

# Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing \*

Hai Zhao(赵海)<sup>†</sup>, Wenliang Chen(陈文亮)<sup>‡</sup>, Chunyu Kit<sup>†</sup>, Guodong Zhou\*

<sup>†</sup>Department of Chinese, Translation and Linguistics

City University of Hong Kong

83 Tat Chee Avenue, Kowloon, Hong Kong, China

<sup>‡</sup>Language Infrastructure Group, MASTAR Project

National Institute of Information and Communications Technology

3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289

\*School of Computer Science and Technology

Soochow University, Suzhou, China 215006

haizhao@cityu.edu.hk, chenwl@nict.go.jp

## Abstract

This paper describes our system about multilingual semantic dependency parsing (SR-Only) for our participation in the shared task of CoNLL-2009. We illustrate that semantic dependency parsing can be transformed into a word-pair classification problem and implemented as a single-stage machine learning system. For each input corpus, a large scale feature engineering is conducted to select the best fit feature template set incorporated with a proper argument pruning strategy. The system achieved the top average score in the closed challenge: 80.47% semantic labeled F1 for the average score.

## 1 Introduction

The syntactic and semantic dependency parsing in multiple languages introduced by the shared task of CoNLL-2009 is an extension of the CoNLL-2008 shared task (Hajič et al., 2009). Seven languages, English plus Catalan, Chinese, Czech, German, Japanese and Spanish, are involved (Taulé et al., 2008; Palmer and Xue, 2009; Hajič et al., 2006; Surdeanu et al., 2008; Burchardt et al., 2006; Kawahara et al., 2002). This paper presents our research for participation in the semantic-only (SROnly) challenge of the CoNLL-2009 shared task, with a

---

This study is partially supported by CERG grant 9040861 (CityU 1318/03H), CityU Strategic Research Grant 7002037, Projects 60673041 and 60873041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the "863" National High-Tech Research and Development of China.

highlight on our strategy to select features from a large candidate set for maximum entropy learning.

## 2 System Survey

We opt for the maximum entropy model with Gaussian prior as our learning model for all classification subtasks in the shared task. Our implementation of the model adopts L-BFGS algorithm for parameter optimization as usual. No additional feature selection techniques are applied.

Our system is basically improved from its early version for CoNLL-2008 (Zhao and Kit, 2008). By introducing a virtual root for every predicates, The job to determine both argument labels and predicate senses is formulated as a word-pair classification task in four languages, namely, Catalan, Spanish, Czech and Japanese. In other three languages, Chinese, English and German, a predicate sense classifier is individually trained before argument label classification<sup>1</sup>. Note that traditionally (or you may say that most semantic parsing system will do so) argument identification and classification are handled in a two-stage pipeline, while ours always tackles them in one step, in addition, predicate sense classification are also included in this unique learning/test step for four of all languages.

---

<sup>1</sup>At first, we adopted the same learning framework for these three languages as the other four. However, during the feature template selection that will be described in Section 5, we observed that a few features for these three languages obviously did good to sense classification but did harm to argument identification/classification, or vice versa. This fact suggested that sense classification and argument identification/classification should use quite different feature sets. That is why we finally use an individual sense classifier for these three languages.

### 3 Argument Pruning

We keep using a word-pair classification procedure to formulate semantic dependency parsing. Specifically, we specify the first word in a word pair as a predicate candidate (i.e., a semantic head, and noted as  $p$  in our feature representation) and the next as an argument candidate (i.e., a semantic dependent, and noted as  $a$ ). We do not differentiate between verbal and non-verbal predicates and our system handles them in the exactly same way.

When no constraint available, however, all word pairs in the an input sequence must be considered, leading to very poor efficiency in computation for no gain in effectiveness. Thus, the training sample needs to be pruned properly. As predicates overtly known in the share task, we only consider how to effectively prune argument candidates.

We adopt five types of argument pruning strategies for seven languages. All of them assume that a syntactic dependency parsing tree is available.

As for Chinese and English, we continue to use a dependency version of the pruning algorithm of (Xue and Palmer, 2004) as described in (Zhao and Kit, 2008). The pruning algorithm is readdressed as the following.

Initialization: Set the given predicate candidate as the current node;

- (1) The current node and all of its syntactic children are selected as argument candidates.
- (2) Reset the current node to its syntactic head and repeat step (1) until the root is reached.

Note that the given predicate candidate itself is excluded from the argument candidate list for Chinese, that is slightly different from English.

