Towards Efficient Document Content Sharing in Social Networks

Saša Nešić, Francesco Lelli, Mehdi Jazayeri
Faculty of Informatics, University of Lugano
Lugano, Switzerland
sasa.nesic@unisi.ch
francesco.lelli@unisi.ch
mehdi.jazayeri@unisi.ch

Dragan Gašević
School of Computing and Information System, Athabasca University
Brookhaven National Lab
Athabasca, Canada
dgasevic@acm.org

ABSTRACT

Social network services have enabled the increasing sharing of digital content (e.g., images, videos and audios). However, despite the fact that office documents hold a significant amount of users' digital content, office documents have not yet been sufficiently exploited by social networks. The main reason for this is that existing office document architectures/formats are not open enough for selective access, reuse and commenting of document parts. As a response to this problem we have developed a new document architecture, namely the Semantic Document Architecture (SDArch), which enables the annotation of document content with semantic and social context annotation and provides easy access and reuse of the desired document parts. In this paper we focus on the social context annotation (SCA) that we have introduced to capture implicit information about the usage of document content in the context of the social network. We present a ranking algorithm that uses SCA along with user profile information, to get more personalized search results. The current version of SDArch prototype, which implements the algorithm, is also discussed.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Design, Algorithms

1. INTRODUCTION AND MOTIVATION

Social Network Services (SNSs) (e.g., Friendster, MySpace, LinkedIn and Facebook), which have become very popular recently, provide a multitude of ways for Web users to collaborate. By using SNSs, people are now better connected and can easily access, reuse or comment content that is authored by other people. Office-like desktop documents (e.g., Word, PowerPoint and PDF) hold a significant percentage of digital content stored in personal computers. Accordingly, they represent an important source of content that could be shared in social networks. Based on existing desktop document formats [4], a number of systems have been devised to support sharing of document content. Document management systems such as Docushare [7] take entire documents as the primary unit of sharing. Others such as Documentum [6] and Interwowen [9] parse the documents using plug-ins to common document authoring tools (e.g., Microsoft Word), extract the content and publish it on the Web. There are also systems, such as Keyholes [12], which support selective sharing - sharing of particular segments of document content by supporting constrained viewing of parts of a shared document. Moreover, group editors [19] provide simultaneous shared access to documents. Although aforementioned systems provide different ways of document sharing, they are not primarily designed for collaborative environments such as emerging on-line social networks.

Existing office document formats keep document content closed into format specific elements, so that it is barely accessible across application boundaries. In the last few years several XML-based document formats have been developed, such as ODF [15] and OOXML [16] (the default formats of OpenOffice and MS Office documents respectively), which opened a way towards simpler transformation of native word processor formats to and from other formats, by providing export/import bridges. However, using one-to-one export/import bridges is unsuited to the highly dynamic online world, where the number of document formats grows constantly. All that limits the document sharing among users working on different operating systems or using different office applications. Next, only entire documents can be considered as resources which can be identified, referenced and shared. Documents are organized into units (e.g.,paragraphs, tables and sections), but these units are not uniquely identified and can not be put in explicit relationships with some 'outside world' resources (e.g., peoples, organizations and places). It is difficult to access and reuse a document unit without obtaining the entire document. In practice, people share entire documents when portions would suffice. Presently, selective sharing of document content is cumbersome, requiring cut and paste which is a laborious and error prone process. Moreover, existing document management systems are not open enough for collaborative document authoring and editing. In software development, the Versions Control Systems (VCS) software keeps track of all changes in a set of files, and allows several developers to collaborate. In office-like document management, there are some similar initiatives such as Microsoft's SharePoint, but they are still significantly less effective and less commonly used than VCS systems. Finally, there are many limitations concerning the content annotation of existing office-like desktop documents. The annotation is usually restricted to predefined annotation vocabularies such as Dublin Core (DC) [8] and Learning Object Metadata (LOM) [11]. Extending annotation vocabularies with a new user-defined annotation type is difficult; it requires the extension of the document annotation schema, which is tedious and not always possible. Moreover, the schema usually provides annotation elements only for a whole document; it is rarely possible to annotate parts of the document (e.g., to specify a creator of specific paragraphs or tables within the document).

We believe that a new universal document architecture providing both a new document model and a collection of services for sharing, annotating and interlinking document content is necessary. The new architecture should fill the gap between individuals' desktop documents and further prepare document content for the envisioned Semantic Web (or the Web of data). The Semantic Web, as an extension of the Web, aims at providing a framework for linking data (not documents) and adding logical assertions over linked data. In other words, the new document architecture should enable unique identification of fine-grained document content units (CUs), easy access, reuse and linkage of document CUs with other uniquely identified resources, and flexible annotation of document CUs with a whole span of existing types of the annotation (e.g., predefined annotation metadata such as DC and LOM, semantic ontology-based annotations and social tagging).

