Learning to Discover Subsumptions between Software Engineering Concepts in Wikipedia

Xiang Dong[†], Kai Chen[†], Jiangang Zhu[†], Beijun Shen[‡]
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, Shanghai, China
Email: [†]dongxiang@sjtu.edu.cn, [†]voyageckg@sjtu.edu.cn, [†]jszjgtws@sjtu.edu.cn, [‡]bjshen@sjtu.edu.cn

***Abstract*—Wikipedia contains large-scale concepts and rich semantic information. A number of knowledge base construction projects such as WikiTaxonomy, DBpedia, and YAGO have acquired data from Wikipedia. Despite the huge amount of relations in Wikipedia, the semantic relations (i.e. subsumptions) between domain concepts are rather sparse, especially in software engineering (SE) area. Hence, it is difficult to derive a software engineering knowledge base directly from Wikipedia. Meanwhile, domain knowledge base has become indispensable to a growing number of applications in software engineering. So the discovery of missing semantic relations between software engineering concepts in Wikipedia is essential. In this paper, we propose an approach for automatically discovering the missing subsumption relations between software engineering concepts. Specifically, we extract the SE domain concepts from Wikipedia firstly. And secondly, we design a machine learning based algorithm with some novel features to calculate the semantic relevancy between concepts. Thirdly, we offer and utilize a semi-supervised model to incorporate the features, which discovers the SE subsumptions. Experimental results show that our approach can effectively find the missing subsumption relations between software engineering concepts. Finally, we build a taxonomy which contains 193,593 concepts together with 357,662 subsumption relations. Compared with the taxonomies which are extracted from general-purpose knowledge bases such as WikiTaxonomy, YAGO and Schema.org, our dataset has a larger scale in software engineering domain.**

***Index Terms*—Subsumption Extraction, Software Engineering, Wikipedia**

I. Introduction

Wikipedia is the largest online encyclopedia in the world. It contains more than 5,027,000 concepts, and its extensive network of links, categories and infoboxes provide a variety of explicitly defined semantics. Thus, some knowledge bases use Wikipedia as the data source of relation extraction. For example, Wikitaxonomy [1], DBpedia [2], and YAGO [3] are the large-scale knowledge bases which are derived from Wikipedia.

However, these general-purpose knowledge bases only contain a small quantity of concepts and relations in software engineering (SE) domain. There is no one owning more than 1,000 SE concepts or relations. For instance, the subsumption between "Sandbox (computer security)[1]" and "Virtualization software[2]" is a subsumption relation in software engineering, and "Sandbox" is an important SE concept, but they can not be found in existing knowledge bases. Especially, a lot of relations between SE concepts are not discovered when analyzing knowledge data from Wikipedia, however, they are more important for SE researches and practices such as semantic relatedness calculation [4] [5] [6], document correlation analysis [7], and defect prediction.

In this paper, we propose an approach for automatically discovering the SE domain subsumptions in Wikipedia, so that the missing SE semantic subsumption can be gathered and established. However, to complete the work, we would face these challenges:

- Software engineering concepts are always composed of domain-specific terms. While natural language processing techniques like segmentation or pos tagging are the basis of some Web-based approaches for taxonomy construction, directly applying these approaches to our scenario will lead to poor results.
- Several structural information is contained in Wikipedia. How to design an algorithm to incorporate this information when detecting subsumptions between concepts is another important topic.
- The invalid SE entities would increase the difficulty of subsumptions prediction when taking the Wikipedia structural information into consideration.

To solve the challenges above, we propose a machine learning based approach for domain subsumption prediction. Particularly, it leverages several features from different aspects to measure the semantic relatedness between concepts. Thus, the confidence of relation prediction would be increased. The semi-supervised model has been proposed to process the information. We split the whole machine learning process into several iterations to extract and optimize the relations set. So in this paper, our work and contributions mainly include:

A. Using wiki-based features for classifiers training: we utilize the structure based features in Wikipedia, and combine with the lexical, co-occurrence and distribution features to calculate similarity of different concepts. It increases the performance of subsumption extraction.

