Relevance Feedback in the Bayesian Network Retrieval Model: An Approach Based on Term Instantiation

Luis M. de Campos[1], Juan M. Fernández-Luna[2], and Juan F. Huete[1]

[1] Dpto. de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática.
Universidad de Granada, 18071 – Granada, Spain.
{lci,jhg}@decsai.ugr.es
[2] Departamento de Informática, Escuela Politécnica Superior.
Universidad de Jaén, 23071 – Jaén, Spain.
jmfluna@ujaen.es

Abstract. Relevance feedback has been proven to be a very effective query modification technique that the user, by providing her/his relevance judgments to the Information Retrieval System, can use to retrieve more relevant documents. In this paper we are going to introduce a relevance feedback method for the *Bayesian Network Retrieval Model*, founded on propagating partial evidences in the underlying Bayesian network. We explain the theoretical frame in which our method is based on and report the results of a detailed set of experiments over the standard test collections Adi, CACM, CISI, Cranfield and Medlars.

1 Introduction

When a user formulates a query to an *Information Retrieval System* (IRS), this software attempts to find those documents that closely match the representation of the user's information need. The posterior examination of the ranking offered by the IRS may show that a few amount of documents are relevant according to the user's judgments. This fact could be due, among other factors, to the user's incapacity of expressing his information need by means of an exact query. In that moment, the user could apply several of the *query modification* techniques that have been developed in the field of *Information Retrieval* (IR) to improve the query, and therefore, obtain those documents that fulfill the user's relevance criteria.

Among the query modification techniques, we are going to highlight the *Relevance Feedback* [12], that generates a new query starting from the *relevance judgments* (i.e., which retrieved documents are relevant and which are not relevant to the initial query) provided by the user, when she/he inspects the ranking of documents returned by the IRS for the original query. The IRS updates that original query modifying the importance of the terms it contains (*term reweighting*), and adding new terms which are considered useful to retrieve more relevant documents (*query expansion*). This method has been successfully applied in a

F. Hoffmann et al. (Eds.): IDA 2001, LNCS 2189, pp. 13–23, 2001.

great variety of IR models [13], among them, the Croft and Turtle's *Inference Network* model [14], based on *Bayesian Networks*.

Founded on probabilistic methods, Bayesian networks [9] have been proven to be a good model to manage uncertainty, even in the IR environment, in which they have been successfully applied as an extension of the probabilistic IR model, because they offer important advantages to deal with the intrinsic uncertainty with which IR is pervaded [5,14]. Also based on these probabilistic tools, the Bayesian Network Retrieval model (BNR) was introduced in [2] as an alternative to the existing methods based on Bayesian networks [7,10,14].

In this paper we want to explain a proposal for relevance feedback within the BNR model, showing the theoretical frame in which it is based on and the experimental results obtained when it has been applied to several well known standard test collections.

To carry out this task, this paper is organized as follows: Section 2 introduces the BNR model, showing its most important features. Section 3 deals with a theoretical overview of our proposals of relevance feedback for the BNR model. Later, in Section 4, we will explain how we have put in practice some of the ideas developed in the previous section. Section 5 shows the experimental results obtained with CACM and four additional collections. Finally, Section 6 discusses about the conclusions of this work.

2 Description of the Bayesian Network Retrieval Model

A Bayesian network is a *Directed Acyclic Graph* (DAG), where the nodes represent the variables from the problem we want to solve. In that kind of graph, the knowledge is represented in two ways [9]: *(a)* Qualitatively, showing the (in)dependencies between the variables, and *(b)* Quantitatively, by means of conditional probability distributions which shape the relationships. In IR problems we can distinguish between two different sets of variables (nodes in the graph): The set of terms in the glossary from a given collection and the set of documents, grouped in the *term* and *document* subnetworks, respectively. Each term, T_i, consists on a binary random variable taking values in the set $\{\bar{t}_i, t_i\}$, where $\bar{t}_i$ stands for 'the term T_i is not relevant', and t_i represents 'the term T_i is relevant'. Similarly, a variable referring to a document D_j has its domain in the set $\{\bar{d}_j, d_j\}$, where in this case, $\bar{d}_j$ and d_j respectively mean 'the document D_j is not relevant for a given query', and 'the document D_j is relevant for a given query'.

Focusing on the structure of the network, the following guidelines have been considered to determine the topology of the graph [2]:

- For each term that has been used to index a document, there is a link between the node representing that keyword and the node associated with the document it belongs to.
- The relationships between documents only occur through the terms included in these documents.

