Verification Based on Hyponymy Hierarchical Characteristics for Web-Based Hyponymy Discovery

Lili Mou, Ge Li*, Zhi Jin, and Lu Zhang

Software Institute, School of EECS, Peking University, Beijing, P.R. China
Key Laboratory of High Confidence Software Technologies, Peking University,
Ministry of Education, Beijing, P.R. China
{moull12,lige,zhijin,zhanglu}@sei.pku.edu.cn

Abstract. Hyponymy relations are the skeleton of an ontology, which is widely used in information retrieval, natural language processing, etc. Traditional hyponymy construction by domain experts is labor-consuming, and may also suffer from sparseness. With the rapid development of the Internet, automatic hyponymy acquisition from the web has become a hot research topic. However, due to the polysemous terms and casual expressions on the web, a large number of irrelevant or incorrect terms will be inevitably extracted and introduced to the results during the automatic discovering process. Thus the automatic web-based methods will probably fail because of the large number of irrelevant terms. This paper presents a novel approach of web-based hyponymy discovery, where we propose a term verification method based on hyponymy hierarchical characteristics. In this way, irrelevant and incorrect terms can be rejected effectively. The experimental results show that our approach can discover large number of cohesive relations automatically with high precision.

Keywords: Ontology, hyponymy learning, term verification.

1 Introduction

Ontologies are a formal model to represent domain knowledge, which is widely used in information retrieval, natural language processing and various other applications [1]. Hyponymy is the skeleton of a domain ontology, expressing "is-a-kind-of" or "is-a-subclass-of" relations [2]. For example, "Java" is a kind of "programming language" (PL), denoted as "Java $\sqsubseteq$ PL". "Java" is the hyponym while "PL" is the hypernym.

Traditional hyponymy relations are often constructed by domain experts, which is a labor- and time- consuming task. Moreover, manually-constructed knowledge is typically suffering from severe data sparseness. As reported in Section 6, the hyponymy relations in WordNet under certain root concepts cover only 3%-5% of what we have discovered.

In 1992, Hearst proposed an automatic hyponymy acquisition approach [3]. Given a large corpus, lexical-syntactic patterns (e.g. "A such as B, C") are used to extract hyponymy $B, C \sqsubseteq A$. Based on Hearst's work, many other researchers proposed approaches to discover more and more hyponyms in a bootstrapping framework [4]. Pattern learning approaches are also proposed to discover lexical-syntactic patterns as well

* Corresponding author.

R. Buchmann et al. (Eds.): KSEM 2014, LNAI 8793, pp. 81–92, 2014.

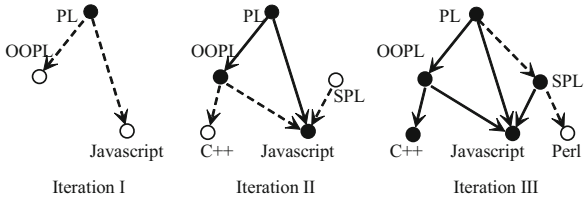

Fig. 1. The iterative hyponymy discovering process. Black dots and solid lines are existing terms and relations; blank dots and dashed lines are newly selected terms and relations in each iteration.

as hyponyms [5]. However, the hyponymy relations acquired in this fashion are typically loose and low-coupling because they are not confined in a certain domain, which fail to form the skeleton of a domain ontology.

To overcome this problem, a naïve and direct approach is to search on the Internet and discover hyponymy iteratively from a starting concept. Figure 1 illustrates the iterative searching process. Given the root concept, say programming language (PL), we search on the search engine, and use lexical-syntactic patterns to extract hyponyms of PL, e.g., "javscript", "object-oriented PL" (OOPL). Then we extract both hyponyms and hypernyms of the newly extracted concepts and we obtain "C++", "scripting programming language". So on and so forth, we acquire the "cohesive" hyponymy relation under PL, which form the skeleton of the domain ontology.

