Constructing a Knowledge Base of Coding Conventions from Online Resources

Junming Cao, Tianjiao Du, Beijun Shen*, Wei Li, Qinyue Wu, Yuting Chen
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, Shanghai, China
{junmingcao, tjsoulshe, bjshen, li_wei, wuqinyue, chenyt}@sjtu.edu.cn

***Abstract*—Coding conventions are a set of coding guidelines used by software developers to improve the readability of source code, increase software maintainability, and promote the reuse of coding patterns. In this paper, we introduce *CCBase*, a knowledge base of coding conventions, that was constructed from online resources. Specifically, CCBase was constructed as follows. We designed the ontology of the coding convention domain, crawled data related to coding conventions from a variety of online resources, and then extracted entities and relations using an NLP-enabled rule matching method. To uncover the latent relations, we further proposed a similarity metric to reveal the similar-to and relate-to relations, and developed a RCE algorithm to establish a unified type hierarchy of coding conventions. The resulting knowledge base contains 3139 coding conventions for Java and C++, with 3761 entities and 767 relations. Furthermore, we have extended the usability of CCBase by developing a question answering system on the base. We have conducted experiments to evaluate CCBase. The experimental results show that CCBase has a wide coverage on entities and relations in coding conventions domain, and the QA system achieves an F1 score of 84.5% on 214 questions raised in StackOverflow.**

Keywords - Knowledge Base; Coding Convention; Type Hierarchy; Question Answering

I. INTRODUCTION

Coding conventions are a set of guidelines for a particular programming language that recommend programming styles, practices, and methods for each aspect of a program written in that language. During increasingly large and complex software development, programmers are strongly encouraged to follow these guidelines to help improve the readability, reliability, and maintainability of their source code [1]. These coding conventions can also assist code related software engineering activities, like auto-detection of code bad smells [2] and code analysis [3].

However, programmers now encounter the following problems when applying coding conventions. One is that coding conventions specified in single document are incomplete because it could hardly cover a • ll coding details, and also the relations between coding conventions could not be expressed explicitly. The second problem is that coding conventions are inconvenient to access. Programmers need to know relevant keywords to search using a search engine like Google or search in documents, which is especially difficult for some novice programmers who lack professional knowledge.

In order to solve the above problems, we construct a coding conventions knowledge base, CCBase. CCBase is a domain-specific knowledge base, which is constructed from online resources using a top-down approach. Specifically, we first design the ontology of coding conventions domain. Then we collect data related to coding conventions from various online resources and extract entities and relations with an NLP-enabled rule matching method. The main challenge is to discover latent relations between coding conventions, including similar-to, relate-to, and especially subsumption relations, from these heterogeneous textual documents. So we designed a similarity metric to discover similar-to and relate-to relations, and propose the RCE (Relation based Cluster Expansion) algorithm to establish a unified type hierarchy of coding conventions and assign types of each coding convention. Finally, we develop a question answering system over CCBase to answer natural language questions automatically. CCBase, its SPARQL interface, and QA system can be accessed in our online platform[1].

Our main contributions are summarized as follows:

1) To our best knowledge, CCBase is the first knowledge base of coding conventions. It contains 3139 coding conventions of Java and C++, 3761 entities and 767 relations.

2) We propose the RCE algorithm to establish a unified type hierarchy of coding conventions. Structures of online resources entail original type hierarchies for coding conventions. However, some coding convention resources lack a hierarchy. The hierarchy extracted from one document is usually unilateral, and also different from another extracted hierarchy. Besides, every coding convention only has one type value with the original type hierarchy, which is also not comprehensive. Therefore, a novel unsupervised algorithm (RCE) is designed to build a unified type hierarchy according to the similar-to relations and assign new type values to coding conventions.

