A Joint Model for Parsing Syntactic and Semantic Dependencies

Xavier Lluís and **Lluís Màrquez**
TALP Research Centre – Software Department (LSI)
Technical University of Catalonia (UPC)
{xlluis,lluism}@lsi.upc.edu

Abstract

This paper describes a system that jointly parses syntactic and semantic dependencies, presented at the CoNLL-2008 shared task (Surdeanu et al., 2008). It combines online Peceptron learning (Collins, 2002) with a parsing model based on the Eisner algorithm (Eisner, 1996), extended so as to jointly assign syntactic and semantic labels. Overall results are 78.11 global F_1, 85.84 LAS, 70.35 semantic F_1. Official results for the shared task (63.29 global F_1; 71.95 LAS; 54.52 semantic F_1) were significantly lower due to bugs present at submission time.

1 Introduction

The main goal of this work was to construct a joint learning architecture for syntactic-semantic parsing and to test whether the syntactic and semantic layers can benefit each other from the global training and inference.

All the components of our system were built from scratch for this shared task. Due to strong time limitations, our design decisions were biased towards constructing a simple and feasible system. Our proposal is a first order linear model that relies on an online averaged Perceptron for learning (Collins, 2002) and an extended Eisner algorithm for the joint parsing inference.

Systems based on Eisner algorithm (Carreras et al., 2006; Carreras, 2007) showed a competitive performance in the syntactic parsing of the English language in some past CoNLL shared tasks. Also, we believe that extending the Eisner algorithm to jointly parse syntactic and semantic dependencies it is a natural step to follow.

Note that syntactic and semantic tasks are related but not identical. Semantic dependencies can take place between words loosely related by the syntactic structure. Another difficulty is that state of the art SRL systems (Surdeanu et al., 2007) strongly rely on features extracted from the syntactic tree. The joint model grows syntactic and semantic structures at the same time, so features extracted from the syntactic tree (e.g., a syntactic path between a modifier and a distant predicate) are not available or expensive to compute within the Eisner algorithm. We overcome this problem again with a very simple (though not elegant) solution, consisting of introducing a previous syntactic parsing step.

2 System architecture

This section briefly describes the main components of our system: 1) Preprocessing, 2) Syntactic parsing, 3) Predicate identification, 4) Joint syntactic-semantic parsing, and 5) Postprocessing.

In *preprocessing*, the training corpus is traversed and feature extraction performed. Main features are borrowed from pre-existing well-known systems (see next subsection). The initial *syntactic parsing* is based on an Eisner parser trained with Perceptron and it is merely intended to allow the extraction of syntactic-based features for all the following phases (which share exactly the same feature set extracted from these parse trees). *Predicate identification* recognizes predicates by applying SVM classifiers[1] and a set of simple heuristic rules. The *joint syntactic-semantic parsing* phase

[1]We used SVM-light (see *www.joachims.org* for details).

is the core module of this work. It simultaneously derives the syntactic and semantic dependencies by using a first order Eisner model, extended with semantic labels and trained with averaged Perceptron. Finally, *postprocessing* simply selects the most frequent sense for each predicate.

2.1 Preprocessing and feature extraction

All features in our system are calculated in the preprocessing phase. We use the features described in McDonald et al. (2005) and Carreras et al. (2006) as input for the syntactic parsing phase, except for the dynamic features from Carreras et al. (2006). The joint syntactic-semantic parser uses all the previous features and also specific features for semantic parsing from Xue and Palmer (2004) and Surdeanu et al. (2007). The features have been straightforwardly adapted to the dependency structure used in this shared task, by substituting any reference to a syntactic constituent by the head of that constituent. About 5M features were extracted from the training corpus. The number of features was reduced to ~222K using a frequency threshold filter. A detailed description of the feature set can be found at Lluís (Forthcoming 2008).

2.2 Syntactic parsing

Our system uses the Eisner algorithm combined with an online averaged Pereceptron. We define the basic model, which is also the starting point for the joint model. Let L be the set of syntactic labels, $x = x_1, \ldots, x_n$ a sentence with n words, and $\mathcal{Y}(x)$ the set of all possible projective dependency trees for x.

A dependency tree $y \in \mathcal{Y}(x)$ is a labeled tree with arcs of the form $\langle h, m, l \rangle$ that is rooted on an artificial node, 0, added for this purpose. The head, h, and modifier, m, for a dependency index words in the sentence and can take values in $0 \leq h \leq n$ and $1 \leq m \leq n$. $l \in L$ is the label of the dependency.

