Multi-classification Document Manager

A rich ontology-based approach for Semantic Desktop

Paulo Maio, Nuno Silva, Ricardo Brandão, Jorge Vasconcelos, and Fábio Loureiro

Abstract We propose a lightweight technological system for managing user's documents according to multiple classification dimensions. The core of the proposal is the application of ALC-expressive ontologies for capturing the multi-property-based classification of documents. The ontology is then responsible for representing (i) the properties that serve for the document classification (e.g. authors, subjects, types), and (ii) the classes of documents specified based on the properties' values of the documents. Once the ontology is populated with data captured from the documents via parsers/analyzers, an inference engine logically classify documents according to the classes.

1 Introduction

A large amount of our information, both in the professional and private domains, is stored in the form of files on our personal computers. These are manipulated by the so called file managers (also known as file browsers and navigators) or specific applications. From the user perspective, current file systems are based on two principles. First, documents are classified according to a single hierarchy: the subdirectory structure. Second, each document is given a single fixed name that is the way users indicate the document to access.

Furthermore, the physical dimension of the documents (e.g. location, serialization) is abstracted by applications (e.g. email client), but preventing or complicating the relation between documents. For example, an email message is related to a specific presentation document in the file system.

Instead, users require an integrated logical view of the documents regardless of their physical location, serialization or manipulating application, which current file-system model and file managers are inadequate to fully satisfy. The so called Semantic Desktop applications [1, 2] emerged in this context, providing an integrated view of several sources and types of document regardless of their physical dimen-

Paulo Maio, Nuno Silva, Ricardo Brandão, Jorge Vasconcelos and Fábio Loureiro
{pam,nps,jrmjb,1060479,1070987}@isep.ipp.pt
GECAD and Department of Informatics, School of Engineering – Polytechnic of Porto
Porto, Portugal

sion, but instead considering their content and logical dimensions, thus treating every document as first order citizen. Yet, semantic desktop applications still lack the following features:

- Multi-classification of documents, both formally and informally according to the documents' meta-data, and not only by its physical location in the hierarchy (e.g. the music documents would be listed in the music folder independently of where the file is physically located, but instead dependent on their content);
- Browse documents by multiple paths, i.e. not the (single) physical subdirectory hierarchy but instead several hierarchies representing different classification dimension which turns to be a graph (e.g., a music document would be accessible directly not only by its type but also by its subject);
- Search for documents according to their content, classification, location, etc., and maintain the search constantly updated as a new class of documents (e.g. all documents whose subject is "DEIT" and created after July 2010).

Summarizing, we envisage a system application that allows the management of document such the users do not need to deal with the physical dimension of the document, nor being limited to the single classification provided by the file system.

We propose a lightweight technological system for managing the user's documents according to multiple classification dimensions. The core of the proposal is the application of description logics [3] ALC-expressive ontologies [4] for capturing the multi-property-based classification of documents. The ontology is then responsible for representing (i) the properties that serve for the document classification (e.g. authors, subjects, types), and (ii) the classes of documents specified based on the properties' values of the documents. Once the ontology is populated with data captured from the documents via parsers/analyzers, an inference engine logically classify documents according to the classes' conditions.

The following section further characterizes the problem and the context. The section 3 introduces and describes the proposed system architecture. The section 4 details the process of applying ontologies and inference engines in the system for capturing rules and classify documents. The section 5 presents the experiments and evaluation of the proposed system. Finally the section 6 discusses the evaluation results and in the light of that suggests some directions for future work.

2 Context

Current mainstream file systems are externally exposed as a single hierarchical structure of directories (folders) and files. Besides the physical location, files and folders are annotated with meta-data related to the physical dimension (e.g. creation date, author, access rights). Recent file managers (e.g. Windows Explorer) are able to read application specific meta-data from files and allow its manipulation (Fig. 1). Yet, file managers are unable to:

- Multi-classify files (or documents) and access files accordingly;
- Search for files and save the query as an always update folder;

- Manage files and specific applications document (email, contacts, appointments, bookmarks) as first-order citizens, thus preventing their association with files.

