Journal of Systems Integration, 8, 379–404 (1998)
© 1998 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

A Ranking-Based Document Retrieval Model: The TEXPROS Approach

CHIH-YING WANG cywang@homer.njit.edu
Institute for Integrated Systems Research, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

PETER A. NG Peter_Ng@unomaha.edu
Department of Computer Science, College of Information Science and Technology, University of Nebraska at Omaha, Omaha, NE 68182

Abstract. In this paper, a ranking-based document retrieval model is proposed to incorporate with the browsing process. In TEXPROS (TEXt PROcessing System), the interactive browsing process is designed to allow the interactions between the system and a user for forming a strategy of retrieving documents and information from the document base. By gathering information, users could reformulate queries dynamically. During the browsing sessions, a predicate augmented an infrastructure (called Operation Network) is used to present the information about the document types, the synopses of the documents and where documents are deposited. The outcome of using the concept-based retrieval for searching requested documents with partial descriptions could be a large volume of returned documents. The ranking model is used to rank the returned documents according to the degree of their relevancy to the query. Based on the TEXPROS's dual models, an approach to creating the representatives of documents and queries is described as a basis for the proposed ranking model. By integrating the ranking model and the browsing system as a whole, an open retrieval environment is created, which can be customized for different application domains.

1. Introduction

In [1, 2, 3, 4, 5, 6, 7], various models of the information retrieval system have been proposed. The Boolean model compares Boolean query statements with the terms which represent a document. The probabilistic model computes the probabilities of relevance of a collection of documents. The vector-space model represents both the user's queries and the documents by a set of controlled terms (such as, the index terms) and computes the similarities between them. In this model, both the document and the user's query are also represented by a set of index terms. The traditional Boolean model has two valid values 0 and 1 for characterizing the similarity between the query and the returned documents. A document is retrieved only if there is a closely exact match between the sets of the controlled terms of the document and the query. Therefore, for this model, it is difficult to derive a ranking list for the relevant documents. In probabilistic model [7], although the returned documents can be ranked according to their relevance to a given query, the ranking approach does not improve the retrieval effectiveness, because it is difficult to obtain the values for the term-occurrence parameters (such as, term dependencies) [8]. In the vector-space model, the document and the query are represented by the vectors, which allow to compute the distance between them. Then, the closer similarity between a document and a given query has a shorter distance. The vector-space model is easy to use and generate. A shortcoming of this model is that users have difficulty for specifying the dependencies between terms appeared in the document and the query.

In 1949, G. K. Zipf formalized the relationship between the frequency of term occurrences and their ranking order [9]. Most of the models which support the ranking method use the concepts of the term weights and the probabilities for the terms to be appeared both in queries and documents [8, 10, 11]. Based on the frequency of a term occurred in a collection of existing documents, the weight of the term (in the vector-space model) and the initial probability of relevance of the collection of these documents (in the probabilistic model) can be computed. These models help to locate the relevant information during the retrieval phase.

There are two major concerns for computing a ranking list of the relevancy of documents based on the weights and the frequencies of terms occurred in the documents in the real-world office automation environment. The first concern is that, in an office environment, the document tends to fall into a small number of certain document types. They are stored in some storage units, such as folders. For retrieving a document, a user normally knows only its type or the folder where it is located. However, the traditional information retrieval systems cannot handle the situation when users only remember certain descriptions or properties of a document to be retrieved, since all the index terms are extracted from the contents of the documents. An example of a description of a document is "perhaps, on the right hand side of the letter, it has a graphical representation, ***50*** in which the ***0*** contains a string of characters ***acm***." The second concern is about the ranking. One of the reasons for incorporating the ranking into information systems is that the size of the returned documents from a query is enormously huge and the user has no idea which document should be examined first. For the uncertainty of what the user needs in the early query phase, the system might return thousands of documents. In the evaluation phase, the user gathers some information from the returned documents to refine their original query for obtaining more relevant documents. However, the system may give an incorrect ranking of the returned documents, since the original query is so vague. An example of this case is that a document, which probably has the most valuable information for the user, could be ranked last.

For solving the first concern, we need a mechanism which allows users to apply the vague query on both the frame instance and document bases (the data) and the system catalog (the meta-data of the data). For solving the second concern, a mechanism is required which could summarize the returned result regardless of its extremely large volume of returned documents. The summary of the result provides the potential, useful information to the user as early as possible in the evaluation phase.

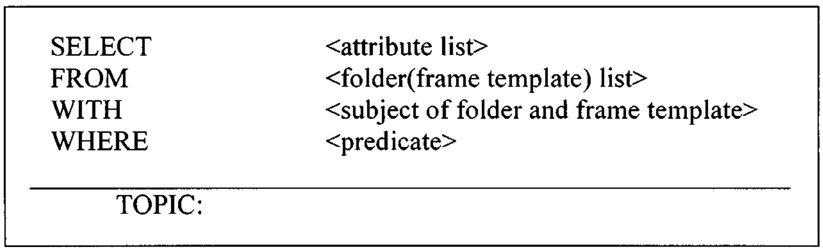
This paper is organized as follows: in Section 2, the dual models and the representation of documents in TEXPROS are described. In Section 3, the retrieval mechanism and its supporting system architecture are given. The operation network is presented as an effective way for encapsulating information which is relevant to a given query. The ranking model is introduced in Section 4, which tackles the problems mentioned above, by taking the dual models and the retrieval mechanism into consideration. In Section 5, the concept of application domains is used to demonstrate the open retrieval environment of the proposed retrieval system, which is one of the major characteristics resulting from the integration of the ranking model and the browsing process. Finally, the conclusion and the future research are described.

2. TEXPROS Data Model

The TEXPROS employs a dual-modeling approach to describe, classify, categorize, file and retrieve documents. The document model consists of two hierarchies: a document type hierarchy (DTH), which depicts the structural organization of the documents, and a folder organization (FO), which represents the user's real-world document filing system. The DTH captures the structural similarity of different types of documents. In a user's office environment, by identifying common properties for each document class, documents are partitioned into different classes. Each document class is represented by a frame template, which describes the common properties in terms of attributes [12, 13] of the documents of the class and is referred to as the document type (or simply type) of that class. As a powerful abstraction for sharing similarities among document classes while preserving their differences, the frame templates are related by specialization and generalization and are organized as a document type hierarchy whose members are related by an is-a relationship. The is-a relationship and the mechanism of inheritance help to reduce the complexity of models and redundancy in specifications [14]. Upon the arrival of a new document, it is classified into a document type [15]; based upon the frame template of the document type, information is extracted pertinent to the significance of the user [16] to yield a synopsis of the document, which we called a frame instance. The frame instance of a document type consists of a set of attribute-value pairs, where values are obtained from the original document it represented. The frame instances of different types are deposited in folders over time [16, 17, 18, 19]. Hence we consider that folders are heterogeneous repositories and are related by an inclusion relationship to form a folder organization, which is one of the common ways of organizing and storing documents for their retrieval in an office environment. The folder organization is defined by a user corresponding to the user's view of the document organization, which is obtained by repeatedly dividing documents for particular areas of discourse into groups until well-defined groups are reached. Filing a frame instance into a folder of the folder organization, it traverses from rooted folder into various folders provided the frame instance has met the folders' criteria, which are described in terms of predicates [18, 19]. Strictly speaking, the original documents and their associated frame instances are stored in the original document base and the frame instance base, respectively. The folders consist of only pointers, which point to where the frame instances are located [18, 19, 24, 25]. However, taking DTH into account, these frame instances can be grouped together according to their types and additional premises, such as date, related subject, etc.

3. Retrieval System in TEXPROS

The TEXPROS employs an integrated system catalog to provide a centralized retrieval environment for processing incomplete, imprecise and vague queries, as well as providing



```
SELECT        <attribute list>
FROM          <folder(frame template) list>
WITH          <subject of folder and frame template>
WHERE         <predicate>
_____________________________________________
     TOPIC:
```

Figure 1. Query interface.

meaningful responses to users when empty answers are arisen [12, 20, 21]. A user issues a complete query by using the query interface, as shown in Figure 1 [12, 21]. In addition to each term appearing in the query, which must be consistent with the index term used in the database, no variable is allowed to appear in the complete query. Otherwise, the query is said to be incomplete. The imprecise query arises if some of the components specified in the query do not match any internal representation of the system. It is quite often that the user does not even know what the retrieving target is. If the user does not have any idea of how to specify a complete query for his request, the "TOPIC" part will be used repeatedly to describe his retrieval goal. Vague queries can be entered to the system until sufficient information obtained to the extent that the user is able to use this information to construct a complete query for his request.

