KNOWLEDGE REPRESENTATION AND ACQUISITION IN SUPPORTING DOCUMENT FILING AND RETRIEVAL

Xien Fan
Department of Computer Science, College of Staten Island
City University of New York,
Staten Island, NY 10314
USA

Fang Sheng
Software House International, Inc.
2 River view Road,
Somerset, NJ 08873
USA

Gary Thomas
University of Missouri at Rolla
Rolla, Missouri 45409,
USA

Peter A. Ng
Department of Computer Science, University of Nebraska-Omaha
Omaha, NE 68182,
USA

ABSTRACT

The effectiveness and efficiency have been the major challenges in the document and information retrieval field. Personal information systems can be built based on user's special preference and interests on documents, which can be used to dramatically improve the efficiency and accuracy of document retrieval. This paper presents a flexible dual-model framework that gets user involved in document modeling. Being familiar with the document model, user can specify complex and precise queries. A knowledge-based approach to document filing and retrieval is presented. To support such a flexible document model, a knowledge representation model is proposed to store the domain knowledge. Machine-learning algorithms are proposed to acquire the domain knowledge.

KEY WORDS
Document retrieval, knowledge representation, machine learning

1. INTRODUCTION

Nowadays hundreds of thousands of documents are processed in various office environments. With the booming of Internet technologies, documents in electronic formats have been increased dramatically, which brings huge impacts on offices from government to resident home. Effective and efficient document storing and retrieving is becoming more challenging and difficult. A considerable amount of research work has been focused on the automatic document and information management [1][2][3][4][5][6][7].

Document retrieval in an information system is most often accomplished through keyword search. The common technique behind keyword search is indexing. The major drawback of such a search technique is its lack of effectiveness and accuracy. It is very common that a typical keyword search over the Internet identifies hundreds or even thousands of documents as the potentially desired records. However, often few of them are relevant to users' interests. This sometimes makes a search ineffective, and even worse, the results obtained from the search could be inapplicable, because it may not be practical or acceptable to users to examine the returned documents one by one.

A solution for dealing with such a chaotic phenomenon is ranking the returned documents according to their degree of significance or other factors. Many search tools can be used for ranking the return documents based on pre-defined criteria. However, to achieve an acceptable
A dual modeling approach is employed for describing, classifying, filing and retrieving documents [8]. This dual-model consists of two hierarchies: a document type hierarchy which describes the conceptual structure of the documents, and a folder organization which is used as the filing model. By identifying common properties for each document class, documents are classified into different classes. Each document class is represented by a frame template which describes the common properties in terms of attributes and is referred to as the document type (or simply type).

Given a new document, the classification subsystem [9] is responsible for finding its best fitting type based on the analysis of the layout and logical structure of the document. The extraction subsystem [10] will then summarize the document from the viewpoint of its frame templates, and yields a synopsis of the document which is called a frame instance. Frame instances, which represent the original documents, are then processed in filing and retrieval.

The frame instances of various types are deposited in folders. Folders are heterogeneous repositories and are organized to form a folder organization, which is one of the common ways of organizing and storing documents for their retrieval in an office. The folder organization is defined by a user based upon his/her 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. Each folder has a user defined criterion for governing the automatic document filing. The folder organization captures the user's knowledge about document organization. It provides a flexible model which can fit into different application domains since it is defined by the user. The folder organization is an underlying structure of an agent-based filing architecture. Each folder is associated with a filing agent. In the remainder of this section, we shall briefly review the formal specification of the folder organization. Details can be found in [11].

**Definition 1.** (Folder Organization) A folder organization is a two tuple, $\mathcal{FO}(G, \Delta) = \{G(V, E), \Delta\}$, where

1. $G(V, E)$ (also denoted as $G(\mathcal{FO})$) is a rooted directed acyclic graph, and
   - each vertex in $V(G)$ corresponds to a folder;
   - each edge $(f_i, f_j)$ defines a link from $f_i$ to $f_j$, which can be an OR-link or an AND-link; and
2. $\Delta = \{\delta_f | f \in V(G)\}$ is a set of predicates, where $\delta_f$ is the predicate of a folder $f$.