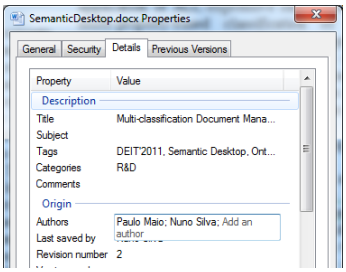

Fig. 1 Snapshot of the properties dialog box of Windows Explorer

Most of the file systems encompass the concept of logical link between folders and files. In most of the file systems, a link (shortcut in Windows OS) is a physical file whose content is a reference to another physical file. Consequently, there is no logical classification of the referred physical file, but instead a physical specification of the classification. From the user perspective this process is neither intuitive nor easy (to create but especially to maintain). This considerably differs from the hyperlink concept popularized in the Web page, where the link is in fact a part of a document. This small difference causes an enormous difference in terms of usability and implementation (i.e. it is not necessary to have a physical file as a hyperlink to another file/document).

When a web page is created online (i.e. when it is requested), the content (e.g. links to other documents) not only depends on the available document but also on the user request. Based on this observation, the envisaged document manager has to provide similar features in respect to the search and classification of documents, i.e. (i) allow searching the repository based on several parameters (the user request), and (ii) classify the documents (available documents/data) according to the query (the user request).

3 Architecture

The proposed system architecture is depicted in Fig. 2. The core components are described next:

- the source repositories, which contain the documents to manage;
- a set of analyzers, that are responsible for reading the sources, distinguish individual documents and extract respective meta-data. These analyzers depend on the type of the source repositories. E.g. an analyzer for NTFS is different from an analyzer for Ext2 or from an email client content analyzer;
- the internal repository, is where the document meta-data and classification rules are stored for internal use;

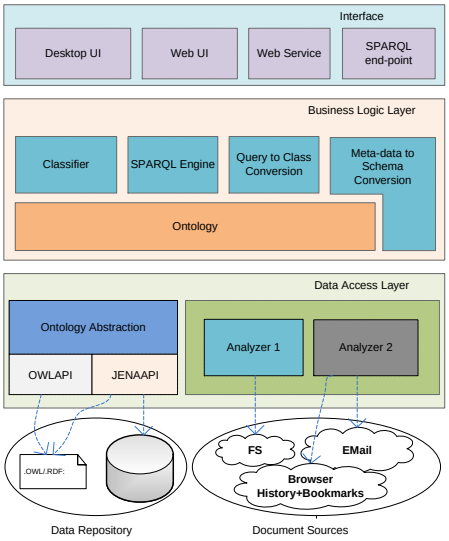

Fig. 2 System architecture

- the ontologies (i) model schema respecting the meta-data generated by the analyzers and (ii) capture the classification rules and queries. Hence, ontologies depend on the analyzers capabilities to extract meta-data from source repositories and form the classification and queries initiatives. The role of ontologies is further described in section 4;
- the classification engine (classifier) is a generic, off the shelf inference engine, responsible for the actual document classification. The schema and classification rules from the ontologies are applied by the classifier for classifying the documents according to their meta-data;
- the system interface provides the mechanisms for different entities to use the system. It should provide ways to (i) query, (ii) change classification rules and (iii) update documents meta-data. Not only the human user is considered in the process (desktop/web application and SPARQL) but also other systems (through web services and SPARQL).

4 Classification

This section describes the details of the document multi-classification feature. This feature includes three processes: (i) define the classification rules, (ii) convert the user defined queries into classification rules, (iii) the actual document classification. As introduced in previous section, these processes are founded in two core components: ontologies and classifier. In this particular case, the classification process will be responsible for:

- Retrieving all documents for a given specific class(ification);
- Realization of documents, i.e. determine the classes documents belongs to;
- Determining whether documents statically, user-based classified do not violate descriptions and axioms described by the classification rules.

The classification process will allow two other non-functional requirements:

- Check the satisfiability of a class(ification), i.e. determines whether a document can exist that would be classified as such. Checking satisfiability permits determine whether the rule generated (either by the user or by the "Query to Rule") is valid;
- Determines the subsumption of class(ification), i.e., determines whether class(ification) A is more general than class(ification) C. This allows defining hierarchical relation between defined classes.