B. Automatic labeling and iterating: we define the constraint-based rules to label the negative data, and adopt subsumption patterns to extract the positive data. During each iteration of the semi-supervision, several constraints are defined to automatically optimize the relations set, and in this way, we shrink the cycle of data handling.

[‡] Corresponding Author

[1]https://en.wikipedia.org/wiki/Sandbox_(computer_security)

[2]https://en.wikipedia.org/wiki/Category:Virtualization_software

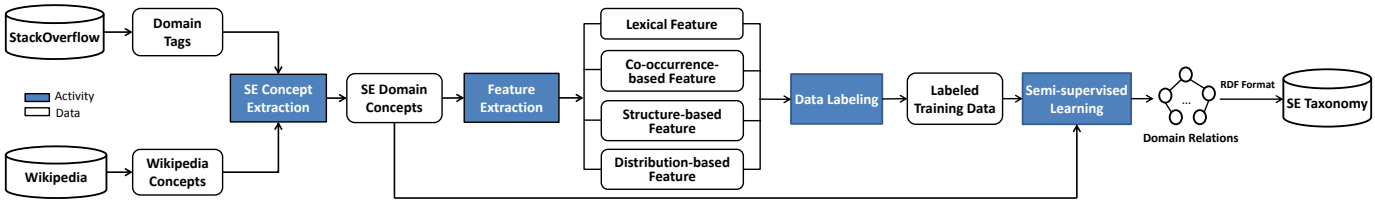

Fig. 1: Our approach for extracting SE relations in Wikipedia

C. Extracting SE subsumptions and building taxonomy: we discover and extract the SE subsumptions in Wikipedia, and build the large-scale taxonomy in software engineering domain. The experimental results show our taxonomy contains the large-scale concepts and deep semantic hierarchies.

II. Related Work

There consists amount of researches on relations extraction in general area. Auer et al. presented the DBpedia system [2] which generates RDF statements by extracting the attribute-value pairs contained in infoboxes of Wikipedia (e.g. country = [[the People's Republic of China]] is a key-value pair in ShangHai page). Suchanek et al. built the YAGO system which refers to relation exploration [3]. The method of YAGO construction is getting all the "is-a" hierarchy in WordNet and then mapping into Wikipedia for corresponding instances. Merging Wikipedia with WordNet makes YAGO contain over 200,000 classes and 400,000 subsumption relations. Weld et al. in Kylin Ontology Generator (KOG) [8] built a subsumption hierarchy by combining Wikipedia infoboxes with WordNet using statistical-relational learning. WikiTaxonomy [1] used the way of finding some special rules for subsumption in Wikipedia category structures. For example, "Companies listed on NASDAQ" is a category of "Microsoft" and it is the plural format, so WikiTaxonomy refers the pair as a subsumption relation. Furthermore, WikiTaxonomy uses inference method to extract the relationship again based on the existing concepts, and it totally obtains 105,000 concepts with the precision of 88%. Zhishime [9] is the trial for relations discovery based on Chinese Wikipedia. In addition, Lin et al. [10] used Freebase as the supplement of Wikipedia, finding the relations by solving invalid entities in Wikipedia. Wu et al. presented a special method to construct probase [11] system by assigning the probabilities to each relationship. It extracts 2.7 million concepts.

As for relations discovery in SE domain, Lextcal Views [12] applied some natural language processing techniques to automatically extract and organize concepts relationship from software identifiers in a WordNet-like structure. But more software programming terms would not be included in this relation set because Lexical Views only use the software identifiers as its input. Zhu et al. [13] extracted hypernym-hyponym relations between tags in StackOverflow, and built a taxonomy called Software.zhishi.schema. It obtains 38,205 concepts and 68,098 relations, which still has the rise of space in scale.

