Building a Domain Knowledge Base from Wikipedia: a Semi-supervised Approach

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

***Abstract*—Knowledge bases are becoming indispensable to software engineering and knowledge engineering. However, the existing domain knowledge bases are always artificially constructed and small-scale. In this paper, we propose a semi-supervised approach to domain concepts detection and software engineering knowledge base construction from Wikipedia. First, the approach selects domain relevant tags from Stackoverflow. Then, it matches Wikipedia entities and expands the concept set through an improved label propagation algorithm. A rule-based method is designed to discover semantic relations including `relate`, `subclassOf` and `equal` by analyzing structural information of Wikipedia. A relation derivation mechanism is presented to optimize the relation set. We finally construct SEBase, a domain-specific knowledge base of software engineering. Experimental results show the high accuracy of the integrated concepts and relations. Compared with other knowledge bases, SEBase has the widest coverage of concepts and relations in software engineering.**

***Index Terms*—Knowledge Base; Software Engineering; Semi-supervised; Domain Concept; Semantic Relation**

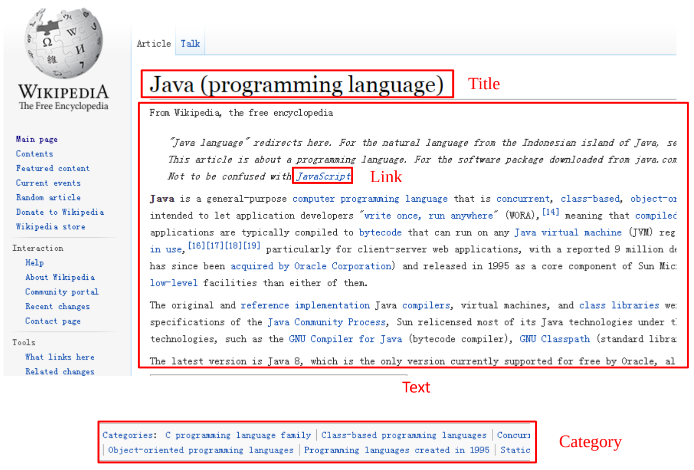
I. Introduction

Knowledge bases play an important role in software engineering and knowledge engineering. For example, in program comprehension, knowledge bases are used to compute the semantic similarities between words from the comments and identifiers in software [1]. In software maintenance, knowledge bases provide an effective way to measure the relatedness between documents [2].

Research on general knowledge bases has been quite mature. Some famous achievements like DBpedia [3], Yago [4], WikiTaxonomy [5], BabelNet [6] and Probase [7] are proposed in recent years. A large number of concepts and relations with high accuracy are detected in those knowledge bases. However, the knowledge in them is not specific and not in-depth enough when we focus on some particular domains like software engineering.

Building a domain-specific knowledge base is a difficult task requiring skills in logic and ontological analysis. Domain experts need to determine the scope of the domain concepts and construct relations for them. However, manual approaches are time-consuming and small-scale. Although there has been a considerable amount of prior research on automatic construction of software engineering knowledge base, a high-quality knowledge base is still lacking because learning knowledge from any single data source cannot achieve desirable effect and domain concepts are difficult to distinguish.

Knowledge learning can be encyclopedic-based or web-based. For the encyclopedic-based approaches, researchers mainly focus on Wikipedia for its abundant structural information. Wikipedia is the most comprehensive and authoritative knowledge source in the world, which has a total of 17 million entries composed of *title*, *redirect title*, *comment*, *text* and other information.



Java (programming language)

From Wikipedia, the free encyclopedia

"Java language" redirects here. For the natural language from the Indonesian island of Java, se
This article is about a programming language. For the software package downloaded from java.com
Not to be confused with JavaScript.