Ontologies will be used by the classifier in runtime for classify the documents. Ontologies are characterized according to several dimensions [5] including the dimension of the community using it and the size and dynamics of the domain. In this context though, the formalism and expressivity level, which implies the reasoning complexity, are the main dimensions to consider. While ontology technology is around for many years, development boosted with the advent of the semantic web [6] in the last decade, giving rise to technology appropriate in this context.

The semantic web ontology language (OWL) [7] proposes several dialects depending on the required expressivity and reasoning complexity. OWL DL is a dialect that adopts Description Logics principles, namely, high expressivity while maintaining decidability. Decidability is a fundamental requirement because it is necessary to classify documents in finite time: its reasoning complexity is NExpTime-complete. It is our conviction that ALC-expressive ontologies are sufficient for capturing the required rules of the envisaged scenarios. Being a less expressive than SHOIN, the system can also benefit from its smaller reasoning complexity (i.e. ExpTime-complete), which is more adequate for online user-oriented systems.

In order to capture the document classification rules it is necessary to define modeling principles that fit the requirements. Ontology design patterns are commonly used for many different modeling problem types [8]. For the problem in hands, we have decided to adopt the ontology design patterns called "definable class" and "primitive class".

Definition 1 (Primitive class). $A \subseteq C$ (A is a subset of C). I.e. A is an atomic class that is a subclass of the complex class C. This defines the necessary conditions (defined in C) of class A. Consequently, the instances of a primitive class must be explicitly defined as such.

Definition 2 (Definable class). $A \equiv C$, (A is equivalent to C), where A is an atomic class (concept in DL terminology) and C is a complex class. The ≡ symbol determines that any instance that satisfies the right-side defined conditions is an instance of class A.

A complex class is defined in term of the Attribute Language with Complements (ALC) whose constructs are: $\bot$, A, $\neg C$, $C \wedge D$, $C \vee D$, $\exists R.C$, $\forall R.C$, where:

- $\bot$ is the empty set or an inconsistency;
- A is a class name (atomic class);

- C and D are complex classes;
- R.C, R is a role whose range is of type C.

Next is presented an example of an ALC-expressive ontology for the document multi-classification domain:

1. MusicDoc ≡ Doc ∧ ∃hasType.(MP3∨WMA)
2. MyDoc ≡ Doc ∧ ∃hasAuthor.(ME)
3. Doc ⊆ ∃hasType.Type ∧ ∃hasAuthor.Person
4. Person ≡ ME ∨ ELVIS
5. ME ∧ ELVIS≡⊥
6. Type ≡ MP3 ∨ WMA ∨ HTM
7. MP3∧WMA≡⊥;MP3∧HTM≡⊥;WMA∧HTM≡⊥

The first class (classification rule) defines MusicDoc class(ification) as a definable class that encompasses all the Docs whose extension is either MP3 or WMA. The second class defines MyDocs as all Docs whose author is ME. Third class defines Doc as anything that as type and an author. The fourth and fifth lines define a set partition for Person and lines six and seven define a set partition for Type.

Consider now the following individuals and property value definition according to the previous ontology:

1. d1∈Doc, hasAuthor(d1,ME), hasType(d1,MP3)
2. d2∈Doc, hasAuthor(d2,ELVIS), hasType(d2,MP3)

Through the classification process, the classifier would infer the following:

1. d1∈MusicDoc, d1∈MyDoc
2. d2∈MusicDoc
3. MusicDoc≡{d1,d2}
4. MyDoc≡{d1}

5 Experiments

The experiments envisage evaluating the proposal and determine (i) how intuitive the multi-classification of documents is and (ii) how satisfactory the query to classification conversion is. For that we fully developed the data access and the business layers as described, and developed the desktop UI (Fig. 3a) and the web UI (Fig. 3b) of the interface layer. The desktop UI application combines the hierarchical physical structure of the source repositories, with the user defined classifications. The web UI application instead, abandoned the hierarchical view and presented the user defined classes of documents only. We prepared the experiments in three laptop computers with documents from the file system and email messages.

