Comparing Ontological Concepts to Evaluate Similarity

Anton Andrejko, Michal Barla, and Michal Tvarožek

Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
{andrejko, barla, tvarozek}@fiit.stuba.sk

Abstract. Various stages of data processing need to compare the concepts they work with. In case of the object-oriented paradigm we often use methods, which allow us to compare objects to other objects. In this paper, we present a method for the comparison of ontological concepts and the evaluation of their similarity. We identified two types of comparisons. First, the evaluation of distance between concepts by exploiting the taxonomy of classes in the used ontology. Second, the comparison of concepts' attributes, where we consider object type attributes and data type attributes. We describe the Concept Comparer software tool (ConCom), which compares concepts from an existing job offer domain ontology and quantitatively expresses their similarity.

1 Introduction

The importance of data and thus of data processing in contemporary systems is on a steady rise. Probably every information system needs some means to compare its data, e.g. for data sorting or data maintenance. Identifying differences between data is also crucial for recommender systems. The advent of the semantic web and thus that of ontological repositories introduced new challenges in the field of data processing. One of them is the comparison of ontological concepts (instances) and the evaluation of their similarity. Ontologies as such do not provide sophisticated means of concept comparison. Furthermore, the lack of proper comparison tools negatively affects the development and evaluation of systems such as adaptive presentation frameworks or user modeling frameworks.

In closed information spaces (e. g., well defined electronic courses) one can define relations between concepts explicitly. This approach is not feasible for open information spaces where new data are added automatically. We propose a method that compares ontological concepts and evaluates their measure of similarity. We also consider the scrutability of the results, which explains the reasons why a specific evaluation was made – what are the common and different aspects of the compared concepts.

The paper is structured as follows: In section 2 we give an overview of the current state of the art in the area. Section 3 describes the proposed algorithm of concept comparison. The software tool *ConCom* that realizes the method is described in section 4. Finally, in section 5 we give our concluding remarks.

2 Related works

Existing approaches aimed at ontology comparison focus on the reduction of manual effort that is necessary while comparing different ontologies to achieve increased reusability and interoperability between different knowledge bases. The process of comparing two different ontologies is known as ontology matching or ontology mapping.

There are several systems that compare (match) ontologies. *H-MATCH* uses four different matching models to provide a wide spectrum of metrics [1]. Linguistic and context weighting features are used. For linguistic comparison a thesaurus derived from WordNet is employed.

The *COMA++* tool extends the previous prototype of the ontology matching tool *COMA*. *COMA++* uses OWL class hierarchies and RDF labels to support matching between ontologies [2]. The similarity between two matching elements is expressed as the distance in the taxonomy tree.

The *GLUE* tool is based on machine learning techniques [3] and consists of three main modules – Distribution estimator, Similarity estimator and Relaxation labeler. The *Similarity estimator* module uses a predefined similarity function to compute the similarity measure for each pair of concepts and generates a similarity matrix of the compared concepts.

While all these approaches deal with the mapping of ontologies, our approach is focused on the comparison of individual concepts, which is a part of the ontology mapping process. In our approach however, we also consider the user model during the comparison which affects the weights of individual properties for the final result.

3 Our approach to concept comparison

Based on the aforementioned issues, we identified several beneficial usage scenarios for which a suitable concept comparison tool is required. We propose a method of concept comparison for ontological instances that should be suitable for the following scenarios:

- The evaluation of the *overall concept similarity*, e.g. for clusterization algorithms, semantic annotation tools or repository maintenance tools.
- The evaluation of similarity and the following *identification of similar and/or different aspects of concepts*, e.g. for user modeling tools [4].
- The evaluation of the *similarity of shared aspects of concepts*, e.g. for user suitability evaluation tools [5].

3.1 Method overview

The proposed method is based on the recursive evaluation of different comparison strategies with different metrics and their aggregation using different weights into a final measure of similarity. The used weights can be either static, computed at run time or optionally taken from a user model.

The similarity of two instances can be evaluated based on the similarity of classes the instances belong to and based on the attributes and their values of the instances. We defined the following comparison strategies based on the classes the instances belong to:

- Evaluation of the number of common classes for instances belonging to multiple classes simultaneously.
- Evaluation of the distance of the compared concepts in the taxonomy tree of classes in the ontology.
- Evaluation of the similarity of classes to which the instances belong based on their attributes.

We defined the following comparison strategies based on the instances and their attributes:

- Evaluation of the number and "size" of common, different and undefined attributes of instances.
- Evaluation of the similarity of instances based on the values of their common attributes.

Input. The input of the method consists of two identifiers (URIs) of individuals that should be compared and an optional user model identifier (URI). The method itself can also be used to compare individuals from different ontologies, in which case the mapping between these ontologies can be specified as another optional input. Furthermore, the required mode of operation (based on the aforementioned scenarios) and the desired metrics must be specified.

Output. The results of the method consist of the desired similarity metric(s) and an optional hierarchical tree of evaluations of similar/different aspects of the compared instances for further processing and scrutability.