Upon receiving a query from a user, the input query is checked whether it is a formal query or a vague query. The vague query is then passed to the browser, which goes through the system catalog via of object networks for looking up relevant information (i.e., all frame templates possibly related to the user's request), and possible repositories of information attributes to describe the properties of the data to be retrieved. Once the input query is stated formally according to the specifications, the query is transferred to the query transformation mechanism, in which the algebraic query formulation transforms a complete query into a set of algebraic queries, which are to be processed by the query processor to assist in answering the corresponding user's original query. If the formal query is incomplete, the context construction mechanism is applied repeatedly to generate a complete query. Finally, the query processor executes the set of algebraic queries after its formulation. When processing of queries fails by responding with an empty answer, the original query is passed to the query generalizer to produce cooperative explanation for the empty answer.

The system catalog contains the actual meta-data knowledge of the document filing organization, and the domain knowledge (the relevant contents of documents) of the document base for increasing the effectiveness of the document retrieval system. This supports not only the procedure of query processing as a traditional system catalog does, but also the query transformation, browser and generalizer mechanisms [12, 21]. A uniform representation, such as frame, is employed for describing the meta-data and domain knowledge, and the contents of documents. This unified approach allows to use the same methods for retrieving and managing the knowledge at system and operational levels, and thus, eliminates problems of duplicate knowledge and translation between different knowledge representations.

In addition to reflecting the meta-data of the document filing organization, the system catalog also includes a thesaurus. The thesaurus comprises synonyms with semantic associations, and the descriptions of the associations of the synonyms in terms of folders, frame templates and attributes. It provides users with the flexibility of using different terms to refer to the same term, and thus, provides the unification of key terms, which appear in original documents. Therefore, in combining with the thesaurus, the system catalog is one of the facilities for augmenting the power of the retrieval system, which deals with not only systematic queries but also incomplete and vague queries. The system catalog is stored in the system folder in an unified manner as we store the frame instances (the data). Figure 2 depicts the system catalog, and the description of its frame templates can be referred to the literature in [12, 20, 21].

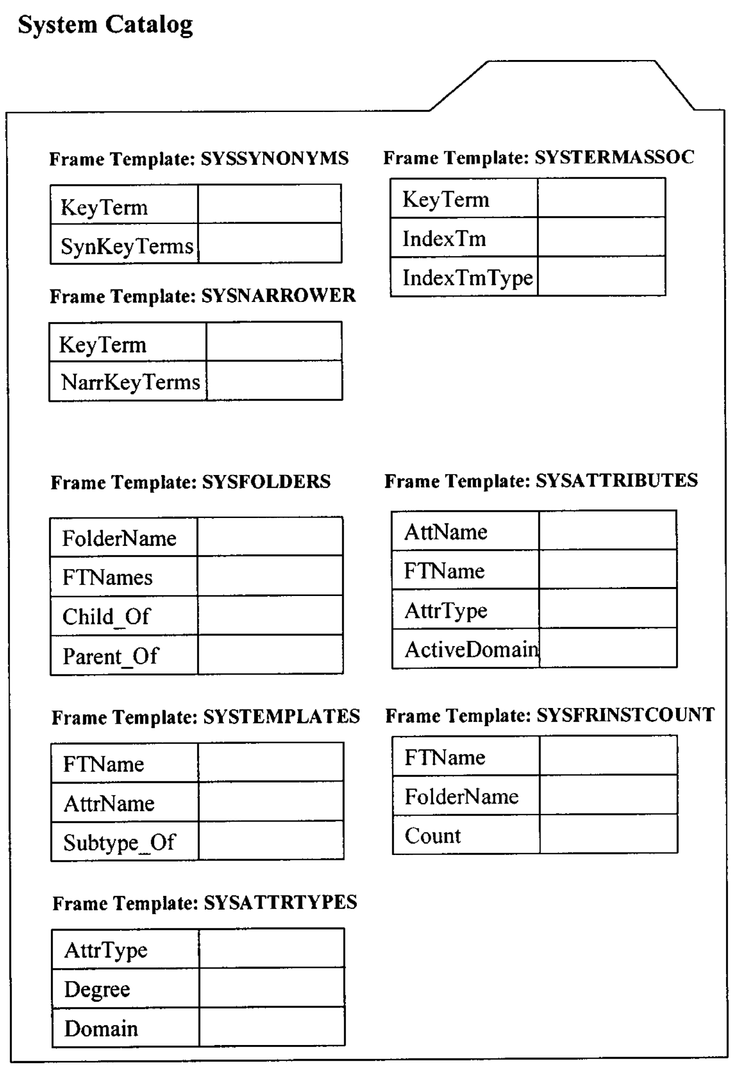
Taking the advantage of representing uniformly the system catalog and database itself, in TEXPROS, an object network (O-Net) [12, 20, 21] is created to present the view of the schema (meta-data) of the database (about the document type hierarchy and the folder organization) and the database itself (the frame instances of various types within the folders). The object network contains four different types of objects in the system (namely, the *Folder,* the *Frame Template,* the *Attribute,* and the *Value*) and specifies their relationships among these objects. It is not only preserves the DTH (whose information contains in SYSTTEMPLATE) and FO (whose information contains in SYSFOLDER), but also captures the relationships among the different objects.

The object network always represents a snapshot of a subset of the system catalog. For each topic entered by a user, the browser (a component of the TEXPROS) [12, 20, 21, 24] is able to respond with an object network, which represents all the information related to the topic, by browsing through the system frame instances in the system catalog. The browser mechanism accepts any query for more than one topic entered by the user. Several individual object networks, each of which is associated with a topic, are constructed first. Connecting these object networks to form a connected object network is depending on the relatedness of the corresponding topics [12, 20, 21].

In the remainder of this section, we shall describe the browsing process applied on the object network, which represents the relevant information contained in the system catalog.

3.1. Architecture of Browser

A browsing technique consists of two components, namely the browser and the browsing process. A browser is an organization of query formulation, network construction, search engine, etc., which supports the browsing process. Figure 3 shows the architecture of the browser, which is a major component of the retrieval subsystem in TEXPROS [24]. The components of the browser are divided into three layers: the controller layer, the service provider layer, and the storage system layer. The controller layer contains the controllers, which monitor the interactive browsing process. The service provider layer provides the service upon the requests directed from the first layer. The storage system layer is the place



System Catalog

Frame Template: SYSSYNONYMS

KeyTerm	
SynKeyTerms	

Frame Template: SYSTERMASSOC

KeyTerm	
IndexTm	
IndexTmType	

Frame Template: SYSNARROWER

KeyTerm	
NarrKeyTerms	

Frame Template: SYSFOLDERS

FolderName	
FTNames	
Child_Of	
Parent_Of	

Frame Template: SYSATTRIBUTES

AttName	
FTName	
AttrType	
ActiveDomain	

Frame Template: SYSTEMPLATES

FTName	
AttrName	
Subtype_Of	

Frame Template: SYSFRINSTCOUNT

FTName	
FolderName	
Count	

Frame Template: SYSATTRTYPES

AttrType	
Degree	
Domain	

Figure 2. The system catalog.

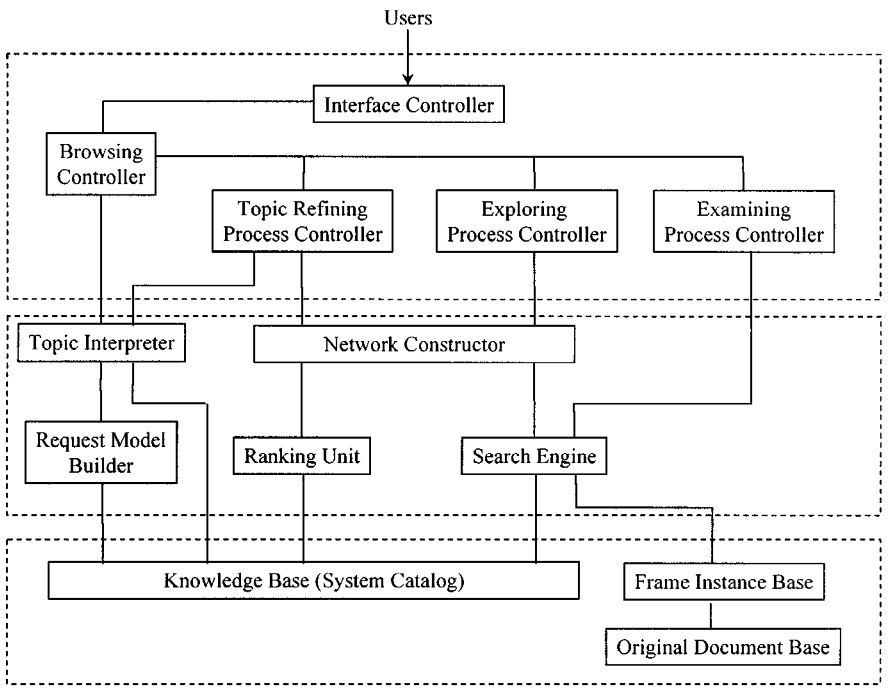


Figure 3. The architecture of the browser.

where all the meta-data and data are stored. This layer provides the low-level operations directly to the storage system from the second layer.