The Semantic Web technologies, Resource Description Framework (RDF) and ontologies in particular, provide structures and mechanisms to interconnect both shared document content and people from the social networks in a meaningful way [3]. By using agreed-upon semantic vocabularies (i.e., ontologies) to describe documents, people and the connections that bind them all together, the social networks can operate via common semantics thus improving document content sharing. SDArch is our proposal for a new document architecture, which relies on the aforementioned Semantic Web technologies. In this paper we are focused primarily on social aspects of the proposed architecture. The rest of the paper is organized as follows. In Section 2, we present SDArch by briefly describing the semantic document model (SDM) and SDArch services. Then in Section 3, we discuss the social context annotation of document CUs. In Section 4, we describe a user model of SDArch users and in Section 5 we explain how we use the information extracted from SCA and values of user preferences specified in the user model to get more personalized search results. Section 6 provides a report on the development of SDArch prototype. In Section 7, we describe the MS Office add-in that we have developed to enable Office users to access the services implemented in the prototype. The evaluation plans (Section 8) and the conclusion (Section 9) conclude the paper.

2. SEMANTIC DOCUMENTS

In order to enable document content to be efficiently discovered, linked and shared across application, enterprise and community boundaries, office-like desktop documents should be completely open and queryable with a content organized in uniquely identified and well annotated units [14]. SDArch [13] is our attempt to address these issues. It introduces the semantic document model and provides a collection of services and user interfaces that form a single integrated document environment. In the rest of the section we briefly discuss the SDM and the SDArch services respectively.

2.1 Semantic Document Model - SDM

We have created the SDM being inspired by the IBM's Darwin Information Architecture (DITA) [5] which divides digital content into small, self-contained topics that can be reused in different deliverables. The formal description of the SDM consists three ontologies: the document ontology, the document annotation ontology and the document change ontology. We have developed the ontologies using OWL language and they are available at [20].

Document ontology [13, 20] defines a document as a graph composed of document nodes (*DNs*). It defines types of *DNs* and relationships among them. Two main types of *DNs* are: content nodes (*CNs*) and knowledge nodes (*KNs*). *CNs* represent content units in their most basic form (i.e., raw digital content) and can be specialized into discrete *CNs* (e.g., *Graphic* and *TextFragment*) and continuous *CNs* (e.g., *Audio*, *Video* and *Simulation*). *KNs* represent content units which aggregate several *CNs* and add navigation among them (e.g., *Paragraph*, *Section*, *Slide* and *Table*). A paragraph that consists of several *TextFragments* and *Graphics* ordered in a given order is an example of the *KN*. Moreover, the document ontology defines properties that link concepts from domain ontologies to *DNs*. The concepts from domain ontologies which are linked to a *DN* represent the conceptualization of the phenomena/knowledge kept in the *DNs*.

Annotation ontology [13, 20] defines a common interface (concepts and properties) for adding annotations to *DNs*. It introduces the *hasAnnotation* property and *DNAnnotation* concept, which act as annotation binding to a *DN*. The annotations (i.e., instances of *DNAnnotation*) are uniquely identified resources that consist of the annotation identifier, the annotation type and the annotation value. Based on the nature of the annotation type, the annotation value can be either binary value (e.g., text, graphic, audio) or instance of the annotation specific concepts.

Change ontology [13, 20] defines concepts and properties for the formal description of possible changes to *DNs*. The main concept is the *DNChange*, which along with a set of its properties enables capturing information about *DN's* change (e.g., deleted *CNs*, added new *CNs* and reordered *CNs*). Similarly to the annotations, the changes (i.e., instances of the *DNChange*) are uniquely identified resources linked to *DNs*.

A semantic document (see Figure 1) is an RDF graph whose nodes are uniquely identified instances of the *DNs* concepts defined in the document ontology. To RDF nodes are linked concepts from the domain ontologies, *DN* annotations and *DN* changes. The *DNs* of the *CN* type (e.g., *TextFragment*, Graphic, *Audio* and *Video*) also hold their binary contents. However, since current implementations of RDF repositories are not meant to store large chunks of bi-

