p_Q(d_j|\bar{t}_i)}{p_Q(d_j|t_i)},\ 1\right) \tag{13}$$

   where $p_Q(\cdot) = p(\cdot|Q)$. Therefore, we have

$$\lambda_{D_j}(T_i) = \left(\frac{p(d_j|\bar{t}_i,\ Q)}{p(d_j|t_i,\ Q)},\ 1\right) \tag{14}$$

   This expression considers how the posterior probability of relevance of a document is affected by the addition of a new term to the query. Using equation 3, $p(d_j|t_i,\ Q)$ and $p(d_j|\bar{t}_i,\ Q)$ can be computed by means of

$$\begin{aligned} p(d_j|t_i,\ Q) &= \alpha_j \sum_{T_k \in D_j} tf_{kj} idf_k^2 p(t_k|t_i,\ Q) \\ &= \alpha_j \left(\sum_{T_k \in D_j, k \neq i} tf_{kj} idf_k^2 p(t_k|t_i,\ Q) + tf_{ij} idf_i^2 \right) \\ p(d_j|\bar{t}_i,\ Q) &= \alpha_j \sum_{T_k \in D_j} tf_{kj} idf_k^2 p(t_k|\bar{t}_i,\ Q) \\ &= \alpha_j \sum_{T_k \in D_j, k \neq i} tf_{kj} idf_k^2 p(t_k|\bar{t}_i,\ Q) \end{aligned} \tag{15}$$

   The next target should be to estimate $p(t_k|t_i,\ Q)$ and $p(t_k|\bar{t}_i,\ Q)$ for each term T_k, $k \neq i$, belonging to D_j. These computations require propagation in the network considering both the query Q and the term T_i as evidences. This may be a time-consuming process, because it implies as many propagations as twice the number of positive expansion terms. We will therefore try to find an

[6] Note that for positive expansion terms, $n_{rt_i} > 0$ and $n_{\bar{r}t_i} = 0$.

approximation of these probabilities with the information that we possess. On the one hand, as $\bar{t}_i$ is the most probable state that the term variable T_i can take on, it is reasonable to think that it does not add more information to the information already provided by Q, so we assume $p(t_k|\bar{t}_i, Q) \approx p(t_k|Q)$. On the other hand, because both terms T_k and T_i index the relevant document D_j, we assume that they are positively correlated, although very weakly, given Q. Combining these assumptions by means of an or-gate (Pearl, 1988), we obtain $p(t_k|t_i, Q) \approx \epsilon + (1 - \epsilon)p(t_k|Q)$. This local query expansion method is called *qe-l1*.

2. Another approach to computing the vector $\lambda_{D_j}(T_i)$ assumes that the query plays the role of a fictitious document that has also been observed as relevant. Therefore, in the set of observed components we include both the document D_j and the query Q. The likelihood vector then becomes

$$\lambda_{D_j}(T_i) = \left(\frac{p(d_j, Q|\bar{t}_i)}{p(d_j, Q|t_i)}, 1\right) \tag{16}$$

Note that the value $p(d_j, Q|t_i)$ can be written as

$$p(d_j, Q|t_i) = \frac{p(d_j|t_i, Q)p(t_i|Q)p(Q)}{p(t_i)} \tag{17}$$

and a similar expression is obtained for $p(d_j, Q|\bar{t}_i)$. Therefore, an equivalent formula for the vector $\lambda_{D_j}(T_i)$ is:

$$\lambda_{D_j}(T_i) = \left(\frac{\frac{p(d_j|\bar{t}_i, Q)p(\bar{t}_i|Q)}{p(\bar{t}_i)}}{\frac{p(d_j|t_i, Q)p(t_i|Q)}{p(t_i)}}, 1\right) \tag{18}$$