III. Approach

A. Approach Overview

In order to discover the subsumption relations between the software engineering concepts, a machine learning based approach is proposed, as shown in Fig 1. It consists of four key activities:

1) SE Concept Extraction: through the input of SE domain tags in StackOverflow and Wikipedia concepts, we extract the SE domain concepts as preparation of the relations discovery.

2) Feature Extraction: features are launched for machine learning, and these are divided into four classes: *Lexical Features*, *Co-occurrence-based Features*, *Structure-based Features* and *Distribution-based Features*.

3) Data Labeling: it generates the labeled data for classifiers. For negative data, we use constrains-based method for training. And for positive data, Hearst [14] patterns are adopted to get some subsumptions for positive data generation.

4) Semi-supervised Learning: the semi-supervised model divides the whole learning process into several iterations. And in each iteration, we review the trained results of the current iteration and update high confidence data as the input of the next.

Finally, it extracts 357,662 subsumptions in our relation set, and based on the discovered subsumptions, we build the SE domain taxonomy in RDF format. We deploy our taxonomy on the website[3]. Following subsections will describe these activities in detail.

B. Domain Extraction

The process of domain extraction can be illustrated as following steps: first, we get the domain tags from StackOverflow, and it contains tens of thousands of items. Through the word-embedding method and voting mechanism, we filter out all the domain-irrelevant tags and remain the seed words which are needed. Second, in order to get structural information from Wikipedia, we dump the Wikipedia XML file[4] and obtain a large-scale set which contains 5,027,000+ concepts. Third, based on the built concept repository, we extract the domain concepts by using seed words derived from the StackOverflow. As the result, we obtain the software engineering corpora with 193,593 concepts, which is for the next relations discovery works.

[3]https://datahub.io/dataset/setaxonomy
[4]https://dumps.wikimedia.org/enwiki/20150901/

C. Feature Extraction

We adopt classification models to determine the correct subsumption relation of each concept pair, for which several effective features are designed. The purpose of feature engineering is to quantitatively characterize the similarities or relatedness between concepts. We divide all the features into four classes: *Basic Features*, *Co-occurrence-based Features*, *Structure-based Features* and *Distribution-based Features*.

1) ***Basic Features****:* this type of feature is aimed at capturing the lexical similarity between two concepts, and it includes two features: *Head correlation calculation* and *Unbalanced common substring*.

Feature 1. Head correlation calculation*:* "head" means the core word of concept. As for the analysis of head matching, our approach is to build the semantic tree by using Stanford Parser [15]. We judge and extract the core noun phrase from the tree as the head of the term. After obtaining the subsumption's head, we adopt *WUP* [16], which is the similarity calculation method based on WordNet structure, to measure the relatedness. The *WUP* formula is as follows:

$$WUP\left(X,\ Y\right)\ =\ \frac{2\cdot depth\left(\mathrm{LCA}\left(H_x\ ,\ H_y\right)\right)}{depth\left(H_x\right)+depth\left(H_y\right)} \tag{1}$$

where H_x, H_y are the head of the concepts *X* and *Y*. *LCA* is the lowest common ancestor of H_x and H_y in WordNet.

However, if there exists several heads in one concept, we compute *WUP* for all, and then compare and select the maximum value. For example, as for concept "Software using the Apache License" and "Android (operating system)", we fetch the heads "Software" and "Apache license" in first concept, and fetch the "Android" in second one. Then, we calculate the *WUP* values both between "Software" & "Android" and "Apache license" & "Android". The maximum of these two values is served as the final feature figure of the relation between "Software using the Apache license" and "Android (operating system)".

Feature 2. Unbalanced common substring*:* this feature is designed to calculate the longest common substring of two concepts, and we get the unbalanced lexical length character into formulating construction:

$$f\left(X,Y\right)\ =\ \frac{\mathit{LCS}\ \left|X,Y\right|}{Len\left(X\right)} \tag{2}$$

where *Len(X)* means the length of *X*, and *LCS(X, Y)* means the longest common substring of *X* and *Y*. The increase of value means *X* is more likely to be the hypernym of *Y*.