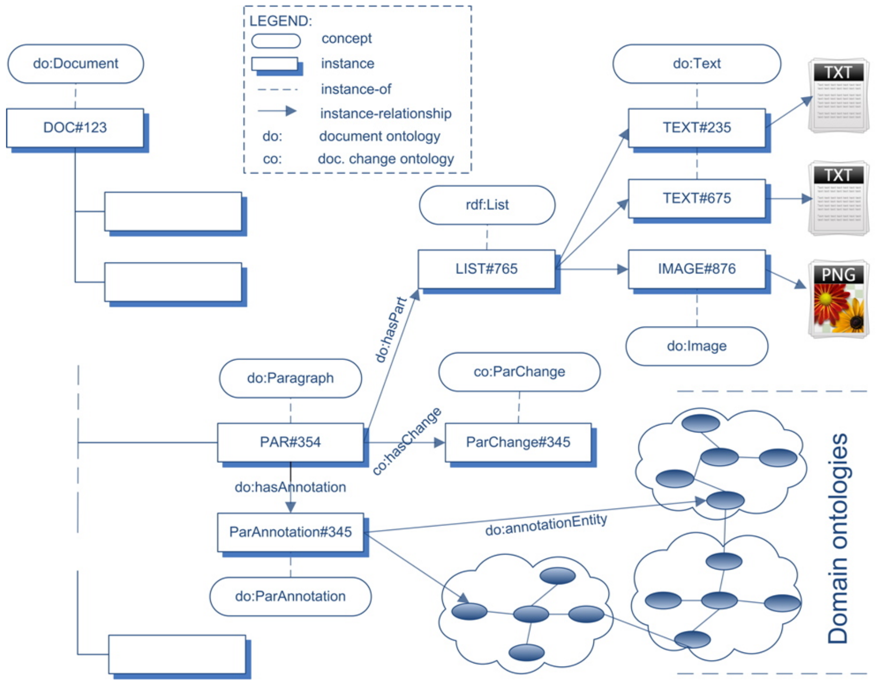

Figure 1: Illustration of a semantic document

nary content [23] and we store binary content into binary content repository and link it to the corresponding RDF nodes. *DNs* of both types *CN* and *KN* are uniquely identified resources which can be reused end included in different semantic documents. In the rest of the paper when we talk about the reuse of document content, we will use a term document content unit (CU) referring to both *CNs* and *KNs*.

2.2 SDArch services

We have designed the SDArch as a service oriented distributed architecture consisting of three layers. From the bottom up, they are data layer, services layer, and presentation layer.

Data layer contains the semantic document repository, the binary content repository and the text index. The semantic document repository is an RDF repositories that stores instances of the SDM. Binary content of document CUs is placed into appropriate content files and then into the binary data repository. Text data of the binary data store is indexed and enables full-text search as a supplement to the semantic search (Section 5.1) over RDF data. SDArch maintains one single text index which is updated every time a new semantic document is added to the repository.

Services layer provides the services that implement the functionalities of SDArch. The services are grouped and accessible through the interface of four web services: User and Group Management WS, Ontology Management WS, Semantic Document Search WS and Semantic Document Authoring WS. Together with the data layer the services layer composes the SDArch middleware which can be delivered and installed as a single software unit.

Presentation layer is the top layer of the SDArch, which provides a user interface for the SDArch services. The layer can contain both the Web-based and desktop applications. Since we decided to use existing, well-established document formats for human-readable browsing of semantic documents, we are mainly focused on extending the GUI of the existing document application suits instead of creating completely new applications. In this way we let users take advantage of semantic documents while still working within familiar environments.

3. SOCIAL CONTEXT ANNOTATION - SCA

One of the main objectives of SDArch is to enable users to easily discover, access, and reuse desired document CUs of different levels of granularity. The discoverability of desired CUs depends strongly on the quality of CU annotations. In this section we present the social context annotation that we have introduced to capture implicit information about the usage of document CUs within a social network.

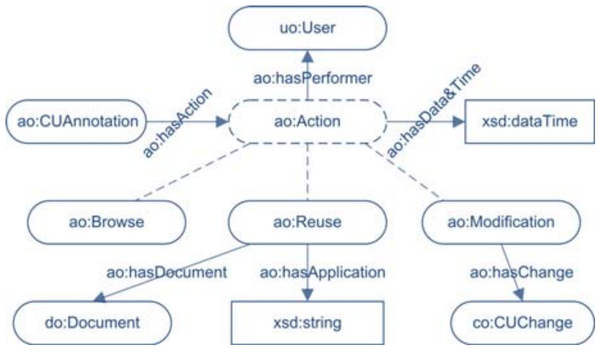

Figure 2: SCA concepts and properties

In order to enable SCA the annotation ontology introduces three concepts: *Browse*, *Reuse* and *Modification* which are all sub-concepts of the abstract *Action* concept (see Figure 2). As their names suggest these concepts represent three different kinds of actions a user can perform with a CU. All the three concepts are characterized by the action date and time and the person who performs the action. *Reuse* also keeps information about document which reuses the CU and the document application used to create/edit the document. Moreover, the *Modification* concept keeps information of the change made to the CU during the modification. The change is represented as an instance of the *DNChange* concept from the change ontology. Every time a user interacts with a document CU, new annotation data is added to the CU. By mining SCA, for each CU from the semantic document repository we can extract the following set of information:

Number of Reuses: $N_{reuses} \in \mathbb{N}$

Number of Modifications: $N_{modifications} \in \mathbb{N}$

Number of Browses: $N_{browses} \in \mathbb{N}$

List of Users: $L_{users} = \{u_1, u_2, ..., u_m\}$ where u_i is the URI of the i_{th} user from the list;

List of Applications: $L_{applications} = \{app_1, app_2, ..., app_p\}$ where app_i is the ID of the i_{th} application from the list;

Reuse Times: $T_{reuses} = \{t_{r_1}, t_{r_2}, ..., t_{r_n}\}$ where $t_{r_i} \in \mathbb{N}$ is a time of the i_{th} reuse of the CU, represented as a number of UNIX timestamps;

Modification Times: $T_{modifications} = \{t_{m_1}, t_{m_2}, ..., t_{m_q}\}$ where $t_{m_i} \in \mathbb{N}$ is a time of the i_{th} modification of the CU, represented as a number of UNIX timestamps;

Browse Times: $T_{browses} = \{t_{b_1}, t_{b_2}, ..., t_{b_r}\}$ where $t_{b_i} \in \mathbb{N}$ is a time of the i_{th} browse of the CU, represented as a number of UNIX timestamps;

In the next section we first discuss SDArch user model and then, in Section 5 we explain how we use information from SCA along with the user profile information to get more personalized search results.

