On Designing the SONCA System

Linh Anh Nguyen[1] and Hung Son Nguyen[2]

[1] Institute of Informatics
[2] Institute of Mathematics
University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{nguyen,son}@mimuw.edu.pl

Abstract. The SYNAT project aims to develop a universal, open hosting and communication platform for network knowledge resources for science, education and open information society. The stage B13 of this project aims to develop methods and algorithms of semantic indexing, classification and retrieval using dictionaries, thesauri and ontologies as well as methods of processing and visualizing results. The methods and algorithms aim to support the dialogue with the repositories of text and multimedia resources gathered on some servers. To realize the objectives of the stages B13 and B14 of the SYNAT project we plan to develop and implement a system called SONCA (Search based on ONtologies and Compound Analytics).

We present ideas and proposals for the SONCA system. The main idea is to allow combination of metadata-based search, syntactic keyword-based search and semantic search, and to use ranks of objects. Semantic search criteria may be keywords, concepts, or objects (for checking similarity). Search criteria based on metadata play the role of exact restrictions, while syntactic keywords and semantic search criteria are fuzzy restrictions. To enable metadata-based search, an appropriate document representation is used. To enable syntactic keyword-based search, each document (object) is stored together with information about the terms occurring in its text attributes. The terms are normalized and only important ones are stored. To enable semantic search, the representation of each document (object) is extended further with the most important concepts that characterize the document. Such concepts belong to the main ontology of SONCA.

We provide an abstract model for the SONCA system, an instantiation of that model, some ideas for the user interface of SONCA as well as proposals for increasing efficiency of the query answering process.

1 Introduction

The SYNAT project [13], realized in 2010-2013 by 16 scientific institutions, aims to develop a universal, open hosting and communication platform for network knowledge resources for science, education and open information society. Our institution (MIMUW[1]) realizes two out of 52 stages of the SYNAT project. One

[1] Faculty of Mathematics, Informatics and Mechanics, University of Warsaw.

R. Bembenik et al. (Eds.): Intelligent Tools for Building a Scient. Info. Plat., SCI 390, pp. 9–35.
springerlink.com

of them is to develop methods and algorithms of semantic indexing, classification and retrieval using dictionaries, thesauri and ontologies as well as methods of processing and visualizing results. The methods and algorithms aim to support the dialogue with the repositories of text and multimedia resources gathered on some servers. To realize the objectives we plan to develop and implement a system called SONCA (Search based on ONtologies and Compound Analytics). The additional aim is to test our methods and prepare for an integration of our system with the systems developed by the other partners of the SYNAT project.

Here are some assumptions for the SONCA system:

- The system offers searching objects like publications, authors, institutions and other related information. Documents may be written in English, Polish or other languages.
- The search engine extends metadata-based search and keyword-based search with semantic search.
- The system is general-purpose, running on a large repository of gathered objects (articles, institutions, authors ...). It does not assume to rely on specific domains only.
- The system is extensible. It can work with different types of domain-specific knowledge.

In this paper, we present ideas and proposals for the searching track of the SONCA system. The main idea is to combine different search approaches including metadata-based search, syntactic keyword-based search and semantic search, and to use ranks of objects. Semantic search criteria may be keywords, concepts, or objects (for checking similarity). Search criteria based on metadata play the role of exact restrictions, while syntactic keywords and semantic search criteria are fuzzy restrictions. To enable metadata-based search, an appropriate document representation is used and the database of SONCA is designed accordingly. To enable syntactic keyword-based search, each document (object) is stored together with information about the terms occurring in its text attributes. The terms are normalized and only important ones are stored. To enable semantic search, the representation of each document (object) is extended further with the most important concepts that characterize the document. Such concepts are specified by the main ontology of SONCA, which may consist of several subontologies.

1.1 On the Web Ontology Language OWL

OWL is a standardized Web ontology language, with the first and second versions recommended by W3C in 2004 [14] and 2009 [15], respectively. These versions, OWL 1 and OWL 2, have some sublanguages (species/profiles):

- OWL 1 DL and OWL 2 DL support those users who want maximum expressiveness without losing computational completeness.

- OWL 1 Lite, OWL 2 EL, OWL 2 QL and OWL 2 RL are restricted versions with PTIME data complexity of the "DL" sublanguages.
- OWL 1 Full and OWL 2 Full support those users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

OWL 1 DL is based on the description logic $\mathcal{SHOIN}$ [11], while OWL 2 DL is based on a more expressive description logic $\mathcal{SROIQ}$ [10]. Other well-known description logics (DLs) are $\mathcal{ALC}$, $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$. DLs are fragments of classical first-order logic and are variants of modal logics, used to describe the domain of interest by means of individuals, concepts and roles. A concept is interpreted as a set of individuals, while a role is interpreted as a binary relation between individuals. The satisfiability checking problem is EXPTIME-complete in $\mathcal{ALC}$ and $\mathcal{SHIQ}$; NEXPTIME-complete in $\mathcal{SHOIN}$ and $\mathcal{SHOIQ}$; and N2EXPTIME-complete in $\mathcal{SROIQ}$.

According to the recent survey [28], the third generation ontology reasoners that support $\mathcal{SHIQ}$ or $\mathcal{SROIQ}$ are FaCT, FaCT++, RACER, Pellet, KAON2 and HermiT. The reasoners FaCT, FaCT++, RACER and Pellet are based on tableaux, KAON2 is based on a translation into disjunctive Datalog, and HermiT is based on hypertableaux. That is, all of the listed reasoners except KAON2 are tableau-based. According to [30], KAON2 provides good performance for ontologies with rather simple TBoxes, but large ABoxes[2]; however, for ontologies with large and complex TBoxes, existing tableau-based reasoners still provide superior performance. The current version of KAON2 does not support nominals and cannot handle large numbers in cardinality statements. The reasoners FaCT, FaCT++, RACER and Pellet use traditional tableau decision procedures like the ones for $\mathcal{SHIQ}$ [12], $\mathcal{SHOIQ}$ [11], $\mathcal{SROIQ}$ [10]. These procedures use backtracking to deal with disjunction (e.g., the union constructor). Their search space is an "or"-tree of "and"-trees. Despite advanced blocking techniques (e.g., anywhere blocking), their complexities are non-optimal (e.g., NEXPTIME instead of EXPTIME for $\mathcal{SHIQ}$; N2EXPTIME instead of NEXPTIME for $\mathcal{SHOIQ}$; and N3EXPTIME instead of N2EXPTIME for $\mathcal{SROIQ}$). Similarly, the decision procedure of HermiT also has a non-optimal complexity [6]. To obtain optimal and efficient tableau decision procedures for DLs one may consider the optimization techniques developed in [7,8,38,32,40,39,36,37,35].

The profiles OWL 2 EL, OWL 2 QL and OWL 2 RL are restricted sublanguages of OWL 2 FULL with PTIME data complexity. They are based on the families of DLs $\mathcal{EL}$ [1,2], DL-Lite [3] and DLP (Description Logic Programs) [9], respectively. As fragments of DLs with PTIME data complexity there are also Horn-$\mathcal{SHIQ}$ [25], Horn-$\mathcal{SROIQ}$ [42], and the deterministic Horn fragments of $\mathcal{ALC}$ and regular description logics [31,33]. Formalisms that combine a DL with a rule language were studied, among others, in [27,4,26,29,5].

[2] A TBox is a set of terminology axioms, including role axioms. In the literature of DLs, sometimes role axioms are grouped into an RBox. An ABox is a set of individual assertions.

1.2 Using Ontologies for the SONCA System

Ontologies can be used for:

- increasing flexibility in using different names for the same attribute/relation
- understanding the meanings of terms occurring in documents.

The first purpose is useful for the user interface (e.g., for transforming a user's query to a formal query) and for integrating modules implemented by different partners of the SYNAT project. Ontologies in OWL or similar languages can be used for this purpose.

The second purpose is useful for:

- matching or measuring similarity between:
  - term – concept – document
  - concept – document
  - document – concept – document
- answering queries
- the user interface (for navigation and improving queries).

The second purpose seems more important (or at least harder to be fulfilled) than the first one. In our opinion, ontologies in OWL are not suitable for the second purpose due to the following reasons:

- Most documents of the SONCA system (like publications and webpages) are unstructured or semi-structured. They are PDF files, (OCR'ed) texts, HTML or XML documents, but not OWL-based documents.
- Reasoning in OWL DL has a very high complexity.
- The knowledge base of a very large system like SONCA may be inconsistent. Despite that paraconsistent reasoning has been studied for DLs (e.g., in [41,34]), the techniques may be not advanced enough.

