WORD SENSE DISAMBIGUATION BASED ON STRETCHABLE MATCHING OF THE SEMANTIC TEMPLATE

WEI WANG AND DEGEN HUANG*

School of Computer Science and Technology, Dalian University of Technology
No.2 Linggong Road, Ganjingzi District, Dalian City, Liaoning Province, China

HAITAO YU

Faculty of Library, Information and Media Science
University of Tsukuba, Tsukuba, Japan

(Communicated by Guoshan Zhang)

ABSTRACT. It is evident that the traditional hard matching of a fixed-length template cannot satisfy the nearly indefinite variations in natural language. This issue mainly results from three major problems of the traditional matching mode: 1) in matching with a short template, the context of natural language cannot be effectively captured; 2) in matching with a long template, serious data sparsity will lead to a low success rate of template matching (i.e., low recall); and 3) due to a lack of flexible matching ability, traditional hard matching is more prone to failure. Therefore, this paper proposed a novel method of stretchable matching of the semantic template (SMOST) to deal with the above problems. We have applied this method to word sense disambiguation in the natural language processing field. In the same case of using only the SemCor corpus, the result of our system is very close to the best result of existing systems, which shows the effectiveness of new proposed method.

1. Introduction. As a vital issue of natural language processing, word sense disambiguation (WSD), although it has been studied and addressed for a long time, still remains an open research problem [20, 22]. The performance of WSD systems has witnessed certain improvements in recent years in tandem with further research on machine learning, especially the wide application of deep learning. A WSD system can be knowledge-based, supervised, unsupervised, semi-supervised, or a combination of these forms. The knowledge-based method, which extracts information from electronic dictionaries or other knowledge sources for disambiguation, mainly relies on the scale and details of the words that are described in knowledge resources [21, 6, 8, 2]. The supervised method, which trains a special model for disambiguation on a manually labeled corpus, has one flaw in that a large-scale labeled corpus must be provided [37, 31, 24]. The unsupervised method, which directly trains a model on a large-scale unlabeled corpus for disambiguation to avoid the higher costs of a manually labeled corpus, still has a large performance gap...
compared to the supervised method [33, 23]. The semi-supervised method, which is a combination of the supervised and unsupervised methods, trains a model on a small-scale manually labeled corpus and a large-scale unlabeled corpus [23, 34].

Currently, the supervised method outperforms the others, but there are two problems that need to be solved: 1) it must consider how to further expand the manually labeled corpus so that the performance can be improved, and 2) determine how to fully discover valid information in an existing labeled corpus. It is known that even with the same size corpus, if the information for disambiguation can fully be explored, then the higher quality information that can be extracted will result in higher performance.

In order to explore more valid information for WSD in the existing corpus, this paper proposes a method based on the stretchable matching of the semantic template (SMOST) for word sense disambiguation. It has three main characteristics: 1) the semantic information of words in context is extended to alleviate data sparsity, 2) the word-order information in the context is considered to improve precision, and 3) a stretchable matching, rather than traditional fixed-length matching, is used to solve the problem of low matching success rates.

