Using Machine Learning to Support Continuous Ontology Development

Maryam Ramezani[1], Hans Friedrich Witschel[1], Simone Braun[2], Valentin Zacharias[2]

[1]SAP Research and [2]FZI Forschungszentrum Informatik, Karlsruhe, Germany

Abstract. This paper presents novel algorithms to support the continuous development of ontologies; i.e. the development of ontologies during their use in social semantic bookmarking, semantic wiki or other social semantic applications. Our goal is to assist users in placing a newly added concept in a concept hierarchy. The proposed algorithm is evaluated using a data set from Wikipedia and provides good quality recommendation. These results point to novel possibilities to apply machine learning technologies to support social semantic applications.

1 Introduction

There are two broad schools of thought on how ontologies are created: the first views ontology development akin to software development as a - by and large - one off effort that happens separate from and before ontology usage. The second view is that ontologies are created and used at the same time, i.e. that they are continuously developed throughout their use. The second view is exemplified by the Ontology Maturing model [1, 2] and by the ontologies that are developed in the course of the usage of a semantic wiki.

Machine learning, data mining and text mining methods to support ontology development have so far focused on the first schools of thought, namely on creating an initial ontology from large sets of text or data that is refined in a manual process before it is then used. In our work, however, we focus on using machine learning techniques to support continuous ontology development, in particular we focus on one important decision: given the current state of the ontology, the concepts already present and the sub/super concept relations between them - where should a given new concept be placed? Which concept should become the super concept(s) of the new concept?

We investigate this question on the basis of applications that use ontologies to aid in the structuring and retrieval of information resources (as opposed to for example the use of ontologies in an expert system). These applications associate concepts of the ontology with information resources, e.g. a concept “Computer Science Scholar” is associated to a text about Alan Turing. Such systems can use the background knowledge about the concept to include the Alan Turing text in responses to queries like “important British scholars”. Important examples for such systems are:

– The **Floyd** case management system developed at SAP. In that system, cases and other objects (that are attached to cases, such as documents) can be tagged freely with terms chosen by the user. These terms can also be organized in a semantic network and this can be developed by the users. The Floyd system is usually deployed with a semantic network initially taken from existing company vocabulary.

- The **SOBOLEO** system [3] uses a taxonomy developed by the users for the collaborative organization of a repository of interesting web pages. There is also a number of similar social semantic bookmarking applications [4].
- The **(Semantic) Media Wiki** [5] system uses a hierarchy of categories to tag pages. We can view categories as akin to concepts and support the creation of new categories by proposing candidate super-categories.

All these system are "Web 2.0" style semantic applications; they enable users to change and develop the ontology during their use of the system. The work presented in this paper assists users in this task by utilizing machine learning algorithms. The algorithms suggest potential super-concepts for any new concept introduced to the system.

The rest of this paper is organized as follows. In section 2 we will present the previous research in this area and discuss how our work differs. In section 3 we describe the proposed algorithm for the recommendation of superconcepts. In section 4 we describe the methodology, the dataset and the results from the evaluation before section 5 concludes the paper.

2 Related Work

Many researchers have proposed the idea of creating ontologies from social tagging applications; from the terms users have assigned to information resources. [6] was one of the first who proposed social tagging systems as a semantic social network which could lead to the emergence of an ontology. An idea that is based on the emergent semantics proposed by [7] and the vision of a community of self-organizing, autonomous agents co-operating in dynamic, open environments, each organizing knowledge (e.g. document instances) and establishing connections according to a self-established ontology.

Van Damme et al. propose a 6-step methodology for deriving ontologies from folksonomies by integrating multiple techniques and resources [8]. These techniques comprise Levenshtein metric to identify similar tags, co-occurence and conditional probability to find broader-narrower relations and transitive reduction and visualization to involve the community. Future work shall include other existing resources like Google, WordNet, Wikipedia, ontologies for mapping. Likewise, [9] try to automatically enrich folksonomies using existing resources. They propose two strategies, one based on WordNet, the other using online ontologies, in order to map meaning and structure information to tags. Monachesi and Markus [10] developed an "ontology enrichment pipeline" to enrich domain ontologies with social tagging data. They evaluated different similarity measures to identify tags related to existing ontology concepts. These are symmetric (based on Jaccard) and asymmetric co-occurence and cosine similarity both of resource and user. They excluded tf and tfidf measures because they could not find any additional benefit in their test. Finally, they use DBpedia in combination with a disambigation algorithm based on Wikipedia in order to place the identified tags into the ontology. [11] suggests mapping tags to an ontology and presents the process of mapping in a simple example.