For these reasons, we propose to use a thesaurus extended with fuzzy relationships between the concepts to "understand" the meanings of terms occurring in documents. We call that thesaurus the *main ontology* of SONCA. It may consist of several subontologies. We can construct it from sources like DBpedia [16], WordNet [17] and plWordNet [18]. In this paper we address the construction of this main ontology and its use for query answering. We omit ontologies in OWL, which does not mean they are not useful for SONCA and SYNAT.

1.3 The Contributions and the Structure of This Work

In this paper we provide an abstract model for the SONCA system, an instantiation of that model, some ideas for the user interface of SONCA as well as proposals for increasing efficiency of the query answering process.

The abstract model includes notions and requirements of:

- the collection of terms
- the main ontology

– three abstract document-representations
– abstract queries (together with their meanings).

The second document representation extends the first one with information about terms occurring in the document, and the third document representation extends the second one with *ObjectRank* and *RelatedConcepts* of the document.

Our instantiation of that abstract model provides definitions and detailed descriptions for the model, including:

– a function computing concepts similar to a given concept
– a method of computing *ObjectRank* of an object
– a function computing *RelatedConcepts* of an object
– several functions measuring similarity between terms, concepts and objects.

The rest of this paper is structured as follows. In Section 2 we present our abstract model of SONCA. In Section 3 we present our instantiation of that model. Section 4 is devoted to the user interface of SONCA. Section 5 contains our proposals for increasing efficiency of the query answering process. Section 6 contains some concluding remarks. Appendix A is devoted to DBpedia and its use for the SONCA system.

2 An Abstract Model for SONCA

2.1 Terms, Concepts and Ontologies

A *term* is a string representing a word or a phrase of a few words, which has been normalized using some algorithm (e.g., a stemming algorithm). The maximal number of words in a term is parameterized by *MaxTermLength* and chosen appropriately. The importance of a term t_i in a text d_j (which can be an element of a document) is usually measured by *tf-idf*$_{i,j}$ (term frequency - inverse document frequency), which is defined as $tf_{i,j} \times idf_i$, where $tf_{i,j}$ stands for the frequency of t_i in d_j, and idf_i stands for the general importance of term t_i in the set of all documents of the system. There are different possible definitions of $tf_{i,j}$ and idf_i [19].

The system contains information about important terms, which is a collection TERMS of pairs (t_i, idf_i). Only terms t_i with high enough idf_i will be stored in this collection. The threshold for *idf* is parameterized by *MinIdf* and chosen appropriately.

Consider, for example, the text "information and decision systems". As terms in this text one can consider "information", "and", "decision", "system", "decision system", "information and decision", "information and decision system". Notice that we exclude the phrases "information and", "and decision", "and decision systems". Among the listed terms, "and" probably has very low *idf* value and is not stored.

The system implements a function $Terms(d_j, h)$ that, for a given text d_j and a *tf-idf* threshold h, returns information about the most important terms t_i occurring in d_j in the form of a collection of pairs $(t_i, tf_{i,j})$ with *tf-idf*$_{i,j} \geq h$.

The system also implements a function $Sim_{st}(s,t)$, which measures the similarity between a string and a term.

A *concept* is understood here as a meaning, which is specified, amongst others, by an ID and a list of pairs $(label, weight)$, where $label$ is a term and $weight$ is a real number in $(0,1]$ representing the accuracy that the term has that meaning.

The system implements a function $Sim_{tc}(t,c)$, which measures the similarity between a term t and a concept (with ID) c, i.e., the accuracy that the term t has the meaning c.

An *ontology* is a compact data structure containing information about concepts and the relationships between them. For a given ontology, the system implements a function $SimilarConcepts(c,n,s)$, where c is a concept (ID), n is a natural number and s is a real number in $(0,1]$. This function returns a list of no more than n pairs (c_i, s_i) such that $c_1, c_2, \ldots$ are the concepts of the ontology most similar to c and $s_i \geq s$ is the similarity between c and c_i. (This similarity measure between concepts need not be a commutative operator.) The system also implements a function $PossibleConcepts(t)$ which returns the set of concepts which may correspond to the term t.

2.2 Abstract Document-Representations

The database of SONCA contains a set of objects, which may be publications, authors, institutions, etc. Each object is represented as a tree-like document (like an XML document). The original form of a publication may be an image. However, in the current paper we assume that such a document has been OCR'ed and we have its text-based representation, possibly with references to images representing the whole paper and its figures. Images are not treated as documents; they are stored in a specific way and only used for visualization in user interfaces, but not for the search process. Using the terminology of XML, an object is an element; each complex element has a name, possibly some attributes, and contents, which are simple values (e.g., texts, numbers, references) or complex elements; each attribute has a name and a simple value. Elements and attributes of XML are similar, but with the differences that, values of attributes are simple (i.e. without structure), attributes are functional, and the order of attributes are inessential, while the order of elements is essential.

In the current paper, we use the term "attribute" to refer to both "attribute" and "element" as the ones of XML. When necessary we use adjectives like simple, complex, functional to describe attributes. In our settings, the order of attributes with the same name in a given scope is essential. Documents and attributes have types, which are simple or complex. A value of a simple type is either a literal (e.g., a number, a date, a reference, or a short text) or a long text.

We give below an exemplary document, which consists of pairs of the form (an attribute name: a value or a list of values), where values may be complex structures:

```
ID: 1024
Type: 1
```

```
Title: "Rudiments of rough sets"
AuthorID: 1121, 1432
JournalID: 2124
Volume: 177
Number: 1
Year: 2007
Pages: "3-27"
Abstract: ...
Keyword: "vague concepts", "information and decision systems", ...
Section:
  (Section 1):
     Title: "Introduction"
     Text: ...
     Citation:
        (Citation 1):
           Reference: 3243, 3324, 4532
           Text: ...
        ...
  ...
...
```

We distinguish *external objects* as the ones with ID (like publications and authors) from *internal objects* that do not have ID (like sections in a paper). By a *document*, in a narrower sense, we mean a publication, but in a broader sense, we mean any external object.

By ADR1 we call the above discussed abstract representation of documents.

By ADR2 we call the representation that extends ADR1 with information about the most important terms occurring in a document. In particular, in ADR2 each simple attribute of a document whose value is a text d_j will be stored together with $Terms(d_j, h)$, where the *tf-idf* threshold h is parameterized ($h = MinTfIdf$[full path of the attribute]) and chosen for each attribute in an appropriate way. Furthermore, we assume that, for each document x in ADR2, the set of all normalized words of text attributes of x, denoted by $words(x)$, and the set of all terms of text attributes of x, denoted by $terms(x)$, are stored together with x.

By ADR3 we call the representation that extends ADR2 for an external object with information about *ObjectRank* and the most important concepts related with that object. This representation is discussed below.

Some attributes link different external objects (e.g., authorship and references). In this way, objects are connected and create a graph like a web. *ObjectRank* serves the same role as *PageRank* (see, e.g., [20]).

In ADR3, each publication is stored with a list of pairs $(c_1, s_1), \ldots, (c_n, s_n)$, where $c_1, \ldots, c_n$ is the list of the concepts most related to the topic of the publication, and each s_i represents the similarity between c_i and the topic, in the scale $(0, 1]$. Only pairs (c_i, s_i) with high enough s_i are stored. That is, we take the pairs (c_i, s_i) with s_i greater than or equal to a parameter *MinConSimForDoc*. The number of such pairs is bounded by a limit parameterized by *MaxConceptsForDoc*, which is chosen appropriately.

Similarly, other external objects like authors, journals or conferences are stored with a list of the most important related concepts (representing the fields the author works in or the fields the journal or conference involves).

We use *RelatedConcepts* as the attribute representing the list of the most important related concepts of a given document in ADR3.

Documents are added to the system at some stage in ADR1. They are then extended to ADR2, and after that to ADR3, possibly in a few stages (for improving the quality of the additional information).

2.3 Abstract Queries

In this subsection we define a general form for queries in SONCA. Such a form is not provided by the user, but by the user interface module. It is a form which requires preprocessing before the query is passed to the search module.