Java is a general-purpose computer programming language that is concurrent, class-based, object-or
intended to let application developers "write once, run anywhere" (WORA), [14] meaning that compiled
applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) reg
in use, [16][17][18][19] particularly for client-server web applications, with a reported 9 million de
has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Mic
low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries we
specifications of the Java Community Process, Sun relicensed most of its Java technologies under t
technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libra

The latest version is Java 8, which is the only version currently supported for free by Oracle, al

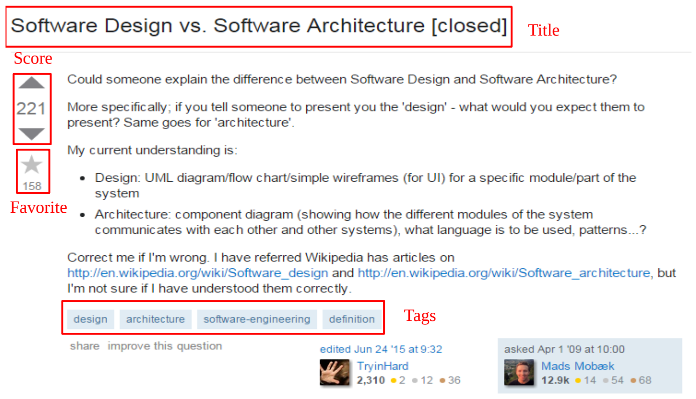
Categories: C programming language family | Class-based programming languages | Concurr
| Object-oriented programming languages | Programming languages created in 1995 | Static

Fig. 1: An example from Wikipedia

For web-based approaches, general knowledge bases learn knowledge from kinds of web pages but domain knowledge bases only consider a part of relevant web pages. In recent years, researchers pay more attention to Stackoverflow. Stackoverflow is one of the most famous QA websites about software engineering and provides a tagging system for users to annotate questions freely. The tags of Stackoverflow are more relevant to software engineering.

Generally speaking, encyclopedic-based approaches can achieve higher accuracy but worse coverage than that of web-based approaches. We select Wikipedia as our main knowledge source and select Stackoverflow as supporting source to obtain the advantages of both. Joorabchi et al. [8] provided a mapping method based on machine learning to interlink Stackoverflow tags with Wikipedia concepts. Unlike their approach, we detect the Wikipedia entities which have high relevance to Stackoverflow tags by matching and expansion methods.

DOI reference number: 10.18293/SEKE2016-051



Software Design vs. Software Architecture [closed]

221

158

Could someone explain the difference between Software Design and Software Architecture?

More specifically; if you tell someone to present you the 'design' - what would you expect them to present? Same goes for 'architecture'.

My current understanding is:

- Design: UML diagram/flow chart/simple wireframes (for UI) for a specific module/part of the system
- Architecture: component diagram (showing how the different modules of the system communicates with each other and other systems), what language is to be used, patterns...?

Correct me if I'm wrong. I have referred Wikipedia has articles on http://en.wikipedia.org/wiki/Software_design and http://en.wikipedia.org/wiki/Software_architecture, but I'm not sure if I have understood them correctly.

design architecture software-engineering definition

share improve this question

edited Jun 24 '15 at 9:32
TryinHard
2,310 ●2 ●12 ●36

asked Apr 1 '09 at 10:00
Mads Mobæk
12.9k ●14 ●54 ●68

Fig. 2: An example from Stackoverflow

The problem is non-trivial and poses unique technical challenges as follow:

1) Since most of the Stackoverflow tags are domain relevant, they are provided by ordinary users freely. A large number of low quality tags exist. If domain relevance of tags cannot be ensured, the accuracy of final results cannot meet our expectations definitely. Therefore, how to ensure the quality of tags remains to be solved.
2) Traditional matching methods are difficult to interlink Stackoverflow tags and Wikipedia entities. For example, tag "java" and entity "java man" seem to be related but it is unreasonable to treat the latter as a software engineering concept.
3) Although a part of domain concepts can be detected by matching method, more concepts cannot be discovered because tags are not comprehensive enough. For example, "Ocaml" is a programming language but no tag can match it. So how to detect domain concepts which have low relevance to tags is also a challenging problem.

In order to solve challenges above, we decompose the problem into two sub-problems - ***domain concepts detection*** and ***semantic relations discovery***. For the first sub-problem, we select domain relevant tags from Stackoverflow, and make head matching with Wikipedia entities to obtain a part of domain concepts. Then, we treat Wikipedia as a large knowledge network, and propose a semi-supervised method based on an improved label propagation algorithm to acquire domain network of software engineering. For the second sub-problem, we discover three types of semantic relations including `relate`, `subclassOf` and `equal` by analyzing structural information of Wikipedia. We also propose a relation deriving method to delete mistaken and redundant relations.