4. USER MODEL OF SDARCH USERS

We have designed SDArch user model to correspond with a general model of a social network user described by FOAF ontology. SDArch user model is described by a user-model ontology. The ontology reuses a set of concepts and properties from the FOAF ontology and introduces a concept *Preference* with a set of properties *preferenceId*, *preferenceLabel*, *preferenceImportance*, *preferenceNumValue* and *preferenceEnumValue* to capturing the user's preferences regarding the choice of document CUs for reuse (see Figure 3). The value of *preferenceImportance* property defines the importance of the preference for the user. Different preferences have different importance for different users. We defined a range of *preferenceImportance* property to be a set of real numbers from the range $[0, 1]$, where 0 means the preference is not important for the user at all and 1 the preference is highly important. A value of the *preferenceImportance* property is specified by the user manually for each preference. Based on the nature of the preference, the preference may have numeric or enumerated (i.e. lists) value. The properties *preferenceNumValue* and *preferenceEnumValue* keep numeric and enumerated preference values respectively. Preference values, both numeric and enumerated, are learned automatically over time by monitoring user activities.

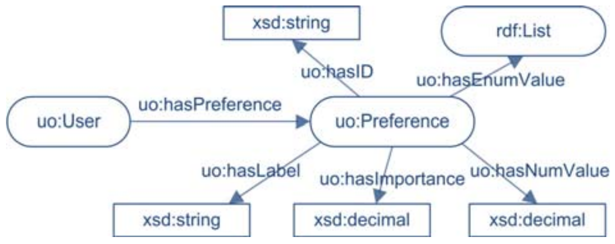

Figure 3: User model ontology - preference part

Preferences that we introduce in the user model correspond closely to the information that can be extracted from the social context annotations. We now list all the preferences and briefly discuss each of them.

1. Preferred Authors ($Pref_1$): this preference specifies an ordered list of document authors who share their documents in the social network. The order is formed based on a number of document CUs that the user has reused from each of the authors. The first author in the list is one from whom the user has reused the most CUs. Initially, the list is empty and gets populated over time. Every time the user reuses a CU, the list is updated and potentially reordered.

2. Preferred Applications ($Pref_2$): this preference specifies an ordered list of document authoring/editing applications. The order of an application in the list is determined by the number of reused CUs which originate from documents that can be browsed and edited by that application. Similarly to the list of preferred authors, the list of preferred applications is initially empty and gets populated over time. Every time the user reuse a CU, the list is updated and potentially reordered.

3. Preferred Number of Reuses ($Pref_3$): this preference specifies if the user rather reuses CUs which have been reused many times or CUs rarely reused. The preference has numeric value from the range $[0, 1]$, 0 meaning that the user always reuses the least reused CUs of all CUs that are retrieved as search results and 1 otherwise.

4. Preferred Number of Modifications ($Pref_4$): this preference specifies if the user rather reuses CUs which have been modified many times, that is, CUs with many versions or rarely modified CUs. The preference has numeric value from the range $[0, 1]$, 0 meaning that the user always reuses CUs which have been modified the least times of all the CUs retrieved as search results and 1 otherwise.

5. Preferred Number of Browses ($Pref_5$): this preference specifies if the user rather reuses CUs which have been browsed many times or rarely browed CUs. The preference has numeric value from the range $[0, 1]$, 0 meaning that the user always reuses CUs which have been browsed the least times of all CUs retrieved as search results and 1 otherwise.

6. Preferred Time of Reuse ($Pref_6$): this preference specifies if the user rather reuses CUs which have been reused recently or CUs reused long time ago. The preference has numeric value from the range $[0, 1]$, 1 meaning that the user always reuses CUs which have been reused the most recently of all CUs retrieved as search results and 0 otherwise.

7. Preferred Time of Modification ($Pref_7$): this preference specifies if the user rather reuses CUs which have been modified recently or CUs modified long time ago. The preference has numeric value from the range $[0, 1]$, 1 meaning that the user always reuses CUs which have been modified the most recently of all CUs retrieved as search results and 0 otherwise.

8. Preferred Time of Browse ($Pref_8$): this preference specifies if the user rather reuses CUs which have been browsed recently or CUs browsed long time ago. The preference has numeric value from a range $[0, 1]$, 1 meaning that the user always reuses CUs which have been browsed the most recently of all CUs retrieved as search results and 0 otherwise.