2) ***Co-occurrence-based Features****:* some relations hold the subsumptions with low lexical similarities. Thus, we leverage the co-occurrence information, which is in the content of the concept, to measure the semantic relatedness between two concepts.

For the co-occurrence-based features, we first define the formula of co-occurrence calculation as follows:

$$r\left(a,\ b\right)=\frac{\log\left(\max\left(\left|I_a\right|\ ,\ \left|I_b\right|\right)\right)-\log\left(\left|I_a\ \cap\ I_b\right|\right)}{\log\left(\left|W\right|\right)-\log\left(\min\left(\left|I_a\right|\ ,\ \left|I_b\right|\right)\right)} \tag{3}$$

The formula represents Normalized Google Distance (NGD), where a and b are the concepts of Wikipedia, and I_a, I_b are the inner-link sets in article. Feature 3 and Feature 4 are defined based on this formula.

Feature 3. Abstract co-occurrence calculation*:* abstract is a brief introduction, and contains the concept's definition. So abstract can effectively reveal the main characteristics of current concept.

$$f_{abs}\left(a,\ b\right)\ =\ r\left(I_{abs\_a}\ ,\ I_{abs\_b}\right) \tag{4}$$

We define the abstract co-occurrence calculation feature $f_{abs}\left(a,\ b\right)$ above, and I_{abs_a}, I_{abs_b} are the link sets in abstract structure of a and b.

Feature 4. Text co-occurrence calculation*:* text is the main text body of concept, and it is responsible for comprehensively describing the concept with enough related information. We define the co-occurrence calculation in text:

$$f_{txt}\ \left(a,\ b\right)\ =\ r\left(I_{txt\_a}\ ,\ I_{txt\_b}\right) \tag{5}$$

where I_{txt_a}, I_{txt_b} are the link sets in main text body of concepts.

Feature 5. Category co-occurrence calculation*:* the category lists the concepts which current concept belonged to, so some potential subsumptions are contained in the category. In this feature, we calculate the co-occurrence in category structure. Furthermore, if *a* is in the category of *b* or vice versa, they maybe more likely a subsumption, so we add weight to this situation based on *NGD* formula. The $f_{cate}(a,b)$ is as follows:

$$\frac{\log\left(\max\left(\left|I_a\right|\ ,\ \left|I_b\right|\right)\right)-\log\left(\left|I_a\cap I_b\right|+f_{bls}\left(a,b\right)\right)}{\log\left(\left|W\right|\right)\ -\ \log\left(\min\left(\left|I_a\right|\ ,\ \left|I_b\right|\right)\right)} \tag{6}$$

In the $f_{cate}(a,b)$ above, $f_{bls}(a,b)$ represents the weight of inclusion. We define $f_{bls}(a,b)$ in equation (7), and μ,ν>0:

$$f_{bls}\left(a,\ b\right)\ =\ \begin{cases}\mu & ,\ a\ \in I_{cat\_b}\\ \nu & ,\ b\ \in I_{cat\_a}\\ 0 & ,\ others\end{cases} \tag{7}$$

where I_{cat_a} and I_{cat_b} are category sets of *a*, *b*. Based on the analysis of weight impact, we set $\mu=5$, $\nu=5$ in our experiment.

3) ***Structure-based Features****:* The *Co-occurrence-based Features* are only calculated for relevancy in the content, but the analysis in Wikipedia structure is also important. Thus, we launch the *Structure-based Features*.

Feature 6. Similarity calculation of guideline*:* guideline is a Wikipedia structure, and it lists the chapters which would be illustrated in the text body of page. For example, in page "Binary search Tree", the guideline includes chapters "Definition", "Operations" and "Examples of applications" etc., and the "Operations" also includes "Searching", "Insertion" and "Deletion". These listed chapters are mapped to the text body and described in detail. Thus, guideline reveals the whole structure of the text, and it is also one of the

important structures. We use *Jaccard* to calculate the similarity in guideline.

$$Jaccard\left(I_{gdl\_a},\ I_{gdl\_b}\right) \ = \ \frac{I_{gdl\_a} \ \cap \ I_{gdl\_b}}{I_{gdl\_a} \ \cup \ I_{gdl\_b}} \qquad (8)$$

where I_{gdl_a}, I_{gdl_b} are the chapter sets of guidelines.