To the best of our knowledge, our work is the first to build a software engineering knowledge base by interlinking Stackoverflow tags and Wikipedia entities. Our contributions mainly include:

- We systematically explore the knowledge provided by Wikipedia and Stackoverflow, and leverage a series of methods including tag selection, head matching and label propagation to detect domain concepts.
- We analyze structural information of Wikipedia, and discover three types of semantic relations from Wikipedia. A derivation mechanism is designed to optimize the relation set.
- We carry out a comprehensive set of experiments to evaluate our approach. The results show our approach can outperform the several existing knowledge bases in terms of accuracy and coverage significantly. Thus, we publish SEBase on https://datahub.io/dataset/se-base which will benefit many applications in software engineering.

II. Related Work

A. Knowledge Base Construction

Knowledge base construction has aroused extensive attentions in research community. The construction methods can be encyclopedic-based or Web-based. For the encyclopedic-based methods, researchers mainly focus on extracting concept hierarchies from Wikipedia. DBpedia [3] is a crowd-sourced community effort to extract structured information from Wikipedia. WikiTaxonomy [5] builds a taxonomy from the Wikipedia category system. Yago [4] and BabelNet [6] both interlink Wikipedia entities to WordNet [9] synsets.

Regarding Web-based methods, it can be free text based or social tag based. For the free text based methods, Probase [7] builds the largest taxonomy which contains over 2.7 million classes from 1.7 billion Web pages. For social tag based methods, Xiance Si et al. [10] estimated the conditional probability between tags and designed a greedy algorithm to eliminate the redundant relations. Zhishi.schema [11] is the first achievement to learn knowledge from tags and categories in popular Chinese websites.

B. Software Engineering Knowledge Base

Software engineering knowledge base is a type of domain knowledge base. However, there are only a limited number of researches which are related to software engineering knowledge base. Kavi Mahesh et al. [12] proposed LOaD-IT, a concept network to help software developers read technical documents faster. Mr.Izzeddin A.o. el at. [13] constructed a programming language ontology. Software.zhishi.schema [14] uses tags of Stackoverflow build a software engineering knowledge base. We broaden the scale of concepts based on Software.zhishi.schema by interlinking Stackoverflow tags to Wikipedia entities.

C. Set Expansion

The most challenging part to build software engineering knowledge base is domain concepts detection. Set expansion is the most effective method to discover concepts. It can be text-based and graph-based. For the text-based approaches, Richard C. Wang et al. [15] proposed a language-independent set expansion method based on pattern selection. KnowItAll [16] is famous for its high effect. It extracts information from the web and induce the rule templates. Regarding graph-based approaches, label propagation algorithm(LPA) is widely used and it is a semi-supervised method. Jierui Xie et al. [17] [18] focused on community detection using a neighborhood strength

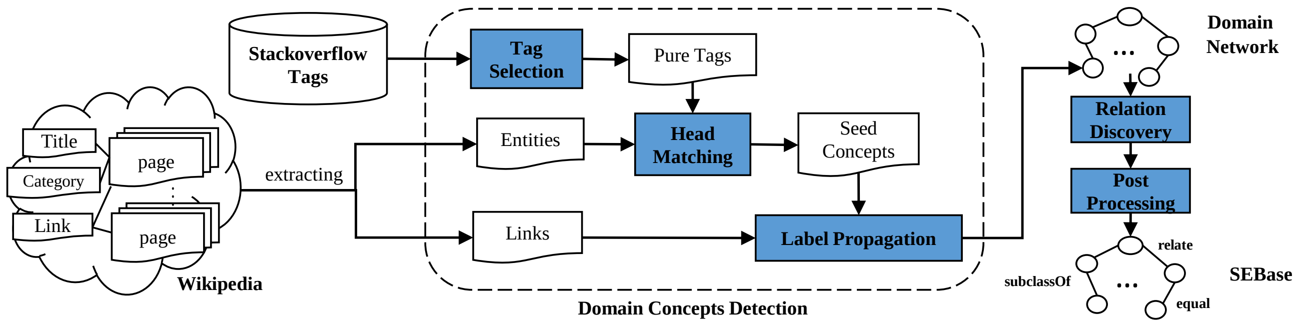


Fig. 3: Our approach to SEBase construction

driven LPA. Liu et al. [19] designed an improved LPA in large-scale bipartite networks to detect community. In this paper, we propose a Wikipedia-based LPA called WLPA to detect community of software engineering.