• Documents are conditionally independent given the terms that they contain. Thus, if we know the relevance (or irrelevance) values for all the terms indexing document D_i then our belief about the relevance of D_i is not affected by knowing that another document D_j is relevant or irrelevant.

These assumptions have some effects on the structure of the network: On one hand, links joining terms and documents must be directed from term nodes to document nodes in the graph and, on the other hand, there are not links between document nodes.

With regard to the relationships among term nodes, the model allows terms being dependent among each other. In order to capture the relationships between terms in the collection, a Bayesian network learning algorithm is used to construct the term subnetwork. Taking into account considerations of efficiency in the learning and inference stages, some restrictions on the structure of the learned graph have been imposed. Particularly, 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 [3].

Once we know the structure of the graph, the final step to completely specify a Bayesian network is to estimate the probability distributions stored in each node. Three different cases have to be considered:

• Term nodes having no parent: In this case we store marginal distributions, estimated as follows: $p(t_i) = \frac{1}{M}$ and $p(\bar{t}_i) = \frac{M-1}{M}$, M being the number of terms in a given collection.

• Term nodes with parents: For each node, we need to store a set of conditional probability distributions, one for each possible configuration of the parent set. These distributions are simply estimated using frequencies of cooccurrence of terms in the documents of the collection.

• Document nodes: In this case, the estimation of the conditional probabilities is more problematic because of the huge number of parents that a document node has. For example, if a document has been indexed with 30 terms, we need to estimate and store 2^{30} (approx. 1.07×10^9) probabilities. Therefore, instead of explicitly computing these probabilities, the BNR model uses a *probability function*, which returns a conditional probability value when it is called during the propagation stage, each time that a conditional probability is required. In this paper, we are going to use the following probability function, based on the cosine measure [12]: for a given configuration $\pi(D_j)$ of the set of terms in document D_j, the probability of relevance of document D_j is

$$p(d_j \mid \pi(D_j)) = \alpha_j \sum_{T_i \in D_j, T_i = t_i} \mathrm{tf}_{ji} \mathrm{idf}_i^2 \tag{1}$$

where tf_{ji} is the frequency of the term T_i in the document D_j, idf_i is its inverse document frequency and α_j is a normalizing constant computed as $1/\sqrt{\sum_{T_i \in D_j} \mathrm{tf}_{ji}^2 \mathrm{idf}_i^2}$.

Once the entire network is completely built, and given a query submitted to the system, the retrieval process starts by placing the evidences, i.e., the terms belonging to the query, in the term subnetwork by setting their states to *'the*

term is relevant'. The propagation process is run, obtaining for each document its probability of relevance given that the terms in the query are also relevant. Then, the documents are sorted by their posterior probability to carry out the performance evaluation process.

Taking into account the size of the Bayesian networks used by the BNR model, even for small document collections, general purpose inference algorithms can not be applied due to efficiency considerations. To solve this problem, the propagation process in the whole network is substituted by an inference process in two steps, but ensuring that the results are the same that the ones obtained using exact propagation in the entire network [6]: First, we propagate the query just only in the term subnetwork by means of the Pearl's Exact Propagation Algorithm in Polytrees [9], and compute, for each term, its probability of being relevant given the query submitted to the IRS, $p(t_i \mid Q)$. In the second step, we compute the probability that each document is relevant given the query by means of the evaluation of the following formula:

$$p(d_j \mid Q) = \alpha_j \sum_{T_i \in D_j} \mathrm{tf}_{ji} \mathrm{idf}_i^2 \; p(t_i \mid Q) \tag{2}$$

For more details about the BNR model and its comparison with other retrieval engines also based on Bayesian networks, please refer to [2].

3 Description of the Methodology for Relevance Feedback

In the BNR model, the proposed methodology for relevance feedback is based on the following idea, centered basically in the way in which new evidential information is entered in the Bayesian network: By evaluating a set of retrieved documents obtained as a consequence of running a query, the user obtains new pieces of evidence for some of the variables in the network, that may help to discriminate which documents are relevant to our information need.

For example, whenever a term T_i indexes a document judged relevant by the user, perhaps we should increase the belief supporting the assert "term T_i is relevant"; we speak about *partial evidence* because, by using only document's judgments of relevance we can not be completely certain about the truth of the previous assertion. Nevertheless, this information must be combined with the previous evidence supported by the original query, in order to formulate a new query to be processed by the I.R. system.

