Comparing Instances of the Ontological Concepts

Anton Andrejko and Mária Bieliková

Faculty of Informatics and Information Technologies,
Slovak University of Technology,
Ilkovičova 3, 842 16 Bratislava, Slovakia
{andrejko,bielik}@fiit.stuba.sk

Abstract. Knowing user's rating of displayed concepts we can acquire some characteristics for personalization purposes by analyzing the content. In this paper we present a method for comparison of instances of ontological concepts. During the comparison we examine the relationship of the compared concepts based on the used class taxonomy. We also traverse all of their attributes, which might be either data type or object type attributes. For each attribute we look for adequate attribute in the second instance to be compared. When comparing attributes, a simple comparison of strings does not give satisfactory results as the semantic dissimilarities need to be evaluated. We extend achieved similarity measure with content's attribute that caused different rating according to defined threshold values.

1 Introduction

On the Web there are many applications that somehow try to differ from the others and satisfy potential visitors or even make them use offered services again. One of the approaches is personalization provided by adaptive applications. Information about the user is held in the user model as a characteristic and consequently personalization of the content, navigation or presentation can be accomplished according to it.

One way to acquire the user's characteristics is to ask the user explicitly or observe one's behavior while working with the application (implicit feedback). Also mining user characteristics from logs can be helpful. The logging can be processed on client-side and/or server-side. In [1] there is described a method for acquisition of user characteristics from logs with semantics while an estimation of the user's actions from the logs is performed aimed at the user model update. The logging (and implicit feedback in general) requires sequential processing to transform acquired information into the user characteristics.

In this paper we use explicit feedback (user's rating given to the displayed concept) and we focus on analyzing of the content, namely similarity between concepts. Especially, while working in the Semantic Web environment we have concepts in ontological representation, i.e. we have meta-data describing the

semantics of the displayed content parts that we can use to deduce user's characteristics.

Knowing user's rating of displayed concepts we can acquire some characteristics by analyzing the content. Since the rating varies on different concepts we need to figure out reasons why it is low or high. For instance, consider concepts describing job offers in information technology (IT). One can stumble in hundreds of offers on the Web that advertise position for Java programmers requiring high school education, at least three years of previous experience, knowing basics in Web technologies, providing motivating salary, etc. Let us have two of them that differ only in the job location. Let us have the first one located somewhere in Europe, e.g. in London and the second one in the United States, e.g. in Washington, D.C. Assume we get different ratings for these two job offers. Probably the variety of evaluation could have been caused by the job location attribute. As the matter of fact, it is not important if the user from Europe prefers to work in Europe (high rating for job offer located in London) or we have the case of an adventurer who wants to try an overseas job (high rating for Washington).

From different ratings given to different concepts we can deduce particular user's characteristics (its value). On the other hand, in case we get the same ratings for different concepts we can deduce user's preferences (characteristic only). From our last example we could deduce that job location is not important factor for the user.

Therefore, we need to identify common and different aspects of the ontological concepts. To achieve this goal we have decided to compare concepts and determine the level of similarity for particular aspects.

The paper is structured as follows. In section 2 we give an overview of the current state of art in the area. Section 3 deals with the numerical evaluation of the similarity. In section 4 we describe the proposed method that realizes comparison of ontological instances on the basis of recursive traversing its structure. Finally, section 5 gives our concluding remarks.

2 Related work

So far, we have used the term concept mostly intuitively, but a concept is a set or a class of individual objects that can have simple properties, often called attributes, which are typically attached to the corresponding concepts [2]. The concept is sometimes used in place of class, where classes are a concrete representation of concepts [3]. At this point we need to point out that there are two different ways how to think about ontological concepts – intensional and extensional.

Considering intensional approach, the concept consists of a set of attributes that are its descriptors, whereas using extensional approach, the concept consists of a set of objects, i.e. instances of ontological concepts. The intensional approach is typical in creating domain ontologies on the Semantic Web, whereas the extensional approach is used mostly in Formal Concept Analysis (FCA) aimed at support the user in analyzing and structuring a domain of interest [4].