The way how enumerated preference values (i.e., values of $Pref_1$ and $Pref_2$) evolve over time is straight forward. Every time the user reuses a CU, the information about the author and the application is extracted from the CU's annotations and the corresponding lists of preferred authors and applications are updated and potentially reordered. The way how numeric preference values (i.e., values of $Pref_3$ - $Pref_8$) evolve over time is more complex. We will explain that on the example of $Pref_3$, that is the preferred numbers of reuses. All the other preferences from $Pref_4$ to $Pref_8$ follow the same approach. Let's mark $preferenceNumValue$ of $Pref_3$ as $p \in [0, 1]$. The initial value of p is 0 and then over time it changes based on reused CUs. With N we will denote a number of all CUs that the user has reused so far. Now suppose that the user creates a new document and searches existing semantic documents for a CU to reuse. The search engine returns a set of CUs, $\mathbb{CU} = \{cu_1, ..., cu_n\}$, each of them being reused a certain number of times $\mathbb{N}_{cu} = \{n_1, ..., n_n\} \quad n_i \in \mathbb{N}$. After previewing the retrieved CUs, the user decides to reuse cu_i content unit. The reuse triggers the update of the preference value p and a new value p' is calculated as follows:

$$p' = \frac{N * p + Pref_3(cu_i)}{N + 1} \qquad (1)$$

$Pref_3(cu_i) \in [0, 1]$ is a weight of cu_i for the preference $Pref_3$ in the scope of the retrieved set of content units $\mathbb{CU}$ which is calculated as:

$$Pref_3(cu_i) = \frac{n_i - n_{min}}{n_{max} - n_{min}} \qquad (2)$$

n_{min}, n_{max} represent minimum and maximum elements of the $\mathbb{N}_{cu}$. In the case of $n_{min} = n_{max}$, meaning actually that the user does not have the possibility of different choices regarding the given preference, the value p of the preference stays intact.

5. COLLABORATIVE PERSONALIZED SEARCH

There are two general approaches to personalized search: i) query modification or query expansion based on user profiles [18] and ii) re-ranking the search results using individual profile information [10]. In our approach, namely the collaborative personalized search, we utilize the information extracted from the social context annotations and the preference values from the user profile to re-rank search results so that the new order better corresponds with the user preferences. The core of the collaborative personalized search is the personalized ranking algorithm. In the rest of the section we first briefly describe the search model that we use in the SDArch and then give detailed description of the algorithm.

5.1 The Semantic Search

In the search model namely the semantic search, that we apply in the SDArch, both document CUs and user queries are represented by concept weight vectors. Actually, as a result of the ontology-based annotation of semantic documents, every CU is annotated with a set of ontological concepts from the domain ontology, with each concept having calculated its weight. A concept weigh determines the relevance of the concept for the CU. The weights of all concepts form the concept weight vector of the CU. Similarly, a user query (usually a set of keyword terms) is represented by a concept set and a corresponding concept weight vector. We call such represented queries, semantic queries. Both the ontology-based annotation and the process of forming semantic queries should be done automatically. However, the SDArch does not restrict the possibility of the manual ontology-based annotation as well as forming semantic queries by manually choosing appropriate ontological concepts. It is just a matter of features of the GUI. The important assumption in our search model is that semantic documents and semantic queries share the same domain otologies or that exists the efficient ontology-allignment system. Having both document CUs and queries represented as the concept weight vectors, the similarity between them and then the ranking of search results can be calculated by comparing the deviation of angles between the concept weight vectors [2]. Such ranking of search results is semantically correct but does not consider the user distinctiveness at all.

5.2 Personalized Ranking

In this section we present the personalized ranking algorithm which ranks CUs retrieved by the semantic search to correspond better with the user preferences. The algorithm utilizes the information extracted from the social context annotations (1) - (8) of the retrieved set of content units $\mathbb{CU}$ and the set of preferences $\mathbb{P}$ from the user profile. The output of the algorithm is a re-ranked set $\mathbb{CU}$. The general idea of the algorithm is following. First, the algorithm calculates ranks of each of the CUs for each of the preferences. Then, the final ranks of the CUs are calculated as the sum of their ranks for each preference multiplied by the preference importance factors. The algorithm distinguishes between to types of preferences, the preferences with enumerated values ($Pref_1$, $Pref_2$) and the preferences with numeric values ($Pref_3$,...,$Pref_8$). For each of them it provides corresponding functions for calculating the rank values.

The algorithm (see Algorithm 1) starts by extracting the information (1), (2), ..., (8) for each $cu_i \in \mathbb{CU}$ (line 3-5). For example, in the prototype implementation (Section 6) this is achieved by executing a set of SPARQL queries over the semantic context annotations. Next (lines 6-13), for the preferences with enumerated values, $Pref_1$ and $Pref_2$, for each $cu_i \in \mathbb{CU}$ the algorithm calculates the their rank values. First, (line 8) for each CU it calculates the similarity between the user's preference lists (i.e. the list of preferred authors or the list of preferred applications) and the CU's lists (i.e., the list of the CU's users (4), and the list of the CU's applications (5)).

$$Simil(\overrightarrow{P_j}, \overrightarrow{W_i}) = \frac{\overrightarrow{P_j} * \overrightarrow{W_i}}{|\overrightarrow{P_j}||\overrightarrow{W_i}|} \qquad (3)$$

The similarity (3) is calculated as the cosine of the angle between vectors $\overrightarrow{P_j}$ and $\overrightarrow{W_i}$. Vector $\overrightarrow{P_j} = [p_{j1}, ..., p_{jm}]$,