3.2. Browsing Process

Proceeded through the interactions between the users and the system, a browsing process is designed to guide users to locate the needed information [20, 22, 23, 24, 25], by letting users to explore the system knowledge and to examine the frame instances and their corresponding documents. Figure 4 depicts a browsing process. There are three main subprocesses: the topic refining process, the exploring process and the examining process. The browsing process is proceeded interactively between the system and the user. The whole process of formulating a "good" query from a sequence of vague queries can be described by a sequence of browsing process cycles. A browsing process cycle begins with receiving a users' query, as an input for the network construction process, followed by one of the three subprocesses. And the browsing cycle ends with a newly refined vague query or exit requested by the user.

The topic refining subprocess translates the users' vague query into a topic and constructs

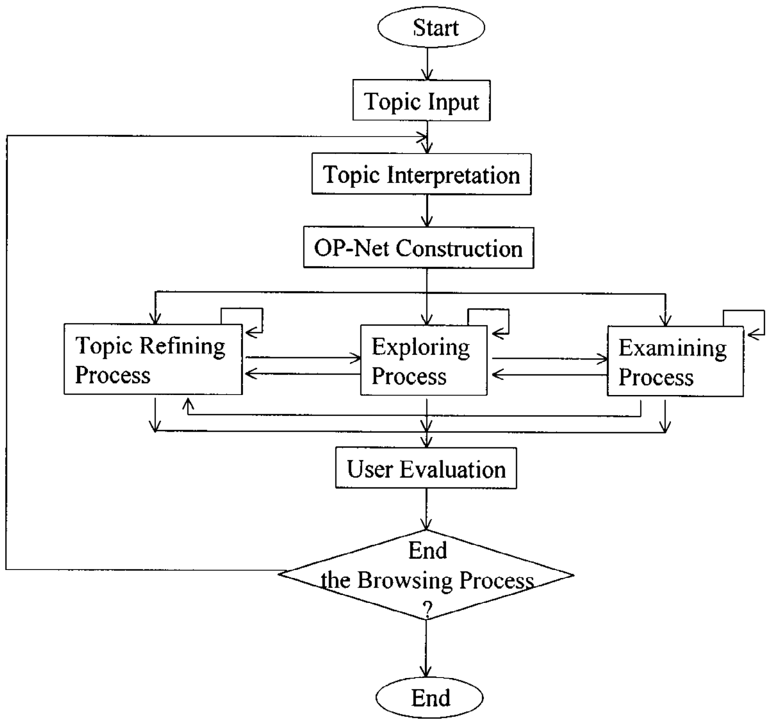


Figure 4. The browsing process.

its operation network, a subset of the object network, in which the objects related to the vague query and their relationships are identified and displayed to the user. Normally, the initial query does not precisely describe the intention of a user because of inadequate or incorrect information provided by the users [26]; and therefore, the result returned by the browser may not always be satisfied by the users. The exploring subprocess helps users to find out the interrelationships among objects (e.g, folders, templates, attributes and values) in the operation network. Users can activate the exploring process by choosing an object (i.e., a node) from the operation network. An exploring network for this significant node is constructed, where the node acts like a topic node in topic refining process. And thus, this process helps users gain more knowledge which is helpful for refining the original vague query. The only difference between the construction of the exploring network and the operation network is that the exploring network is constructed based on the existing operation network instead of consulting the system catalog and the frame instance base. Proceeding through the objects of the operation network or exploring network, the examining process allows users to browse through the objects at the frame instance level (the database itself). Browsing through the frame instances helps resolve the ambiguity problem created by having more than one frame instance satisfying the conditions, which are set by the vague query. Finally, an evaluation subprocess allows users to review the result, to gather some information about the system, and to decide what step is to be taken next.

3.3. Operation Network

For a vague query as an input, the network construction process constructs an operation network, a subset of object network, in which the objects related to the vague query and their relationships are identified and displayed to the user. The object network is the underlying structure upon which, an operation network is derived with respect to each vague query. The browsing process uses the operation network to organize the returned results and respond rapidly to the user's input. In the remainder of this section, we shall give the definition of the operation network.

Definition 1: (Operation Network). An operation network (OP-Net) is a four-tuple, $OP = [T_{OP}, P_{OP}, FI_{OP}, G(V, E)]$, where

1. T_{OP} is a topic related to the context of the browsing process;
2. P_{OP} is a predicate related to the topic T_{OP};
3. $FI_{OP} = \{fi \mid fi$ is a frame instance satisfying $P_{OP}\}$, and
4. G(V, E) is a graph, where
   - each node in V(G) is a frame instance repository, and
   - each edge (i, j) between two repositories FIR(i) and FIR(j) represents that these two repositories have at least one common frame instance.

The object network (O-Net) [12, 20] is transformed into an operation network. The original objects in the O-Net become frame instance repositories in OP-Net. Each frame instance repository preserves its object type and stores a set of frame instances. Before a user issues a query, each object, according to its property, is used to set up the condition for qualifying the associated frame instances. For example, let *Folder(CIS)* be an object. The associated frame instances and the condition can be represented by {fi | fi is a frame instance in a *Folder*, whose name is *CIS* }.

For an issued query, the object identification process identifies firstly the possible object types for each term in the query. For example, upon the arrival of a query *CIS*, the object identification process concludes that the *CIS* could be an instance of a *Folder* or a *Value*. These two objects are used to set up the condition for the query. For this case, the possible relevant frame instances and the condition can be represented by {fi | (fi is a frame instance) AND (fi is in the *CIS Folder*) OR (fi has *CIS* as a *Value*)}. After a query is issued, the frame instance is qualified to be in a frame instance repository, if it satisfies the predicate defined in the user's query and the object, which is corresponding to the frame instance repository.

By observation, we tends to remember the contents of a document; but, to an extent, we have difficulty to retrieve it by specifying query in a formal retrieval system unless we are familiar with the meta-data. Motro proposed an important concept “access by values” [23]. By providing “access by values,” the system gives users an intuitive way of retrieving the information. However, this concept could be extended by considering its applications in an office automation environment. The values in “access by values” are not restricted to those values referred to the traditional database system. Applying in TEXPROS, these values can also refer to the instances of various object types, such as the *Folder*, the *Template*, and the *Attribute* types. This helps not only to cover more relevant documents, but improve the recall. High recall implies that more documents are retrieved. The large volume of returned documents may create a problem for users during the evaluation phase. Users have no idea where to start the evaluation process for reviewing these documents. In this paper, we shall propose document ranking as one of the solutions for solving this problem by giving their degrees of relevancy with respect to the given query.

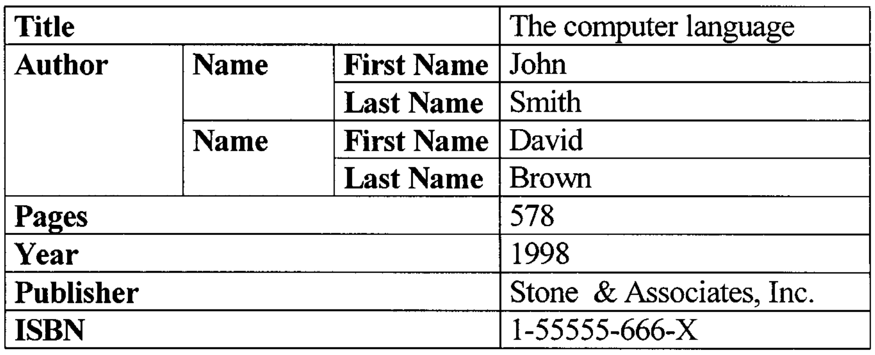
As the volume of returned document becomes larger, it is more difficult to summarize the returned result given to the users. By introducing the concept of the frame instance repository, we can organize the returned documents by categorizing the returned documents and associating them with the relevant objects. Since every object has its specific semantics, it provides useful information to users during the browsing process.

4. Ranking Model

At the second layer, there is a ranking unit, which produces a ranking list. The ranking list specifies the degree of relevancy of the frame instances of the returned documents with respect to a given query in FI_{OP}. After an OP-Net is constructed, the network constructor allows to make a request to the ranking unit to rank the relevancy between the frame instances of the returned documents with respect to the current query. In the following sections, we will describe the ranking model along with the ranking function.