However, in such an iterative process, "irrelevant term explosion" is a common problem, where the number of irrelevant terms grows exponentially. When extracting hyponyms and hypernyms, irrelevant terms might be included because of polysemous terms and casual expressions. Even if very few mis-matched or irrelevant terms are extracted during one iteration, they will introduce even more other irrelevant terms in the next iteration, so on and so forth, so that the system will soon break down within one or two iterations.

In this paper, we propose a term verification method based on *Hyponymy Hierarchical Characteristics* (HHC). As we know, hyponymy is a specific kind of relation between two terms. A complete, well-formed hyponymy hierarchy has some inherent characteristics, e.g., hyponymy transitivity. These characteristics can be quantified and used to evaluate the degree to which a hyponymy hierarchy is well-formed. For a growing hyponymy hierarchy, its characteristics tend to be satisfied to a large degree if the newly discovered terms are probably correct. On the contrary, if an irrelevant or incorrect term is introduced, the characteristics are likely violated to a large degree. So, by quantifying these characteristics and using some appropriate thresholds, we can judge whether a newly discovered term should be included or not. Therefore, irrelevant terms can be removed as soon as possible during the iterative searching process.

2 Related Work

Hearst proposed the "such as" pattern to extract hyponymy in [3]. Z. Kozareva raised the doubly anchored pattern (DAP) for instance discovery in [6]. In the pattern "NP such as NP_1 and X", they specify at least one instance NP_1 in advance. Then they discover

other instances by applying the pattern iteratively. H. Eduard et al. use the backward doubly-anchored pattern (DAP^{-1}) [4] to iteratively extract instances and intermediate concepts. Sanchez extracts immediate anterior word as an classification method [7], e.g., lung cancer $\sqsubseteq$ cancer. R. Snow et al. discover lexical-patterns and hyponymy relations iteratively in [5]. They use patterns to extract relations, and relations to extract patterns iteratively.

Many researchers focus on removing irrelevant terms during their hyponymy discovery. Co-occurrence is a common technique to judge the relevance between two terms [8,9,7]. [10] uses content words for domain filtering. [11] extracts instances from free texts coupled with semi-structured html pages. In [12,5], a group of patterns are used to determine whether a relation holds.

In our previous work [13], we proposed a term verification method based on text classification, where a training corpus should be provided in advance. In [14], we considered hyponymy transitivity. In this paper, we extend the method in [14] and semantically verify the candidate results by three hyponymy hierarchical characteristics. With irrelevant terms removed effectively, the system can search iteratively and automatically.

3 Overview of Our Approach

Our approach of "cohesive" hyponymy discovery is an iterative process. Given a root concept, we do web searching and pattern matching to extract candidate hyponyms and/or hypernyms. Then HHCs are used to judge whether the candidates are relevant to the root. Selected candidates are then searched iteratively. Figure 2 illustrates the overview of our approach and the main steps are listed as follows.

Step 1 We first specify a root term r.
Step 2 We construct a search query by a lexical-syntactic pattern and r. The query is sent to a search engine. Web snippets returned by the search engine are collected.
Step 3 By matching the lexical-syntactic pattern "A is a|an B", noun phrases A, B are extracted as candidate hyponyms.
Step 4 Since a noun phrase is not necessarily a term which represents a common concept in the human mind (e.g., "a well-designed PL"), we verify whether the noun phrases extracted in Step 3 are terms with a pragmatic method. If A appears in a pattern "A is a|an", then A is regarded as at term.
Step 5 Three types of hyponymy hierarchical characteristics are used to verify whether a term is relevant to r. Irrelevant terms are removed.
Step 6 We search the newly selected terms for both hyponyms and hypernyms on the web in the next iteration. (Go to Step 2.)

In Section 4, we introduce HHCs in detail. Based on these characteristics we design our candidate hyponymy verifying method in Section 5.