For the sake of clarity, we shall refer the local predicate of a folder $f$ to the predicate of the folder $f$, which is distinguished from the notion of a global predicate.
Definition 2. (OR-link) A link \((f_i, f_j)\) is called OR-link if for any frame instance \(\omega\), \(\delta_j(\omega) \land (\omega \in f_i) \Rightarrow \omega \in f_j\)

Definition 3. (AND-link) A link \((f_i, f)\) \((1 \leq i \leq n)\) is called AND-link if it has the following properties:
1. For any frame instance \(\omega\), \(\delta_i(\omega) \land (\omega \in f_i) \land \cdots \land (\omega \in f_n) \Rightarrow \omega \in f\)
2. \((f_i, f), (f_i, f), \ldots, (f_i, f)\) are the only incoming AND-links of \(f\).

Definition 4. (Global Predicate) Let \(\varphi_f\) denote the global predicate of any folder \(f\). According to the definition of OR-link and AND-link, global predicates are given as following:
1. The global predicate of the root folder is TRUE.
2. Let \(f_1, f_2, \ldots, f_n, f'_1, f'_2, \ldots, f'_n\) be all the parent folders of \(f\) where \((f_i, f)\) \((1 \leq i \leq n)\) is OR-link and \((f'_i, f)\) \((1 \leq i \leq m)\) is AND-link. Then

\[
\varphi_f = \delta_f \land \left( \sum_{i=1}^{n} \varphi_i \lor \prod_{j=1}^{m} \varphi'_j \right)
\]

Intuitively, the definition of OR-link and AND-link define the semantics of a folder organization, and determine how a frame instance is filed into the folder organization. The global predicate governs the content of a folder.

3. PREDICATE-BASED REPRESENTATION OF DOCUMENTS

To support knowledge-based document filing and retrieval, a language is needed for users to make assertions on documents. It is used as criteria for governing document filing, or can be used for specifying queries. Using the same language for document filing criteria and document retrieval makes it possible for the search engine to take advantage of how documents are organized and quickly reduce the search space to specific folders. We employed predicate as the language for users to specify their knowledge about documents. The user's knowledge about documents can be direct content or indirect content of documents. The direct content of documents that is in users interest is carried by frame instances, which includes keywords and structural information. Frame templates define the conceptual structure of documents. Predicates allow users to match keywords against any attributes of the underline frame templates. Very often, a user may want to specify knowledge that is not directly contained in documents. For example, consider a predicate specifying that the document is discussing specific accidents that are caused by vehicles. In this case, accident is an object that is related to the article and cause is considered to be the property of the accident and may not be obtained directly from the document. The predicate specifies the semantic and behavioral information of an object, which is not contained in original document but can be derived from pre-constructed domain knowledge base.

Predicates are made directly or indirectly on frame instances, which contain conceptual information of the documents. Attributes are used as identifiers in accessing information from frame instances. Predicates are statements of objects. Only two kinds of objects can appear in predicates. One is the frame instances. The other is objects which are somehow related to the frame instances, i.e appearing in the values of some attributes.

Definition 5. A predicate clause has the form \(g(o, b)\) where
1. \(o\) is an object whose property will be specified by the predicate clause. \(o\) is either a frame instance (denoted as \(\omega\)) or an attribute of document that relates an object to the document.
2. \(g\) is the name of the predicate clause and can be an attribute of object \(o\);
3. \(b\) is either a value or a variable.

Intuitively, predicate clauses are used to specify some characteristics of a frame instance or the property of an object that is related to a frame instance. For example, Date(o, 4/25/96) denotes that the frame instance \(\omega\) is dated 4/25/96. Cause(Accident, vehicle) denotes that the document contains data for accidents caused by vehicle.

A predicate clause is called a goal predicate clause if its second parameter is a value. An assignment predicate clause is the one whose second parameter is a variable. A goal predicate clause is a statement. It is either true or false. While an assignment predicate clause is to assign a value, which makes the predicate clause true, to its second parameter. For example, Class(Product, DBMS) represents that the document is talking about specific DBMS product. Affiliation(Author, x) will assign the affiliation of the author to the variable \(x\).