The above pruning algorithm has been shown effective. However, it is still inefficient for a single-stage argument identification/classification classification task. Thus we introduce a virtual argument label ‘\_NoMoreArgument’ to alleviate this difficulty. If an argument candidate in the above algorithm is labeled as such a label, then the pruning algorithm will end immediately. In training, this assistant virtual label means no more samples will be generated for the current predicate, while in test, the decoder will not search more argument candidates any more.

This adaptive technique more effectively prunes the argument candidates. In fact, our experiments show 1/3 training memory and time may be saved from it.

As for Catalan and Spanish, only syntactic children of the predicate are considered as the argument candidates.

As for Czech, only syntactic children, grandchildren, great-grandchildren, parent and siblings of the predicate are taken as the argument candidates.

As for German, only syntactic children, grandchildren, parent, siblings, siblings of parent and siblings of grandparent of the predicate are taken as the argument candidates.

The case is somewhat sophisticated for Japanese. As we cannot identify a group of simple predicate-argument relations from the syntactic tree. Thus we consider top frequent 28 syntactic relations between the predicate and the argument. The parser will search all words before and after the predicate, and only those words that hold one of the 28 syntactic relations to the predicate are considered as the argument candidate. Similar to the pruning algorithm for Chinese/English/German, we also introduce two virtual labels ‘\_leftNoMoreArgument’ and ‘\_rightNoMoreArgument’ to adaptively prune words too far away from the predicate.

### 4 Feature Templates

As we don’t think that we can benefit from knowing seven languages, an automatic feature template selection is conducted for each language.

About 1000 feature templates (hereafter this template set is referred to  $FT$ ) are initially considered. These feature templates are from various combinations or integrations of the following basic elements.

**Word Property.** This type of elements include word *form*, *lemma*, part-of-speech tag (*PoS*), *FEAT* (additional morphological features), syntactic dependency label (*dprel*), semantic dependency label (*sem

```
l
```*) and characters (*char*) in the word form (only suitable for Chinese and Japanese)<sup>2</sup>.

**Syntactic Connection.** This includes syntactic head (*h*), left(right) farthest(nearest) child (*lm, ln, rm, rn*), and high(low) support verb or noun. We explain the last item, support verb(noun).

<sup>2</sup>All lemmas, PoS, and *FEAT* for either training or test are from automatically pre-analyzed columns of every input files.

| FEAT $n$        | 1       | 2   | 3   | 4      | 5    | 6     | 7     | 8   | 9   | 10  | 11  |
|-----------------|---------|-----|-----|--------|------|-------|-------|-----|-----|-----|-----|
| Catalan/Spanish | postype | gen | num | person | mood | tense | punct |     |     |     |     |
| Czech           | SubPOS  | Gen | Num | Cas    | Neg  | Gra   | Voi   | Var | Sem | Per | Ten |

Table 1: Notations of FEATs

From the predicate or the argument to the syntactic root along with the syntactic tree, the first verb(noun) that is met is called as the low support verb(noun), and the nearest one to the root is called as the high support verb(noun).

**Semantic Connection.** This includes semantic head (*semhead*), left(right) farthest(nearest) semantic child (*sem<sub>l</sub>m*, *sem<sub>r</sub>l<sub>n</sub>*, *sem<sub>r</sub>m*, *sem<sub>r</sub>n*). We say a predicate is its argument’s semantic head, and the latter is the former’s child. Features related to this type may track the current semantic parsing status.

**Path.** There are two basic types of path between the predicate and the argument candidates. One is the linear path (*linePath*) in the sequence, the other is the path in the syntactic parsing tree (*dpPath*). For the latter, we further divide it into four sub-types by considering the syntactic root, *dpPath* is the full path in the syntactic tree. Leading two paths to the root from the predicate and the argument, respectively, the common part of these two paths will be *dpPathShare*. Assume that *dpPathShare* starts from a node  $r'$ , then *dpPathPred* is from the predicate to  $r'$ , and *dpPathArgu* is from the argument to  $r'$ .

**Family.** Two types of children sets for the predicate or argument candidate are considered, the first includes all syntactic children (*children*), the second also includes all but excludes the left most and the right most children (*noFarChildren*).

**Concatenation of Elements.** For all collected elements according to *linePath*, *children* and so on, we use three strategies to concatenate all those strings to produce the feature value. The first is *seq*, which concatenates all collected strings without doing anything. The second is *bag*, which removes all duplicated strings and sort the rest. The third is *noDup*, which removes all duplicated neighbored strings.

In the following, we show some feature template examples derived from the above mentioned items.

*a.lm.lemma* The lemma of the left most child of the argument candidate.

*p.h.dprel* The dependant label of the syntactic head of the predicate candidate.

*a.pos+p.pos* The concatenation of *PoS* of the argument and the predicate candidates.