3) We develop a coding convention question answering system over CCBase, CCQA. The main algorithm of CCQA is subgraph matching, and we make two significant improvements to this algorithm. First, the entity linking method is changed to identify the entities regarding coding conventions in the question. Second, we collect common question templates and recognize

DOI reference number: 10.18293/SEKE2019-123
* Corresponding author
[1] http://202.120.40.28:4463/

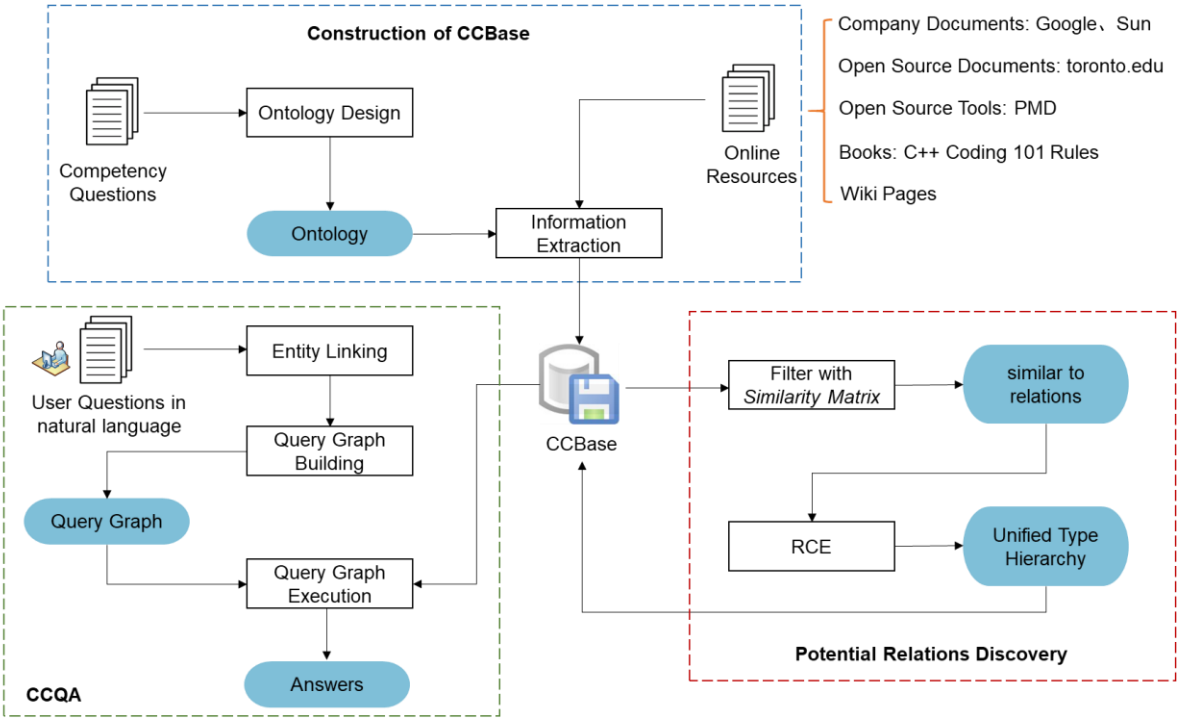

Figure 1. Overview of Our Approach

these templates from user questions, which improves the accuracy of subgraph building

4) A set of comprehensive experiments has been carried out to evaluate CCBase. The results show, CCBase is larger and more hierarchical than existing knowledge bases regarding code conventions; and our QA system achieves an F1 score of 84.5% on 214 questions raised in StackOverflow.

II. Related Work

A. Construction of Knowledge Base

There are two ways to construct a knowledge base: top-down and bottom-up. Top-down means pre-defining the ontology of a knowledge base, and then importing entities and relations according to the ontology into the knowledge base. Knowledge bases of specific domains mostly adopt this way [5]. Bottom-up means directly obtaining entities and relations by syntactic analysis without ontology, which is common for general knowledge bases [6]. These two ways include similar steps such as information extraction and knowledge fusion.

In the software engineering domain, a few researchers have tried to build a domain knowledge base, like SEBase [5] and APIBase [7]. However, to our best knowledge, there is no published work on coding convention knowledge base.

