Personalized Comparing Instances of Domain Ontology Concepts

Anton Andrejko, Mária Bieliková

Institute of Informatics and Software Engineering
Faculty of Informatics and Information technologies
Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava
E-mail: {andrejko,bielik}@fiit.stuba.sk

Abstract

Characteristics in the user model can be acquired and maintained in several ways, namely by means of implicit or explicit feedback, observation of user's actions, log analysis, etc. We use analysis of the content that is presented to a user. We assume that comparing documents and analyzing their common and different aspects can lead to information about user's interests. We work in the Semantic Web environment where ontologies are used as a mean for content representation. We proposed a recursive method to evaluate similarity of ontological concepts. But computed similarities do not take into account user's individuality, i.e. the same results are computed for each user. User's perception of the similarity measure is subjective. Therefore, we extend the method to compute similarity with regard to the user.

1 Introduction

Personalization is usually based on the user model holding significant information about each user stored as characteristics. The more information about the user is available the more accurate personalization can be provided. Since user characteristics are changing as the user works with an application, the user model should always reflect the real user. Therefore, user's characteristics need to be maintained or new ones acquired if necessary.

Approaches to user characteristics (e.g., implicit or explicit feedback, observation of user's actions) differ in how intrusive towards the user they are and how confident information about the user acquire. We use an analysis of the presented content. In combination with explicit feedback, which is considered as the most confident source of information, user characteristics can be acquired. We assume that comparing documents, analyzing common and different aspects can lead to information about user's interests.

Let us consider two job offers that are in all properties identical but differ in the *job location*. Let us locate the first one in Washington, D.C. and another in London. Figure 1 illustrates this example in the user interface of our personalized faceted browser Factic [14].

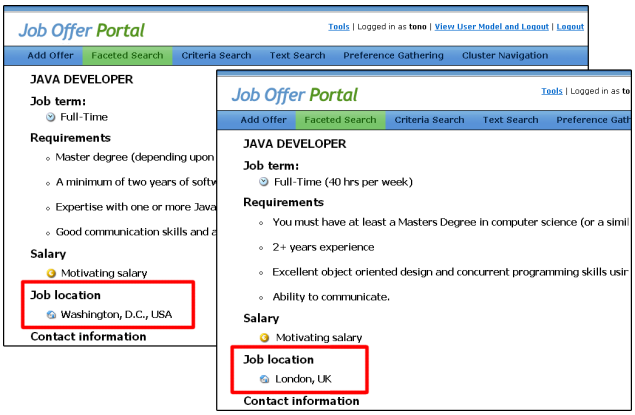

Figure 1. Offers with different job location.

From different ratings given to the offers we can deduce values of particular user characteristics. Higher value assigned to the offer in Washington, D.C. could reveal the job location as important for the user. On the other hand, the same ratings given to different contents could reveal that the job location is unimportant since it had no influence on the interest. This simple example illustrates the importance of content analysis but sophisticated heuristics can be used to discover more information about the user (i.e., *preferences*).

Existing approaches compute the same results for each user and do not take into account user's individuality. In our experiment a user estimated similarity of job offer pairs on a scale from 0 (nothing in common) to 7 (equivalent). Some of pairs appeared twice as a check sample in the set. These pairs were not evaluated identically but with acceptable close values, which shows that the user does not necessarily evaluate the same content the same way if an adequately large scale is provided and specially if there is a time delay between evaluations (i.e., user's interests might

have changed). That supports our assumption that similarity estimation should be personalized.

The paper is structured as follows. In Section 2 we give on overview of the related work. Section 3 presents our method for comparing instances of ontological concepts. In Section 4 we describe an extension to the comparing method to consider also user's individuality. In Section 5 an experimental evaluation of the method is described. At last, in Section 6 we present our concluding remarks.

2 Related Work

The Semantic Web applications typically use ontologies as a base for metadata representation and reasoning. Mainly for the purpose of ontology management several approaches to comparison of ontology concepts or their instances were developed. There are several overviews aimed mostly at ontology matching and related fields [6, 7]. We focus mainly on approaches that primarily consider ontology instances and deal with their similarity.