4 Hyponymy Hierarchical Characteristics

As mentioned in Section 1, the cohesive hyponymy relations under a root term have some inherent characteristics which can be for term verification. In this paper, we

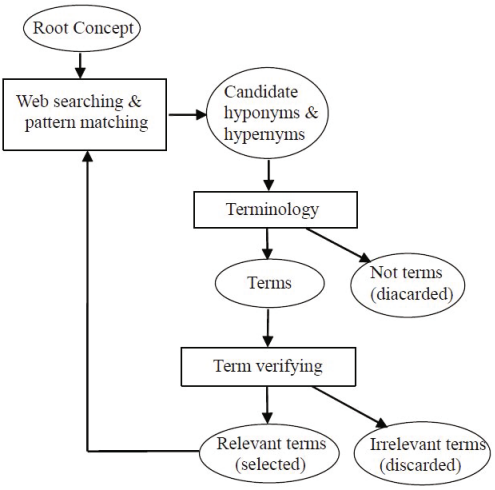

Fig. 2. Overview of our approach

propose three types of *Hyponymy Hierarchical Characteristics* (HHC), namely (1) *hyponymy transitivity characteristic*, (2) *hyponym-hypernym ratio decreasing characteristic* and, (3) *hyponym overlapping characteristic*.

A complete, well-formed hyponymy hierarchy will exhibit these hyponymy hierarchical characteristics, which can be used to evaluate the degree to which a growing hyponymy hierarchy is well-formed. The characteristics of a growing hyponymy hierarchy tends to be satisfied if the newly discovered term is probably correct. On the contrary, if an incorrect or irrelevant term is introduced to the growing hyponymy hierarchy, these characteristics tend to be violated to a large degree. We quantify these characteristics, and use thresholds to judge whether a candidate term should be included or not, introduced in detail in the rest part of this section.

4.1 Hyponymy Transitivity Characteristic

As we know, hyponymy is a transitive relation. For terms a, b, c, if $a \sqsubseteq b$ and $b \sqsubseteq c$, it can be inferred that $a \sqsubseteq c$. We call this *Hyponymy Transitivity Characteristic* (HTC).

Since the web contains massive information, these semantically redundant expressions all may stated explicitly on the web, and they can be cross-validated by each other.

In particular, if we want to get hyponyms or hypernyms of a term t, we first get candidate hyponyms and hypernyms of t. For each candidate c (either $c \sqsubseteq t$ or $t \sqsubseteq c$), we search potential hyponyms and hypernyms of c. In this case, hyponymy transitivity can be used to verify c. There are four typical scenarios when we verify c. Whether c is *cross-validated* by HTC in each scenario is shown in Figure 3 and explained as follows.

Scenario 1: If c is a candidate hyponym of t, and if c is the hyponym or hypernym of any one of other already selected terms (t excluded), then c is regarded as a relevant term since it is cross-validated by other hyponymy relations.

Scenario 2: If c is a candidate hypernym of t, and c is a hyponym of some other select terms, then c is also cross-validated.

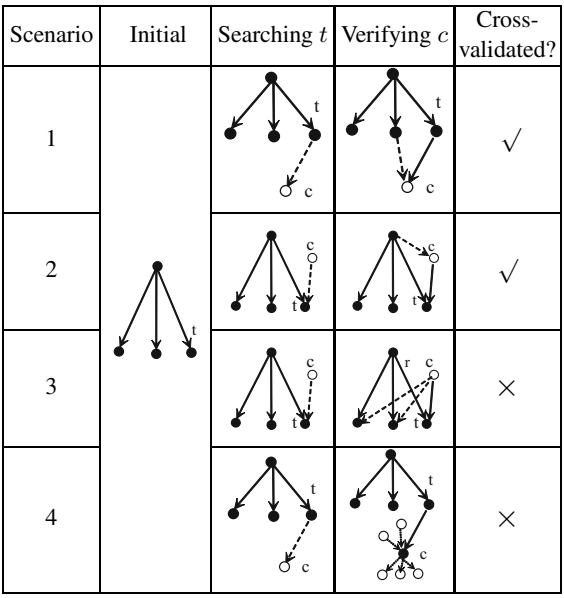

Fig. 3. Scenarios when verifying a candidate term by HTC

Scenario 3: In this scenario, c is a candidate hypernym of t, and some other selected terms are the hyponym of c. If no other hyponymy relations can infer $c \sqsubseteq r$, then c is not cross-validated.

