Semantic Role Labelling With Chunk Sequences

Ulrike Baldewein, Katrin Erk, Sebastian Padó
Saarland University
Saarbrücken, Germany
{ulrike,erk,pado}@coli.uni-sb.de

Detlef Prescher
University of Amsterdam
Amsterdam, The Netherlands
prescher@science.uva.nl

Abstract

We describe a statistical approach to semantic role labelling that employs only shallow information. We use a Maximum Entropy learner, augmented by EM-based clustering to model the fit between a verb and its argument candidate. The instances to be classified are *sequences* of chunks that occur frequently as arguments in the training corpus. Our best model obtains an F score of 51.70 on the test set.

1 Introduction

This paper describes a statistical approach to semantic role labelling addressing the CoNLL shared task 2004, which is based on the the current release of the English PropBank data (Kingsbury et al., 2002). For further details of the task, see (Carreras and Màrquez, 2004).

We address the main challenge of the task, the absence of deep syntactic information, with three main ideas:

- Proper constituents being unavailable, we use *chunk sequences* as instances for classification.
- The classification is performed by a *maximum entropy model*, which can integrate features from heterogeneous data sources.
- We model the fit between verb and argument candidate by *clusters* induced with EM on the training data, which we use as features during classification.

Sections 2 through 4 describe the systems' architecture. First, we compute chunk sequences for all sentences (Sec. 2). Then, we classify these sequences with maximum entropy models (Sec. 3). Finally, we determine the most probable chain of sequences covering the whole sentence (Sec. 4). Section 5 discusses the impact of different parameters and gives final results.

2 Chunk Sequences as Instances

All studies of semantic role labelling we are aware of have used constituents as instances for classification. However, constituents are not available in the shallow syntactic information provided by this task. Two other levels of granularity are available in the data: words and chunks. In a pretest, we found that words are too fine grained, such that learners find it very difficult to identify argument boundaries on the word level. Chunks, too, are problematic, since one third of the arguments span more than one chunk, and for one tenth of the arguments the boundaries do not coincide with any chunk boundaries.

We decided to use *chunk sequences* as instances for classification. They can describe multi-chunk and part-chunk arguments, and by approximating constituents, they allow the use of linguistically informed features. In the sentence in Figure 1, *Britain's manufacturing industry* forms a sequence of type `NP_NP`. To make sequences more distinctive, we conflate whole clauses embedded deeper than the target to `S`: For the target *transforming*, we characterise the sequence for *to boost exports* as `S` rather than `VP_NP`. An argument boundary inside a chunk is indicated by the part of speech of the last included word: For *boost* the sequence is `VP(NN)`.

To determine "good" sequences, we collected argument realisations from the training corpus, generalising them by simple heuristics (e.g. removing anything enclosed in brackets). The generalised argument sequences exhibit a Zipfian distribution (see Fig. 2). `NP` is by far the most frequent sequence, followed by `S`. An example of a very infrequent argument chunk sequence is `NP_PP_NP_PP_NP_VP_PP_NP_NP` (in words: *a bonus in the form of charitable donations made from an employer 's treasury*).

The chunk sequence approach also allows us to consider the *divider chunk sequences* that separate arguments and targets. For example, `A0`s are usually divided from the target by the empty divider, while `A2` arguments are

```
Britain 's   manufacturing industry is   transforming itself to  boost exports
NNP     POS VBG          NN      VBZ VBG         PRP  TO  NN   NNS
[NP ] [NP                    ] [VP          ] [NP] [VP    ] [NP ]
                                                   [S          ]
```

Figure 1: Part of a sentence with part of speech, chunk and clause information

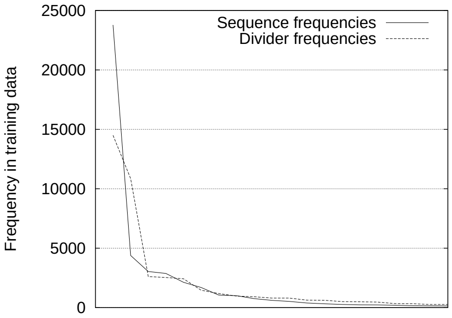

Figure 2: Frequency distribution for the 20 most frequent sequences and dividers in the training data

separated from it by e.g. a typical `A1` sequence. Generalised divider chunk sequences separating actual arguments and targets in the training set show a Zipfian distribution similar to the chunk sequences (see Fig. 2).

