An Efficient Ontology-Based Expert Peering System

Tansu Alpcan, Christian Bauckhage, and Sachin Agarwal

Deutsche Telekom Laboratories
10587 Berlin, Germany
(tansu.alpcan, christian.bauckhage, sachin.agarwal)@telekom.de

Abstract. This paper proposes an expert peering system for information exchange. Our objective is to develop a real-time search engine for an online community where users can ask experts, who are simply other participating users knowledgeable in that area, for help on various topics. We consider a graph-based representation scheme consisting of an ontology tree where each of node corresponds to a (sub)topic. Consequently, the fields of expertise (profile) of the participating experts are represented as subtrees of this ontology. Since user queries can also be mapped to similar tree structures, assigning queries to relevant experts becomes a problem graph matching. A serialization of the ontology tree allows us to use simple dot products on the ontology vector space as an efficient way to address this problem. We conduct extensive experiments with different parameterizations and observe that our approach is efficient and yields promising results.

1 Motivation and Background

Document retrieval studies the problem of matching user queries against a given set of typically unstructured text records such as webpages or documents. Since user queries, too, may be unstructured and can range from a few keywords to multi-sentenced descriptions of the desired information, stop word removal, stemming, and keyword spotting usually precede the actual retrieval step. Given correspondingly purged dictionaries, most systems for document retrieval and text classification rely on the *vector space model* of documents. It represents documents and queries by term-by-document vectors and allows for approaches based on statistical learning. Recent research in this area includes the use of support vector machines [1], probabilistic semantic indexing [2], or spectral clustering [3].

However, despite their dominant role, methods relying on term-by-document vectors suffer from several drawbacks. For instance, they cannot capture relations among terms in a single document and have to assume a static dictionary in order to fix the dimension of the vectors. Graph-based models, in contrast, easily cope with shortcomings like these and thus provide a promising alternative approach to document retrieval. In an early contribution, for example, Miller [4] considers bipartite matchings between documents and queries which are given in terms of co-occurrence graphs. In more recent work, Schenker et al. [5, 6] propose a graph structure for documents and queries that accounts for sequences of words. Matches are computed based on a k-nearest neighbors

(kNN) criterion for graphs and it shows that this outperforms common vector-based kNN retrieval.

In this paper, we assume a different view on graph-based document retrieval. Focusing on development of a peer-to-peer (P2P) communication mechanism for an online community, we describe a retrieval system that exploits semantic structures for text-based classification. In the community we are concerned with, users can sign on and identify themselves as experts for certain domains and fields of knowledge. After signing on, users may either address the community with problems they need help on, or –if they are qualified– can respond to other users' questions. Rather than mediating the communication between community members by an forum-style mechanism, we aim at a solution that, given a user query, automatically proposes appropriate experts who then can be contacted directly via, for example, instant messaging.

Our approach relies on a comprehensive ontology tree describing relevant fields of knowledge where each node corresponds to a single subject or topic described by a bag of words. Similarly, we associate each query and expert with a bag of words of flexible size and define a similarity measure to compare two bags. Utilizing an algorithm which will be described in detail in Section 2, this formulation enables us to represent entities such as queries or experts as subtrees of the ontology at hand. Furthermore, serialization of the ontology tree allows for defining an *ontology-space*. Therefore, queries as well as experts can equivalently be represented as vectors in this linear vector space. The problem of peering queries and experts therefore becomes a problem of tree matching which we in turn address using dot product operations between respective vectors.

Ontology-based document retrieval and search has become an active area of research. Especially ontology building from a set of documents and term similarity measures have found increased attention [7, 8]. In recent work more closely related to our scenario, Wu et al. [9] study an expert matching problem similar to ours. However, their approach differs from our solution for they apply ontologies to compute path-length-based distances between concepts upon which they base several definitions of similarity measures for documents. The approach presented in this paper, in contrast, exploits hierarchal coarse-to-fine information contained in the ontology and measures document similarities in semantics induced vector spaces. Moreover, while the algorithms in [9] require manual intervention, our scheme is fully automatic. Finally, preliminary experiments we conduct demonstrate that our approach leads to a higher performance in terms of precision and recall than the one in [9].

The rest of the paper is structured as follows: in the next section we describe our approach and algorithms developed in detail. Section 3 presents a demonstrative experimental study and discusses its results. The paper concludes with a summary and remarks on future research directions in Section 4.