Scenario 4: If no hyponyms and hypernyms of c have relations with any selected term (t excluded), then c is not cross-validated.

4.2 Hyponymy-Hypernym Ratio Decreasing Characteristic ($R_{HH}DC$)

We define *Hyponym-Hypernym Ratio* (R_{HH}) as follows[1].

$$R_{\text{HH}}(t) = \frac{\#\text{Hyponym}(t) + 1}{\#\text{Hypernym}(t) + 1}$$

where adding one in the equation is a widely used smoothing technique used in many NLP tasks [15].

For two terms a, b, If $a \sqsubseteq b$, then $R_{\text{HH}}(a) < R_{\text{HH}}(b)$. This is called *Hyponymy-Hypernym Ratio Decreasing Characteristic* (R_{HH}DC).

The above fact can be proven by the definition of hyponymy relation. If $a \sqsubseteq b$, b inherits all the hyponyms of a. So b has more hyponyms than a. Likewise, a has more hypernyms than b. Therefore $R_{\text{HH}}(a) < R_{\text{HH}}(b)$.

Though we cannot get the entire hyponymy relations, the phenomenon still holds true that when we search a term, a higher level term tends to have more hyponyms and less

[1] Hyponym(t) is the hyponyms of t, Hypernym(t) is the hypernyms of t. # refers to the number of elements in a set.

hypernyms, and thus has a higher R_{HH} value. A preset threshold is used to determine whether a candidate term satisfies R_{HH}DC.

Formally, if $c \sqsubseteq t$ and $R_{\text{HH}}(c) > t_1 \cdot R_{\text{HH}}(t)$, then we say R_{HH}DC is violated, which indicates an anomaly. A typical cause of R_{HH}DC violation is the presence of polysemous terms. For example, "scheme" is a kind of PL. Meanwhile, "scheme" has many different meanings, one of which refers to a schema or an outline. Therefore R_{HH}(scheme) may be much larger than R_{HH}(PL). Terms that violate R_{HH}DC cannot be used for verifying other terms with hyponymy transitivity (introduced in Section 4.1). Otherwise, a large number of irrelevant terms might be mis-validated.

4.3 Hyponym Overlapping Characteristic (HOC)

We define *Hyponym Overlapping Proportion* (HOP) of terms a, b as follows.

$$P_{\text{HO}}(a,b) = \frac{\#\left\{\text{Hyponym}(a) \bigcap \text{Hyponym}(b)\right\}}{\#\text{Hyponym}(a)}$$

In a complete, well-formed hyponymy hierarchy, if $a \sqsubseteq b$, then $P_{\text{HO}}(a,b)$=1, because Hyponym(a) $\subseteq$ Hyponym(b). Though we cannot extract the complete hyponyms of a and b, $P_{\text{HO}}(a,b)$ is probably high if $a \sqsubseteq b$. This is called *Hyponym Overlapping Characteristic* (HOC). A threshold is also used to determine whether HOC is satisfied.

When discovering hyponymy relations under the root term r by iterative searching, we calculate $P_{\text{HO}}(c,r)$ to verify a candidate term c. If $P_{\text{HO}}(c,r)$ is greater than a preset threshold t_2, HOC is satisfied, which indicates that t might be a relevant term.

