Detecting Similarities in Ontologies with the SOQA-SimPack Toolkit

Patrick Ziegler, Christoph Kiefer, Christoph Sturm,
Klaus R. Dittrich, and Abraham Bernstein

DBTG and DDIS, Department of Informatics, University of Zurich,
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
(pziegler,kiefer,sturm,dittrich,bernstein)@ifi.unizh.ch

Abstract. Ontologies are increasingly used to represent the intended real-world semantics of data and services in information systems. Unfortunately, different databases often do not relate to the same ontologies when describing their semantics. Consequently, it is desirable to have information about the similarity between ontology concepts for ontology alignment and integration. This paper presents the SOQA-SimPack Toolkit (SST), an ontology language independent Java API that enables generic similarity detection and visualization in ontologies. We demonstrate SST's usefulness with the SOQA-SimPack Toolkit Browser, which allows users to graphically perform similarity calculations in ontologies.

1 Introduction

In current information systems, ontologies are increasingly used to explicitly represent the intended real-world semantics of data and services. Ontologies provide a means to overcome heterogeneity by providing explicit, formal descriptions of concepts and their relationships that exist in a certain universe of discourse, together with a shared vocabulary to refer to these concepts. Based on agreed ontological domain semantics, the danger of semantic heterogeneity can be reduced. Ontologies can, for instance, be applied in the area of data integration for data content explication to ensure semantic interoperability between data sources.

Unfortunately, different databases often do not relate to the same ontologies when describing their semantics. That is, schema elements can be linked to concepts of different ontologies in order to explicitly express their intended meaning. This complicates the task of finding semantically equivalent schema elements since at first, semantic relationships between the concepts have to be detected to which the schema elements are linked to. Consequently, it is desirable to have information about the similarity between ontological concepts. In addition to schema integration, such similarity information can be useful for many applications, such as ontology alignment and integration, Semantic Web (service) discovery, data clustering and mining, semantic interoperability in virtual organizations, and semantics-aware universal data management.

The task of detecting similarities in ontologies is aggravated by the fact that a large number of ontology languages is available to specify ontologies. Besides traditional ontology languages, such as Ontolingua [5] or PowerLoom[1], there is a notable number of ontology languages for the Semantic Web, such as SHOE[2], DAML[3], or OWL[4]. That is, data semantics can often be described with respect to ontologies that are represented in various ontology languages. In consequence, mechanisms for effective similarity detection in ontologies must be capable of coping with heterogeneity caused by the use of different ontology languages. Additionally, it is desirable that different similarity measures can be employed so that different approaches to identify similarities among concepts in ontologies can be reflected.

For instance, assume that in an example scenario, a developer of an integrated university information system is looking for semantically similar elements from database schemas that relate to the following ontologies to describe their semantics: (1) the Lehigh University Benchmark Ontology[5] that is represented in OWL, (2) the PowerLoom Course Ontology[6] developed in the SIRUP project [21], (3) the DAML University Ontology[7] from the University of Maryland, (4) the Semantic Web for Research Communities (SWRC) Ontology[8] modeled in OWL, and (5) the Suggested Upper Merged Ontology (SUMO)[9], which is also an OWL ontology. Assume further that there are schema elements linked to all of the 943 concepts which these five ontologies are comprised of. Unless suitable tools are available, identifying semantically related schema elements in this set of concepts and visualizing the similarities appropriately definitely turns out to be time-consuming and labor-intensive.

In this paper, we present the SOQA-SimPack Toolkit (SST), an ontology language independent Java API that enables generic similarity detection and visualization in ontologies. Our main goal is to define a Java API suitable for calculating and visualizing similarities in ontologies for a broad range of ontology languages. Considering the fact that different databases often do not relate to the same ontologies, we aim at calculating similarities not only within a given ontology, but also between concepts of *different* ontologies. For these calculations, we intend to provide a generic and extensible library of ontological similarity measures capable of capturing a variety of notions of "similarity". Note that we do not focus on immediate ontology integration. Instead, we strive for similarity detection among different pre-existing ontologies, which are separately used to explicitly state real-world semantics as intended in a particular setting.

[1] `http://www.isi.edu/isd/LOOM/PowerLoom/`
[2] `http://www.cs.umd.edu/projects/plus/SHOE/`
[3] `http://www.daml.org`
[4] `http://www.w3.org/2004/OWL/`
[5] `http://www.lehigh.edu/~zhp2/univ-bench.owl`
[6] `http://www.ifi.unizh.ch/dbtg/Projects/SIRUP/ontologies/course.ploom`
[7] `http://www.cs.umd.edu/projects/plus/DAML/onts/univ1.0.daml`
[8] `http://www.ontoware.org/projects/swrc/`
[9] `http://reliant.teknowledge.com/DAML/SUMO.owl`

This paper is structured as follows: Section 2 gives an overview of the foundations of the SOQA-SimPack Toolkit and Section 3 presents SST's functionality and architecture in detail. In Section 4, the SOQA-SimPack Toolkit Browser is illustrated, which allows users to graphically perform similarity calculations in ontologies. Section 5 discusses related work and Section 6 concludes the paper.

2 Foundations of the SOQA-SimPack Toolkit

In this section, the SIRUP Ontology Query API [22] and SimPack [2] are presented, which form the basis for the SOQA-SimPack Toolkit.

2.1 The SIRUP Ontology Query API

To overcome the problems caused by the fact that ontologies can be specified in a manifold of ontology languages, the SIRUP Ontology Query API (SOQA) [22] was developed for the SIRUP approach to semantic data integration [21]. SOQA is an ontology language independent Java API for query access to ontological metadata and data that can be represented in a variety of ontology languages. Besides, data of concept instances can be retrieved through SOQA. Thus, SOQA facilitates accessing and reusing general foundational ontologies as well as specialized domain-specific ontologies through a uniform API that is independent of the underlying ontology language.

