Software.zhishi.schema: A Software Programming Taxonomy Derived from Stackoverflow

Jiangang Zhu[1], Haofen Wang[2], and Beijun Shen[1]*

[1] School of Software, Shanghai Jiao Tong University, Shanghai, 200240, China
{jszjgtws,bjshen}@sjtu.edu.cn
[2] East China University of Science and Technology, Shanghai, 200237, China
whfcarter@ecust.edu.cn

Abstract. In this paper, we are the first to construct a software programming taxonomy from Stackoverflow. More precisely, we propose a machine learning based method with novel features to capture the hierarchical semantic structure of tags in Stackoverflow. A graph pruning algorithm is applied to eliminate the conflicts by constructing a Directed Acyclic Graph (DAG). As a result, our dataset, named Software.zhishi.schema, contains 38,205 concepts together with 36,249 subsumption relations. In order to further test the usability of our published data, we adopt a similarity computing task of words from software programming which is one of the most fundamental tasks in the software repository mining area. The results show that our dataset can outperform other knowledge bases due to its high coverage with finer-grained domain concepts.

1 Introduction

Taxonomy is playing a more and more important role in software engineering. For example, in software maintenance such as measuring quality and predicting defects, taxonomies can be used to measure the relatedness between documents and create links between bugs and committed changes. While WordNet is one of the most widely used taxonomy in the world, it does not play well in software engineering due to the low coverage of software programming terms. With the development of Wikipedia, several taxonomies such as Yago Taxonomy and WikiTaxonomy have been developed and published on the Web. However, Wikipedia cannot capture the fast changes of techniques in software engineering so that it always fails to update in time. Also, some fine-grained terms about software programming cannot be found in these taxonomies. Lack of a real-world useful software programming taxonomy cramps the development of semantic applications in software engineering.

Recently, Stackoverflow[3] has becoming one of the most popular online QA Web sites for software programming. A key feature of Stackoverflow is that it allows users to annotate questions with tags. These tags represent vocabularies

* Corresponding author
[3] http://stackoverflow.com/

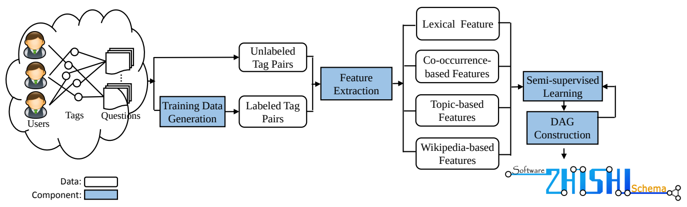

Fig. 1. The approach of constructing taxonomy from Stackoverflow

about software programming. They can also reflect the fast changing nature of technique terms because they are created on the fly by Web users. So the large amount of tags provide a promising way to build the taxonomy. In this paper, we propose a machine learning based approach to construct the taxonomy. To the best of our knowledge, we are the first to build a software programming taxonomy from Stackoverflow. Our contributions mainly include: (1) We propose a semi-supervised learning approach with novel features to detect subsumption relations between tags from Stackoverflow. (2) The largest public available taxonomy about software programming has been published with several access mechanisms. (3) An application study is carried out to show the effectiveness of our dataset.

2 Approach

Features We adopt a classification model to determine the correct subsumption relation of each tag pair. The details of features we used are as follows: **Lexical Feature** *LCS:* the token-based longest common sub-string asymmetric similarity. **Co-occurrence-based Features** *CQ:* the Normalized Google Distance on the questions that the tags in the candidate tag pair have ever occurred in. *CT:* the Normalized Google Distance on the tags that the tags in the candidate tag pair have ever occurred with. *CW:* the Normalized Google Distance on the wiki descriptions that the tags in the candidate tag pair have ever occurred in. *CS:* the Normalized Google Distance on the sentences in all wiki descriptions that the tags in the candidate tag pair have ever occurred in. *CU:* the Normalized Google Distance on the users who have ever annotated the tags in the candidate tag pair. **Topic-based Features** *TW:* the KL-divergence on the topic distributions of the wiki descriptions of the candidate tag pair. *TQ:* the KL-divergence on the topic distributions of the randomly selected questions of the candidate tag pair. **Wikipedia-based Features** *EW:* the cosine similarity on the Wikipedia-based representations (similar to ESA [1]) of the wiki descriptions of the candidate tag pair. *EQ:* the cosine similarity on the Wikipedia-based representations of the randomly selected questions of the candidate tag pair.

Semi-supervised Learning with Constraints The whole workflow of our approach can be seen in Figure 1. We apply a simple but effective semi-supervised learning framework - self-training. Moreover, in order to avoid error propagation to the following iterations, we add two constraints and filter out incorrect or redundant subsumptions by leveraging the global structure information. We have identified two general types of constraints in subsumption detection problem: (1)

Cycle Conflict Constraint: For two tags a and b, if there is a hypernym path from a to b, then we could constrain the subsumption set to have no hypernym path from b to a because subsumption relation is asymmetric. (2) *Transitive Redundancy Constraint:* Given three tags a, b and c, if a subsumes b, b subsumes c and a subsumes c, then the subsumption relation between a and c is redundant due to the transitivity of subsumption. These constraints are quite important to guarantee the quality of newly-added training data in each iteration. *Cycle Conflict Constraint* can avoid introducing error examples. While the *Transitive Redundancy Constraint* can remove redundancies in each iteration. This will guide the learner to learn more fine-grained subsumption relations. To deal with the constraints, in each iteration, we first construct a weighted direct graph. Then, a pruning algorithm is applied to find an optimal taxonomy. We use the Support Vector Machine (SVM) algorithm with RBF kernel to train the binary classifier. As for training data, we propose an effective rule-based method to create labeled data. Some lexical-syntactic patterns (similar to Hearst pattern. e.g. NP1 is a/an NP2) on descriptions of tags are applied to generate candidate subsumption relations.