However, because we add new terms to hyponyms of r in each iteration, Hyponym(t) $\bigcap$ Hyponym(r) grows larger naturally during the iterations. P_{HO} is not stable in different iterations. Therefore, we multiply a heuristic regularization factor and extend $P_{\text{HO}}(c,r)$ to the following equation.

$$P_{\text{HO}}(c,r) = \frac{(N+1)\ln(N+1)}{(\#\text{Hyponym}(c)+1)\ln(S+1)}$$

Here $N = \#\left\{\text{Hyponym}(c) \bigcap \text{Hyponym}(r)\right\}$, S is the size of the entire selected hyponyms of r. Adding-one smoothing is also applied for P_{HO}.

Hyponymy overlapping characteristic can deal with the situation where those hyponymy relations are not extracted explicitly from the web. For example, "Turing complete PL (TCPL) $\sqsubseteq$ PL" is not stated explicitly on the web. Therefore TCPL cannot be cross-validated by HTC because none of the known terms is stated explicitly as the hypernym of TCPL. However, when we examine the hyponyms of TCPL, we find the majority of them are known hyponyms of PL, e.g., javascript, ruby, matlab etc. So, $P_{\text{HO}}(\text{TCPL}, \text{PL})$ is large and TCPL is probably a relevant term to PL.

5 Verifying Candidate Terms Based on Hyponymy Hierarchical Characteristics

In this part, we explain our method of candidate term verification based on the three types of HHCs introduced in Section 4.

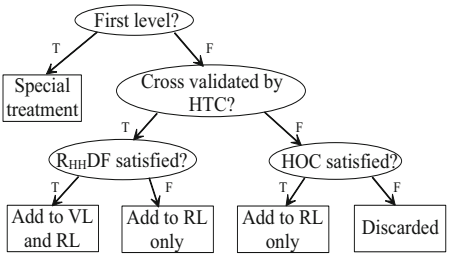

Fig. 4. The decision process of verifying a candidate term based on hyponymy hierarchical characteristics

To implement our semantically verification approach, we need to maintain two lists of terms: `Result List` (RL) and `Cross Validate List` (VL). RL is the hyponyms obtained as the result; VL is a subset of RL for which we have higher confidence, and VL is used for cross-validation. These two lists are updated in two iterations.

Figure 4 gives the decision process of candidate term verification. For a candidate term c, if c is cross-validated by HTC with VL (Scenario 1 and Scenario 2 in Subsection 4.1), c is added to RL. Furthermore, if R_{HH}DC constraint of c is satisfied, i.e., $R_{\text{HH}}(c) < t_1 \cdot R_{\text{HH}}(c\text{'s hypernym})$, then c is considered as a relevant term with high confidence, and is added to VL. If R_{HH}DC constraint of c is not satisfied, c is added to RL only since it may indicate that c is a polysemous term like "scheme".

If c is not cross-validated by HTC, but after we obtain the hyponyms of c, if the hyponym overlapping proportion is large, i.e., $\text{HOP}(c, r) > t_2$, this may indicate the absence of some hyponymy relations. Therefore we regard c as a relevant term and add c to RL. However, to be on the safe side, c is not added to VL.

For those terms that are neither cross-validated by HTC nor satisfying HOC constraint, they are discarded.

There still remains one problem to solve. Since there are no selected terms in the first iteration, how can we validate candidate terms at the beginning? In our approach, given the root term r, we get candidate hyponym c, and search c on the web. If r is extracted as the hypernym of c, then c is selected. This seems trivial, but can effectively remove some noises caused by html parsing errors, sentence segmenting errors, etc.

6 Experimental Results

We carried out two experiments under the root terms "programming language" and "algorithm". The root terms are assigned respectively by human in advance. Then the system works automatically by our approach.

Our approach needs two parameters t_1 and t_2, which are set to 3 and 0.1 empirically. These two parameters are fixed between the two experiments. The number of iterations is limited to 3.