Other researchers have started solving the details of the problem using information retrieval techniques. [12] suggest creating a hierarchical taxonomy of tags by calculating the cosine similarity between tags, i.e. each new tag added to the system will be categorized as the child of the most similar tag. If the similarity value is less than a pre-defined threshold then the new tag will be added as a new category, which is a new child for the root. The problem with this algorithm is that there is no heuristic to find the parent-child relation. Any new similar tag will be considered as a child of the most similar tag previously added to the system even though it might be more general than the other tag. Markines et al. [13] present different aggregation methods in folksonomies and similarity measures for evaluating tag-tag and resource-resource similarity. Marinho et al. [14] use frequent itemset mining for learning ontologies from folksonomies. In this work, a folksonomy is enriched with a domain expert ontology and the output is a taxonomy which is used for resource recommendation.

The approach taken in this paper is different from the ones mentioned above in the sense that we suggest a recommendation approach to support end users in the collaborative maturing of ontologies [1, 2], i.e. where anybody can add a new element to the ontology, and refine or modify existing ones in a work-integrated way. That means the ontology is continuously evolving and gradually built up from social tagging activities and not derived once at a specific time from the folksonomy. Our work provides a supporting tool for such ontology building by helping users with recommendation of semantic relationships, specifically super-subconcept relationships, between a new concept and the existing concepts.

3 Algorithm for Recommending Super-Concepts for New Concepts

We propose an algorithm for the recommendation of super-concepts for a new concept. This algorithm uses an existing concept hierarchy and assists the user in finding the right place for a new concept.

3.1 Degree of Sub-Super Relationship in a Concept Hierarchy

First we define a measure for the distance between a super concept and its sub concepts. We consider the shortest path distance between two concepts, starting from the sub concept and allowing only upward edges to be used to arrive at the super-concept. We call this“super-sub affinity” (“SSA”). To clarify how we find SSA, consider a concept hierarchy with a root A and two sub concepts B and C. Then SSA(A,B)=1, SSA(A,C)=1 and SSA(B,C)=0. If B has a sub-concept D, then SSA(A,D)=1/2. Note that SSA is not a symmetric relation, distinguishing it from common semantic similarity measures. In fact, the definition of SSA entails “if SSA(A,B)$\neq$0 then SSA(B,A)=0”. We define SSA(A,A)=1. For more details about *SSA*, please refer to [18]. We store all SSA values in an $n \times m$ matrix where n is the number of concepts which have at least one sub-concept, m is the total number of concepts in the hierarchy, and the matrix diagonal is always 1. We will use this matrix in our recommendation algorithm for discovering super-concepts. The transpose of this matrix can be used for suggesting sub-concepts using the same algorithm. However, in this work, we focus only on recommending super-concepts.

3.2 Concept Similarity

In this section we define measures used to compare the similarity of the new concept to the existing concepts. We consider measures that use similarities in the concept names as well as measures that use contextual cues, i.e. secondary information available about the use of the concept.

For **string-based similarity** we use standard Jaccard similarity to find the degree of similarity among concepts with compound labels. Jaccard similarity is defined as: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. where A and B are (multi-word) concepts. Using the Jaccard measure, the string-based similarity between each concept C_i and the new target concept C_t is defined as $sim_s(C_t, C_i) = J(C_t, C_i)$. For example the Jaccard similarity between two concepts "Computer" and "Computer Science" would be 1/2. Using this similarity measure, we find the set of k most similar concepts to the target concept C_t and we call this set N_s.

