Knowledge Discovery in Web-Directories: Finding Term-Relations to Build a Business Ontology

Sandip Debnath[1], Tracy Mullen[3], Arun Upneja[2], and C. Lee Giles[1,3]

[1] Department of Computer Sciences and Engineering
[2] School of Hotel, Restaurant and Recreation Management
[3] School of Information Sciences and Technology,
The Pennsylvania State University, University Park, PA 16802 USA
debnath@cse.psu.edu, tmullen@ist.psu.edu, aupneja@psu.edu, giles@ist.psu.edu

Abstract. The Web continues to grow at a tremendous rate. Search engines find it increasingly difficult to provide useful results. To manage this explosively large number of Web documents, automatic clustering of documents and organising them into domain dependent directories became very popular. In most cases, these directories represent a hierarchical structure of categories and sub-categories for domains and sub-domains. To fill up these directories with instances, individual documents are automatically analysed and placed into them according to their relevance. Though individual documents in these collections may not be ranked efficiently, combinedly they provide an excellent knowledge source for facilitating ontology construction in that domain. In (mainly automatic) ontology construction steps, we need to find and use relevant knowledge for a particular subject or term. News documents provide excellent relevant and up-to-date knowledge source. In this paper, we focus our attention in building business ontologies. To do that we use news documents from business domains to get an up-to-date knowledge about a particular company. To extract this knowledge in the form of important "terms" related to the company, we apply a novel method to find "related terms" given the company name. We show by examples that our technique can be successfully used to find "related terms" in similar cases.

1 Introduction

With the number of documents on the Web in trillions, less time to search for the right document, and inefficiencies or limitations of search engine technologies, Web-directories are an important way of organising Web documents. Examples include Yahoo directories, Google directories or DMOZ directories, MSN directories and other similar (mostly) hierarchical clusters or taxonomies of documents on the Web. Web-directories are nowadays becoming more valuable for several reasons. First of all, novice or first-time users sometimes may not necessarily know what keyword to search with to get documents in certain area of interest. Lack of proper keyword in certain specific domain can hinder the possibility of getting valuable documents. Secondly for even expert users, it is always helpful to filter valuable documents and arrange them in some fashion to save their time. These taxonomies help the users by filtering valuable documents and arranging them in some fashion to save their time.

Learning term-relationships is considered one of the most useful steps in the context of knowledge discovery, construction of knowledge-bases (e.g. domain ontologies) or knowledge management issues. Our main goal is to build business ontologies for major public companies listed in Forbes list. These ontologies will be part of our business knowledge-base, which will be used for analysing textual information (such as corporate news sources, whitepapers or annual reports etc.) for individual companies.

To construct these ontologies for individual companies, we needed up-to-date information, in the form of useful terms e.g. from news articles available in business Web-sites. We can learn useful terms from these sources, which in the later stages, can be incorporated into the ontology with human assistance. We use OWL-Lite for creating the ontology. Although this part is relevant, however in this paper, due to space constraints, we mainly focus on the pre-processing of documents so that we can have all related terms for a company (in this case) extracted from the corresponding news document set.

Though we do not have enough space for the details of the ontology construction process here (we give a block-diagram in Figure 1), however, in this paper, we present the learning model (shown as a dotted rectangle surrounding "Related Term-vector Generator") involved in this work. The model is used to retrieve important terms to be included in the ontology.

Our approach of finding useful terms borrows ideas from query expansion or query term re-weighting. We basically are looking for "related term-vector" (defined later) for a particular company. This is an on-line learning process, where we do not rely on dictionary, thesaurus, or word-net to generate the "related term-vector", which can be thought of as query-expansion. We use news articles to learn the term co-occurrences for the companies and we used company names as query terms. This knowledge is used to build the "related term-vector" and the corresponding ontology. As new news articles are introduced we do the analysis on-the-fly to update the *ontology*.

2 Related Work

Building ontology is a complex process. Though large-scale ontologies exist, they may not be appropriate for specific purpose. According to Noy [17], ontologies should be built for specific purpose or reason. To elaborate on that she advised ontology-builders to follow yet we need to build ontology for a particular domain, such as business domain. Moreover we need to keep it relevant and up-to-date. Building ontologies completely manually is time-consuming, and erroneous. It is also difficult to modify ontologies manually. Automatic ontology building is a hard problem.