Definition 6. (Predicate Constraint) A predicate constraint \(\theta\) is a relation between variables and values, where \(a\) and \(b\) are either a variable or a value, and \(\theta \in \{=, \neq, \in, \exists, <, >, \leq, \geq\}\).

Definition 7. (Atomic predicate) An atomic predicate is either a goal predicate clause or a n-tuple \((P_1, P_2, \ldots, P_n)\), where \(P_i, 1 \leq i \leq n\), is either an assignment predicate clause or a predicate constraint,
Definition 8. (Predicate)
1. An atomic predicate is a predicate.
2. If \( P \) is a predicate, then \( \neg P \) is a predicate.
3. If \( P \) and \( Q \) are predicates, then \( (P \lor Q) \) and \( (P \land Q) \) are also predicates.

4. DOCUMENT RETRIEVAL

With the dual-model and the predicate-based representation of documents, the system can support a multi-level retrieval strategy. This section reviews the basic idea of this retrieval strategy. Details can be found in [11]. The browsing subsystem is introduced in [12].

Queries are specified in terms of predicates. The retrieval process starts with knowledge-based query pre-processing. It narrows down the searching space using the information contained in the query, with the help of the knowledge base. The algorithm is given in [11]. Since the related frame instances, which are collected in groups (called folders) based on the user predefined criteria for the folders, appear closely together, browsing in a collection of frame instances can be conducted effectively. The frame instance collection search strategy uses information obtained from the folder organization and the document type hierarchy. Given a query formula, the document type \( T \) of the retrieved documents will be identified using specific information contained in the system catalog. This will restrict the searching on a particular document type. The searching space can also be reduced to a particular folder \( f \), which contains all relevant documents. This folder can be identified by finding the smallest folder whose global predicate can be derived from the query formula. That is, the searching spaces can be reduced to \( T \land f \), focusing on frame instances of a particular type within a particular folder.

After the pre-processing, queries will be processed by applying content-based search on the generated small set of frame instances. This query process is text-based. It depends partially on an exact match between the values of the content identifiers (i.e., the structured part of frame instances) and the attribute values used in the query formulations. Information retrieval decisions may also depend on the contents of the unstructured part of frame instances. However, the size of the unstructured part of frame instances is considerably small in comparison with the contents of the corresponding original documents. For this case, various classical text retrieval methods, such as inverted indexing, clustering searching, etc. are applicable.

It is possible that the searching may have to go through the contents of original documents although this is what we are trying to avoid. This could happen because a frame instance contains only the synopsis of its corresponding original document. Then, various methods, including content-based multimedia information retrieval, could be applied.

5. THE KNOWLEDGE BASE

The knowledge used in evaluating predicates consists of two known facts, and knowledge about application domain. The known facts include properties of documents and related objects. Frame instances contain information extracted from documents based on user's preferences. Properties of objects that are related to documents are stored in object base. The knowledge of application domain is stored in domain knowledge base. It describes the context in which properties of objects can be specified using predicates.

5.1. OBJECT BASE

Object base allows users to make assertion about documents beyond their direct content, because the knowledge contained in object base is usually not contained in documents. The object base consists of a set of object pages. Each object page is associated with one object. The object base contains different knowledge in different application domains, and therefore it is domain dependent. Knowledge or facts about one particular object is encapsulated into one object page in the form of a number of (attribute, values) pairs. Attributes in an object page are called the property names. They can be multi-valued. Each property name, together with one of its value, defines a property of the object.

5.2. DOMAIN KNOWLEDGE BASE

An application domain may be divided into sub-domains, and each of the sub-domains may have, in turn, sub-domains. A domain defines the context in which objects that are related to documents can be described. Each of the domains may have various properties of interest to describe objects. They can be organized as domain organization and property relation.

Definition 9. (Domain Organization) Given a domain \( D_0 \), a domain organization \( \mathcal{DO}(D_0) = \mathcal{D}(\mathcal{V}, \mathcal{E}) \), is a rooted, directed tree where:
1. Each vertex in \( \mathcal{V}(D_0) \) (also denoted as \( \mathcal{V}(\mathcal{DO}) \)) corresponds to a domain.
2. Each edge \( (\mathcal{D}_i, \mathcal{D}_j) \) denotes that \( \mathcal{D}_j \) is a sub-domain of \( \mathcal{D}_i \).