Context-based cues aim at using the context that the new concept has been used in to find similar concepts. Context has been defined by Dey [15] as any information that can be used to characterize the situation of an entity. In a social tagging system, for example, the context of a new tag entered into the system can be distinguished by the related resources, links between the resources, users who enter the tag, time, language and geographical information. In this work, we use the resources associated to a concept as a feature set to determine the context of the concept. We represent each concept C as a vector over the set of resources, where each weight, $w(r_i)$, in each dimension corresponds to the importance of a particular resource, r_i.

$$C = \langle w(r_1), w(r_2) ... w(r_{|R|}) \rangle \quad (1)$$

In calculating the vector weights, a variety of measures can be used. The weights may be binary, merely showing that one or more users have associated that concept to the resource, or it may be finer grained using the number of users that have associated that concept to the resource. With either weighting approach, a similarity measure between two vectors can be calculated by using several techniques such as the Jaccard similarity coefficient or Cosine similarity [16]. Cosine similarity is a popular measure defined as

$$Cosine(C1,C2) = \frac{C1.C2}{||C1|| \, ||C2||} \quad (2)$$

In this work, we use binary weighting for representing concepts as a vector of pages and Cosine similarity to find similar concepts. Thus, the similarity between each concept C_i and the new target concept C_t is defined as $sim_c(C_t, C_i) = Cosine(C_t, C_i)$. Using this similarity measure, we find the set of k most similar concepts to the target concept C_t and we call this set N_c.

We define a **hybrid similarity measure** by combining the string-based and contextual-based similarity measures. For that purpose, we use a linear combination of the similarity values found in each approach.

$$Sim_h(C_i, C_t) = \alpha Sim_s(C_i, C_t) + (1 - \alpha) Sim_c(C_i, C_t) \quad (3)$$

where $Sim_h(C_i, C_t)$ is the hybrid similarity value, and α is a combination parameter specifying the weight of string-based approach in the combined measure. If $\alpha = 1$, then

$Sim_h(C_i,C_t) = Sim_s(C_i,C_t)$, in other words the neighbors are calculated only based on string-similarity. On the other hand, if $\alpha = 0$, then only the contextual information is used for finding similar concepts. We choose the proper value of alpha by performing sensitivity analysis in our experimental section.

3.3 Prediction Computation

Based on the Super-Sub Affinity and the similarity measures defined above, we can now predict the degree of sub-super relationship (SSA) between the new concept and every other concept in the hierarchy. Our proposed algorithm is inspired by the popular weighted sum approach for item-based collaborative filtering [17].

Formally, we predict the SSA between the target concept and all other concepts C_i in the hierarchy as follows.

$$SSA_p(C_i,C_t) = \frac{\sum_{C_n \in N} SSA(C_i,C_n) * sim(C_t,C_n)}{\sum_{C_n \in N} sim(C_t,C_n)} \tag{4}$$

where $SSA_p(C_i,C_t)$ stands for the predicted SSA value for the pair (C_i,C_t), $SSA(C_i,C_n)$ stands for the actual SSA for (C_i,C_n), and $sim(C_t,C_n)$ is the similarity value between the target concept and neighbor concept which can be either string-based (Sim_s), contextual (Sim_c) or the hybrid (Sim_h) similarity. Thus N can be either N_s,N_c or N_h as described in section 3.2. Basically, SSA_p is predicted based on the location of the existing concepts that are similar to C_t; it becomes large when many of C_t's neighbors are close to the current candidate concept C_i in terms of SSA. Hence, the best candidates for becoming a super-concept of C_t are those C_i for which $SSA_p(C_i,C_t)$ is maximal. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range. In this work we have defined the direct sub-super affinity as 1. Thus, the nearer the prediction of $SSA(C_i,C_t)$ to 1, the more probable that C_i is super-concept of C_t.

3.4 Recommendation

Once the SSA values for all existing concepts and the new concept are calculated, the concept(s) with the highest SSA can be recommended as super-concept for the new concept. We can recommend a list of top *n* concepts with highest SSA prediction or we can use a threshold value and only recommend concepts with predicted SSA higher than the threshold. The threshold value (between 0 and 1) represents the “confidence” of the algorithm in recommendations. If there are no similar concepts found in step 1 or the predicted SSA values are lower than the threshold, the system does not make a recommendation which might mean that the new concept should be added as a new independent concept at the top of the hierarchy or that the system is not able to find the right place for the new concept.

4 Evaluation and Results

4.1 Data Set

To test our algorithms we need a Web 2.0 application where users can easily add new concepts and create semantic relations. We decided to use Wikipedia which is the most suitable web 2.0 application at hand. Hepp [19] theoretically proves Wikipedia as a reliable and large living ontology. We treat the categories of Wikipedia as concepts and the existing relationships between "Subcategories" as the seed concept hierarchy. Each category in Wikipedia has several associated pages, which we use as a context vector for the category as described in section 3.2. Thus, each category is represented as a binary vector over the set of pages. The weight of each page r_i for category C_j is 1 if page r_i is associated to category C_j and 0 otherwise.