4.1. Ranking Model—TEXPROS Approach

In TEXPROS, a frame instance is a synopsis of an original document and is defined as a set of attribute-value pairs [13, 17]. Figure 5 is an exemplary frame instance of a document of the *Book* type. In the frame instance, the repeating group of attributes (e.g. Author Name) and the composite attribute (e.g. Name) are allowed. However, for illustration purpose, we only consider the template without repeating group of attributes and composite attributes. Each document is classified as a document type based upon the document type hierarchy. Based upon of its type (represented by its frame template), information is extracted from the document to form a frame instance. As the original document is stored in the document base, by depositing its corresponding frame instance into folders of the folder organization, we mean that these folders have pointers pointing to the frame instance located in the frame instance base. Therefore, the folder organization, the document type hierarchy and its corresponding frame instance are considered as the properties of a document. We, thus, define the *signature* for documents.



Title			The computer language
Author	Name	First Name	John
		Last Name	Smith
	Name	First Name	David
		Last Name	Brown
Pages			578
Year			1998
Publisher			Stone & Associates, Inc.
ISBN			1-55555-666-X

Figure 5. The frame instance of a book.

Definition 2: Signature for a document. The signature of a document D, Sig(D) = $[F_D, T_D, Fi]$, where

1. F_D is the set of folders where D is deposited;
2. T_D is the document type of D, and
3. Fi is the frame instance corresponding to D.

For example, let Sig(D) = [(CIS, PHD), MEMO, {(Sender, John), (Receiver, Tom), (Subject, TAmeeting), (Date, 9/11/96), (CC, CIS)}] be the signature of a document D. Then there is a document D of the memo type and is deposited in the folders, *CIS* and *PHD*. The information about the content of a document is revealed by the frame instance in the signature.

Now, we shall define the signature for user's query. Given an original query Q, applying the query rewriting and normalization processes, Q is rewritten as Q', which is in the disjunctive normal form.

Therefore, Q' can be represented by $\bigcup_{i=1}^{r}$ qi, where each qi is in the conjunctive normal form. We call qi the And-clause. For each And-clause qi, its signature is defined as follows.

Definition 3: The signature for And-Clause. The signature of a qi, Sig(qi) = $[F_{qi}, T_{qi}, A_{qi}, V_{qi}]$, where

1. F_{qi} is the set of folders appeared in qi;
2. T_{qi} is the set of templates appeared in qi;
3. A_{qi} is the set of attributes appeared in qi, and
4. V_{qi} is the set of values appeared in qi.

We extract information about the sets of folders, templates, attributes and values from each qi in Q and put them in the signature. Sometimes, some of these information may not be available. For instance, the And-Clause $qi = (\mathit{VALUE}[\mathit{D.\ Sanders}]\ \mathit{AND\ FOLDER}[\mathit{CIS}])$ does not contain any information about the document type and the attributes. For this case, we shall treat the information about the document type and the attributes, which are not in the qi, as insignificant and don't care values in the signature of the And-clause qi. We shall use "*" in $Sig(qi)$ to indicate the don't-care values. Therefore, the signature of qi is $Sig(qi) = [\{\mathit{CIS}\}, *, *, \{\mathit{D.\ Sanders}\}]$. With the definition of the And-clause, we now define the signature of the query.

Definition 4: The signature for the query. The signature of a query Q, Sig(Q) $=$ $[\mathrm{F_Q}, \mathrm{T_Q}, \mathrm{A_Q}, \mathrm{V_Q}]$, where

1. $\mathrm{F_Q}$ is the set of folders appeared in Q;
2. $\mathrm{T_Q}$ is the set of templates appeared in Q;
3. $\mathrm{A_Q}$ is the set of attributes appeared in Q, and
4. $\mathrm{V_Q}$ is the set of values appeared in Q.

Clearly, the signature of a query Q, $\mathrm{Sig(Q)} = \bigcup_{\mathrm{i=1}}^{\mathrm{r}} \mathrm{Sig(qi)}$.

Both the signatures of a document and the And-clauses appeared in a user's query are structurally identical. This allows us to determine the degree of relevancy of a frame instance with respect to a user's query Q by comparing the similarity between the document and the query, which is, in turn, determined by the similarity of these two signatures for the document and the user's query. The similarity between a document and the user's query Q can be determined by computing the similarity between the document and each And-clause qi in the user's query Q. In this section, we will describe how we can compute the similarity of the signatures for a returned document and a given query. Thus, it is possible to rank the degree of relevancy of frame instances with respect to user's query in our system.

4.2. Ranking Function

Given a user's query, the retrieval system returns documents which qualify the And-clauses of the query. The ranking function is used to compute the similarity between a returned document and the given query. The query is represented in the disjunctive normal form. The ranking function needs to take into consideration the dependency between terms introduced by AND and OR operators. Our approach is to use the signature to take care the AND operator; then we take care the OR operator by summing up the scores of each And-clause. The document receiving higher score has the higher rank. The similarity is computed using the following equation.

Equation 1:

$$\text{Sim(Fi, Q)} = \sum_{qi \in Q} (Sim(F_D, F_{qi}) + Sim(T_D, T_{qi}) + Sim(Fi_D, (A_{qi}, V_{qi}))).$$

The equation consists of the following three parts.

1. Sim($\mathrm{F_D}$ $\mathrm{F_{qi}}$)

Sim($\mathrm{F_D}$, $\mathrm{F_{qi}}$) can be perceived as an equation for computing the similarity between the two sets of folders, one containing the document *D*, and the other is specified in an And-clause qi. This similarity is the ratio of the number of common frame instances in the folders, specified in the And-clause qi and containing the document *D*, over the total number of frame instances appeared in the folders containing *D* and in the folders specified in qi.

Let

$$\begin{aligned} \mathrm{F_{qi}} &= \{\mathrm{F_{Qil}} \mid \mathrm{F_{Qil}} \text{ is a folder, } 1 \leq 1 \leq \mathrm{n}\}, \text{ and} \\ \mathrm{F_D} &= \{\mathrm{F_{Dj}} \mid \mathrm{F_{Di}} \text{ is a folder, } 1 \leq \mathrm{j} \leq \mathrm{m}\}. \end{aligned}$$

be the sets of folders, where n and m are the numbers of the folders in $\mathrm{F_D}$ and $\mathrm{F_{qi}}$ respectively. Let $FI(FQil)$ represent the frame instance set associated with the folder $\mathrm{F_{Qil}}$. Then, $\mathrm{FI_{qi}}$ and $\mathrm{FI_D}$, the numbers of frame instances in $\mathrm{F_{qi}}$ and $\mathrm{F_D}$, respectively, can be computed as follows:

$$\begin{aligned} \mathrm{FI_{qi}} &= \mathrm{FI}\left(\bigcup_{l=1}^{n} \mathrm{F_{Qil}}\right), \text{ and} \\ \mathrm{FI_D} &= \mathrm{FI}\left(\bigcup_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{F_{Di}}\right). \end{aligned}$$

If $\mathrm{F_{qi}} = \emptyset$ (i.e. there is no folder specified in the And-clause qi), then Sim($\mathrm{F_D}$, $\mathrm{F_{qi}}$) is equal to 0. When $\mathrm{F_{qi}} \neq \emptyset$, Sim($\mathrm{F_D}$, $\mathrm{F_{qi}}$) can be computed using the following equation.

Equation 2:

$$Sim(F_D, F_{qi}) = \frac{2\#(FI(F_D \cap F_{qi}))}{\#(FI(F_D)) + \#(FI(F_{qi}))}.$$

In this equation, #A represents the number of elements in the set A.

2. Sim ($\mathrm{T_D}$, $\mathrm{T_{qi}}$)

In the ranking function, the second component is related to the information of the document types. Each frame template represents a document type, and is identified by a set of attributes. For instance, if we consider electronic mails as a document type called E-MAIL, then it is reasonable to assume that the frame template of the E-MAIL type is specified by the attributes Sender, Receiver, Subject, and cc. In this case, we say that the document T_D is an email and is identified by $A_D = \{Sender, Receiver, Subject, cc\}$.

Let $T_{qi} = \{T_{qip} \mid T_{qip}$ is a frame template, $1 \le p \le k\}$ be a set of frame templates appeared in an And-clause qi. Each T_{qip} can be identified by a set of attributes A_{qip}. Let A_{qi} be a set of common attributes, which appeared in the set of frame templates T_{qi} specified in the And-clause qi. Then A_{qi} can be represented as follows:

$$A_{qi} = \bigcap_{p=1}^{k} A_{qip}.$$

Let T_D be the document type of the document D and let T_{qi} the set of frame templates appeared in an And-clause qi. If $T_{qi} = \emptyset$, then $Sim(T_D, T_{qi})$ is equal to 0. When $T_{qi} \neq \emptyset$, we compute $Sim(T_D, T_{qi})$ by using the following equation.