Definition 10. (Property Relation) Given a \( \mathcal{DO}(D_0) = \mathcal{D}(\mathcal{V}, \mathcal{E}) \), a property relation of a domain \( \mathcal{D}_i \) with respect
to \( DO(D_0) \), \( PR_{DO(D_0)}(D) = D(N,L) \), is a rooted, directed tree where:
1. For the domain \( D_0 \), the corresponding vertex is the rooted vertex of the rooted, directed tree \( D(N,L) \), where \( D_0 \) is not equal to \( D_0 \) and \( D_0 \) is a subdomain of a vertex in \( V(D_0) \).
2. Each non-rooted vertex in \( M(D) \) (also denoted as \( M(PR_{DO(D_0)}(D))) \) corresponds to a property.
3. Each directed, hollow edge \((D_0,P)\) in \( L(D_0) \) denotes that \( P_i \) is a property of the domain \( D_0 \).
4. Each directed edge \((P_i,P_n)\) denotes that property \( P_n \) implies property \( P_i \).

**Definition 11.** (Domain Knowledge) Domain knowledge is a set of two-tuple, \( DK = \{ (\{D_0 \}, \{V_i \}, \{E\}) \mid \forall D \in V(D_0) \land (D \neq D_0) \} \), which is a set of property relations defined for the subdomains of the domain organization of \( D_0 \) (i.e., \( DO(D_0) \)).

Let \( D \in V(DO) \) (1 ≤ i ≤ n), be a domain in the domain organization \( DO(D_0) \). \( D_0 \), \( D_{n-1} \), \ldots, \( D_0 \) is the full name of domain \( D_0 \) if and only if \((D_0, D_1), (D_1, D_2), \ldots, (D_{n-1}, D_n)\) are the directed edges in \( DO(D_0) \) which means that \( D_i \) is the super domain of \( D_{i+1} \), for 0 ≤ i ≤ n-1. Values of a domain are called domain instances. (For example, CIS is a domain instance of the domain Department.) Let \( D_{i}, D_{i-1}, \ldots, D_0 \) be the full name of domain \( D_i \), and \( D_n, D_{n-1}, \ldots, D_0 \) be domain instances of \( D_x \), and \( D_{n-1}, D_{n-2}, \ldots, D_0 \) respectively. Then \( D_i : D_{i-1} : \ldots : D_0 \) is called the full name of domain instance \( D_i \). The full name of a domain (or domain instance) is the precise description of the domain (or domain instance).

### 6. KNOWLEDGE ACQUISITION

To support flexible document model, a learning agent is developed for building the object base and acquiring domain knowledge. This agent is a learning process over the life time of the system. And knowledge acquired in one application cycle may be reused in another as long as the two cycles have the similar application domain.

#### 6.1. OBTAINING DOMAIN KNOWLEDGE

As we discussed early, frame instances are organized based on a user-defined folder organization. The folder organization tells what is the user's interest regarding different folder organization. The folder organization can be extracted from the folder organization.

Let \( DO \) be the domain organization being created. Let \( D \in DO \) denote that \( D \) is a domain, \( DX(D) \) denote that \( D \) is a domain instance of domain \( D \), \( D \supset D_j \) denote that domain \( D_j \) is a sub-domain of \( D \), \( (a, v) \in PR(D) \) denote that \( (a, v) \) is a property in domain \( D \), and \( (a, v) \Rightarrow (p, u) \) denote that property \( (a, v) \) implies property \( (p, u) \). Let \( P_i \) \((O_o, v_o) \Rightarrow P_2 (O_o, v_0) \) denote that \( P_1 (O_o, v_0) \) and \( P_2 (O_o, v_0) \), which are predicate clauses, are local predicates of folder \( f \) and \( g \) respectively, where \( f \) is the only parent of \( g \).

The following learning rules are used by the domain agent to acquire the domain knowledge.