The approach that takes into account both ways is described in [4]. It proposes a method for computing similarity of FCA concepts and allows identification of different concepts that are semantically close. In FCA concept is defined within a context (O, A, R) where O stands for set of objects, A for set of attributes and R is binary relation between O and A. Considering all the concepts in this context the *concept lattice* can be constructed. The main drawback of this method is that similarity ontology holding similarity relations between attributes and entity names from domain ontology has to be given. For instance, semantic similarity between "city" and "capital" can be evaluated to 0.8. After we have domain ontology and context *similarity graph* can be constructed. The advantage of this approach is that total similarity for more than 2 concepts can be expressed as one number. Furthermore, the similarity of concepts from different contexts can be computed as well.

In [5] there is proposed a method of processing concept comparison emphasizing structural aspect of the concept. The comparison is accomplished in two phases. The first phase is focused on preprocessing the concepts that are about to be compared. Two graphs are built – *inheritance graph* that organizes ontological concepts according to a generalization hierarchy and *similarity graph* in which nodes relate to concept and edges have assigned similarity degree. In the second phase the similarity is evaluated according to three steps. First, flat structural similarity is computed exploiting structural slots (*part*, *related*, *predicate*). Second, hierarchical structure is exploited by using results from previous step and extending them by further elements according to the hierarchical relationships. In the third step, the final similarity measure between concepts is computed as a result of combination of two previous steps. The described method was evaluated in SymOntos system in the course of the European project aimed at construction and maintenance of tourism ontologies.

The problem of the similarity identification in ontologies is not a completely new idea and it is known as ontology mapping or ontology matching. Its aim is to increase reusability and interoperability between different ontologies covering the same application domain. In [6] there is described an approach aimed at identification of changes in ontology versions that uses comparing of instances as one of heuristics.

The approach described in [7] uses three independent similarity assessments to determine similarity between concepts. The first level is construction of the model that deals with synonyms to ensure that synonyms refer to the same entity. The second level incorporates semantics by using distinguishing features. The third level uses semantic relations (e.g. *is-a* relation) as the subject of comparison to find out whether connected entities are related to the same set of entity classes. Then, the distance between two concepts is measured by the shortest path.

In [8] there is described an approach that realizes ontology mapping in four stages, that include similarity of labels (classes, instances, relations), instances, structures and previous mapping results verified by the application. While comparing instances the Edit-Distance [9] method is used together with Glue approach. The GLUE tool is based on machine learning techniques [10]. It consists

of three main modules (Distribution estimator, Similarity estimator and Relaxation labeler). From our point of view, Similarity estimator model is interesting. It uses predefined similarity function to compute a similarity value for each pair of concepts and generates the similarity matrix referring to concepts being compared.

The common sign for all the mentioned approaches is that they do not try to investigate reasons that caused similarity for further processing. From this point of view we consider our approach as contributive.

3 Similarity evaluation

In general, we can formally define the similarity for two arbitrary objects x, y as follows [11]:

- $sim\,(x, y) \in [0..1]$,
- $sim\,(x, y) = 1 \rightarrow x = y$: concepts are identical,
- $sim\,(x, y) = 0$: concepts are different, they have nothing in common,
- $sim\,(x, x) = 1$: similarity is reflexive,
- $sim\,(x, y) = sim\,(y, x)$: the similarity is symmetric.

We count the total similarity continuously step by step as the different evaluation strategies are used, i.e. each strategy contributes to the total similarity with its own partial similarity evaluation as counted for two given objects. In this case we do not consider object as a part of the ontology triple (subject, predicate and object). Here, an object is everything what is intended to be compared. For instance, it can be an attribute (both object and data type), an object assigned to the object type attribute or a literal, what is a string in general. The total similarity measure of two concepts is given as:

$$totalSimilarity = \frac{\sum partialSimilarity}{evaluationCount},$$

where *partialSimilarity* is a number from interval $\langle 0, 1\rangle$ and says how much are compared objects similar while using particular comparing strategy, *evaluationCount* is the number of comparisons that have been processed.

