Semantic Document Architecture for Desktop Data Integration and Management

Saˇsa Neˇsi´c1, Dragan Gaˇsevi´c2, Mehdi Jazayeri1
1Faculty of Informatics, University of Lugano, Lugano, Switzerland
2School of Computing and Information Systems, Athabasca University, Athabasca, Canada
1{sasa.nesic, mehdi.jazayeri}@usi.ch, 2dgasevic@acm.org

Abstract

The paper presents a novel desktop document architecture, namely SDArch, which attempts to integrate data from heterogeneous desktop applications into a unified desktop information space. To achieve this, SDArch introduces a new document representation model, which establishes explicit semantic links between fine-grained units of document data based on the conceptualization of their semantics. The SDArch semantic search and navigation services, which run on this unified desktop information space, aim to improve the search and navigation within desktop data, thus improving the effectiveness and efficiency of desktop users in carrying out their daily tasks. We report on a usability evaluation of the SDArch prototype and an experimental evaluation of the proposed semantic search. The evaluation results are promising. We present an analysis of these results.

1. Introduction

Over the last decade, personal desktops have faced the problem of information overload due to increasing computational power, easy access to the Web and cheap data storage. Moreover, an increasing number of diverse end-user applications have led to the problem of information fragmentation. Accordingly, discovering and accessing information stored on the desktop represent the main challenges for desktop users carrying out their daily activities. Several research projects in the field of the Semantic Desktop [6][10][2][4] have attempted to solve these problems by applying Semantic Web technologies [1] on local personal desktops. They attempt to set up an additional semantic layer on top of the existing desktop infrastructure. On this layer, desktop resources are identified by their URIs and described by RDF descriptions using agreed upon vocabularies (ontologies) [1]. However, the desktop information space differs from the Web in many aspects, which hamper the full potential of the applied Semantic Web technologies. A key problem with the use of RDF in the context of the desktop is the modeling of the relationship between a resource URI from the semantic layer containing resource descriptions and the resource itself (i.e., the content of the resource). While on the open Web it is acceptable that links can become broken, on the desktop, the integrity of resources and their descriptions must be ensured [5]. Accordingly, the existing semantic desktop approaches need synchronization mechanisms to propagate modifications to the RDF layer. Due to the complexity of the current desktop infrastructures, the design and implementation of such mechanisms represent the main challenge. The other problem of having a separate semantic layer is the limited ability to identify and describe fine-grained, self-contained information units stored inside conventional desktop documents [5]. This is due to existing conventional document formats, which provide limited mechanisms for structuring and identifying the actual document content.

In this paper we present a novel desktop document architecture, called the Semantic Document Architecture or SDArch, which should solve the problems described above and facilitate desktop data management. With SDArch, desktop users are able to search and navigate across their desktop data more efficiently, thus completing their tasks more efficiently. SDArch introduces a new representation model called the Semantic Document Model (SDM), which integrates the semantic layer into the core of the document data representation structures. The model is inline with the Linked Data principles [3], representing documents as RDF linked data sets, in which document units are identified by globally unique URIs, annotated by an arbitrary number of ontological concepts, interlinked with other document units via explicit semantic links, and connected by the content stream to their binary content. While semantic documents (i.e., instances of SDM) can be browsed and edited in a number of application specific formats, each semantic document has only one permanent instance which is integrated into a unified desktop data graph and stored into a desktop RDF repository. On top of the desktop RDF repository, we have built a service-oriented middleware with a set of services that allow application software to interact with semantic documents. Moreover, the repository exposes a publicly-available SPARQL endpoint, thus enabling remote access to
the desktop data and its integration into the global Linked Open Data (LOD) cloud \[3\]. Together, the RDF repository, the middleware and the application software that utilizes the middleware services compose SDArch as a three-tier service-oriented architecture.

The remainder of the paper is structured as follows. Section 2 outlines the main characteristics of SDM. In Section 3, we describe the design principles of SDArch. In Section 4, we give a more detailed description of the SDArch services, which implement semantic annotation, indexing and linking of document units from semantic documents. Then, in Section 5, we describe the SDArch search and navigation services and show an example of their integration into application software. Section 6 presents the results of the preliminary evaluation of the proposed architecture. We have a look at some related efforts in Section 7, before concluding the paper with some final remarks in Section 8.

