Multiple Collection Searching: An approach with two term layers in the Bayesian network retrieval model

Divya V
Assistant Professor, Dept of CSE
Sri Vellappally Natesan College of Engineering
Pallickal, Alappuzha (DT)
divyavijayan1984@gmail.com

***Abstract*—**As hundreds or even thousands of collections are available on the Internet, the IR community must cope with the problem of searching multiple collections. To build a single index for all collections is practically prohibited by its obvious drawback: it is too slow, because searching such a gigantic index takes a long time to complete. Worse, this search may not complete due to network resource limits in case of hundreds of collections. This paper describes how to use Bayesian inference network, a probabilistic approach, to solve the problems in searching multiple collections. An efficient learning method to capture the relationships among terms contained in a given document collection, for improving the retrieval performance, as well as their use for retrieval purposes, is also shown.

Keywords—BNRM, Bayesian inference network, collection selection, information retrieval, document retrieval

1. Introduction

Nowadays, the retrieval of information is becoming more and more important with the widespread use of Internet in our everyday tasks. The field of information retrieval (IR) has been defined by Salton and McGill [1] as the subject concerned with the representation, storage, organization, and accessing of information items. In this paper, we mainly focus our attention on two main IR tasks: selecting appropriate collections representing the information, and the way in which we access information items.

We shall focus our research on the use of uncertain inference models for IR [2]. These models represent an extension of the classical probabilistic model [3], providing a framework for the integration of several sources of evidence. The use of these models is based on the fact that most tasks in this area may be described as uncertain processes [4]. The theoretical justification for these models is based on the probability ranking principle' [5] which states that the best overall retrieval effectiveness will be achieved when documents are ranked in decreasing order of their probability of relevance.

In the last decades, Bayesian networks [6] have become one of the most promising methodologies to manage uncertainty. Bayesian networks combine a qualitative representation of the problem, by means of a graphical representation of the dependences (and also independences) between the variables involved in the problem, with quantitative representation of the uncertainty, using a probabilistic approach. The main advantage of this formalism is that it can be performed efficiently by probabilistic computing.

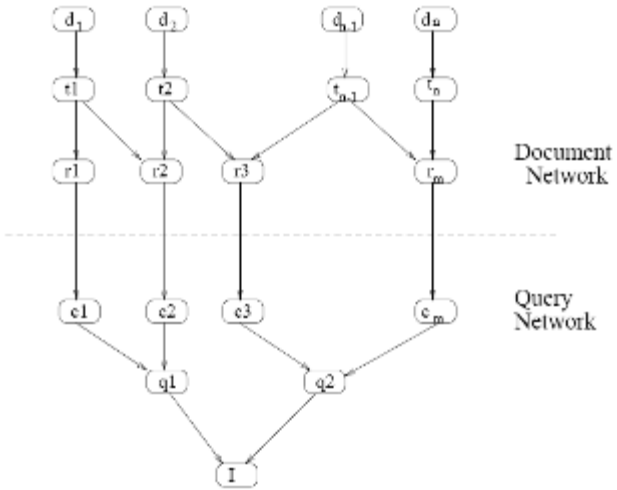
Since the information collection available is getting larger and larger day by day, it is difficult for the users to get the precise and relevant results from a single large collection [7]. One way to overcome this difficulty is to narrow the search of a large index to a portion of the index, which leads to partition of the index. The natural way to partition is for each partition to correspond to one collection. Heuristics are needed to determine which partitions (collections) are most useful.

In this paper, we discuss how to use the Bayesian inference network to solve the problems in searching multiple collection searching. The rest of this paper is organized as follows: Section 2 gives the background. Section 3 describes how to select the useful collections and how to retrieve and rank the documents from selected collections are explained in section4. The system architecture is detailed in section 5. An example is illustrated in section 6 and a brief conclusion is given in section 7.

2. Background

A Bayesian inference network [8] is a directed, acyclic dependency graph (DAG) in which nodes represent propositional variables or constants and edges represent dependence relations between propositions.