2. **Motivation.** Template matching is a common method that is used in natural language processing. Generally, it is implemented by matching the words in a test sentence with those in a template. As shown in Figure 1, the words in a test sentence are denoted by “Ws” (word of a sentence), and words in templates are denoted as “Wt” (word of a template). Template matching is successful when \( Ws1 = Wt1, Ws2 = Wt2, \cdots \) and \( Ws6 = Wt6 \) are true. This kind of word-by-word exact matching can usually provide the best result. However, as a result of the extreme flexibility of natural language, in general, sentences may or may not have some unimportant random words \( wr \), which are introduced into or removed from basic sentences at any time. This may lead to template matching failure due to the low success rate of exactly matching the test sentence with existing templates. As shown in Figure 2, the matching result is not good due to the existence of random words “wr1” and “wr2”. Even if the templates can be created with all the random words, the scale of the templates would be massive, and when the template length becomes longer, the combination explosion problem that is caused by the number of the combinations of many ambiguous words over a vast range is inevitable and is not feasible in reality. We suppose that the effects of the random words could be ignored by skipping them during the matching template. In this case, a longer matching distance and an extremely high matching success rate could be obtained as a result of the flexible word-order natural language. This paper conducted a study based on this assumption. Figure 3, Figure 4 and Figure 5 illustrate the template’s stretchable matching. In Figure 3, by skipping the random words wr1 and wr2 in the test sentence, the template matching is successful. In Figure 4, matching while ignoring the random words wr1 and wr2 in the template is successful. In Figure 5, matching by skipping random words wr1, wr2 and obstructing word Ws6 in the test sentence corresponding to Wt6 in the template is successful. This paper applies the method of SMOST to disambiguate word senses. The contexts to the left and the right of the ambiguous target word are matched using the stretchable matching of the semantic templates in the template bank, and then the matching scores of the contexts to the left and the right are combined to determine the target word’s sense using a maximum score strategy.
3. Relevant works. For word sense disambiguation, existing technologies are as follows: 1) use the word semantic information and the word itself to better cover similar words when the word context may easily lead to data sparsity [4, 25, 11, 35, 36, 17]; 2) introduce the syntactic information of the context in order to obtain the correlation of the collocations more efficiently [24, 33, 36]; 3) adopt the neural network model for the word order instead of an unordered model such as
the bag of words [34, 9, 28, 27, 10]; 4) apply various embedding methods such as word embedding [7], word sense embedding [3, 32] and the context vector [18]; 5) utilize the topic model that identifies the contextual topic so that ambiguous words and inappropriate senses can be filtered out earlier [2][12][30]; 6) employ a graph-based method with the random walk [19, 30, 26]; 7) use the probability weighted voting method with dynamic self-adaptation [13]; 8) use the gloss-augmented neural network (GAS) [14]; 9) apply the co-attention model (CAN) and hierarchical co-attention model (HCAN) [15]; 10) utilize the generative adversarial networks (WSD-GAN) to combine both supervised-based and knowledge-based approaches [5]; and 11) use SyntagNet which is a large-scale manually disambiguated lexical-semantic combination resource [16].

In similar template matching methods, Roberto Navigli and Paola Velardi [21] proposed the SSI (structural semantic interconnection) algorithm, which performs word sense disambiguation using the matching of the semantic templates that are generated by the words and their senses in context-free grammar. The result shows that the SSI has high precision but low recall. Myunggwon Hwang and Pankoo Kim [6] proposed an adapted-relation structure (A-RS) algorithm for noun WSD that takes the SSI [21] as a reference but does not use the various types of RS knowledge-bases and the rich knowledge sources such as WordNet. Jia Ke-liang [8] proposed a method of soft template matching to solve the low recall problem of hard template matching, which only matches at the word granularity level and only at fixed positions. W.K. Chan Samuel [1] proposed a method of semi-supervised WSD using template matching. The templates containing contextual information are created by using a large-scale unlabeled corpus and then correcting the smallest labeled corpus using a sliding window with a fixed size of 3.

4. Method. We proposed a new method that is illustrated from Figure 6 to Figure 12. In Figure 6, there is a test sentence containing several ambiguous words. The words in the test sentence are denoted by \( W \). The words are from \( W_1 \) to \( W_n \). Some are monosemic, such as \( W_{i-3} \), \( W_{i+1} \) and \( W_{i+3} \), and some are ambiguous words, such as \( W_1 \) with two senses \( S_{1(1)} \) and \( S_{1(2)} \) and \( W_{i-4} \) with three senses \( S_{i-4(1)} \), \( S_{i-4(2)} \) and \( S_{i-4(3)} \). \( W_i \) is an ambiguous word that has four senses \( S_{i(1)} \), \( S_{i(2)} \), \( S_{i(3)} \) and \( S_{i(4)} \). The lower part of Figure 7 is a sense-labeled template in the template bank, and the template’s words are denoted by \( TW \). There are words from \( TW_1 \) to \( TW_m \), and each has a word sense from \( S_1 \) to \( S_m \). The word \( TW_k \) in the template is related to the word \( W_i \) that is to be disambiguated in the test sentence.

4.1. Definition of semantic template. A semantic template is a word sense chain that consists of the WordNet senses of the words in a whole/partial labeled sentence. For example, a semantic template “1: 15:00::, 1:21:00::, 2:42:03::, 4:02:00::, 5:00:01:taxable:00” is derived from the WordNet senses of the words in the sentence “United States, income, is, not, subject”. Of course, the template also contains part-of-speech (POS) and lemma information, which is omitted here.