Feature 7. Similarity calculation of infobox: infobox is another structure provided by Wikipedia, and it lists the attributes to present the information of concept. We take I_{info_a}, I_{info_b} into formula (8) to calculate the *Jaccard* value of infobox, and the I_{info_a} and I_{info_b} mean the attribute sets of infoboxes.

4) ***Distribution-based Features:*** this feature is for measuring the difference between two topic distributions of two different concepts.

Feature 8. Divergence calculations: as for the subsumptions that have low similarity in content and structure, we adopt *KL-divergence* [17] to do the further analysis. In the process of *KL-divergence*, we use *LDA* [18] to build the Wikipedia topic-based distribution, and calculate the probability distributions of the two concepts. Finally, we compute the *KL-divergence* between the two distributions.

$$D_{KL}\left(p_{wa} \ || \ p_{wb}\right) \ = \ \sum_{n=1}^{N} p_{wa}\left(n\right) \log \frac{p_{wa}\left(n\right)}{p_{wb(n)}} \qquad (9)$$

where $p_{wa}(n)$ and $p_{wb}(n)$ are the probability of *n*-th topic in the topic distributions of *a*, *b*.

D. Data labeling

Subsumption detection is usually treated as a binary class classification problem. And classification is a supervised learning, which requires labeled data for training. Thus, the generation activity should be taken into consideration for training adequate and fine distribution labeled data.

As for the positive data, we use Hearst patterns in abstract of each page to find some subsumptions as our positive data. The following patterns are used in our approach:

NP1{,} "such as" NPList2
NP1 {,}"and other" NP2
NP1 {,}"including" NPList2
NP1 "is a" NP2
NP1 "is the" NP2 "of" NP3

Part of subsumptions are extracted by Hearst, and we will select some subsumptions and calculate the feature values for them, to label as the positive data.

As for the negative data, we first extract some relations between concepts, and calculate the feature values. Through feature-based constraints, we predict if these relations meet the negative conditions. For example, about the undetermined relation (X, Y), we proceed the judgment on following constraints:

1) $WUP(X,Y) < M$
2) $Len(X) > Len(Y)$
3) $Jaccard\ _{info} = 0$
4) $Jaccard\ _{gdl} = 0$
5) $|KL(X,Y) - KL(Y,X)| < N$

where $WUP(X,Y)$ calculates the *WUP* value between *X* and *Y*, $Len(X)$ gets the length of *X*, and $KL(X,Y)$ obtains the *KL-divergence* between *X* and *Y*. In these constraints, *M* and *N* are set as ascertained value to meet the requirement of our engineering. If any constraint above is satisfied, relation *(X, Y)* will be labeled as negative. In our experiment, *M* = 0.4 and *N* = 0.003.

E. Semi-supervised learning

We design a semi-supervised learning algorithm to extract SE relations, which splits the whole training process into several iterations. In the first iteration, we import the labeled training data and SE domain concepts. And in the following iteration, the input is the optimized results processed by the last iteration. We totally enforce 5 iterations and build a SE domain relation set as outcome.

However, the data quality in previous iteration could make a huge impact on the next iteration, therefore some incorrect data if not discarded may spread along the progress of iteration and cause more inaccuracy. In order to avoid this mistake, we define some rules between two adjacent iterations to filter out the incorrect relations. In this paper, we launch three useful constraints as follows:

- *Constraint 1:* ring conflict constraint. Given the two concepts a and b, if there exists a path from *a* to *b*, the path from *b* to *a* is not allowed. This is because the subsumed relation is asymmetric. For example, If we find a subsumption "Sorting algorithms" → "Quicksort", and then find "Quicksort" → "Sorting algorithms", the relation "Quicksort" → "Sorting algorithms" should be removed.
- *Constraint 2:* transitive redundancy constraint. Given concepts a,b and c. If a subsumes b, and b subsumes c, then a must subsume c due to the subsumption satisfy transitivity. This subsumption is unneeded in our relation set. For instance, the concept "word2vec" subsumed into "deep learning" is unnecessary because "word2vec" is subsumed by "deep learning" and "deep learning" is subsumed by "machine learning".
- *Constraint 3:* synonymous conflict constraint. if concept *a* and concept *b* are synonymous, they must not be subsumptive. Therefore, such relations like "Ordered binary tree"→"Binary search tree", or "Heap"→"Heap" would be removed in this constraint.

These three constraints are very important for ensuring the fine quality of the data output in each iteration, because constraint 1 and 2 contribute to avoiding pulling in the incorrect instance, and constraint 3 helps cast off the redundancy. This mechanism optimizes the result and guarantees our approach to learn more relations with fine granularity.

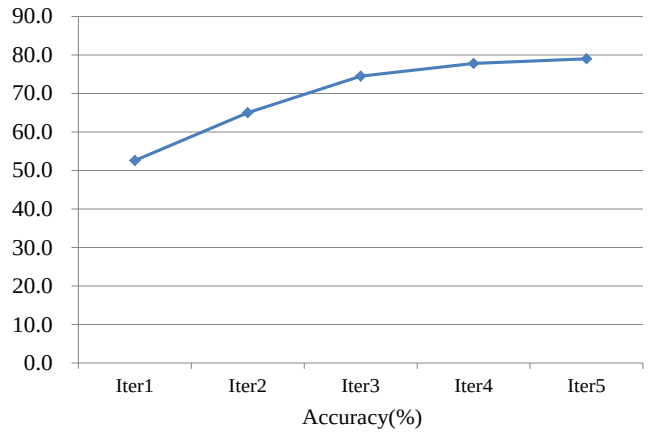

Fig. 2: Accuracy tendency for each iteration

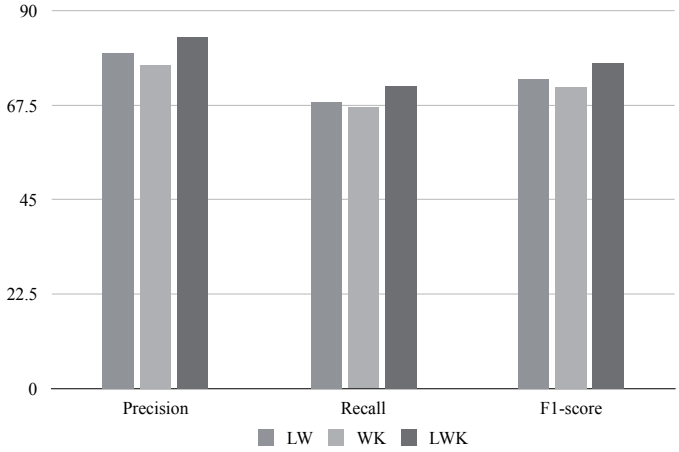

Fig. 3: Comparison of different feature groups

TABLE I: Result sets of three models

Models	Relation numbers	Accuracy
Decision Tree	333,659	77.3%
Naive Bayes	351,457	78.1%
SVM	357,662	79.7%

IV. Experiment

A. Experiment Setup

1) Data handling: some necessary data are processed in our approach. During the data extraction in SE concept extraction activity, we totally obtain 193,593 software engineering domain concepts. And in the data labeling activity, we finally generate 4,181 labeled training data, which includes 2,018 positive data and 2,163 negative data, and these labeled data are used for training the classifiers.

2) Accuracy evaluation: as for the accuracy evaluation in our experiment, we mainly use manual judgment to finish this work. For the classification results in each iteration, we invited 5 partners in our laboratory to join the accuracy calculation work. We randomly select about 10,000 relations extracted by the certain classifier in each iteration, and assigned approximately 2,000 relations to each member. We require them to judge the exactness of these relations by labeling "Yes", "No" or "Uncertain" for each relation. After finishing each part of works, we gather all the results and estimate the prediction accuracy of classifiers.