In general, ontology languages are designed for a particular purpose and, therefore, they vary in their syntax and semantics. To overcome these differences, the SOQA Ontology Meta Model [22] was defined. It represents modeling capabilities that are typically supported by ontology languages to describe ontologies and their components; that is, concepts, attributes, methods, relationships, instances, and ontological metadata. Based on the SOQA Ontology Meta Model, the functionality of the SOQA API was designed. Hence, SOQA provides users and applications with unified access to metadata and data of ontologies according to the SOQA Ontology Meta Model. In the sense of the SOQA Ontology Meta Model, an ontology consists of the following components:

– Metadata to describe the ontology itself. This includes name, author, date of last modification, (header) documentation, version, copyright, and URI (Uniform Resource Identifier) of the ontology as well as the name of the ontology language the ontology is specified in. Additionally, each ontology has extensions of all concepts, attributes, methods, relationships, and instances that appear in it.
– Concepts which are entity types that occur in the particular ontology's universe of discourse — that is, concepts are descriptions of a group of individuals that share common characteristics. In the SOQA Ontology Meta Model, each concept is characterized by a name, documentation, and a definition that includes constraints;[10] additionally, it can be described by attributes,

[10] In SOQA, axioms/constraints are subsumed by the definitions of the particular meta model elements.

methods, and relationships. Further, each concept can have direct and indirect super- and subconcepts, equivalent and antonym concepts, and coordinate concepts (that are situated on the same hierarchy level as the concept itself). For example, ontology language constructs like `<owl:Class...>` from OWL and `(defconcept...)` from PowerLoom are represented as concepts in the SOQA Ontology Meta Model.

– Attributes that represent properties of concepts. Each attribute has a name, documentation, data type, definition, and the name of the concept it is specified in.
– Methods which are functions that transform zero or more input parameters into an output value. Each method is described by a name, documentation, definition, its parameters, return type, and the name of the concept the method is declared for.
– Relationships that can be established between concepts, for instance, to build taxonomies or compositions. Similar to the other ontology components, a name, documentation, and definition can be accessed for each relationship. In addition, the arity of relationship, i.e., the number of concepts it relates, as well as the names of these related concepts are available.
– Instances of the available concepts that together form the extension of the particular concept. Each instance has a name and provides concrete incarnations for the attribute values and relationships that are specified in its concept definition. Furthermore, the name of the concept the instance belongs to is available.

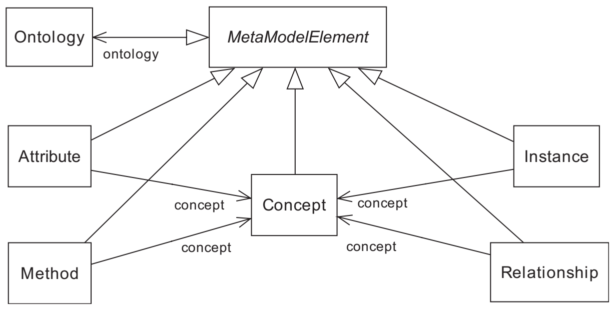


Fig. 1. Overview of the SOQA Ontology Meta Model as a UML Class Diagram

A UML class diagram of the SOQA Ontology Meta Model is shown in Figure 1. Note that the SOQA Ontology Meta Model is deliberately designed not only to represent the least common denominator of modeling capabilities of widely-used ontology languages. In deciding whether or not to incorporate additional functionality that is not supported by some ontology languages, we opted for including these additional modeling capabilities (e.g., information on methods, antonym concepts, ontology authors, etc.), provided that they are useful for users of the SOQA API and available in important ontology languages.

Architecturally, the SOQA API reflects the Facade [6] design pattern. That is, SOQA provides a unified interface to a subsystem that retrieves information from ontologies, which are specified in different ontology languages. Through the SOQA Facade, the internal SOQA components are concealed from external clients; instead, a single point for unified ontology access is given (see Figure 2). For example, the query language SOQA-QL [22] uses the API provided by the SOQA Facade to offer declarative queries over data and metadata of ontologies that are accessed through SOQA. A second example for an external SOQA client is the SOQA Browser [22] that enables users to graphically inspect the contents of ontologies independent of the ontology language they are specified in. Last, but not least, (third-party) Java applications can be based on SOQA for unified access to information that is specified in different ontology languages. Possible application areas are virtual organizations, enterprise information and process integration, the Semantic Web, and semantics-aware universal data management.

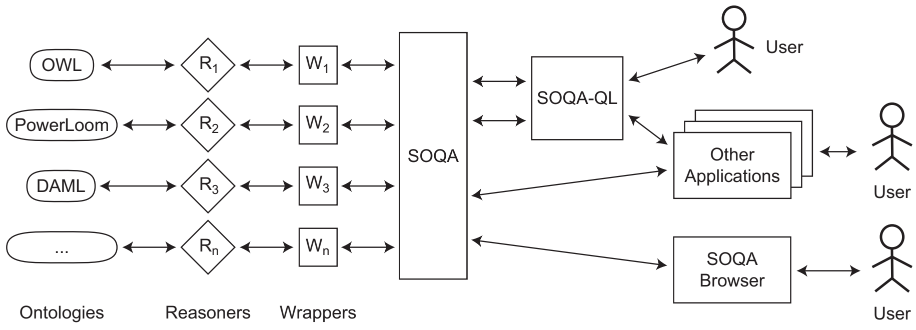


Fig. 2. Overview of the SOQA Software Architecture

Internally, ontology wrappers are used as an interface to existing reasoners that are specific to a particular ontology language (see Figure 2). Up to now, we have implemented SOQA ontology wrappers for OWL, PowerLoom, DAML, and the lexical ontology WordNet [11].