Ontology building process can be top-down, bottom-up or middle-out. Uschold et.al. proposed a manual ontology building process, parts of which can be made automatic. According to them the manual process consist of (1) identifying the key concepts and relationships between them, (2) committing to the basic terms such as class, entity, relation etc., (3) choosing a representation language, (4) integrating existing ontologies. Our method of discovering the knowledge by way of finding related terms for a company name falls under the first step. Though discovering relationships between terms is a whole different subject of research and out of scope for a discussion here, we mainly focus here on the prelude, which is finding the related terms. We believe that automating each individual sub-process of the ontology building process is the only way to automatise it as a whole.

Learning term-term relationships has its root in Information Retrieval (IR) research in the context of relevance feedback and is used mainly for query modification. The two main trends in this research is query term re-weighting and query expansion.

Harman [8,9,11,10] examined these two trends in a probabilistic model. There he discussed the question of adding best possible terms [8] with the query. Term re-weighting is an essential part of relevance feedback process, which has been investigated by Salton et. al. [23,12] in addition to their experiments with variations of probabilistic and vector space model [22]. Smeaton and van Rijsbergen [25] investigated query expansion and term re-weighting using term-relationships. The results from these experiments are largely negative. Query expansion via Maximum Spanning Tree shows poor performance for unexpected query. The same happened using Nearest Neighbour approach too. They cited the reason behind this as the difficulty in estimating the probability. We introduced a simple way of estimating the probability for a set of documents. Two words are associated if they co-occur in a sentence or nearby-sentences. In our belief, this is more realistic and reasonable approach and this can be viewed as learning the relationships from a set of documents.

User specific information has been used to expand the query [1]. Personal construct theory has been used in this [13] paper. We analyse the document set to find out the probability of co-occurrence and use that instead of user-preference. A user-centric evaluation of ranking algorithms can be found in [5]. In [15] Ramesh worked with Sen-Tree model to find term-dependencies. In [21] researchers used a logical approach to describe the relationship between a term and a document. They used a *probabilistic argumentation system* and claimed after Rijsbergen that IR systems are a form of uncertain inference. In order to be relevant to a term, a document must imply the term. Our approach can be seen from this angle, that each term must imply the occurrence of the related term with a certain probability. Our idea of term-relationship is somewhat similar to term co-occurrence [19]. In that work Peat and Willett explained the limitations of using term co-occurrence. But they used a more generic calculation of similar terms by three similarity measures, *cosine*, *dice*, and *tanimoto*. They used the whole document to find the term co-occurrence instead of a small region. We believe that the way the similarity measures are formed and the way term co-occurrences are identified are too generic in nature to find any useful result. Instead of completely relying on the document statistics, if we can exploit the usual constructs of natural language sentence formation and use a smaller region of terms rather than the whole document, we can achieve better result.

Other researchers such as Turtle and Croft [26], Fung et. al. [6], Haines and Croft [7], Neto [16], Pearl [18] and Silva [24] also contributed in this area of research. Some of the researchers viewed the document as a sample of the collection of terms. Terms occur randomly in the document with some probability distribution and these distributions are not always known. Even if we do not know the real distribution, document-term-relationship or term-term relationship can play a major role.

3 Our Approach

We introduce a new way to look into term co-occurrences. Natural language documents are not just arbitrary array of words. Words co-occuring in sentences or nearby sentences, relate to each other more closely that in two distant sentences. As described above, the reason behind the failure of term co-occurrences as studied in [19] can possibly be improved. We introduce "Weighted-Sentence" (*WS*) based term co-occurrence.

The system architecture is shown in Figure 1 for a category. The document collection is pre-filtered but not ranked. In taxonomies or Web-directories, candidate documents for a category are already organised using shallow filtering techniques, as explained above. This means that we have got the preliminary document set for a company or category. We use this document set to generate the related-term vectors for the query (category/company name).

For the rest of the discussion, it will be easier to think of the company or category name as the keyword in question. Actually, not only just for the sake of discussion, but sometimes documents are clustered in the same way in reality too. For example we will see that in financial news domain of Yahoo [1], news articles are filtered according to the occurrence of the ticker symbol in those articles. Even if it is not the case, for the sake of generality we can assume that the set of documents are pre-filtered for the keyword in question.

