Semantic Role Labeling Using Lexical Statistical Information

Simone Paolo Ponzetto and **Michael Strube**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp/

Abstract

Our system for semantic role labeling is multi-stage in nature, being based on tree pruning techniques, statistical methods for lexicalised feature encoding, and a C4.5 decision tree classifier. We use both shallow and deep syntactic information from automatically generated chunks and parse trees, and develop a model for learning the semantic arguments of predicates as a multi-class decision problem. We evaluate the performance on a set of relatively 'cheap' features and report an F_1 score of 68.13% on the overall test set.

1 Introduction

This paper presents a system for the CoNLL 2005 Semantic Role Labeling shared task (Carreras & Màrquez, 2005), which is based on the current release of the English PropBank data (Palmer et al., 2005). For the 2005 edition of the shared task are available both syntactic and semantic information. Accordingly, we make use of both clausal, chunk and deep syntactic (tree structure) features, named entity information, as well as statistical representations for lexical item encoding.

The set of features and their encoding reflect the necessity of limiting the complexity and dimensionality of the input space. They also provide the classifier with enough information. We explore here the use of a minimal set of compact features for semantic role prediction, and show that a feature-based statistical encoding of lexicalised features such as predicates, head words, local contexts and PoS by means of probability distributions provides an efficient way of representing the data, with the feature vectors having a small dimensionality and allowing to abstract from single words.

2 System description

2.1 Preprocessing

During preprocessing the predicates' semantic arguments are mapped to the nodes in the parse trees, a set of hand-crafted shallow tree pruning rules are applied, probability distributions for feature representation are generated from training data[1], and feature vectors are extracted. Those are finally fed into the classifier for semantic role classification.

2.1.1 Tree node mapping of semantic arguments and named entities

Following Gildea & Jurafsky (2002), (i) labels matching more than one constituent due to non-branching nodes are taken as labels of higher constituents, (ii) in cases of labels with no corresponding parse constituent, these are assigned to the partial match given by the constituent spanning the shortest portion of the sentence beginning at the label's span left boundary and lying entirely within it. We drop the role or named entity label if such suitable constituent could not be found[2].

[1] All other processing steps assume a uniform treatment of both training and test data.

[2] The percentage of roles for which no valid tree node could be found amounts to 3% for the training and 7% for the development set. These results are compatible with the performance of the employed parser (Collins, 1999).

2.1.2 Tree pruning

The tagged trees are further processed by applying the following pruning rules:

- All punctuation nodes are removed. This is for removing punctuation information, as well as for aligning spans of the syntactic nodes with PropBank constituents[3].
- If a node is unary branching and its daughter is also unary branching, the daughter is removed. This allows to remove redundant nodes spanning the same tokens in the sentence.
- If a node has only preterminal children, these are removed. This allows to internally collapse base phrases such as base NPs.

Tree pruning was carried out in order to reduce the number of nodes from which features were to be extracted later. This limits the number of candidate constituents for role labeling, and removes redundant information produced by the pipeline of previous components (i.e. PoS tags of preterminal labels), as well as the sparseness and fragmentation of the input data. These simple rules reduce the number of constituents given by the parser output by 38.4% on the training set, and by 38.7% on the development set, at the cost of limiting the coverage of the system by removing approximately 2% of the target role labeled constituents. On the development set, the number of constituents remaining on top of pruning is 81,193 of which 7,558 are semantic arguments, with a performance upper-bound of 90.6% F_1.

2.1.3 Features

Given the pruned tree structures, we traverse the tree bottom-up left-to-right. For each non-terminal node whose span does not overlap the predicate we extract the following features:

Phrase type: the syntactic category of the constituent (NP, PP, ADVP, etc.). In order to reduce the number of phrase labels, we retained only those labels which account for at least 0.1% of the overall available semantic arguments in the training data. We replace the label for every phrase type category below this threshold with a generic `UNK` label. This reduces the number of labels from 72 to 18.

Position: the position of the constituent with respect to the target predicate (*before* or *after*).

Adjacency: whether the right (if before) or left (if after) boundary of the constituent is *adjacent*, *non-adjacent* or *inside* the predicate's chunk.

Clause: whether the constituent belongs to the clause of the predicate or not.

Proposition size: measures relative to the proposition size, such as (i) the number of constituents and (ii) predicates in the proposition.

Constituent size: measures relative to the constituent size, namely (i) the number of tokens and (ii) subconstituents (viz., non-leaf rooted subtrees) of the constituent.

