Word Sense Disambiguation via PropStore and OntoNotes for Event Mention Detection

Nicolas Fauceglia, Yiu-Chang Lin, Xuezhe Ma, and **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
USA
{fauceglia, yiuchanl, xuezhem, hovy}@cs.cmu.edu

Abstract

In this paper, we propose a novel approach for Word Sense Disambiguation (WSD) of verbs that can be applied directly in the event mention detection task to classify event types. By using the PropStore, a database of relations between words, our approach disambiguates senses of verbs by utilizing the information of verbs that appear in similar syntactic contexts. Importantly, the resource our approach requires is only a word sense dictionary, without any annotated sentences or structures and relations between different senses (as in WordNet). Our approach can be extended to disambiguate senses of words for parts of speech besides verbs.

1 Introduction

The task of Word Sense Disambiguation (WSD) is to identify the correct meaning or sense of an ambiguous word in a given context. As a fundamental task in natural language processing (NLP), WSD can benefit applications such as machine translation (Chan et al., 2007) and information retrieval (Stokoe et al., 2003). Most of the top performance WSD systems (Agirre and Soroa, 2009; Zhong and Ng, 2010), however, rely on manually annotated data or on lexical knowledge bases (e.g., WordNet), which are highly expensive to create.

In this paper, we propose a novel approach for Word Sense Disambiguation of verbs using the PropStore. With the help of PropStore, our approach can utilize information about verbs' appearance in syntactic contexts similar to the target sentence. This information significantly enriches the given contexts, and makes our approach obviate the need for annotated data and knowledge bases. The only resource our approach requires is a word sense dictionary that defines the senses and their descriptions for each word. Obviously, this dictionary is much easier to acquire than resources such as annotated data or Wordnet. Moreover, our approach can be extended to disambiguate senses of words in other types of part-of-speech. We demonstrate in this paper how our WSD method can be applied to the event mention detection task to classify event types.

1.1 Event Mention Detection Task

Event understanding has recently attracted a lot of attention[1]. A fundamental task in event understanding is to conduct Event Mention Detection (EMD). The EMD task requires a system to identify text spans in which events are mentioned, and to provide their attributes. The major attribute studied in recent EMD tasks (Li et al., 2013; Li et al., 2014) is *event mention type*, which is one of the most salient attributes relating to its semantics. In this paper, we propose a novel method on identifying event mention types. In particular, we focus on one major source of event mentions: verb-based mentions.

Given a list of possible candidates, the event mention detection task consists in identifying the type of each candidate mention (being one of the predefined event types or other). In this paper, we simply regard all verbs as mention candidates. In this setting, event mention detection can be cast as a verb sense disambiguation task, where the target senses are simply event types. We argue that our

[1] https://sites.google.com/site/wsevents2014/home

Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015, pages 11–15,
Denver, Colorado, June 4, 2015.

method is especially suitable for this task because it naturally captures argument information (which is proven to be important in previous tasks) in a distributional manner.

2 The PropStore

The PropStore is a proposition knowledge base, essentially a triple store implemented as a database of relations between words, created using Wikipedia articles.

Each relation is represented in the form of a triple of two words connected by a relation:

$$< word_1, relation, word_2 >$$

Each word is an instance in the PropStore dictionary, and consists of its original form, as present in the text[2], the POS, lemma, and language. Each triple can occur one or thousands of times in the corpus. For each occurrence of a triple in a sentence, we store a new entry in the PropStore with that information.

The PropStore supports different types of relations, including dependency, semantic role, etc., and for each type, many values, including *nsubj*, *dobj*, etc. The current implementation of the PropStore uses just a single type of relation: dependency.

The source of the triples is every Wikipedia article available for each supported language. Each article is parsed and POS tagged using the Fanse Dependency Parser (Tratz and Hovy, 2011). For each triple occurrence in the corpus, we store the source article title, the sentence number, and the position of the child word in the sentence. This way, for every occurrence of a triple within a sentence, we can re-build the sentence, and also we can distinguish between two occurrences of the same triple in a sentence, allowing us to chain two or more triples in a tree configuration.

With this structure we can query the database to retrieve, for example, all sentences with a particular relation configuration; all verbs which have a particular *dobj*; all subjects of a given verb; two or more siblings of a shared head; or more complicated configurations, with their frequencies.

[2]except for normalized expressions such as numbers, punctuation and foreign words

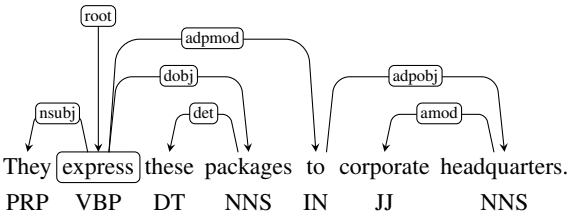

Figure 1: Dependency tree and POS tags for the example sentence.