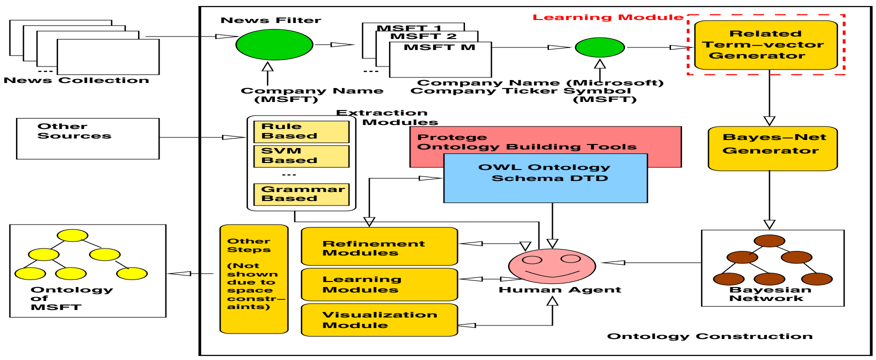

Fig. 1. The architecture of our system describing the ontology building process. We collect up-to-date and useful information from several different sources including financial news documents. News filter filters news for a specific company (e.g. MSFT or Microsoft). We describe the theory behind the "Related Term-vector Generator" in this paper.

Let us assume that in a category C there are in total M documents $D_1, D_2, \ldots, D_M$ pre-filtered for the query w. This constitutes the document collection C. So

$$C = \{D_1, D_2, \ldots D_M\} \tag{1}$$

Each of these documents can be thought of as a set of sentences. Therefore, if document D_d has N_d number of sentences in it, then we can write

$$D_d = \{{S^d}_1, {S^d}_2, \ldots {S^d}_{N_d}\} \text{ where } {S^d}_j : j^{th} \text{ sentence in document } D_d \tag{2}$$

[1] http://finance.yahoo.com

Similarly we can also imagine each individual sentences as a set of words and extend the same notation. Hereafter we removed the superscript of S_i to reduce the notational clumsiness. So if S_i contains P_i number of words in it then

$$S_i = \{{w^i}_1, {w^i}_2, \ldots {w^i}_{P_i}\} \text{ where } {w^i}_k \text{ represents the } k^{th} \text{ word in sentence } S_i \tag{3}$$

3.1 Related-Term Vector

Related-term vector of a query w is a vector of all the words which co-occur with w and are important (ranked higher than a given threshold). Basically it is the set of terms for which the term co-occurrence measure between them and the query is higher than a threshold. The difference between conventional co-occurring terms [19] and our approach is that we consider words in the same sentence or neighbouring sentences (will be explained in section 4.1) as probable candidates for related-term vector. We will come to the implementation part of it, (where this concept will be made clearer) where we took another assumption that no "verb", "preposition" or so-called stop-words are included in the related word-set. So according to this definition, if a sentence does not contain the query keyword w, then the related word-set of w for that sentence is a null-vector.

At this point, we assume that we have a function Ψ which generates all the related-terms of w when it is applied to a sentence S_j containing w. Therefore,

$$\overrightarrow{\Psi}(w, S_i) = \begin{cases} \langle\overrightarrow{{w^r}_1, {w^r}_2, {w^r}_3, \ldots {w^r}_{r_i}}\rangle, & w\mathcal{R}{w^r}_j \; and \, w \in S_i \\ \overrightarrow{\phi}, & otherwise \end{cases} \tag{4}$$

Ψ generates a vector of related-terms of the query w. Considering the fact that function Ψ generates a vector, we can write $\Psi(w, S_i)$ as $\overrightarrow{\Psi}(w, S_i)$. $\mathcal{R}$ implies that w and ${w^r}_j$ are related. r_i is the total number of related words in this sentence S_i.

An example would be useful here. In a sentence *"Microsoft (MSFT) CEO Bill Gates announced today about the upcoming version of windows operating system, codenamed longhorn"* , the words/phrases like "CEO", "Bill Gates", "today", "upcoming", "version", "windows", "longhorn" etc. could be all related to the keyword "Microsoft". Given this sentence and the keyword "Microsoft", the function Ψ will generate the related-term vector which may include the above words depending on the threshold used.

Similar to the above derivation (3) we can see that in case of a whole document D_j and for the whole document collection C in category C,

$$\overrightarrow{\Psi}(w, D_j) = \sum_{k}^{N_j} \overrightarrow{\Psi}(w, S_k) \text{ and } \overrightarrow{\Psi}(w, G) = \sum_{j}^{M} \overrightarrow{\Psi}(w, D_j) \tag{5}$$

where the summation indicates a vector addition.