We assume that the system implements the following functions:

- $Sim^{syn}_{so}(x, y)$ is a function that maps a pair consisting of a phrase (string) x and an external object (e.g., a publication) y to a real number in $[0, 1]$ representing the syntactic similarity between x and y.
- $Sim^{sem}_{so}(x, y)$ (resp. $Sim_{co}(x, y)$ or $Sim_{oo}(x, y)$) is a function that maps a pair consisting of a phrase (resp. a concept or an external object[3]) x and an external object (e.g., a publication) y to a real number in $[0, 1]$ representing the semantic similarity between x and y.

Simple attributes whose values are expected to be long texts (e.g., the text of a section) are called *long-text attributes*. The part of the database of SONCA without values of long-text attributes is called the *metadata* of the system.

A *general query* is a tuple

$$\Phi = (\Phi_{Metadata}, \Phi_{SynPhrases}, \Phi_{SemPhrases}, \Phi_{Concepts}, \Phi_{SimObjects}, \Phi_{weighting})$$

where

- $\Phi_{Metadata}$ is a logical formula over the metadata of the system, with only one free variable standing for the queried object; the formula can be an SQL-like query, a first-order logic formula, a fixpoint formula, a rule-based query (e.g., a Datalog or stratified Datalog$^\neg$ query), a description-logic-like query, etc
- $\Phi_{SynPhrases}$ is
  - either a set of weighted phrases, i.e., a set of pairs (s_i, w_i), where each s_i is a string and the corresponding w_i is a weight for s_i (the phrases from $\Phi_{SynPhrases}$ are syntactically matched with queried objects)
  - or a Boolean combination using $\wedge$ and $\vee$ of fuzzy expressions of the form $Sim^{syn}_{so}(s_i, x)$ with the same variable x, where each s_i is a phrase and x stands for the queried object; in this case, $\Phi_{SynPhrases}$ is a fuzzy function with argument x

[3] e.g., a publication or an author.

- $\Phi_{SemPhrases}$ can be described similarly as for $\Phi_{SynPhrases}$ except that the word "syntactically" is replaced by "semantically" and $Sim_{so}^{syn}(s_i, x)$ is replaced by $Sim_{so}^{sem}(s_i, x)$
- $\Phi_{Concepts}$ is
  - either a set of weighted concepts (used to matching with queried objects)
  - or a Boolean combination using $\wedge$ and $\vee$ of fuzzy expressions of the form $Sim_{co}(c_i, x)$ with the same variable x, where each c_i is a concept and x stands for the queried object; in this case, $\Phi_{Concepts}$ is a fuzzy function with argument x
- $\Phi_{SimObjects}$ is
  - either a set of weighted objects (used to "semantically" matching with queried objects w.r.t. similarity of topics/fields)
  - or a Boolean combination using $\wedge$ and $\vee$ of fuzzy expressions of the form $Sim_{oo}(o_i, x)$ with the same variable x, where each o_i is an external object and x stands for the queried object; in this case, $\Phi_{SimObjects}$ is a fuzzy function with argument x
- $\Phi_{weighting}$ is a (monotonic) function from $[0, 1]^4 \times \mathbb{R}$ to $\mathbb{R}$.

Notice that the given form of queries enables combination of searching based on metadata, searching based on syntactic keywords and "semantic" searching.

The user interface module may interact with the user to improve the query by eliminating $\Phi_{SemPhrases}$, $\Phi_{SimObjects}$ and modifying $\Phi_{Concepts}$. It is more likely that the user will give a list of phrases (resp. concepts, objects) instead of a set of weighted phrases (resp. concepts, objects). In that case, the system uses an appropriate conversion method. (Note that the first element in a list usually has a greater weight than the others.)

The result of a query Φ is specified as follows:

- Assume an appropriate definition for the fuzzy Boolean operators $\wedge$ and $\vee$. For example, $(x \wedge y) = (x.y)$ and $(x \vee y) = (x + y - x.y)$. An alternative definition is $(x \wedge y) = min(x, y)$ and $(x \vee y) = max(x, y)$.
- Let X be the set of all external objects x (with an appropriate type) of the system such that $\Phi_{Metadata}(x)$ holds.
- For $\Phi_{SynPhrases} = \{(s_1, w_1), \ldots, (s_n, w_n)\}$ and $x \in X$, define the matching score of $\Phi_{SynPhrases}$ for x as follows:

$$ms_s^{syn}(\Phi_{SynPhrases}, x) = \frac{\Sigma_{i=1}^{n}(w_i \times Sim_{so}^{syn}(s_i, x))}{\Sigma_{i=1}^{n} w_i}.$$

- If $\Phi_{SynPhrases}$ is a fuzzy function then define

$$ms_s^{syn}(\Phi_{SynPhrases}, x) = \Phi_{SynPhrases}(x)$$

- For $\Phi_{SemPhrases} = \{(s_1, w_1), \ldots, (s_n, w_n)\}$ and $x \in X$, define the matching score of $\Phi_{SemPhrases}$ for x as follows:

$$ms_s^{sem}(\Phi_{SemPhrases}, x) = \frac{\Sigma_{i=1}^{n}(w_i \times Sim_{so}^{sem}(s_i, x))}{\Sigma_{i=1}^{n} w_i}$$

– If $\Phi_{SemPhrases}$ is a fuzzy function then define

$$ms_s^{sem}(\Phi_{SemPhrases}, x) = \Phi_{SemPhrases}(x)$$

– For $\Phi_{Concepts} = \{(c_1, w_1), \ldots, (c_n, w_n)\}$ and $x \in X$, define the matching score of $\Phi_{Concepts}$ for x as follows:

$$ms_c(\Phi_{Concepts}, x) = \frac{\Sigma_{i=1}^n (w_i \times Sim_{co}(c_i, x))}{\Sigma_{i=1}^n w_i}$$

– If $\Phi_{Concepts}$ is a fuzzy function then define

$$ms_c(\Phi_{Concepts}, x) = \Phi_{Concepts}(x)$$

– For $\Phi_{SimObjects} = \{(o_1, w_1), \ldots, (o_n, w_n)\}$ and $x \in X$, define the matching score of $\Phi_{SimObjects}$ for x as follows:

$$ms_o(\Phi_{SimObjects}, x) = \frac{\Sigma_{i=1}^n (w_i \times Sim_{oo}(o_i, x))}{\Sigma_{i=1}^n w_i}$$

– If $\Phi_{SimObjects}$ is a fuzzy function then define

$$ms_o(\Phi_{SimObjects}, x) = \Phi_{SimObjects}(x)$$

– For $x \in X$, define the matching score of Φ for x as follows:

$$ms(\Phi, x) = \Phi_{weighting}(y_1, y_2, y_3, y_4, y_5)$$

where

$$\begin{aligned} y_1 &= ms_s^{syn}(\Phi_{SynPhrases}, x) \\ y_2 &= ms_s^{sem}(\Phi_{SemPhrases}, x) \\ y_3 &= ms_c(\Phi_{Concepts}, x) \\ y_4 &= ms_o(\Phi_{SimObjects}, x) \\ y_5 &= ObjectRank(x) \end{aligned}$$

– The result of the query Φ is the set of objects $x \in X$ that have the highest scores $ms(\Phi, x)$. The maximal number of objects to return is parameterized by *MaxResults*.

2.4 Summary

The proposed abstract model for SONCA depends on:

– computing a collection of terms (TERMS) and defining a function $Terms(d, h)$, where d is a text and h is a *tf-idf* threshold
– using a general-purpose ontology with the following functions:

  - $PossibleConcepts(t)$, where t is a term
  - $Sim_{tc}(x, y)$, where x is a term and y is a concept
  - $SimilarConcepts(c, n, s)$, where c is a concept, n is the maximal number of returned concepts which are similar to c, and s is a threshold for similarity
- using the abstract document-representation ADR3 (in particular, computing and using *ObjectRank* and *RelatedConcepts* of objects)
- defining and implementing the following similarity functions:
  - $Sim_{st}(x, y)$, where x is a string and y is a term
  - $Sim_{so}^{syn}(x, y)$, where x is a string and y is an object (a document)
  - $Sim_{so}^{sem}(x, y)$, where x is a string and y is an object (a document)
  - $Sim_{co}(x, y)$, where x is a concept and y is an object (a document)
  - $Sim_{oo}(x, y)$, where x and y are objects (documents)
- using appropriate values for the parameters *MaxTermLength*, *MinIdf*, *MinTfIdf*[_], *MaxConceptsForDoc*, *MinConSimForDoc*, *MaxResults*
- using an appropriate definition for the fuzzy Boolean operators $\wedge$ and $\vee$.