The approach aimed at the identification of changes in ontology versions on the level of the ontology schema and instances using various heuristics is described in [13]. The approach described in [12] deals with synonyms first to ensure that synonyms refer to the same objects. Then, semantics is incorporated and at last semantic relations (e.g., *is-a* relation) are used to find out whether connected entities are related to the same set of entity classes. The distance between concepts is measured by the shortest path. Similarity evaluation in [8] includes similarity of labels, instances, structures and previous mapping results verified by the application.

A method that compares instances of tourism ontology concepts in two phases is described in [5]. First, two graphs are built, i.e. the *inheritance graph* that organizes ontological concepts according to a generalization hierarchy and the *similarity graph* in which nodes relate to concepts and edges have assigned similarity degree. Similarity estimation consists of structural properties and hierarchical structure. The final similarity measure is computed as a result of combination of previous steps. The main drawback is that a similarity ontology holding similarity relations between properties and instances' names from domain ontology must be provided for the similarity graph.

PROMPT is an algorithm for ontology merging and alignment [10] that provides a user with a guidance. It starts with creating an initial list of matches based on class names where linguistic similarity metrics are employed. The user either selects one of provided suggestions or uses the editing environment to perform changes in the ontology. The next step consists of automatic operations according to the previous choice and is repeated in cycles. When a conflict occurs, a list of solutions is provided. PROMPT performs merging concepts, properties, relations between concepts and properties, and copies parts of a hierarchy (e.g., classes including their parents). We consider name matching as a drawback, i.e. names do not have to carry meaning when automatic approaches are used to build or populate an ontology.

A commonality for mentioned approaches is that they do not take into account a user nor investigate causes of the similarity. The results of the approach described in [3] say that automated similarity estimation mimics to human similarity measure if different strategies are used according to the clusters of users. Here, users gave reasons of their assessments which are basis for machine learning algorithm that assigns users to a cluster. However, personalizing similarity to a group is still not the personalization as commonly understood – adapting to a particular user.

In [4] there is described an application recommending restaurants. Similarity between the restaurants is computed according to the independent properties that reflect user's perception or interest, such as "niceness". A *local similarity metric* is defined for each property what allows determining the similarity of two restaurants with regard to that property. A combination of the metrics creates a *retrieval strategy*. The advantage of this approach is comparing at the property level. However, only defined criteria are considered and it is a *one time use* application where user profiles are not retained.

The comparison with an ideal instance related to a particular domain (here job offers) to recommend similar ones is used in a method for searching based on user's criteria [11]. The method also allows searching for the offers that do not entirely fulfill criteria of the ideal offer. Similarity of particular properties of the offer is computed as a distance between their values. The computed distances are afterwards converted to a degree of the similarity taking into account the biggest possible distance. To distinguish importance of the particular properties for a user, a precision is introduced that reflects the user's subjective tolerance. The user is allowed to specify for each criteria, whether it has to be fulfilled and its importance.

3 Recursive Method of Computing Similarity

We proposed a method that computes similarity between instances of ontological concepts based on recursive evaluation of the properties two instances consist of. The main idea of the method is based on looking for common pairs of properties in both instances and their sequential processing. Detailed description of the method is provided in [1].

In our work we use the OWL Web Ontology Language[1] for the ontology representation and particularly its DL sublanguage. *Datatype* and *object* properties are used to as-

[1] OWL Web Ontology Language, http://www.w3.org/2004/OWL/

sert specific facts about instances. Datatype properties express relations between concept instances and RDF literals and XML Schema datatypes. Object properties express relations between two instances. The rough principle of the method illustrating comparison of two instances $instanceA$ and $instanceB$ is shown in Algorithm 1.

Algorithm 1 Recursive method basics

```
function GETSIMILARITY(instanceA, instanceB)
    set similarity to 0.0
    set counter to 0
    store properties for instanceA and instanceB
      to properties

    foreach property in properties do
        increment counter
        if property is in both instances then
            store connected elements
              to elementX and elementY
            add COMPUTESIMILARITY(elementX, elementY)
              to similarity
        else
            add 0.0 to similarity
        end if
    end foreach

    return similarity/counter
end function

function COMPUTESIMILARITY(elementX, elementY)
    if property is datatype then
        return GETDATASIMILARITY(elementX, elementY)
    else
        set similarity to 0.0
        add GETOBJECTSIMILARITY(elementX, elementY)
          to similarity
        add GETSIMILARITY(elementX, elementY)
          to similarity
        return mean value of similarity
    end if
end function
```

The process of the comparison begins with acquiring all properties from both instances. The following occurrences of a property are possible: single in both instances, multiple in both instances, single/multiple in one instance only.