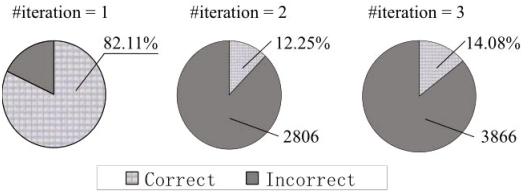

Fig. 5. Percentage of correct candidate terms and number of incorrect terms in each iteration in Experiment I before pruning by HHCs

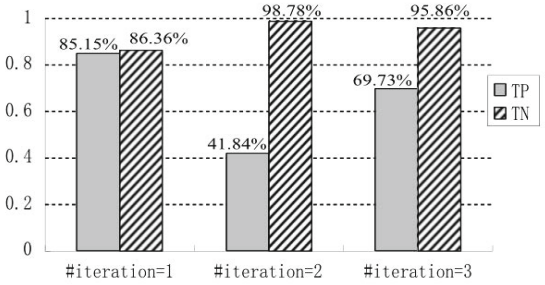

Fig. 6. TP and TN of term verification in each iteration in Experiment I

6.1 Experiment I

In the first experiment, we discovered hyponymy under the root term "programming language." The system has totally accepted 975 terms and discarded 7002 terms. Full sized evaluation is executed on the system-accepted terms; for those terms that are discarded by our system, we randomly sample about 100 terms and report the sampling statistics. Each term is annotated by human, which falls into three categories, namely CORRECT, INCORRECT and BORDERLINE. Those in BORDERLINE category are not counted in our result.

Accuracy of Term Verification. In this part, we evaluate our candidate term verification approach based on hyponymy hierarchical characteristics.

In Figure 5, we present the percentage of the actual correct terms and the number of incorrect terms in each iteration. In the first iteration, it achieves 82.11% accuracy for the candidate terms that are extracted. In the second and third iteration, the accuracy among the 3000-4000 total candidates is only about 13%. Therefore, the "irrelevant term explosion" problem is severe in the iterative hyponymy discovery framework. Selecting correct terms is like gold mining — finding small quantities of the wanted among massive unwanted.

In our approach, candidate terms are verified semantically by hyponymy hierarchical characteristics. True positive rate (TP) and true negative rate (TN) are used to evaluate accuracy. Figure 6 shows TP and TN in the three iterations. Except for the special treatment in the first iteration, TN rate is high (over 95%) in the remaining iterations,

Table 1. Recall against Wikipedia list and TIOBE index

	Our results	Standard			
		Wikipedia List	TIOBE Top 20	TIOBE Top 50	TIOBE Top 100
#	693	645	20	50	100
Recall	-	30.43%	100%	92%	79%

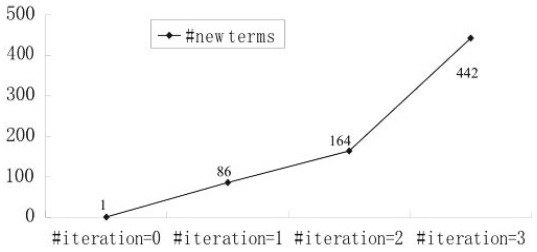

Fig. 7. Number of terms we get in each iteration in Experiment I

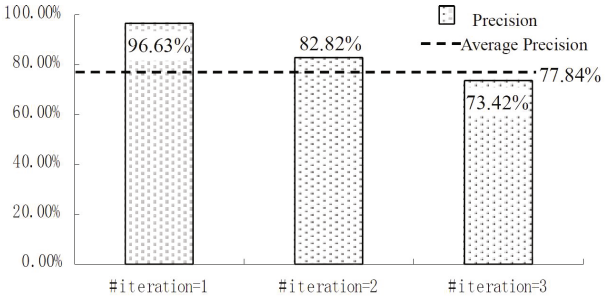

Fig. 8. Precision in each iteration in Experiment I

which ensures the system can work automatically and effectively. TP rate is relatively low at the first glance, since our candidate verifying approach is very strict. One reason for the low TP rate is that some programming languages are only mentioned in one or a few scientific papers, and therefore it cannot be cross-validated by either HTC or HOC. However, on second thought, it seems reasonable since we as human beings can hardly confirm a hyponym of PL if it is mentioned only once on the web. Despite the relatively low TP, the number of selected terms is large and the recall of commonly used PL is also high.