The values of $p(\bar{t}_i)$ and $p(t_i)$ are known, and the values $p(\bar{t}_i|Q)$ and $p(t_i|Q)$ were calculated when the original query was processed. So, we only need to compute $p(d_j|\bar{t}_i, Q)$ and $p(d_j|t_i, Q)$. Note that these values are the same as those we used in the previous method, *qe-l1,* and are calculated using Equation 15, by computing $p(t_k|t_i, Q)$ and $p(t_k|\bar{t}_i, Q)$. In this case, we use two different approaches to approximate these probabilities: the first, denoted by *qe-l2,* follows the same approach considered for *qe-l1*. The second, denoted by *qe-l3,* approximates these probabilities in a slight different way: $p(t_k|\bar{t}_i, Q) \approx p(t_k|Q)$, and $p(t_k|t_i, Q) \approx \beta + p(t_k|Q)$, and β being a small value. This approximation represents another way of shaping the assumption that the terms T_i and T_k are almost independent given Q.

TABLE 2. Main characteristics of the test collections.

Collection	Documents	Terms	Queries	*3p. Avg.*
Adi	82	828	35	0.36
CACM	3,204	7,562	52	0.34
CISI	1,460	4,985	76	0.17
Cranfield	1,398	3,857	225	0.42
Medlars	1,033	7,170	30	0.63

TABLE 3. *%C 3p. Avg.* values for term reweighting.

Collection	*tr-ins*	*tr-rep*
Adi	20.12	12.25
CACM	29.90	68.55
CISI	21.45	58.99
Cranfield	3.92	34.18
Medlars	1.45	22.66

Experimental Results

With the aim of observing the behavior of our feedback methods in different environments, we carried out several experiments with five well-known standard test collections: Adi, CACM, Cranfield, CISI, and Medlars. Several comments must be made beforehand: the number of documents that the IR system returns to the user is 15 ($|R_Q| = 15$). The performance measure employed is the percentage of change of the average *precision* for three intermediate points of *recall* (0.2, 0.5, and 0.8) with respect to the results obtained after submitting the original queries to the system. This measure will be denoted $\%C\ 3p.\ Avg$. We evaluate the feedback performance by using the *Residual Collection* method (Chang, Cirillo, & Razon, 1971), which removes all the documents that the user has seen in the first relevance judgment step from the collection. Moreover, we use all the queries, even those where no relevant document has been retrieved. Several important preprocessing steps necessary to obtain a vectorial representation of documents and queries, such as indexing and stemming, have been carried out using the facilities provided by the SMART system (Salton & McGill, 1983). The ϵ parameter used in the *qe-l1* and *qe-l2* methods has been set to 0.0075, and the β value in method *qe-l3* has been set to $1/|R_Q|$.

The main characteristics of the five collections (number of documents, terms and queries), as well as the average precision at the three intermediate points of recall, $3p.\ Avg.$, obtained by the BNR model for all the queries, are shown in Table 2.

Table 3 shows the results of term reweighting (*%C 3p. Avg.* values) using the two different approaches to handle positive and neutral query terms: *tr-ins* and *tr-rep.* Negative query terms are always penalized using Equation 6. We can observe that except in the case of Adi, the method *tr-rep,* which rewards positive and neutral query terms, performs considerably better than *tr-ins.* When performing term reweighting, we therefore recommend the *tr-rep* method, penalizing negative query terms.

Table 4 shows the results of the experiments about query expansion. To focus on the impact of added terms, the original query terms will remain unchanged, instantiating all of them to relevant. In all cases, we instantiate the negative expansion terms to nonrelevant and do not take into account neutral expansion terms. For positive expansion terms we test all the proposed methods, naive (*qe-n*), global (*qe-gmle, qe-gbe1,* and *qe-gbe1*) and local (*qe-l1, qe-l2,* and *qe-l3*).

TABLE 4. *%C 3p. Avg.* values for query expansion.