The question is what will happen if we compare concepts with mutually different number of attributes. We need to make a decision whether we consider smaller or higher number of attributes.

Let us assume we have two concepts x, y that are identical. As we defined earlier the similarity measure of two identical concepts equals 1. Let us enrich the concept y with some additional attribute. If we do not consider all the attributes in the enumeration, the *evaluationCount* will not change and the total similarity measure while using the same strategies for the evaluation will remain 1. Since numerically evaluated similarity equal 1 means an identity, it is not true in this case and it contradicts one of the conditions we defined above. Therefore, we consider the higher number of attributes while counting total similarity measure.

Similarly, let us return to the problem when we have not found respective attribute in the second instance. In this case we do not need to count the similarity – similarity of the object and none object is zero. But we should consider this difference in the instances. Therefore, we increase only *evaluationCount* for every such object.

There is a space to use more strategies to evaluate similarity between two objects. The overview of methods that can be used is described in [12]. For instance, we can use different similarity strategies to evaluate similarity between two short strings "JobOffer" and "job offer". Different strategies can lead to various results of the partial similarity. Here, we could get similarity equal one but also similarity lower then one depending on used heuristics. Achieved similarity thus also relates to accuracy. Therefore, we take this fact into account and we use two ways to evaluate the total similarity measure:

- *similarity minimization* – if different partial similarities were counted we consider the smallest value as an addition to the total similarity,
- *similarity maximization* – if different partial similarities were counted we consider the highest value as an addition to the total similarity.

4 Similarity of Ontological Concepts Instances

The ontology contains additional information that can be acquired from its structure and thus help get better results than comparing instance's attributes only *one-to-one*. The evaluation of the similarity between ontological concepts, which are represented as class instances, is therefore performed according to the instance's structure. In the Figure 1 is shown a simple example of the part of ontology instance representing a job offer. The job offer ontology was developed in the course of the project NAZOU [13].

Every instance can consist of either object type or data type attributes. Attributes are in the figure represented as arcs. We use italic font for data type attributes (e.g. *jo:maxAmount*) in order to distinguish them from object type attributes. Moreover, object type attributes can be multiple (e.g. *jo:hasPrerequisity*). For simplicity, in the Figure 1 are multiple attributes surrounded by a rounded box.

4.1 Recursive Traversing of the Concept Structure

We have proposed a method based on the recursive evaluation of the attributes and objects they are connected to. The inputs and outputs of the method are depicted in the Figure 2.

The similarity evaluation begins with an estimation of the attributes that are directly connected to the instance identifier (the attribute corresponds to the predicate if we consider ontology representation as the set of triples – subject, predicate and object). We use DISTINCT keyword in the query for acquiring attributes. It assures that we will not get back two attributes with the same