$j \in \{1,2\}$ is a preference list weight vector of the preference $Pref_j$. For preference $Pref_1$, the weights in the preference list weight vector represent numbers of CUs that the user has reused from each of the authors. For preference $Pref_2$, the weights in the preference list weight vector represent numbers of CUs that the user has reused from documents authored by each of the applications. Vector $\overrightarrow{W_i} = [w_{i1}, ..., w_{im}]$, $w_{ij} = 0 \vee 1$ is the weight vector of the cu_i content unit. For $Pref_1$, vector $\overrightarrow{W_i}$ is formed so that for each author from the preference list who is also in the list of cu_i's users, it has a weight 1 and 0 otherwise. For $Pref_2$ the vector $\overrightarrow{W_i}$ is formed so that for each application from the preference list which is also in the list of cu_i's applications, it has a weight 1 and 0 otherwise. In the next step (line 11), the algorithm calculates rank value of each CU with regard to calculated similarities of the all CUs $Simil(\mathbb{CU}|Pref_j) = \{Simil(\overrightarrow{W_1}, \overrightarrow{P_j}), ..., Simil(\overrightarrow{W_n}, \overrightarrow{P_j})\}$:

$$RankValue1(cu_i|Pref_j) = \frac{Simil(\overrightarrow{W_i}, \overrightarrow{P_j})}{max(Simil(\mathbb{CU}|Pref_j))} \quad (4)$$

The rank value of the content unit with the maximum $Simil$ is 1 and the rank values of the others fell in the range (0,1).

Algorithm 1

```
91: INPUT $\mathbb{CU} = \{cu_1, ..., cu_n\}, \mathbb{P} = \{Pref_1, ..., Pref_8\}$
92: OUTPUT $\mathbb{CU}'$ {re-ranked $\mathbb{CU}$ set}
93: for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
94:         FIND: (1),(2),...,(8) by searching SCA
95: end for
96: for $j = 1$ to 2 do
97:         for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
98:                 $Simil(cu_i|Pref_j)$
99:         end for
910:        for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
911:                $RankValue1(cu_i|Pref_j)$
912:        end for
913: end for
914: for $j = 3$ to 8 do
915:        for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
916:                $Diff(cu_i|Pref_j)$
917:        end for
918:        for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
919:                $RankValue2(cu_i|Pref_j)$
920:        end for
921: end for
922: for all $cu_i$ such that $cu_i \in \mathbb{CU}$ do
923:        $RankValue(cu_i)$
924: end for
925: SORT $\mathbb{CU}$ in decreasing order of $RankValues$
```

Next (lines 14-21), for the preferences with numeric values, $Pref_3 - Pref_8$ for each $cu_i \in \mathbb{CU}$ the algorithm calculates corresponding rank values. First (line 16), it calculates the difference $Diff(cu_i|P_j)$ between p_j (i.e., a preference value that comes from the user profile) and $Pref_j(cu_i)$ (i.e., weight of cu_i regarding the preference $Pref_j$ in the scope of the retrieved set of content units $\mathbb{CU}$

$$Diff(cu_i|Pref_j) = |p_j - Pref_j(cu_i)| \quad (5)$$

$Pref_j(cu_i)$ is calculated by the formula (10). For $Pref_3$, $Pref_4$ and $Pref_5$ the set N_{cu} represents numbers of reuses, modifications and browses of each CU respectively. For $Pref_6$, $Pref_7$ and $Pref_8$ the set N_{cu} represents times (i.e., number of time-stamps) of the last reuse, last modification and last browse of each CU respectively. In the next step (line 19) the algorithm calculates rank value of each CU with regard to calculated differences of the all CUs $Diff(\mathbb{CU}|P_j) = \{Diff(cu_1|P_j), ..., Diff(cu_n|P_j)\}$:

$$RankValue2(cu_i|Pref_j) = \frac{min(Diff(\mathbb{CU}|P_j))}{Diff(cu_i|P_j)} \quad (6)$$

The rank value of the content unit with the minimum $Diff$ is 1 and the rank values of the others fell in a range (0,1).

Finally, when the rank values of the all CUs regarding each of the preferences are calculated, the algorithm calculates the general rank value of each of the CUs (line 23):

$$RankValue(cu_i) = \sum_{j=1}^{2} RankValue1(cu_i|Pref_j) * if_j + \sum_{j=3}^{8} RankValue2(cu_i|Pref_j) * if_j \quad (7)$$

where if_j represents the importance factor of the preference $Pref_j$. The value of this factor comes from the user profile (Section 4). At the end, the CUs are sorted by their general rank values in the decreasing order.

6. PROTOTYPE IMPLEMENTATION

We have implemented the first version of the SDArch prototype and we are currently preparing it for the first round of the evaluation. As we described in the Section 2.2, the SDArch is a distributed service oriented architecture (SOA) with three layers: the data layer, the service layer and the presentation layer. The rest of the section provides some implementation details of each of the layers.

The data layer contains the semantic documents repository, binary data store and text index. In the prototype we have implemented the semantic document repository based on Sesame 2 [22] RDF repository. For reading, writing and searching RDF statements from the semantic document repository, we have used SemWeb [21], the SemanticWeb/RDF library written for Mono and .NET. For the implementation of the text index we have used the Apache Lucene Information Retrieval library [1].

The service layer is implemented by using Windows Communication Foundation (WCF) programming model. The internal functionalities of each of the services are grouped in functional modules which are implemented in Component Object Model (COM) technology. The communication among the services and between the services and applications from the presentation layer is done by exchanging SOAP messages over HTTP. Together with the data layer the service layer composes the SDArch middleware which can be delivered and installed as a software unit. We have deployed the prototype of the SDArch middleware on the university research server and currently all the services are publicly available at [20].