DAG Construction We introduce a graph-based algorithm that works well in practice to convert subsumption relation set into an optimal tree-structured taxonomy. We first construct a weighted directed graph from the subsumption relation set. The weight of the edge is the confidence generated by SVM classifier.Then, we apply Edmonds' algorithm[4] to find a maximum optimum branching of a weighted directed graph. The resulting taxonomy will be optimal with the sum of the edge weights is maximized.

3 Preliminary Results and Web Access

Subsumption Accuracy Evaluation According to the results, the accuracy increases consistently when we perform more iterations. In particular, after the seventh iteration, the learner with constraints achieves the best accuracy of $85.96\% \pm 2.21\%$ (1,000 samples with *Wilson interval*).

Linked Data Software.zhishi.schema creates URIs for all concepts. The pattern `http://seonto.apexlab.org/stackoverflow/tag/[label]` comprises of two parts. `http://seonto.apexlab.org/stackoverflow/tag/` is the namespace. The other part is the tag label. We use `rdfs:subClassOf` for `subclassOf` relations. When Semantic Web agents that accept "application/rdf+xml" content type access our server, resource descriptions in the RDF format will be returned. Our dataset is available at `http://datahub.io/dataset/software-zhishi-schema`.

Lookup Service We provide a lookup service for users to access Software.zhishi.schema. The service is available at `http://seonto.apexlab.org/lookup`. Given a query, all tags whose labels exactly match the query are returned. We can click on any parent tag or child tag to switch to another page view. Such an interaction stands for navigation in our taxonomy. As a representative example shown in Fig. 2, our taxonomy contains a hyponym path like "data structures" → "tree" → "binary tree" → "binary search tree" → "avl tree".

[4] `http://en.wikipedia.org/wiki/Edmonds'_algorithm`

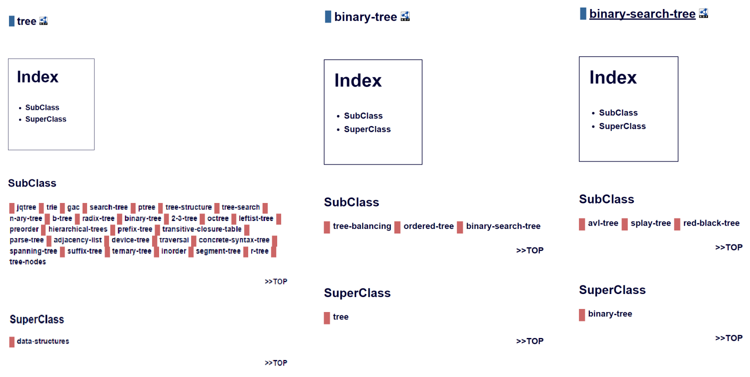
tree
Index
SubClass
SuperClass
SubClass
jqtree trie gac search-tree ptree tree-structure tree-search n-ary-tree b-tree radix-tree binary-tree 2-3-tree octree leftist-tree preorder hierarchical-trees prefix-tree transitive-closure-table parse-tree adjacency-list device-tree traversal concrete-syntax-tree spanning-tree suffix-tree ternary-tree inorder segment-tree r-tree tree-nodes
>>TOP
SuperClass
data-structures
>>TOP
binary-tree
Index
SubClass
SuperClass
SubClass
tree-balancing ordered-tree binary-search-tree
>>TOP
SuperClass
tree
>>TOP
binary-search-tree
Index
SubClass
SuperClass
SubClass
avl-tree splay-tree red-black-tree
>>TOP
SuperClass
binary-tree
>>TOP

Fig. 2. An Example of Software.zhishi.schema

SPARQL Endpoint We also provide a SPARQL endpoint for professional users at `http://seonto.apexlab.org/sparql`.

4 Application

In this section, we investigate whether our dataset can outperform others by constructing a similarity computing experiment on terms from software programming. It is one of the most fundamental tasks in many research areas in software engineering. Unfortunately, there is no benchmark data set for semantic relatedness measuring on terms from software programming. In our experiment, we make our own data set and offer it as a standard for testing terms semantic relatedness[5]. We use the WUP [2] similarity as a semantic relatedness metric on our dataset. WUP similarity on WordNet and ESA [1] are used as comparison methods. We leverage Spearman rank correlation ρ as the evaluation measure. Experimental result shows that our dataset significantly improves the performance of term semantic relatedness measurement. The results of WUP on WordNet and ESA are only 0.0238 and 0.2244 respectively while WUP similarity on our dataset can achieve the spearman correlation of 0.4608.

Acknowledgements This research is supported by 973 Program in China (Grant No. 2015CB352203), National Natural Science Foundation of China (Grant No. 61472242, Grant No. 61402173) and the Fundamental Research Funds for the Central Universities (Grant No: 22A201514045).

References

1. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJCAI. vol. 7, pp. 1606–1611 (2007)
2. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics. pp. 133–138. Association for Computational Linguistics (1994)

[5] The test collection is available at `http://seonto.apexlab.org/termsim`