The basic document retrieval inference network [8],



shown in Figure 2.1, consists of two component networks: a document network which is built once for a given collection and a query network which is built at query processing time. The document network consists of document nodes (d_i), text representation nodes (t_i), and concept representation nodes (r_i). A document node corresponds to an abstract document, and a text representation node corresponds to a specific text representation of a document. A concept node corresponds to an index unit; it can be a term or a phrase. The link from a *t* node to an *r* node means that the document is "about" the particular concept.

Figure 2.1 A document retrieval inference network

The query network consists of the query concept nodes (c_i), query nodes (q_i) and an information need node (I). A query concept corresponds to a basic unit to construct a query. In the simplest case [9], the query concepts are the same as the representation concepts so that each query concept has exactly one parent (this is the case of the INQUERY). A query node represents an individual query. Information need can be represented by several queries. The weights stored in the I node represent the importance of each query.

The retrieval task is to calculate the belief that the information need is met given that a particular document is observed. Documents are ranked and presented to the user ordered by their belief scores.

3. Collection Selection

The task of collection selection is to identify the collections containing the most documents about the information need. Its major part is collection ranking. After the collections are ranked, how many top collections are presented to users can be determined by users (designating a number before searching) or a clustering algorithm (clustering collections and returning the collections in the top clusters).

Collection ranking can be addressed by a collection retrieval inference network or CORInet for short which is similar to Figure 2.1with only a single difference: d nodes in Figure 2.1 is replaced with C nodes which represent the abstract collections, *t* nodes correspond to a specific representation of collections and *r* nodes correspond to the concepts in the collections. The belief stored in the *r* node should be directly proportional to the number of documents about the information need.

The number of documents about a particular term r_i in the collection C_m can be estimated by:

$$DF(r_i|C_m) = |\{d_j|d_j \in C_m \wedge P(r_i|d_j) > l\}|,$$

Where *l* is a threshold, if $P(r_i|d_j) > l$, the term r_i is assigned to the document d_j.

Although it is possible to get a better threshold *l* by learning from the query sets and collections whose relevant information is available, it is reasonable and convenient to set *l* to the default belief, which is equivalent to the assumption that a term is assigned to a document when it is observed at least once in that document. Then

$$DF(r_i|C_m) = df_{im};$$

where df_{im} is the number of documents that the term t_i is observed in the collection C_m. The biggest benefit of setting *l* as the default belief is that it can be obtained without knowledge about the *tf* and *idf* of each term in each document so that expensive computing and storage requirements are avoided.

Let $P(r_i|C_m)$ denote the belief that estimates the importance of a particular collection (C_m) for a particular term (r_i) based on the number of documents about r_i in C_m.

The query processing in the CORI net is the same as that in the normal document inference network except that the proximity operators are replaced by Boolean AND, because the location information is not stored in the CORI net due to its high storage and computing costs.

Since the CORI net only keeps partial information on the contents of each collection, it has moderate storage requirements and scales with the growing number of available collections [10].

4. Document Retrieval from selected collections

The simple Bayesian network retrieval model involves two different set of variables; i.e document nodes and term nodes. This model is not concerning with term relation ship. If we employ term to term relation ship, then the retrieval performance of this model [13] is usually better than that of the simple one.

4.1. Topology: two- term layers

In this topology we shall include explicit dependence relationships between *Tj* and each term in *Rp* (*Tj*) (the set of those *p* terms most closely related to *Tj*, measured in a certain way). The new graph will use two layers of nodes to represent the term sub network: we duplicate each term node *Tk* in the original layer to obtain another term node *Tk* , thus forming a new term layer, *T'*. The arcs connecting the two layers go from *Ti' Є Rp (Tj)* to *Tj*. Therefore, in the new Bayesian network the set of variables is *V = T U T' U D.* The parent set of any original term node *Tj Є T* is defined as Pa *(Tj) = Rp (Tj).* We use this topology, a bipartite graph, because it will support a very fast propagation algorithm in the term sub network. The complete Bayesian network contains three simple layers (see Figure 4.1), without connections between the nodes in the same layer, and this fact will be essential for the efficiency of the inference process in the whole network.