Predicate: the predicate lemma, represented as the probability distribution $P(r|p)$ of the predicate p of taking one of the available r semantic roles. For unseen predicates we assume a uniform distribution.

Voice: whether the predicate is in *active* or *passive* form. Passive voice is identified if the predicate's PoS tag is `VBN` and either it follows a form of *to be* or *to get*, or it does not belong to a VP chunk, or is immediately preceded by an NP chunk.

Head word: the head word of the constituent, represented as the probability distribution $P(r|hw)$ of the head word hw of heading a phrase filling one of the available r semantic roles. For unseen words we back off on a phrasal model by using the probability distribution $P(r|pt)$ of the phrase type pt of filling a semantic slot r.

Head word PoS: the PoS of the head word of the constituent, similarly represented as the probability distribution $P(r|pos)$ of a PoS pos of belonging to a constituent filling one of the available r semantic roles.

Local lexical context: the words in the constituent other than the head word, represented as the

[3] We noted during prototyping that in many cases no tree node fully matching a role constituent could be found, as the latter did not include punctuation tokens, whereas in Collins' trees the punctuation terminals are included within the preceding phrases. This precludes *a priori* the output to align to the gold standard PropBank annotation and we use therefore pruning as a recovery strategy.

averaged probability distributions of each i-th non-head word w_i of occurring in one of the available r semantic roles, namely $\frac{1}{m}\sum_{i=1}^{m} P(r|w_i)$ for m non-head words in the constituent. For each unseen word we back off by using the probability distribution $P(r|pos_i)$ of the PoS pos_i of filling a semantic role r[4].

Named entities: the label of the named entity which spans the same words as the constituent, as well as the label of the largest named entity *embedded* within the constituent. Both values are set to `NULL` if such labels could not be found.

Path: the number of intervening NPB, NP, VP, VP-A, PP, PP-A, S, S-A and SBAR nodes along the path from the constituent to the predicate.

Distance: the distance from the target predicate, measured as (i) the number of nodes from the constituent to the lowest node in the tree dominating both the constituent and the predicate, (ii) the number of nodes from the predicate to the former common dominating node[5], (iii) the number of chunks between the base phrase of the constituent's head and the predicate chunk, (iv) the number of tokens between the head of the constituent and the predicate.

2.2 Classifier

We used the YaDT[6] implementation of the C4.5 decision tree algorithm (Quinlan, 1993). Parameter selection (99% pruning confidence, at least 10 instances per leaf node) was carried out by performing 10-fold cross-validation on the development set.

Data preprocessing and feature vector generation took approximately 2.5 hours (training set, including probability distribution generation), 5 minutes (development) and 7 minutes (test) on a 2GHz Opteron dual processor server with 2GB memory[7]. Training time was of approximately 17 minutes. The final system was trained using all of the available training data from sections 2–21 of the Penn TreeBank. This amounts to 2,250,887 input constituents of which 10% are non-`NULL` examples. Interestingly, during prototyping we first limited ourselves to training and drawing probability distributions for feature representation from sections 15–18 only. This yielded a very low performance (57.23% F_1, development set). A substantial performance increase was given by still training on sections 15–18, but using the probability distributions generated from sections 2–21 (64.43% F_1, development set). This suggests that the system is only marginally sensitive to the training dataset size, but pivotally relies on taking probability distributions from a large amount of data.

In order to make the task easier and overcome the uneven role class distribution, we limited the learner to classify only those 16 roles accounting for at least 0.5% of the total number of semantic arguments in the training data[8].

2.3 Post-processing

As our system does not build an overall sentence contextual representation, it systematically produced errors such as embedded role labeling. In particular, since no embedding is observed for the semantic arguments of predicates, in case of (multiple) embeddings the classifier output was automatically post-processed to retain only the largest embedding constituent. Evaluation on the development set has shown that this does not significantly improve performance, still it provides a much more 'sane' output. Besides, we make use of a simple technique for avoiding multiple `A0` or `A1` role assignments within the same proposition, based on constituent position and predicate voice. In case of multiple `A0` labels, if the predicate is in active form, the second `A0` occurrence is replaced with `A1`, else we replace the first occurrence. Similarly, in case of multiple `A1` labels, if the predicate is in active form, the first `A1` occurrence is replaced with `A0`, else we

[4] This feature was introduced as the information provided by lexical heads does not seem to suffice in many cases. This is shown by head word ambiguities, such as `LOC` and `TMP` arguments occurring in similar prepositional syntactic configurations — i.e. the preposition *in*, which can be head of both `AM-TMP` and `AM-LOC` constituents, as in *in October* and *in New York*. The idea is therefore to look at the words in the constituents other than the head, and build up an overall constituent representation, thus making use of statistical lexical information for role disambiguation.