B. Type Hierarchy Building

Types are common in knowledge bases to organize entities, and type hierarchy is their key knowledge or meta-knowledge. [8] proposed an entity-driven approach to construct type hierarchy of knowledge base systems without hierarchy structures. The type hierarchy construction problem is similar to the community detection problem. Semi-supervised algorithms, like LPA [9], are widely used in community detection, and [10] proposed SLPA to deal with overlapping communities. However, these methods aren't suitable for our work, because our relations in CCBase are too sparse to propagate labels from a few seed entities, and structures of documents are very helpful to build the type hierarchy. Thus in this paper, we propose a novel unsupervised algorithm (RCE) by fully utilizing structures of documents.

C. Question Answering over Knowledge Base

Some research effort has been conducted to KBQA (Question Answering over Knowledge Base) systems [12][13], which led to major advances. So far there exist two mainstreams of KBQA methods. One mainstream is semantic parsing. The main idea of this kind of solution is to translate the questions into logical forms such as query graph, then generate executable queries [12]. Information retrieval is another mainstream, which selects candidate answers directly and then ranks these answers by various approaches, such as deep learning [13].

Our work belongs to the first one. Since natural language is complex and ambiguous, semantic parsing usually requires multiple steps, like part-of-speech tagging and entity linking.

III. Construction of CCBase

We construct CCBase in a top-down way for the following reasons: 1) Bottom-up is difficult to meet the quality requirements of domain-specific knowledge base. 2) The complexity of entities and relations in the coding conventions domain is tractable enough to be designed in advance. 3) Entities and relations could not be automatically obtained by syntactic analysis, so ontology is necessary to guide the extraction of entities and relations.

The overall approach is shown in Fig. 1. We first design the ontology of CCBase, then extract the information from the semi-structured and unstructured data, and finally discover the latent relations between coding conventions.

A. *Ontology Design*

We collect massive coding conventions from various online resources, including coding conventions published online by companies, standards organizations, research groups and experts, coding conventions in open source tools, books, wiki pages, etc. The ontology is initialized from the investigation of these data. We use Protégé[2], an open source software developed by Stanford, to design the ontology.

Then we use the competency question-driven method to perform ontology improvement [14]. We select 30 competency questions from 214 coding convention related questions from StackOverflow, and improve the ontology until these questions could be answered with the ontology. The final ontology has 11 key concepts and 14 kinds of relationships, as shown in Fig. 2.

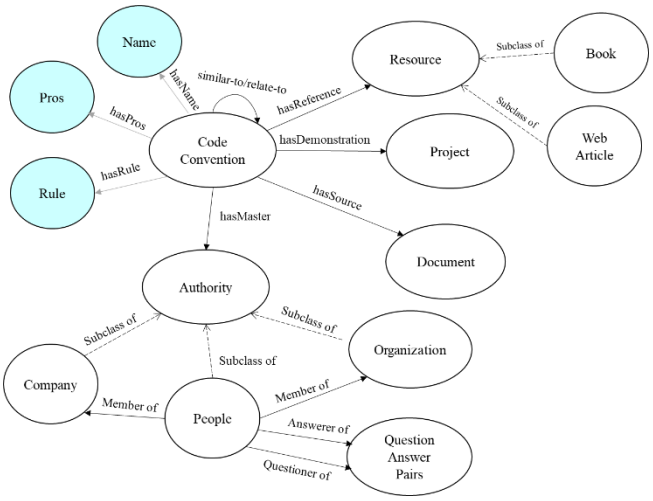

Figure 2. Ontology of CCBase

B. *Information Extraction*

Guided by the ontology, we extract instance data from collected textual materials and store them in CCBase.

The syntactic analysis approach [15] is widely used to extract <subject, predicate, object> triples from sentences, but it isn't applicable for constructing CCBase. It is because entities and property values of coding conventions could not be directly collected from sentences, and also predicates in triples parsed by syntactic analysis approach could not be used as relations in CCBase. Therefore, we propose a semi-automated method to import entities and relations into CCBase, which consists of four steps:

1) Parse file structures of documents.
2) Define a set of rules based on keywords like "example", "benefits" and "author" to match candidates of entities, relations, and properties of entities.
3) Extract candidate entities and relations by rule matching.
4) After quality checking by experts, the final results are imported into CCBase.