As instances to be classified, we consider all sequences whose generalised sequence and divider each appear at least 10 times for an argument in the training corpus, and whose generalised sequence and divider appear together at least 5 times. The first cutoff reduces the number of sequences from 1089 to 87, and the number of dividers from 999 to 120, giving us 581,813 sequences as training data (about twice as many as words), of which 45,707 are actual argument labels. The additional filter for sequence/divider pairs reduces the training data to 354,916 sequences, of which 43,622 are actual arguments. We pay for the filtering by retaining only 87.49% of arguments on the training set (83.32% on the development set).

3 Classification

3.1 Maximum Entropy Modelling

We use a log-linear model as classifier, which defines the probability of a class c given an feature vector $\vec{v}$ as

$$p(c|\vec{v}) = \frac{1}{Z} \prod_i e^{\alpha_i f_i(v,c)}$$

where Z is a normalisation constant, $f_i(v, c)$ the value of feature v_i for class c, and α_i the weight assigned to f_i. The model is trained by optimising the weights α_i subject to the *maximum entropy* constraint which ensures that the *least committal* optimal model is learnt. We used the `estimate` software for estimation, which implements the LMVM algorithm (Malouf, 2002) and was kindly provided by Rob Malouf.

We chose a maximum entropy approach because it can integrate many different sources of information without assuming independence of features. Also, models with minimal commitment are good predictors of future data. Maxent models have found wide application in NLP during the recent years; for semantic role labelling (on FrameNet data) see (Fleischman et al., 2003).

3.2 Classification Procedure

The most straightforward procedure would be to have the classifier assign all argument classes plus `NOLABEL` to sequences. However, this proved ineffective due to the prevalence of `NOLABEL`: Since this class makes up more than 80% of the training sequences, the classifier concentrates on assigning `NOLABEL` well.

Therefore, we divide the task of automatic semantic role assignment into two classification subtasks: *argument identification* and *argument labelling*. Argument identification is a binary decision for all sequences between `LABEL` (semantic argument) and `NOLABEL` (no semantic argument), which allows us to pool the frequencies of all argument labels. Argument labelling then assigns proper semantic roles only to those sequences that were recognised as `LABEL`s in the first step.

3.3 Features

We experimented with four types of features: *shallow* (mostly co-occurrence and distance statistics), *higher-level* (linguistically informed), *divider* and *em* (results of the EM-clustering).

Shallow Features. Our shallow features comprise statistics on the current sequence and its position as well as on the target: the sequence itself, the target lemma, the length of the current sequence in chunks, its absolute frequency, its position (before or after the target, as first or last sequence in the sentence), its distance to the target in question and its embedding depth in comparison with the target (with regard to clause embedding). We also count how often we have seen the current sequence as an argument for the current target lemma, and as which argument. Other features describe the context of the sequence: whether it is embedded in an admissible sequence or embeds one, and a two-chunk history. We also list the arguments for which the sequence is the best candidate, judging by its frequency.

Higher-Level Features. Our higher-level features comprise a heuristically determined *superchunk* label which is an abstraction of the chunk sequence (one of NP, VP, PP, S, ADVP, ADJP, and the rest class THING), the preposition of the sequence (if it either starts with or is directly preceded by a preposition), and the lemma and part of speech of the heuristically determined head of the sequence. We also check if the sequence in question is an NP (by its superchunk) directly before or after the target, if the sequence contains prepositions in unusual positions, if it consists of the word *n't* or *not*, and if the target lemma is passive.