III. Approach

In this section, we describe our proposed approach in detail.

A. Approach Overview

We propose a semi-supervised approach to software engineering knowledge base construction from Wikipedia. As shown in Figure 3, it consists of five main components, namely *Tag Selection*, *Head Matching*, *Label Propagation*, *Relation Discovery* and *Post Processing*. *Tag Selection* tries to select software engineering tags from Stackoverflow. *Head Matching* matches tags with Wikipedia entities to obtain a part of domain concepts. *Label Propagation* leads to discover a domain network including concepts and relations. *Relation Discovery* is to extract three types of relations from the domain network. *Post Processing* is to delete mistaken and redundant relations by three type of relation deductions. Finally we build a domain knowledge base originated from Stackoverflow tags and composed of Wikipedia concepts and relations.

B. Tag Selection

Since tags in Stackoverflow are provided by normal users freely, the domain relevance of tags is unreliable. For example, “music” and “jave” are in the tag set, but the former is domain irrelevant and the latter is a wrong writing of “java”. A selection is necessary to delete domain irrelevant and mistaken tags. The standards of selection are based on two phenomenons. The first is that the tags from questions with high *voting score* and *favorite number* are more domain relevant. This phenomenon can be interpreted as that the users always pay more attention to software engineering related questions. Another truth is that the tags with high frequency are more reliable than tags with low frequency. We select tags that have high occurrence frequency and extract them from questions with high concern rate. The selection method is simple but effective. The details of the selection are as follows:

1) We select questions which have the top 10% *voting score* and the top 10% *favorite number*.
2) We select tags from those questions and make a ranking by occurrence frequency .
3) We select the top 30% tags.

C. Head Matching

The purpose of head matching is to detect Wikipedia entity which has high relevance with the selected tags. The details of the matching are as follows:

1) We first unify the format of tags and entities. All letters are transformed into lowercase. A part of tags and entities are split by “-” and the others are split by spaces. We replace underscores and hyphens by spaces. Many tags and entities contain a version number, and we delete it directly because it does not affect semantic meaning. For our running example, “machine-learning” is transformed to “machine learning”, “Visual Studio 2012” is transformed into “visual studio”.
2) For each tag and entity, we select its headword as matching standard. We use a simple but effective rule to extract headword: If the tag or entity is a single word, headword is itself. Otherwise, if it contains prepositions such as “of”,“in” or “for”, then headword is the previous word of the preposition. If not then headword is the last word. For example, we extract “sort” from “heap sort” and “architecture” from “logic architecture of hadoop”. It is worth mentioning that some Wikipedia entities such as “java (programming language)” have label. We extract headword from its label. In addition, we use StandfordNLP tool to stem the headword. For some running examples, “programming” is transformed into “program”, “algorithms” is transformed into “algorithm”.
3) For each Wikipedia entity, we check if there are one or more tags with same headword. Once the requirement is met, we confirm the entity is a software engineering concept. In order to compare faster, we establish inverse index on headword. We compute digital fingerprint of headword as index.

D. Label Propagation

The head matching only captures the semantic relation between software engineering concepts in an explicit way, however it cannot detect the implicit relations between them. For example, “Ocaml” is a programming language but no tag has sematic relation with it. So “Ocaml” cannot be detected by head matching, but it is exactly a domain concept. In order to solve this problem, we propose a set expansion method by leveraging structural information to detect the implicit relations between domain concepts.