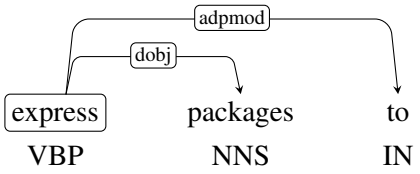

Figure 2: $Sig(express, X)$

Previous work us=ed a similar PropStore approach to build a Structured Distributional Semantics Model for event coreference (Goyal et al., 2013).

3 Our Approach for WSD

In the following, we use $X = x_1, x_2, \ldots, x_n$ to denote a generic sentence. For a given sentence X and a word $x \in X$ which we want to disambiguate, we define the signature of x in X, $Sig(x, X)$, as a "small part" of the dependency parse of X including x. For example, the sentence *They express these packages to corporate headquarters* is shown in Figure 1 along with its dependency tree, and Figure 2 gives the signature of *express*, $Sig(express, X)$, in this sentence.

Suppose x has m different senses in a dictionary (e.g., OntoNotes), $v_1, \ldots, v_m$, our task is to predict the correct sense of x in the given sentence X. This is done by selecting the sense with the highest score:

$$v^* = \underset{v}{\operatorname{argmax}}\, score(v, x, X)$$

To simplify the model, we restrict the score function only to the signature of x:

$$score(v, x, X) = score(v, Sig(x, X))$$

which means we only use the context information within the signature of $Sig(x, X)$, ignoring other information.

3.1 WSD Algorithm via PropStore

The intuition of our approach is that verbs which appear in the same signature should have similar senses. Based on this assumption, we can define the score function $score(v, Sig(x, X))$ by two steps: first querying PropStore to collect all the verbs that appear in the same signature of $Sig(x, X)$; second defining the similarity measure for two words: $sim(x_1, x_2)$.

Specifically, to disambiguate verb $x \in X$, we first query PropStore to get the list of verb candidates:

$$W(Sig(x, X)) = \{w : Sig(w, X) \in PropStore\}$$

Here $W(Sig(x, X))$ is the set of all the verbs which appear in the same signature $Sig(x, X)$. Besides the verb list, we can also get the weight (frequency) $\theta(w)$ of each verb $w \in W(Sig(x, X))$ from PropStore.

With the list of verb candidates and their weights, we can define the score function as follows:

$$score(v, Sig(x, X)) = \sum_{w \in W} Sim(v, w)\theta(w)$$

where $Sim(v, w)$ is a function to measure the similarity between a verb w and a word sense v of x.

To define this similarity function $Sim(v, w)$, we utilize the short description of word sense v in the dictionary. We extract all the verbs in the short description of word sense v and denote it as $W(v)$. Then

$$Sim(v, w) = \frac{1}{|W(v)|} \sum_{w' \in W(v)} sim(w, w')$$

where $sim(w_1, w_2)$ is the similarity function between two verbs. To summarize, we define the similarity between a verb and a word sense as the average similarities between the verb and all the verbs which appear in the short description of this word sense.

Now, there are three remaining problems to resolve to complete the WSD algorithm:

1. How to extract signature structure $Sig(x, X)$ for the verb x in the given sentence X.
2. How to query PropStore to obtain the set of verb candidates $W(Sig(x, X))$.
3. How to measure similarity between two verbs $sim(w_1, w_2)$.

3.1.1 Extract Signature

For the first problem, the signature of a word is extracted by applying syntactic rules. Currently, we only extract the objects and prepositional modifiers (if any exist) of the verb we want to disambiguate. In the examples shown in Section 4, the signatures extracted by our simple rules perform well.

3.1.2 PropStore Query

For the second one, we query PropStore with $Sig(x, X)$ to get the lemmas of all the verbs that occur in the same signature structure as the target one. After querying PropStore, it returns a list of top k candidate words (verbs) $W = \{w_1, w_2, ..., w_k\}$ with their corresponding frequency of occurrence in descending order. For example, for the *head-and-children* template, which consists of a target head node, and two or more children, linked through a relation, we should formulate the query as follows:

```
rel1 = ('dobj', ('N', 'package'))
rel2 = ('adpmod', ('IN', 'to'))
sig = head_and_children('V', rel1, rel2)
verbs = propstore.query(sig)
```

Then we obtain a list of verbs occurring in contexts with 'package' (POS: N) as direct object and a 'adpmod' dependency relation pointing to 'to' (POS: IN) along with their frequencies.