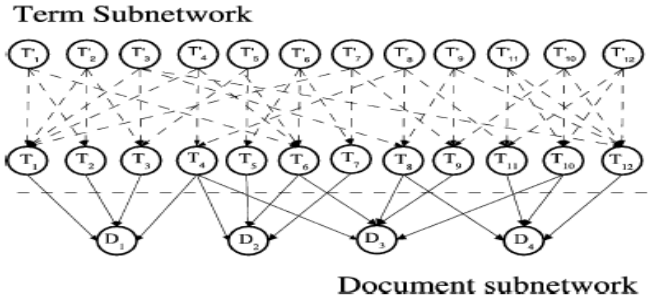


Figure 4.1 Topology of BNRM with two term layer

4.2. Learning term relationships: Clustering terms

To build the term sub network described previously, we have to determine which is the set *Rp*(*Tj*), for each term

node *Tj*, i.e., the set of terms that are most similar to *Tj*. We have designed a relatively simple method to obtain *Rp*(*Tj*).

In our case, a good learning algorithm would need to include in Pa (*Tj*) only terms that are positively correlated with *Tj*. The following expression, a maximum likelihood estimator, could be used to measure the strength of their co-occurrence relationship, from the perspective of the term *Tj*:

$$\text{strength } (Tj, Ti) = \frac{N_{TiTj}}{N_{Ti}}$$

i.e. a coefficient that measures the ratio between the number of documents in which *Tj* co-occurs with *Ti*, with respect to the total number of documents in which *Ti* occurs. When this quotient is close to 1.0, this means that almost all the documents indexed by *Ti* are also indexed by *Tj* so that *Tj* is quite similar to *Ti*. But anomalous behavior is observed in some cases: if, for instance, *Ti* occurs only in one document and *Tj* occurs in that document, the result would be 1.0; on the other hand, if we have the case in which *Ti* and *Tj* share five documents, of the five in which *Ti* is in the collection, the ratio is the same, but we would say that *Ti* and *Tj* are more closely related in the second example, although the value obtained by above equation is the same in both cases. To solve this problem, we will use a Bayesian estimator [11] instead of the maximum likelihood estimator:

$$\text{strength } (T_j, T_i) = \frac{N_{TiTj} + 1}{N_{Ti} + 2}$$

When this estimator is used, a new problem arises: imagine a pair of terms such as $N_{TiTj} = 0$, and also $N_{Ti} = 1$, then the strength (*Tj*, *Ti*) would be 0.33, a value that would be always greater than the one obtained when these terms co-occur once, for instance, and $N_{Ti} > 4$, a situation that is not very logical. To solve this problem, we have employed a modified strength function,

strength' (*Tj*, *Ti*) = 0 , when $N_{TiTj} = 0$, and
strength' (*Tj*, *Ti*) = strength*(Tj, Ti)*, otherwise

4.3.Specifying qualitative information

Once the graph has been built, the probability distributions stored in each node of the network must be estimated. Thus, all the root nodes, i.e., those which do not have parents, will store marginal distributions. In our specific case, the only nodes of this type are term nodes placed in the first term layer. For each root term node, we have to assess *p*(T*i*) and *p(.Ti)*; we use the following estimator: *p(Ti) = (1/M)* and *p(.Ti)* = 1−*p(Ti)* (*M* is the number of terms in the collection). The nodes with parents (term and document nodes) will store conditional probability distributions, one for each of the possible configurations that their parent nodes can take. Terms nodes in the second term layer must store the conditional probabilities *p(Ti*|pa*(Ti))*, where pa(*Ti*) is a configuration of values associated to the set of parents of *Ti*. Analogously, document nodes must store *p(Dj*|pa*(Dj))*.