Divider Features. These are shallow and higher-level features related to the divider sequences: the divider itself, its superchunk, and we state whether, judging by the divider, the sequence is an argument. A similar feature judges this by the combination of divider and sequence.

Features based on EM-Based Clustering. We use EM-based clustering to measure the fit between a target verb, an argument position of the verb, and the head lemma (or head named entity) of a sequence.

EM-based clustering, originally introduced for the induction of a semantically annotated lexicon (Rooth et al., 1999), regards classes as hidden variables in the context of maximum likelihood estimation from incomplete data via the *expectation maximisation* algorithm.

In our application, we aim at deriving a probability distribution $p(y)$ on verb-argument pairs y from the training data. Using the key idea that y is conditioned on an unobserved class $c \in C$, we define the probability of a pair $y = (y_1, y_2) \in \mathcal{Y}_1 \times \mathcal{Y}_2$ as:

$$\begin{aligned} p(y) &= \sum_{c \in C} p(c, y) = \sum_{c \in C} p(c) p(y|c) \\ &= \sum_{c \in C} p(c) p(y_1|c) p(y_2|c) \end{aligned}$$

The last line is warranted by the assumption that y_1 and y_2 are independent and are only conditioned on c. This assumption makes clustering feasible in the first place. We use the EM algorithm to maximise the incomplete data log-likelihood $L = \sum_y \tilde{p}(y) \ln p(y)$ as a function of the probability distribution p for a given empirical probability distribution $\tilde{p}$.

In two additional features, we substitute the head word by the sequence and divider characterisation respectively, using EM clustering to measure the fit between target verb, argument position, and sequence (or divider).

4 Finding the Best Chain of Sequences

Classification only determines probable argument labels for individual chunk sequences. We still have to determine the most probable *chain* of chunk sequences (such as A0 A1) that covers the whole sentence.

Recall that there are about 1.6 times as many sequences as words, many of which overlap; therefore, exhaustive searching is infeasible. Instead, we first run a beam search with a simple probability model to identify the n most probable chains of chunk sequences. Then, we re-rank them to take global considerations into account.

Beam Search. For each sentence, we build an agenda of partial argument chains. We calculate the probability of each chain as $P_c(S_1, S_2, \ldots) = \prod_i P_a(S_i)$, thereby assuming independence of sequences. For each sequence, we add the *three* most probable classes assigned by the argument labelling step. The result of the beam search are the n most probable (according to P_c) chains that cover the whole sentence. We found that increasing the beam width n to more than 20 increased performance only marginally.

Re-ranking. Due to the independence assumption in the beam search, chains that are assigned high probability may still be globally improbable. We therefore multiply each chain's probability P_c by its empirical probability P_e in the training data, using P_e as a prior. However, since these counts are still sparse, we exploit the fact that duplicate argument labels (i.e. discontinuous arguments) are relatively infrequent in the PropBank data by discounting chains with duplicate arguments by a factor d, which we empirically optimised as $d = 0.3$.

5 Experiments and Results

Optimising Step 1 (Argument Identification). On the development set, we explored the impact of different features from Section 3.3 on Step 1. Our optimal model contained as shallow features: all except the sequence's position; as divider features: divider sequence; as higher-level features: the preposition and the superchunk; as EM features: all. Adding more features deteriorated the model.

Feature Sets	Precision	Recall	F_1-Score
all	0.733	0.601	0.661
all - shallow	0.549	0.149	0.234
all - higher-level	0.683	0.636	0.658
all - divider	0.648	0.617	0.632
all - em	0.681	0.649	0.664
all ($\alpha = 0.41$)	**0.683**	**0.648**	**0.665**

Table 1: Different models for argument identification (evaluation scores category-specific for LABEL)

Table 1 presents an overview of different combinations of feature sets. We optimised category-specific F_1-score for LABEL, since only examples with LABEL are forwarded to Step 2. The first line (all) shows that the main problem in the first step is the recall, which limits the amount of arguments available for Step 2. For this reason, we varied the parameter α of the classification procedure: $\texttt{LABEL}(s)$ if $P(\texttt{LABEL}|s) > \alpha$. We found the optimal category-specific F-score for $\alpha = 0.41$, increasing the recall at the cost of precision.