The semi-automated method is more accurate than the syntactic analysis method, while it does not cost as much as fully human collection method. Fig. 3 shows some entities and relations gained from information extraction, except for similar-to and relate-to relations, which would be discovered further in section C.

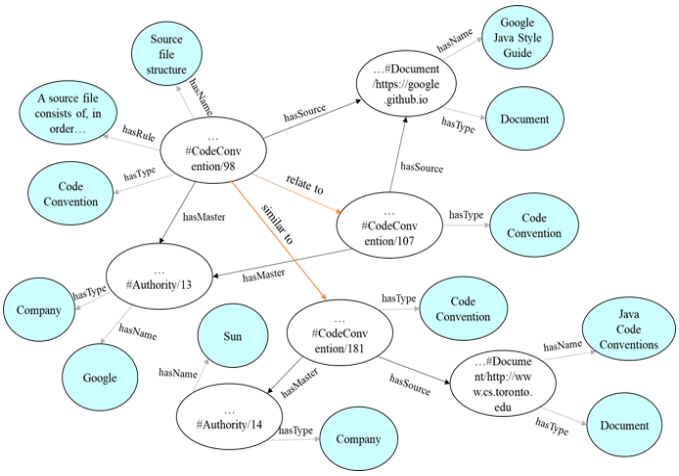

Figure 3. One Fragment of Instances in CCBase

C. *Relation Discovery*

It is necessary to further discover the latent relations between entities. According to the ontology structure of CCBase, the relations between entities include the relations between different coding conventions, and relations between coding conventions and other types of entities. The latter, like hasSource and hasMaster in Fig. 2, could be obtained through information extraction. Thus we focus on discovering latent relations between coding conventions.

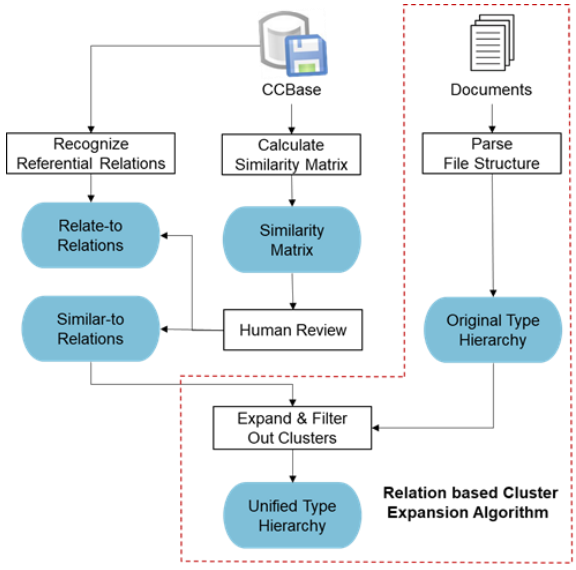

Figure 4. Relation Discovery

We propose a semantic similarity measuring approach to discover these following relations, as shown in Fig. 4.

[2] https://protege.stanford.edu/

1) **similar-to**. [16] lists a set of widely accepted metrics to measure the similarity between entities. Considering most properties of entities in CCBase are long texts, we adopt WHIRL as the similarity metric, which is based on TF-IDF. Then we set up a threshold by experiments to obtain the *Similarity Matrix*, which contains entity pairs with high WHIRL metric values. Finally, experts decide on whether entity pairs in *Similarity Matrix* have similar-to relations.