When the property has a single occurrence in both instances (e.g., the *hasStartDate*), then the similarity of related elements (*instances* if object, or *literals* if datatype properties) is evaluated using different similarity metrics. The comparison of *datatype* properties ends after a metric is used to compute the similarity measure between the related literals. For *object* properties a metric for related instances is computed (e.g., *taxonomy distance*) and further comparison is performed recursively on the respective instances until literals are reached or until there are no properties left.

Multiple occurrences of properties (e.g., *hasPrerequisite* property) in an instance are the most complex case we have to address. In this case, two sets are constructed which contain elements which are connected to the examined property in the first and second instance respectively. These two sets can have different cardinalities – the problem is to identify (i.e., to match) similar elements between these two sets. We use our similarity measure to identify such element pairs, which are then compared and the computed similarity contributes to the total similarity between the two instances. However, the identified pairs do not provide satisfactory results in some cases. For example, if in the first instance the *hasPrerequisite* property has the value *"Java or C programming"* and in the second instance multiple values *"Java programming"* and *"C programming"* consistent results are difficult to achieve. In our approach a pair with higher similarity according to the used similarity metric is selected, but more complex heuristics can be proposed and employed to identify a $1:n$ mapping.

If single or multiple occurrence of a property occurs only in one instance, we estimate the similarity of the elements attached to the property as equal zero. It is based on the similarity definition, i.e. the similarity equals zero if two objects are entirely different. Here, we assume that instances are entirely different in the property, since a value is assigned to the property in one instance only.

Variety of comparison metrics can be employed to compute similarity. We proposed two groups of metrics according to the property's type since they must be treated differently due to their different nature. The description of the metrics is beyond the scope of this paper.

We compute the total similarity of two instances as the mean value of the similarities computed between the elements connected to particular properties. Let us compare two instances $InstA$ and $InstB$. Let $PropertySM$ be a similarity measure (SM) that is computed for elements connected to a common property. The similarity measure for two instances $sim\,(InstA, InstB)$ is computed as follows:

$$\frac{\sum_{i=0}^{|A\cap B|} PropertySM_i\,(elementA, elementB)}{|A \cup B|}, \qquad (1)$$

where $elementA$ and $elementB$ are the elements (instances or literals) connected to the i-th property. Since there can be datatype or object properties, we introduce the *General similarity measure* that encapsulates all the similarity measures that are available. It is computed for the elements connected to a property (e.g., *PropertySM* used in the Equation 1 is its special case). The *General similarity measure* fulfills the same conditions as defined for the similarity and it gets values from the range $\langle 0, 1\rangle$.

The *General Similarity Measure* is computed with regard to the type of property. In the case of datatype property a metric is used according to the type of literal. For object properties the similarity measure for the related instances is computed as the aggregation of the following partial similarity measures:

- *Label-based SM* is computed employing string metrics if labels holding meaningful information are present (if instances were acquired automatically meaningful labels are usually not present),
- *Property-based SM* is computed if instances have additional properties that are used to invoke a recursive computation of the *General similarity measure*,
- *Taxonomy Distance SM* computes a distance between instances in a taxonomy.

4 Personalized similarity and characteristics

The aggregate of the partial similarities is always the same no matter what the context is. To improve the accuracy of our similarity evaluation method with respect to individual users' preferences (if a user model is available), we introduce weights that personalize the similarity estimation which allows us to compute personalized similarity for individual users. Similarity of two instances $sim(InstA, InstB)$ is computed as:

$$\frac{\sum_{i=0}^{|A\cap B|} weight_i \times PropertySM_i\,(elementA, elementB)}{\sum weight_i}, \quad (2)$$

where semantics of variables is the same as in Equation 1. The $weight$ variable has values in the range $\langle 1, w\rangle$ based on the match between the property and the value of the corresponding characteristic in the user model.