If we treat `category` and `redirect title` as types of `link`, Wikipedia can be regarded as a large knowledge network composed of entities and links. The linked entity is related to current entity. Correlation degree is proportional to the time of links. We extract links from Wikipedia page and use them to quantitatively characterize the similarities and relatedness between entities. There are more links between concepts in the same domain than that in different domains. In order to expand the domain concepts set, we develop an improved label propagation algorithm (LPA) called WLPA. The details of the algorithm are shown in Table I.

TABLE I: DESCRIPTION OF WLPA ALGORITHM

```
Algorithm: WLPA
Input: seed concepts set $S$, Wikipedia entities set $W$
       Wikipedia link set $L$
Procedure:
 1: Construct a knowledge network $G(V, E)$.
    $G$ is directed graph, $V = W$
    $edge(u, v) = \infty$ if node u and v connect by redirect title
    $edge(u, v) = o$ if node u and v connect by category
    $edge(u, v) = l(u, v)$ if node u and node v connect by link
    where $l(u, v)$ equals to the number of link from node u to v
    $\infty \gg$ o $\gg$ max$\{l(u, v)\}$
 2: Initialize nodes with unique labels.
    $\forall$n$\in S$, $c_n = l_Y$
    $\forall$n$\in V\backslash S$, $c_n = l_N$
 3: Update the label of nodes.
    $\forall$n$\in V\backslash S$ $c_n = l_Y$
    if $\exists$ $u$, $c_u = l_Y$, $e(u, n) = \infty$ or $e(n, u) = \infty$ or
    $\sum e(u, n) > \sum e(v, n)$ or $\sum e(n, u) > \sum e(n, v)$
    where $c_u = l_Y$, $c_v = l_N$, $e$ != $\infty$
    $c_n = l_N$ else.
 4: If not converged, continue to 3.(The convergence always be guaranteed
    because in each iteration, transforming label $l_Y$ into $l_N$ is impossible, due
    to no more nodes with label $l_N$ generated. The iterations depends on the
    richness of seed concepts. It typically requires 5 or 6 iterations. )
 5: Return final label set $\{c_n\}$
Output: labeled network $G$
```

The edges in knowledge network G can be divided into three types that *redirect title* weighs ∞, *category* weighs o and normal link weighs the number of links. At the initial step, we set nodes of seed concepts with stationary label l_Y. Those nodes would never change its label because they are detected by head matching and we treat them as "seed". At the propagation step, we derive the label of nodes by comparing the sum of weight between l_Y neighbors and l_N neighbors, where l_Y and l_N indicate that the node is a domain concept or not. The weight of in-edges and out-edges are respectively computed. A propagation running example is shown in Figure 4.

The difference between WLPA and original LPA [20] mainly includes:

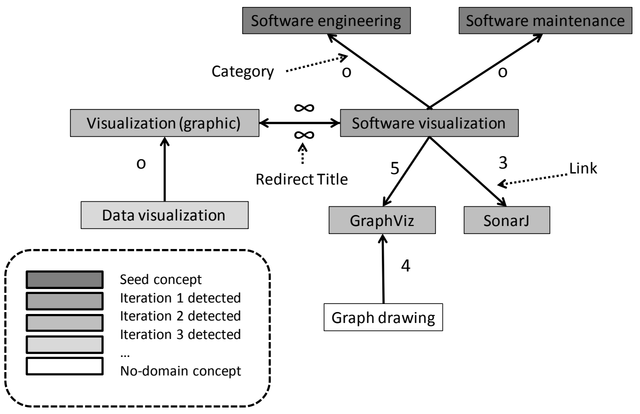


Fig. 4: A running example of label propagation

- Links in Wikipedia knowledge network are directed so we construct a directed graph for WLPA. But original LPA works on undirected graph.
- Links between two entities can be tight or loose. We show this feature by giving edges times of link as weight. But edges of LPA graph do not have any weight.
- In order to ensure the convergence, WLPA sets stationary label for seed concepts.

All of the above changes make WLPA more appropriate for Wikipedia knowledge network. Finally, WLPA generates a domain network composed of software engineering concepts.