2) **relate-to**. There are two types of relate-to relations in CCBase. One is referential relations between coding conventions from the same document. The description of a coding convention may refer to other coding conventions in the same document. For example, the coding conventions named "*Package Statement*" in the Google Java Style Guide is described as "*The package statement is not line-wrapped. The column limit (Section 4.4, Column limit: 100) does not apply to package statements.*" It refers to the coding convention named "*Column limit: 100*". For this type of relations, we could find them through information extraction. Another kind of relate-to relations come from entity pairs in the *Similarity Matrix* that do not have similar-to relations.

3) **subsumption**. There is an original type hierarchy of code conventions in each document. For example, coding convention named "*Naming Convention*" includes "*Function Naming Convention*", "*Variable Naming Convention*", etc. However, original type hierarchies have three shortcomings as described in the introduction section. Thus, we propose the RCE algorithm to establish a unified type hierarchy for coding conventions from all documents.

Algorithm: RCE

Input: Given entity set *E*, document set *D*, original type hierarchy set *S*. S_{ij} is the j_{th} primary type of type hierarchy S_i from D_i and S_{ijk} is the k_{th} secondary type belonging to S_{ij}.

Procedure:

1: Expand candidate entity clusters with the same type *C* according to relations.

for *i*, *j* **in** range(0, length(*D*)), range(0, length(S_i)):
  $C_{ij}.entity \leftarrow \emptyset$
  $C_{ij}.layer \leftarrow primary$
  for *k* **in** range(0, length(S_{ij})):
    C_{ijk} = Entities of S_{ijk}
    for *e* **in** Entities of S_{ijk}
      $C_{ijk}.entity \leftarrow C_{ijk}.entity \cup similarEntities(e)$
      $C_{ijk}.layer \leftarrow secondary$
      $C_{ij}.entity \leftarrow C_{ij}.entity \cup C_{ijk}.entity$

2: Filter out replicated clusters and clusters with too few entities. Name every cluster to get type hierarchy *R*.

for c_1, c_2 **in** *C*:
  if similarity$(c_1, c_2) > \theta$:
    C.remove(c_2)
    similarity(c_1, c_2) = $|c_1.entity \cap c_2.entity| / |c_1.entity \cup c_2.entity|$
for *c* **in** *C:*
  if |c| $< \delta$:
    C.remove(*c*)
  else:
    $t.entity \leftarrow c.entity$
    $t.layer \leftarrow c.layer$
    $t.name \leftarrow$ select one name from original types of $t.entity$ **or** make a new name
    T.push(t)

3: Generate type lists for entities.

for t **in** T:
  for e **in** t:
    if *t.layer* is primary:
      e.primary_type_list.push(t.name)
    else:
      e.secondary_type_list.push(t.name)

Output: Unified Type hierarchy *T*, entities set E' with primary and secondary type lists.

The unified type hierarchy *T* holds two layers: the primary layer, and the secondary layer. As we expand clusters with similar-to relations in Step 1, some entities would belong to multiple clusters and finally multiple types, like entities in Freebase. Thus, we use lists to store primary types and secondary types in Step 3. Although there are no direct relations between entities that share the same types, we could group these entities easily by type. This is the reason that we take it as a kind of relation. As a result, a unified type hierarchy for coding conventions is built with 16 primary types and 53 secondary types, as shown in Fig. 5.

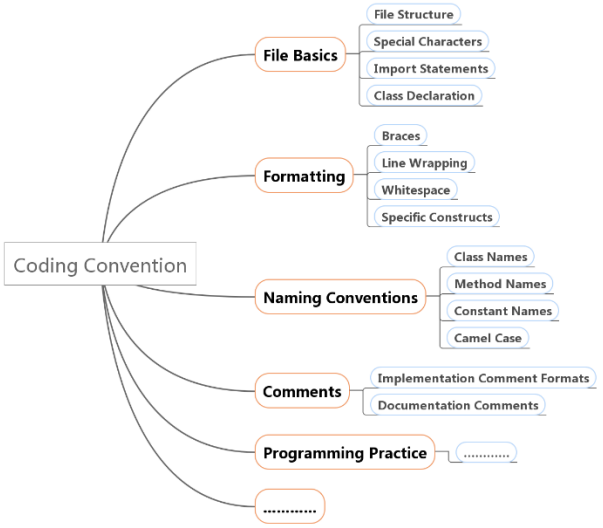

Figure 5. Part of the Unified Type Hierarchy

When applied in CCBase, RCE has the following advantages over LPA and LPA-based algorithms, like SLPA:

- As LPA-based algorithms are semi-supervised, they need many labeled seeds or dense relations between entities to propagate labels, but relations in CCBase are too sparse. Since RCE is unsupervised, it does not suffer from this problem.
- RCE fully utilizes original type hierarchies of documents, while only one layer of original type hierarchies could be used as labels in LPA-based algorithms.

IV. Question Answering over CCBase

To demonstrate the value of CCBase, we develop a question answering system over it, called CCQA. It can assist programmers to retrieval information about coding conventions in a more natural manner.

Inspired by Hu et al.'s work [4], we propose the LE (long entity) Node-First framework to answer coding convention questions by subgraph matching. As Fig. 1 shows, we first extract semantic relations based on the dependency tree of question sentences to build a semantic query graph *Qu*. A semantic relation is a triple *<rel; arg1; arg2>*, where *rel* is a relation phrase, and *arg1* and *arg2* are its associated node phrases. After that, a SPARQL query statement is generated from *Qu* and then executed to get final answers.

LE (long entity) Node-First framework improves Hu et al.'s work from the following two points.

First, since entities about coding conventions are usually complete sentences instead of words or phrases, we use Jena Full Text Search and combine rule-based method for entity linking. We merge words within specific property of entities into one node to obtain clearer sentence structures, and thus the further generated dependency tree can achieve higher accuracy.

Second, when building a query graph *Qu*, the algorithm of [4] also extracts wh-words (what, how, why etc.) as nodes. However, if a question only contains one entity and does not contain any wh-word or relation, the query graph *Qu* will only be formed as one node and the query will be failed. Thus it could not answer Yes/No questions and declarative sentence. To improve the ability of CCQA, we collect some common question templates, such as questions begin with "*Is there any*". These templates will also be recognized as nodes from questions.

So far CCQA has been developed as a plugin in IntelliJ IDEA, which can be downloaded from our Github project[3].

V. Evaluation

Several experiments have been conducted to evaluate CCBase and CCQA.

A. Performance of Information Extraction

We construct CCBase in a top-down way, extracting entities and relations from unstructured documents guided by ontology. To evaluate the effectiveness of this method, we compare it with two bottom-up extraction methods: the popular open information extraction tool – *open IE* [20] and a domain-specific extraction method – HDSKG [15]. Three popular metrics are selected: precision, recall and F1 score.

We collect 8 documents about coding conventions as the dataset of this experiment, and then we ask three experts to label the data manually. Fig. 6 shows the results of the comparison. Open IE and HDSKG both extracts the dependencies from sentences to generate relation triples. However, the entities and relations in coding convention domain are too complex to be directly extracted from one single sentence. The top-down extraction method outperforms HDSKG by 44.2% in F1 score.

[3] https://github.com/14dtj/code-convention-robot

Furthermore, we also conduct an experiment to compare the algorithms of relation discovery. We use LPA, SLPA, and RCE to build different versions of knowledge bases. The results are shown in Fig. 7. We can find that SLPA and RCE perform much better than LPA, because types generated by LPA do not overlap, which is unreasonable for coding conventions. Benefiting from the original type hierarchies of documents, RCE outperforms SLPA by 4.3% in F1 score.