1. If \( DX(o, d) \rightarrow P(o, a \cdot v \cdot d) \) then \( D \in DO(D) \) and \( DX(D) \).
2. If \( DX(o_1, d_1) \rightarrow DX(o_2, d_2 \cdot d_1) \) and \( D_2 \in DO(D) \), then \( DX(D_2) \cdot d_1 \) and \( D_2 \supset D \).
3. If \( P(o, u \cdot d) \rightarrow DX(o, a \cdot v \cdot d) \) and \( DX(D) \), then \( (a, v), (p, u) \in PR(D) \) and \( (a, v) \Rightarrow (p, u) \).
4. If \( (a, v), (p, u) \in PR(D) \) and \( u \) is narrow term of \( v \), then \( (a, v) \Rightarrow (p, u) \).

### 6.2. BUILDING OBJECT BASE

As we see in the previous section, the domain knowledge can be extracted from the folder organization. But the knowledge about the objects that are involved in specifying predicates can not be acquired directly from the folder organization. For collecting facts about these objects, the agent must know what, how and where to learn. We define topic as the basic concept of learning.

**Definition 12.** (topic) A Topic is a four-tuple \((o, a, v, d)\), where \( o \) is an object, \((a, v) \) is a property, \( d \) is a domain instance. A Topic \((o, a, v, d)\) is called a fact if predicate \( d(o, v \cdot d) \) is true.

**Generating Learning Topics**

A learning topic is a question which inquires any part of a topic. For example, \(?(o, a, v, d) \) is asking who has property \((a, v) \) in domain instance \( d \), while \(?(o, a, v, d) \) is asking if \((o, a, v, d) \) is a fact. There are three rules for generating learning topics.

1. If \( P(o, v \cdot d) \) is a predicate, then \(?(o, a, v, d) \) is a learning topic, also called YN learning topic.
2. If \( DX(D) \) and \((a, v) \in PR(D) \), then \(?(o, a, v, d) \) is a learning topic, also called WH learning topic, for each domain instance \( d \) of domain \( D \).
3. A learning topic \((?o, a, v, d) \) has higher priority than \(?o, p, u, d) \) if \( DX(D) \) and \(((p, u), (a, v)) \) is a link in \( PR(D) \).

**Learning Rules**

Learning rules are used by the learning agent to collect facts. The "yes" answer of a YN learning topic \(?o, a, v, d) \), which is learned from the filing process or from the user, will generate the fact \((o, a, v, d) \). For example, a predicate clause is evaluated to be true by the inference engine, or a
frame instance is filed into a folder by user which implies that the local predicates along the filing path are true.

The WH learning topics are raised to the users in descending order of their priorities. The user interface allows user to specify where the needed facts can be found. If objects other than frame instances appear in predicates in a folder organization, it means that user wants to organize the documents based on the knowledge of the objects which are somehow related to the documents. This implies that these objects are in user's interest. So it is reasonable to assume that user can provide this information or train the system how to get the information. We identified two kinds of sources that a user can provide. The first source is documents. Some documents contain facts about the evolving objects. For instance, a product release document contains the basic information of the product and an accident report form contains the basic information of the accident. For other documents that are talking about specific product or accident, this basic information is the knowledge that may not be contained directly in the documents.

It is also possible that the knowledge of these objects may be maintained in a database. For example, a human resource database contains information about people in a specific organization. A product vender stores its products in database. A user is allowed to write a query in response to a learning topic.

7. CONCLUSION

A dual modeling approach and a predicate-based representation of documents are employed to support the conceptual-based information representation and manipulation, and convey meanings from stored information within documents. This approach also supports flexible and dynamic modeling by allowing users to get involved in specifying folder organization and filing criteria. This gives the system the capability to be used in various application domains and serves as personalized information system.

The paper also presents a structure for storing domain knowledge which is needed in document filing and retrieval. The knowledge base lets the system to be able to communicate with the user in specifying a folder organization. Therefore the system can organize the documents the same way as the user expects. Knowing how the documents are organized inside the system, the user can specify precise queries, which in turn increase the retrieval efficiency and effectiveness. A learning agent for acquiring the needed domain knowledge is defined. The knowledge base structure and machine learning algorithms need further extended and verified in various application domains.

REFERENCES