The presentation layer of the SDArch is the platform independent and can contain both desktop and the Web-based applications. As a long-term goal we plan to develop a completely new application suite for browsing, authoring and editing the semantic documents. For the current version of the prototype our strategy was to extend some of the well known office suites to add support for the semantic documents. The reason for that was to leave users still working in familiar environments while take the advantage of the

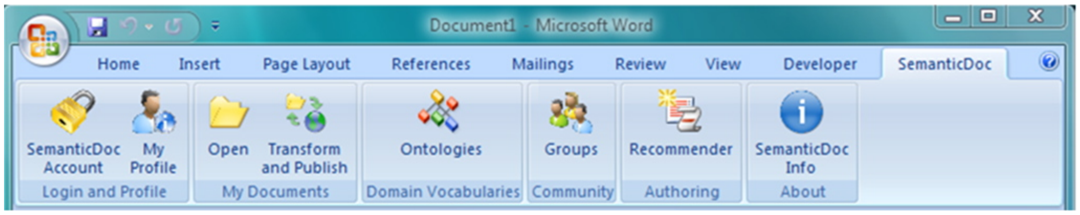

Figure 4: SemanticDoc Tools

semantic documents. Moreover, we want to show the interoperability of the proposed SDM and the existing office document formats. We have chosen to extend the MS Office 2007 mostly because of its wide usage and popularity. Accordingly, we have developed an MS Office add-in that we named 'SemanticDoc'. In the next section we discuss the add-in in more detail.

7. SEMANTICDOC ADD-IN

The SemanticDoc extends MS Office GUI with a set of tools that enable users to deal with semantic documents. In other words it provides the access to the SDArch services from MS Office (i.e., MS Word and MS PowerPoint). Since the SDArch enables users to share their documents and to form a social network around shared documents, by SemanticDoc we turned MS Office into a social environment. The SemanticDoc tools can be accessed through SemDocument ribbon tab (see Figure 4) and are grouped into several toolboxes. The design of the added ribbon tab follows the main design principles of the MS Office. In the rest of the section we briefly outline main characteristics of each of the SemanticDoc tools. More detailed information, snapshots and demos of the tools can be found at SDArch Web page [20].

Login and profile tools: In order to participate in the SemanticDoc social network and share semantic documents the user has to open an account by providing a valid OpenID [17]. By opening the account the user gets an empty, default profile. 'My Profile' tool enables the user to manage her/his profile. Besides the basic information such as name, address, interests and projects, the user can specify the list of people from the network with whom she/he wants to share the documents. Moreover, for each of the preference (Section 4) she/he can specify the importance factor. The preference values are learned automatically and the user can not set them manualy, which is in accordance to the proposed user mode. The profile is stored on the server side and the access and edit of the profile data is done by invoking the corresponding methods of the User and Group Management WS.

Group manager: Participants of the social network can form smaller groups of users interested in particular topics. By using the group manager tool every user can create a new group by specifying the topic of interest with some background information (e.g., shot description and the Web references). Currently, there is no any restriction to join a group; all groups are available to everybody. The idea of forming groups was not to get many separated document repositories, but to get documents better classified within the same repository. Moreover, group documents share the same conceptualization of the topic's domain, i.e., domain ontologies, which is the key assumption of the semantic search. The users of a group use the same ontology/ies while creating semantic documents to be shared within the group. To access and manage the group information the group manger tool invokes the corresponding methods of the User and Group Management WS.

Ontology manager: Domain ontologies play one of the key roles in the semantic documents. In the semantic search a concept weight vector of every CU is compared with a query concept weight vector (Section 5.1) and based on the calculated similarities CUs are ranked within the search results. The ontology manager tool enables users to manipulate with ontologies that are shared in the social network. The user can list all the ontologies present in the network as well as ontologies related to a specific group. For each ontology the user can browse the available metadata (e.g., creator, short description and creation time and data) and the lists of the ontology's concepts and properties. Moreover, if the ontology repository does not contain appropriate ontologies for the user's documents, the user can upload new ontology to the ontology repository. The tool invokes the methods of the Ontology Management WS to access and manipulate the shared ontologies.

Document transformer and publisher: This tool enables the user to transform the active office document (i.e., the document opened in MS Word or MS PowerPoint) into the semantic document and to publish it into shared semantic document repository. Before initiating the transformation the user specifies the ontology/ies that describe the domain of the document's topic. The quality of the semantic document annotation depends strongly on the quality of the chosen ontologies. Moreover, the user specifies if she/he wants to publish the document to the specific group or not. If so, the ontlgies have to be chosen among the group's ontologies. The transformation and publishing is done by invoking appropriate methods of the Semantic Document Authoring WS.