The experiments ran for three groups of 3 persons with 3 different levels of proficiency (Basic, Medium and High-level proficiency using file managers). None of the users was previously familiar with the documents or the application. Additionally, we defined three tasks. In task 1 the user is required to search for the docu-

ments for a specific project (subject). The expected result is a set of files and email messages. In task 2 the user is required to define a new folder BlueBerry whose documents' colour is Blue and smell like berries. It was expected that the user creates a primitive class whose necessary conditions correspond the specified characteristics. In task 3 the user is required to search for blue documents. It was expected the user creates a definable class BlueDocs subsumed by the BlueBerry class.

Fig. 3 Application screenshots: A) Desktop UI and B) Web UI

Table 1 depicts a summary of the experiments. Columns A reflect the success accomplishing the task (either y or n). Colunms C reflect the user perception of the complexity of the task. Columns S reflect the user facility understanding the task results. Colunms C and S are valued in the range 1-4. The evaluation results of the Web UI are depicted in shaded cells whereas Desktop UI results are depicted in clear cells.

Table 1 Summary of experiments

U	P	Task 1						Task 2						Task 3					
		A		C		S		A		C		S		A		C		S	
1	B	Y	Y	2	2	3	3	Y	Y	2	2	3	3	Y	Y	2	2	1	3
2	B	N	Y	3	2	1	2	Y	Y	4	2	3	3	Y	Y	4	4	1	3
3	B	Y	Y	3	1	2	3	Y	Y	3	2	3	4	Y	Y	3	2	1	4
4	M	Y	Y	1	1	4	4	Y	Y	1	1	4	4	Y	Y	1	1	2	4
5	M	Y	Y	1	1	4	4	Y	Y	2	1	4	4	Y	Y	2	1	2	4
6	M	Y	Y	2	2	4	4	Y	Y	2	2	4	4	Y	Y	2	1	2	4
7	H	Y	Y	1	1	4	4	Y	Y	1	1	4	4	Y	Y	1	1	3	4
8	H	Y	Y	1	1	4	4	Y	Y	2	1	4	4	Y	Y	1	1	3	4
9	H	Y	Y	1	1	4	4	Y	Y	1	1	4	4	Y	Y	1	1	4	2

6 Conclusions

Considering the evaluation results, we conclude that:

- The users are able to understand the concept of multi-classification;

- The users accomplish simple classification of documents based on their characteristics;
- The more the users are proficient with file managers and understand the hierarchical concept, the more they are able to understand and deal with multi-classification;
- The users do not like or do not understand subsumption relations between defined classes. This is clear from the comparison of satisfaction columns on task 3. Because on the Web UI app the subsumption relation was not represented, users understood and were satisfied with results. Instead, in Desktop UI application, they did not understand the result. The exception is user 9 that understood the subsumption relation and did not liked the fact that it was not represented in the Web UI app.

Accordingly, it is necessary to research more on the need for representing subsumption relation between user defined classes as it might be a factor contributing for misunderstandings. Also, because multi-classification conceptually gives rise to a graph, it would be interesting to analyze the effect of such representation on the user.

Acknowledgments

This work is partially supported by the Portuguese MCT-FCT project COALESCE (PTDC/EIA/74417/2006).

References

1. Papailiou, N., Christidis, C., Apostolou, D., Mentzas, G., Gudjonsdottir, R.: Personal and Group Knowledge Management with the Social Semantic Desktop. Proc. Collaboration and the Knowledge Economy: Issues, Applications and Case Studies. (2008).
2. Decker, S., Park, J., Quan, D., Sauerman, L.: The Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure. Proceedings of Semantic Desktop Workshop at the ISWC, Galway, Ireland. (2005).
3. Baader, F.: The description logic handbook: theory, implementation, and applications. Cambridge University Press (2003).
4. Description Logic Complexity Navigator, http://www.cs.man.ac.uk/~ezolin/dl/.
5. Hepp, M., De Leenheer, P., de Moor, A.: Ontology management: semantic web, semantic web services, and business applications. Springer-Verlag New York Inc (2007).
6. Lee, T.B., Hendler, J., Lassila, O.: The Semantic Web. Scientific American. 284, 34-43 (2001).
7. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref/.
8. Ontology Design Patterns . org (ODP) - Odp, http://ontologydesignpatterns.org/wiki/Main_Page.