Equation 3:

$$Sim(T_D, T_{qi}) = \frac{2\#(A_D \cap A_{qi})}{\#(A_D) + \#(A_{qi})}.$$

The similarity between T_D and T_{qi} is the ratio of the total number of common attributes, which are in the document type T_D and are appeared in all the frame templates T_{qip} specified in the And-clause qi, over the total number of attributes, which are in T_D or appeared in all the frame templates T_{qip} specified in the And-clause qi.

3. $Sim(Fi_D, (A, V)_{qi})$

Let Fi_D be the frame instance of a returned document, which contains a set $(A,V)_D$ of attribute-value pairs. Let $Fi_D = (A,V)_D$ Let $(A,V)_{qi}$ be the set of attribute-value pairs appeared in the user's query qi. Without losing any generality, we assume that both qi and Fi_D are referred to the same index terms for the attributes, if they are of the same semantics. (This can be done by consulting the system catalog of the TEXPROS.) And we shall use V_D to refer to $(A,V)_D$ and V_{qi} to refer to $(A,V)_{qi}$ if there is no confusion.

In computing $Sim(Fi_D, (A,V)_{qi})$, we take the values from the frame instance Fi_D, and the values appeared in user's query qi into consideration. In the vector model [2], the index terms form a vector space. In this vector space, we use the distance between vectors to compute the similarity between a user's query qi and a return D (which is represented by its frame instance Fi_D). Both the query and the document are characterized in terms of vectors. Each value of an attribute-value pair appeared in its frame instance Fi_D is referred to as an index term occurred in the returned document D. For each document D, there corresponds a frame instance which consists of a set $(A,V)_D$ of attribute-value pairs, and therefore there corresponds a set V_D of values appeared in the document D. Likewise, we use V_{qi} to refer to the set of values for the set $(A,V)_{qi}$ of attribute-value pairs appeared in the user's query qi. In [3], the weight of a value V in the set of V_D can be computed as follows:

Equation 4:

$$W(V) = \log\left(\frac{N}{n}\right) + 1,$$

where W(V) is the weight of a value V, which is occurred in a document D; N is the total number of documents in the document-base of the system, and *n* is the number of documents containing the value V.

Then the common value set V_c between the set V_D of values occurred in the returned document D and the set V_{qi} of values specified in the user's query qi is $V_C = V_D \cap V_{qi}$. Let

$$\begin{aligned} V_C &= \{V_i \mid V_i \text{ is a value}, 1 \leq i \leq r\}. \\ V_D &= \{V_j \mid V_j \text{ is a value}, 1 \leq j \leq s\}. \\ V_{qi} &= \{V_k \mid V_k \text{ is a value}, 1 \leq k \leq t\}. \end{aligned}$$

Then Sim(Fi_D, (A, V)$_{qi}$) is Sim(V_D, V_{qi}) which can be computed as follows.

Equation 5:

$$Sim(V_D, V_{qi}) = \frac{\sum_{Vi \in V} (W(V_i))^2}{\sqrt{\sum_{V_j \in V_D} (W(V_j))^2 \times \sum_{V_k \in V_{qi}} (W(V_k))^2}}.$$

In the remainder of the section, we shall give a ranking example.

4.3. Example

Given a user's query **Q** and two frame instances **f1** and **f2**, the similarity between the query and each frame instance will be computed. The query and the description of these two frame instances are given in Figure 6. Assume that **Q** could be rewritten as **Q**′, after the object identification and the query rewriting processes.

Q′ ≡ (ATTRIBUTE(Sender) **AND** VALUE(Roy) **AND** TEMPLATE(Memo)
 AND VALUE(TAMeeting) **AND** VALUE(CIS))
 OR (ATTRIBUTE(Sender) **AND** FOLDER(Roy) **AND** TEMPLATE
 (Memo) **AND** VALUE(TAMeeting) **AND** VALUE(CIS))
 OR (ATTRIBUTE(Sender) **AND** VALUE(Roy) **AND** TEMPLATE(Memo)
 AND VALUE(TAMeeting) **AND** FOLDER(CIS))
 OR (ATTRIBUTE(Sender) **AND** FOLDER(Roy) **AND** TEMPLATE
 (Memo) **AND** VALUE(TAMeeting) **AND** FOLDER(CIS)).

The query **Q**′ consists of the following four And-clauses:

$\mathbf{q_1}$ ≡ ATTRIBUTE(Sender) **AND** VALUE(Roy) **AND** TEMPLATE(Memo)
 AND VALUE(TAMeeting) **AND** VALUE(CIS);

$\mathbf{q_2}$ ≡ ATTRIBUTE(Sender) **AND** FOLDER(Roy) **AND** TEMPLATE(Memo) **AND** VALUE(TAMeeting) **AND** VALUE(CIS);

$\mathbf{q_3}$ ≡ ATTRIBUTE(Sender) **AND** VALUE(Roy) **AND** TEMPLATE(Memo) **AND** VALUE(TAMeeting) **AND** FOLDER(CIS);

$\mathbf{q_4}$ ≡ ATTRIBUTE(Sender) **AND** FOLDER(Roy) **AND** TEMPLATE(Memo) **AND** VALUE(TAMeeting) **AND** FOLDER(CIS).

The signature for these And-clauses are as follows:

Sig($\mathbf{q_1}$) = [∗, {Memo}, {Sender}, {Roy, TAMeeting, CIS}];
Sig($\mathbf{q_2}$) = [{Roy}, {Memo}, {Sender}, {TAMeeting, CIS}];
Sig($\mathbf{q_3}$) = [{CIS}, {Memo}, {Sender}, {Roy, TAMeeting}];
Sig($\mathbf{q_4}$) = [{Roy, CIS}, {Memo}, {Sender}, {TAMeeting}].

The signatures for two documents, whose frame instances are f1 and f2, are as follows:

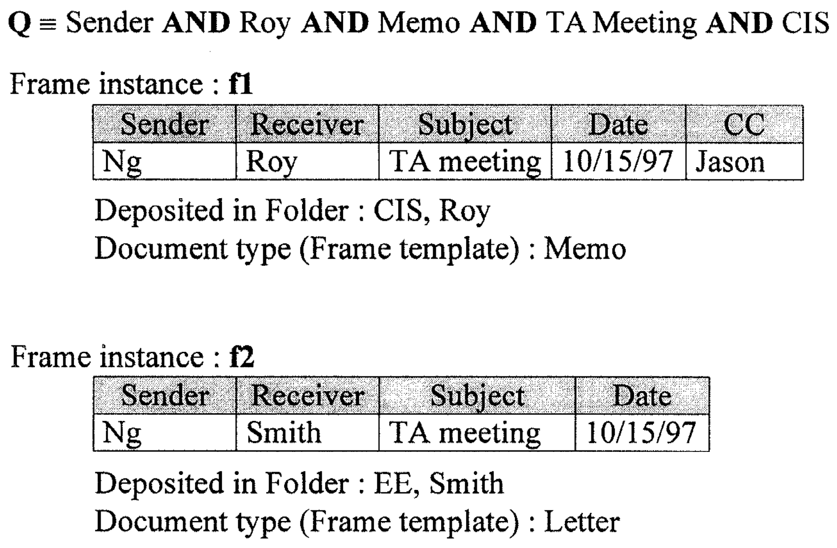
Sig(**f1**) = [{CIS, Roy}, {Memo}, {(Sender, Ng), (Receiver, Roy), (Subject, TAMeeting), (Date, 10/15/97), (CC, Jason)}];
Sig(**f2**) = [{EE, Smith}, {Letter}, {(Sender, Ng), (Receiver, Smith), (Subject, TAMeeting), (Date, 10/15/97)}].

The similarity between the document and the query is the summation of the similarity between the two sets of folders, one containing the documents D and the other is specified in an And-clause qi, the similarity between the document type T_D of the document D and the set of frame templates T_{qi}, appeared in an And-clause qi, and the similarity between the values from the frame instance Fi_D and the values appeared in user's query qi. In brief, the similarity between the document and the query can be computed in terms of the similarities with respect to the folders, templates (document types) and values contained in their corresponding frame instances. Consider the And-clause $\mathbf{q_3}$ and the frame instance **f1**. The similarity between the document and the query can be computed using the following steps.