2.2 SimPack

SimPack is a generic Java library of similarity measures for the use in ontologies. Most of the similarity measures were taken from the literature and adapted for the use in ontologies. The library is generic, that is, the measures can be applied to different ontologies and ontology formats using wrappers. The question of similarity is an intensively researched subject in the computer science, artificial intelligence, psychology, and linguistics literature. Typically, those studies focus on the similarity between vectors [1, 17], strings [14], trees or graphs [18], and objects [7]. In our case we are interested in the similarity between resources in ontologies. Resources may be concepts (classes in OWL) of some type or individuals (instances) of these concepts. The remainder of this section will discuss different types of similarity measures, thereby explaining a subset of the measures implemented in SimPack.[11]

Vector-based Measures One group of similarity measures operates on vectors of equal length. To simplify their discussion, we will discuss all measures as the similarity between the (binary) vectors $\mathbf{x}$ and $\mathbf{y}$, which are generated from the resources R_x and R_y of some ontology O. The procedure to generate these vectors depends on how one looks at the resources. If the resources are considered as sets of features (or properties in OWL terminology), finding all the features for both resources results in two feature sets which are mapped to binary vectors and compared by one of the measures presented in Equation 1 through 3. For instance, if resource R_x has the properties *type* and *name* and resource R_y *type* and *age*, the following vectors $\mathbf{x}$ and $\mathbf{y}$ result using a trivial mapping M_1 from sets to vectors:

$$\mathbb{R}_x = \{type, name\} \Rightarrow \mathbf{x}' = \begin{pmatrix} 0 \\ name \\ type \end{pmatrix} \Rightarrow \mathbf{x} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

$$\mathbb{R}_y = \{type, age\} \Rightarrow \mathbf{y}' = \begin{pmatrix} age \\ 0 \\ type \end{pmatrix} \Rightarrow \mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Typically, the cosine measure, the extended Jaccard measure, and the overlap measure are used for calculating the similarity between such vectors [1]:

$$sim_{cosine}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 \cdot ||\mathbf{y}||_2} \tag{1}$$

$$sim_{jaccard}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2^2 + ||\mathbf{y}||_2^2 - \mathbf{x} \cdot \mathbf{y}} \tag{2}$$

$$sim_{overlap}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\min(||\mathbf{x}||_2^2, ||\mathbf{y}||_2^2)} \tag{3}$$

In these equations, $||\mathbf{x}||$ denotes the L^1-norm of $\mathbf{x}$, i.e. $||\mathbf{x}|| = \sum_{i=1}^{n} |x_i|$, whereas $||\mathbf{x}||_2$ is the L^2-norm, thus $||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$. The cosine measure quantifies the similarity between two vectors as the cosine of the angle between the two vectors whereas the extended Jaccard measure computes the ratio of the number of shared attributes to the number of common attributes [19].

[11] We have also introduced a formal framework of concepts and individuals in ontologies but omit it here due to space limitations. Please refer to [2] for further details about the formal framework.

String-based Measures A different mapping M_2 from the feature set of a resource makes use of the underlying graph representation of ontologies. In this mapping, a resource R is considered as starting node to traverse the graph along its edges where edges are properties of R connecting other resources. These resources in turn may be concepts or, eventually, data values. Here, these sets are considered as vectors of strings, $\mathbf{x}$ and $\mathbf{y}$ respectively. The similarity between strings is often described as the edit distance (also called the Levenshtein edit distance [9]), that is, the minimum number of changes necessary to turn one string into another string. Here, a change is typically either defined as the insertion of a symbol, the removal of a symbol, or the replacement of one symbol with another. Obviously, this approach can be adapted to strings of concepts (i.e., vectors of strings as the result of mapping M_2) rather than strings of characters by calculating the number of insert, remove, and replacement operations to convert vector $\mathbf{x}$ into vector $\mathbf{y}$, which is defined as $xform(\mathbf{x}, \mathbf{y})$. But should each type of transformation have the same weight? Is not the replacement transformation, for example, comparable with a deleting procedure followed by an insertion procedure? Hence, it can be argued that the cost function c should have the behavior $c(delete) + c(insert) \geq c(replace)$. We can then calculate the worst case (i.e., the maximum) transformation cost $xform_{wc}(\mathbf{x}, \mathbf{y})$ of $\mathbf{x}$ to $\mathbf{y}$ by replacing all concept parts of $\mathbf{x}$ with parts of $\mathbf{y}$, then deleting the remaining parts of $\mathbf{x}$, and inserting additional parts of $\mathbf{y}$. The worst case cost is then used to normalize the edit distance resulting in

$$sim_{levenshtein}(R_x, R_y) = \frac{xform(\mathbf{x}, \mathbf{y})}{xform_{wc}(\mathbf{x}, \mathbf{y})} \tag{4}$$

Full-text Similarity Measure We decided to add a standard full-text similarity measure sim_{tfidf} to our framework. Essentially, we exported a full-text description of all concepts in an ontology to their textual representation and built an index over the descriptions using Apache Lucene[12]. For this, we used a Porter Stemmer [13] to reduce all words to their stems and applied a standard, full-text TFIDF algorithm as described in [1] to compute the similarity between concepts.

TFIDF counts the frequency of occurrence of a term in a document in relation to the word's occurrence frequency in a whole corpus of documents. The resulting word counts are then used to compose a weighted term vector describing the document. In such a TFIDF scheme, the vectors of term weights can be compared using one of the vector-based similarity measures presented before.

Distance-based Measures The most intuitive similarity measure of concepts in an ontology is their distance within the ontology [15], defined as the number of sub-/super-concept (or *is-a*) relationships between them. These measures make use of the hierarchical ontology structure for determining the semantic similarity between concepts. As ontologies can be represented by rooted, labeled and

[12] `http://lucene.apache.org/java/docs/`

unordered trees where edges between concepts represent relationships, distances between concepts can be computed by counting the number of edges on the path connecting two concepts. Sparrows, for example, are more similar to blackbirds than to whales since they reside closer in typical biological taxonomies. The calculation of the ontology distance is based on the specialization graph of concepts in an ontology. The graph representing a multiple inheritance framework is not a tree but a directed acyclic graph. In such a graph, the ontology distance is usually defined as the shortest path going through a common ancestor or as the shortest path in general, potentially connecting two concepts through common descendants/specializations.