3.2 Class based similarity evaluation

Class based similarity focuses on the aspect of semantic similarity between the compared instances – whether or not they represent semantically similar concepts.

Common and different classes. Since instances in ontologies can belong to multiple classes simultaneously, one way of measuring the similarity of instances at class level is to determine the number of common and different classes they belong to. Consequently, if two instances belong to several classes simultaneously, they are more similar than instances instances that have no common class (except the base class e.g., owl:thing).

Distance in the taxonomy tree. The similarity of concepts can be determined as a measure of the distance of the classes they belong to in the taxonomy tree of the respective ontology. This evaluation can be done by using different metrics.

One possibility is to look for the nearest common parent class in the taxonomy. In this case, the distance is the number of steps required to connect the classes of the two individuals via a common parent class. A example of a simple taxonomy of programming languages with three instances is shown in Fig. 1.

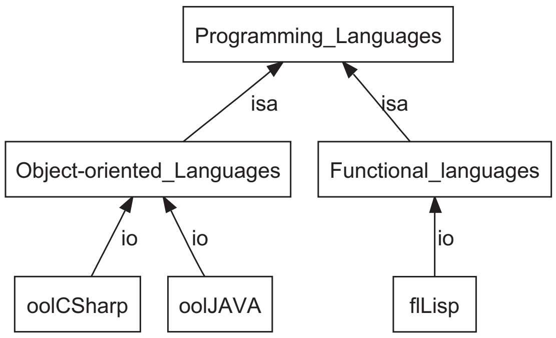

Fig. 1. Example of a simple class taxonomy of programming languages. The distance between *CSharp* and *Java* is zero since they both belong to the same class. This is obviously shorter than the distance of any object-oriented programming language to a functional programming language, e.g. *Lisp*, which is connected via the *Programming_ Languages* class in two steps.

If the compared individuals belong to several classes simultaneously, we compute the distances for each pair of classes separately. Here, different metrics can be used that take either the shortest, longest or average distance as a result. Furthermore, the aggregation of distances can be weighted based on the relative size or similarity of the respective classes.

Class similarity. In order to evaluate the similarity of classes the instances belong to we process the attributes of the respective classes and identify common and different attributes. In case of object type attributes (i.e. not literals) we also recursively compare the classes of the attributes themselves. Similar classes have more equivalent attributes than less similar classes, while classes that have little in common are likely to have little if any equivalent attributes.

Again, if the compared individuals belong to multiple classes simultaneously, different metrics and aggregation methods can be used.

3.3 Instance based similarity evaluation

While class based similarity evaluation relies on the high-level semantic similarity of classes two instances belong to, instance based similarity examines the attribute values of individual instances. It determines the "practical" level of similarity between two instances, which might even belong to the same class.

For example, class based similarity can determine that a *city* is quite similar to a *settlement*, yet only instance based similarity is able to determine that *New York* – an instance of *city* is quite different from *Black Rock* – an instance of *settlement*.

Number of common, different and undefined attribute values. The easiest way of comparing two instances is to determine the total number of their common, different and undefined attributes. For each common attribute of the compared instances we determine whether its values are equal, different or undefined. More similar instances will have more attributes with equal values than less similar ones. Furthermore, if a value for an attribute is undefined, it implies a lower measure of dissimilarity than if it were defined but different.

Similarity of common attributes with different values. Once different values of common attributes of instances are identified, we determine their measure of dissimilarity. Object type attributes and data type attributes must be treated differently due to their different nature. For different values of object type attributes, we use our method recursively and determine their respective similarity, whereas for data type attributes we directly compare the respective literals.

When comparing literals, a simple string comparison does not give satisfactory results, e.g. "computer skills", "Computer Skills", "COMPUTER SKILLS" only differ in capitalization, yet would appear totally different. The semantic dissimilarities in texts must be evaluated. A method for such evaluation was proposed in [6] and compares texts on several levels (whole text, sentences, words, symbols).

Different sets of predefined or computed weights can be used to aggregate the results of several attributes into one final value. For example, when comparing two user models to find out whether a recommender system can use the ratings of one user to recommend content to another user we should not consider user names as they are not relevant. Therefore, their respective weights would be zero.

Furthermore, weights can be computed by a given metric (e.g., the depth of the object property hierarchy) or from a user model using *relevance* of the respective attributes as described in [7].

3.4 Aggregation of results

Since the proposed comparison strategies can be combined in various ways it is necessary to aggregate their results in order to compute a single comprehensive measure of similarity. Thus, after evaluating individual strategies and their corresponding metrics for the compared instances, we use a set of predefined weights to aggregate partial results into a final similarity value. However, it is also important to provide partial results of individual strategies since partial results might be important for specific applications.

4 Experimental evaluation

4.1 Prototype implementation

The *ConceptComparer – ConCom* is a software tool developed in the context of NAZOU[1] project that implements the proposed method of concept comparison.

The *ConCom* tool will be implemented in JAVA and will use Corporate memory [8] of project NAZOU to access information stored in the ontological repository Sesame[2].