Step 1: Compute { *Sim(F_D, F_{qi})* }

In this example, $F_D = F_{f1}$ = {Roy, CIS} and $F_{qi} = Fq_3$ = {CIS}. Assume that the folders Roy and CIS have 6 and 10 frame instances, respectively, of which five frame instances are in common. According to the Equation 2,

$$Sim(F_D, F_{qi}) = \frac{2\#(FI(F_D \cap F_{qi}))}{\#(FI(F_D)) + \#(FI(F_{qi}))} = \frac{2 * 10}{11 + 10} = 0.95.$$



Q ≡ Sender **AND** Roy **AND** Memo **AND** TA Meeting **AND** CIS

Frame instance : **f1**

Sender	Receiver	Subject	Date	CC
Ng	Roy	TA meeting	10/15/97	Jason

Deposited in Folder : CIS, Roy
Document type (Frame template) : Memo

Frame instance : **f2**

Sender	Receiver	Subject	Date
Ng	Smith	TA meeting	10/15/97

Deposited in Folder : EE, Smith
Document type (Frame template) : Letter

Figure 6. The ranking example.

Step 2: Compute Sim(T_D, T_{qi})

The similarity between the document and the targeted document requested by the query is defined based upon the common attributes contained in the document types. For this example, since $T_D = T_{f1} = \{\text{Memo}\}$ and $T_{qi} = T_{q3} = \{\text{Memo}\}$ are of the same type. Therefore the similarity is 1.

Step 3: Compute Sim(Fi_D, $(A, V)_{qi}$)

Assume that there are 400 documents in the system. 100 of these documents contain the value *Roy* and 200 of them contain the value *TA Meeting*. Then the weight for the value *Roy* and *TA meeting* can be computed as follows:

$$W(Roy) = \log\left(\frac{400}{100}\right) + 1 = 1.60;$$

$$W(TA\,Meeting) = \log\left(\frac{400}{200}\right) + 1 = 1.30.$$

Table 1. The result of the computation of the similarity.

	q_1	q_2	q_3	q_4	**Similarity**
f1	1.46	1.96	2.46	2.32	8.20
f2	1.09	1.17	1.11	1.25	4.62

Similarly, assume $W(Ng) = 1.20$, $W(10/15/97) = 2$, $W(CIS) = 1.02$, $W(Smith) = 2.5$, and $W(Jason) = 2.6$. Then by using the Equation 5,

$$\begin{aligned} Sim(V_D, V_{qi}) &= \frac{\sum_{Vi\in V}(W(V_i))^2}{\sqrt{\sum_{V_j\in V_D}(W(V_j))^2 \times \sum_{V_k\in V_{qi}}(W(V_k))^2}} \\ &= \frac{(1.60)^2 + (1.30)^2}{\sqrt{\begin{matrix}((1.20)^2 + (1.60)^2 + (1.30)^2 + (2)^2 + (2.6)^2) \\ \times\ ((1.60)^2 + (1.30)^2)\end{matrix}}} \\ &= 0.51. \end{aligned}$$

Step 4: Compute Sim(Fi, qi)

$$\begin{aligned} \text{Sim(Fi, Q)} &= Sim(F_D, F_{qi}) + Sim(T_D, T_{qi}) + Sim(Fi_D, (A, V)_{qi}) \\ &= 0.95 + 1 + 0.51 = 2.46. \end{aligned}$$

We have shown the computation of the similarity for the And-clause **q3**. The similarity between a document associated with the frame instance **f1** and the query Q′ requires to compute the similarity for the And-clause q_3 and the rest of And-clauses. The complete result of the similarity between the And-clauses $\mathrm{q_i}$s and the frame instances **f1** and **f2** is given in Table 1. It shows that the document of **f1**, which has a higher score, is closer to the user's request.

5. Application Objects

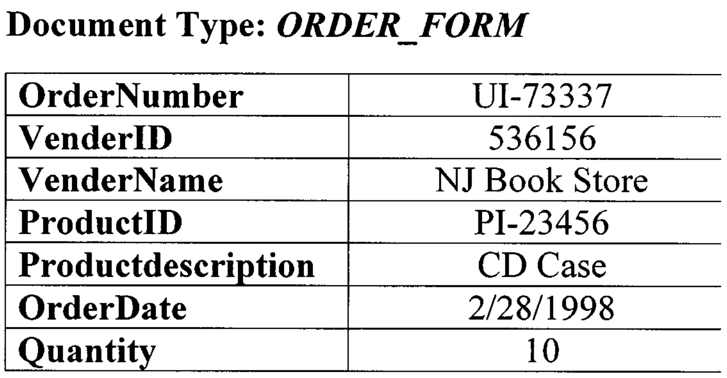
In [24, 27], the transformation of the original object network (O-Net) [12, 20, 21] into an operation network (OP-Net) was described. The transformed network is used as an underlying structure for browsing [24, 25, 27]. One of the goals of transforming the original object network into an operation network is to include the associations among the frame instances stored in the frame instance base and the objects stored in the system catalog [12, 21]. Another goal is to provide an environment which is better than (in the sense of preciseness and efficiency) the object network for browsing. Given a topic, an OP-Net is constructed to compose all relevant objects of four different object types, which are the folders, the templates, the frame attributes and their values.

The relevancy of the objects is defined by the predicates related to the given topic. These predicates qualify which frame instances can be appeared in the OP-Net. For each satisfied frame instance, the derived objects are the folders, which are the repositories of this frame instance in the folder organization (using the IsInFolder and HasFInstance relations), the template, which is the document type of its associated document (using the IsOfTemplate and HasInstance relations), the attributes, which characterize the properties of its document type (using the ContainsAttr and IsInInstance relations), and the values, which constitute the synopsis of its associated document (using the ContainsVal and IsInFinstance relations). The system catalog, which is the depository of these information (called the meta-data knowledge) and the information about the database itself, is central to the whole system and supports all the activities. Thus, the relations between the frame instances and objects in the system are captured in the system catalog [24].

As described in Section 3.1, the major components of the browser system are divided into three layers. At the first layer, there are various controllers, which monitor the entire browsing process. The second layer contains the topic interpreter, the request model builder, the network constructor, the ranking unit, and the search engine. As service providers, these components provide services to the controllers at the first layer. At the third layer, the storage management system consists of three major components: the system catalog, the frame instance base and the original document base. The frame instances are physically stored in the frame instance base with pointers pointing to their associated original documents which are stored in the document base. Various indexing techniques can be used when constructing the frame instance base [18, 19, 28]. A collection of frame instances could be partitioned into subcollections of frame instances based on their document types and additional premises. The folders of the folder organization are simply the depositories of the pointers, pointing at the frame instances, which are located in the frame instance base. Thus, the notion of the dual models—the folder organization and the document type hierarchy, is applied to construct an indexing organization of the original document base for precise and fast document retrieval.

The basic structures of the components in the third layer create an open environment for different application domains. For example, in an office environment, the documents are used and processed by certain departments and therefore they are closely associated with the application domains, such as the departments. The purchasing department may use a standard *order form* to place different orders for purchasing various goods. A frame instance of a purchasing order form is given in Figure 7. It should be noted that the order form completed for ordering various goods could be treated as a frame instance. In fact, the order forms can be referred to as the template of a document type, called *ORDER_FORM*. In this case, there is a relation between the department and the document type. For example, the purchasing department has the order forms, and the purchasing department processes the order forms for purchasing various goods. After the filing process, the relations between the department and the folders (the depositories of these frame instances) are also created. For example, the purchasing department owns, and therefore has an access to, these folders which are the depositories of these order forms. These new relations encapsulate additional semantics from the application domain.

In the following discussion, the objects of the four original object types are referred to



Document Type: *ORDER_FORM*

OrderNumber	UI-73337
VenderID	536156
VenderName	NJ Book Store
ProductID	PI-23456
Productdescription	CD Case
OrderDate	2/28/1998
Quantity	10

Figure 7. A frame instance of a purchasing order form.

as the *system objects*, while the objects of different application domains are referred to as *application objects*. The application objects themselves can be described as a network. For simplicity, assume that the layer of application objects can only be attached directly to the folder and the frame template layers. In order to capture the relationships between the document and the application objects, the original OP-Net is extended to include an *application object layer*. Figure 8 depicts an OP-Net with an application domain layer, such as the *department* for this case. The original OP-Net is referred to as the *system object layer*. By introducing the application object layer, the definition of the signature of a document can be generalized to include the application objects.

Definition 5: Signature for a document (contains the application oriented object layers). The signature of a document D, Sig(D) = $[I_S, I_A, Fi]$, where

1. $I_S = \{F_D, T_D\}$, where
   - F_D is the set of folders where D is deposited, and
   - T_D is the document type of D; and
2. $I_A = \{O_i | O_i$ is the set of application object types associated with the document D, $1 \leq i \leq n\}$, where n is the number of the application oriented object types attached to the original system; and
3. Fi is the frame instance corresponding to *D*.