4.2. Working process. Word sense disambiguation, as a whole, is a process of continuously matching the ambiguous words and their senses in the test sentence’s context with those in the template’s context.

1) List all of the senses of each word in the test sentence according to the word sense item in WordNet. For example, in Figure 6, \( W_1 \) has senses \( S_{1(1)} \) and \( S_{1(2)} \), \( W_i \) has \( S_{i(1)} \), \( S_{i(2)} \), \( S_{i(3)} \) and \( S_{i(4)} \), \( W_n \) has \( S_{n(1)} \), \( S_{n(2)} \), \( S_{n(3)} \),
2) Take position $i$, where the word $W_i$ is to be disambiguated, as a central point whose left and right sides have two windows of width $L/2$, respectively, and where $L$ depends on the specific cases. The case of $L/2=4$ is selected as an example in this paper, as shown in Figure 7.

3) Match each word sense on the left side of the word sense $S_k$ in template with all word senses on the left side of the word sense $S_i$ in test sentence, as shown in figure 8. In the same way, match those on the right side of word sense $S_k$. The matching algorithm is given in Algorithm 1 in Figure 13.
4) Filter all node indexes of the matched word sense items in template to the left of word \( W_i \) in descending order. In contrast, filter those to the right of word \( W_i \) in ascending order. For example, the word senses of word \( W_{i-2} \) are arranged in descending order and those of words \( W_{i+2} \) and \( W_{i+4} \) are arranged in ascending order in Figure 9. The ordering operation aims to find the next nearest node from the current node and pass up the others in order to achieve fast, reasonable pruning. This creates the node chain with the most nodes within a certain matching distance and avoids the combination explosion problem, which is generated by a very large number of combinations that involve too many unnecessary nodes within a long sentence.

5) Create one node chain from right to left by finding the node index that is just lower than the current node index starting with \( W_{i-1} \) until word \( W_{i-4} \) (in this example) on the position of the width \( L/2 \). Likewise, do this from left to right. This obtains all the matched word sense node chains that are created by some ordered word sense node index, which are hereinafter referred to as the “node chains”. As shown in Figure 10, two node chains, \( S_{k-3} - S_{k-2} - S_{k-1} \) and \( S_{k-2} - S_{k-1} \), are created on the left side of \( W_i \). For the right side of \( W_i \), the node chains \( S_{k+1} - S_{k+2} - S_{k+3} \) and \( S_{k+3} - S_{k+4} \) are created. A chain from \( W_{i-2} \) to \( W_{i-4} \) bypassing word \( W_{i-3} \) is also shown. In this paper, creating node chain starts from every node in the test sentence to ensure that there is no loss of useful information and to discard the noise words.

6) Calculate the score of each node chain on the left side of word \( W_i \) (refer to the Algorithm 2 in Figure 14) and obtain \( \text{LeftChainMaxScore} \), which is one node chain on the left side with the maximum score, by using the function \( \text{MaxChainScore}() \). Similarly, obtain \( \text{RightChainMaxScore} \), which is the node chain on the right side of word \( W_i \) with the maximum score.

7) Add up the scores of both the left node chain and the right node chain, namely, \( \text{TotalScore}_1 = \text{LeftChainMaxScore} + \text{RightChainMaxScore} \), and obtain the matching score of the first template in the template bank and its corresponding sense item \( S_i \). In Figure 10, the process to obtain \( \text{TotalScore}_1 \) and \( \text{SenseItem}_1 \) is shown. \( \text{SenseItem}_1 \) is the word sense \( S_i \) of word \( TW_k \) in the template, and it is matched with \( S_{i(2)} \), which is one of several word senses of word \( W_i \) in the test sentence.

8) Follow the above steps to match all templates in the template bank to get all match results.

9) Filter and rank all the scores of the node chains that are created by the matched templates for word \( W_i \). Take the node chains with the maximum scores \( \text{Top1-TopX} \) and their corresponding sense items as candidates.
10) Using the optimal decision-making algorithm (refer to Algorithm 3 in Figure 15), choose the sense item with the highest score as the sense item for word $W_i$. Figure 11 illustrates the process of selecting the final sense for word $W_i$.

In particular, when no template corresponding to the target word in the template bank, obtain template through the word sense of target word to continue for disambiguation, which is shown in Figure 12. In addition, choose the most frequent
word sense of the target word as the final result when no template is still found by
the word sense of target word.