The dependency parser (dp) is interested in finding the best scored tree for a given sentence x:

$$\text{dp}(x, \mathbf{w}) = \operatorname*{arg\,max}_{y \in \mathcal{Y}(x)} \text{score\_tree}(y, x, \mathbf{w})$$

Using an arc-based first order factorization, the function score_tree$(y, x, \mathbf{w})$ is defined as the summation of scores of the dependencies in y:

$$\sum_{\langle h,m,l \rangle \in y} \text{score}(\langle h, m, l \rangle, x, \mathbf{w}),$$

where $\mathbf{w}$ is the weight vector of the parser, computed using an online perceptron. The weight vector $\mathbf{w}$ can be seen as a concatenation of $|L|$ weight vectors of d components, one for each of the labels: $\mathbf{w} = (\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(l)}, \ldots, \mathbf{w}^{(|L|)})$. A function ϕ is assumed to extract features from a dependency $\langle h, m, l \rangle$ and from the whole sentence x. This function represents the extracted features as a d-dimensional vector.

With all these elements, the score of a dependency $\langle h, m, l \rangle$ is computed as a linear function:

$$\text{score}(\langle h, m, l \rangle, x, \mathbf{w}) = \phi(\langle h, m, l \rangle, x) \cdot \mathbf{w}^{(l)}$$

2.3 Predicate identification

We identified as verb predicates all verbs excluding the auxiliaries and the verb *to be*. These simple rules based on the POS and lemma of the tokens are enough to correctly identify almost all verb predicates. With regard to noun predicates, we directly identified as predicates the lemmas which appeared always as predicates with a minimum frequency in the training corpus. The remaining noun predicates were identified by a degree-2 polynomial SVM. This classifier was trained with the same features used in subsequent phases, but excluding those requiring identified predicates.

2.4 Joint syntactic and semantic Parsing

The previously described basic parsing model will be extended to jointly assign semantic dependency labels. Let S be the set of semantic labels. Note that at this point, a sentence x has a set of q words already identified as predicates. We will refer to them as $p_1, \ldots, p_q$, where $p_i \in \{1, \ldots, n\}$. We consider that each dependency has a set of semantic tags $l_{sem\,p_1}, \ldots, l_{sem\,p_q}$ one for each sentence predicate p_i. Also, we consider an extra *no-argument* label in the set of semantic labels S. Thus, an extended dependency d_s is defined as:

$$d_s = \left\langle h, m, l_{syn}, l_{sem\,p_1}, \ldots, l_{sem\,p_q} \right\rangle,$$

where l_{syn} denotes the syntactic label for the dependency.

Again, the best parse tree is that maximizing the score of a first order factorization:

$$\text{dp}(x, \mathbf{w}, y') = \operatorname*{arg\,max}_{y \in \mathcal{Y}(x)} \text{score\_tree}(y, x, \mathbf{w}, y')$$

$$\text{score\_tree}(y, x, \mathbf{w}, y') =$$

$$= \sum_{\langle h,m,\mathbf{l} \rangle \in y} \text{score}(\langle h, m, \mathbf{l} \rangle, x, \mathbf{w}, y'),$$

where the dependency label is now extended to $\mathbf{l} = \langle l_{syn}, l_{sem\,p_1}, \ldots, l_{sem\,p_q} \rangle$ and y' denotes the precomputed syntax tree. The score of a syntactic-semantic dependency is:

$$\text{score}\left(\langle h, m, \mathbf{l}\rangle, x, \mathbf{w}, y'\right) =$$

$$\text{syntactic\_score}\left(h, m, l_{syn}, x, \mathbf{w}\right) +$$

$$\text{sem\_score}\left(h, m, l_{sem\,p_1}, \ldots, l_{sem\,p_q}, x, \mathbf{w}, y'\right)$$

The syntactic score is computed as described in the basic model. Finally, the semantic scoring function computes the semantic score as the sum of the semantic scores for each predicate semantic label:

$$\text{sem\_score}\left(h, m, l_{sem\,p_1}, \ldots, l_{sem\,p_q}, x, \mathbf{w}, y'\right) =$$

$$\sum_{l_{sem\,p_i}} \frac{\phi_{sem}\left(\langle h, m, l_{sem\,p_i}\rangle, x, y'\right) \cdot \mathbf{w}^{(l_{sem\,p_i})}}{q}$$

Note that each sentence x has a different number of predicates q. To avoid an excessive weight of the semantic component in the global score and a bias towards sentences with many predicates, the score is normalized by the number of predicates in the sentence.