2 Model and Approach

We present an ontology-based semantic model and approach to address the query-expert peering problem. Specifically, we describe the structure of the ontology, a simple similarity measure, and a mapping algorithm followed by the expert peering scheme.

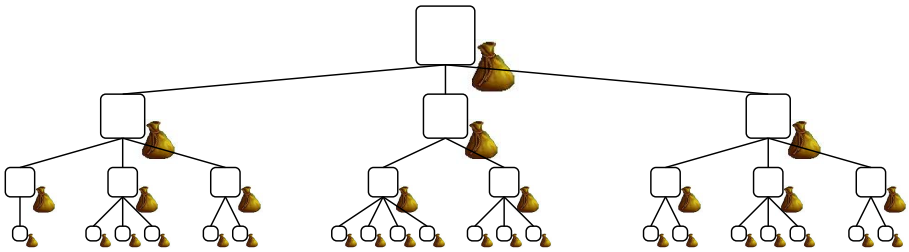


Fig. 1. Didactic example of an ontology tree where each node is associated with a bag of words. In our implementation, the bag of words of a higher level node contains keywords regarding the corresponding topic as well as the union of all bags of words of its descendants.

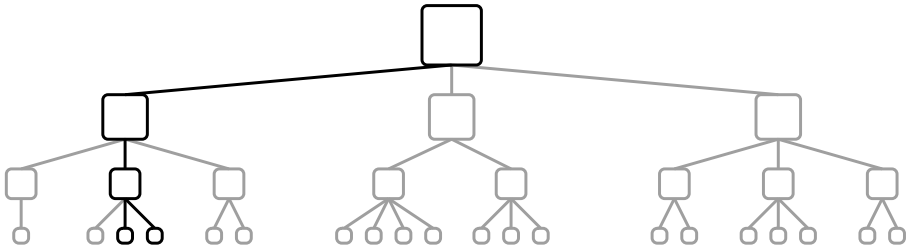
2.1 The Ontology

We consider a strictly hierarchical ontology or knowledge tree $T = (\mathcal{N}, \Sigma)$ consisting of a set of nodes or subjects $\mathcal{N} = \{n_1, \ldots, n_N\}$ and a set of edges Σ between them such that each subject $n \in \mathcal{N}$ has a unique parent node corresponding to a broader subject (see Fig.1). Other than this assumption the approach we develop in this paper is independent of the nature and contents of the specific ontology tree chosen.

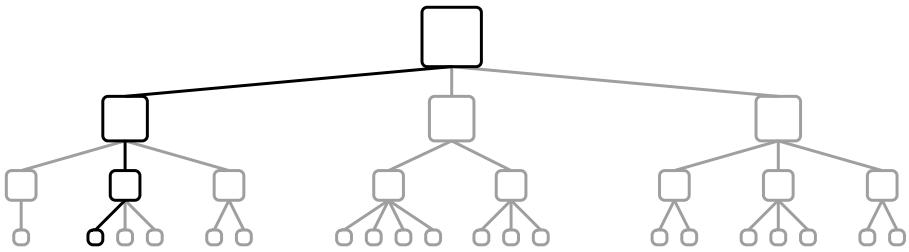
Let us define for notational convenience $\mathcal{C}(n)$ as the set of children and $p(n)$ as the unique parent of node n. We associate each node n with a representative *bag of words* $\mathcal{B}(n) := \{w_1, \ldots, w_{B_n}\}$, where w_i denotes the i^{th} word. The bag of words can be for example obtained by processing a collection of related texts from online and encyclopedic resources using well-known natural language processing methods. Subsequently, we optimize all of the bag of words $\mathcal{B}$ in the ontology both vertically and horizontally in order to strengthen the hierarchical structure of the tree and to reduce redundancies, respectively. First, in the vertical direction, we find the union of bag of words of each node n and the ones of its children $\bar{\mathcal{B}}(n) = \mathcal{B}(n) \cup [\cup_{i \in \mathcal{C}(n)} \mathcal{B}(i)]$. Then, we replace $\mathcal{B}(n)$ with $\bar{\mathcal{B}}(n)$ for all $n \in \mathcal{N}$. Next, in the horizontal direction, we find the overlapping words among all children of a node n, $\tilde{\mathcal{B}}(n) = \cap_{i \in \mathcal{C}(n)} \mathcal{B}(i)$, subtract these from each $i \in \mathcal{C}(n)$ such that $\mathcal{B}(i) = \mathcal{B}(i) \setminus \tilde{\mathcal{B}}(n)$, and repeat this for all $n \in \mathcal{N}$.