5. **Algorithms.** This paper applies different algorithms in various processes of word sense disambiguation including punctuation processing, matching a sense of word (Algorithm 1), obtaining the score of a matched node chain (Algorithm 2) and obtaining the final word sense (Algorithm 3).

5.1. **Punctuation processing.** Sometimes, punctuation is a mark that separates a sentence into multiple parts. To improperly combine the contextual information before or after punctuation can sometimes introduce incorrect information into matching. Therefore, there is a rule in this paper to stop matching word senses when some punctuation is detected, which may result in not reaching the specified window width $L/2$.

5.2. **Matching a sense of word.** Plenty of factors should be taken into consideration for template matching, including words, lemmas (lemmatization), POSs, and word senses. In WordNet, a word sense has three-level sub-codes. For example, a word sense “1:06:02::” has three-level sub-codes that are WNsub1(“1”), WNsub2(“06”) and WNsub3(“02”) respectively. By applying different weights, the total score can be calculated, as shown in Figure 13.

**Algorithm 1. Matching a Sense of Word**

```plaintext
1 Initialization: OneWordSenseScore ← 0
2 Get word sense information of both the words in template and the words in test sentence.
3 Test sentence data: Word Test, POS Test, Lemma Test, WN sub1 Test, WN sub2 Test, WN sub3 Test.
4 Template sentence data: Word Bank, POS Bank, Lemma Bank, WN sub1 Bank, WN sub2 Bank, WN sub3 Bank.
5 if POS Test = POS Bank, then
6 Set word match result: WordMatchScore ← 1
7 if Lemma Test = Lemma Bank, then
8 Set lemma match result: LemmaMatchScore ← 1
9 if WN sub1 Test = WN sub1 Bank, then
10 Set WN sub1 match result: WN sub1MatchScore ← 1
11 if WN sub2 Test = WN sub2 Bank, then
12 Set WN sub2 match result: WN sub2MatchScore ← 1
13 if WN sub3 Test = WN sub3 Bank, then
14 Set WN sub3 match result: WN sub3MatchScore ← 1
15 Clear WN sub3 match result: WN sub3MatchScore ← 0
16 goto 22
17 else
18 return OneWordSenseScore
19 else
20 return OneWordSenseScore
21 else
22 Clear lemma match result: LemmaMatchScore ← 0
23 goto 8
24 else
25 Clear word match result: WordMatchScore ← 0
26 goto 6
27 else
28 return OneWordSenseScore
29 Get word sense match score: OneWordSenseScore ← WordMatchScore * LemmaMatchScore * LemmaWeight + 
   WN sub1MatchScore * WN sub1Weight + WN sub2MatchScore * WN sub2Weight + WN sub3MatchScore * WN sub3Weight
30 return OneWordSenseScore
```

**Figure 13. Matching a Sense of Word (Algorithm 1)**
5.3. **Obtaining the score of a matched node chain.** The interval gap algorithm is introduced to only select the nodes whose distance is within the limited gap between two nodes. The closer the nodes are to the target word, the higher the scores are, as shown in Figure 14.

5.4. **Obtaining the final word sense.** The final result from candidate Top1 to TopX are selected, as shown in Figure 15.

---

**Algorithm 2. Obtaining the Score of a Matched Node Chain**

```plaintext
1 Initialization: Node_Last ← 0; Node_n ← FirstNode; TotalFinalScore ← 0;
2 Get the gap between two nodes: Gap(n) ← Position(Node_n) − Position(Node_Last)
3 if Gap(n) < MaxGapLimit, then
4 Get the node distance weight according to the distance to target word:
   DistanceWeight(n) ← (1.0 − Position(Node_n) / (MaxLimitLength + 1))^k
5 Get Node Score using distance weight:
   FinalScore (Node_n) ← MatchScore (Node_n) * DistanceWeight (n)
6 Update total score of matched semantic chain:
   TotalFinalScore ← TotalFinalScore + FinalScore(Node_n)
7 if Node_n = EndNode, then
8   return TotalFinalScore;
9 else
10   set next node: Node_Last ← Node_n; Node_n ← NextNode;
11   goto 2;
12 else
13   return TotalFinalScore;
```