One possibility to determine the semantic similarity between concepts is sim_{edge} as given in [16] (but normalized), which is a variant of the edge counting method converting from a distance (dissimilarity) into a similarity measure:

$$sim_{edge}(R_x, R_y) = \frac{2 * MAX - len(R_x, R_y)}{2 * MAX} \tag{5}$$

where MAX is the length of the longest path from the root of the ontology to any of its leaf concepts and $len(R_x, R_y)$ is the length of the shortest path from R_x to R_y.

A variation of the edge counting method is the conceptual similarity measure introduced by Wu & Palmer [20]:

$$sim_{con} = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \tag{6}$$

where N_1, N_2 are the distances from concepts R_x and R_y, respectively, to their **M**ost **R**ecent **C**ommon **A**ncestor $MRCA(R_x, R_y)$ and N_3 is the distance from $MRCA(R_x, R_y)$ to the root of the ontology.

Information-Theory-based Measures The problem of ontology distance-based measures is that they are highly dependent on the (frequently) subjective construction of ontologies. To address this problem, researchers have proposed measuring the similarity between two concepts in an ontology in terms of information-theoretic entropy measures [16, 10]. Specifically, Lin [10] argues that a class (in his case a word) is defined by its use. The information of a class is specified as the probability of encountering a class's (or one of its descendants') use. In cases where many instances are available, the probability p of encountering a class's use can be computed over the instance corpus. Alternatively, when the instance space is sparsely populated (as currently in most Semantic Web ontologies) or when instances are also added as subclasses with is-a relationships (as with some taxonomies), then we propose to use the probability of encountering a subclass of a class. The entropy of a class is the negative logarithm of that probability. Resnik [16] defined the similarity as

$$sim_{resnik}(R_x, R_y) = \max_{R_z \in S(R_x, R_y)} [-\log_2 p(R_z)] \tag{7}$$

where $S(R_x, R_y)$ is the set of concepts that subsume both R_x and R_y, and $p(R_z)$ is the probability of encountering a concept of type z (i.e., the frequency of concept type z) in the corresponding ontology.

Lin defined the similarity between two concepts slightly differently:

$$sim_{lin}(R_x, R_y) = \frac{2 \log_2 p(MRCA(R_x, R_y))}{\log_2 p(R_x) + \log_2 P(R_y)} \tag{8}$$

Intuitively, this measure specifies similarity as the probabilistic degree of overlap of descendants between two concepts.

3 The SOQA-SimPack Toolkit

The SOQA-SimPack Toolkit (SST) is an ontology language independent Java API that enables generic similarity detection and visualization in ontologies. Simply stated, SST accesses data concerning concepts to be compared through SOQA; this data is then taken as an input for the similarity measures provided by SimPack. That is, SST offers ontology language independent similarity calculation services based on the uniform view on ontological content as provided by the SOQA Ontology Meta Model. SST services that have already been implemented include:

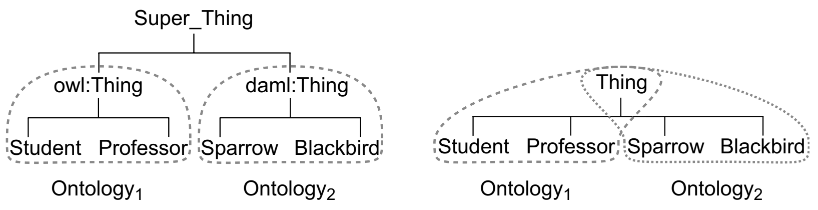
- Similarity calculation between two concepts according to a single similarity measure or a list of them.
- Similarity calculation between a concept and a set of concepts according to a single or a list of similarity measures. This set of concepts can either be a freely composed list of concepts or all concepts from an ontology taxonomy (sub)tree.
- Retrieval of the k most similar concepts of a set of concepts for a given concept according to a single or a list of similarity measures. Again, this set of concepts can either be a freely composed list of concepts or all concepts from an ontology taxonomy (sub)tree.
- Retrieval of the k most *dis*similar concepts of a set of concepts for a given concept according to a single or a list of similarity measures. As before, a freely composed list of concepts or all concepts from an ontology taxonomy (sub)tree can be used to specify the set of concepts.

Note that for all calculations provided by SST, the concepts involved can be from *any* ontology that is connected through SOQA.[13] That is, not only is it possible to calculate similarities between concepts from a single ontology (for example, Student and Employee from the DAML University Ontology) with a given set of SimPack measures, but also can concepts from different ontologies be used

[13] Generally, this is every ontology that can be represented in an ontology language. In fact, it is every ontology that is represented in a language for which a SOQA wrapper is available.

in the very same similarity calculation (for example, Student from the PowerLoom Course Ontology can be compared with Researcher from WordNet). For all SST computations, the results can be output textually (floating point values or sets of concept names, depending on the service). Alternatively, calculation results can automatically be visualized and returned by SST as a chart.