2.2 Mapping to Ontology-space

The described ontology tree can easily be serialized by, for example, ordering its nodes from top to bottom and left to right. Hence, we obtain an associated vector representation of the tree $\mathbf{v}(T) \in \mathbb{R}^N$. An important aspect of our algorithm is the representation of entities as subtrees of the ontology (see Fig. 2) and equivalently as vectors on the so called *ontology-space* $S(T) \subset \mathbb{R}^N$. In order to map expert profiles and queries, which are given by arbitrary keyword lists, onto subtrees, we use a similarity measure between any entity representable by a bag of words and the ontology tree. In this paper, we choose the subsequenly described measure and mapping algorithm. However, a variety of similarity measures can be used towards this end.



(a) An expert on two topics represented by leaves of the ontology can be modeled as a subtree.



(b) A query on a topic represented by a leave of the ontology can be modeled as a subtree.

Fig. 2. Didactic example of how experts and queries can be mapped to subtrees of an ontology and how this leads to the most suitable peering. For instance, a query on electromagnetism (a topic represented by a leave node) is, in a wider sense, a query on theoretical physics which, in turn, is a query in the area of physics in general. Therefore, even if there is no expert on electromagnetism, an expert on quantum mechanics and computational physics might be able to help, since these, too, are topics in theoretical physics.

It is customary to define a global dictionary set as $\mathcal{D} := \cup_{n \in \mathcal{N}} \mathcal{B}(n)$ of cardinality M and an M-dimensional *dictionary-space* $S(\mathcal{D}) \subset \mathbb{R}^M$ by choosing an arbitrary ordering. Thus, each node or item i is associated with an occurrence vector $\mathbf{w}(i)$ in the dictionary space indicating whether or not a word appears in the respective bag of words:

$$\mathbf{w}(i) := [I(w_1), \ldots, I(w_M)], \quad \mathbf{w}(i) \in S(\mathcal{D}), \tag{1}$$

where $I(w_j) = 1$ if $w_j \in B(i)$ and $I(w_j) = 0$ otherwise. Note that the vectors $\mathbf{w}$ are usually sparse as the cardinality of $B(i)$ is usually much smaller than the one of $\mathcal{D}$.

We now define an example similarity measure $r(i, j)$ between two entities i and j (with respective bag of words $B(i)$, $B(j)$ and vectors $\mathbf{w}(i)$, $\mathbf{w}(j)$):

$$r(i, j) := \frac{\sum_{k \in B(i) \cap B(j)} 1}{\sqrt{\mid B(i) \mid}\sqrt{\mid B(j) \mid}}, \tag{2}$$

where $\mid \cdot \mid$ denotes the cardinality of a set. Note that this measure actually corresponds to the the cosine of the angle between occurrence vectors but clearly does not require to assemble a global dictionary $\mathcal{D}$ for its computation.

Input: bag of words $\mathcal{B}(i)$, ontology tree T, similarity measure r
Output: corresponding subtree describing entity i and its vector representation $\mathbf{v}(i)$

/ Compute the similarity between the given entity's bag of words $\mathcal{B}(i)$ and the ones of the tree nodes iteratively from top to bottom */*

1. consider the highest semantic categories $\{n_{h_1}, \ldots, n_{h_H}\}$, i.e. each node n immediately below the root node, and compute the similarities $r(i, n)$
2. determine the node $n_k \in \{n_{h_1}, \ldots, n_{h_H}\}$ with the highest similarity
3. add n_k to the resulting subtree and set the corresponding vector entry to 1
4. consider the child nodes $\{n_{c_1}, \ldots, n_{c_C}\}$ of n_k and for each child node n compute the similarities $r(i, n)$
5. compute the mean μ and the standard deviation σ of the resulting similarities
6. consider all nodes $\{n_k\}$ in the current set of children for which $r(i, n_k) > \mu + \alpha\sigma$ where $\alpha \geq 0$ is a fixed parameter
7. for each node n_k in the set $\{n_k\}$ continue with step 3, until the lowest level of the tree is reached

Fig. 3. Algorithm to map an entity i characterized by an arbitrary list of keywords to a subtree of an ontology whose nodes are associated with bags of words.