Collection	*qe-gmle*	*qe-gbe1*	*qe-gbe2*	*qe-l1*	*qe-l2*	*qe-l3*	*qe-n*
Adi	41.30	37.62	21.90	18.19	17.23	31.98	11.90
CACM	3.00	48.40	59.45	51.12	51.37	51.73	−84.71
CISI	35.98	35.97	34.77	37.25	35.21	34.81	−61.59
Cranfield	77.20	78.61	67.41	79.76	81.52	65.87	−9.78
Medlars	10.05	7.64	7.03	4.42	2.39	7.14	−87.42

We can observe that query expansion always improves the results (all the percentages of change are positive), except when we use the naive approach, which degrades the performance considerably in four collections. The reason for this behavior of *qe-n* is that in this case the positive expansion terms are instantiated to relevant. Therefore, they are being considered as query terms, thus changing the original sense of the query and confusing the retrieval process. In the light of these results, we will discard this method for subsequent experiments.

When we focus on a single document collection, there are great differences in performance between the six methods considered (e.g., in CACM, the performance varies from 3% for *qe-gmle* to 59.45% for *qe-gbe2*), but these differences are not so important in the average values (the maximum difference between the mean results of the five collections is around 8%, and less than 3% for the medians). We show our preferences using the *qe-gbe1* method (always with negative expansion), which obtains the best mean and median, and exhibits a more robust behavior. On the other hand, *qe-gmle* is more unstable: it obtains the best values for some collections and the worst for others.

We have carried out another more limited series of experiments with query expansion, to confirm our intuition about the important role played by negative expansion in our feedback model: The negative expansion terms in this case will not be added to the query [they will receive a vector (1, 1) instead of a vector (1, 0)]; only positive expansion terms will be considered, using in this case the method *qe-gbe2.* Alternatively, we also consider the case where only negative expansion terms are used and no positive expansion term is added to the query. Table 5 displays the results of these experiments.

We can observe that the performance decreases considerably when we do not use negative expansion (second column in Table 5). However, the use of negative expansion alone (third column) gives good results, similar to when we consider both negative and positive expansion (fourth column). The conclusions are clear: negative query expansion is necessary, positive query expansion alone is not very useful, in some cases it may even be self-defeating. Only in one case, CACM, does the additional use of positive expansion clearly improve the result offered by negative expansion. Therefore, the best choice is to put negative and positive query expansion into practice simultaneously.

TABLE 5. Effects on the *%C 3p. Avg.* values in query expansion when we do not use either negative expansion (ne) or positive expansion (pe).

Collection	*qe-gbe2* with *pe* without *ne*	*qe-gbe2* without *pe* with *ne*	*qe-gbe2* with *pe* with *ne*
Adi	−7.73	31.81	21.90
CACM	12.19	30.09	59.45
CISI	−0.07	34.75	34.77
Cranfield	2.36	65.87	67.41
Medlars	−0.34	6.96	7.03

If we compare term reweighting with query expansion, Tables 3 and 4, we can see that in some cases (for CACM, CISI, and Medlars) term reweighting gives better results than query expansion, whereas sometimes the opposite holds (for Adi and Cranfield).

Table 6 displays the results obtained by combining term reweighting and query expansion, using the two term reweighting methods and the six different techniques for positive query expansion. In all the cases, we use negative query expansion.

These experiments offer a somewhat surprising result: although the term reweighting method *tr-rep* was clearly better than *tr-ins,* this is no longer true when they are combined with query expansion. In fact, *tr-ins* + *qe* is systematically much better than *tr-rep* + *qe* for two collections, Adi and Cranfield, and slightly better for CACM. We believe that the reason for this behavior may be that in these collections, the major part of the improvement achieved is due to query expansion, which dominates term reweighting, as we have already mentioned. However, when the two techniques are combined, the weights assigned by *tr-rep* to the positive or neutral query terms[7] dominate the weights of the positive expansion terms, thus driving the retrieval process towards the wrong place. We conjecture that a more equilibrated reweighting scheme for query terms, combined with query expansion, would produce better results.