2. Semantic Document Model

We have created the semantic document model (SDM) \[8\], aiming to provide the infrastructure for the integration of semantically related data units, which are fragmented into diverse desktop application formats, thus unifying the desktop information space and improving personal information management \[10\]. Moreover, the model intends to bridge the gap between the desktop information space and the global LOD cloud, so that desktop users can directly access and integrate information from the LOD cloud as well as publish their desktop data to the LOD cloud.

SDM represents document units as RDF nodes, each of which being uniquely identified by means of an HTTP dereferencable URI \[3\]. In order to enable semantic linking of document units over shared semantics, SDM provides mechanisms to conceptualize the semantics of document units and to link them to the document units’ RDF nodes as semantic annotations. The conceptual representation of a document unit’s semantics is achieved by a combination of an arbitrary number of ontological concepts along with their relevance weights for the document unit. By having semantic annotations linked to document units’ RDF nodes, SDM establishes semantic links between nodes which share the same annotation concepts.

SDM is formally described by the smd ontology \[8\], which specifies possible types of document units (e.g., smd:paragraph, smd:section, smd:table and smd:illustration), types of hierarchical and structural relationships between document units (e.g., smd:hasPart, smd:isPartOf and smd:belongsTo), and the semantic annotation and semantic linking interfaces. The semantic annotation interface consists of the smd:Annotation entity and its two properties: the smd:annotationConcept, which holds a reference to the concept from an annotation ontology and the smd:conceptWeight that determines the relevance of the annotation concept for the document unit it annotates. The semantic linking interface consists of the smd:SemanticLink entity and the following properties: the smd:unitOne and smd:unitTwo, which refer to the document units to be linked; the smd:linkConcept that holds the reference to the ontological concept that annotates both units and determines the type of the semantic relationship; and the smd:linkStrength property whose value determines the strength of the semantic relationship between the units. In addition, every RDF node, representing a document unit, has one content stream connecting the binary content of the document unit, stored into the binary content repository, to the document unit’s RDF node.

3. Architecture Design

We have designed SDArch as a three-tier architecture composed of a data layer, service-oriented middleware and presentation layer. The data layer contains the desktop RDF repository, the binary data repository, the concept index and the text index. The RDF repository stores semantic documents, that is, RDF instances of SDM, while the binary content of document units are placed into the binary content repository. SDArch maintains one single concept index that enables semantic search (Section 5) over RDF data and a single text index that enables content-based search over the binary content of document units. The middleware provides a set of services that implement functionalities of SDArch, defines communication protocols among the services and provides a service registry for registering or unregistering services from the middleware. Two main usage scenarios we focussed on while designing the current prototype of the SDArch middleware are: 1) semantic document authoring and publishing by transforming conventional documents into semantic documents and 2) semantic search and navigation across semantically integrated collections of desktop documents. New scenarios can be easily enabled by adding new services to the SDArch middleware and making them communicate with the existing services. Figure 2 shows an illustration of SDArch depicting internal processes and relationships among services for the two above-mentioned usage scenarios. In the following two sections we describe in more detail the services and processes related to both scenarios. For the prototype development of the SDArch middleware, we used Windows Communication Foundation (WCF) .NET framework. As an example of the user interface of the SDArch services, we have developed a set of tools called ‘SemanticDoc’ and integrated them into MS Office. Implementation details, snapshots, and demos of the use of the SDArch prototype and SemanticDoc tools can be found on our project Web page \[\text{http://www.semanticdoc.org}\].
4 Semantic Document Authoring

We distinguish between two possible scenarios for authoring semantic documents. The first one is to transform documents of conventional document formats (e.g., MS Word, MS PowerPoint or PDF documents) into semantic documents. The second one is to use application software developed specifically for authoring semantic documents from scratch. In order to let users continue to work in familiar environments while taking advantage of the new architecture as well as to enable conversion of existing documents into semantic documents, in the current SDArch prototype we implemented the first authoring scenario. Moreover, it is also inline with our hypothesis that semantic documents can be browsed and edited in a number of application specific formats, that is, there can be many possible document views but there is only one permanent semantic document copy integrated into the desktop data graph [8].