**Figure 14.** Obtaining the Score of a Matched Node Chain (Algorithm 2)

6. **Experiments and discussion.** This paper has conducted some experiments using the Unified Evaluation Framework [29] and the resource provided by the website http://lcl.uniroma1.it/wsdeval. Firstly, the experimental results using different algorithms is listed in Table 1. P1 and P2 are two different parameter configurations. Among the results, the ones with the Max. score, the ones with the Max. vote and the ones with the Max. of score * vote were listed respectively. The results show that the ones with the Max. score is the worst and there are some small differences between the ones with the Max. vote and the ones with the Max. of score * vote. This result shows that the effect of voting cannot be ignored. Secondly, comparing the results of other existing systems is listed in Table 2. Since our system extracts semantic templates from only the SemCor corpus to disambiguate word senses, we chose the comparative data from the paper [28] and [29], and took the system It Makes Sense (IMS) [37] as a baseline, which is a supervised learning based WSD system by training only the SemCor corpus. The result of our system is close to the result of IMS. This result shows our new method is effective. Thirdly, comparing the experimental results of the systems using template matching method is listed in Table 3. As the result of other two systems [8] and [1] using template matching method are on Chinese test corpus, they cannot be directly compared here, so they are not listed. In addition, since comparable systems used the earlier Senseval-3 test sets rather than the new version of the Unified Evaluation Framework [29], we experimented with the same earlier
Algorithm 3. Obtaining the Final Word Sense

1. Initialization: set Top1, Top2, ..., TopX to 0; set MatchedChainPointer to 1.
2. Get the score and word sense of one matched semantic chain by MatchedChainPointer.
   For example,
   TotalFinalScore(MatchedChainPointer) = 130; WN(MatchedChainPointer) = 1.27:00:
3. Rank chain scores with word senses to get the top1 to topX.
   For example,
   1) Top1 = 200; WN(1) = 1:09:00::
   2) Top2 = 168; WN(2) = 1:06:02::
   3) Top3 = 118; WN(3) = 1:09:00::
   ... ...
   10) TopX = 40; WN(X) = 1:04:00::
4. if MatchedChainPointer = END, then
5.   Get the final word sense according to the max score and votes of word sense.
   For example,
   WN(1) = 1:09:00:: Score_1 = 200 * 2; MaxScore = 200, vote = 2;
   WN(2) = 1:06:02:: Score_2 = 168 * 1; MaxScore = 168, vote = 1;
   ... ...
   WN(X) = 1:04:00:: Score _X= 40 * 1; MaxScore = 40, vote = 1;
6. return WN(itemwithMaxScore(Score_1, Score_2, ..., Score_X));
7. else
8.   Set the next pointer of matched semantic chain: MatchedChainPointer ++;
9.   goto 2

Figure 15. Obtaining the Final Word Sense (Algorithm 3)

resources including SemCor2.1 and WordNet2.1. Experimental results show that our approach still has advantages over their systems.

Table 1. Comparison of F1 scores on our systems with different algorithms on five test sets

| Res.   | Different algorithms       | Sen2 | Sen3 | Sem07 | Sem13 | Sem15 |
|--------|---------------------------|------|------|-------|-------|-------|
|        | SMOST Max.score P1         | 65.8 | 63.9 | 57.6  | 62.0  | 65.6  |
|        | SMOST Max.score P2         | 66.3 | 64.6 | 57.8  | 61.7  | 65.5  |
|        | SMOST Max.vote P1          | 68.0 | 67.9 | 59.8  | 64.2  | **70.0** |
|        | SMOST Max.vote P2          | 68.8 | **68.3** | 60.2 | 64.2  | 67.5  |
|        | SMOST Max.vote*score P1    | 67.7 | 67.1 | 58.9  | **64.7** | 69.2  |
|        | SMOST Max.vote*score P2    | **68.9** | **68.0** | **61.1** | 64.4  | 66.6  |

Table 2. Comparison of F1 scores on several systems using supervised learning method on five test sets

| Res.   | System                     | Sen2 | Sen3 | Sem07 | Sem13 | Sem15 |
|--------|----------------------------|------|------|-------|-------|-------|
|        | MFS                        | 65.6 | 66.0 | 54.5  | 63.8  | 67.1  |
|        | IMS baseline(Zhong2010)    | 70.9 | 69.3 | 61.3  | 65.3  | 69.5  |
| SemCor | BLSTM(Raganato2017)        | 71.4 | 68.8 | 61.8  | 65.6  | 69.2  |
| 3.0    | Seq2Seq(Raganato2017)      | 68.5 | 67.9 | 60.9  | 64.3  | 67.3  |
|        | SMOST (this paper)         | **68.9** | **68.3** | **61.1** | **64.7** | **70.0** |
Table 3. Comparison of F1 scores on the systems using template matching method on Sen3 test set