E. Relation Discovery

In this subsection, we discover semantic relations from domain network. Three type of relations including `relate`, `subclassOf` and `equal` are extracted. The edges of domain network are composed of Wikepedia structure relations including *link*, *category* and *redirect title*. They are manually constructed by Wikipedia entity contributors. We put those high accuracy relations into our relation sets. The conversion rules are as follows:

TABLE II: STRUCTURE BASED CONVERSION RULES

Situation	Relation
A is B's "Redirect Title"	A `equal` B,B `equal` A
A is in B's "Category" set	B `subclassOf` A

In addition, We use Normalized Google Distance [21] to calculate the degree of correlation between two concepts and construct `relate` relation. Specifically, the Normalized Google Distance (NGD) between two concepts A and B is:

$$\texttt{NGD (A,B)} = \frac{max\{logf(A), logf(B)\} - logf(A, B)}{logN - min\{logf(A), logf(B)\}} \quad (1)$$

where *N* is the total number of edges; f(A) and f(B) are the number of out-edge of concepts A and B, respectively; f(A, B) is the number of nodes that both A and B out-connected.

In order to control the number of this kind of relations, we select five of the most relevant concepts for a concept.

F. Post Processing

We provide a derivation mechanism to delete mistaken or redundant relations. The relation deduction rules are shown in Table III, where A=B means that there is an `equal` relation between A and B, A≃B means that there is a `relate` relation from A to B, A→B means that there is a `subclassOf` relation from A to B.

TABLE III: DERIVATION RULES

Situation	Type	Operation
A→B,B→C,A→C	Transitive Redundancy	Delete (A→C)
A=B,A≃B	Synonym Conflict	Delete (A≃B)
A=B,A→B	Synonym Conflict	Delete (A→B)
A→B,A≃B	Synonym Conflict	Delete (A≃B)

IV. EVALUATION

In the version of the data we downloaded, Stackoverflow has 38,205 tags while Wikipedia has about 17,000,000 entities. Each entity has an average of 45.2 links, 1.8 categories and 0.23 redirect titles. We finally obtain 128,674 concepts and 607,341 relations about software engineering. In this section, we show the experimental results of our proposed approach. We mainly focus on accuracy and coverage of concepts and relations in SEBase.

A. Selected Tag Quality Evaluation

We first evaluate the quality of selected Stackoverflow tags. The accuracy evaluation has to be done by persons manually because of semantic comprehension required. However, due to the large number of concepts and relations, it is impossible to evaluate all of them by hand. Therefore, we design a random sampling strategy and a labeling process. We manually annotate some targets of sampled concepts or relations. The accuracy assessment on the sampled subset can further be used to approximate the correctness of the whole set. Three students from our laboratory are invited to participate in the labeling process. We provide them with three choices namely *agree*, *disagree* and *unknown* to label each sample. Then we can compute the average accuracy.

We generate four tag sets for comparison. They are original set, question selected set (QS), frequency selected set (FS), question and frequency selected set (QS+FS). We randomly extract 500 tags from four tag set as samples. Students annotate each tag independently. Then, we select 1,000 high correlated tags from original set to compute the coverage of four tag set. Because the tags are treated as seed, we consider the precision is more important than the recall. $F_{0.5}$ score is used to compare comprehensive quality of four sets. The results are evaluated in terms of precision, recall and $F_{0.5}$, as shown in Figure 5 where we can confirm the necessity of *Tag Selection*.

After question selection and frequency selection, the $F_{0.5}$ score achieves 92.82%.

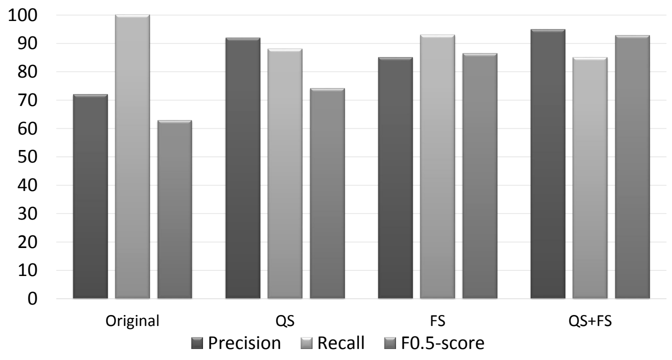


Fig. 5: Tag quality evaluation

B. Seed Concepts Quality Evaluation

This experiment is to evaluate the performance of *Head Matching*. We generate three matching results for comparison. They are original result, head matching result and stemmed result. Original result is generated by matching tags and entities directly. Head matching result is generated by matching headwords. Stemmed result is generated by using stemmed headwords. The sampling is similar to tag quality evaluation. The results are evaluated in terms of precision, recall and F1-score, as illustrated in Figure 6.