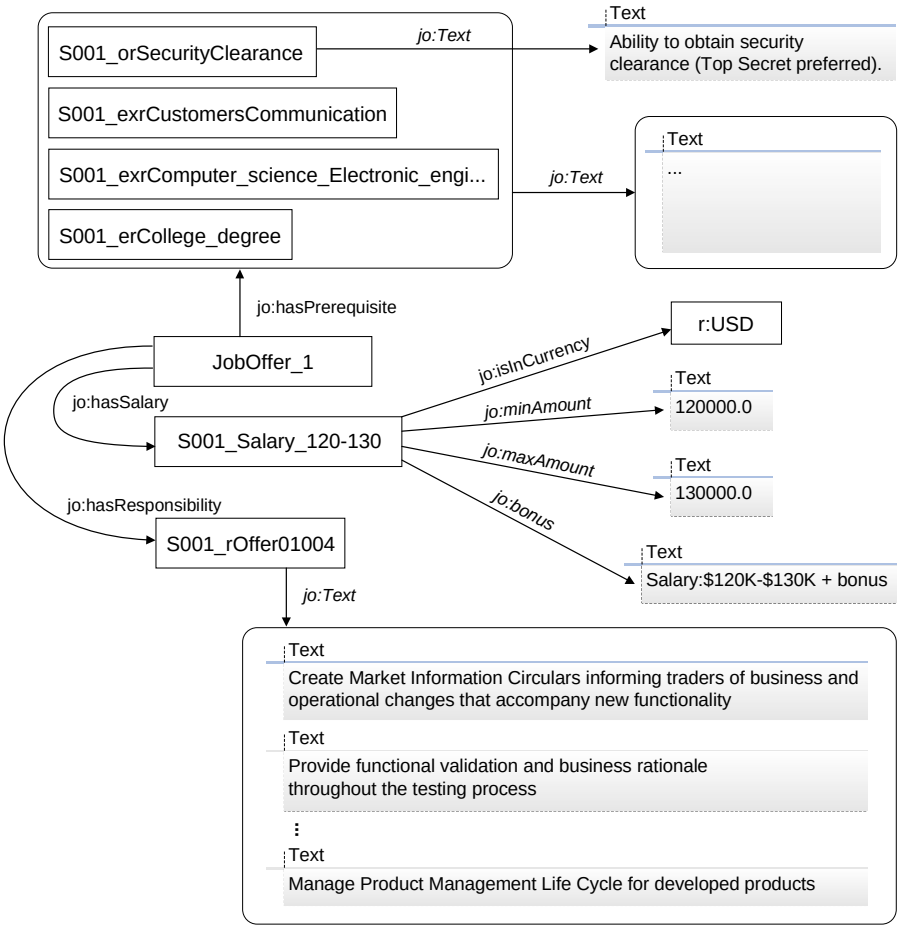

Fig. 1. An example of the ontological instance representing a part of job offer. Every object in the figure has its unified identifier but we present only object's label.

identifier. Here, we eliminate the default w3c attributes[1], since they can be found in both instances and have the same values. Their evaluation would be wasting of the effort with no profit.

For each attribute we look for the same attribute in the second instance. At this point it depends what is the type of the examined attribute. When data type attribute is evaluated, the algorithm ends by using strategies intended for comparing strings. Object type attributes are processed recursively by using assigned strategies until we reach data type level with evaluating as we mentioned above.

[1] e.g. *http://www.w3.org/1999/02/22-rdf-syntax-ns#type*

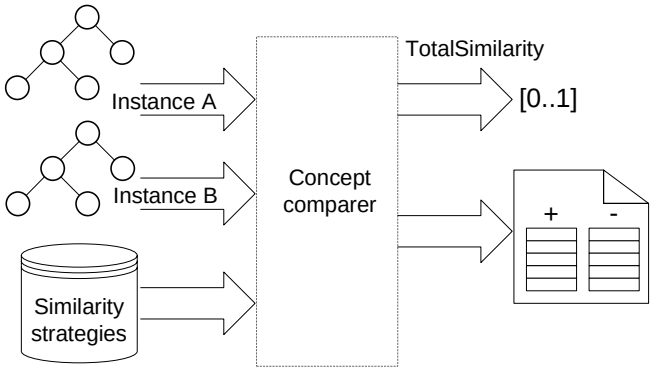

Fig. 2. Inputs and outputs of the method. The input for the comparing instances of ontological concepts is two instances and a set of similarity strategies. The output is quantitatively expressed similarity measure between instances and output object that consists of two sets of attributes – a positive and a negative.