Function Sim_{so}^{sem} may be defined by using *Terms*, *PossibleConcepts*, Sim_{st}, Sim_{tc}, *SimilarConcepts* and the attributes *RelatedConcepts* of objects. Functions Sim_{co} and Sim_{oo} may be defined by using function *SimilarConcepts* and the attributes *RelatedConcepts* of objects.

3 An Instantiation of the Abstract Model

In this section, we describe how the abstract model presented in the previous section can be implemented. There are other possible realizations of that abstract model, and what proposed in this section also allows different ways of implementation.

3.1 Collection TERMS and Involved Functions

Collection TERMS can be constructed either by extracting terms from the database of SONCA or by importing terms from existing datasets. Consider the first case. Assume that most documents have been stored in the SONCA system using ADR1 representation. For each text d occurring in the system, every phrase of d consisting of no more than *MaxTermLength* words is checked and, if it is acceptable as a term, will be normalized and added to the collection TERMS. For example, a phrase like "and decision" is not accepted as a term, and a phrase "decision systems" can be normalized to "decision system". At the end, the measure idf_i of each element t_i of TERMS will be computed; if $idf_i > MinIdf$ then idf_i will be stored together with t_i in TERMS, else the term t_i is deleted from TERMS. After that we assume that *idf* values of elements of TERMS are rescaled to fit into the interval $(0, 1]$. For the case TERMS is constructed by importing terms from existing datasets, the parameters *MaxTermLength* and *MinIdf* are also used in an appropriate way, and rescaling to the interval $(0, 1]$ is also performed.

Function $Terms(d, h)$, where d is a text and h is a *tf-idf* threshold, may be defined as follows:

- initialize rs to an empty list
- for every phrase t of d consisting of no more than $MaxTermLength$ words
  - let t' be the normalized form of t
  - if t' occurs in TERMS then
    - let x be the tf measure of t in d
    - let y be the idf measure of t in TERMS
    - if $x.y \geq h$ then add the pair (t, x) to rs
- sort rs decreasingly w.r.t. the second coordinates of the pairs
- return rs.

Having collection TERMS and function $Terms(d, h)$ defined, documents represented in ADR1 can be enriched to ADR2 in the usual way.

Function $Sim_{st}(x, y)$ for a string x and a term y from TERMS may be defined as follows, where *SimST1* and *SimST2* are parameters of SONCA with values in $(0, 1]$:

- if the normalized form of x is equal to y then return 1
- let s_x be the set of normalized forms of words occurring in x
- let s_y be the set of normalized forms of words occurring in y
- if $s_x = s_y$ then return *SimST1* else $s := s_x \cap s_y$
- if $s = s_x$ then $a := 1$ else $a :=$ *SimST2*
- if $s = s_y$ then $b := 1$ else
  - let b_0 be the maximal idf measure of a word from s w.r.t. TERMS
  - let b_1 be the idf measure of y w.r.t. TERMS
  - $b := (b_0/b_1) \times (|s|/|s_y|)$ (one can modify this formula somehow, e.g., by applying *sqrt* to the whole formula or a part of it)
- if the words of s occur in x in the same order as in y (ignoring the differences caused by normalization) then $c := 1$ else $c :=$ *SimST1*
- return $a \times b \times c$.

3.2 The Main Ontology and Involved Functions

The SONCA system may use a number of ontologies, but they are combined into one, which is called the *main ontology* of SONCA. This ontology contains information about concepts and relationships between them.

A concept is specified by the concept ID, the ID of the component ontology (if necessary), and a list of pairs $(label, weight)$ called the *list of labels* of the concept, where $label$ is a term and $weight$ is a real number in $(0, 1]$ representing the accuracy that the term has the concept's meaning. In the case information about the $weight$ is not available, we assume that it is equal to 1. Information about concepts may be stored in a few tables (e.g., component ontologies, concepts, labels of concepts). IDs of concepts are unique. For convenience, sometimes we just write "concept" to mean "concept-ID".

We index essential words occurring in labels of concepts by using a table WORD-CONC consisting of pairs $(word, concept\text{-}ID)$ together with an index on the attribute "word". For each word w occurring in label l of a concept c: let w' be the normalized form of w; if w' occurs in TERMS then add the pair (w, c) to that table.

We use a table CONC-CONC to store relationships between concepts. It consists of tuples of the form $(ID_1, ID_2, type, similarity)$, where ID_1 and ID_2 are IDs of two concepts, *type* is the type of the relationship, and *similarity* is a real number in $(0, 1)$ representing the degree that the first concept is similar to the second concept. In the case information about the *similarity* is not available, it will be computed from the *type*, according to some strategy based on parameters. For example, the type "is a subconcept of" (e.g., Man is a subconcept of $Human$) may have $similarity = 0.8$, while the type "is a superconcept of" may have $similarity = 0.4$ (notice the asymmetry).

The main ontology of SONCA can be constructed from DBpedia, in particular, from the datasets "Articles Categories", "Categories (Skos)", "Links to YAGO2" and "Word Net Classes". See the appendix for more details.

Function $PossibleConcepts(t)$, which returns the set of concepts which may correspond to the term t, can be defined as follows:

- if the dataset "Disambiguation Links" of DBpedia can be used to determine the concepts corresponding to t then use it and return the result
- $rs := \emptyset$
- add to rs all concepts that have t as a label
- for each word w occurring in t:
  - let w' be the normalized form of w
  - for each pair (w', c) occurring in WORD-CONC, add c to rs
- return rs.

Function $Sim_{tc}(x, y)$, where x is a term and y is a concept, may be defined as follows:

- let the list of labels of y be $(l_1, w_1) \ldots (l_k, w_k)$
- let $t\text{-}conorm_{tc}$ be the $t\text{-}conorm$ used for Sim_{tc} (e.g., $\perp_{max}$ or $\perp_{sum}$)[4]
- return $t\text{-}conorm_{tc}\{Sim_{st}(x, l_i) \times w_i \mid 1 \leq i \leq k\}$.

Function $SimilarConcepts(c, n, s)$, where c is a concept, n is the maximal number of expected concepts similar to c, and s is a threshold for similarity, may be defined as shown on page 22.

3.3 Computing *ObjectRank* and *RelatedConcepts* of Objects

Recall that *ObjectRank* and *RelatedConcepts* of objects are information to be added to ADR2 to obtain ADR3. To compute *ObjectRank* values of publications, authors and institutions we create a graph consisting of those objects as vertices, with edges specified as follows:

[4] $\perp_{max}(a, b) = max(a, b)$ and $\perp_{sum}(a, b) = a + b - a.b$, for $a, b \in [0, 1]$.

Function. SimilarConcepts(c, n, s)

```
let t-conorm-SimCon be the t-conorm used for SimilarConcepts;
rs := {(c, 1)},  min := 1,  count := 1,  limit := s,  used := ∅;
while (rs \ used) ≠ ∅ do
    let (c1, w1) be a pair from rs \ used with the largest w1;
    add (c1, w1) to used;
    foreach tuple (c1, c2, _, w2) in table CONC-CONC do
        w2' := w1 × w2;
        if w2' ≥ limit and (count < n or w2' > min) then
            if there exists (c2, w2'') ∈ rs then
                replace (c2, w2'') in rs by (c2, t-conorm-SimCon(w2', w2''))
            else if count < n then
                add (c2, w2') to rs and increase count by 1
            else
                find a pair (c3, min) ∈ rs;
                replace (c3, min) in rs by (c2, w2')
            update the variable min to min{w | (_, w) ∈ rs};
            if count = n then limit := min
return rs
```

- for each publication x:
  - for each publication x' cited by x, add the edge (x, x') to the graph
  - for each author y of x, add the edges (x, y) and (y, x) to the graph
  - for each institution z declared in x as an affiliation of an author of x, add the edge (x, z) to the graph
- for each author y and each current institution z of y, add the edges (y, z) and (z, y) to the graph.

Having that graph constructed, *ObjectRank* values of the objects can be computed as *PageRank* values of Web pages, treating edges of the graph as links between Web pages. Some modifications can be investigated, e.g., assigning weights to the edges, starting with different *ObjectRank* values for different objects, or adding other kinds of edge/vertex to the graph.