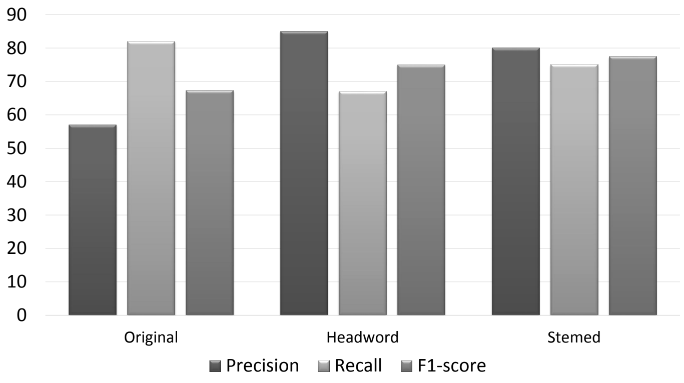


Fig. 6: Seed concepts quality evaluation

The evaluation shows the necessity of *Head Matching*. The precision of directly matching is less than 60% despite the recall is high. Head matching improves precision to 82.4%. The stemming operation balances precision and recall, and therefore the F1-score reaches 77.42%.

C. Final Results Evaluation

Next we evaluate the final concept and relation results. The sampling of concepts is similar to that of the evaluation above. For relations, we randomly extract 500 `equal`, 500 `relate` and 500 `subclassOf` relations as samples. Students label each relation independently. Then, we select same numbers of correct relations from Wikipedia to compute the coverage. The evaluation results are shown in Figure 7.

For concepts, our approach achieves precision of 76.7%, and recall of 92.4%. For relations, our approach achieves expected precision but unsatisfactory recall. This is because we mainly focus on domain concept detection and discover relation from

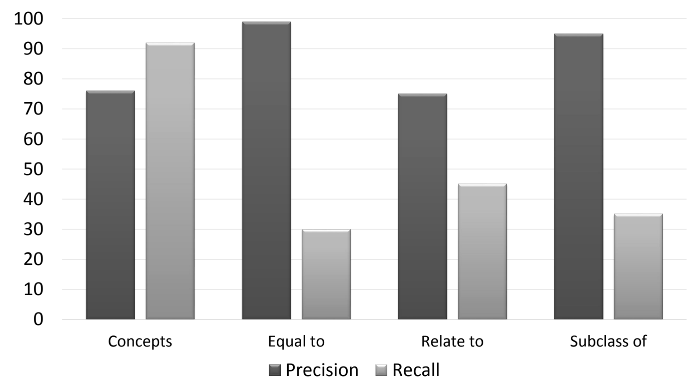


Fig. 7: Final results evaluation

Wikipedia structure. However, only 27% of Wikipedia entities have *category* and 23% of entities have *redirect title*.

D. Comparison with Other Datasets

Since there are no published famous software engineering knowledge bases, we compare SEBase with the subsets about software engineering extracted from other well-known general taxonomies namely Yago and WikiTaxonomy. We also compare SEBase with our previous work, Software.zhishi.schema (SZS), which is constructed from Stackoverflow. The comparison results are shown in Table IV.

TABLE IV: Comparison With Other Taxonomies

	SEBase	**SZS**	**Yago**	**WikiTaxonomy**
Concept Number	128,674	38,205	898	711
Concept Overlap	15,441	0	29	27
Subclass of	77,204	68,098	870	630
Equal to	15,441	0	29	27
Relate to	514,696	0	0	0

As for the relation number, SEBase is much larger than any other datasets. However, 76.2% relations of SEBase are `relate` and subsumption relations are relatively too few. Regarding to the granularity and richness of concepts, SEBase is more fine-grained than other existing datasets.