We describe the algorithm to implement the Ψ function and the related-term vector generating process in subsection 4.1.

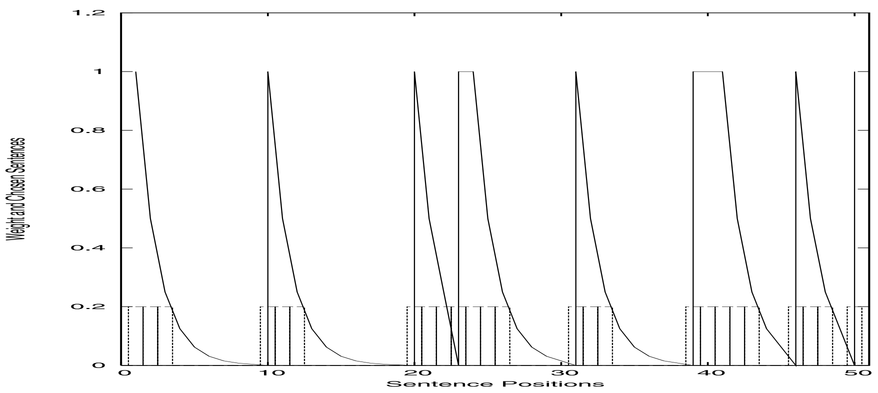

Fig. 2. The weighting function $W(j)$ and the set of sentences included in $N(S_i)$ as shown by the bar graphs. Each bar indicates a sentence. The preliminary sentence set was 1, 10, 20, 23, 24, 31, 39, 40, 41, 46, and 50 and the final extended sentence set is 1-3, 10-12, 20-26, 31-33, 39-43, 46-48, and 50.

4 Theory and Implementation

Here we elaborate our theory behind related-term vector.

4.1 Weighted Sentence Based Related-Term Vector

We introduce a novel concept of finding related-term vectors based on weighted sentences. On-line publishable natural language texts are almost always written in a coherent way. That means once a topic is mentioned, a few consecutive sentences are devoted to describe the topic. We exploit this this editing style to extract related-term vector by using our weighted sentence (WS) based method.

As mentioned earlier, related-term vector basically depends on the concept of term co-occurrences. From equation (4)

$$\overrightarrow{\Psi}(w, S_i) = \begin{cases} \overrightarrow{\langle w^r{}_1, w^r{}_2, w^r{}_3, \ldots w^r{}_{r_i} \rangle}, & w \mathcal{R} w^r{}_j \; and \; w \in N(S_i) \\ \overrightarrow{\phi}, & otherwise \end{cases} \tag{6}$$

Let us proceed step-by-step assembling all the concepts necessary to get the generating function for related-term vector. Let us first modify the concept of "sentences" to an "extended sentence-set". This will also help us define the weighted sentence (*WS*) based method. We introduce a function $N(S_i)$ which takes the position of a sentence S_i and generates the extended sentence-set. First we define the following function to get the weighting factor.

$$W(w, j) = e^{(j-i)log(\tau)} \text{ where } w \in S_i, j \geq i, \tau = \text{ threshold} \tag{7}$$

Here $W(w, j)$ is a weighting factor for all consecutive sentences at position j after the sentence i containing the query w. In our implementation, $\tau = 0.5$. With this definition we define $N(S_i)$ as

$$N(S_i) = \{S_k | k \geq i, W(w,k) \geq \epsilon\} \tag{8}$$

In our implementation $\epsilon = 0.2$. Figure 2 shows the $W(w,j)$ for a sample document where sentences at positions position 1, 10, 20, 23, 24, 31, 39, 40, 41, 46, and 50 contain the query term. In this particular case sentences at positions 1-3, 10-12, 20-26, 31-33, 39-43, 46-48, and 50 will be included in corresponding $N(S_j)$s. This idea is the result of an empirical study about the effect of sentences in document relevance and we showed that this technique can be useful to increase document relevance.