Values *RelatedConcepts* of objects can be computed in the following order:

1. values *RelatedConcepts* of journals and conferences
2. preliminary values *RelatedConcepts* of publications, computed without using citations and information about authors
3. preliminary values *RelatedConcepts* of authors
4. refined values *RelatedConcepts* of publications and authors.

Consider computing *RelatedConcepts* for a journal/conference x which is described by a list of topics $(t_1, \ldots, t_k)$, where each t_i is a term (already normalized). When such a description is not available, use the name of x as a "topic". It can be assumed that each t_i corresponds to a concept, or at most two concepts. Here, we can ignore the limits *MinConSimForDoc* and *MaxConceptsForDoc*. *RelatedConcepts* of x may be computed as follows, where

ParamRC1, *ParamRC2*, *ParamRC3* are parameters (e.g., with values 100, 0.4, 0.8, respectively):

- $rs := \emptyset$
- for each $1 \leq i \leq k$:
  - choose $y_i \in \mathit{PossibleConcepts}(t_i)$ with a maximal value $\mathit{Sim}_{tc}(t_i, y_i)$
  - add the pair $(y_i, \mathit{Sim}_{tc}(t_i, y_i))$ to rs
- for each $1 \leq i \leq k$:
  - let $C := \{c \mid (c, \_) \in \mathit{SimilarConcepts}(y_i, \mathit{ParamRC1}, \mathit{ParamRC2})\}$
  - choose $z_i \in \mathit{PossibleConcepts}(t_i) \setminus C$ with a maximal value $\mathit{Sim}_{tc}(t_i, z_i)$
  - if $\mathit{Sim}_{tc}(t_i, z_i) \geq \mathit{Sim}_{tc}(t_i, y_i) \times \mathit{ParamRC3}$) or ($\mathit{Sim}_{tc}(t_i, z_i) \geq \mathit{Sim}_{tc}(t_i, y_i) \times \mathit{ParamRC2}$ and z_i is "supported" by rs) then add the pair $(z_i, \mathit{Sim}_{tc}(t_i, z_i))$ to rs
- return rs.

Recall that a publication in ADR1 form has a tree-like structure, where each leaf can be specified as a pair (a, d) with a being the sequence of component attributes forming the path from the root to the leaf and d being the value of the leaf (i.e., the value of the last attribute in the sequence). One can treat a as a complex attribute. Let $\mathit{WeightForRC}[a]$ be a parameter standing for the weight of a for computing values *RelatedConcepts* of publications. Different complex concepts may have different weights. For example, an explicit keyword or a term in the title should have a higher weight than a term occurring in the text of a section.

Let x be a publication in ADR2 form. A preliminary value *RelatedConcepts* for x can be computed by function `RelatedConceptsOfPub`(x) given on page 24, where *UsingSupportsForRC* is a Boolean parameter, *ParamRC4* is a parameter of type natural number, *ParamRC5* and *ParamRC6* are parameters with values in $(0, 1]$. Exemplary values for these parameters are $\mathit{ParamRC4} = 100$, $\mathit{ParamRC5} = 0.1$ and $\mathit{ParamRC6} = 0.6$. A term t in a text d of a complex attribute a may corresponds to a concept c. If w_1 is the *tf-idf* measure of t in d w.r.t. TERMS, $w_2 = \mathit{Sim}_{tc}(t, c)$ and w_3 is the weight of a, then the weight of c for x is increased by $w_1 \times w_2 \times w_3$. The function `RelatedConceptsOfPub` accumulates weights for each possible related concept. In the case the flag *UsingSupportsForRC* is set on, "support" for each related concept c is computed, based on its similarity to the other related concepts, and is used to modify the weight of c. After that, concepts with too small weights are eliminated, and weights of the remaining concepts are rescaled to fit into the interval $(0, 1]$.

In the case values of $\mathit{WeightForRC}[\_]$ are normalized to the interval $(0, 1]$, one can use a *t-conorm* denoted by $\mathit{t\text{-}conorm\text{-}RC}_P$ to modify function `RelatedConceptsOfPub` as follows (which may result in a worse function):

- replace $w' + \mathit{WeightForRC}[a_i] \times w$ in line 2 by

$$\mathit{t\text{-}conorm\text{-}RC}_P(w', \mathit{WeightForRC}[a_i] \times w)$$

- replace $w' + w''$ in line 2 by $\mathit{t\text{-}conorm\text{-}RC}_P(w', w'')$
- delete line 2 (i.e., rescaling is not needed anymore).

Function. RelatedConceptsOfPub(x)

Input: a publication x in ADR2 form (specified as a document ID).
Output: a set of weighted concepts which are the most related to the topic of x.

let all the text leaves of the tree-representation of x be $(a_1, d_1), \ldots, (a_n, d_n)$, not counting the terms used for extending ADR1 to ADR2;
$rs := \emptyset$;
foreach $1 \leq i \leq n$ **do**
    $rs_i := \emptyset$;
    let $(t_1, w_1), \ldots, (t_k, w_k)$ be the list of pairs (term, frequency) associated with d_i in the ADR2 of x;
    foreach $1 \leq j \leq k$ **do**
        foreach $c \in PossibleConcepts(t_j)$ **do**
            add the pair $(c, w_j \times idf(t_j) \times Sim_{tc}(t_j, c))$ to rs_i
    delete from rs_i pairs (c, w) with "too low" w;
    foreach $(c, w) \in rs_i$ **do**
        if *there exists* $(c, w') \in rs$ **then**
            replace (c, w') in rs by $(c, w' + WeightForRC[a_i] \times w)$
        else add the pair $(c, WeightForRC[a_i] \times w)$ to rs
delete from rs pairs (c, w) with "too low" w;
if *UsingSupportsForRC* **then**
    $supports := rs$;
    foreach $(c, w) \in rs$ **do**
        foreach $(c', w') \in SimilarConcepts(c, ParamRC4, ParamRC5)$ **do**
            if *there exists* $(c', w'') \in supports$ **then**
                replace (c', w'') in *supports* by $(c', w' + w'')$
    foreach $(c, w') \in supports$ **do**
        let w be the value such that $(c, w) \in rs$;
        replace (c, w) in rs by $(c, w \times ParamRC6 + w' \times (1 - ParamRC6))$
keep in rs only $MaxConceptsForDoc$ pairs (c, w) with the highest w;
rescale values of the second components of pairs from rs to the interval $(0, 1]$ in an appropriate way (to reflect similarities of the corresponding concepts to the topic of the publication x);
delete from rs pairs (c, w) with $w < MinConSimForDoc$;
return rs

Having preliminary values *RelatedConcepts* of publications computed, preliminary values *RelatedConcepts* of authors representing their research areas can be computed next. We can assume that the topics of publications of an author x are more concrete than the main research areas of x. A preliminary value *RelatedConcepts* for an author x can be computed by function `ResearchAreasOfAuthor`(x) given on page 25, where *ParamRC7* is a parameter of type natural number and *ParamRC8* is a parameter with value in $(0, 1]$. Exemplary values for these parameters are $ParamRC7 = 100$ and $ParamRC8 = 0.4$.

Function. ResearchAreasOfAuthor(x)

```
1  let y_1, ..., y_n be the publications of x, already in ADR3 form;
2  rs := ∅;
3  foreach 1 ≤ i ≤ n do
4      foreach (c, w) ∈ RelatedConcepts(y_i) do
5          foreach (c', w') ∈ SimilarConcepts(c, ParamRC7, ParamRC8) do
6              if there exists (c', w'') ∈ rs then
7                  replace (c', w'') in rs by (c', w'' + w × w')
8              else add (c', w × w') to rs
9  keep in rs only MaxConceptsForDoc pairs (c, w) with the highest w;
10 rescale values of the second components of pairs from rs to the interval (0, 1] in
   an appropriate way (to reflect similarities of the corresponding concepts to the
   research areas of the author x);
11 delete from rs pairs (c, w) with w < MinConSimForDoc;
12 return rs
```

Function. $\mathrm{Sim}_{so}^{syn}(x, y)$

Input: a string x and an object y.
Output: a value in $[0, 1]$ representing similarity between x and y.

```
1  let the set of normalized words of x be {u_1, ..., u_m};
2  sum := 0,  sum_2 := 0;
3  foreach 1 ≤ i ≤ m do
4      sum := sum + idf(u_i);
5      if u_i ∈ words(y) then sum_2 := sum_2 + idf(u_i)
6  let the set of terms of x be {t_1, ..., t_n};
7  sum' := 0,  sum'_2 := 0;
8  foreach 1 ≤ i ≤ n do
9      sum' := sum' + idf(t_i);
10     if t_i ∈ terms(y) then sum'_2 := sum'_2 + idf(t_i)
11 return (sum_2/sum) × ParamSim1 + (sum'_2/sum') × (1 − ParamSim1)
```

One can also use a *t-conorm* denoted by *t-conorm-RC*$_A$ to modify function `ResearchAreasOfAuthor` as follows (which may result in a worse function):

- replace $w'' + w \times w'$ in line 3 by *t-conorm-RC*$_A(w'', w \times w')$
- delete line 3 (i.e., rescaling is not needed anymore).