V. Conclusion and Future Work

In this paper, we build SEBase, a large-scale knowledge base of software engineering which contains 128,674 concepts and 607,341 relations. Our approach interlinks Stackoverflow tags and Wikipedia entities, and uses a semi-supervised WLPA to learn knowledge from Wikipedia. The experiments show the high quality of concepts and relations in SEBase.

For future work, we plan to improve SEBase from three aspects. We will try to extract more hyponymy relations based on classifier. On the other hand, we will attempt to deep learning technology to obtain purer tags. Moreover, it is would be interesting to improve matching method more effective by keyword extraction and topic model.

VI. Acknowledgement

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

References

[1] G. Sridhara, E. Hill, L. Pollock, and K. Vijay-Shanker, “Identifying word relations in software: A comparative study of semantic similarity tools,” in *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*, pp. 123–132, IEEE, 2008.

[2] R. Wu, H. Zhang, S. Kim, and S.-C. 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*, pp. 15–25, ACM, 2011.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[4] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” in *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, ACM, 2007.

[5] S. P. Ponzetto and M. Strube, “Wikitaxonomy: A large scale knowledge resource.,” in *ECAI*, vol. 178, pp. 751–752, 2008.

[6] R. Navigli and S. P. Ponzetto, “Babelnet: Building a very large multilingual semantic network,” in *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 216–225, Association for Computational Linguistics, 2010.

[7] W. Wu, H. Li, H. Wang, and K. Q. Zhu, “Probase: A probabilistic taxonomy for text understanding,” in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 481–492, ACM, 2012.

[8] A. Joorabchi, M. English, and A. E. Mahdi, “Automatic mapping of user tags to wikipedia concepts: The case of a q&a website–stackoverflow,” *Journal of Information Science*, p. 0165551515586669, 2015.

[9] G. A. Miller, “Wordnet: a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[10] X. Si, Z. Liu, and M. Sun, “Explore the structure of social tags by subsumption relations,” in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 1011–1019, Association for Computational Linguistics, 2010.

[11] H. Wang, T. Wu, G. Qi, and T. Ruan, “On publishing chinese linked open schema,” in *The Semantic Web–ISWC 2014*, pp. 293–308, Springer, 2014.

[12] K. Mahesh, A. Nagarajan, A. R. Balevalachilu, and K. Prasad, “Curating semantic linked open datasets for software engineering,” in *Proceedings of the 2013th International Conference on Posters & Demonstrations Track-Volume 1035*, pp. 61–64, CEUR-WS. org, 2013.

[13] I. Abuhassan and A. M. AlMashaykhi, “Domain ontology for programming languages,” *Journal of Computations & Modelling*, vol. 2, no. 4, pp. 75–91, 2012.

[14] J. Zhu, H. Wang, and B. Shen, “Software. zhishi. schema: A software programming taxonomy derived from stackoverflow,” in *14th IEEE International Semantic Web Conference (ISWC)*, IEEE, 2015.

[15] R. C. Wang and W. W. Cohen, “Language-independent set expansion of named entities using the web,” in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 342–350, IEEE, 2007.

[16] S. Schiff, “Know it all,” *The New Yorker*, vol. 31, no. 07, 2006.

[17] J. Xie and B. K. Szymanski, “Community detection using a neighborhood strength driven label propagation algorithm,” in *Network Science Workshop (NSW), 2011 IEEE*, pp. 188–195, IEEE, 2011.

[18] J. Xie and B. K. Szymanski, “Towards linear time overlapping community detection in social networks,” in *Advances in Knowledge Discovery and Data Mining*, pp. 25–36, Springer, 2012.

[19] X. Liu and T. Murata, “Community detection in large-scale bipartite networks,” in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT’09. IEEE/WIC/ACM International Joint Conferences on*, vol. 1, pp. 50–57, IET, 2009.

[20] K. Kothapalli, S. V. Pemmaraju, and V. Sardeshmukh, “On the analysis of a label propagation algorithm for community detection,” in *Distributed Computing and Networking*, pp. 255–269, Springer, 2013.

[21] R. L. Cilibrasi and P. Vitanyi, “The google similarity distance,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 3, pp. 370–383, 2007.