If we consider the appropriate term reweighting method for each collection, we can observe that its combination with query expansion outperforms the results obtained by either technique separately. Moreover, when we use *tr-ins,* the best query expansion method is *qe-gbe2,* followed by *qe-gbe1* and *qe-l3.* For *tr-rep,* all the query expansion

[7] Remember that these terms are rewarded considerably when using *tr-rep.*

Table 6. *%C 3p. Avg.* values for combined term reweighting and query expansion.

	qe-gmle		*qe-gbe1*		*qe-gbe2*		*qe-l1*		*qe-l2*		*qe-l3*	
Collection	*tr-ins*	*tr-rep*	*tr-ins*	*tr-rep*	*tr-ins*	*tr-rep*	*tr-ins*	*tr-rep*	*tr-ins*	*tr-rep*	*tr-ins*	*tr-rep*
Adi	104.2	62.81	95.0	62.14	105.0	53.53	82.98	50.68	78.76	47.50	105.77	60.13
CACM	6.33	62.91	67.1	64.46	72.1	63.69	70.58	61.32	69.46	62.63	67.93	63.46
CISI	8.2	69.74	42.9	70.17	39.0	69.66	48.49	65.02	45.76	63.02	42.51	69.71
Cranfield	99.1	57.92	99.9	61.00	107.7	55.68	90.53	68.76	93.46	70.21	101.87	54.75
Medlars	−32.6	24.70	7.9	24.63	12.4	24.42	9.97	29.09	−11.97	23.64	−4.79	24.43

methods perform in a similar way, except *qe-l2,* which is somewhat worse.

We therefore recommend a combination of term reweighting and query expansion. Depending on the term reweighting technique used, its best query expansion technique shall be recommended. For *tr-ins, qe-gbe2,* and *tr-rep,* all with the exception of *qe-l2* could be equally applied.

A general conclusion applicable to term reweighting, query expansion, and their combination, is that the effectiveness of relevance feedback is highly related to the quality of the initial retrieval: the better the retrieval, the worse the results of relevance feedback. This conclusion is drawn from the observation of the high negative correlation between the *3p. Avg.* (see Table 2) and the *%C 3p. Avg.* values, for each collection.

Although it is very difficult to compare feedback methods when the retrieval engines are different, to conclude this section, we shall compare the performance of the Relevance Feedback methods designed for the BNR model with the results shown in (Salton & Buckley, 1990). In this article, the authors examined several Relevance Feedback methods, showing their effectiveness with four of the five collections that we have used in our experimentation (all except Adi).

We compare the best results obtained with query term reweighting and the *qe-gbe2* expansion method from Table 6 with the *Ide* (*dec hi*) and *Probabilistic* (*adjusted revised derivation*) methods.[8]

In Table 7, we show, for each method and each collection, the average precision of the initial run (*3p. Avg.—I.R.*), but computed using the reduced collection, i.e., removing the retrieved documents in that run and evaluating with the remaining documents, and the average precision of the first iteration of feedback (*3p. Avg.—1R.*).

Although the average of precision for the three recall points in three of the four collections is higher with Salton and Buckley's methods than with our techniques, we should mention that the conditions in which the experiments have been carried out are probably different, in that the set of documents retrieved in the first run will not be the same, and therefore, the evaluations are performed with a different set of documents.[9] We also notice how the average precision obtained with the first run in the reduced collection is always better in our methods than theirs. This means that the BNR model performs slightly better. Consequently, rather than looking at the percentage of change, we should look at the value of the average precision of the first iteration. In this case, our methods are more effective in CACM and CISI than theirs, and less in the other two collections. Nevertheless, the relevance feedback of the BRN model behaves in a similar way to these two methods.

Concluding Remarks

In this article, we have introduced a methodology for relevance feedback in the Bayesian Network Retrieval model, based on the idea of translating the user's relevance judgments about the retrieved documents into partial evidences (likelihood ratios) about the relevance of terms, which can be inserted and propagated in the Bayesian network to obtain the new posterior probabilities of relevance of the documents. This methodology has been used to design several query term reweighting and query expansion techniques. Our methods focus on the terms that appear in the retrieved documents and distinguish, on one hand, between query and nonquery terms, and on the other hand, between positive, negative, and neutral terms, depending on whether they only appear in relevant, irrelevant, or both classes of documents. Different kinds of partial evidences are used to weight the different types of terms.