In brief, the I_S and the I_A represent the properties of a document regarding to system objects and application objects, respectively. For example, for the frame instance *d* given in Figure 7, the signature of the document *d* can be represented as follows:

Sig(**d**) = [{{CIS}, {ORDER_FORM}}, {{**Purchasing**}}, {(OrderNumber,

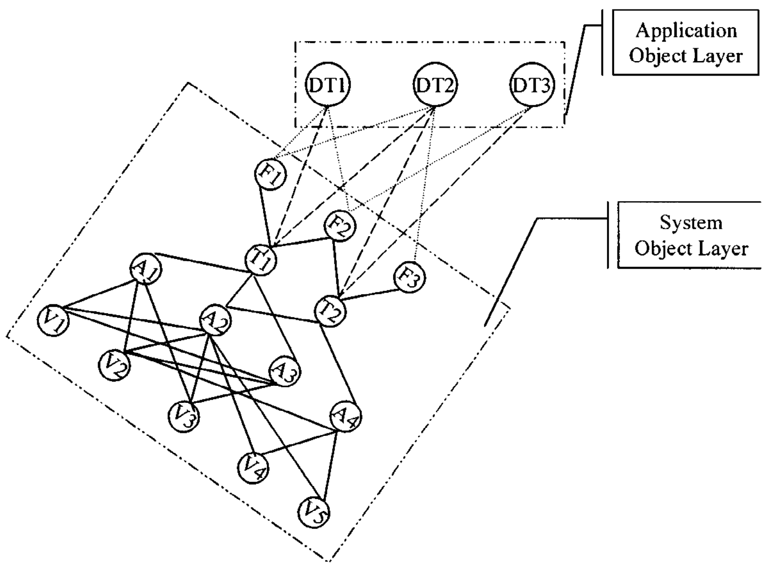


Figure 8. The OP-Net after attaching the application object layer.

UI – 73337), (VenderID, 536156), (VenderName, NJBookStore), (ProductDescription, CDCase), (OrderDate, 2/28/1998), (Quantity, 10)}].

Likewise, the definitions of the signature of an And-clause and the signature of a query can be generalized to include the application domains in the same way.

Let $\mathrm{Sim}(\mathrm{DT}_\mathrm{D}, \mathrm{DT}_\mathrm{qi})$ be the similarity between two sets of application objects, such as the object *department*, which has an access to the document D, and the other is specified in an And-clause qi. Without losing any generality. Let DT_D and DT_qi be the sets of application objects which are related the document *D* and specified in an And-clause qi, respectively. That is,

$$\begin{aligned}\mathrm{DT}_\mathrm{qi} &= \{\mathrm{DT}_\mathrm{Qil} \mid \mathrm{DT}_\mathrm{Qil} \text{ is a department}, 1 \le \mathrm{l} \le \mathrm{n}\}, \text{ and}\\ \mathrm{DT}_\mathrm{D} &= \{\mathrm{DT}_\mathrm{Dj} \mid \mathrm{DT}_\mathrm{Di} \text{ is a department}, 1 \le \mathrm{j} \le \mathrm{m}\},\end{aligned}$$

where n and m are the number of the folders in DT_D and DT_qi respectively. Let $\mathrm{FI}(\mathrm{DT}_\mathrm{Qil})$ represent the frame instance set associated with an application object DT_Qil. Then FI_D and FI_qi, the sets of frame instances associated with the sets of application objects DT_D and DT_qi, respectively, can be computed as follows:

$$\mathrm{FI}_\mathrm{qi} = \mathrm{FI}\left(\bigcup_{\mathrm{l}=1}^{\mathrm{n}} \mathrm{DT}_\mathrm{Qil}\right), \text{ and}$$

$$\mathrm{FI_D} = \mathrm{FI}\left(\bigcup_{i=1}^{m} \mathrm{DT_{Di}}\right).$$

When $\mathrm{DT_{qi}} \neq \emptyset$, the $\mathrm{Sim(DT_D, DT_{qi})}$ can be computed by the following equation.

$$Sim(DT_D, DT_{qi}) = \frac{2\#(FI(DT_D \cap DT_{qi}))}{\#(FI(DT_D)) + \#(FI(DT_{qi}))}.$$

In the remaining section, we shall discuss the tasks required for creating application objects at the application object layer of the extended OP-Net.

1. Identify the application objects. An application object is an object at the application object layer of the extended OP-Net, if there is a document in the original document base, which has a relationship with the object. This application object is said to be document-oriented.

2. Convert the application objects into their corresponding frame instance repositories. Since the application objects are document-oriented, the relation between application objects and documents can be formalized as the conditions for qualifying the associated frame instances. Thus, the application objects are converted into their corresponding frame instance repositories.

3. Create new index maps in the system catalog. The application object is similar to the folder object in the sense that they all provide the logical view of the physical frame instance base. The system catalog requires three new index maps between the application objects and the frame instances, between the application objects and the folders, and between the application objects and the templates. Figure 8 depicts the conceptual view of an OP-Net, consisting of the system object layer and the application object layer.

6. Conclusion and the Future Research

In this paper, a document retrieval model for TEXPROS is described. In general, TEXPROS allows users, firstly to retrieve documents and information through identifying their frame templates (the document types) and to match the given values against the values appeared in the frame instances of a single class or several classes (each class is represented by a frame templates). Secondly, it allows users to manipulate and query folders, and to perform folder-at-a-time operations. Thirdly, TEXPROS allows users to browse through the folders containing frame instances of a specified document type, and the contents of frame instances of a particular type contained in a specified folder. Browsing the folders and the contents of frame instances help users reformulate queries dynamically. Fourthly, if the user only has a rough idea or he can only describe partially the requested documents, he may perform the concept-based retrieval. In the concept-based retrieval, there is no clear distinction between documents that qualify the specified condition and those that do not; some documents are more relevant, while others are less so. For such "fuzzy" types of queries, TEXPROS always returns a list of documents, ranked according to the degree of their relevance to the query.

With the realm of this strategy of retrieving documents, in TEXPROS, we employ a three level architecture of a document repository to store documents. At the first level, the storage contains orginal documents (called the original document base). A physical storage containing frame instances is at the second level (called the frame instance base). Analogous to the inverted indexing, each frame instance has a pointer, pointing to its corresponding original document, besides which, it contains the most relevant information of the document, in a precise and succinct manner, pertinent to the significance of the user. The third level is the folder organization. Each folder is a virtual repository for a set of frame instances, which is called the logical storage for the frame instances. It is called a virtual repository because we only store pointers, pointing to the frame instances at the second level. We adopt this multilevel access structure as the system repository architecture to support direct access to documents, which requires retrieving their corresponding frame instances through the use of specific information, such as attributes and document type, from the document type hierarchy, and the folders containing these frame instances [28].

Central to the query processing system is the system catalog. The catalog contains the metadata (i.e., the description of the document type hierarchy and the folder organization) of the database (i.e., the frame instances of documents) and a thesaurus. The thesaurus contains synonymous terms of a term that are relevant to the user, terms that are semantically equivalent, and correspondences between terms and the index terms of folder, template or attribute name types actually residing in the database. Since the uniform representation of the system catalog and the database itself is adopted, the user can retrieve information from the system catalog using the same query format to retrieve any general frame instances in the database. We then employ an object network (O-Net) to describe the snapshot of a subset of information containing in the system catalog. It includes schema elements, data elements and the dual modeling relationships. The schema elements give the description of the folder organization and document type hierarchy. This description includes the frame instances, the folders, the frame templates, and the attributes at different levels. They are connected using the relationships: is-place-in, has-type, is-identify-by, and is-a-combination-of. The data elements are the description of the database itself in terms of pairs of the attributes (of frame templates) and their values. The attributes and their values are connected using the relationships: includes and is-A-of-FT-in-f-having-B. Thus, this object network provides a path for looking up relevant information from the related system frame instances [12, 20, 21]. For example, given a vague query “TOPIC John Smith,” once John Smith is identified as a folder name, the object network governs the search for the parent and the children folders of the folder John Smith.

As shown in Figure 4, a typical browsing process consists of the following phases: the topic interpretation and rewriting, the network construction, the topic refining, the exploring and result examining. The browsing process proceeds interactively as follows. Upon the arrival of a user’s request (or called topic), which could be a vague query, a typical browsing cycle begins with the network to derive from an O-Net, with respect to the given topic, an operation network (OP-Net), which is an underlying structure for browsing. A major difference between the OP-Net and O-Net is that, associated with each node of the

OP-Net, there are frame instances which satisfy the given queries. However, after consulting the system catalog, the topic interpretation and rewriting could give a fully detailed representation of the user's queries in a normal form. After reviewing the OP-Net, the user refines the original topic if he/she knows how to modify it. If the user requires additional knowledge for stating a more precise topic, the user can enter the exploring phase to issue queries within the realm of the current OP-Net. Given the query, the system constructs an exploring network (E-Net, a subnet of an OP-Net or E-Net) from an existing OP-Net or E-Net. Without the exploring process, the refining process requires reconstructing the OP-Net for each modification of the original queries. For example, if the original query *(CIS AND John)* is refined as *(CIS AND John AND Letter) OR (CIS AND John AND Memo)*, the system well have to reconstruct the OP-Net from the O-Net for each refinement during the topic refining process. On the contrary, the exploring process constructs the E-Net from the OP-Net (or E-Net) of the original query. Finally, the system knowledge from the system catalog and the frame instances are returned by the system during the browsing session.