Instead of explicitly computing and storing these probabilities, we use a *probability function* (also called a canonical model of multicausal interaction [6]). Each time that a given conditional probability is required during the inference process, the probability function will compute and return the appropriate value. We have developed a new general canonical model: for any configuration pa(*Dj*) (i.e., any assignment of values to all the term variables in *Dj*), we define the conditional probability of relevance of *Dj* as follows:

$$p(dj|\text{pa}(Dj)) = \sum_{Ti \in Dj,\, ti \in \text{pa}(Dj)} wij \qquad (1)$$

where the weights w_{ij} verify that $0 \le w_{ij}$, for all *i*, *j* and

$\sum_{Ti \in Dj} wij \le 1$ for all *j*.

The expression *Ti* ∈ pa*(Dj)* in Eq. (1) means that we only include in the sum those weights *wij* such as the value assigned to the corresponding term *Ti* in the configuration pa(*Dj*) is *ti*. So, the more terms that are relevant in pa(*Dj*) the greater the probability of relevance of *Dj*. The specific weights, *wij* used in this paper by our models, for each document *Dj* ∈ *D* and each term *Ti* ∈ *Dj*, are:

$$w_{ij} = \frac{TF_{ij} * IDF_i^2}{\sqrt{\sum_{T_k \in D_j} TF_{kj} * IDF_k^2}}$$

$TF_{kj} * IDF_k^2$=Product of TF and IDF of terms other than i^{th} term in the j^{th} document.

Finally, we have to define the conditional probabilities *p(Tj*|pa*(Tj))* for the terms in the original term layer *T*. For the same reasons we used probability functions in the document layer, we use a probability function belonging to the general canonical model defined in Eq. (1), where the weights *vij* measure the influence of each *T ' j* ∈ Pa*(Tj)* on term *Tj*:

$$p(tj|\text{pa}(Tj)) = \sum_{T_i' \in \text{Pa}(Tj)} vij \qquad (2)$$

4.4. Retrieving documents

Given a query *Q* submitted to our system, the retrieval process starts placing the evidence in the term sub network: the state of each term *T'iQ* belonging to *Q* is fixed to *t'iQ* (relevant). Then the inference process is run in the whole network obtaining, for each document *Dj*, its probability of relevance given that the terms in the query are also relevant, *p(dj|Q)*. Finally, the documents are sorted in decreasing order of probability to carry out the evaluation process. Taking into account the large number of nodes in the Bayesian network and the fact that it contains cycles and nodes with a great number of parents, general purpose inference algorithms cannot be applied due to efficiency considerations, even for small document collections. To solve this problem, we have designed a specific inference method which is composed of two stages, and constitutes an exact propagation algorithm (by virtue of the properties of the canonical model being used and the layered topology of the network [12]):

(1) The computation of the posterior probability of relevance of the term nodes belonging to *T*, *p(tj|Q)*,for all *Tj*, which is carried out by simply evaluating the following expression:

$$p(tj|Q) = \sum_{T i' \in \text{Pa}(Ti)} v_{ij}\, p(t_i'|Q) \qquad (4)$$

Notice that $p(t_i'|Q) = 1.0$ if $t_i' \in Q$, and $1/M$ otherwise (because terms in the *T*_ layer are marginally independent, the posterior probability of the terms which are not in the query coincides with their prior probability, $p(t_i'|Q) = p(t_i')$

= 1/*M* and the probability of the query terms is equal to 1); by substituting the weights *vij* in Eq. (3), the final expression for the calculation of *p(tj|Q)* is:

$$p(tj|Q) = \frac{1-\beta}{Sj} \sum_{Ti' \in Pa(Tj),i!=j} strength\_(Tj, Ti)\, p(ti'|Q) + \beta p(tj'|Q) \quad (5)$$

(2) The evaluation of the posterior probability of relevance of the document nodes, *p(dj|Q)* which can be carried out using the information obtained in the previous step, in the following way:

$$p(dj|Q) = \sum_{Ti \in Pa(Dj)} wijp(ti|Q) \quad (6)$$

5. System Design

We implemented a system for searching multiple collections of pdf documents. The architecture of the system is shown in figure 5.