Using concepts from different ontologies in the same similarity calculation is enabled by the fact that in SST, all ontologies are incorporated into a single ontology tree. That is, the root concepts of the available ontologies (e.g., owl:Thing) are direct subconcepts of a so-called Super_Thing root concept. This makes it possible that, for instance, not only vector- and text-based similarity measures, but also distance-based measures that need a contiguous, traversable path between the concepts can be applied to concepts in SST. Alternatively, we could have replaced all root concepts of all ontologies with one general Thing concept. This, however, is a first step into the direction of ontology integration by mapping semantically equivalent concepts from different ontologies and not our goal in this research (consequentially, the ontologies should then completely be merged). Moreover, replacing the roots with Thing means, for example for OWL ontologies, that all direct subconcepts of owl:Thing from arbitrary domains are put directly under Thing and, thus, become immediate neighbors, blurring which ontology and domain a particular concept originates from. This is illustrated in Figure 3: Whereas the university domain of ontology$_1$ and the ornithology domain of ontology$_2$ remain separated in the first case, they are jumbled in the second. However, not mixing arbitrary domains is essential for distance-based similarity measures which found their judgments on distances in graphs (in Figure 3(b), Student is as similar to Professor as to Blackbird, due to the equality of the graph distances between Student, Professor, and Blackbird). Hence, we opted for introducing the Super_Thing concept as the root of the tree of ontologies in the SOQA-SimPack Toolkit.



(a) Classifying All Ontologies Below Super_Thing

(b) Replacing the Root Concepts of All Ontologies with Thing

Fig. 3. Comparison of Approaches to Building a Single Tree for a Set of Ontologies

Like SOQA, the SOQA-SimPack Toolkit architecturally reflects the Facade design pattern: SST provides a unified interface to a subsystem which is in charge of generic similarity calculations based on data from ontologies that are specified in different ontology languages. The SST Facade shields external clients from its internal components and represents a single access point for unified ontological similarity services (see Figure 4). External users of the services provided by the SST Facade include:

– The SOQA-SimPack Toolkit Browser that is a tool to graphically perform similarity calculations in ontologies independent of the ontology language they are specified in;
– (Third-party) Java applications that use SST as a single point of access to generic similarity detection and visualization services as provided by the SST Facade. Possible application areas are ontology alignment and integration, Semantic Web (service) discovery, data clustering and mining, semantic interoperability in virtual organizations, and semantics-aware universal data management.

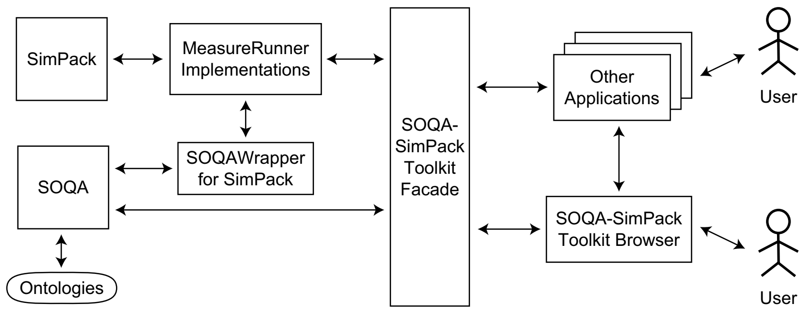

SimPack
MeasureRunner Implementations
SOQAWrapper for SimPack
SOQA
Ontologies
SOQA-SimPack Toolkit Facade
Other Applications
SOQA-SimPack Toolkit Browser
User
User

Fig. 4. Overview of the SOQA-SimPack Toolkit Software Architecture

Behind the SOQA-SimPack Toolkit Facade, MeasureRunner implementations are used as an interface to the different SimPack similarity measures available. Each MeasureRunner is a coupling module that is capable of retrieving all necessary input data from the SOQAWrapper for SimPack and initiating a similarity calculation between two single concepts for a particular similarity measure. For example, there is a TFIDFMeasureRunner that returns a floating point value expressing the similarity between two given concepts according to the TFIDF measure. More advanced similarity calculations, such as finding the k most similar concepts for a given one, are performed by tailored methods in the SOQA-SimPack Toolkit Facade itself based on the basic services supplied by underlying MeasureRunner implementations. By providing an additional MeasureRunner, SST can easily be extended to support supplementary measures (e.g., new measures or combinations of existing measures). Hence, the SOQA-SimPack Toolkit provides not only means for generic similarity detection, but can also be a fruitful playground for development and experimental evaluation of new similarity measures.

The SOQAWrapper for SimPack as another internal component of SST is in charge of retrieving ontological data as required by the SimPack similarity measure classes. This includes, for example, retrieval of (root, super, sub) concepts, provision of string sequences from concepts as well as depth and distance calculations in ontologies. Basically, all of this is done by accessing the necessary ontological data according to the SOQA Ontology Meta Model through SOQA and by providing the requested information as expected by SimPack. Summing up, the MeasureRunner implementations together with the SOQAWrapper for SimPack integrate both SOQA and SimPack on a technical level on behalf of the SST Facade.

Based on its Facade architecture, the SOQA-SimPack Toolkit provides a set of methods for ontology language independent similarity detection and visualization in ontologies. The following three method signatures (S1) to (S3) illustrate how similarities can be calculated with SST:

```
public double getSimilarity(String firstConceptName,           (S1)
   String firstOntologyName, String secondConceptName,
   String secondOntologyName, int measure)

public Vector<ConceptAndSimilarity> getMostSimilarConcepts( (S2)
   String conceptName, String conceptOntologyName,
   String subtreeRootConceptName, String subtreeOntologyName,
   int k, int measure)

public Image getSimilarityPlot(String firstConceptName,       (S3)
   String firstOntologyName, String secondConceptName,
   String secondOntologyName, int[] measures)
```

In the examples given before, method signature (S1) provides access to the calculation of the similarity between the two given concepts — the similarity measure to be used is specified by an integer constant (e.g., SOQASimPackToolkit-Facade.LIN_MEASURE for the measure by Lin). Note that in SST, for each concept we have to specify which ontology it originates from (parameters `firstOntologyName` and `secondOntologyName`, respectively). This is necessary since in SST's single ontology tree (into which all ontologies are incorporated), concept names are generally *not* unique anymore. For example, in case that more than one OWL ontology is used for similarity calculations, we have more than one owl:Thing concept as a direct subconcept of Super_Thing. Distinguishing which ontology the particular owl:Thing is the root of is essential (e.g., for graph-based measures) since the (direct) subconcepts for each owl:Thing concept differ. (S2) enables SST clients to retrieve the k most similar concepts for the given one compared with all subconcepts of the specified ontology taxonomy (sub)tree. In the result set, ConceptAndSimilarity instances contain for each of the k concepts the concept name, the name of its ontology, and the respective similarity value. Finally, (S3) computes the similarity between two concepts according to a set of measures and sets up a chart to visualize the computations.