In the next stage, values *RelatedConcepts* of publications can be refined by taking into account also preliminary values *RelatedConcepts* of the cited papers and the authors. After that, values *RelatedConcepts* of authors can also be refined by taking into account also refined values *RelatedConcepts* of their publications. We do not go into details, but want to emphasize that weights of topics of cited papers should be small enough unless it is known which cited papers are most related to the considered publication.

3.4 Computing Other Similarity Functions

Function $Sim^{syn}_{so}(x, y)$, where x is a string and y is an object, may be defined as shown on page 25, where parameter *ParamSim1* has a value in $(0, 1]$, e.g., 0.7.

Function. $\mathrm{Sim}^{sem}_{so}(x, y)$

Input: a string x and an object y.
Output: a value in $[0, 1]$ representing similarity between x and y.

let *t-conorm-*Sim^{sem}_{so} be the *t-conorm* used for Sim^{sem}_{so};
$rs := 0$;
let $Terms(x, ParamSim2) = ((t_1, w_1), \ldots, (t_n, w_n))$;
foreach $1 \leq i \leq n$ **do**
  foreach $c \in PossibleConcepts(t_i)$ **do**
    foreach $(c', w) \in SimilarConcepts(c, ParamSim3, ParamSim4)$ **do**
      if *there exists* $(c', w') \in RelatedConcepts(y)$ **then**
        $rs :=$ *t-conorm-*$Sim^{sem}_{so}(rs, Sim_{st}(x, t_i) \times Sim_{tc}(t_i, c) \times w \times w')$
return rs

Function. $\mathrm{Sim}_{co}(x, y)$

Input: a concept x and an object y.
Output: a value in $[0, 1]$ representing similarity between x and y.

let *t-conorm-*Sim_{co} be the *t-conorm* used for Sim_{co};
$rs := 0$;
foreach $(c, w) \in SimilarConcepts(x, ParamSim3, ParamSim4)$ **do**
  if *there exists* $(c, w') \in RelatedConcepts(y)$ **then**
    $rs :=$ *t-conorm-*$Sim_{co}(rs, w \times w')$
return rs

Function. $\mathrm{Sim}_{oo}(x, y)$

Input: objects x and y.
Output: a value in $[0, 1]$ representing similarity of y to x.

let *t-conorm-*Sim_{oo} be the *t-conorm* used for Sim_{oo};
$concepts := \emptyset$;
foreach $(c, w) \in RelatedConcepts(x)$ **do**
  foreach $(c', w') \in SimilarConcepts(c, ParamSim3, ParamSim4)$ **do**
    if *there exists* $(c', w'') \in concepts$ **then**
      replace (c', w'') in *concepts* by $(c',$ *t-conorm-*$Sim_{oo}(w'', w \times w'))$
    else add $(c', w \times w')$ to *concepts*
$sum := 0$, $sum_2 := 0$;
foreach $(c, w) \in concepts$ **do**
  $sum := sum + w$;
  if *there exists* $(c, w') \in RelatedConcepts(y)$ **then**
    $sum_2 := sum_2 + \min(w, w')$
return sum_2/sum

Function $Sim_{so}^{sem}(x, y)$, where x is a string and y is an object, may be defined as shown on page 26, where parameter *ParamSim2* is a *tf-idf* threshold, parameter *ParamSim3* is of type natural number, and parameter *ParamSim4* has a value in $(0, 1]$. Exemplary values for *ParamSim3* and *ParamSim4* are 100 and 0.4, respectively.

Function $Sim_{co}(x, y)$, where x is a concept and y is an object, may be defined as shown on page 26.

Function $Sim_{oo}(x, y)$, which measures similarity of an object y to an object x, may be defined as shown on page 26. This function first extends the list of weighted concepts related with x, and then checks similarity of y to that list of concepts.

4 On the User Interface for SONCA

As mentioned before, the SONCA system enables combination of metadata-based search, syntactic keyword-based search and semantic search. A simple search may depend only on basic metadata. Most existing websites for searching publications use such kind of search, including:

- SCOPUS Search [21]
- Advanced Search - Scirus
- Google Advanced Scholar Search
- CiteSeerX Advanced Search
- Advanced Product Search - Elsevier
- arXiv.org Search
- Biomedical Search [22].

Among the above websites the interface of "SCOPUS Search" seems most flexible. It allows also "author search" and "affiliation search". Note that syntactic keyword-based search is used for the attribute "abstract" in some of the mentioned websites. The website "Google Advanced Scholar Search" puts more emphasis on syntactic keyword-based search, but still uses some metadata.

A simple user interface for SONCA may use fields for typing in:

- conditions on basic metadata
- a list or a Boolean expression of syntactic keywords
- a list or a Boolean expression of semantic keywords.

Basic metadata may include, e.g., the attributes used for "SCOPUS Search". We may allow also attributes like *ObjectRank*, the number of citations of a publication, or the H-index of an author. Using the simple interface, the system decides itself what weights to associate with the keywords, how to combine the conditions on fuzzy matching, and how to weight the results.

An advanced user interface for SONCA should allow sophisticated ways for:

- specifying conditions on metadata
- providing a list of weighted syntactic keywords, possibly for some concrete long-text attributes or for the whole queried document

- specifying conditions on fuzzy matching using semantic keywords, concepts and objects
- specifying a method for combining the conditions and weighting the results.

Consider the first item of the above list. Here are some proposals:

- We can define a user logical data model, based on the relational database model of SONCA, by using appropriate views and hiding or blocking access to certain information. We can then allow the user to write SQL-like queries using that data model.
- Alternatively, we can allow the user to construct description-logic-like queries by using concept names, role names and constructors from a certain list.
- We can allow the user to query an external system (e.g., DBpedia by using SPARQL) and use the returned results in specifying a query to the SONCA system. We can also allow using search results returned by SONCA to specify another query to SONCA.
- We can allow the user to specify and use additional predicates by some recursive rule language, e.g., semipositive Datalog$^\neg$ with the standard semantics.

The idea of querying an external system like DBpedia can also be adopted for specifying keywords or concepts for a query to SONCA.

The part involving semantic search in a query may consist of three components $\Phi_{SemPhrases}$, $\Phi_{Concepts}$ and $\Phi_{SimObjects}$. By preprocessing the query and interacting with the user, the interface module may convert that part to a list or an expression of only weighted concepts.

The interface module may help the user in constructing or refining a query by using the main ontology. The approach of the DOPE Browser [23] for navigating and refining queries is interesting and can be adopted for the SONCA system.

5 On Efficient Computation

There are two kinds of computation: pre-computation, e.g., for computing *RelatedConcepts* and *ObjectRank* of objects; and online computation, e.g., for answering queries. Accuracy seems more important for pre-computation, while efficiency seems more important for online computation. In this section, we present some proposals for increasing efficiency of the query answering process.

Consider the case when the part involving semantic search of a query consists of a set of weighted phrases, a set of weighted concepts, and a set of weighted objects (used for checking similarity). That part can be converted to a set of weighted concepts, e.g., by function *ConvertSemSubquery* given on page 29. The conversion result can be presented to the user and improved by him/her.

Having a set of weighted concepts for matching with documents, we first extend it by function *ExtendConcepts* given on page 29 and then match concepts of the resulting set directly with the documents by function *DirectSemMatching* given on page 30.