For the object type attribute two sets of objects that are connected to that attribute (e.g. *hasPrerequisite*) in each instance are acquired. In these sets we can apply variety of strategies related to object attributes. Comparing all objects from two sets mutually gives complexity $O(n^2)$ but can be decreased since we eliminate identical identifiers after they are found. On the other hand, we have to deal with the left over attributes to which the respective attributes in the second instance have not been found. Since we compare instances from the same ontology we do not process similarity evaluation for them, i.e. we do not look for match between two attributes names. We only adjust the final similarity measure. The following piece of pseudo code demonstrates the basic idea of the algorithm:

```
1. ID = FirstInstance
2. get all attributes where ID stands for Subject
3. for each attribute
4.   pick an attribute from SecondInstance
5.     if attribute is data type
6.       use string strategies to evaluate similarity
7.     if attribute is object type
8.       use class based similarity strategies to
           evaluate similarity
9.       ID = Object
10.      go to line 2
11.  count partial similarity
12. count total similarity
```

As we can see in the example above, we use identifier ID to be able acquire attributes for any object and use the algorithm recursively. In the first run there

is assigned URI of the first instance to the ID. After all the attributes are acquired the examination and comparing with the second instance can start.

While traversing the ontological structure we have to consider the fact that an attribute can have its *inverse attribute* as well. For instance, Washington, D.C. is connected to the job offer with the attribute *jo:hasDutyLocation*. Since more than one job offer can be located in Washington, D.C., we require all of them to refer to the same object (e.g. Washington, D.C. *jo:isDutyLocationOf* others job offers). If we did not take inverse attribute into account the traverse algorithm would continue through it to other instances. The same problem causes symmetric attributes; therefore we filter out all the inverse and symmetric attributes.

4.2 Investigating rating's reasons

Our goal is to evaluate similarity between instances of ontological concepts and also to investigate the reasons of user's rating given to the displayed content. From the user's evaluation we can deduce user's likes or dislikes. If the content includes an attribute that the user likes it will likely influence her rating towards higher (or positive) values. On the other hand, attributes of the content that the user dislikes will influence rating towards lower (or negative) values.

To be able distinguish between preferences (negative and positive) we introduce a threshold value. Since one can also express neutral attitude we have proposed two threshold values – *positive* and *negative* defined respectively to the preferences. Those values divide attributes into three sets. The purpose of searching for rating's reason is to transform attributes into the user model characteristics for further personalization. The transformation is not the concern of this paper. The adequate personalization requires characteristics we are confident about that to reflect real preferences of the user. Therefore, we are interested only in the most outer sets and we omit the neutral set. We propose for positive threshold similarity measure 0.85 and 0.15 for negative threshold. The adjustment of the thresholds will be a matter of further experiments.

The lists with attributes are being filled as the instances are being compared. After the similarity between objects is evaluated according to reached similarity measure, the attribute, the object is related to, is assigned to the one of the lists. We store in the lists URIs of the attributes.

5 Conclusions

The paper proposes the method that processes comparison of instances of ontological concepts. The method is based on the recursive traversing of instance and comparing particular parts with respective parts in the second instance. Several different strategies can be used. Besides expressing similarity between instances (we use different strategies to evaluate data type and object attributes) we focus on reasons of rating given to the displayed concept. We proposed two thresholds to distinguish between user's preferences and according to thresholds we build

two sets of attributes – a positive and a negative. Those can be used by other tools to be transformed into user characteristics in the user model.

The evaluated similarity can be useful as a support for clusterization algorithms [14], semantic annotation tools [15] or repository maintenance tools [16]. Furthermore, the part realizing identification of similar or different aspects can be used for personalization purposes by user modeling tools.

The proposed method is not very suitable to compare instances from different ontologies since we do not evaluate similarity between attributes. This might be one of future extensions of the method. Furthermore, we plan to extend the method with user model involvement while evaluating similarity. Considering user preferences from the user model makes the result of comparison more accurate from the personalization point of view. In the future, we want to focus on structure of the instance while counting similarity and introduce weighting respecting the depths of the instance since it can vary in different ontologies.