Figure 1: SDArch layers - illustration

The transformation process starts with the SemanticDoc RDFizer service which scans a document to be transformed, recognizes document units (e.g., sections, paragraphs, tables, illustrations and slides), generates RDF nodes and URIs of document units, generates RDF links which represent structural and hierarchical relationships between the units and stores the generated RDF data into the desktop RDF repository. For each document unit, the SemanticDoc RDFizer calls the annotation, indexing and linking services. These three services all rely on the discovery of concepts from domain ontologies which conceptualize human-consumable information stored in document units. The concept discovery is realized by the knowledge extraction service, which utilizes domain ontologies and applies the concept exploration algorithm (CEA) [9], to discover ontological concepts of the document units and to calculate their relevance weight to the document units. As domain ontologies, the service uses existing ontologies from the Semantic Web. There already exist a number of widely used, shared ontologies, many of which are applicable for desktop data. By using shared ontologies we enable different applications, either on the same or networked desktops, to interpret desktop data in a semantically correct way.

The annotation service uses the discovered ontological concepts and their corresponding weights to form the semantic annotations, which are then linked to the RDF nodes of the document units, via the annotation interface (Section 2). The indexing service adds an entry for each document unit to the concept index. The entry contains a weighted concept vector [9] of the document unit’s annotation concepts. After the semantic annotation and indexing, the next step is to link semantically related document units by establishing semantic links between their RDF nodes. If two document units are annotated by the same ontological concept, then the two document units share some semantics, which determine the semantic relationship between them. The linking service computes the strength of the semantic relationship between the units over the annotation concept as a product of the concept’s weights of each unit. Then, the service generates an RDF instance of the semantic link interface (Section 2), thus creating the semantic link between the document units. By setting up semantic links between document units originating from heterogeneous, application-specific formats, desktop data becomes connected into an integrated, desktop-data graph stored into the desktop RDF repository. Moreover, by exposing the SPARQL endpoint to the desktop RDF repository, we contribute to the LOD cloud [3] by adding data from desktop documents.

5 Semantic Search and Navigation

In the traditional hypertext Web, browsing and searching are often seen as the two dominant modes of interaction [1]. While browsers provide the mechanisms for navigating the information space, search engines are the place where navigation begins. In order to bring this paradigm to the desktop information space, SDArch provides the semantic search and navigation services. We now describe the main features of the two services and present some examples of their integration into an existing, well-established document authoring environment such as MS Office.

The search process normally starts with users constructing a query that reflects their information needs. As the initial form of the user query, the semantic search service takes a free-text query. The service then models the semantic meaning of the query by forming a weighted query concept vector [9] that we refer to as a semantic query. To do this, the search service actually employs the knowledge extraction service, which discovers query concepts and calculates their relevance weights for the query. The search
service forms semantic queries in the same way the indexing service represents document units in the concept index. This results in both the document units and the user query being represented by the corresponding weighted concept vectors. The rest of the search process is as follows. From the concept index, the search service discovers document units which are indexed by some of the concepts from the semantic query. Then, the service calculates the similarity between the discovered document units and the semantic query, by calculating the cosine similarity between the document units’ concept vectors and the query’s concept vector. Based on the calculated similarity, the service ranks the document units and forms the retrieval list. Finally, for each document unit from the list, the service queries the RDF repository to acquire the details of the document unit and sends them to the user. Figure 2 a) shows the interface of the ‘SemanticDoc’ search add-in that we have developed to enable MS Office users to get access to the search service.

The semantic navigation service enables users to traverse semantic documents by navigating along the semantic links. The navigation process assumes the existence of an exploratory interface through which the users interact with the semantic navigation service. Figure 2 b) shows the ‘SemanticDoc’ browser that we have developed and integrated into MS Office as a supplement to the ‘SemanticDoc’ search add-in. The navigation process starts by the user browsing a document unit retrieved by the search service and clicking on a label one of the unit’s annotation concepts, which represents a potential semantic link. This user action activates the navigation service, which takes the URIs of the document unit of the clicked concept, forms a corresponding SPARQL query (Figure 3), and executes it against the RDF repository. The document unit can be linked to many document units via the same semantic link determined by the clicked concept, thus the query can return multiple document units. The query orders the returned document units by the strength (Section 4) of the semantic link between them and the initial document unit. Finally, the service acquires the details of the document units, and sends them to the user.