| Resource       | System                                      | Recall | Precision | F1    |
|----------------|---------------------------------------------|--------|-----------|-------|
| multi-res.     | SSI (Navigli2004)                          | 68.40  | 68.50     | 68.45 |
|                | SSI-10words context (Hwang2008)            | 90.96  | 57.30     | 70.31 |
| SemCor2.1      | A-RS-10words context(Hwang2008)            | 56.80  | 75.53     | 64.84 |
| +WordNet2.1    | SMOST (this paper)                         | 100.0  | 59.84     | 74.87 |

7. Conclusion. The stretchable matching method proposed in this paper has been applied to word sense disambiguation and has achieved a good effect, indicating the application value from our method because semantic template matching has high matching coverage, good interpretability and no training required. This method breaks through the limitation encountered in the existing template that can only be hard matched in a fixed position; moreover, it can bypass some obstacle nodes to continue to achieve longer matching, such as matching a whole sentence. Furthermore, it can obtain a more flexible matching degree than the word template by combining the elastic multilevel word sense matching. Through the unique node index filtering method, the combination explosion problem can be prevented when there are too many ambiguous words in long sentences, which ensures that long sentence matching can be well implemented. Judging from the results of the word sense disambiguation, it should have favorable universality and could be applied to other tasks in NLP.