A software prototype called *ConCom* (concept comparer) that realizes the proposed method is being developed. The future work includes integration with other tools and two types of experiments.

First group of experiments will be focused on verification of the self-standing software prototype and adjusting thresholds. Here, we want to approve that similarity between concepts fulfill similarity criteria described in section 3, namely identity, symmetry and reflexivity.

The second group of experiments will be focused on attributes acquired while comparing instances and their usefulness for the *Log Analyzer* tool that estimates user characteristics by analyzing user behavior within a system.

Acknowledgment

This work was partially supported by the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04.

References

1. Andrejko, A., Barla, M., Bieliková, M., Tvarožek, M.: User characteristics acquisition from logs with semantics. In: ISIM '07 Information Systems and Formal Models: 10th International Conference on Information System Implementation and Modeling. (2007) 103–110
2. Baader, F., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. In: The description logic handbook: Theory, implementation, and applications. Cambridge University Press (2003)
3. Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C. In: A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. The University Of Manchester (2004)
4. Formica, A.: Ontology-based concept similarity in formal concept analysis. Information Sciences **176**(18) (2006) 2624–2641
5. Formica, A., Missikoff, M.: Concept similarity in symontos: An enterprise management tool. The computer Journal **45**(6) (2003) 583–595

6. Tury, M., Bieliková, M.: An approach to detection ontology changes. In: ICWE '06: Workshop proceedings of the sixth international conference on Web engineering, Palo Alto, California, ACM Press (2006)
7. Rodríguez, M.A., Egenhofer, M.J.: Determining semantic similarity among entity classes from different ontologies. IEEE transactions on knowledge and data engineering **15**(2) (2003) 442–456
8. Liu, X., Wang, Y., Wang, J.: Towards a semi-automatic ontology mapping – an approach using instance based learning and logic relation mining. In: Proceedings of Fifth Mexican International Conference on Artificial Intelligence (MICAI'06), IEEE (2006)
9. AnHai, D., Jayant, M., Robin, D., Pedro, D.and Alon, H.: Learning to match ontologies on the semantic web. The VLDB Journal **12**(4) (2003) 303–319
10. Doan, A.H.e.a.: Learning to map between ontologies on the semantic web. In: Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA, ACM Press (2002)
11. Bisson, G.: Why and how to define a similarity measure for object based representation systems. In: Towards Very Large Knowledge Bases. (1995) 236–246
12. Andrejko, A., Barla, M., Tvarožek, M.: Comparing ontological concepts to evaluate similarity. In: Tools for Acguisition, Organisation and Presenting of Information and Knowledge: Research Project Workshop. (2006) 71–78
13. Návrat, P., Bieliková, M., Rozinajová, V.: Acquiring, organising and presenting information and knowledge from the web. In: CompSysTech'06, Veliko Turnovo, Bulgaria, Bulgarian Chapter of ACM (2006)
14. Frivolt, G., Pok, O.: Comparison of graph clustering approaches. In Bieliková, M., ed.: Proceedings in IIT-SRC 2006, Veliko Turnovo, Bulgaria, Slovak University of Technology (2006) 168–175
15. Laclavík, M., Šeleng, M., Gatial, E., Balogh, Z., Hluchý, L.: Ontology based text annotation - OnTeA. In Duzi, M., Jaakkola, H., Kangassalo, H., Kiyoki, Y., eds.: Information Modelling and Knowledge Bases XVIII, Amsterdam, IOS Press (2006)
16. Ciglan, M., Babík, M., Laclavík, M., Budinská, I., Hluchý, L.: Corporate memory: A framework for supporting tools for acquisition, organization and maintenance of information and knowledge. In Duzi, M., Jaakkola, H., Kangassalo, H., Kiyoki, Y., eds.: Proceedings of 9th International Conference ISIM'06 "Information Systems Implementation and Modelling", Brno, MARQ Ostrava (2006) 185–192