6. Preliminary Evaluation

As a preliminary evaluation of SDArch we have conducted two evaluation studies: 1) the usability evaluation of the SDArch prototype and the SemanticDoc tools, and 2) the experimental evaluation of the proposed semantic annotation, indexing and retrieval.

6.1 The Usability Evaluation

The usability evaluation has been designed to consider user effectiveness, efficiency and satisfaction in using the SDArch prototype and SemanticDoc tools. It consisted of a set of tasks related to authoring of course material, that is, preparing a PowerPoint presentation for the ‘Software Design Patterns’ topic. Recent research in learning content authoring has shown that most authors reuse and modify existing content, available in their own archives or on the Web [2], rather than authoring from scratch. Therefore, the efficient exploration of data on user desktops is the main prerequisite to efficient authoring. In the evaluation, 12 participants, from three universities, took part. All participants were volunteers and were familiar with software design patterns. The evaluation document set consisted of more than 70 topic’s related documents. The documents were available in their original formats (i.e., Word, PowerPoint and PDF) and as semantic documents generated by SDArch and stored into desktop RDF repository. All participants were asked to perform a set of seven tasks, designed so that their successful completion results in creation of an overall presentation of an appropriate quality level. Moreover, we asked the participants to perform the tasks two times arranged in two continuous but unlimited time sessions. One half of the participants first used the system with conven-
tional desktop tools and then the system with SDArch, and the other half used the systems in the opposite order. The evaluation session consisted of two phases, namely observation and feedback. In the observation phase, we were observing the participants and tracking their behavior using screen recordings while they were conducting the evaluation tasks. All participants performed the tasks on our PC with remote access control.

Figure 4: Average and median task execution times

With respect to user effectiveness, we tracked how many and which tasks participants completed successfully. Results showed that all participants successfully completed all tasks using both systems, which was mostly because of unlimited time sessions and the genuine motivation of the participants. With respect to user efficiency we measured the execution times, the amount of mouse clicks and the number of window switches during the tasks’ executions. Figure 4 presents the average and median execution times, for each task for both systems. From the figure we can see that, regarding task execution time, our system outperforms the conventional system.

Table 1 shows the relative performance of the participants when using our system with respect to the use of the conventional system. For example, a value of 87.1% indicates that participants using our system needed 87.1% of the mouse clicks that participants using the conventional system needed. All values from Table 1 are less than 100%, which means that the performance of the participants regarding both number of mouse clicks and window switches are better when using our system for all tasks. The measured execution times, number of mouse clicks and number of window switches showed that the efficiency of the participants when using our system is better than their efficiency when using the conventional system, with respect to the measured criteria.

The participants provided us their subjective feedback about the usefulness and ease-of-use of our system by filling a questionnaire at the end of the evaluation session. The questionnaire contained 9 statements, first 5 statements (S1-S5) designed to gather a subjective evaluation of the system usefulness, and the following 4 statements (S6 - S9) designed to gather subjective evaluation of the ease-of-use of the system. Statements S5 and S9, which express the overall satisfaction regarding the usefulness and ease-of-use respectively, were the two highest-rated statements with an average rating of 4.8 out of 5. The other statements were also rated as highly positive with average ratings ranging from 4.1 to 4.7.

6.2 The Experimental Evaluation

The first goal of the experimental evaluation was to show if our semantic annotation approach can produce a substantial amount of semantic annotations, which is crucial for the semantic linking and integration of desktop data. The second goal was to show how accurate the generated semantic annotations are and to which extent they can improve search on the desktop. The detailed discussion with complete evaluation results are presented in [9]. In this paper we give just a brief overview.

We have designed the experimental evaluation as a proof of concept and it was not meant to address issues of scalability or efficiency. The document set that we used was composed of 350 Word documents containing records for steel, aluminum, copper, titanium, and other metals. The annotation ontology we used the METALS ontology, which contains over 3,500 concepts about metals and their applications. Both the document set and the ontology, we obtained from KEY-to-METALS company, which maintains one of the world’s most comprehensive metals database. We have transformed the evaluation document set into five collections of semantic documents, by applying five different concept discovery strategies: S1 - simple syntactic matching, S2 - lexically expanded syntactic matching, and S3, S4, S5 - lexically expanded syntactic matching and the semantic matching. The first two strategies are present in most existing semantic annotation approaches. The last three strategies comprise all the features of the concept exploration algorithm [9] implemented by the SDArch knowledge extraction service (Section 4), and only differ in the values of some of the algorithm’s parameters. For each strategy, Table 2 shows the total number and the average weight of