Given the similarity measure we present an efficient mapping from the dictionary space to the ontology-space $S(\mathcal{D}) \rightarrow S(T)$ through the algorithm in Fig. 3. Using this algorithm, any item i can be represented as a subtree of the ontology or alternatively as a vector $\mathbf{v}(i) \in S(T)$ with one-zero entries on the ontology-space. We note that the algorithm in Fig. 3 is inherently robust due to its top-to-bottom iterative nature and usage of the ontology's hierarchical structure. In other words, it solves a series of classification problems at each level of the tree with increasing difficulty but in a sense of decreasing importance. We will discuss this in the next section in more detail. The optimizations of the ontology tree described in Section 2.1 also add to the robustness of the mapping, especially the aggregation of bag of words from leaves to the root.

2.3 Query-Expert Peering

As the first step of query-expert peering, we convert each query to a bag of words and associate each expert with one, too. The expert's bag of words can be derived, for example, by processing personal documents such as resumes, webpages, blogs, etc. The algorithm in Fig. 3 enables us then to represent any query or expert as a subtree of the ontology as well as a binary vector on the ontology-space. Thus, the query-expert peering problem becomes one of graph (tree) matching which we in turn address by using the ontology tree to span the corresponding linear space. There are two important advantages to this approach:

1. The ontology (vector) space has a much smaller dimension than the commonly used term-by-document spaces; also, it avoids the need for maintaining inefficient, large, and static dictionaries.

2. Each dimension of the ontology-space, which actually corresponds to a node (subject), has inherent semantic relations with other nodes. One such relation is hierarchical and immediately follows from the tree structure of the ontology. However, it is also possible to define other graph theoretic relations, for example, by defining overlay graphs.

We now describe a basic scheme for query-expert peering. Let us denote by q a query with its bag of words $\mathcal{B}(q)$ and by $\mathcal{E} = \{e_1, \ldots, e_E\}$ a set of experts represented by the respective bag of words $\mathcal{B}(e_i),\ i = 1, \ldots, E$. Our objective is to find the best set of experts given the query. Using the approach in Section 2.2 we map the query and experts to subtrees of the ontology and hence to vectors $\mathbf{v}(q)$, and $\mathbf{v}(e_1), \ldots, \mathbf{v}(e_E)$, respectively, on the ontology space $S(T)$. Then, we define a matching score m between a query and an expert

$$m(q, e) := \mathbf{v}(q) \cdot \mathbf{v}(e), \tag{3}$$

as the dot product of their vectors. Subsequently, those experts with the highest ranking matching scores are assigned to the query.

3 Experiments

We conduct a set of preliminary offline experiments to numerically study the performance of the system developed. We present next the experiment setup followed by numerical results and a discussion of them.

3.1 Experiment Setup

We begin the experimental setup by selecting an ontology and associate each of its nodes with a bag of words as described in Section 2.1. In this paper we choose (rather arbitrarily) a 245 node subset of an ontology[1] prepared by the Higher Education Statistics Agency (HESA), an educational institution in the United Kingdom. The bag of words for each node is obtained via the following procedure:

1. The node's name is used in finding 10 top ranked documents through *Yahoo!* search web services[2].
2. The obtained HTML documents are converted to text (ASCII) format and concatenated into a single document.
3. This resulting document is further processed using the Natural Language Toolkit (NLTK) [10] by (a) tokenizing, (b) stop word removal, and (c) stemming with Porter's stemmer [11], which finally yields the bag of words.

We next randomly generate experts for the purpose of offline experiments. We consider three types of experts: one knowledgeable in a single specific topic (represented by a subtree of ontology ending at a single leaf node), one with two specific topics

[1] http://www.hesa.ac.uk/jacs/completeclassification.htm
[2] http://developer.yahoo.com/search/

(branches), and one with three topics. Each randomly generated pool of experts contains equal number of each type.

One can devise a variety of methods for random query generation. However, the procedure for generating queries with a known answer (an ordering of best matching experts) is more involved. We choose to overcome is difficulty by generating a separate "query" bag of words for each node of the ontology following the steps above. We ensure that these bags of words are obtained from documents completely disjoint from the ones used to obtain node-associated bag of words. Thus, we generate queries by randomly choosing a node from the ontology and a certain number of keywords from its "query" bag of words. Since we know which node the query belongs to, we easily find a "ground truth" subtree or vector associated with the query which in turn allows computing the "best" ordering of experts for peering. This yields a basis for comparison with the result obtained from the generated query.