3.1.3 Word Vectors

To measure the similarity between two words, we compute the distance between their corresponding word vectors which are trained by the word2vec continuous bag of words model (Mikolov et al., 2013). For training, we ran 15 iterations for vectors with 50 dimensions and a window size of 8, with 25 negative examples and the downsampling threshold being $1e^{-4}$. Slightly different from typical training methods, we treat the same word with different POS tags as different words so they do not share the same vector. In other words,

mail.noun and **mail.verb** are two different vectors instead of one. This is reasonable for doing WSD because distinct POS implies distinct senses. Accordingly, the distance between two words can be calculated by their Euclidean distance in the vector space:

$$dist\,(w_1, w_2) = \|\mathbf{v_1} - \mathbf{v_2}\|$$

and the similarity can be defined as the negative of distance:

$$sim(w_1, w_2) = -dist\,(w_1, w_2)$$

4 Example Results

In this section, we provide three example sentences to illustrate our WSD approach and show the corresponding result. From OntoNotes, *"express-v"* has the following three senses:

- "convey, show, state in some form"
- "press out physically"
- "to mail or post something via a rapid method"

An example sentence for the third sense is *"X = They* **express** *these packages to corporate headquarters."*. Its signature, $Sig(express, X)$, was previously shown in Fig 1 and Fig 2. The signature is composed of two *triples*, $< express, dobj, packages >$ and $< express, adpmod, to >$ with their first words anchored together. We then query the PropStore to get the set of candidate verbs $W(Sig(express, X))$ and their corresponding weight (frequency) $\theta(w)$ for each verb $w \in W(Sig(express, X))$ in descending order.

The resulting top 5 words and their weights are shown in Table 1. The most frequent word in PropStore that occurs in the same signature as $Sig(express, X)$ is *"deliver"*, which does make sense because *"deliver packages to"* is a common usage and it provides hints to disambiguate the sense of *"express"*. (*"deliver"* here is semantically closer to sense3 than the others.)

After applying the WSD algorithm mentioned in Sec 3.1, we obtain the result shown in Table 2.

word	frequency
deliver	76
offer	35
provide	28
send	25
sell	20

Table 1: Top 5 words in $W(Sig(express, X))$ and their frequency.

v_i	-Score(v_i)
sense1	9922.32
sense2	10069.5
sense3	9236.79

Table 2: $-Score(v_i)$ obtained from the WSD algorithm.

We use the value of $-Score(s_i)$ for reading convenience. Therefore, the best sense is given by the lowest score. In this example sentence, the best sense for *"express"* is sense3.

Another example sentence for the first sense of $express$ is *"X' = Picasso's Guernica vividly* **expresses** *the horrors of war."*. The signature and WSD results are shown in Fig 3, Table 3 and Table 4, respectively.

The last example is *"X" = She* **pronounced** *her first syllables at six months."*, where *"pronounce"* is the word to disambiguate. From OntoNotes, *"pronounce-v"* has two senses: "utter in a certain way"(sense1) and "pronounce judgement on"(sense2). Fig 4, Table 5 and Table 6 shows the corresponding siganture and results.

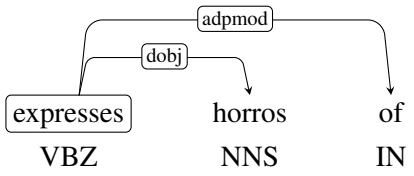

Figure 3: $Sig(express, X')$

word	frequency
know	3
demonstrate	1

Table 3: Top 5 words in $W(Sig(express, X'))$ and their frequency.

v_i	-Score(v_i)
sense1	202.49
sense2	216.91
sense3	206.43

Table 4: $-Score(v_i)$ obtained from our WSD algorithm.

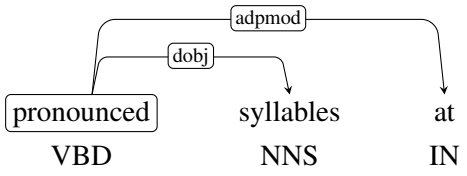

Figure 4: $Sig(pronounce, X'')$

word	frequency
have	3
leave	2
change	1
add	1
use	1

Table 5: Top 5 words in $W(Sig(pronounce, X''))$ and their frequency.

v_i	-Score(v_i)
sense1	343.36
sense2	426.71

Table 6: $-Score(v_i)$ obtained from our WSD algorithm.

5 Conclusion

In this paper, we propose a approach for Word Sense Disambiguation (WSD) of verbs using PropStore. Our approach does not require any annotated data or lexical knowledge base except an word sense dictionary. From the examples we showed in this paper, our approach can successfully disambiguate the senses of verbs *express* even when the hints from the given contexts are weak. Our approach can disambiguate senses for other POSs, too. Moreover, we described how our approach can be applied to event mention detection task to classify mention types.

There is a wide range of possible future work. First, we will build an automated system to perform all the steps together. Second, we will evaluate our approach for WSD on benchmark data sets, such as OntoNotes, and compare with current top WSD systems. At last, we will apply our approach to some really semantic tasks like event mention detection and event coreference resolution.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 33–41, Athens, Greece, March.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 33. Citeseer.

Kartik Goyal, Sujay Kumar Jauhar, Huiying Li, Mrinmaya Sachan, Shashank Srivastava, and Eduard Hovy. 2013. A structured distributional semantic model for event co-reference. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 467–473, Sofia, Bulgaria, August.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing Information Networks Using One Single Model. In *Proceedings the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP2014)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Christopher Stokoe, Michael P. Oakes, and John Tait. 2003. Word sense disambiguation in information retrieval revisited. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR 03*, pages 159–166, New York, NY, USA. ACM.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden, July.