Our first proposal is to preprocess the part involving semantic search of a query by using functions *ConvertSemSubquery* and *ExtendConcepts* (or similar

Function. ConvertSemSubquery($\Phi_{SemPhrases}, \Phi_{Concepts}, \Phi_{SimObjects}$)

Input: $\Phi_{SemPhrases} = \{(s_1, u_1), \ldots, (s_h, u_h)\}$, $\Phi_{Concepts} = \{(c_1, v_1), \ldots, (c_k, v_k)\}$, $\Phi_{SimObjects} = \{(o_1, w_1), \ldots, (o_l, w_l)\}$, where each s_i is a string, each c_i is a concept, each o_i is an object, and each u_i (resp. v_i, w_i) is the weight of s_i (resp. c_i, o_i).

Output: a set of weighted concepts.

let *t-conorm-CSS* be the *t-conorm* used for this function;
concepts := $\{(c_1, v_1), \ldots, (c_k, v_k)\}$;
foreach $1 \leq i \leq h$ **do**
    let $Terms(s_i, ParamSim2) = ((t_1, x_1), \ldots, (t_n, x_n))$ and $sum = \Sigma_{i=1}^{n} x_i$;
    foreach $1 \leq j \leq n$ **do**
        foreach $c \in PossibleConcepts(t_j)$ **do**
            if *there exists* $(c, v) \in concepts$ **then**
                replace (c, v) in *concepts* by $(c, t\text{-}conorm\text{-}CSS(v, u_i \times (x_j/sum) \times Sim_{st}(s_i, t_j) \times Sim_{tc}(t_j, c)))$
            else add $(c, u_i \times (x_j/sum) \times Sim_{st}(s_i, t_j) \times Sim_{tc}(t_j, c))$ to *concepts*
foreach $1 \leq i \leq l$ **do**
    let $sum = \Sigma\,\{v \mid$ there exists $(c, v) \in RelatedConcepts(o_i)\}$;
    foreach $(c, v) \in RelatedConcepts(o_i)$ **do**
        if *there exists* $(c, v') \in concepts$ **then**
            replace (c, v') in *concepts* by $(c, t\text{-}conorm\text{-}CSS(v', w_i \times v/sum))$
        else add $(c, w_i \times v/sum)$ to *concepts*
return *concepts*

Function. ExtendConcepts(*concepts*)

Input: a set *concepts* = $\{(c_1, w_1), \ldots, (c_k, w_k)\}$, where each c_i is a concept and w_i is the weight of c_i.

Output: another set of weighted concepts.

let *t-conorm-EC* be the *t-conorm* used for this function;
rs := $\emptyset$;
foreach $1 \leq i \leq k$ **do**
    foreach $(c, w) \in SimilarConcepts(c_i, ParamSim3, ParamSim4)$ **do**
        if *there exists* $(c, w') \in rs$ **then**
            replace (c, w') in *rs* by $(c, t\text{-}conorm\text{-}EC(w', w_i \times w))$
        else add $(c, w_i \times w)$ to *rs*
return *rs*

ones) and then apply function *DirectSemMatching* (or a similar one) to score documents from a given set X w.r.t. that part of the query. This replaces the functions ms_s^{sem}, ms_c, ms_o and requires a change for the function ms proposed in Section 2.3. The point is that preprocessing the query does not depend on documents and can be done efficiently, while direct matching can also be done efficiently using set-oriented operations.

Function. DirectSemMatching($concepts, X$)

Input: a set $concepts = \{(c_1, w_1), \ldots, (c_k, w_k)\}$, where each c_i is a concept and w_i is the weight of c_i; and a set X of objects
Output: a set of pairs (x, ms), where $x \in X$ and ms is a matching score for x w.r.t. *concepts*

```
1 rs := ∅;
2 sum := w_1 + ... + w_k;
3 foreach x ∈ X do
4     sum_2 := 0;
5     foreach 1 ≤ i ≤ k do
6         if there exists (c_i, w'_i) ∈ RelatedConcepts(x) then
7             sum_2 := sum_2 + min(w_i, w'_i)
8     add (x, sum_2/sum) to rs
9 return rs
```

Given a query $\Phi = (\Phi_{Metadata}, \Phi_{SynPhrases}, \Phi_{SemPhrases}, \Phi_{Concepts}, \Phi_{SimObjects}, \Phi_{weighting})$, we can apply $\Phi_{Metadata}$ first to restrict the set of possible results to a set X of objects. If the component $\Phi_{Metadata}$ is empty or the returned set X is still too big, it is desirable to restrict the set further before scoring the objects using the other components of Φ. For this aim one can adopt the restriction that each result of the search must be an object containing a term specified by $\Phi_{SynPhrases}$. If the set is still too big, a further restriction may be the assumption that each result of the search must be an object "related" to a concept specified by $\Phi_{SemPhrases}$, $\Phi_{Concepts}$ or $\Phi_{SimObjects}$ (e.g., a concept specified by $ExtendConcepts(ConvertSemSubquery(\Phi_{SemPhrases}, \Phi_{Concepts}, \Phi_{SimObjects}))$).

6 Conclusions

We have designed the SONCA system using the query-oriented approach. Our design allows combination of metadata-based search, syntactic keyword-based search and semantic search, and the use of *ObjectRank*. In our approach, semantic search is based on measuring similarity between terms, concepts and documents. Our proposals have carefully been chosen to guarantee practicability of the system. However, implementation and evaluation were not undertaken, and it is highly probable that the design, in particular, the detailed definitions of functions, can be significantly improved.

Acknowledgements. This work was supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the strategic scientific research and experimental development program: "Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. IJCAI 2005, pp. 364–369. Morgan-Kaufmann Publishers (2005)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope further. In: Proc. of the OWLED 2008 DC Workshop on OWL: Experiences and Directions (2008)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The L-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)
4. Drabent, W., Małuszyński, J.: Well-founded semantics for hybrid rules. In: Marchiori, M., Pan, J.Z., de Sainte Marie, C. (eds.) RR 2007. LNCS, vol. 4524, pp. 1–15. Springer, Heidelberg (2007)
5. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the Semantic Web. ACM Trans. Comput. Log. 12(2), 11 (2011)
6. Glimm, B., Horrocks, I., Motik, B.: Optimized description logic reasoning via core blocking. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 457–471. Springer, Heidelberg (2010)
7. Goré, R., Nguyen, L.A.: EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In: Olivetti, N. (ed.) TABLEAUX 2007. LNCS (LNAI), vol. 4548, pp. 133–148. Springer, Heidelberg (2007)
8. Goré, R., Widmann, F.: Sound global state caching for ALC with inverse roles. In: Giese, M., Waaler, A. (eds.) TABLEAUX 2009. LNCS (LNAI), vol. 5607, pp. 205–219. Springer, Heidelberg (2009)
9. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. WWW 2003, pp. 48–57 (2003)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. KR 2006, pp. 57–67. AAAI Press (2006)
11. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. Autom. Reasoning 39(3), 249–276 (2007)
12. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8(3) (2000)
13. http://www.synat.pl
14. http://www.w3.org/TR/owl-guide/
15. http://www.w3.org/TR/owl2-overview/
16. http://wiki.dbpedia.org/About
17. http://wordnet.princeton.edu/
18. http://plwordnet.pwr.wroc.pl/wordnet/
19. http://en.wikipedia.org/wiki/Tf-idf
20. http://pl.wikipedia.org/wiki/PageRank
21. http://www.scopus.com/search/form.url
22. http://www.biomedsearch.com/search.html
23. http://www.w3.org/2001/sw/sweo/public/UseCases/Elsevier/
24. http://www.w3.org/TR/skos-reference/
25. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive Datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
26. Knorr, M., Alferes, J.J., Hitzler, P.: A coherent well-founded model for hybrid MKNF knowledge bases. In: Proc. ECAI 2008, Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 99–103. IOS Press (2008)