<table>
<thead>
<tr>
<th>Task</th>
<th>Rel. performance for (mouse clicks)</th>
<th>Rel. performance for (window switches)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>1</td>
<td>87.1%</td>
<td>86.9%</td>
</tr>
<tr>
<td>2</td>
<td>79.6%</td>
<td>86.4%</td>
</tr>
<tr>
<td>3</td>
<td>75.3%</td>
<td>68.8%</td>
</tr>
<tr>
<td>4</td>
<td>78.9%</td>
<td>83.8%</td>
</tr>
<tr>
<td>5</td>
<td>51.1%</td>
<td>53.3%</td>
</tr>
<tr>
<td>6</td>
<td>57.1%</td>
<td>56.6%</td>
</tr>
<tr>
<td>7</td>
<td>80.4%</td>
<td>86.2%</td>
</tr>
</tbody>
</table>

Table 1: Relative user performance when using the SDArch with respect to conventional desktop

Table 2 shows the total number and the average weight of
the syntactic and semantic matches. These results indicate that our annotation model (i.e., $S_3$, $S_4$ and $S_5$) has potential to produce substantial amount of semantic annotations. To evaluate the performance of the proposed semantic search we formed ten queries related to the data of the evaluation document collection and asked three KEY-to-METALS engines to assess the relevance of document units (i.e., paragraphs) of the collection to the queries. The queries were then executed against each of the five semantic document collections. The measured values of the precision and recall for each of the five semantic document collections showed that the proposed semantic search improves retrieval in terms of both precision and recall.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Syntactic matching</th>
<th>Semantic Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num.</td>
<td>Avg. Weight</td>
</tr>
<tr>
<td>$S_1$</td>
<td>1524</td>
<td>2.56</td>
</tr>
<tr>
<td>$S_2$</td>
<td>3182</td>
<td>3.62</td>
</tr>
<tr>
<td>$S_3$</td>
<td>3182</td>
<td>3.62</td>
</tr>
<tr>
<td>$S_4$</td>
<td>3182</td>
<td>3.62</td>
</tr>
<tr>
<td>$S_5$</td>
<td>3182</td>
<td>3.62</td>
</tr>
</tbody>
</table>

Table 2: Concept discovery results for strategies ($S_1$-$S_5$)

7. Related Work

The idea of using Semantic Web technologies to enhance data interoperability and information management on the personal desktops has been widely researched. A number of Semantic Desktop projects such as [6][10][7][4] have the goal of providing a semantic infrastructure that covers all applications and integrates information sources that users operate on. All of these projects attempt to enhance the existing desktop infrastructures by adding an additional semantic layer, providing semantic descriptions that refer to actual resources. In such scenarios, the semantic integration of desktop resources should happen at the semantic layer by interlinking descriptions of semantically related resources instead of linking actual resources. The main problem here is the propagation of modifications to resources and their relationships to the semantic layer. Moreover, the existing Semantic Desktop approaches also pay little attention to the identification and annotation of small, self-contained data units from conventional desktop documents. They rather consider whole documents as identifiable resources. Linked Data [3] could address this problem, but while there are more and more tools available for publishing Linked Data on the Web, there is still a lack of such tools for local desktops. To the best of our knowledge, only the approach presented in [11] offers the model that integrates the semantic layer into the actual desktop file system, enabling explicit semantic links between desktop files. However, the model recognizes only entire files as identifiable and linkable resources. With respect to semantic annotation of desktop documents, many approaches exist[12], but none of them offer a solution for interlinking data from different documents, which is annotated by the same semantic annotations.

8. Conclusions

In this paper we present a novel desktop document architecture, called SDArch, which attempts to semantically integrate the desktop information space and provides the set of services which aim to improve the effectiveness and efficiency of desktop users in completing their daily tasks. The results of the usability and experimental studies conducted with the architecture prototype were promising. In future work we plan to continue to evaluate the usability of SDArch services as well as the performances of the proposed semantic annotation, indexing and search and to provide further statistical analysis of the collected data.

References