As seen, the iterative process will introduce a huge number of irrelevant terms as candidates. Our term verification method can effectively remove the irrelevant terms before they grow exponentially.

Precision and Recall. In this part, we evaluate precision and recall of our result.

Figure 7 presents the number of new terms in each iteration. Totally, we get 692 correct hyponym terms under the term PL. We compare the number to two open hyponymy

Table 2. Number of hyponyms under "programming language" in WordNet, WordNet+40k, and our result

WordNet	WordNet+40k	Our result
34	77	692

Table 3. Number of hyponyms under "algorithm" in WordNet, WordNet+40k, and our result

WordNet	WordNet+40k	Our result
3	8	223

lexicons. In particular, one is WordNet 2.1[2] ; the other is an automatic enlarging of WordNet by machine learning approaches [5][3], which adds up to 40,000 concepts to WordNet. As shown in Table 2, the number of hyponyms we get is much more significant than the other two related works.

Figure 8 shows the precision during the iterative searching process. The overall precision reaches 77.89%.[4]

Calculating recall is difficult because we can hardly find a complete, authentic gold standard. WordNet is not suitable to be the gold standard since it suffers from severe term sparseness. We find two lists that are suitable to compare our results to.

1. Wikipedia list[5]. Wikipedia reflects the collective wisdom of the online community. Everyone can edit and contribute to Wikipedia so that this list is comparatively complete, containing 645 instances of PL.
2. TIOBE index[6], which lists top 20/50/100 PLs indexed by TIOBE programming community. Comparing to TIOBE index, we can see the recall on common programming languages.

Table 1 shows the number of concepts in the golden standards, and the recall calculated against these standards. As we can see, more than 30% instances of Wikipedia lists are recalled; for the TIOBE top 20/50/100 PL index, our recall is 100%, 92% and 79% respectively, which is also high.

This experiment shows our approach has acquired both high precision (77.89% on average) and high recall against Wikipedia list and TIOBE index.

6.2 Experiment II

In the second experiment, we extract hyponymy relations under the root term "algorithm". The parameters of our algorithm remains unchanged.

Figure 9 presents the TP and TN of our term verification approach. As shown in Figure 10, we totally obtain 379 terms under "algorithm" while the average precision

[2] WordNet is available on `http://wordnet.princeton.edu/`

[3] This work is available on `http://ai.stanford.edu/~rion/swn/`

[4] Those terms annotated as BOARDERLINE are not counted.

[5] `http://en.wikipedia.org/wiki/List_of_programming_languages`

[6] `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`

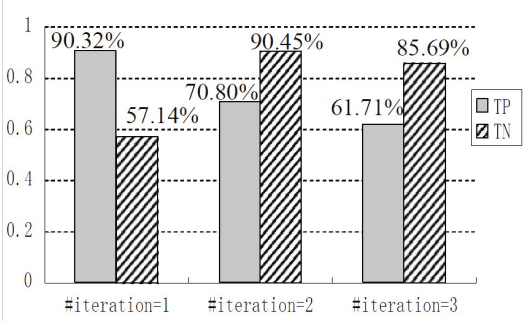

Fig. 9. TP and TR of term verification in each iteration in Experiment II

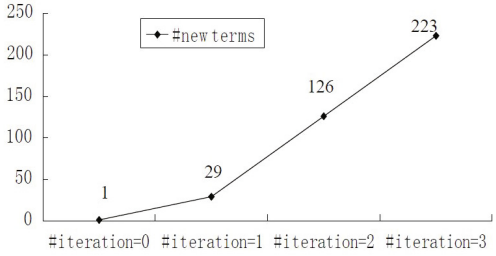

Fig. 10. Number of terms we get in each iteration in Experiment II

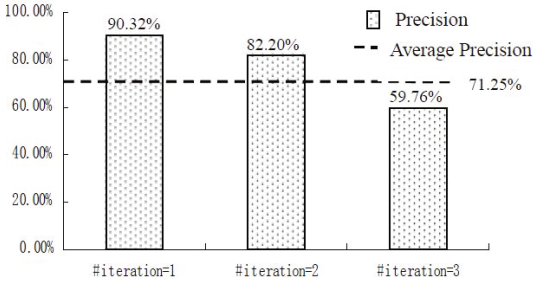

Fig. 11. TP and TN of term relevance verifying in each iteration in Experiment II