For running our experiments, we focused on a small part of the English Wikipedia. We started from the category "Computer Science" as the root concept and extracted the sub-categories by traversing with breadth first search through the category hierarchy. Our final data set has over 80,000 categories. However, for our experiments we created three smaller data sets to compare how the size and properties of the seed concept hierarchy impact the results. Our smallest data set has 3016 categories with 47,523 associated pages. The medium data set has 9931 with 107,41 associated pages and the large data set has 24,024 categories with 209,076 pages. The average depth of the hierarchy is 3, 6 and 9 for those three data sets respectively.

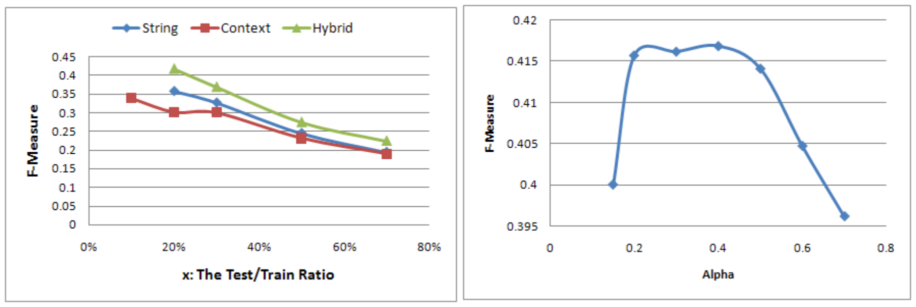

Fig. 1. Comparison of F-measure for different approaches by changing the test/train ratio x(on the left) and sensitivity of α in the hybrid algorithm(on the right)

4.2 Evaluation Methodology and Metrics

We divided the data set into a training set and a test set. Since we were interested to know how the density of the seed ontology affects the results, we introduced a variable that determines what percentage of data is used as training and test sets; we call this variable *x*. A value of x = 20% would indicate 80% of the data was used as training set and 20% of the data was used as test set. We remove all information of test cases from the data set to evaluate the performance of the algorithm. For each experiment, we calculate the SSA values before and after removing the test cases. If the test case has sub-concept and super-concept, after removing the test case, its sub-concepts will be directly connected to its super-concepts and the algorithm has to intelligently discover its original place in between the two concepts. For evaluation we adopt the common

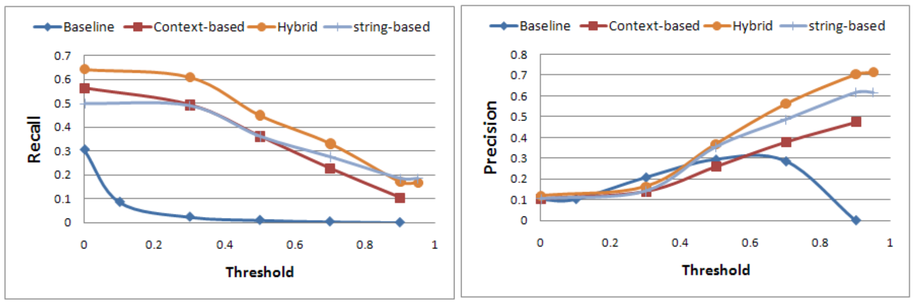

Fig. 2. Comparison of Precision and Recall of different algorithms for different threshold values

recall and precision measures from information retrieval. Recall measures the percentage of items in the holdout set that appear in the recommendation set and is defined as: $recall = C_h \cap C_r|/|C_h|$ where C_h is the set of holdout concepts and C_r is the set of recommended concepts. Precision measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as: $precision = |C_h \cap C_r|/|C_r|$. In order to compare the performance of the algorithms, we also use F-measure defines as

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

4.3 Experimental Results

In this section we present our experimental results of applying the proposed recommendation algorithm to the task of ontology maturing. In all experiments, we change the threshold value from 0 to 0.95 and record the values of precision and recall. As the threshold value increases, we expect precision to increase and recall to decrease since we recommend less items with higher confidence. In addition, to get a better view of performance of the algorithms, we use the notion of recall at N. The idea is to establish a window of size N at the top of the recommendation list and find recall for different numbers of recommendations. As the number of recommendation increases, we expect to get higher recall. In assessing the quality of recommendations, we first determined the sensitivity of some parameters. These parameters include the neighborhood size k, the value of the training/test ratio x, and the combination factor α. Our results show (not shown here) that there is a trade-off between better precision or better recall depending on the neighborhood size. We select $k = 3$ as our neighborhood size which gives better

