A Framework of Personalized Intelligent Document and Information Management System

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

A framework for building a personal information system is introduced. A personal information system can not only help users managing their personal documents, but also act as an intelligent agent to search and collect useful documents from existing information system including Internet on behalf of users. The system employs a flexible dual-model approach. A document type hierarchy describes the conceptual structure of the documents. The folder organization, serves as document filing model, emulates the real world structure for organizing and storing documents in an office environment. The system supports complex queries. Users can search documents based on knowledge about the documents on different level of abstraction. This paper presents a framework that integrates various information retrieval techniques.

Keywords: Information Retrieval, Document Processing, Knowledge-based System.

INTRODUCTION

The amount and complexity of information is currently growing at an incredible rate. 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.

To provide efficient and effective information retrieval, it is essential to use any knowledge that users may have about the documents, which can be in any degree of abstraction. It can be categorized into four levels. The first level is keywords in the documents. The second one is the subject level. The third level is the knowledge about layout and logical structure of the documents. The fourth level is domain knowledge about how documents are related and used in the application domain. There are various information retrieval technologies focusing on each level [1][2][3][6][7][9][11]. We believe all these technologies have their own strength and shortcomings. One of the major common problems is that users have different interests and interpretation of documents. It is almost impossible for a pre-defined document model to satisfy all users’ different preferences.

Our solution is personalized information system that can be built in standalone or based on existing information system. Personal information system takes advantage of users' familiarity with their documents and allows users to define document model based on their interest and interpretation of documents. Another difference of personal information system compared with traditional information system is the use of document filing model. The basic idea is that, rather than focuses on how to search documents in a large document base, personal information system tries to collect and organize documents in a way that they are ready to deliver when needed. Document retrieval is usually equivalent to identifying a specific folder. Finally, our approach provides a framework that can integrate various information retrieval techniques.

In addition to managing personal documents, personal information system can be used as front end of traditional information system that takes care of user's special preference on documents and acts as a search agent.
collecting documents that are in user's interests. Personal
information systems are expected to provide most of
information that users may need and therefore
significantly reduce the users' need to search traditional
information system. It also frees the traditional
information system from supporting untrained users who
has special preferences on documents and demand user-
friendly, effective document retrieval.

After an overview of the system, the paper will introduce
how documents are classified and filed into the document
base, what knowledge is used and how it is acquired. A
rule-based architecture will be proposed to integrate
various information retrieval techniques that retrieve
documents using knowledge on different abstraction level.
A document search agent will be proposed for searching
and collecting documents that are in user' interests from
Internet.

SYSTEM OVERVIEW

DOCPROS (Document Processing System) is a
computerized environment for personal users to
manipulate their documents. The system provides
functional capabilities of collecting, classifying, filing,
storring, retrieving, and reproducing documents, as well as
extracting, browsing, retrieving and synthesizing
information from a variety of documents. It consists of a
document collecting agent, a document classification
component, a document extraction component, a
document filing component, a document browsing and
retrieval component, a document evaluation engine and a
knowledge base (Fig. 1).

A dual modeling approach is employed for describing,
classifying, filing and retrieving documents. This dual-
model consists of two hierarchies: a document type
hierarchy that describes the conceptual structure of the
documents, and a folder organization that 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 that 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 component is
responsible for finding its best fitting type based on the
analysis of the layout and logical structure of the
document. The extraction component will then
summarize the document from the viewpoint of its frame
templates, and yields a synopsis of the document that is
called a frame instance.

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.

An intelligent document collecting agent is responsible
for searching and collecting documents periodically from
existing information system including Internet. The agent
analyzes the folder organization and learns what
documents are in user's interests. User can control the
agent by adjusting the criteria in folder organization.

![Figure 1. System Architecture](image)

DOCPROS (Document Processing System) is a
computerized environment for personal users to
manipulate their documents. The system provides
functional capabilities of collecting, classifying, filing,
storring, retrieving, and reproducing documents, as well as
extracting, browsing, retrieving and synthesizing
information from a variety of documents. It consists of a
document collecting agent, a document classification
component, a document extraction component, a
document filing component, a document browsing and
retrieval component, a document evaluation engine and a
knowledge base (Fig. 1).

A dual modeling approach is employed for describing,
classifying, filing and retrieving documents. This dual-
model consists of two hierarchies: a document type
hierarchy that describes the conceptual structure of the
documents, and a folder organization that 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 that 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 component is
responsible for finding its best fitting type based on the
analysis of the layout and logical structure of the
document. The extraction component will then
summarize the document from the viewpoint of its frame
templates, and yields a synopsis of the document that is
called a frame instance.

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 KNOWLEDGE BASE

Knowledge about documents is categorized into four
levels based on degree of abstraction. The first level is
keywords that are contained in the documents. These keywords are extracted from documents during document classification and are stored in frame instances. The second one is the subject level. The relationships between keywords and concepts are stored in thesaurus.

The third level is the knowledge about layout and logical structure of the documents. It includes the common DWGs for each document type, the document type hierarchy that represent the logical structures of documents, and the relationship between the layout structure and logical structure. Knowledge at this level is learned during document classification and the training process.

The fourth level is domain knowledge regarding how documents are related and used in the application domain. A learning agent [4] is developed for acquiring domain knowledge. The major challenge for the learning agent is to determine what kind of knowledge is likely known to users and is also useful to improve the effectiveness and efficiency of document retrieval, as well as how to acquire the knowledge. A good starting point is the folder organization. Defined by users, the folder organization captures users’ preference in document organization. The filing criteria indicates what knowledge is familiar to the user and is more likely be used as criteria to search documents.