Optimising Step 2 (Argument Labelling). We performed the same optimisation for Step 2, using the output of our best model of Step 1 as input. The best model for Step 2 uses all shallow features except the sequence's position; all higher-level features but negation; all divider features; no EM-clustering features. Table 2 shows the performance of the complete system for different feature sets. We also give two upper bounds for our system, one caused by the arguments lost in the sequence computation, and one caused by the arguments missed by Step 1.

Feature Sets	Precision	Recall	F_1-Score
Upper Bound 1	1.00	0.833	0.833
Upper Bound 2	1.00	0.648	0.786
all	**0.649**	**0.416**	**0.507**
all - shallow	0.104	0.064	0.079
all - higher-level	0.616	0.393	0.482
all - divider	0.642	0.415	0.504

Table 2: Different models for argument labelling (based on the best argument identification model)

The final model on the test set. Our best model combines the two models for Steps 1 and 2 indicated in boldface. Table 3 shows detailed results on the test set.

Discussion. During the development phase, we compared the performance of our final architecture with one that did not filter out on the basis of infrequent dividers as outlines in Sec. 2. Even though we lose 7.5% of the arguments in the development set by filtering, the F-score improves by about 12%. This shows that intelligent filtering is a crucial factor in a chunk sequence-based system.

The main problem for both subtasks is recall. This might also be the reason for the disappointing performance of the EM features, since the small amount of available training data limits the coverage of the models. As a consequence, EM features tend to increase the precision of a model at the cost of recall. At the overall low level of recall, the addition of EM features results in a virtually unchanged performance for Step 1 and even a suboptimal result for Step 2.

For both of our subtasks, adding more features to a given model can harm its performance. Evidently, some features predict the training data better than the development data, and can mislead the model. This can be seen as a kind of overfitting. Therefore, it is important to test not only feature sets, but also single features.

The two subtasks have rather different profiles. Table 1 shows that Step 1 hardly uses higher-level features, while the single divider feature has some impact. Step 2, on the other hand, improves considerably when higher-level features are added; divider features are less important (see Table 2). It appears that the split of semantic role labelling into argument identification and argument labelling mirrors a natural division of the problem, whose two parts rely on different types of information.

	Precision	Recall	$F_{\beta=1}$
Overall	65.73%	42.60%	51.70
A0	80.70%	56.92%	66.76
A1	59.60%	48.32%	53.37
A2	53.49%	28.99%	37.60
A3	45.10%	15.33%	22.89
A4	60.00%	18.00%	27.69
A5	0.00%	0.00%	0.00
AM-ADV	35.88%	15.31%	21.46
AM-CAU	14.29%	2.04%	3.57
AM-DIR	45.00%	18.00%	25.71
AM-DIS	58.33%	29.58%	39.25
AM-EXT	36.36%	28.57%	32.00
AM-LOC	40.23%	15.35%	22.22
AM-MNR	39.36%	14.51%	21.20
AM-MOD	97.98%	72.11%	83.08
AM-NEG	87.37%	65.35%	74.77
AM-PNC	43.48%	11.76%	18.52
AM-PRD	0.00%	0.00%	0.00
AM-TMP	55.94%	25.84%	35.35
R-A0	0.00%	0.00%	0.00
R-A1	0.00%	0.00%	0.00
R-A2	0.00%	0.00%	0.00
R-A3	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-PNC	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	0.00%	0.00%	0.00

Table 3: Details of the best model on the test set

References

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labelling. In *Proc. of CoNLL-2004*, Boston, MA.

M. Fleischman, N. Kwon, and E. Hovy. 2003. Maximum entropy models for FrameNet classification. In *Proc. of EMNLP'03*, Sapporo, Japan.

P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proc. of HLT*, San Diego, California.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of CoNLL-02*, Taipei, Taiwan.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proc. of ACL'99*.