27. Levy, A.Y., Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artif. Intell. 104(1-2), 165–209 (1998)
28. Mishra, R.B., Kumar, S.: Semantic web reasoners and languages. Artif. Intell. Rev. 35(4), 339–368 (2011)
29. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
30. Motik, B., Sattler, U.: A comparison of reasoning techniques for querying large description logic ABoxes. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 227–241. Springer, Heidelberg (2006)
31. Nguyen, L.A.: A bottom-up method for the deterministic horn fragment of the description logic $\mathcal{ALC}$. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 346–358. Springer, Heidelberg (2006)
32. Nguyen, L.A.: An efficient tableau prover using global caching for the description logic ALC. Fundam. Inform. 93(1-3), 273–288 (2009)
33. Nguyen, L.A.: Horn knowledge bases in regular description logics with PTime data complexity. Fundam. Inform. 104(4), 349–384 (2010)
34. Nguyen, L.A.: Paraconsistent and approximate semantics for the OWL 2 web ontology language. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 710–720. Springer, Heidelberg (2010)
35. Nguyen, L.A.: A cut-free exptime tableau decision procedure for the logic extending converse-PDL with regular inclusion axioms. CoRR, abs/1104.0405 (2011)
36. Nguyen, L.A.: Cut-free expTime tableaux for checking satisfiability of a knowledge base in the description logic $\mathcal{ALCI}$. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 465–475. Springer, Heidelberg (2011)
37. Nguyen, L.A.: A cut-free expTime tableau decision procedure for the description logic SHI. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part I. LNCS(LNAI), vol. 6922, pp. 572–581. Springer, Heidelberg (2011), http://arxiv.org/abs/1106.2305
38. Nguyen, L.A., Szałas, A.: expTime tableaux for checking satisfiability of a knowledge base in the description logic $\mathcal{ALC}$. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS (LNAI), vol. 5796, pp. 437–448. Springer, Heidelberg (2009)
39. Nguyen, L.A., Szałas, A.: Checking consistency of an ABox w.r.t. global assumptions in PDL. Fundam. Inform. 102(1), 97–113 (2010)
40. Nguyen, L.A., Szałas, A.: Tableaux with global caching for checking satisfiability of a knowledge base in the description logic SH. T. Computational Collective Intelligence 1, 21–38 (2010)
41. Nguyen, L.A., Szałas, A.: Three-valued paraconsistent reasoning for semantic web agents. In: Jędrzejowicz, P., Nguyen, N.T., Howlet, R.J., Jain, L.C. (eds.) KES-AMSTA 2010. LNCS, vol. 6070, pp. 152–162. Springer, Heidelberg (2010)
42. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Proc. KR 2010. AAAI Press (2010)

A Appendix: On DBpedia

DBpedia is a very useful source of structured information. It allows us to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. DBpedia can be used as a main ontological source for SONCA, as it contains a large multi-domain ontology, links to Wikipedia pages and connections to other information sources like WordNet.

The DBpedia data set describes more than 3.5 million "things" with over half a billion "facts". The current version DBpedia 3.6 is available for download at `http://wiki.dbpedia.org/Downloads36`. The most important datasets for SONCA are:

Articles Categories: links from resources to categories using the SKOS vocabulary [24].
Categories (Skos): information about which concept is a category and how categories are related using the SKOS Vocabulary.
Disambiguation Links
Links to YAGO2: YAGO type information for DBpedia resources and the YAGO class hierarchy (the YAGO Classification is derived from the Wikipedia category system using Word Net).
Word Net Classes: classification links to RDF representations of WordNet classes.

The dataset Ontology Infobox Properties is one of the four high-quality "mapping-based" datasets in the /ontology/ namespace of DBpedia, but we can use it as an "external source". Other useful datasets for SONCA are, for example, Homepages, Persondata, External Links, Redirects, Links to DBLP, Links to Cyc.

Information from datasets like Categories (Labels), Titles, Links to Wikipedia Article can (probably) be computed from identifications of "things".

The dataset DBpedia Ontology is too small and seems useless for SONCA. For example, it does not contain words/concepts "Computer Science" (in any combination of cases). Consequently, the dataset Ontology Infobox Types, which contains triples of the form $object rdf:type $class (concept) is not useful for SONCA either.

Here is some exemplary information from the preview of datasets:

- Articles Categories:
  - `<http://dbpedia.org/resource/Aristotle>`
    `<http://purl.org/dc/terms/subject>`
    `<http://dbpedia.org/resource/Category:Metaphysicians>` .
  - `<http://dbpedia.org/resource/Aristotle>`
    `<http://purl.org/dc/terms/subject>`
    `<http://dbpedia.org/resource/Category:Humor_researchers>` .
  - `<http://dbpedia.org/resource/Aristotle>`
    `<http://purl.org/dc/terms/subject>`
    `<http://dbpedia.org/resource/Category:History_of_science>` .

  - ... (on "Aristotle")
  - We have a problem here. Namely, in DBpedia, "concept" means a concept defined in the dataset DBpedia Ontology. As mentioned before, this dataset is too small and rather useless for SONCA. Hence, we have to take "categories" as concepts in a broader sense. But, "Aristotle" is not an instance of "History_of_science". An alternation is to use the YAGO Classification.
- Categories (Skos):
  - `<http://dbpedia.org/resource/Category:Futurama> <http://www.w3.org/2004/02/skos/core#broader> <http://dbpedia.org/resource/Category:Comic_science_fiction> .`
  - `<http://dbpedia.org/resource/Category:Futurama> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2004/02/skos/core#Concept> .`
  - `<http://dbpedia.org/resource/Category:World_War_II> <http://www.w3.org/2004/02/skos/core#broader> <http://dbpedia.org/resource/Category:Wars_involving_Poland> .`
- Disambiguation Links:
  - `<http://dbpedia.org/resource/Alien> <http://dbpedia.org/ontology/wikiPageDisambiguates> <http://dbpedia.org/resource/Alien_%28law%29> .`
  - `<http://dbpedia.org/resource/Alien> <http://dbpedia.org/ontology/wikiPageDisambiguates> <http://dbpedia.org/resource/Alien_%28franchise%29> .`
  - `<http://dbpedia.org/resource/Alien> <http://dbpedia.org/ontology/wikiPageDisambiguates> <http://dbpedia.org/resource/Alien_%28film%29> .`
  - Information of this kind is useful for tagging documents with concepts.
- Links to YAGO2:
  - `<http://dbpedia.org/resource/Aristotle> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/class/yago/AncientGreekPhilosophers> .`
  - `<http://dbpedia.org/resource/Aristotle> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/class/yago/4th-centuryBCPhilosophers> .`
  - `<http://dbpedia.org/resource/Aristotle> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/class/yago/HumorResearchers> .`
  - ... (on "Aristotle")
- Word Net Classes:
  - `<http://dbpedia.org/resource/%24_%28film%29> <http://dbpedia.org/property/wordnet_type> <http://www.w3.org/2006/03/wn/wn20/instances/synset-movie-noun-1> .`

- <http://dbpedia.org/resource/%26_%28song%29> <http://dbpedia.org/property/wordnet_type> <http://www.w3.org/2006/03/wn/wn20/instances/synset-phonograph_record-noun-1> .
- <http://dbpedia.org/resource/%2703_Bonnie_%26_Clyde> <http://dbpedia.org/property/wordnet_type> <http://www.w3.org/2006/03/wn/wn20/instances/synset-phonograph_record-noun-1> .
- The reason for using this dataset is that a keyword may belongs to different concepts and WordNet contains information about "frequency counts" that is useful for tagging documents with concepts.

Note that YAGO2 Class and DBpedia Category are different notions. The dataset Links to YAGO2 provides classes together with a good relation "is-instance-of" from objects (resources) to classes. As mentioned before, there is no relation like "is-instance-of" from resources to categories. A similar relation from resources to categories is called "subject". For example, "Aristotle" has a subject belonging to the category "History_of_science".

There are the following semantic relations between categories (skos:semanticRelation), given in a hierarchy:

- skos:related
  - skos:relatedMatch
- skos:broaderTransitive
  - skos:broader
    - skos:broadMatch
- skos:narrowerTransitive
  - skos:narrower
    - skos:narrowMatch
- skos:mappingRelation
  - skos:closeMatch
    - skos:exactMatch
  - skos:relatedMatch
  - skos:broadMatch
  - skos:narrowMatch

We can restrict to categories of DBpedia, without using Yago classes. With this assumption, the most important notions and information from DBpedia for SONCA are:

- resource (object) and the relationship resource-category (subject) from the dataset Articles Categories
- the relationships category-category (skos:semanticRelation) from the dataset Categories (Skos)
- information from the datasets Disambiguation Links and Word Net Classes (used for choosing resources/categories and tagging documents with them)
- properties from the dataset Ontology Infobox Properties, which will be used as an additional feature for query languages and user interfaces for SONCA and are not required being stored in the database of SONCA.

Of course, other information from DBpedia (e.g., abstracts) are also useful for SONCA to a certain extent.