3) Semi-supervised learning: for assuring the stability of extraction in semi-supervised, we implement three models for information processing, which are decision trees, naive Bayes and SVM. These models extract relations and get three different result sets, and we select the best one as our final result and put it into taxonomy with the RDF format.

B. Result Analysis

1) Accuracy evaluation of semi-supervised learning: we do the accuracy evaluation of semi-supervised learning in order to check the effect of several iterations and the constraints mentioned in III(E). As for the used three classifiers, we separately calculate the accuracy of each algorithm. And in every iteration, we get the average accuracy of three models at current iteration, as the accuracy value. In this way, we ensure the stability of evaluation. The tendency chart of precision is Fig 2.

2) Results analysis of three models: after the whole semi-supervised process, the final results extracted by three models are shown in Table I. We could find the results trained by SVM is considered the best, and we use SVM results as our final relationship set which contains 357,662 subsumptions with 79.7% accuracy.

C. Feature contribution analysis

We design the experiment to analyze whether these features are helpful for the relationship prediction. In the experiment, all the features first are divided into three classes: *Lexical Features*, *Wiki-based Features*(including *Co-occurrence-based Features* and *Structure-based Features)*, and *KL-Divergence Feature*. Then, we combine these classes into three groups. The first group consists of *Lexical Features* and *Wiki-based Features*(donated as *LW*), and the second group includes *Wiki-based Features* and *KL-Divergence Feature*(donated as *WK*), for the third group, we use all the features(donated as *LWK*). As for the three groups, we trained the three SVM classifiers separately and get three different results. The precision, recall and F1-scores is been calculated for the three result sets and we make comparisons in Fig 3.

The result shows *LWK* performs best, the values of precision, recall and F1-score are higher than *LW* and *WK*, which means all the features have an effect on subsumption extraction. Compared with *WK*, the *LW* performs better. It mainly because *Lexical Features* is adept at finding overt subsumptions which contain semantics relations, and the *Lexical Features* could predict the relation when *Wiki-based Features* are symmetry. On the contrary, *KL-Divergence Feature* aims at finding the crytic subsumptions, it could be small amount and lower confidence compared with *LW*.

TABLE II: Comparison with other works

	Ours	Software. zhishi.schema	SE subset of		
			YAGO	WikiTaxonomy	Schema.org
Concept Number	**193593**	38205	898	711	10
Subsumption Number	**357662**	68098	870	630	0
Maximun Depth	**31**	28	3	6	1
Minimum Depth	**1**	1	2	1	1
Average Depth	**7.02**	6.99	2.24	1.39	1.00

D. Taxonomy Comparison

Because of the absence of public software engineering taxonomy, we first take the software engineering taxonomy built by Zhu et al. [13] into consideration, the result set is called Software.zhishi.schema. Some general knowledge bases are also imported into our experiment, such as Yago Taxonomy, WikiTaxonomy, and Schema.org. We compare the SE domain subset of above works, and provide the indicators such as concept number, subsumption number, maximum depth, minimum depth and average depth for each collection in order to illustrate the granularity and richness.

From the Table II, we can see in our taxonomy, the maximum depth reaches 31 and the average depth is 7.02, which contains the larger concept and subsumption numbers than other works.

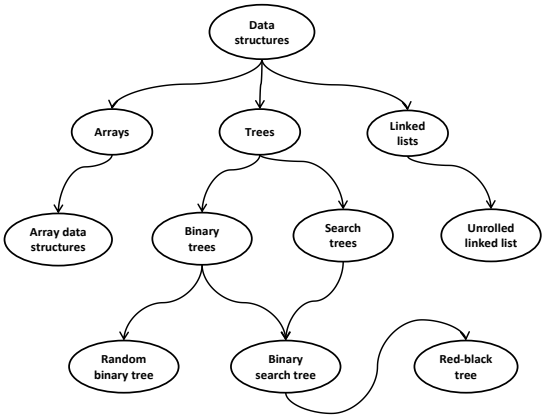

Fig. 4: A small fragment of our taxonomy

We capture a small fragment of our taxonomy tree as the demonstration, and shown in Fig 4, some unnecessary nodes are ignored. It reveals the granularity and hierarchy of our result set in this figure. The complete taxonomy has been deployed on our website.