Finally, we use the similarity measure and mapping algorithm described in Section 2 to compute the expert peering, i.e. the set of experts $R(q)$ with highest matching scores given a query q. Then, as described above the "ground truth" vectors are used to calculate another set of matching experts $A(q)$ of a certain given cardinality. The recall and precision measures are calculated as the average of $Nbr = 1000$ such queries in these experiments:

$$recall = \frac{1}{Nbr}\sum_{i=1}^{Nbr}\frac{\mid A(q_i) \cap R(q_i) \mid}{\mid A(q_i) \mid}, \quad precision = \frac{1}{Nbr}\sum_{i=1}^{Nbr}\frac{\mid A(q_i) \cap R(q_i) \mid}{\mid R(q_i) \mid}.$$

3.2 Numerical Results

We now present and discuss the numerical results. In the experiments we choose the following specific parameter values: the number of query keywords (out of respective "query" bag of words) $\{20,\ 40,\ 60,\ 80,\ 100\}$, the number of experts $\{50,\ 100\}$, and the parameter α of the algorithm in Fig. 3 $\{0.0,\ 1.0\}$.

We first limit the cardinality of A to one, i.e. there is only a single expert in the "correct" peering set. The precision and recall versus the range of parameters in this case is shown in Figures 4(a) and (b), respectively. Aiming to find only the single best matching expert is clearly over restrictive and leads to poor results. In fact, given the uncertainties within the underlying representation mechanisms it is neither very meaningful to expect such degree of accuracy nor required for the application areas considered.

Next, the best matching experts are taken to be the ones with the three highest ranking scores. The precision and recall improve drastically for all parameter choices as observed in Figures 5(a) and (b), respectively. This result demonstrates the robustness of our expert peering scheme: it's performance improves gradually when accuracy restrictions are eased. This is further illustrated by Figures 6(a) and (b), where the performance further increases when the number of best matching experts is increased to the top six ranks.

Our observations on and interpretations of results with respect to the values of other parameters include:

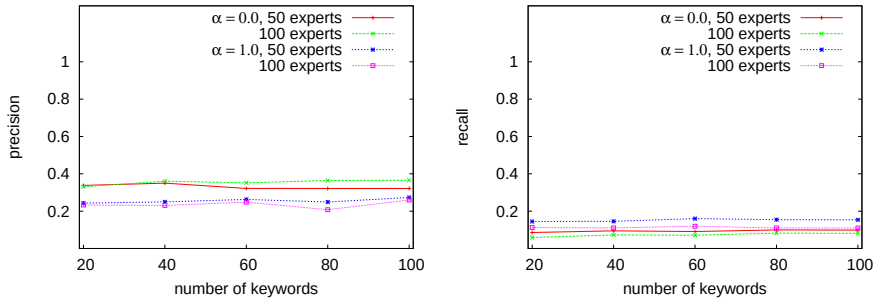


Fig. 4. (a) Precision and (b) recall for a range of parameters when we find only the best matching expert to each query.

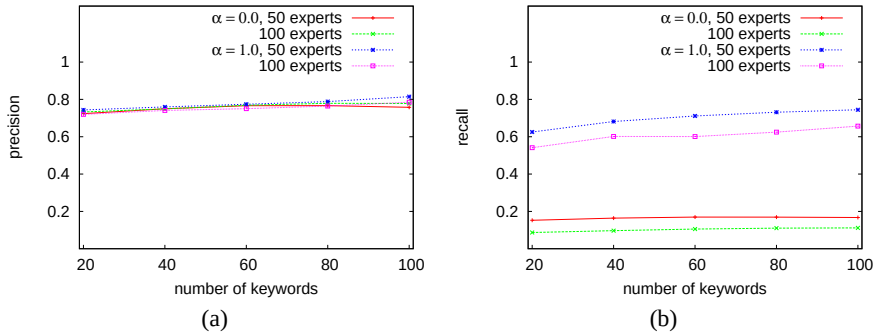
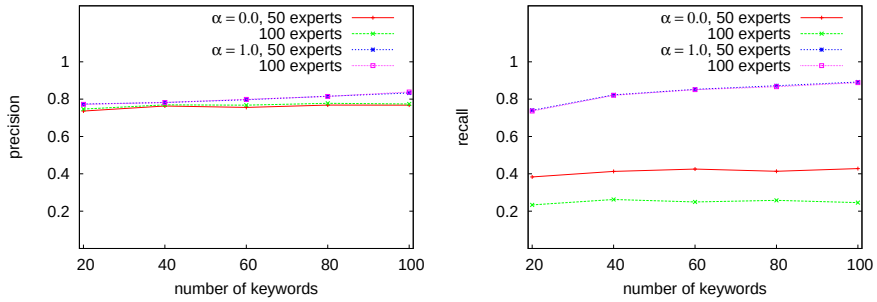