*p<sub>-1</sub>+posp.pos* *PoS* of the previous word of the predicate and *PoS* of the predicate itself.

*a:p|dpPath.lemma.bag* Collect all lemmas along with the syntactic tree path from the argument to the predicate, then removed all duplicated ones and sort the rest, finally concatenate all as a feature string.

*a:p.highSupportNoun|linePath.dprel* Collect all dependant labels along with the line path from the argument to the high support noun of the predicate, then concatenate all as a feature string.

*(a:p|dpPath.dprel)+p.FEAT1* Collect all dependant labels along with the line path from the argument to the predicate and concatenate them plus the first FEAT of the predicate.

An important feature for the task is *dpTreeRelation*, which returns the relationship of  $a$  and  $p$  in a syntactic parse tree and cannot be derived from combining the above basic elements. The possible values for this feature include *parent*, *sibling* etc.

## 5 Automatically Discovered Feature Template Sets

For each language, starting from a basic feature template set (a small subset of *FT*) according to our previous result in English dependency parsing, each feature template outside the basic set is added and each feature template inside the basic set is removed one by one to check the effectiveness of each feature template following the performance change in the development set. This procedure will be continuously repeated until no feature template is added or removed or the performance is not improved.

There are some obvious heuristic rules that help us avoid trivial feature template checking, for example, *FEAT* features are only suitable for Catalan, Czech and Spanish. Though *FEAT* features are also available for Japanese, we don’t adopt them for this language due to the high training cost. To simplify feature representation, we use *FEAT1*, *FEAT2*,

|    | Ca | Ch | Cz | En | Gr | Jp | Sp |
|----|----|----|----|----|----|----|----|
| Ca | 53 |    |    |    |    |    |    |
| Ch | 5  | 75 |    |    |    |    |    |
| Cz | 11 | 10 | 76 |    |    |    |    |
| En | 11 | 11 | 12 | 73 |    |    |    |
| Gr | 7  | 7  | 7  | 14 | 45 |    |    |
| Jp | 6  | 22 | 13 | 15 | 10 | 96 |    |
| Sp | 22 | 9  | 18 | 15 | 9  | 12 | 66 |

Table 2: Feature template set: argument classifier

|    | Ch | En | Gr |
|----|----|----|----|
| Ch | 46 |    |    |
| En | 5  | 9  |    |
| Gr | 17 | 2  | 40 |

Table 3: Feature template set: sense classifier

and so on to represent different *FEAT* for every languages. A lookup list can be found in Table 1. According to the list, *FEAT4* represents *person* for Catalan or Spanish, but *Cas* for Czech.

As we don't manually interfere the selection procedure for feature templates, ten quite different feature template sets are obtained at last. Statistical information of seven sets for argument classifiers is in Table 2, and those for sense classifiers are in Table 3. Numbers in the diagonals of these two tables mean the numbers of feature templates, and others mean how many feature templates are identical for every language pairs. The most matched feature template sets are for Catalan/Spanish and Chinese/Japanese. As for the former, it is not so surprised because these two corpora are from the same provider.

Besides the above statistics, these seven feature template sets actually share little in common. For example, the intersection set from six languages, as Chinese is excluded, only includes one feature template, *p.lemma* (the lemma of the predicate candidate). If all seven sets are involved, then such an intersection set will be empty. Does this mean human languages share little in semantic representation? :)

It is unlikely to completely demonstrate full feature template sets for all languages in this short report, we thus only demonstrate two sets, one for English sense classification in Table 4 and the other for Catalan argument classification in Table 5<sup>3</sup>.

<sup>3</sup>Full feature lists and their explanation for all languages will be available at the website, <http://bcmi.sjtu.edu.cn/zhaohai>.

|   |  |
|---|--|
| - | <i>p.lm.pos</i>                                |
| - | <i>p.rm.pos</i>                                |
| - | <i>p.lemma</i>                                 |
| - | <i>p.lemma</i> + <i>p.lemma</i> <sub>1</sub>   |
| - | <i>p.lemma</i> + <i>p.children.dprel.noDup</i> |
| - | <i>p.lemma</i> + <i>p.currentSense</i>         |
| - | <i>p.form</i>                                  |
| - | <i>p.form</i> <sub>-1</sub> + <i>p.form</i>    |
| - | <i>p.form</i> + <i>p.form</i> <sub>1</sub>     |

Table 4: Feature set for English sense classification

## 6 Word Sense Determination

The shared task of CoNLL-2009 still asks for the predicate sense. In our work for CoNLL-2008 (Zhao and Kit, 2008), this was done by searching for a right example in the given dictionary. Unfortunately, we later found this caused a poor performance in sense determination. This time, an individual classifier is used to determine the sense for Chinese, English or German, and this is done by the argument classifier by introducing a virtual root for every predicates for the rest four languages<sup>4</sup>. Features used for sense determination are also selected following the same procedure in Section 5. The difference is only predicate related features are used for selection.