**Predicate Based Representation of Documents**

A predicate-based language is adopted for specifying both document filing criteria and document retrieval queries. One of the reasons we use predicate-based language is its flexibility. It can be used to specify knowledge about document at all four levels. The direct content of documents that is in users interest is carried by frame instances, which includes keywords and structural information. 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 may not be contained in original document but can domain knowledge base.

Predicates are made directly or indirectly on frame instances. 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 1.** 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 $o$) 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, $\text{Date}(o, 4/25/96)$ denotes that the frame instance $o$ is dated 4/25/96. $\text{Cause}(\text{Accident, vehicle})$ denotes that the document is relevant to 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, $\text{Class}(\text{Product, DBMS})$ represents that the document is talking about specific DBMS product. $\text{Affiliation}(\text{Author, x})$ will assign the affiliation of the author to the variable x.

**Definition 2.** (Predicate Constraint) A predicate constraint $\alpha \theta \beta$ is a relation between variables and values, where $a$ and $b$ are either a variable or a value, and $\theta \in \{=, \neq, \in, \not\in, <, >, \leq, \geq\}$.

**Definition 3.** (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 4.** (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.

**Document Filing**

An agent-based approach is used for document filing. Detailed description about document filing and folder organization can be found in [4]. A filing agent is an intelligent object, which monitors its associated folder in a folder organization. The agents communicate with each other through message passing.
The filing algorithm can be summarized as follow:

1. The new document (frame instance) is sent to root agent.
2. For each agent A, when the new frame instance arrives, A’s filing method is invoked which start the document evaluation engine to examine if the frame instance satisfies the filing criterion of the folder.
   a. If it does, deposit the frame instance into A’s folder and send the frame instance to all of A’s descendants.
   b. Otherwise, discard the frame instance.

**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 [10]. The browsing subsystem is introduced in [12].

Queries are specified in terms of predicates. The retrieval process starts with a 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 [10]. 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 \( T \cap f \), 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 text retrieval methods 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.

**DOCUMENT EVALUATION ENGINE**

The document evaluation engine is responsible for determining whether a document meets a criterion. The document evaluation engine uses a rule-based architecture. A rule is a piece of software that tests a document against a predicate and returns a confident value, ranging from \(-1.0\) to \(1.0\), indicating if a document satisfies a predicate.

All rules can be categorized into atomic rules and compound rules. Atomic rules are used to test if a document satisfies an atomic predicate. A base class AtomicRule provides code for testing simple atomic predicates, which involves only matching attribute and value. Compound rules use atomic rules to implement complex logic. A compound rule consists of a list of if-rules connected with logical operators, which represent the if-condition, as well as a list of then-rules. A base class CompoundRule provides code to implements the logic. When the if-condition of a compound rule is evaluated to be true, the compound rule fires all then-rules and itself as true.

**Rule Engine**

A document is considered as a domain. The document evaluation engine (also called rule engine) divides the document domain into sub-domains according to the document structure. Each attribute (a logical part of document) is considered as a sub-domain. Every atomic rule listens and works in a specific sub-domain. Every compound rule listens to all its if-rules. As the rule engine goes through each sub-domain of a document, atomic rules registered to that sub-domain are fired. When all if-rules of a compound rule are fired and returned, the compound rule is fired to test if its if-condition is true and returns the result and fires all then-rules accordingly.

**Rule Generation**

Every rule is an object instance of either AtomicRule or CompoundRule. Simple atomic rules that involves only property-value match can be automatically generated by creating an object instance of AtomicRule and provide the property, value and the object. New rules can be added to the rule base by providing a class inheriting from AtomicRule. This allows new techniques be integrated easily. For example, one can implement an intelligent algorithm to test if a logical part of a document is in certain subject rather than just using keyword match. The implementation can be wrapped into a class inheriting from AtomicRule and added to the rule base.

Another way to create new rule is to create a compound rule by provide a list of if-rules. For example, to add a new rule \( A \cap B \Rightarrow C \), one can create a compound rule and
add rule A and B into the if-rule list, and C into then-rule list of the compound rule.

DOCUMENT COLLECTING AGENT

The personal information system can be in standalone or as a front end of an existing information system. When run as standalone, it acts as a personal document management system. When run as a front end of an existing information system, the system provides support for user's special preference on documents in addition to document management. A document collection agent is devised for searching and collecting documents periodically from existing information system including Internet. The folder organization and the filing criteria indicate what documents are in user's interests. The collecting agent composes queries against the existing information system. Documents returned would go through document filing process, in which each document will be evaluated and deposited into the folders if it meets the filing criteria. This provides user a self-refreshing personal information system. It allows user to search the personal information system locally rather than go outside.

CONCLUSION

A framework for building a personal information system is introduced. A personal information system can not only help users managing their personal documents, but also act as an intelligent agent to search and collect useful documents from existing information system including Internet on behalf of users.

Document retrieval is predicate-driven and knowledge based. Users can control what knowledge should be acquired and used for searching documents. This improves the effectiveness of document retrieval.

The paper also presents a rule-based architecture for document evaluation. New and existing information retrieval techniques can be added to the system as rules in a plug and play manner.

Future work that is currently underway includes developing a rule induction agent to derive new rules based on analysis of existing document base.

References