Using the residual collection method, the proposed techniques have been empirically evaluated on five small standard document collections. They have performed well and in a similar way to that obtained with other models (Haines & Croft, 1993; Harman, 1992; Salton & Buckley, 1990).

By way of conclusion, we could say that the technique of query term reweighting in which the positive and neutral query terms are rewarded increases the performance of the feedback. With respect to query expansion, the negative version is a very useful technique in our model by itself. Combining positive and negative expansions, the results are similar to those obtained with only negative query expan-

[8] Broadly speaking, these methods show the best behavior in the experiments with the four common collections.

[9] Moreover, other experimental conditions are also different. For instance, the three selected recall points are 0.25, 0.50 and 0.75 in (Salton & Buckley, 1990).

Table 7. Comparing query term reweighting and *qe-gbe2* method to *Ide* and *Probabilistic* feedback.

	BNR Model	*tr* + *qe-gbe2*	Salton's model	Ide (dec hi)	Probabilistic
Collection	*3p. Avg.—I.R.*	*3p. Avg.—1R*	*3p. Avg.–I.R.*	*3p. Avg.—1R.*	*3p. Avg.—1R.*
CACM	0.1928	0.3318	0.1459	0.2704	0.2289
CISI	0.1315	0.2231	0.1184	0.1742	0.1436
Cranfield	0.1221	0.2536	0.1156	0.3011	0.3108
Medlars	0.4188	0.5211	0.3346	0.6305	0.5972

sion. The performance of the feedback, based on the application at the same time of term reweighting and query expansion, is a good choice, and improves the results obtained by these techniques separately. Finally, when the performance of our relevance feedback method is compared with others, we can conclude that ours maintains a very good level.

Future work will center on developing new relevance feedback methods for the BNR model, based on the underlying concept of partial evidences, in an attempt to improve the performance obtained with the methods introduced here. This implies designing new or more accurate ways of computing the appropriate likelihood vectors. On the other hand, the query expansion techniques considered in this article include all the positive and negative expansion terms in the new query. We also plan to use a term selection process to determine the terms that will be effectively used to expand the original query, according to certain criteria.

Finally, we plan to test our methods with larger document collections, such as TREC. Several comments need to be made about this task: in terms of propagation, if we see that running a query in the largest collection where we have tested our model only takes a few tenths of a second, we could predict that using a TREC collection would increase this time by a few seconds, a time that would still be acceptable. If this time were too long, there are several ways of reducing the retrieval time. First, the topology of the term subnetwork could be changed, with a smaller polytree connecting only those most important terms. The rest of the terms would be isolated among them. As the size of the polytree is smaller, propagation would be much faster. Alternatively, we could have several smaller Bayesian networks comprising clusters of related terms (in this way, we also reduce the learning time). Propagation will only occur in those networks where there is an evidence, a very fast process. Second, we could apply some of the approximate algorithms that we are implementing based on Pearl's propagation. These algorithms avoid propagation in those areas of the polytree where the belief in the terms will not change, saving a great amount of time without reducing the performance of the propagation.

As the TREC reader can appreciate, there are various ways of modifying the original model if the retrieval and feedback speed were not fast enough for this kind of collection, while maintaining the base of a Bayesian network and all the ideas about the model itself and the feedback methods presented in this article.

Acknowledgment

This work has been supported by the Spanish Ministerio de Ciencia y Tecnología (MCYT) under Project TIC2000-1351.