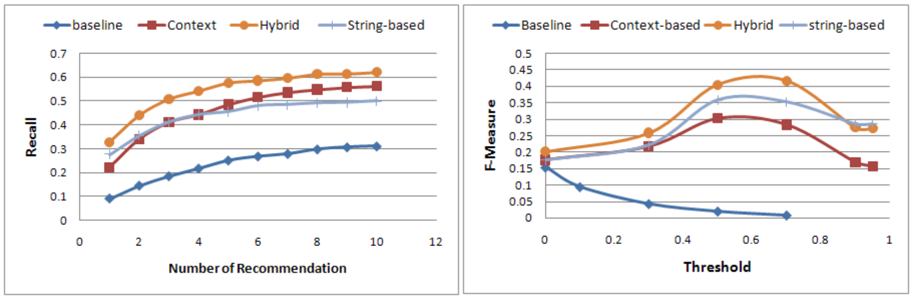

Fig. 3. Comparison of F-measure for different approaches by changing the threshold value(on the right) and Comparison of recall at N for different algorithms

precision for high threshold values. To determine the sensitivity of the value of α in the hybrid algorithm, we conducted experiments with different values of alpha. The result of these experiments is shown in the right chart of figure 1 . From this chart we select the value of $\alpha = .4$ for the hybrid algorithm. To determine the effect of density of the seed concept hierarchy, we carried out an experiment where we varied the value of x from 20% to 70%. Our results are shown in the left chart of figure 1. As expected, the quality of recommendation decreases as we increase x. However, even with x=70%, our algorithm can still produce acceptable recommendations. For further experiments we keep x=20%.

Once we obtained the optimal values of the parameters, we compared the performance of the algorithm when using different similarity computation techniques. In addition,we compared our approach with a baseline algorithm suggested in [12]. The results of this comparison are shown in figure 2 and 3. The baseline algorithm uses the cosine similarity between concepts and recommends the concepts with highest cosine similarity. Basically, the baseline algorithm is similar to the first step of our algorithm where we find the k-nearest neighbors based on contextual cues. Figure 2 shows the precision and recall values as we change the threshold value and figure 3 shows the comparison using F-measure and Recall at N.

4.4 Discussion

Our results show that the hybrid similarity outperforms other approaches for both precision and recall for all thresholds and x values. We can observe from figure 2 that while contextual cues result in less precision than the string-based techniques, they produce slightly higher recall. That shows that string-based techniques can suggest more accurate recommendations but they do not have as much coverage as the context-based ones. Figure 3 compares the same algorithms using different measures. The right chart shows the F-measure for each algorithm as the threshold value changes and the left chart shows recall at N. Basically, we count the correct answers as we recommend N super-concepts. We can observe from these two charts that while hybrid similarity obviously outperforms other similarity measures, it is not trivial to determine if string-based measures are better than the context-based ones. The context-based cues outperform string-based ones when looking at the Recall at N while based on F-measure the string-based techniques outperform context-based ones.

In terms of continuous ontology development, picking a threshold of .7 from the right chart of figure 3 for the hybrid algorithm, this means that users who introduce a new concept into an existing ontology can count on almost 60% (see right of figure 2) of the recommendations received being correct and on a coverage of around 35% when looking through all recommendations and selecting the right ones.

4.5 Qualitative Evaluation

Although the precision and recall values from the experiments above are quite acceptable, we decided to investigate the actual results and find out in what cases the algorithm makes incorrect recommendations. We selected a random new concept "Botnets" and we observed the recommendation outputs. Table 1 shows the top 5 recommendations based on each technique. We can observe in this example that although not equal to the actual Wikipedia super-categories, the recommendations do make some sense.