Suppose that, after analyzing the retrieved documents, we have a strong belief supporting the assert "term T_i is relevant", which is quantified, e.g., as being ten times more than the belief supporting the assert "term T_i is not relevant". This information can be considered as a ratio of likelihood values, i.e., $P(Obs|\bar{t}_i) : P(Obs|t_i) = 1 : 10$, where Obs represents the observation. In order to include this information in the IRS we can use the concept of *partial evidence* [9]: If for a node X (term or document node) we find a new piece of evidence supporting its relevance or irrelevance, the node will receive a message[1] denoted by $\lambda(X)$,

[1] sent by a dummy node, an imaginary child for node X.

encoding a pair of likelihood values, i.e., $\lambda(X) = (P(Obs|\bar{x}), P(Obs|x))$. This message, in a normalized form, will be combined with the whole information that node X obtains from its parents and children in the network.
Remark: The use of λ 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 vector: for instantiated nodes (we know for sure their relevance values) we use $\lambda(X) = (0, 1)$ to indicate that X is relevant and $\lambda(X) = (1, 0)$ indicating that X is not relevant; for uninstantiated nodes that do not receive any evidence, we use the vector $\lambda(X) = (1, 1)$.

The particular implementation of these ideas gives rise to two different mechanisms for relevance feedback in the BNR model, which basically differ in the way that new evidences are included in the model: Document-based and term-based relevance feedback. In this paper we are going to focus in term-based relevance, although we briefly describe the document-based approach too.

Document-based relevance feedback

This method intends to capture the idea that, by evaluating the retrieved documents, the set of evidences obtained by the user are focused on document nodes and not on the particular terms indexing these documents. In order to include these new evidences in the IRS, it would be sufficient to instantiate each judged document as relevant or non-relevant. Therefore, the new query would be $Q_1 = (Q, d_1, d_2, \ldots, d_k, \bar{d}_{k+1}, \ldots, \bar{d}_{|R|})$, where $|R|$ is the number of documents that have been judged by the user.

In the propagation process, and as a consequence of document instantiation, it could be considered that each evaluated document sends a message to its parents (the set of terms indexing the document) encoding that it has been judged as relevant or non-relevant by the user. However, due to the huge number of parents that documents have, exact computation is problematic and we use approximate methods (see [4,6] for details).

Term-based relevance feedback

In this case we are considering that, by evaluating the retrieved documents, the user can obtain new evidences for term nodes exclusively, i.e., the relevance judgments about the retrieved documents translates into partial evidences about the relevance of the terms indexing these documents.

Term-based relevance feedback needs to classify the terms indexing the observed documents in three groups: those terms that only occur in relevant documents (*positive terms*), those that only occur in non-relevant documents (*negative terms*), and those that occur in both types of documents (*neutral terms*). In the next section we discuss term-based relevance feedback in detail, and distinguish between query term reweighting and query expansion. i.e., whether the terms belong to the initial query or not.

4 Term-Based Relevance Feedback in Detail

We are going to introduce some notation that will be used in this section: n_r and $n_{\bar{r}}$ represent the number of retrieved relevant and non-relevant documents,

respectively; n_t and $n_{\bar{t}}$ denote the number of retrieved documents where the term T has been observed and not observed, respectively; n_{rt} is the number of retrieved relevant documents in which the term T has been seen; $n_{\bar{r}t}$ is the number of retrieved non-relevant documents including term T; $n_{r\bar{t}}$ is the number of retrieved relevant documents that do not include term T; $n_{\bar{r}\bar{t}}$ is the number of retrieved non-relevant documents that do not include term T.

4.1 Query Term Reweighting

Terms, T_q, belonging to the initial query were instantiated as relevant. Therefore, after judging the retrieved documents we can still consider them as relevant or, otherwise, we need to decrease the belief supporting the relevance of the terms, i.e., penalize them. Some preliminary experiments have shown that a much better performance is obtained if we only penalize those terms that exclusively occur in non-relevant documents (*negative query terms* $(-qt)$). Nevertheless, since they belong to the original query, we still will assign some belief supporting their relevance. Therefore, for those terms considered completely relevant, i.e., they appear, at least, in one relevant document (*positive query terms* $(+qt)$, and *neutral query terms* $(= qt)$) we use the λ message $\lambda(T_q) = (0, 1)$ and for penalized terms we use a λ message such as $\lambda(T_q) = (\gamma_t, 1)$, with $0 < \gamma_t < 1$. We have tried two different approaches for determining the value γ_t: (i) Use a fixed γ value for all the negative query terms $(\lambda_\gamma(T_q) = (\gamma, 1))$, and (ii) consider a γ_t value sensible to the number of non-relevant documents in which the term occurs: the higher this number is, the higher γ_t will be. The following expression satisfies this requirement, assessing γ_t values verifying $0.5 \leq \gamma_t < 1$:

$$\lambda_{tr}(T_q) = (1 - \frac{1}{n_{\bar{r}t} + 1}, 1) \tag{3}$$

4.2 Query Expansion

Now, we want to determine the impact of the addition of new terms, T_e, to the original query. We call these terms *expansion terms*, which may be positive $(+et)$, negative $(-et)$ or neutral $(= et)$.