The idea behind is that natural language sentences cluster together based on topic. Term-significance is calculated using a formula similar to [20], where term-term significance was calculated per document basis. We calculate it per collection-basis. The significance of a word w_m co-occurring with w in sentence S_j is

$$\sigma_{w_m} = \frac{ptf_m}{\sqrt{\sum_r ptf_r{}^2}} \times \log \Phi \tag{9}$$

where $ptf_m = (\frac{tf_m}{max_r tf_r})$ and where $\Phi = \frac{N}{n_m}$ where

$$tf_m = ntf^j{}_m \times W(w,j | S_j \in N(S_i)) \text{ (from (8))} \tag{10}$$

where $ntf^j{}_i =$ term-frequency of term w_i in sentence S_j. N is the number of total sentences in the document collection C which is (from (1) and (2))

$$N = \sum_k^M N_k \tag{11}$$

$$n_m = \sum_{S_k \notin N(S_i), w_m \in Nouns(S_k)}^M N_k \tag{12}$$

which in simple term is the number of sentences outside the set $N(S_i)$ where the term w_m appears. $Nouns(S_k)$ is an NLP function which gives the set of all nouns of a sentence, taken as its input.

Now $\mathcal{R}$ represents the relatedness between two terms. It is a defined as a relation between w and w_m which produces the candidacy of w_m to be included in the related-term vector depending on some conditions.

$$\mathcal{R} \equiv \{f : w \rightarrow w_m | \sigma_{w_m} > \zeta, w_m \in Nouns(N(S_i))\} \tag{13}$$

Here $Nouns(N(S_i))$ gives the whole set of nouns from the extended *WS* set of S_i. ζ is a threshold.

5 Evaluation

Experimenting and evaluating the processes responsible for automatic ontology construction is a hard problem. First of all the process consists of several sub-process and each has separate goals. Secondly it is hard to quantify the betterness of an ontology over other ontologies. The reason lies in the basics of ontology construction. As Noy [17] said, ontologies are build for specific purposes. Therefor our business ontology for business document analysis can not be compared with another business ontology, constructed for a different goal. For these reasons we decided to provide the experimental results in the form of the related term-set generated by our technique. In future we would like to concentrate on finding out the use of these terms in the context of relevance or some other purpose and can compare the usefulness and betterness of our approach over others.

Table 1. Details of the dataset. We have 112 companies/categories in total but due to the enormous size of the latex table all of them are not shown. Number of pages taken from individual categories are shown in the third column, followed by related-term sets as found by our method.

Symbol	**Company**	**Number of documents**	**Related terms**
AAPL	Apple	86	**AAPL, Apple, Computer, Music, MSFT, Lehman, Steve Jobs, Cowen, etc.**
DELL	Dell Inc.	169	**DELL, Computer, www.dell.com, Michael Dell, HP, Server, IBM, Storage,**
EBAY	eBay	108	**EBAY, Paypal, Amazon, AspenTech, Andale, Auction, Bid, etc.**
GE	General Electric	238	**GE, Finance, General Electric, Schwarzenegger, Capital, China, Aegon, Medical, etc.**
IBM	IBM	387	**IBM, Sco, Linux, Services, Unix, Lego, Equifax, Lotus, eServer, HP, Tivoli, etc.**
JNJ	Johnson and Jonhson	74	**JNJ, Johnson, Merrill, Centocor, Medtronic, etc.**
JPM	JPMorgan Chase and Co	88	**JPM, JPMorgan, Chase, Risk, Equity, Metropoulas, etc.**
MSFT	Microsoft	492	**MSFT, Microsoft, Security, AOL, Apple, Sco, Caldera, Macintosh, Wi-Fi, etc.**

5.1 Data Set

We crawled Yahoo's financial news page [2] starting from summer 2003 and cached individual news articles as appeared in Yahoo web-pages in respective company ticker symbol. We selected 112 stock symbols for this experiment. We converted the HTML pages into text using a combination of our own extraction algorithms *ContentExtractor* and *FeatureExtractor* [4,3,2].

In short they are based on HTML features and information content blocks. We divided the HTML pages into several different blocks, based table, page or other type of boundaries. These blocks are then analysed for the required feature or based on their similarity over a collection. We got over 95% of F-measure in this part. Due to space

[2] http://finance.yahoo.com

constraints we are not showing the results here. The details of the dataset is shown in Table 1 for 8 companies. Due to space constraints we could not show all 112 categories. The total number of documents we analysed for this paper is 2333.

5.2 Experiment

We implemented our algorithms in Perl on Unix platform. We used Alembic workbench [14] (for the $Noun$ function), which is one of the best natural language processing softwares available. From the Table 1 we can see that our approach can be used to extract very useful terms for all these companies.