## 7 Decoding

The decoding for four languages, Catalan, Czech, Japanese and Spanish is trivial, each word pairs will be checked one by one. The first word of the pair is the virtual root or the predicate, the second is the predicate or every argument candidates. Argument candidates are checked in the order of different syntactic relations to their predicate, which are enumerated by the pruning algorithms in Section 3, or from left to right for the same syntactic relation. After the sense of the predicate is determined, the label of each argument candidate will be directly classified, or, it is proved non-argument.

As for the rest languages, Chinese, English or German, after the sense classifier outputs its result, an optimal argument structure for each predicate is determined by the following maximal probability.

$$S_p = \operatorname{argmax} \prod_i P(a_i | a_{i-1}, a_{i-2}, \dots), \quad (1)$$

<sup>4</sup>For Japanese, no senses for predicates are defined. Thus it is actually a trivial classification task in this case.

|   |  |
|---|--|
| - | <i>p.currentSense + p.lemma</i>            |
| - | <i>p.currentSense + p.pos</i>              |
| - | <i>p.currentSense + a.pos</i>              |
| - | <i>p<sub>-1</sub>.FEAT1</i>                |
| - | <i>p.FEAT2</i>                             |
| - | <i>p<sub>1</sub>.FEAT3</i>                 |
| - | <i>p.semrm.semdprel</i>                    |
| - | <i>p.lm.dprel</i>                          |
| - | <i>p.form + p.children.dprel.bag</i>       |
| - | <i>p.lemma<sub>n</sub> (n = -1, 0)</i>     |
| - | <i>p.lemma + p.lemma<sub>1</sub></i>       |
| - | <i>p.pos<sub>-1</sub> + p.pos</i>          |
| - | <i>p.pos<sub>1</sub></i>                   |
| - | <i>p.pos + p.children.dprel.bag</i>        |
| - | <i>a.FEAT1 + a.FEAT3 + a.FEAT4</i>         |
| - | <i>+ a.FEAT5 + a.FEAT6</i>                 |
| - | <i>a<sub>-1</sub>.FEAT2 + a.FEAT2</i>      |
| - | <i>a.FEAT3 + a<sub>1</sub>.FEAT3</i>       |
| - | <i>a.FEAT3 + a.h.FEAT3</i>                 |
| - | <i>a.children.FEAT1.noDup</i>              |
| - | <i>a.children.FEAT3.bag</i>                |
| - | <i>a.h.lemma</i>                           |
| - | <i>a.lm.dprel + a.form</i>                 |
| - | <i>a.lm.form</i>                           |
| - | <i>a.lm<sub>-1</sub>.lemma</i>             |
| - | <i>a.lm<sub>n</sub>.pos (n=0,1)</i>        |
| - | <i>a.noFarChildren.pos.bag + a.rm.form</i> |
| - | <i>a.pthead.lemma</i>                      |
| - | <i>a.rm.dprel + a.form</i>                 |
| - | <i>a.rm<sub>-1</sub>.form</i>              |
| - | <i>a.rm.lemma</i>                          |
| - | <i>a.rm.dprel + a.form</i>                 |
| - | <i>a.lowSupportVerb.lemma</i>              |
| - | <i>a<sub>-1</sub>.form</i>                 |
| - | <i>a.form + a<sub>1</sub>.form</i>         |
| - | <i>a.form + a.children.pos</i>             |
| - | <i>a.lemma + a.h.form</i>                  |
| - | <i>a.lemma + a.pthead.form</i>             |
| - | <i>a<sub>1</sub>.lemma</i>                 |
| - | <i>a<sub>1</sub>.pos + a.pos.seq</i>       |
| - | <i>a.pos + a.children.dprel.bag</i>        |
| - | <i>a.lemma + p.lemma</i>                   |
| - | <i>(a:p dpPath.dprel) + p.FEAT1</i>        |
| - | <i>a:p linePath.distance</i>               |
| - | <i>a:p linePath.FEAT1.bag</i>              |
| - | <i>a:p linePath.form.seq</i>               |
| - | <i>a:p linePath.lemma.seq</i>              |
| - | <i>a:p linePath.dprel.seq</i>              |
| - | <i>a:p dpPath.lemma.seq</i>                |
| - | <i>a:p dpPath.lemma.bag</i>                |
| - | <i>a:p dpPathArgu.lemma.seq</i>            |
| - | <i>a:p dpPathArgu.lemma.bag</i>            |

Table 5: Feature set for Catalan argument classification

where  $S_p$  is the argument structure,  $P(a_i|a_{i-1}\dots)$  is the conditional probability to determine the label of the  $i$ -th argument candidate label. Note that  $P(a_i|a_{i-1}, \dots)$  in equation (1) may be simplified as  $P(a_i)$  if the input feature template set does not concern with the previous argument label output. A beam search algorithm is used to find the parsing decision sequence.