Recommender: The Recommender tool provides the user with the interface to search the semantic document repository. The user can search the whole repository or documents belonging to some groups. The search starts by the user specifying keyword query which is then transformed into the corresponding semantic query. The tool has the auto-completion support that helps users while specifying the keyword query. The terms offered by the auto-completion are concept labels from domain ontologies which had been used for the semantic annotation of the documents present in the repository. The use of suggested terms results in a better quality of generated semantic queries. After the semantic search and the personalized ranking the recommender shows the list of results (i.e., CUs) to the user. For each of retrieved CUs the user can see additional information that come from its annotation data (e.g., the number of reuses, the number and list of users, the list of documents in which it appears and the number of versions). Moreover, if the CU has many versions, the user can navigate among them. Once decides which CU to reuse, the user only needs one mouse click to add the CU to the active document. The tool also tracks the user activities and updates her/his profile. For each of the above mentioned features the tool invokes appropriate methods of the Semantic Document Search WS.

8. EVALUATION PLANS

The usability evaluation of the SDArch is tied closely to the prototype development. As we already stated (Section 6) we have deployed the current prototype and initiated the social network around it. In the first phase our target user group will be our colleges from our two universities (i.e., students, master students and PhD students) as well as some friends from other universities who have already shown interests in our work. Depending on the speed of the network growth and available hardware resources, in the second phase we will also consider the possibility of advertising our social network through some of the popular social networking services such as Facebook, MySpace and Friendster. In order to evaluate the usability of the SDArch we plan to conduct a number of observational and case studies.

The goal of the observational studies will be to produce quantitative data about the social network such as how a number of users, a number of shared documents and a number of reused document CUs change over time. Moreover, these studies will allow us to monitor some of the user profiles and see how their preference values change over time. Based on this we will be able to draw some conclusions of the effectiveness of the proposed ranking algorithm. For example, if after awhile preference values start to be more stable it will clearly show the correctness of the algorithm.

The case studies will be classified into two main groups. The first group will contain long-term case studies (up to six months) that will try to recruit as many network users as possible. The user will be asked to perform a set of tasks on a given data set (i.e., a set of shared semantic documents) and then the feedback will be collected through web-based online questioners. The second group will contain short (one day) case studies conducted with smaller groups of participants. The objective of such case studies will be to obtain data about users' performance when they perform the tasks of the usability test, for example, the time users take to complete a specific task, the number of tasks of various kinds that can be completed within a given time limit, and the number of user errors. Moreover, these kind of case studies will let us be closer to the users and better realize their likes, dislikes, needs and understanding of the system, by talking to them, observing them using the monitoring systems or letting them answer questions verbally or in written form.

9. CONCLUSION

In this paper we presented a new document architecture, namely SDArch which we have developed to improve the sharing of office documents' content among people in online social networks. The architecture is based on the new document model, which enables the annotation of document content of different levels of granularity with semantic and social context annotations. The social context annotation is introduced to capture implicit information about the usage of document content in the context of the social network. This information is then used in the collaborative personalized search along with user preferences to produce more personalized search results. The core of the collaborative personalized search is the personalized ranking algorithm that we presented in the paper. Moreover, SDArch prototype that implements the algorithm and the application/tool example that we developed on the top of the prototype are also discussed. At the end of the paper, we briefly outline our plans for the usability evaluation of the proposed architecture.

10. REFERENCES

[1] Apache Lucene. http://lucene.apache.org/java/docs/.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, June 1999.

[3] J. G. Breslin and S. Decker. The Future of Social Networks on the Internet: The Need for Semantics. *IEEE Internet Computing*, 11(6):86–90, 2007.

[4] D. Dori. The Representation of Document Structure: a Generic Object-Process Analysis. *Handbook on Optical Character Recognition and Document Image Analysis, Word Scientific*, 1995.

[5] DITA. http://www.oasis-open.org/committees/dita.

[6] Documentum. http://www.documentum.com/.

[7] Docushare. http://docushare.xerox.com/.

[8] Doublin Core. http://dublincore.org/.

[9] Interwoven. http://www.interwoven.com/.

[10] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *Proc of the 11th ICIKM*, pages 558–565. ACM Press, 2002.

[11] LOM. http://ltsc.ieee.org/wg12/.

[12] L. Nelson, D. Smetters, and E. Churchill. Keyholes: Selective sharing in close collaboration. In *Proceedings of CHI*, 2008.

[13] S. Nešić. Semantic Document Model to Enhance Data and Knowledge Interoperability. *Annals of Information Systems, Springer*, (In Press).

[14] S. Nešić, D. Gašević, and M. Jazayeri. Semantic document management for collaborative learning object authoring. In *8th IEEE International Conference on Advanced Learning Technologies*, pages 751–755, Santander, Spain, 2008.

[15] OASIS Consortium, Open Document Format for Office Application. www.oasis-open.org/committees/office/.

[16] Office Open XML:. http://openxmldeveloper.org/.

[17] OpenId. http://openid.net/.

[18] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Tumbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Communications of ACM*, 45:50–55, 2002.

[19] A. Prakash. Group editors. *In Computer Supported Co-operative Work*, 7:103–133, 1999.

[20] SDArch. http://www.inf.unisi.ch/phd/nesic/sdms/.

[21] SemWeb. http://razor.occams.info/code/semweb/.

[22] Sesame 2. http://www.openrdf.org/.

[23] M. Sintek, S. Scerri, and S. Handschuh. Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL. In *Proc. of the 4th ESWC*, pages 594–608, 2007.