Beyond access to similarity calculations, the SOQA-SimPack Toolkit Facade provides a variety of helper methods — for example, for getting information about a particular SimPack similarity measure, for displaying a SOQA Ontology Browser [22] to inspect a single ontology, or for opening a SOQA Query Shell to declaratively query an ontology using SOQA-QL [22].

Recall that in our running example from Section 1, a developer is looking for similarities among the concepts of five ontologies. In this scenario, the SOQA-SimPack Toolkit can be used, for instance, to calculate the similarity between the concept base1_0_daml:Professor from the DAML University Ontology and concepts from the other ontologies according to different SimPack similarity measures as shown in Table 1. Behind the scenes, SST initializes the necessary MeasureRunner instances which in turn manage the calculation of the desired similarity values by SimPack based on ontological information retrieved through SOQA. Note that for the plausibility of the calculated results, the SimPack measures as taken from the literature are responsible in general; in case that the available measures do not seem to be suitable for a particular domain, the set of available similarity measures can easily be extended by providing supplementary MeasureRunner implementations for further similarity measures.

Concept	*Conceptual Similarity*	*Levenshtein*	*Lin*	*Resnik*	*Shortest Path*	*TFIDF*
base1_0_daml:Professor	0.7778	1.0	0.8792	2.7006	1.0	1.0
univ-bench_owl:AssistantProfessor	0.1111	0.1029	0.0	0.0	0.0588	0.3224
COURSES:EMPLOYEE	0.1176	0.0294	0.0	0.0	0.0625	0.0475
SUMO_owl_txt:Human	0.1	0.0028	0.0	0.0	0.0526	0.0151
SUMO_owl_txt:Mammal	0.0909	0.0032	0.0	0.0	0.0476	0.0184

Table 1. Comparisons of base1_0_daml:Professor with Concepts from Other Ontologies

In addition to numeric results, the SOQA-SimPack Toolkit is able to visualize the results of similarity calculations. For instance, our developer can retrieve the k most similar concepts for base1_0_daml:Professor compared with all concepts from all five ontologies in our scenario. In response to this, SST can produce a bar chart as depicted in Figure 5. To generate the visualizations, SST creates data files and scripts that are automatically given as an input to Gnuplot[14], which then produces the desired graphics. Thus, the SOQA-SimPack Toolkit can effectively be employed to generically detect and visualize similarities in ontologies according to an extensible set of similarity measures and independently of the particular ontology languages in use.

[14] `http://www.gnuplot.info`

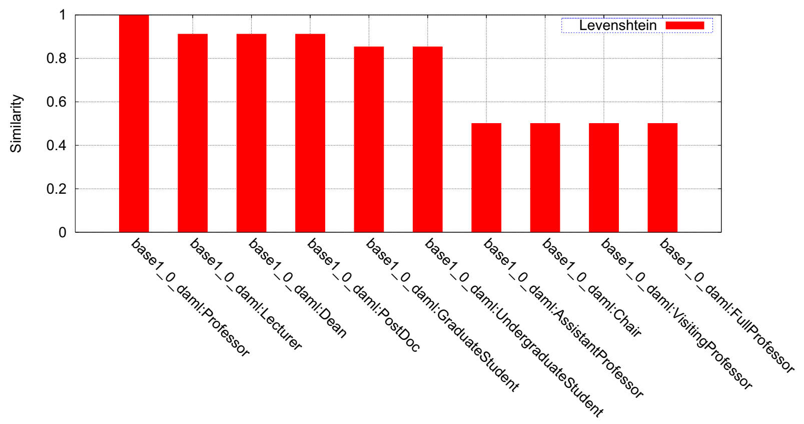


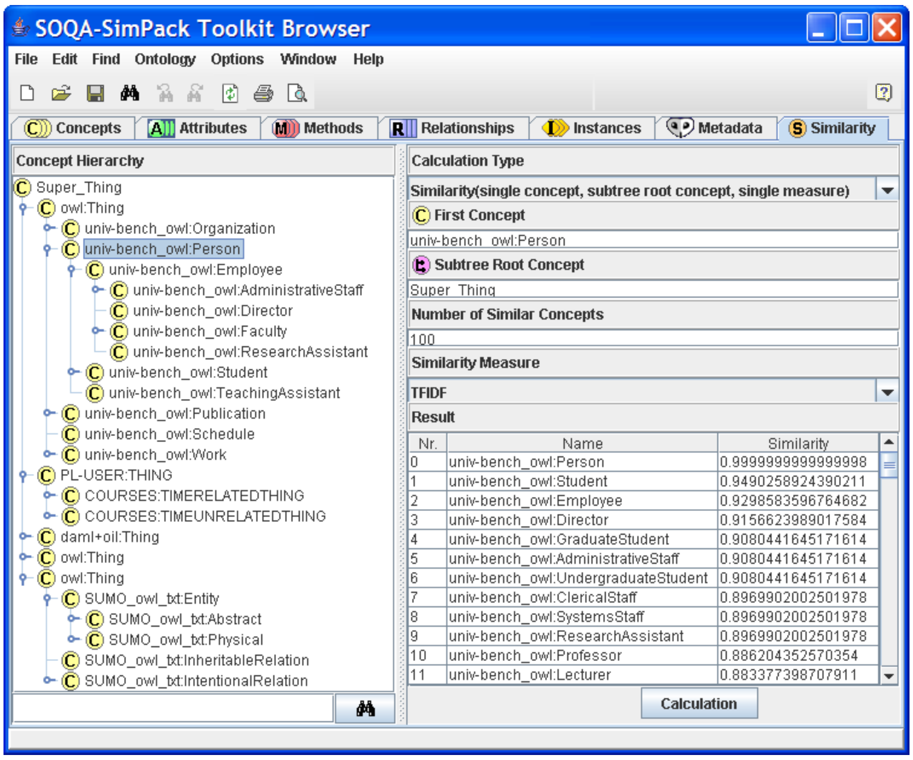
Fig. 5. SST Visualization of the Ten Most Similar Concepts for base1_0_daml:Professor