ConCom takes two URIs of instances as input. Additional optional inputs include a user model URI and mappings of ontological concepts from different ontologies. As output, *ConCom* returns an object, which contains a single number representing the similarity of the compared individuals as well as a set of methods which provide access to the results partial comparison strategies and metrics performed during the comparison.

4.2 Integration with other NAZOU tools

Project NAZOU [9] uses open information base of job offers represented by an ontology. The comparison of ontological instances is a useful service that can be successfully used by other tools of project NAZOU in several stages of data processing and presentation.

The *Ontea* tool, which creates ontological metadata using semantic annotation of textual documents [10] can use concept comparison to determine whether the ontological repository already contains semantically equivalent information.

The *Log Analyzer* tool [4] estimates user characteristics by analyzing user behavior within a system. Concept comparison, which gives information about common and different aspects of concepts can reveal reasons why users behave differently and can thus lead to user characteristics.

Concept comparison can also be useful for the creation of clusters. Clustering tools based on ontological representation such as *Job Cluster Navigator* [11] might yield better results than clustering tools based on probabilistic models of occurrences of words in documents such as *Aspect*.

[1] Project NAZOU, `http://nazou.fiit.stuba.sk`
[2] Sesame ontological repository, `http://www.openrdf.com`

Presentation tools can also benefit from concept comparison. The *Factic* faceted browser [5] uses concept comparison to compare job offer attributes and user preferences concerning job offers stored in a user model [7]. The *Criteria Search* tool enables users to find job offers relevant to a given set of criteria. It is capable of finding jobs that match user criteria more coarsely and sorts the results according to the fulfillment of selected criteria. A simple form of concept comparison is used to order the retrieved job offers.

5 Conclusions

We described issues of concepts and concept comparison in ontologies. We identified the need for comprehensive comparison of ontological instances that could be used a a wide array of data processing tools for the semantic web.

Furthermore, we proposed a method of comparison of ontological individuals and defined a set of comparison strategies that might be used both at the class level and instance level of abstraction. These include the evaluation of the distance of classes in ontologies and the recursive comparison of instances and the values of their attributes. Consequently, we also defined a way of combining individual strategies into one resulting measure of similarity between concepts.

Moreover, we included the “adaptation” of weights of individual attributes of an instance based on the relevance of the attributes stored in a user model. Lastly, we described the evaluation of the proposed method in the form of the *ConCom* tool within project NAZOU together with the possibilities of its integration with other tools of the project.

Future work includes the implementation of the *ConCom* tool, its integration with other tools and its evaluation in the domain of job offers and scientific papers with different weight settings and metrics. Future enhancements to the proposed method might include the definition of additional comparison strategies, improved support for work with different ontologies and broader use of user models.

Acknowledgment

This work was partially supported by the Slovak State Programme of Research and Development “Establishing of Information Society” under the contract No. 1025/04.

References

1. Castano, S., Ferrara, A., Montanelli, S., Racca, R.: From surface to intensive matching of semantic web ontologies. In: DEXA Workshops, IEEE Computer Society (2004) 140–144
2. Aumueller, D., Hai Do, H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In Ozcan, F., ed.: SIGMOD Conference, ACM (2005) 906–908

3. Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Learning to map between ontologies on the semantic web. In: WWW. (2002) 662–673
4. Barla, M.: Interception of user's interests on the web. In Wade, V., Ashman, H., Smyth, B., eds.: 4th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'06, Dublin, Ireland, Springer, LNCS 4018 (2006) 435–439
5. Tvarožek, M.: Personalized navigation in the semantic web. In Wade, V., Ashman, H., Smyth, B., eds.: 4th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'06, Dublin, Ireland, Springer, LNCS 4018 (2006) 467–471
6. Tury, M., Bieliková, M.: An approach to detection ontology changes. In: 1st Int. Workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE 06) at ICWE 2006, Palo Alto, California, ACM Press (2006) Accepted.
7. Andrejko, A., Barla, M., Bieliková, M.: Ontology-based user modeling for web-based information systems. In: Information Systems Development (ISD), Springer (2006)
8. Ciglan, M., Babik, M., Laclavík, M., Budinska, I., Hluchý, L.: Corporate memory: A framework for supporting tools for acquisition, organization and maintenance of information and knowledge. In: Proc. of 9-th Intl. Conf. ISIM'06 "Information Systems Implementation and Modelling", Brno, April, MARQ Ostrava. (2006) 185–192
9. Návrat, P., Bieliková, M., Rozinajová, V.: Methods and tools for acquiring and presenting information and knowledge in the web. In: Int. Conf. on Computer Systems and Technologies, CompSysTech 2005, Varna, Bulgaria (2005)
10. Laclavík, M., et al.: Semantic annotation based on regular expressions. In Vojtáš, P., ed.: ITAT 2005 - Workshop on Theory and Practice of Information Technologies. (2005) 305–306
11. Frivolt, G., Bieliková, M.: Topology generation for web communities modeling. In Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O., eds.: SOFSEM. Volume 3381 of Lecture Notes in Computer Science., Springer (2005) 167–177