Since we assume that the user's likes should have greater influence on the total similarity, we increase the weights of properties for which corresponding characteristics are present in the respective user model and their values match with the compared instances. The exact increase of individual weights is the subject of experiments for any particular domain. The meaning of proposed weights reflects possible cases that may occur:

- *1* – if there is no correlation between a property of the instance and a characteristic in the user model; this weight solves also the problem when the user model is not present and it does not have influence on computing personalized similarity with regard to the user.
- *w* – if there is mach not only between a property of the instance and a characteristic but also there is mach between their values.
- *a value* between the previous two values means that there is a match between the examined property of the instance and the user model, but the related value is not identical, e.g. a city belongs to the same region as the city preferred by the user but it is not that city.

We use an ontology-based overlay user model[2] that was built by *LogAnalyzer* [2]. It contains *characteristics* and *preferences*. A preference indicates that the related property is important for the user but there is no specific value assigned to it. For the preference we consider $weight$ equal half of the the upper bound employed in the computation of the personalized similarity ($weight = w/2$). In the case, when a property occurs in the user model as a characteristic and also as a preference, the $weight$ is computed as their sum, i.e. $weight = w + w/2$.

From the user's evaluation given to content we can deduce user's likes or dislikes (investigate causes). We assume that if the instance includes a property which value the user likes it will likely influence his or her rating towards higher values. On the other hand, the properties with the values that the user dislikes will influence the rating towards lower values. Therefore, we introduce two threshold values that divide properties into three sets according to their similarity values. Since we are interested in the properties that significantly influence user's evaluation and this way also the total similarity, we give up splitting intervals into equal parts. If the similarity computed for a property is greater than the *positive threshold* than the property is assigned to the positive set, if the computed similarity is lower than the *negative threshold* the property is assigned to the negative set.

Properties classified by this method can be further transformed into user characteristics and used for populating or updating existing characteristics in the user model.

5 Experimental evaluation

The evaluation was carried on the job offer ontology developed in the course of the project NAZOU[3] [9]. The smallest dataset contains 100 job offers mostly from the *IT* field. To evaluate our method we implemented a software tool *Concept Comparer* (*ConCom*). It works in two modes, i.e. the total similarity is computed for all properties and if a property occurs in one instance only, then 0 is aggregated. In the second mode, only common properties are considered and the other ones are omitted with no influence on the total similarity.

The aim of the first experiment was to compare results computed in two ways and to specify thresholds. Results for a randomly selected sample of 600 pairs is depicted in the Figure 2. The thresholds were specified experimentally for the job offer domain. We evaluated similarity for 55 000 properties. The properties with similarity equal to 0.0 or 1.0 were not considered to eliminate identities and proper-

[2] Ontology-based User Model, http://nazou.fiit.stuba.sk/home/files/nazou_um.pdf

[3] NAZOU – Tools for acquisition, organization and maintenance of knowledge in an environment of heterogeneous information resources, http://nazou.fiit.stuba.sk

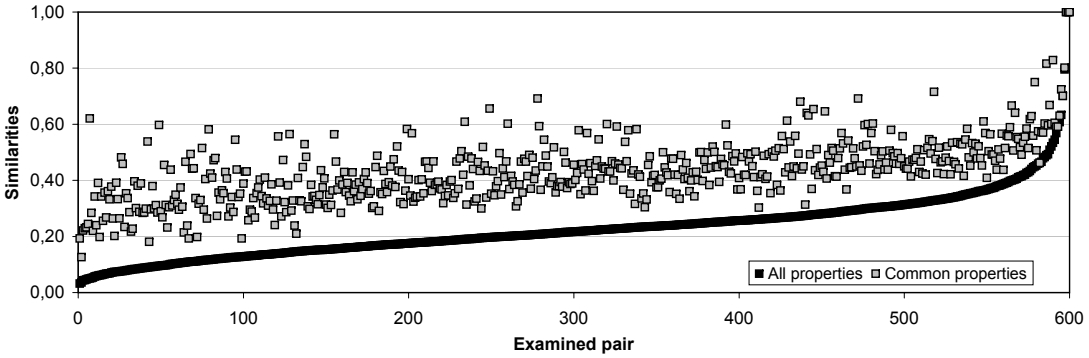

Figure 2. Similarity computed by ConCom considering all/common properties.