• The negative expansion terms will be directly instantiated to non-relevant, i.e., $\lambda(T_e) = (1, 0)$, because we consider that they are not useful at all to retrieve relevant documents.

• With regard to neutral expansion terms, we propose to use the λ vector $\lambda(T_e) = (1, 1)$ (which is equivalent to not considering them). We have also experimented with other alternatives, taking into account the number of documents relevant and not relevant where each term appears, but we obtained worse results.

• For positive expansion terms, the first approach would be to instantiate all of them by using $\lambda(T_e) = (0, 1)$, as normal query terms. Experimental results demonstrate that this is not a good choice, because they are being treated as query terms, thus changing the original sense of the query. The second approach

is based on the use of a contingency table (relating relevance of the documents with appearance of the term) in order to measure the evidence supporting the relevance or non relevance for a given term. We can see the probability of obtaining a relevant document when we consider the term T_e being relevant, i.e., $p(r|t_e)$ (where r stands for relevant document), as an evidence favouring the relevance of T_e. Analogously, the evidence supporting the non-relevance of T_e could be measured by the probability of obtaining a relevant document given that the term is non relevant, i.e., $p(r|\bar{t}_e)$. These two values constitute the λ vector for each positive expansion term:

$$\lambda(T_e) = (p(r|\bar{t}_e), p(r|t_e)) \text{ or equivalently } \lambda(T_e) = (\frac{p(r|\bar{t}_e)}{p(r|t_e)}, 1) \tag{4}$$

Now, different ways to estimate these probabilities give rise to different expansion term weighting methods:

- *qe1*: Using a maximum likelihood estimator

$$p(r|t) = \frac{n_{rt}}{n_t} \quad \text{and} \quad p(r|\bar{t}) = \frac{n_{r\bar{t}}}{n_{\bar{t}}} \tag{5}$$

- Using a Bayesian estimator

$$p(r|t) = \frac{n_{rt} + s_t \frac{n_r}{|R|}}{n_t + s_t} \quad \text{and} \quad p(r|\bar{t}) = \frac{n_{r\bar{t}} + s_{\bar{t}} \frac{n_r}{|R|}}{n_{\bar{t}} + s_{\bar{t}}} \tag{6}$$

where the parameters s_t and $s_{\bar{t}}$ represent the equivalent sample size. The reason to propose this method is that the quality of the maximum likelihood estimation may be low, because we are dealing with few data (the $|R|$ retrieved documents).

Now, we need some alternatives to fix these parameters s_t and $s_{\bar{t}}$. The first one is obtained by setting them to a given fixed value, configuring the *qe2* expansion term weighting method. Another option is to relate s_t and $s_{\bar{t}}$ with the amount of data available to estimate $p(r|t)$ and $p(r|\bar{t})$, respectively (less data implies less reliability and therefore a greater equivalent sample size). So, the *qe3* method uses

$$s_t = |R| - n_t = n_{\bar{t}} \text{ and } s_{\bar{t}} = |R| - n_{\bar{t}} = n_t \tag{7}$$

Finally, another alternative for computing s_t and $s_{\bar{t}}$ can be obtained by taking into account the total number of documents in which a term 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 is in three documents, and these three documents have been retrieved, s_t should be 0, because we are using all the possible data to estimate the probability. The opposite case is when the term indexes a great amount of documents, e.g. one hundred documents, and only one is retrieved: s_t should be high. This behaviour can be obtained by computing the ratio between the frequency of the term in the collection and n_t, in the following way (N is the number of documents in the collection):

$$s_t = \frac{\log(\text{tf}_T + 1)}{\log(n_t + 1)} + 1 \text{ and } s_{\bar{t}} = \frac{\log(N - \text{tf}_T + 1)}{\log(n_{\bar{t}} + 1)} + 1 \tag{8}$$

We use logarithms because in certain cases the values computed for the parameters could be extremely high and they have to be smoothed. We denote this last approach as *qe4*.