[5] These distance measures along the tree path between the constituent and the predicate were kept separate, in order to indirectly include *embedding level* information into the model.

[6] `http://www.di.unipi.it/~ruggieri/software.html`

[7] We used only a single CPU at runtime, since the implementation is not parallelised.

[8] These include numbered arguments (`A0` to `A4`), adjuncts (`ADV`, `DIS`, `LOC`, `MNR`, `MOD`, `NEG`, `PNC`, `TMP`), and references (`R-A0` and `R-A1`).

	Precision	Recall	$F_{\beta=1}$
Development	71.82%	61.60%	66.32
Test WSJ	75.05%	64.81%	69.56
Test Brown	66.69%	52.14%	58.52
Test WSJ+Brown	74.02%	63.12%	68.13

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	75.05%	64.81%	69.56
A0	78.52%	72.52%	75.40
A1	75.53%	65.39%	70.10
A2	62.28%	52.07%	56.72
A3	63.81%	38.73%	48.20
A4	73.03%	63.73%	68.06
A5	0.00%	0.00%	0.00
AM-ADV	60.00%	42.69%	49.88
AM-CAU	0.00%	0.00%	0.00
AM-DIR	0.00%	0.00%	0.00
AM-DIS	75.97%	73.12%	74.52
AM-EXT	0.00%	0.00%	0.00
AM-LOC	54.09%	47.38%	50.51
AM-MNR	58.67%	46.22%	51.71
AM-MOD	97.43%	96.37%	96.90
AM-NEG	97.78%	95.65%	96.70
AM-PNC	42.17%	30.43%	35.35
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	75.41%	71.11%	73.20
R-A0	82.09%	73.66%	77.65
R-A1	72.03%	66.03%	68.90
R-A2	0.00%	0.00%	0.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	98.63%	98.63%	98.63

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

replace the second occurrence.

3 Results

Table 1 shows the results on the test set. Problems are inherently related with the skewed distribution of role classes, so that roles which have a limited number of occurrences are harder to classify correctly. This explains the performance gap on the A0 and A1 roles on one hand, and the A2, A3, A4, AM- arguments on the other.

One advantage of using a decision tree learning algorithm is that it outputs a model which includes a feature ranking, since the most informative features are those close to the root of the tree. In the present case, the most informative features were both distance/position metrics (distance and adjacency) and lexicalized features (head word and predicate).

4 Conclusion

Semantic role labeling is a difficult task, and accordingly, how to achieve an accurate and robust performance is still an open question. In our work we used a limited set of syntactic tree based distance and size metrics coupled with raw lexical statistics, and showed that such 'lazy learning' configuration can still achieve a reasonable performance.

We concentrated on reducing the complexity given by the number and dimensionality of the instances to be classified during learning. This is the core motivation behind performing tree pruning and statistical feature encoding. This also helped us to avoid the use of sparse features such as the explicit path in the parse tree between the candidate constituent and the predicate, and the predicate's subcategorization rule (cf. e.g. Pradhan et al. (2004)).

Future work will concentrate on benchmarking this approach within alternative architectures (i.e. two-phase with filtering) and different learning schemes (i.e. vector-based methods such as Support Vector Machines and Artificial Neural Networks).

Acknowledgements: This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a KTF grant (09.003.2004).

References

Carreras, Xavier & Lluís Màrquez (2005). Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Collins, Michael (1999). *Head-driven statistical models for natural language parsing*, (Ph.D. thesis). Philadelphia, Penn., USA: University of Pennsylvania.

Gildea, Daniel & Daniel Jurafsky (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Palmer, Martha, Dan Gildea & Paul Kingsbury (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pradhan, Sameer, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin & Daniel Jurafsky (2004). Support vector learning for semantic argument classification. *Journal of Machine Learning, Special issue on Speech and Natural Language Processing*. To appear.

Quinlan, J. Ross (1993). *C4.5: programs for machine learning*. San Francisco, Cal., USA: Morgan Kaufmann Publishers Inc.