Table 1. An example: Output of recommendation algorithm using different similarity cues. Recommendations are predicted super concepts of the new concept "Botnets"

Wikipedia	**Contextual Cues**	**String-based Cues**	**Hybrid**
Multi-agent systems	Artificial intelligence	Multi-agent systems	Multi-agent systems
Computer network security	Multi-agent systems	Computer network security	Artificial intelligence
	Computer architecture	Computer security organizations	Computer architecture
	Network architecture	Artificial intelligence	Computer network security
	Distributed computing	Computer architecture	Distributed computing

5 Conclusion and Future Work

We utilized recommender system technologies to support collaborative ontology maturing in a Web 2.0 application. We introduced a hybrid similarity measure by combining contextual and string-based cues to find the super concepts of a new concept. Our evaluation with the Wikipedia category hierarchy shows promising results. From the qualitative results we can see that our recommender in fact can produce better results than the calculated precision and recall indicate. Thus, the system can also be used directly in Wikipedia for improving the current category hierarchy.

In this work, we have used associated pages to a concept as contextual information. As future work, other contextual information such as links between pages, or user information can be considered as well. In addition, evaluation of the algorithms in an actual interactive social semantic bookmarking application can help us answer the question whether the generated recommendations are actually perceived as useful by the user.

Acknowledgments

This work was supported by the MATURE Project co-funded by the European Commission.

References

1. Braun, S., Kunzmann, C., Schmidt, A.: People Tagging & Ontology Maturing: Towards Collaborative Competence Management. In: From CSCW to Web2.0: European Developments in Collaborative Design. CSCW series. Springer, London (2010) 133–154
2. Braun, S., Schmidt, A., Walter, A., Nagypal, G., Zacharias, V.: Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering. In: Proc. of the WWW'07 Workshop on CKC, CEUR-WS vol. 273 (2007)
3. Zacharias, V., Braun, S.: SOBOLEO - Social Bookmarking and Lightweight Ontology Engineering. In: Proc. of the WWW'07 Workshop on CKC, CEUR-WS vol. 273 (2007)
4. Braun, S., Schora, C., Zacharias, V.: Semantics to the Bookmarks: A Review of Social Semantic Bookmarking Systems. In: Proc. of the 5th I-SEMANTICS. (2009) 445–454
5. Krótzsch, M., Vrandecic, D., Vólkel, M., Haller, H., Studer, R.: Semantic Wikipedia. Journal of Web Semantics **5** (2007) 251–261
6. Mika, P.: Ontologies are us: A unified model of social networks and semantics. In: In International Semantic Web Conference. (2005) 522–536
7. Philippe, K.A., Ouksel, A.M.: Emergent Semantics Principles and Issues. In: In Proc. of the 9th Int. Conf. on Database Systems for Advanced Applications, Springer (2004) 25–38
8. Damme, C.V., Coenen, T., Vandijck, E.: Deriving a Lightweight Corporate Ontology form a Folksonomy: a Methodology and its Possible Applications. Scalable Computing: Practice and Experience - Int. J. for Parallel and Distributed Computing **9**(4) (2008) 293–301
9. Angeletou, S., Sabou, M., Motta, E.: Improving Folksonomies Using Formal Knowledge: A Case Study on Search. In: The Semantic Web - ASWC'09. Volume 5926 of LNCS., Springer (2009) 276–290
10. Monachesi, P., Markus, T.: Using Social Media for Ontology Enrichment. In: Proc. of 7th ESWC, Berlin Heidelberg, Springer (2010) 166–180
11. Zhao, N., Fang, F., Fan, L.: An Ontology-Based Model for Tags Mapping and Management. In: Proc. of the Int. Conf. on CSSE, IEEE Computer Society (2008) 483–486
12. Heymann, P., Garcia-Molina, H.: Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical Report 2006-10, Stanford University (2006)
13. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating Similarity Measures for Emergent Semantics of Social Tagging. In: Proc. of the 18th Int. Conf. on WWW, ACM (2009) 641–650
14. Balby Marinho, L., Buza, K., Schmidt-Thieme, L.: Folksonomy-Based Collabulary Learning. In: Proc. of the 7th Int. Conf. on The Semantic Web, Springer (2008) 261–276
15. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing **5**(1) (2001) 4–7
16. Van Rijsbergen, C.: Information Retrieval. Butterworth-Heinemann Newton, USA (1979)
17. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th Int. Conf. on WWW, ACM (2001) 285–295
18. Ramezani, M., Witschel, H.F.: An intelligent system for semi-automatic evolution of ontologies. In: Proceedings of 5th IEEE International Conference on Intelligent Systems IS10. (2010)
19. Siorpaes, K., Bachlechner, D.: Harvesting wiki consensus - using wikipedia entries as ontology elements. In: IEEE Internet Computing. (2006) 54–65