5 Experimental Results

To test our feedback methods, we carried out several, more detailed, experiments with the CACM collection and some other experiments with four additional collections: Adi, Cranfield, CISI and Medlars, with the aim of observing the behaviour of our feedback methods in different environments.

Several remarks have to be done before starting: The number of documents that the IRS gives back to the user is fifteen ($|R| = 15$). The performance measure employed is the percentage of change of the average *precision* for the three intermediate points of *recall* (0.2, 0.5 and 0.8) with respect to the results obtained after submitting to the system the original queries. This value is presented in the last column of all the tables below, where the intermediate columns display the method used to compute the λ values for each type of query or expansion term. Moreover, we evaluate the feedback performance by using the *Residual Collection* method [1], which removes from the collection all the documents that the user has seen in the first relevance judgment step, and we take into account those queries in which no relevant document has been retrieved.

Table 1 shows the results of term reweighting using CACM. The first three experiments try to show how a low, a medium and a high value for the γ parameter in the vector λ_γ for negative query terms, could affect the feedback performance. As we can observe in this table, the higher γ is, the better the results are, although the increase is very low. We run a fourth experiment in which we use the vector λ_{tr}, showing that its performance is very similar to those obtained in the previous experiments, with the main advantage that we do not have to tune the γ parameter.

Exp	+qt	-qt	=qt	%C
1	$(0, 1)$	$(0.2, 1)$	$(0, 1)$	28.23
2	$(0, 1)$	$(0.5, 1)$	$(0, 1)$	29.83
3	$(0, 1)$	$(0.8, 1)$	$(0, 1)$	30.05
4	$(0, 1)$	λ_{tr}	$(0, 1)$	29.90

Table 1. Term reweighting results with CACM.

Table 2 shows the results of the experiments about query expansion with CACM. In order to focus on the impact of added terms, the original query will remain unchanged, instantiating all its terms to relevant. The first two experiments show that expanding the original query with all the positive expansion terms as being original terms is not a good idea, because these new query terms impair the original query. The third experiment applies the $qe1$ weighting approach, and the result has been considerably improved because each positive expansion term is added with a λ vector different from $(0, 1)$. The $qe1$ weighting approach is still very poor, but its modifications by means of the Bayesian estimation (the fourth and fifth experiments, using $qe2$) give a good qualitative jump. The next two experiments show the performance of the $qe3$ and $qe4$

Exp	qt	-et	=et	+et	%C
1	$(0,1)$	$(1,1)$	$(1,1)$	$(0,1)$	-69.42
2	$(0,1)$	$(1,0)$	$(1,1)$	$(0,1)$	-59.73
3	$(0,1)$	$(1,0)$	$(1,1)$	$qe1$	3.0
4	$(0,1)$	$(1,0)$	$(1,1)$	$qe2 : s_t, s_{\bar{t}} = 5$	46.36
5	$(0,1)$	$(1,0)$	$(1,1)$	$qe2 : s_t, s_{\bar{t}} = n_r$	48.40
6	$(0,1)$	$(1,0)$	$(1,1)$	$qe3$	38.76
7	$(0,1)$	$(1,0)$	$(1,1)$	$qe4$	59.45
8	$(0,1)$	$(1,1)$	$(1,1)$	$qe4$	12.19

Table 2. Query expansion results with CACM.

weighting approaches. Observe that *qe4* exhibits the best percentage of change of all the experiments. The last experiment tries to reinforce the idea of how the negative expansion plays an important role in the query expansion of our feedback model: We instantiate the negative expansion terms to $(1,1)$, i.e. these terms are not added to the query, adding only the positive expansion terms using the *qe4* approach. In this case the performance decreases considerably.

Table 3 displays the results obtained (with CACM) by combining term reweighting and query expansion. In these experiments, we have used vector λ_{tr} for negative query term, positive and neutral query terms use vector $(0,1)$, negative expansion terms use vector $(1,0)$, and we compare the different weighting techniques for positive expansion terms. It is clear that the combination of both techniques is very convenient to improve the performance of our relevance feedback method. The best weighting technique for positive expansion terms is *qe4*, whose performance is clearly superior to the rest.

Exp	+et	%C
1	$qe2 : s_t, s_{\bar{t}} = 5$	65.25
2	$qe2 : s_t, s_{\bar{t}} = n_r$	67.11
3	$qe3$	58.30
4	$qe4$	72.08

Table 3. Term reweighting and query expansion experiments with CACM.