is 71.25%, which is also acceptable. The number of hyponyms is also more significant than WordNet and WordNet+40k (See Figure3).

This experiment shows that our approach is general applicable since we do not change the parameters.

From the two experiments, we can see that our approach has extracted a large number of terms with high recall. Incorrect and irrelevant terms can be removed effectively, so that the system can work iteratively and automatically.

7 Conclusion

In this paper, we propose a domain hyponymy discovery approach with term verification based on hyponymy hierarchy characteristics.

Our approach needs very few human supervision. Only a root concept should be given in advance. Then the system can search for domain hyponymy relations automatically and unsupervisedly. At the end of the process, we gain the "cohesive" hyponymy with relatively high precision and recall, which forms the skeleton of the domain ontology.

Acknowledgment. This research is supported by the National Natural Science Foundation of China under Grant Nos. 61232015 and 91318301.

References

1. Clark, M., Kim, M., Kruschwitz, U., Song, D., Albakour, D., Dignum, S., Beresi, U., Fasli, M., Roeck, A.: Automatically structuring domain knowledge from text: An overview of current research. Information Processing and Management 48(3), 552–568 (2012)
2. Buitelaar, P., Cimiano, P., Magnini, B.: Ontology learning from text: methods, evaluation and applications. IOS press (2005)
3. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th Conference on Computational Linguistics, vol. 2 (1992)
4. Hovy, E., Kozareva, Z., Riloff, E.: Toward completeness in concept extraction and classification. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 2 (2009)
5. Snow, R., Jurafsky, D., Ng, A.Y.: Learning syntactic patterns for automatic hypernym discovery. In: Advances in Neural Information Processing Systems (2004)
6. Kozareva, Z., Riloff, E., Hovy, E.H.: Semantic class learning from the web with hyponym pattern linkage graphs. In: Proceedings of ACL 2008: HLT (2008)
7. Sanchez, D., Moreno, A.: A methodology for knowledge acquisition from the web. International Journal of Knowledge-Based and Intelligent Engineering Systems (2006)
8. Cimino, P., Steffen, S.: Learning by googling. SIGKDD Explor. Newsl. 6(2) (2004)
9. Sanchez, D.: Learning non-taxonomic relationships from web documents for domain ontology constructions. In: Data and Knowledge Engineering (2008)
10. Navigli, R., Paola, V., Stefano, F.: A graph-based algorithm for inducing lexical tax- onomies from scratch. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, vol. 3, pp. 1872–1877 (2011)
11. Andrew, C., Justin, B., Richard, W., Hruschka, J., Estevam, R., Mitchell, T.: Coupled semi-supervised learning for information extraction. In: Proceedings of the third ACM International Conference on Web Search and Data Mining, pp. 101–110 (2010)
12. Pantel, P., Pennacchiotti, M.: Espresso: leveraging generic patterns for automatically harvesting semantic relations. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (2006)
13. Mou, L.L., Li, G., Jin, Z., Lu, Y.Y., Hao, Y.Y.: Discovering domain concepts and hyponymy relations by text relevance classifying based iterative web searching. In: Proceedings of the 2012 Asian Parcific Software Engineering Conference (2012)
14. Mou, L.L., Li, G., Jin, Z.: Domain hyponymy hierarchy discovery by iterative web searching and inferable semantics based concept selecting. In: Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference, pp. 387–392 (2013)
15. Jurafsky, D., Martin, J.: Speech And Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall (2009)