References

de Campos, L.M., Fernández, J.M., & Huete, J.F. (1998). Query expansion in information retrieval systems using a bayesian network-based thesaurus. In Proceedings of the 14th uncertainty in artificial intelligence conference, Madison, July 1998 (pp. 53–60).

de Campos, L.M., Fernández-Luna, J.M., & Huete, J.F. (2000). Building bayesian network-based information retrieval systems. In 11th International workshop on database and expert systems applications: 2nd workshop on logical and uncertainty models for information systems (LUMIS), London, September 2000 (pp. 543–552).

de Campos, L.M., Fernández-Luna, J.M., & Huete, J.F. (2002a). The bayesian network retrieval model: Foundations and performance. Submitted to International Journal of Approximate Reasoning.

de Campos, L.M., Fernández-Luna, J.M., & Huete, J.F. (2002b). A layered bayesian network model for document retrieval. Lecture Notes Comput Science, 2291, 169–182.

Cestnik, B. (1990). Estimating probabilities: A crucial task in Machine Learning. In Proceedings of European Conference on Artificial Intelligence (ECAI94), (pp. 147–149).

Chang, Y.K., Cirillo, C., & Razon, J. (1971). Evaluation of feedback retrieval using modified freezing, residual collection and test and control groups. In G. Salton (Ed.), The smart retrieval systems—experiments in automatic document processing (pp. 355–370). Englewood Cliffs, NJ: Prentice Hall, Inc.

Fernández-Luna, J.M. (2001). Modelos de recuperación de información basados en redes de creencia. PhD thesis, E.T.S. Ingeniería Informática. Universidad de Granada.

Fung, R., & Favero, B.D. (1995). Applying bayesian networks to information retrieval. Communications of the ACM, 38(2), 42–57.

Ghazfan, D., Indrawan, M., & Srinivasan, B. (1996). Toward meaningful bayesian networks for information retrieval systems. In Proceedings of the IPMU'96 conference, Granada, July 1996 (pp. 841–846).

Haines, D., & Croft, W.B. (1993). Relevance feedback and inference networks. In Proceedings of the 16th ACM–SIGIR conference on research and development in information retrieval, Pittsburgh (pp. 2–11).

Harman, D. (1992). Relevance feedback revisited. In Proceedings of the 16th ACM–SIGIR conference on research and development in information retrieval, Copenhagen (pp. 1–10).

Jensen, F.V. (1996). An introduction to Bayesian networks. London: University College London Press.

Jones, K.S., Walker, S., & Robertson, S.E. (2000a). A probabilistic model of information retrieval: Development and comparative experiments part 1. Information Processing and Management, 36, 779–808.

Jones, K.S., Walker, S., & Robertson, S.E. (2000b). A probabilistic model of information retrieval: Development and comparative experiments part 2. Information Processing and Management, 36, 809–840.

Maron, M.E., & Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. Journal of the Association for Computer Machinery, 7, 216–244.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: Networks of plausible inference. San Mateo, CA: Morgan and Kaufmann.

Ribeiro-Neto, B.A., & Muntz, R.R. (1996). A belief network model for ir. In H. Frei, D. Harman, P. Schäble, & R. Wilkinson (Eds.), 19th ACM–SIGIR conference (pp. 253–260). ACM.

Robertson, S.E., & Jones, K.S. (1976). Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3), 129–146.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41, 288–297.

Silva, I.R. (2000). Bayesian networks for information retrieval systems. PhD thesis, Universidad Federal de Minas Gerais.

Silva, I., Ribeiro-Neto, B., Calado, P., Moura, E., & Ziviani, N. (2000). Link-based and content-based evidential information in a belief network model. In Proceedings of the 23th international ACM–SIGIR conference on research and development in information retrieval, Athens, July 2000 (pp. 96–103). ACM.

Spink, A., & Losee, R.M. (1996). Feedback in information retrieval. Annual Review of Information Science and Technology, 31, 33–78.

Turtle, H.R. (1990). Inference networks for document retrieval. PhD thesis, University of Massachusetts.

Turtle, H.R., & Croft, W.B. (1990). Inference networks for document retrieval. In Proceedings of 13th international ACM–SIGIR conference on research and development in information retrieval proceedings (pp. 1–24).

Turtle, H.R., & Croft, W.B. (1991). Efficient probabilistic inference for text retrieval. In Proceedings of the RIA0'91 conference (pp. 644–661).