4 The SOQA-SimPack Toolkit Browser

The SOQA-SimPack Toolkit Browser is a tool that allows users to graphically perform similarity calculations and visualizations in ontologies based on the SOQA-SimPack Toolkit Facade. In general, it is an extension of the SOQA Browser [22] enabling users to inspect the contents of ontologies independently of the particular ontology language (i.e., according to the SOQA Ontology Meta Model). Based on the unified view of ontologies it provides, the SOQA-SimPack Toolkit Browser can be used to quickly survey concepts and their attributes, methods, relationships, and instances that are defined in ontologies as well as metadata (author, version, ontology language name, etc.) concerning the ontology itself.

In addition, the SOQA-SimPack Toolkit Browser provides an interface to all the methods of SST through its Similarity Tab (see Figure 6). That is, it is a tool for performing language independent similarity calculations in ontologies and for result visualization. In the Similarity Tab, users can select the similarity service to be run — for example, producing a graphical representation of the similarity calculation between two concepts according to the Resnik measure. Then, input fields are inserted into the Similarity Tab so that all necessary input values can be entered; here, concept names can directly be mouse-dragged from the Concept Hierarchy view and dropped into the respective input field. In the end, the calculated results are shown in tabular or graphical form, depending on the selected service.

In our running example, the SOQA-SimPack Toolkit Browser can first be employed by the developer to quickly get a unified overview of the five ontologies represented in PowerLoom, OWL, and DAML respectively. Subsequently, he or she can use the Similarity Tab and calculate, for instance, the k most similar concepts for univ-bench_owl:Person according to the TFIDF measure. The result



Calculation Type: Similarity(single concept, subtree root concept, single measure)

First Concept: univ-bench_owl:Person

Subtree Root Concept: Super_Thing

Number of Similar Concepts: 100

Similarity Measure: TFIDF

Result

Nr.	Name	Similarity
0	univ-bench_owl:Person	0.9999999999999998
1	univ-bench_owl:Student	0.9490258924390211
2	univ-bench_owl:Employee	0.9298583596764682
3	univ-bench_owl:Director	0.9156623989017584
4	univ-bench_owl:GraduateStudent	0.9080441645171614
5	univ-bench_owl:AdministrativeStaff	0.9080441645171614
6	univ-bench_owl:UndergraduateStudent	0.9080441645171614
7	univ-bench_owl:ClericalStaff	0.8969902002501978
8	univ-bench_owl:SystemsStaff	0.8969902002501978
9	univ-bench_owl:ResearchAssistant	0.8969902002501978
10	univ-bench_owl:Professor	0.886204352570354
11	univ-bench_owl:Lecturer	0.883377398707911

Fig. 6. The SOQA-SimPack Toolkit Browser and its Similarity Calculation Tab

is then presented in a table as shown in Figure 6. Thus, contrasting a conventional scenario where several ontology-language specific tools have to be employed for ontology access, the developer who takes advantage of SST does not have to cope with different ontology representation languages in use. Based on the unified view of ontologies as provided by the SOQA Ontology Meta Model, our developer can generically apply a rich and extensible set of SimPack similarity measures for similarity detection through the services offered by the SOQA-SimPack Toolkit. The results of these calculations can be presented as numerical values, textual lists (of concept names), or visualized in graphics. Hence, similarity detection in ontologies is facilitated and leveraged through the SOQA-SimPack Toolkit and its browser for the benefit of human users and applications.

5 Related Work

Closest to our work is the ontology alignment tool OLA presented by Euzénat et al. [4]. OLA is implemented in Java and relies on a universal measure for comparing entities of ontologies. Basically, it implements a set of core similarity functions which exploit different aspects of entities, such as textual descriptions, inter-entity relationships, entity class membership, and property restrictions. OLA relies on WordNet to compare string identifiers of entities. The main difference to our approach is OLA's restriction and dedication to the alignment of ontologies expressed in OWL-Lite. Using our generic approach, however, it is possible to compare and align entities of ontologies represented in a variety of ontology languages with the same set of similarity measures.

Noy and Musen's approach [12] follows similar goals: the comparison, alignment, and merging of ontologies to improve their reuse in the Semantic Web. The authors implemented a suite of tools called PROMPT that interactively supports ontology merging and the finding of correlations between entities to simplify the overall integration task. Compared to the SOQA-SimPack Toolkit, PROMPT is restricted to the comparison and merging of ontologies expressed in a few common ontology languages, such as RDF, DAML, and OWL. SST, on the other hand, offers the possibility to incorporate ontologies represented in a much broader range of languages. This includes not only ontologies described with recent Semantic Web languages, but also ones represented in traditional ontology languages, like PowerLoom. Furthermore, the SOQA-SimPack Toolkit supports ontologies supplied by knowledge bases, such as CYC [8], and by lexical ontology systems, such as WordNet.