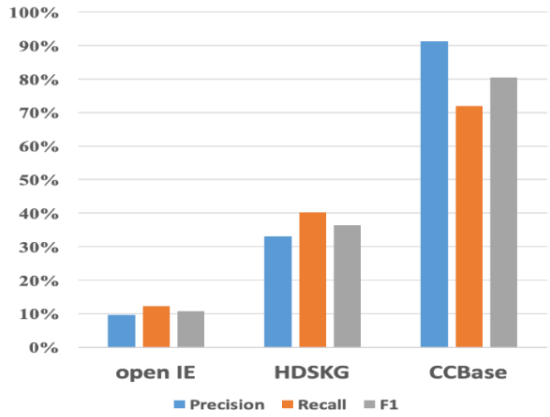

Figure 6. Evaluation of Information Extraction Methods

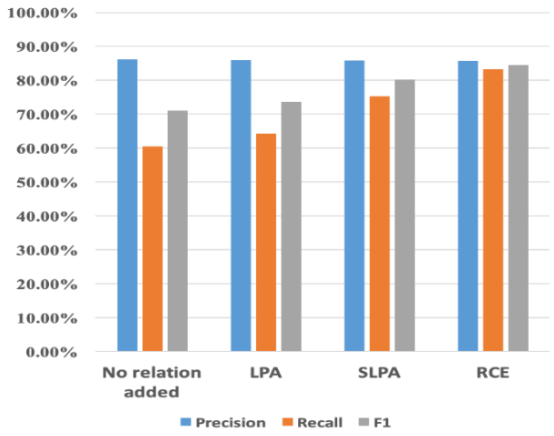

Figure 7. Evaluation of Relation Discover Methods

B. Comparison with other Knowledge Bases

As there are no public knowledge bases in the field of coding convention, we compare CCBase with related subsets of a software engineering knowledge bases such as SEBase [5] and software.zhishi.schema [19]. We also compare it with YAGO [18], a general knowledge base.

TABLE I: Comparison with Other Knowledge Bases

	CCBase	SEBase	zhishi	YAGO
Concept	3761	128	38	31
Subsumption	181	57	50	0
Relate-to	524	12	0	0
Similar-to	62	0	0	0

Table I shows the number of entities and relations of each dataset. We could discover that our knowledge base is larger than other existing datasets as for the entity number related to coding conventions. Besides, the relations between entities are richer, especially as for subsumption and related-to relations.

C. Performance of Question Answering

We crawled 214 code convention questions from StackOverflow as experimental datasets. The performance of a QA system is measured by the ratio of questions that are answered correctly.

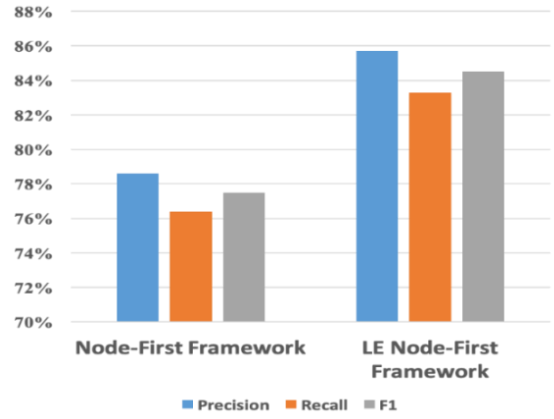

Figure 8. Evaluation of Question Answering Methods

We compare our approach (LE Node-First Framework) with Node-First Framework by Hu et al. [4]. Fig. 8 reveals the results on 214 questions. Node-First Framework adopts CrossWikis dictionary [17] to map entities in user questions, which is not suitable for long entity linking. Besides, it could not handle Yes/No questions and declarative sentences. It is shown that our LE Node-First Framework achieves 84.5% in F1 score, while the F1 score of original Node-First Framework is only 77.5%.

VI. Conclusion

In this paper, we designed and constructed CCBase, the first coding convention knowledge base, from online resources. And for programmer's convenient access, a question answering system over CCBase was further developed. Experiments show that CCBase contains much more entities and relations about coding conventions than previous knowledge bases, and our QA system achieves an F1 score of 84.5% on 214 questions raised in StackOverflow.

As for future work, we will try to extract more entities and relations about coding conventions from Github and other Web sites to enrich CCBase. Moreover, it would be interesting to explore more potential applications based on this CCBase such as code bad smell detection.