The results of the experiments obtained with Adi, Cranfield, CISI and Medlars are displayed in Table 4. The main characteristics of these collections (number of documents, terms and queries), as well as the average precision at the three intermediate points of recall obtained by the BNR model for all the queries are also shown in this table. All the experiments have been carried out using term reweighting and query expansion, and the fixed parameters are the same used in Table 3. Adi behaves more or less as CACM does (obtaining the best performance with *qe4*). Medlars offers the worst percentages of change of the five collections. The reason could be that the retrieval rate in this collection is good. Again, *qe4* is the best weighting method. Cranfield shows an excellent behavior,

Col.	Doc.	Terms	Quer.	3p Avg.	$qe1$	$qe2 : s_t, s_{\bar{t}} = 5$	$qe2 : s_t, s_{\bar{t}} = n_r$	$qe3$	$qe4$
Adi	82	828	35	0.36	104.2	71.7	95.0	68.6	105.0
CACM	3204	7562	52	0.34	6.33	65.3	67.1	58.3	72.1
CISI	1460	4985	76	0.17	8.2	41.0	42.9	42.7	39.0
Cran.	1398	3857	225	0.42	99.1	98.1	99.9	94.6	107.7
Med	1033	7170	30	0.63	−32.6	10.1	7.9	8.4	12.4

Table 4. Main characteristics of the used collections. Term reweighting and query expansion experiments.

even using *qe1*, with the same pattern as CACM. Finally, CISI exhibits a quite uniform behaviour across the different weighting methods.

6 Concluding Remarks

In this paper we have introduced a relevance feedback method for the Bayesian Network Retrieval model based on partial evidences. We have presented the theoretical frame over it is based on, and empirically shown how this method has a robust behaviour with four of the five standard test collections that we have used. The future works will be centered in the development of new relevance feedback methods for the BNR model based on the underlying concept of partial evidences, trying to improve the performance obtained with the methods introduced here.

Acknowledgments: This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Project TIC2000-1351.

References

1. Y. K. Chang, C. Cirillo, and J. Razon. *Evaluation of feedback retrieval using modified freezing, residual collection and test and control groups*, pages 355–370. Prentice Hall, Inc., Englewood Cliffs, NJ, 1971.
2. L. M. de Campos, J. M. Fernández, and J. F. Huete. Building bayesian network-based information retrieval systems. In 11^{th} *International Workshop on Database and Expert Systems Applications:* 2^{nd} *Workshop on Logical and Uncertainty Models for Information Systems (LUMIS)*, 543–552, 2000.
3. L. M. de Campos, J. M. Fernández, and J. F. Huete. Query expansion in information retrieval systems using a bayesian network-based thesaurus. In *Proceeding of the* 14^{th} *Uncertainty in Artificial Intelligence Conference*, 53–60, 1998.
4. L. M. de Campos, J. M. Fernández, and J. F. Huete. Document instantiation for relevance feedback in the Bayesian network retrieval model. Submitted to the ACM-SIGIR'01 Workshop on Mathematical and Formal Methods in IR.
5. R. Fung and B. D. Favero. Applying bayesian networks to information retrieval. *Communications of the ACM*, 38(2):42–57, 1995.
6. J. M. Fernández. Modelos de Recuperación de Información Basados en Redes de Creencia (in Spanish). Ph.D. Thesis, Universidad de Granada, 2001.

7. D. Ghazfan, M. Indrawan, and B. Srinivasan. Towards meaningful bayesian networks for information retrieval systems. In *Proceedings of the IPMU'96 Conference*, 841–846, 1996.
8. D. Harman. Relevance feedback revisited. In *Proceedings of the* 16^{th} *ACM–SIGIR Conference*, 1–10, 1992.
9. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan and Kaufmann, San Mateo, 1988.
10. B. A. Ribeiro-Neto and R. R. Muntz. A belief network model for IR. In H. Frei, D. Harman, P. Schäble, and R. Wilkinson, editors, *Proceedings of the* 19^{th} *ACM–SIGIR Conference*, 253–260, 1996.
11. G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.
12. G. Salton and M. J. McGill. *Introduction to modern Information Retrieval*. McGraw-Hill, Inc., 1983.
13. A. Spink and R. M. Losee. Feedback in information retrieval. *Annual Review of Information Science and Technology*, 31:33–78, 1996.
14. H. R. Turtle and W. B. Croft. Inference networks for document retrieval. In J.-L. Vidick, editor, *Proceedings of the* 13^{th} *ACM–SIGIR Conference*, 1–24, 1990.