REFERENCES

[1] S. W. K. Chan, Generating context templates for word sense disambiguation, AI 2013: Advances in Artificial Intelligence, 8272 (2013), 466–477.
[2] D. S. Chaplot and R. Salakhutdinov, Knowledge-based word sense disambiguation using topic models, 32nd AAAI Conference on Artificial Intelligence (AAAI-18), (2018), 1–8.
[3] X. X. Chen, Z. Y. Liu and M. S. Sun, A United Model for Word Sense Representation and Disambiguation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014 (2014), 1025–1035.
[4] J. P. Chen and W. Yu, Enriching semantic knowledge for WSD, IEICE Trans, E97-D (2014), 2212–2216.
[5] Z. Hu, F. Luo and Y. Tan, et al, WSD-GAN: Word sense disambiguation using generative adversarial networks, The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), 33 (2019), 9943–9944.
[6] M. Hwang and P. Kim, Adapted relation structure algorithm for word sense disambiguation, Proceedings of Third IEEE International Conference on Digital Information Management (ICDIM), 2008 (2008), 684–688.
[7] I. Iacobacci, M. T. Pilehvar and R.Navigli, Embeddings for Word Sense Disambiguation: An Evaluation Study, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, (2016), 897–907.
[8] K. L. Jia, Research of word sense disambiguation based on soft pattern, Key Engineering Materials, 460/461 (2011), 130–135.
[9] M. Kageback and H. Salomonsson, Word sense disambiguation using a bidirectional LSTM, Proceedings of the Workshop on Cognitive Aspects of the Lexicon, (2016), 51–56.
[10] M. Le, M. Postma, J. Urbani and P. Vossen, A deep dive into word sense disambiguation with LSTM, Proceedings of the 27th International Conference on Computational Linguistics, 2018, 354–365.
[11] L. Li and Q. Zhou, Chinese word sense disambiguation based on lexical semantic ontology, Journal of Chinese Language and Computing, 18 (2018), 13–23.
[12] L. L. Li, B. Roth and C. Sporleder, Topic Models for Word Sense Disambiguation and Token-based Idiom Detection, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, (2010), 1138–1147.
[13] W. P. Lu, H. Wu, F. Jian, Y. G. Huang and H. Y. Huang, An empirical study of classifier combination based word sense disambiguation, IEICE Trans, E101-D (2018), 225–233.
[14] F. Luo, T. Liu, Q. Xia and B. Chang, et al., Incorporating glosses into neural word sense disambiguation, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, (2018), 2473–2482.
[15] F. Luo, T. Liu and Z. He, et al., Leveraging gloss knowledge in neural word sense disambiguation by hierarchical Co-attention, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, (2018), 1402–1411.
[16] M. Maru, F. Scozzafava and F. Martelli, et al., SyntagNet: Challenging supervised word sense disambiguation with lexical-semantic combinations, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), (2019), 3532–3538.
[17] S. Melacci, A. Globo and L. Rigutini, Enhancing modern supervised word sense disambiguation models by semantic lexical resources, 2018 LREC, (2018), 1012–1017.
[18] O. Melamud, J. Goldberger and I. Dagan, context2vec: Learning generic context embedding with bidirectional lstm, Proceedings of the 20th SIGNLL onference on Computational Natural Language Learning (CoNLL), (2016), 51–61.
[19] A. Moro, A. Raganato and R.Navigli, Entity linking meets word sense disambiguation: A united approach, Transactions of the Association for Computational Linguistics, 2 (2014), 231–244.
[20] R. Navigli, Word sense disambiguation: A survey, ACM Computing Surveys, 41 (2009), 1–69.
[21] R. Navigli and P. Velardi, Structural semantic interconnection: A knowledge-based approach to word sense disambiguation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005), 1075–1086.
[22] A. R. Pal and D. Saha, Word sense disambiguation: A survey, International Journal of Control Theory and Computer Modeling (IJCTCM), 5 (2015), 1–16.
[23] A. Panchenko, S. Faralli, S. P. Ponzetto and C. Biemann, Using linked disambiguated distributional networks for word sense disambiguation, Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, (2017), 72–78.
[24] S. Papandrea, A. Raganato and C. D. Bovi, SUPWSD: A flexible toolkit for supervised word sense disambiguation, Proceedings of the 2017 EMNLP System Demonstrations, Association for Computational Linguistics, (2017), 103–108.
[25] T. Pasini and R. Navigli, Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, (2017), 78–88.
[26] L.N. Pina and R. Johansson, Embedding Senses for Efficient Graph-based Word Sense Disambiguation, Proceedings of the 2016 Workshop on Graph-based Methods for Natural Language Processing, NAACL-HLT 2016, (2016), 1–5.
[27] A. Popov, Word sense disambiguation with recurrent neural networks, Proceedings of the Student Research Workshop associated with RANLP 2017, (2017), 25–34.
[28] A. Raganato, C. D. Bovi and R. Navigli, Neural sequence learning models for word sense disambiguation, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, (2017), 1156–1167.
[29] A. Raganato, J. Camacho-Collados and R. Navigli, Word sense disambiguation: A unified evaluation framework and empirical comparison, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, (2017), 99–110.
[30] B. S. Rintyarna and R. Sarno, Topic models for word sense disambiguation and token-based idiom detection, 2016 Fourth International Conference on Information and Communication Technologies (ICOICT), (2016), 1–5.
[31] F. Tacoa, D. Bollegala and M. Ishizuka, A context expansion method for supervised word sense disambiguation, 2012 IEEE Sixth International Conference on Semantic Computing, (2012), 339–341.
[32] A. Trask, P. Michalak and J. Liu, SENSE2VEC - A fast and accurate method for word sense disambiguation in neural word embeddings, arXiv:1511.06388, Under Review as a Conference Paper at ICLR 2016, (2015), 1–9.
[33] X. D. Wang, X. R. Tang, W. G. Qu and M. Gu, Word sense disambiguation by semantic inference, 2017 International Conference on Behavioral, Economic, Socio-cultural Computing, (2017), 1–6.

[34] D. Y. Yuan, J. Richardson, R. Doherty, C. Evans and E. Altendorf, Semi-supervised Word Sense Disambiguation with Neural Models, Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, (2016), 1374–1385.

[35] C. X. Zhang, L. R. Sun, X. Y. Gao, Z. M. Lu and Y. Yue, Integrate chinese semantic knowledge into word sense disambiguation, International Journal of Hybrid Information Technology, 8 (2015), 105–116.

[36] C. X. Zhang, L. R. Sun and X. Y. Gao, Determine word sense based on semantic and syntax information, International Journal of Database and Theory and Application, 9 (2016), 17–22.

[37] Z. Zhong and H. T. Ng, It makes sense: A wide-coverage word sense disambiguation system for free text, Proceedings of the ACL 2010 System Demonstrations, (2010), 78–83.

Received January 2020; revised August 2020.

E-mail address: dalianwangwei@foxmail.com
E-mail address: huangdg@dlut.edu.cn
E-mail address: yuhaitao@slis.tsukuba.ac.jp