VII. Acknowledgement

This research was sponsored by the National Key Research and Development Program of China (Project No. 2018YFB1003903), National Nature Science Foundation of China (Grant No. 61472242 and 61572312), and Shanghai Municipal Commission of Economy and Informatization (No. 201701052).

References

[1] Bahman Arasteh, Jalal Najafi, “Programming guidelines for improving software resiliency against soft-errors without performance overhead”, *Computing* 100(9): 971-1003 (2018)

[2] CHEN, H., CHEN, W., and Lee, C. C. (2018). “An Automated Assessment System for Analysis of Coding Convention Violations in Java Programming Assignments”, *Journal of Information Science and Engineering*, *34*(5), 1203-1221.

[3] Tourwe, Tom, and Kim Mens, "Mining aspectual views using formal concept analysis.", *Source Code Analysis and Manipulation, Fourth IEEE International Workshop on*, 2004, pp. 97-106.

[4] Hu S, Zou L, Yu J X, et al, “Answering natural language questions by subgraph matching over knowledge graphs” in *IEEE Transactions on Knowledge and Data Engineering*, *2018*, 30(5): 824-837.

[5] Kai Chen, Xiang Dong, Jiangang Zhu, Beijun Shen, “Building a Domain Knowledge Base from Wikipedia: a Semi-supervised Approach”, *SEKE*, *2016*, pp. 191-196

[6] Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge”, in *SIGMOD ’08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247-1250.

[7] Hongwei Li, Sirui Li, Jiamou Sun, Zhenchang Xing, Xin Peng, Mingwei Liu, Xuejiao Zhao, “Improving API Caveats Accessibility by Mining API Caveats KnowledgeGraph”, *ICSME 2018*, pp. 183-193.

[8] Jiang, Jyun-Yu, Pu-Jen Cheng and Chin-Yew Lin, “Entity-driven Type Hierarchy Construction for Freebase.”, *WWW,2015,* pp. 47-48.

[9] 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.

[10] J. Xie, B. K. Szymanski and X. Liu, "SLPA: Uncovering Overlapping Communities in Social Networks via a Speaker-Listener Interaction Dynamic Process," in *IEEE 11th International Conference on Data Mining Workshops, Vancouver, 2011*, pp. 344-349.

[11] W. Yih, M. Chang, X. He, and J. Gao, “Semantic parsing via staged query graph generation: Question answering with knowledge base,” in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Language Process. Asian Fed. Natural Language Process.*, *2015*, pp. 1321–1331.

[12] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano, “Template-based question answering over RDF data,” in *Proc. World Wide Web*, *2012*, pp. 639–648.

[13] L. Dong, F. Wei, M. Zhou, and K. Xu, “Question answering over free-base with multi-column convolutional neural networks,” in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Language Process. Asian Fed. Natural Language Process*, *2015*, pp. 260–269.

[14] Ren, Yuan, Artemis Parvizi, Chris Mellish, Jeff Z. Pan, Kees van Deemter and Robert Stevens. “Towards Competency Question-Driven Ontology Authoring.” , *ESWC, 2014, pp. 752-767.*

[15] X. Zhao, Z. Xing, M. A. Kabir, N. Sawada, J. Li and S. Lin, "HDSKG: Harvesting domain specific knowledge graph from content of webpages," *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, 2017*, pp. 56-67

[16] A. K. Elmagarmid, P. G. Ipeirotis and V. S. Verykios, "Duplicate Record Detection: A Survey," in *IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, 2007*, pp. 1-16.

[17] V. I. Spitkovsky and A. X. Chang, “A cross-lingual dictionary for english wikipedia concepts,” in *Proc. 8th Int. Conf. Language Resources Eval.*, *2012*, pp. 3168–3175.

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

[19] Zhu, Jiangang, Haofen Wang, and Beijun Shen. "Software. zhishi. schema: A Software Programming Taxonomy Derived from Stackoverflow", In *International Semantic Web Conference (Posters & Demos)*, 2015.

[20] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, “Open information extraction from the web”, *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.