Figure 1 shows an example of a sentence fragment with syntactic and semantic dependencies. The three predicates of the sentence are already identified: $\{p_1 = \text{completed}, p_2 = \text{announced}, p_3 = \text{acquisition}\}$. All dependencies are of the form $d = \langle h, m, l_{syn}, l_{sem\,p_1}, l_{sem\,p_2}, l_{sem\,p_3}\rangle$. Note that semantic labels express semantic relations between a modifier and a predicate that can be anywhere in the sentence. The semantic labeling is not restricted to predicates that are the head of the modifier. In this example, the correct output for the dependency *previously-announced* is $h = \text{announced}$, $m = \text{previously}$, $l_{syn} = \text{AMOD}$, $l_{sem\,p_1} = \text{null}$, $l_{sem\,p_2} = \text{AM-TMP}$, $l_{sem\,p_3} = \text{null}$.

The above described factorization allows the parser to simultaneously assign syntactic and semantic labels and also to maximize a joint syntactic-semantic score of the tree. Note that the semantic scoring function ϕ_{sem} extracts features from the modifier, the head and the predicate of the parsed dependencies. The proposed model allows to capture interactions between syntax and semantics not only because the syntactic and semantic scores are combined but also because the semantic scoring function relies on features extracted from the head-modifier-predicate relations. Thus, the semantic scoring function depends on the syntactic dependency being built, and, in reverse, the semantic score can modify the dependency chosen.

Regarding implementation issues, note that we compute $|L| + |S| \cdot q$ scores to assign $q + 1$ labels to a given dependency. The scores are computed independently for each label. Otherwise, interactions among these labels, would raise the number of possible combined labels to an exponential number, $|L| \cdot |S|^q$, making the exhaustive evaluation infeasible in practice. Also related to efficiency, we apply syntactic and semantic filters in order to reduce the number of score evaluations. In particular, the set of assignable labels is filtered by the POS of the head and modifier (discarding all labels not previously seen in the training corpus between words with the same POS). Another filter removes the core arguments not present in the frame file of each predicate. This strategy allowed us to significantly improve efficiency without any loss in accuracy.

2.5 Postprocess

A simple postprocess assigns the most frequent sense to each identified predicate. Frequencies were computed from the training corpus. Experiments performed combining the best and second output of the joint parser and enforcing domain constraints via ILP (Punyakanok et al., 2004) showed no significant improvements.

3 Experiments and Results

All the experiments reported here were done using the full training corpus, and results are presented on the development set. The number of features used by the syntactic parser is ~177K. The joint parser uses ~45K additional features for recognizing semantic dependencies.

Figure 2 shows the learning curves from epoch 1 to 17 for several subsystems and variants. More specifically, it includes LAS performance on syntactic parsing, both for the individual parser and for the syntactic annotation coming from the joint syntactic-semantic parser. For the latter, also the F_1 score on semantic dependencies and global F_1 results are presented. We can observe that the syntactic LAS scores for the syntactic and joint parsers are very similar, showing that there is no loss in syntactic performance when using the joint syntactic-semantic strategy. Overall re-

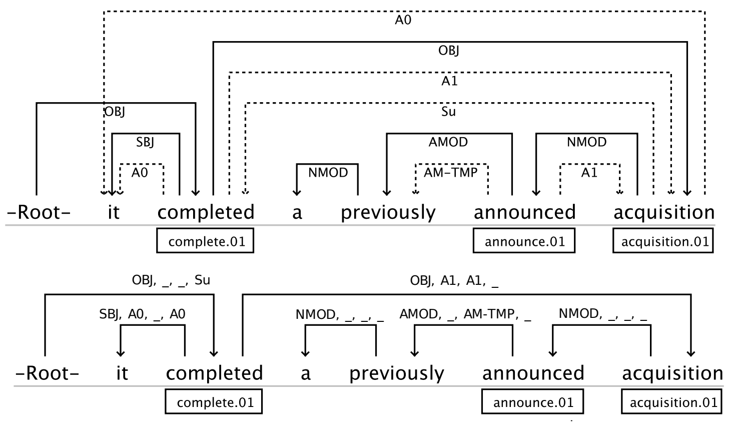

Figure 1: Syntactic and semantic dependencies.

sults are quite stable from epoch 4 (syntax slightly decreases but semantics slightly increases). The overall results on the test set (78.11 global F_1, 85.84 LAS, 70.35 semantic F_1) were computed by using 5 epochs of training, the optimal on the development set.