In this paper, associated with each node of the OP-Net (or E-Net), there is a frame instance repository. It is defined to associate the frame instances in the frame instance base with an object in the system. The frame instance repositories and the inter-relations between them are integrated by the OP-Net. Since the frame instance repository is a predicate-based storage in the sense that the contents of the frame instance repository depend on the predicate of the repository. The characteristics of the system object and the query issued by the user will be translated into predicates. Since the predicate governs the contents of the repository, when users change the query, the contents of the repository are changed accordingly. A repository will be removed from the original OP-Net if it is empty. In other words, the OP-Net is dynamically constructed according to user's query. The novelty of the OP-Net can be summarized as follows:

1. The OP-Net is dynamically constructed. When the user refined the original query by setting stricter conditions, the new OP-Net can be derived from the original one. This improves the performance of constructing the OP-Net from the O-Net.

2. The presentation of OP-Net provides the summary of the result. The information retrieval systems which deal with the vague query, share a common characteristic: the returned result has a higher rate of acceptance if the issued query is well specified. However, in the earlier stage of the retrieval process, the size of the returned result for the vague query is usually extremely large. The OP-Net distributes the result into frame instance repositories. These frame instance repositories provide hints for users how to continue to proceed the retrieval process.

The signatures of documents and queries are defined by taking the dual models—the folder organization and the document type hierarchy, into consideration. The signature is another way for defining the representatives of documents and queries. The signature captures the synopsis of the content and the characteristics of the document. Since the signatures of the document and the query are structurally identical, the similarity between the returned documents and the query specification of the documents to be retrieved can be computed. The existing ranking models, which are basically based on the frequency of term occurrences and the weight of the terms, can be incorporated into our proposed ranking model, since these ranking mechanisms can be used for computing the similarity revealed by the *value* object in the system.

The retrieval model proposed in this paper provides an open environment. Different application objects of various application domains can be incorporated into the existing OP-Net. For example, by integrating the infrastructure of a corporate organization into the existing OP-Net, the system could reduce the size of the amount of the returned documents based upon the given query. The OP-Net is therefore generalized to include application objects in the application object layer, while preserving the other four object type layers (called the system object layer). The existing browsing mechanism can still be used without any changes. The system becomes more flexible by allowing users to customize their system in meeting their needs according to their application domains.

The query language is an interface between users and the system. Currently the system supports only the query, which contains terms connected by the logic operators AND, OR and NOT. This query language is easy to use. However, it is difficult to use this simple language to write queries, when users know more information about what they want. For example, if the user knows the documents are in a folder, say the *CIS* folder, there is no way to specify it. The user can only issue a query containing *CIS* and let the system to figure out that *CIS* is a folder. The drawback is that, in the system, if *CIS* is also a value that appears in some frame instances, the system will return a OP-Net which is corresponding to *Folder(CIS) OR Value(CIS)*, which probably contains some irrelevant documents caused by the *Value(CIS)*. The problem is even more severe when a range is applied to a numerical type of values in the document. For example, the user might want to find all the memos, which has sent out during this month. In this case, a powerful language is required for specifying this condition. The query language is needed for allowing users to express more precisely the information regarding to what they want. Finally, experiments will be conducted to observe whether the proposed ranking-based model with generalized OP-Net could achieve higher recall and precision rate, in comparison with the previous model with OP-Net.

References

1. A. Bookstein, "A comparison of two systems of weighted Boolean retrieval." *Journal of the American Society for Information Science*, pp. 275–279, July 1981.
2. G. Salton, A. Wong and C. S. Yang, "A vector space model for automatic indexing." *Communications of the ACM* 18(11), pp. 613–620, November 1975.
3. K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval." *Journal of Documentation* 28(1), pp. 11–21, March 1972.
4. C. T. Yu and G. Salton, "Effective information retrieval using term accuracy." *Communications of the ACM* 20(3), pp. 135–142, March 1977.
5. W. B. Croft, "Boolean queries and term dependencies in probabilistic retrieval." *Journal of the American Society for Information Science* 37(2), 1986.
6. I. J. Aalbersberg, "A document retrieval model based on term frequency ranks." *Research and Development in Information Retrieval*, pp. 163–172, 1994.
7. N. Fuhr, "Integration of probabilistic fact and text retrieval," in *Proceedings of the 15th Annual International ACM SIGIR Conference*, Denmark, June 1992, pp. 211–222.
8. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill: New York, 1983.
9. G. K. Zipf, *Human Behavior and the Principle of Least Effort*. Addison-Wesley: Reading, MA, 1949.

10. M. Persin, “Document filtering for fast ranking.” *Research and Development in Information Retrieval*, pp. 339–348, 1994.
11. C. J. Van Rijsbergen, *Information Retrieval*. Butterworths: Boston, MA, 1979.
12. Q. Liu and P. A. Ng, *Document Processing and Retrieval: TEXPROS*. Kluwer Academic Publishers: Norwell, MA, 1996.
13. J. T. L. Wang and P. A. Ng, “TEXPROS: An intelligent document processing system.” *International Journal of Software Engineering and Knowledge Engineering* 15(4), pp. 171–196, April 1992.
14. M. Sneoeck and G. Dedene. “Generalization/specification and role in object oriented conceptual modeling.” *Data and Knowledge Engineering* 19(2), pp. 171–195, June 1966.
15. C. Wei, J. T. L. Wang, X. Hao and P. A. Ng, “Inductive learning and knowledge representation for document classification: The TEXPROS approach,” in *Proceedings of 3rd International Conference on Systems Integration*, Sao Paulo, SP, Brazil, August 1994, pp. 1166-1175.
16. X. Hao, “Document Classification and Information Extraction.” Ph.D. dissertation, Department of Computer and Information Science, New Jersey Institute of Technology. UMI Press, 1995.
17. Z. Zhu, J. A. McHugh and P. A. Ng, “A predicate driven document filing system.” *Journal of Systems Integration* 6(3), pp. 373–403, 1996.
18. X. Fan, Q. Liu and P. A. Ng, “A multimedia document filing system,” in *Proceedings of the International Conference on Multimedia Computing and Systems*, Ottawa, Ontario, Canada, pp. 492–499.
19. X. Fan, Q. Liu and P. A. Ng, “Knowledge-based document filing: TEXPROS approach,” in *Proceedings of the 13th International Conference on Advanced Science and Technology in Conjunction with the 2nd International Conference on Multimedia Information Systems*, Schaumburg, Illinois, USA, pp. 58–67.
20. Q. Liu and P. A. Ng, “A browser of supporting vague query processing in an office document system.” *Journal of Systems Integration* 5(1), pp. 61–82, 1995.
21. Q. Liu, “An office document retrieval system with the capability of processing incomplete and vague queries.” Ph.D. dissertation, Department of Computer and Information Science, New Jersey Institute of Technology. UMI Press, 1994.
22. A. Motro, “Browsing in a loosely structured database,” In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, Boston, MA, June 1984, pp. 197–207.
23. A. Motro, “BAROQUE: A browser for relational databases.” *ACM Transactions on Office Information Systems* 4(2), pp. 164–181, April 1986.
24. C. Y. Wang, Q. Liu and P. A. Ng, “Intelligent browser for TEXPROS,” in *ISATED Proceedings of International Conference on Intelligent Information Systems (IIS’ 97)* H. Adeli, ed., IEEE Computer Society Press, pp. 388–398, Dec 8–10, 1997.
25. C. Y. Wang, Q. Liu and P. A. Ng, “Browsing in an information repository,” in *Proceeding of 2nd World Conference on Integrated Design and Process Technology*, M. M. Tanik, etc., eds., IDPT-Vol 2, pp. 48–56, 1996.
26. Q. Kong and G. Chen, “On deductive databases with incomplete information.” *ACM Transactions on Information Systems* 13(3), pp. 354–369, July 1995.
27. C. Y. Wang, “The Intelligent Browser for TEXPROS.” Ph.D. Dissertation. Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ. UMI Press, May 1998.
28. X. Fan, “Knowledge-Based Document Filing for TEXPROS.” Ph.D. Dissertation. Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ. UMI Press, May 1998.