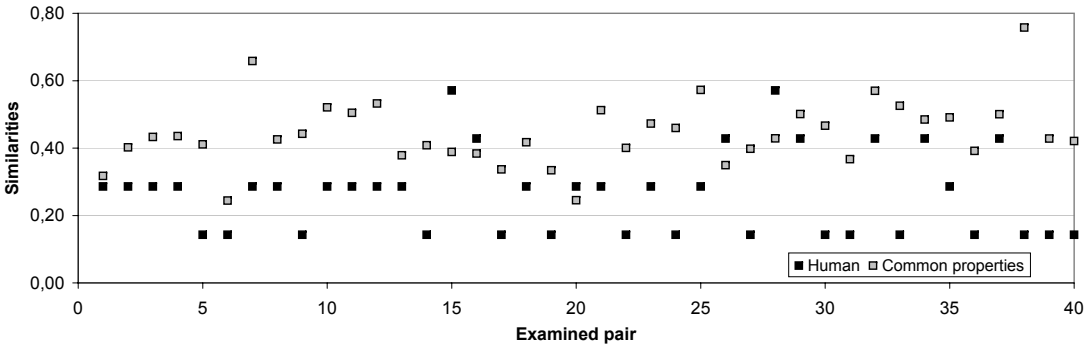

Figure 3. Similarity estimated by a human and by ConCom for common properties.

ties with no occurrence in both instances. The rest of the properties was ordered according to the similarity measure and *Pareto principle* was applied. We split the most influencing 20% in half to select 10% of the highest and 10% of the lowest values. The positive threshold was set to 0.65 and the negative threshold to 0.25. Domain dependence is the subject of the further experiments. Using only common properties resulted in narrow range of similarity values (89% of the computed similarities went from the range 0.30 to 0.75) what exceeds proposed negative thresholds and narrows the range for the positive set. Therefore, the similarity computed for all properties must be used to acquire properties based on thresholds.

Next, a user was involved. A sample of 300 job offer pairs was used where 30 randomly selected pairs appeared twice as a check sample. A user assessed the similarity value on a scale from 0 to 7 and the acquired values were normalized to the similarity interval. The result for a set of 40 pairs is depicted in the Figure 3. We used the similarity computed for common properties to compare with human's evaluation since its values mimic better values from evaluation given by a human. It might have been caused by the fact that a user can more easily evaluate lower amount of (common) properties. We have not found significant differences in control sample evaluation, i.e. in 70% of the cases were the pairs assigned identical values and only in 3.33% of cases was difference three points on the scale. Obviously another user would evaluate the same sample differently.

For further experiments involving the user model we used the similarity computed only for common properties. The used user model consisted of one characteristic only. The job offers that were used in the experiment contained *hasDutyLocation* property and its value was the same as in the user model. The experiment was aimed at figuring the upper weight bound to be used in the personalized similarity computation. A change of the similarity depends on the number of properties that are considered in the comparison. The job offers used in the experiment had an average of 16 properties. The personalized similarities computed for 12 job offers pairs with $w = 2.0$ resulted in the similarity values increased about 0.06 to 0.9 (and about 0.15 to 0.26 if $w = 4.0$) what is significant difference. Therefore, using doubled weights in the personalized similarity is reasonable.

The last experiment was aimed at investigating how user model impacts the similarity enumeration. A user model with three characteristics was employed and doubled weights used. Changes of the similarity caused by employing the user model for 200 pairs are shown in the Figure 4. The user model influences the computed similarity in two