V. Conclusion

In this paper, we propose an approach to discover the subsumptions between SE concepts from Wikipedia. The approach collects domain tags in StackOverflow as seed words, extracts concepts in Wikipedia, and uses machine learning algorithms to extract subsumption relations. We launch multi-dimension features to improve the training precision. As a result, we build the taxonomy which contains 193,593 concepts and 357,662 subsumption relations with the format of RDF. The experimental results show the large-scale and high accuracy of our dataset.

For the future work, we will try to use other datasets to support the current work, because we find it still contains increasing space of scale if the entity invalidations of Wikipedia can be thoroughly solved. It requires implementing the mapping mechanism of entities between Wikipedia with other datasets.

VI. Acknowledgement

This research is supported by 973 Program in China (Grant No. 2015CB352203) and National Natural Science Foundation of China (Grant No. 61472242).

References

[1] Simone Paolo Ponzetto, Michael Strube:WikiTaxonomy: A Large Scale Knowledge Resource. ECAI 2008: 751-752.
[2] Soren Auer, Christian Bizer, Jens Lehmann, Georgi Kobilarov, Richard Cyganiak, and Zachary Ives, DBpedia: A nucleus for a Web of open data, in Proc. of ISWC 2007 + ASWC 2007, 722735, (2007).
[3] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence, pages 163,(2012).
[4] Michael Strube, Simone Paolo Ponzetto: WikiRelate! Computing Semantic Relatedness Using Wikipedia. AAAI 2006: 1419-1424.
[5] Budanitsky, A. & G.Hirst. Evaluating WordNet-based measures of semantic distance. Computational Linguistics 2006, 32(1).
[6] Giriprasad Sridhara, Emily Hill, Lori Pollock, and K Vijay-Shanker. Identifying word relations in software: A comparative study of semantic similarity tools. In ICPC 2008, pages 123132. IEEE, 2008.
[7] Rongxin Wu, Hongyu Zhang, Sunghun Kim, and Shing-Chi Cheung. Relink: recovering links between bugs and changes. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pages 1525. ACM, (2011).
[8] Fei Wu and Daniel Weld, Automatically refining the Wikipedia infobox ontology, in Proc. of WWW-08, (2008).
[9] Xing Niu, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, Yong Yu: Zhishi.me - Weaving Chinese Linking Open Data. International Semantic Web Conference (2) 2011: 205-220
[10] Thomas Lin, Mausam, and Oren Etzioni. No noun phrase left behind: Detecting and typing unlinkable entities. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 893-903, Jeju Island, Korea, July.
[11] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In SIGMOD 2012, pages 481-492. ACM, 2012.
[12] J-R Falleri, Marianne Huchard, Mathieu Lafourcade, Clementine Nebut, Violaine Prince, and Michel Dao. Automatic extraction of a wordnet-like identifier network from software. In Program Comprehension (ICPC), 2010 IEEE 18th International Conference on, pages 413. IEEE, (2010).
[13] Jiangang Zhu, Beijun Shen, Xuyang Cai, Haofen Wang:Building a Large-scale Software Programming Taxonomy from Stackoverflow. SEKE 2015: 391-396
[14] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th conference on Computational linguistics-Volume 2, pages 539-545,(1992).
[15] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In Proceedings of the ACL, (2003).
[16] Wu Z, Palmer M. Verbs semantics and lexical selection. Proceedings of the 32nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1994: 133-138.
[17] Solomon Kullback. Information theory and statistics. Courier Corporation, (1997).
[18] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation. the Journal of machine Learning research, 2003(3): 993-1022.