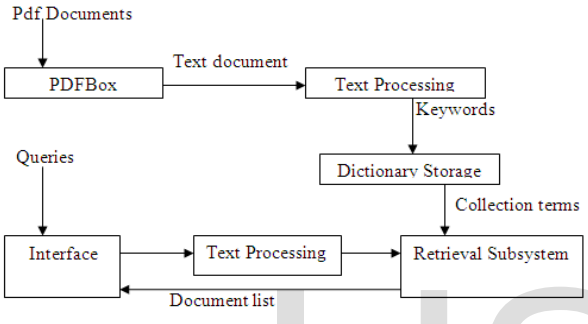


Figure 5.1 Architecture of the system

It is impossible to get keywords from pdf files directly. So we convert them in to text files. For this purpose we use apache PDFBox, which is an open source java library.

Then the text documents corresponding to the pdf files are undergone into text processing for extracting terms key terms of the documents. Then the terms, their dependent terms, the strength of dependence and the importance of terms in the document are stored in a text file. This text file is considered as a dictionary.

The retrieval subsystem is responsible for creating the inference network. It creates a collection net work, document network corresponding to each collection and query network also. The networks other than query network are static in nature. i.e. we create it only once. The query network is dynamic in nature. Whenever a user submits a query, the query networks are formed.

6. Conclusion

In this paper a new topology for representing term relationships, based on a term clustering method, has been presented. Instead of using a polytree as the underlying structure of the term sub network, we have designed a new graph, a bipartite graph (two layers of nodes representing the terms in the collection), that stores the strongest relationships among terms. The main advantage of this graph is that the exact propagation that had to be carried out in the original polytree is replaced by the evaluation of simple expressions, resulting in a very efficient method. The main application of this new model will be the retrieval of text documents, where, taking into account its topology and the whole inference method, we think that it will be competitive and efficient.

References

[1] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, NewYork, 1983.

[2] F. Crestani, M. Lalmas, C.J. van Rijsbergen, L. Campbell, Is this document relevant?. . . probably. A survey of probabilistic models in information retrieval, ACM Computing Survey 30 (4) (1991) 528–552.

[3] C.J. van Rijsbergen, Information Retrieval, second ed., Butter Worths, London, 1979

[4] H.R. Turtle, W.B. Croft, Uncertainty in information retrieval systems, in: Uncertainty Management in Information Systems: From Needs to Solutions, Kluwer Academic Publishers, Dordrecht, 1997, pp. 189–224

[5] S.E. Robertson, The probability ranking principle in IR, Journal of Documentation 33 (4) (1977) 294–304

[6] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan and Kaufmann, 1988.

[7] S.Lawrence, C.L Giles, Accessibility of information on the web, *Nature, 400 (July:8), pp 107-109, 1999*

[8] Howard Turtle and W. Bruce Croft. Evaluation of an Inference Network-Based Retrieval. ACM Transactions on Information Systems, Vol. 9,No.3,July1993,Pages187-222.

[9] Callan, J. P., Croft, W. B., & Harding, S. M., The INQUERY retrieval system, *In Proceeding of the 3rd international conference on database and expert systems applications*, (pp. 78-83). Springer-Verlag, 1992

[10] Z. Lu, J.P. Callan and W.B. Croft., Applying inference networks to multiple collection searching, Technical Report TR96–42, University of Massachusetts at Amherst, Department of Computer Science, 1996

[11] B. Cestnik, Estimating probabilities: a crucial task in machine learning, in: Proceedings of ECAI Conference, 1990, pp. 147–149.

[12] L.M. de Campos, J.M. Fernández, J.F. Huete, The BNR model: foundations and performance of a Bayesian network based retrieval model, Int. J. Approx. Reasoning 34 (2003) 265–285.

[13] L.M. de Campos, J.M. Fern´andez-Luna, and J.F. Huete, Clustering terms in the Bayesian network retrieval model: a new approach with two term-layers, *Applied Soft Computing*, pp. 149~158, April 2004.

IJSER