Ehrig et al. [3] propose an approach that measures similarity between entities on three different layers (data layer, ontology layer, and context layer). Finally, an amalgamation function is used to combine the partial similarities of each layer and to compute the overall similarity between two entities. This approach differs from ours in its strong focus on entity layers and its amalgamation of individual layer-based similarity measures. Whilst it is easily possible to introduce such combined similarity measures through additional MeasureRunner implementations into the SOQA-SimPack Toolkit, we have left experiments with such measures for future work. In addition to this, we intend to extend the set of provided similarity measures in future, e.g., by incorporating measures from the SecondString project[15] which focuses on implementing approximate string-matching algorithms, and from SimMetrics[16] which presents similarity and distance metrics for data integration tasks.

6 Conclusions and Future Work

In this paper, we presented the SOQA-SimPack Toolkit, an ontology language independent Java API that enables generic similarity detection and visualization in ontologies. This task is central for application areas like ontology alignment and integration, Semantic Web (service) discovery, data clustering and mining, semantic interoperability in virtual organizations, and semantics-aware universal data management. SST is founded on (1) the SIRUP Ontology Query API, an

[15] `http://secondstring.sourceforge.net`
[16] `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`

ontology language independent Java API for query access to ontological metadata and data, and (2) SimPack, a generic Java library of similarity measures adapted for the use in ontologies.

The SOQA-SimPack Toolkit is extensible in two senses: First, further ontology languages can easily be integrated into SOQA by providing supplementary SOQA wrappers, and second, our generic framework is open to employ a multitude of additional similarity measures by supplying further MeasureRunner implementations. Hence, the extensible SOQA-SimPack Toolkit provides not only means for generic similarity detection, but can also be a fruitful playground for development and experimental evaluation of new similarity measures.

Contrasting a conventional scenario where several ontology-language specific tools have to be adopted for ontology access, users and applications taking advantage of the SOQA-SimPack Toolkit do not have to cope with different ontology representation languages in use. SST supports a broad range of ontology languages, including not only ontologies described with recent Semantic Web languages, but also ones represented in traditional ontology languages, like PowerLoom. Furthermore, ontologies supplied by knowledge bases, such as CYC, and by lexical ontology systems, such as WordNet, can be used in the SOQA-SimPack Toolkit.

Based on the unified view on ontologies as provided by the SOQA Ontology Meta Model, users and applications can generically apply a rich set of SimPack similarity measures for similarity detection in SST services. By taking advantage of an extensible library of ontological similarity measures, a variety of notions of "similarity" can be captured. Additionally, for all calculations provided by SST, concepts can be used from *any* ontology that is connectible through SOQA. This is accomplished by incorporating all ontologies into a single ontology tree. The results of these calculations can be presented as numerical values, textual lists (of concept names), or visualized in graphics. As an application that is based on SST, we provide the SOQA-SimPack Toolkit Browser, a tool to graphically perform similarity calculations in ontologies independent of the ontology language they are specified in. Thus, similarity detection in ontologies is facilitated and leveraged through the SOQA-SimPack Toolkit and its browser for the benefit of human users and applications.

Future work includes the implementation of additional similarity measures (especially for trees) and the provision of more advanced result visualizations. Besides, we intend to do a thorough evaluation to find the best performing similarity measures in different task domains and to experiment with more advanced, combined similarity measures. In the end, a comprehensive assessment of SST in the context of data and schema integration is planned.

References

1. R. Baeza-Yates and B. d. A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999.
2. A. Bernstein, E. Kaufmann, C. Kiefer, and C. Bürki. SimPack: A Generic Java Library for Similarity Measures in Ontologies. Technical report, University of Zurich,

Department of Informatics. `http://www.ifi.unizh.ch/ddis/staff/goehring/btw/files/ddis-2005.01.pdf`, 2005.

3. M. Ehrig, P. Haase, N. Stojanovic, and M. Hefke. Similarity for Ontologies - A Comprehensive Framework. In *Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, PAKM 2004*, December 2004.
4. J. Euzénat, D. Loup, M. Touzani, and P. Valtchev. Ontology Alignment with OLA. In *3rd EON Workshop, 3rd Int. Semantic Web Conference*, pages 333–337, 2004.
5. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *IJHCS*, 46(6):707–727, 1997.
6. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.
7. D. Gentner and J. Medina. Similarity and the Development of Rules. *Cognition*, 65:263–297, 1998.
8. D. B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):32–38, 1995.
9. V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
10. D. Lin. An Information-Theoretic Definition of Similarity. In *15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.
11. G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.
12. N. F. Noy and M. A. Musen. The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. *IJHCS*, 59(6):983–1024, 2003.
13. M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.
14. P.W.Lord, R. Stevens, A. Brass, and C.A.Goble. Investigating Semantic Similarity Measures Across the Gene Ontology: The Relationship Between Sequence and Annotation. *Bioinformatics*, 19(10):1275–83, 2003.
15. R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and Application of a Metric on Semantic Nets. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.
16. P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *IJCAI*, pages 448–453, 1995.
17. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.
18. D. Shasha and K. Zhang. Approximate Tree Pattern Matching. In *Pattern Matching Algorithms*, pages 341–371. Oxford University Press, 1997.
19. A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In *17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search*, pages 58–64. AAAI, July 2000.
20. Z. Wu and M. Palmer. Verb Semantics and Lexical Selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133–138, New Mexico State University, Las Cruces, New Mexico, 1994.
21. P. Ziegler and K. R. Dittrich. User-Specific Semantic Integration of Heterogeneous Data: The SIRUP Approach. In *First International IFIP Conference on Semantics of a Networked World (ICSNW 2004)*, volume 3226 of *Lecture Notes in Computer Science*, pages 44–64, Paris, France, June 17-19, 2004. Springer.
22. P. Ziegler, C. Sturm, and K. R. Dittrich. Unified Querying of Ontology Languages with the SIRUP Ontology Query API. In *Datenbanksysteme in Business, Technologie und Web (BTW 2005)*, volume P-65 of *Lecture Notes in Informatics*, pages 325–344, Karlsruhe, Germany, March 2-4, 2005. Gesellschaft für Informatik (GI).