Fig. 5. (a) Precision and (b) recall for a range of parameters when we find the three best matching experts to each query.

1. Choosing the larger $\alpha = 1$ value for the algorithm in Fig. 3 leads to improved results. Since this parameter affects the branching threshold value when mapping queries to a subtree of ontology we conclude that increasing it restricts unnecessary branching, and hence noise.
2. The precision remains high regardless of the number of experts and α in Figures 5 and 6. We attribute this to hierarchical structure and robustness of our system.
3. With the correct set of parameters we observe in Fig. 5 and especially Fig. 6 that both the precision and recall are relatively insensitive to the number of experts which indicates scalability.
4. Although the precision and recall slightly increase with increasing number of words in the queries these curves are rather flat demonstrating that our system performs well in peering the experts even when given limited information.



(a) (b)

Fig. 6. (a) Precision and (b) recall for a range of parameters when we find the six best matching experts to each query.

4 Conclusion

In this paper we have presented an ontology-based approach for an expert peering and search system. We have studied the underlying principles of a real-time search engine for an online community where users can ask experts, who are simply other participating users knowledgeable in that area, for help on various topics. We have described a graph-based representation scheme consisting of an ontology tree where each node corresponds to a (sub)topic and is associated with a bag of words. This allows us to represent the fields of expertise (profile) of the participating experts as well as incoming queries as subtrees of the ontology. Subsequently we address the resulting graph matching problem of assigning queries to relevant experts on a vector space, which follows from a serialization of the ontology tree, using simple dot products of respective vectors.

Preliminary experiments utilizing an example ontology demonstrate the efficiency, robustness, and high performance of our algorithm over a range of parameters. These promising results also open the way for future research. One research direction is the refinement of our algorithm toward an adaptive update of the α parameter that controls the branching behavior in subtree generation. Another interesting question is how to make the underlying ontology dynamic by adding, deleting, and merging nodes. Yet another direction is the study of time-varying expert profiles and it's analysis as a dynamic system. We finally note that although the expert peering problem we focus on in this paper has specific properties differing from document retrieval our approach can be applied to that area as well.

References

1. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Press (2002)
2. Hofmann, T.: Latent Semantic Models for Collaborative Filtering. ACM Trans. on Information Systems **22**(1) (2004) 89–115

3. Ding, C.: Document Retrieval and Clustering: from Principal Component Analysis to Self-aggregation Networks. In: Proc. Int. Workshop on Artificial Intelligence and Statistics. (2003)
4. Miller, L.: Document Representation Models for Retrieval Systems. ACM SIGIR Forum **14**(2) (1979) 41–44
5. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of web documents using a graph mode. In: Proc. Int. Conf. on Document Analysis and Recognition. (2003) 240–244
6. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of Web Documents Using Graph Matching. Int. J. of Patter Recognition and Artificial Intelligence **18**(3) (2004) 475–496
7. Lim, S.Y., Park, S.B., Lee, S.J.: Document retrieval using semantic relation in domain ontology. In: Proc. Int. Atlantic Web Intelligence Conf. Volume 3528 of LNAI., Springer (2005) 266–271
8. Chung, S., Jun, J., McLeod, D.: A web-based novel term similarity framework for ontology learning. In: ODBASE: Int. Conf. on Ontologies, Databases and Applications of Semantics., Montpellier, France (2006)
9. Wu, J., Yang, G.: An ontology-based method for project and domain expert matching. In: Proc. Int. Conf. on Fuzzy Systems and Knowledge Discovery. Volume 4223 of LNAI., Springer (2005) 176–185
10. Bird, S., Klein, E., Loper, E.: The natural language toolkit (NLTK) (2001)
11. Porter, M.: An Algorithm for Suffix Stripping. Program **14**(3) (1980) 130–137