6 Conclusion and Future Work

We came up with a novel technique of discovering knowledge from web-directories by finding term-term relations in news article collections available from these web-directories. From our approach and the formula used, we claim that our approach is flexible and it can be applied to any document collection for any query term, if we just replace the company names with the desired query term. In future we would like to create a Bayesian Network from these term-relations which can play a major role in assisting humans to populate the company ontology instances. The formation of Bayesian Network and the proper use of it can also help us quantify the usefulness and to do performance comparison of our approach over others in the context of document relevance.

References

1. Sanjiv K. Bhatia. Selection of search terms based on user profile. In *Proceedings of the ACM/SIGAPP Symposium on Applied computing*, pages 224–233, 1992.
2. Sandip Debnath, Prasenjit Mitra, and C Lee Giles. Automatic extraction of informative blocks from webpages. In *Proceedings of the ACM SAC 2005*, pages 1722–1726, 2005.
3. Sandip Debnath, Prasenjit Mitra, and C Lee Giles. Identifying content blocks from web documents. In *Proceedings of the 15th ISMIS 2005 Conference*, pages 285–293, 2005.
4. Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C Lee Giles. Automatic identification of informative sections from webpages. In *Upcoming journal of IEEE Transactions on Knowledge and Data Engineering*, 2005.
5. Efthimis N. Efthimiadis. A user-centred evaluation of ranking algorithms for interactive query expansion. In *Proceedings of the 16th ACM SIGIR*, pages 146–159, 1993.
6. Robert Fung and Brendan Del Favero. Applying bayesian networks to information retrieval. In *Communications of the ACM*, volume 38(3), pages 42–ff, 1995.
7. David Haines and W. Bruce Croft. Relevance feedback and inference networks. In *Proceedings of the 16th ACM SIGIR*, pages 2–11, 1993.
8. Donna Harman. Towards interactive query expansion. In *Proceedings of the 11th ACM SIGIR*, pages 321–331, 1988.
9. Donna Harman. Ranking algorithms. In *Information Retrieval: Data Structures and Algorithms*, pages 363–392. Englewood Cliffs: Prentice Hall, 1992.
10. Donna Harman. Relevance feedback and other query modification techniques. In *Information Retrieval: Data Structures and Algorithms*, pages 241–263. Englewood Cliffs: Prentice Hall, 1992.

11. Donna Harman. Relevance feedback revisited. In *Proceedings of the 15th ACM SIGIR*, pages 1–10, 1992.
12. Wu Harry and Gerard Salton. A comparison of search term weighting: term relevance vs. inverse document frequency. In *Proceedings of the 4th ACM SIGIR*, pages 30–39, 1981.
13. George A. Kelly. A mathematical approach to psychology. In *B. Maher, Ed. Clinical Psychology and Personality: The Selected Papers of George Kelly*, pages 94–112. John Wiley and Sons, 1969.
14. MITRE. Alembic workbench - http://www.mitre.org/tech/alembic-workbench/.
15. R. Nallapati and J. Allan. Capturing term dependencies using a language model based in sentence tree. In *Proceedings of CIKM 2002*, 2002.
16. Berthier Ribeiro Neto and Richard Muntz. A belief network model for ir. In *Proceedings of the 19th ACM SIGIR*, pages 253–260, 1996.
17. N.F. Noy and C. Hafner. The state of the art in ontology design: A survey and comparative review. In *AI Magazine*, volume 18, pages 53–74, 1997.
18. Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.
19. Helen J. Peat and Peter Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. In *Journal of the American Society for Information Science*, volume 42(5), pages 378–383, 1991.
20. Ulrich Pfeifer, Norbert Fuhr, and Tung Huynh. Searching structured documents with the enhanced retrieval functionality of freewais-sf and sfgate. In *Computer Networks and ISDN Systems*, volume 27(6), pages 1027–1036, 1995.
21. Justin Picard and Rolf Haenni. Modeling information retrieval with probabilistic argumentation systems. In *20th Annual BCS-IRSG Colloquium on IR*, 1998.
22. Gerard Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill, 1968.
23. Gerard Salton and C. Buckley. Improving retrieval performance by relevance feedback. In *Journal of the American Society for Information Science*, volume 41, pages 288–297, 1990.
24. I. R. Silva. Bayesian networks for information retrieval systems. In *PhD thesis, Universidad Federal de Minas Gerais*, 2000.
25. A. Smeaton and C. J. van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. In *Computer Journal*, volume 26(3), pages 239–246, 1983.
26. H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th ACM SIGIR*, pages 1–24, 1990.