## 8 Evaluation Results

Our evaluation is carried out on two computational servers, (1) **LEGA**, a 64-bit ubuntu Linux installed server with double dual-core AMD Opteron processors of 2.8GHz and 24GB memory. This server was also used for our previous participation in CoNLL-2008 shared task. (2) **MEGA**, a 64-bit ubuntu Linux installed server with six quad-core Intel Xeon processors of 2.33GHz and 128GB memory.

Altogether nearly 60,000 machine learning routines were run to select the best fit feature template sets for all seven languages within two months. Both LEGA and MEGA were used for this task. However, training and test for the final submission of Chinese, Czech and English run in MEGA, and the rest in LEGA. As we used multiple thread training and multiple routines run at the same time, the exact time cost for either training or test is hard to estimate. Here we just report the actual time and memory cost in Table 7 for reference.

The official evaluation results of our system are in Table 6. Numbers in bold in the table stand for the best performances for the specific languages. The results in development sets are also given. The first row of the table reports the results using golden input features. Two facts as the following suggest that our system does output stable results. The first is that two results for development and test sets in the same language are quite close. The second is about out-of-domain (OOD) task. Though for each OOD task, we just used the same model trained from the respective language and did nothing to strengthen it, this does not hinder our system to obtain top results in Czech and English OOD tasks. In addition, the automatically discovered feature template sets in this task were used for the joint task of this shared task, and also output top results according to the average score of semantic labeled F1 (Zhao et al., 2009).

|                        | average      | Catalan      | Chinese | Czech        | English      | German | Japanese     | Spanish      |
|------------------------|--------------|--------------|---------|--------------|--------------|--------|--------------|--------------|
| Development with Gold  | 81.24        | 81.52        | 78.32   | 86.96        | 84.19        | 77.75  | 78.67        | 81.32        |
| Development            | 80.46        | 80.66        | 77.90   | 85.35        | 84.01        | 76.55  | 78.41        | 80.39        |
| Test (official scores) | <b>80.47</b> | <b>80.32</b> | 77.72   | 85.19        | 85.44        | 75.99  | <b>78.15</b> | <b>80.46</b> |
| Out-of-domain          | 74.34        |              |         | <b>85.44</b> | <b>73.31</b> | 64.26  |              |              |

Table 6: Semantic labeled F1

|          |                       | Catalan | Chinese | Czech | English | German | Japanese | Spanish |
|----------|-----------------------|---------|---------|-------|---------|--------|----------|---------|
| Sense    | Training memory (MB)  |         | 418     |       | 136     | 63     |          |         |
|          | Training time (Min.)  |         | 11.0    |       | 2.5     | 1.7    |          |         |
|          | Test time (Min.)      |         | 0.7     |       | 0.2     | 0.03   |          |         |
| Argument | Training memory (GB)  | 0.4     | 3.7     | 3.2   | 3.8     | 0.2    | 1.4      | 0.4     |
|          | Training time (Hours) | 3.0     | 13.8    | 24.9  | 12.4    | 0.2    | 6.1      | 4.4     |
|          | Test time (Min.)      | 3.0     | 144.0   | 27.1  | 88.0    | 1.0    | 4.2      | 7.0     |

Table 7: Computational cost

## 9 Conclusion

As presented in the above sections, we have tackled semantic parsing for the CoNLL-2009 shared task as a word-pair classification problem. Incorporated with a proper argument pruning strategy and a large scale feature engineering for each language, our system produced top results.

## References

- Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy.
- Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, and Zdeněk Žabokrtský. 2006. Prague Dependency Treebank 2.0.
- Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*, June 4-5, Boulder, Colorado, USA.
- Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 2008–2013, Las Palmas, Canary Islands.
- Martha Palmer and Nianwen Xue. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.
- Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. AnCorà: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakesh, Morocco.
- Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 88–94, Barcelona, Spain, July 25-26.
- Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 203–207, Manchester, UK, August 16-17.
- Hai Zhao, Wenliang Chen, Jun’ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*, June 4-5, Boulder, Colorado, USA.