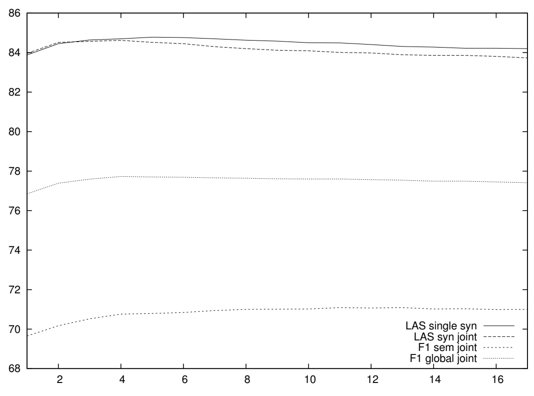

Figure 2: Learning curves for the syntactic-only and joint parsers.

The global F_1 result on the WSJ test corpus is 79.16, but these results drop 9.32 F_1 points on the out-of-domain Brown corpus. Also, a significant performance drop is observed when moving from verb argument classification (74.58 F_1, WSJ test) to noun argument classification (56.65 F_1, WSJ test). Note that the same features were used for training noun and verb argument classifiers. These results point out that there is room for improvement on noun argument classification. Finally, a comparison to a simple equivalent pipeline architecture, consisting of applying the syntactic base parser followed by an independent classification of semantic dependencies (using exactly the same features) revealed that the joint model outperformed the pipeline by 4.9 F_1 points in the annotation of semantic dependencies.

Regarding efficiency, the proposed architecture is really feasible. About 0.7GB of memory is required for the syntactic parser and 1.5GB for the joint parser. Most of these memory needs are due to the filters used. The filters allowed for a reduction of the computational cost by a factor of 5 with no loss in accuracy. These filters have almost no effect on the theoretical upper bound discarding the correct labels for only 0.2% of the syntactic dependencies and 0.44% of the semantic arguments in the development corpus. The semantic extension of the Eisner algorithm requires only a new table with backpointers for each predicate. Using a single processor of an *amd64 Athlon x2 5000+*, the syntactic parser can be trained at 0.2 s/sentence, and the joint parser at 0.3 s/sentence. Efficiency at test times is only slightly better.

4 Discussion

We have presented a novel joint approach to perform syntactic and semantic parsing by extending Eisner's algorithm. Our model allows to capture syntactic-semantic interactions as the computed syntactic-semantic score is globally optimized. The computational cost of the new setting is admissible in practice, leading to fairly efficient parsers, both in time and memory requirements.

Results obtained with the presented joint approach are promising, though not outstanding in the context of the CoNLL-2008 shared task. We believe that there is room for substantial improvement since many of the current system components are fairly simple. For instance, higher order extensions to the Eisner algorithm and well-known tricks for dealing with non-projective structures can be incorporated in our model. Also, we plan to incorporate other subtasks in the training of the joint model, such as predicate identification and argument recognition.

One of the potential drawbacks of our current approach is the need for a syntactic parsing preceding the joint model. This previous parse is simply included to permit the extraction of syntax based features. These features (including the syntactic path) could be dynamically computed when performing the joint parsing in the cases in which the predicate coincides with the head of the modifier being processed. These cases account for only 63.6% of the training corpus arguments. If a predicate is located in a sibling sentence span, the dynamic programming algorithm has not yet chosen which of the possible spans will be included in the final parse tree. Also, the predicate can be located at a lower level within the current span. These cases would require to recompute the score of the current span because syntactic path features are not available. The resulting cost would be prohibitive and approximate search needed. Our previous parsing phase is just an efficient and simple solution to the feature extraction problem in the joint model.

As previously seen, the joint model showed a similar syntactic performance and clearly better semantic performance than an equivalent pipeline system, showing that some degree of syntactic-semantic overlap is exploitable. Regarding the former, there is only a moderate degree (63.6%) of direct overlap between the syntactic head-modifier and semantic predicate-modifier relations. If the semantic score is highly dependent on a correct head the resulting increased score could benefit the choosing of a correct dependency. Otherwise, joint scores can introduce a significant amount of noise. All in all, further research is required in this direction.

Acknowledgements

This research has been partially funded by the Spanish Ministry of Education and Science, projects Trangram (TIN2004-07925-C03-02) and OpenMT (TIN2006-15307-C03-02).

References

Carreras, Xavier, Mihai Surdeanu, and Lluís Màrquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006)*.

Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 11th Conference on Computational Natural Language Learning (CoNLL-2007)*.

Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

Lluís, Xavier. Forthcoming 2008. Joint learning of syntactic and semantic dependencies. Master's thesis, Technical University of Catalonia (UPC).

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*.

Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling 2004*.

Surdeanu, Mihai, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*.

Surdeanu, Mihai, 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)*.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP-2004)*.