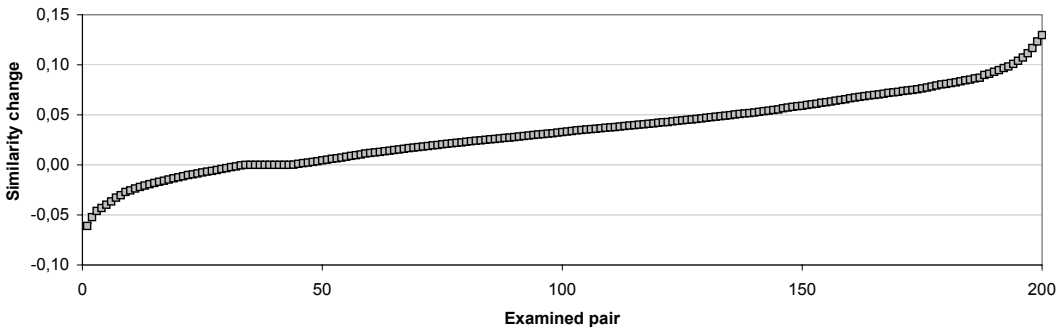

Figure 4. Change in similarity estimation caused by the employed user model.

ways. If compared properties are similar (high values of similarity measure) the personalized similarity increases towards higher values (a positive change in the figure). On the other hand, the similarity decreases to lower values.

6 Conclusions

We present a method for comparing instances of ontological concepts based on the recursive traversing of instance's structure. The final similarity is a result of mean aggregation of the similarities computed for particular properties. The introduction of similarity computed for properties allows us to take advantage of semantics provided by ontological representation and allowed us to extend similarity with personalized weights reflecting users' individuality.

We developed a software tool *ConCom* that realizes the proposed method. The experiments showed that the similarity where only common properties are considered is more suitable if the user model is involved and considering all properties is better to investigate properties that influenced user's evaluation (e.g., interest). We introduce two thresholds that were set experimentally for the job offers application domain – the positive to 0.65 and the negative to 0.25.

The evaluated similarity can be also useful as a support for clustering algorithms, semantic annotation tools or repository maintenance tools. The aim here is to improve semantic search using the method for personalized navigation within ontology instances that represent metadata of large information space. We further plan to focus on relationships between the weights and the number of properties that are present in the compared instances and to investigate domain independence of the proposed method and carry on experiments in the domain of scientific publications.

This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104, the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04 and the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

[1] A. Andrejko and M. Bieliková. Investigating similarity of ontology instances and its causes. In *ICANN 2008*, LNCS 5164, pages 1–10. Springer, 2008.

[2] M. Barla and M. Bieliková. Estimation of user characteristics using rule-based analysis of user logs. In *Proc. of Workshop held at UM2007*, pages 5–14, 2007.

[3] A. Bernstein, E. Kaufmann, C. Burki, and M. Klein. How similar is it? towards personalized similarity measures in ontologies. In *7th Int. Conf. Wirtschaftsinformatik (WI-2005)*, pages 1347–1366, Bamberg, Germany, 2005.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. *UMUAI*, 12(4):331–370, 2002.

[5] A. Formica and M. Missikoff. Concept similarity in symontos: An enterprise management tool. *The Computer Journal*, 45(6):583–595, 2002.

[6] F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics IX*, 4601:1–38, 2007.

[7] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. *The Knowl. Eng. Review*, 18(1):1–31, 2003.

[8] X. Liu, Y. Wang, and J. Wang. Towards a semi-automatic ontology mapping. In *Proceedings of Fifth Mexican International Conference on AI (MICAI'06)*. IEEE, 2006.

[9] P. Návrat, M. Bieliková, and V. Rozinajová. Acquiring, organising and presenting information and knowledge from the web. In *CompSysTech'06*, 2006.

[10] N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proc. of the 17th National Conf. on AI and 12th Conf. on Innovative Applications of AI*, pages 450–455. MIT Press/AAAI Press, 2000.

[11] R. Pázman. Ontology search with user preferences. In *Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, pages 139–147, 2006.

[12] M. A. Rodríguez and M. J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. on Knowl. & Data Eng.*, 15(2):442–456, 2003.

[13] M. Tury and M. Bieliková. An approach to detection ontology changes. In *ICWE '06: Workshop Proc. of 6th Int. Conf. on Web Eng.*, Palo Alto, California, 2006. ACM Press.

[14] M. Tvarožek and M. Bieliková. Personalized Faceted Navigation in the Semantic Web. In *ICWE 2007*, LNCS 4607, pages 511–515. Springer, 2007.