Joint memory-based learning of syntactic and semantic dependencies in multiple languages

Roser Morante, Vincent Van Asch
CNTS - Language Technology Group
University of Antwerp
Prinsstraat 13
B-2000 Antwerpen, Belgium
{Roser.Morante,Vincent.VanAsch}@ua.ac.be

Antal van den Bosch
Tilburg University
Tilburg centre for Creative Computing
P.O. Box 90153
NL-5000 LE Tilburg, The Netherlands
Antal.vdnBosch@uvt.nl

Abstract

In this paper we present a system submitted to the CoNLL Shared Task 2009 performing the identification and labeling of syntactic and semantic dependencies in multiple languages. Dependencies are truly jointly learned, i.e. as if they were a single task. The system works in two phases: a classification phase in which three classifiers predict different types of information, and a ranking phase in which the output of the classifiers is combined.

1 Introduction

In this paper we present the machine learning system submitted to the CoNLL Shared Task 2009 (Hajič et al., 2009). The task is an extension to multiple languages (Burchardt et al., 2006; Hajič et al., 2006; Kawahara et al., 2002; Palmer and Xue, 2009; Surdeanu et al., 2008; Taulé et al., 2008) of the CoNLL Shared Task 2008, combining the identification and labeling of syntactic dependencies and semantic roles. Our system is a joint-learning system tested in the "closed" challenge, i.e. without making use of external resources.

Our system operates in two phases: a classification phase in which three memory-based classifiers predict different types of information, and a ranking phase in which the output of the classifiers is combined by ranking the predictions. Semantic and syntactic dependencies are jointly learned and processed. In the task description no precise definition is given of joint learning. We consider that a joint-learning system is one in which semantic and syntactic dependencies are learned and processed jointly as a single task. In our system this is achieved by fully merging semantic and syntactic dependencies at the word level as the first step.

One direct consequence of merging the two tasks, is that the class space becomes more complex; the number of classes increases. Many machine-learning approaches do not scale well to larger class spaces in terms of efficiency and computer resource requirements. Memory-based learning is a noted exception, as it is largely insensitive to the number of classes in terms of efficiency. This is the primary reason for using memory-based learning. Memory-based language processing (Daelemans and van den Bosch, 2005) is based on the idea that NLP problems can be solved by storing solved examples of the problem in their literal form in memory, and applying similarity-based reasoning on these examples in order to solve new ones. Memory-based algorithms have been previously applied to semantic role labeling and parsing separately (Morante et al., 2008; Canisius and Tjong Kim Sang, 2007).

We briefly discuss the issue of true joint learning of two tasks in Section 2. The system is described in Section 3, Section 4 presents and discusses the results, and in Section 5 we put forward some conclusions and future research.

2 Joint learning

When two tasks share the same feature space, there is the natural option to merge them and consider the merge as a single task. The merging of two tasks will typically lead to an increase in the number of classes, and generally a more complex class space. In practice, if two combined tasks are to some ex-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task, pages 25–30,
Boulder, Colorado, June 2009.

tent related, the increase will tend to be less than the product of the number of classes in the two original tasks, as classes from both tasks will tend to correlate. Yet, even a mild increase of the number of classes leads to a further fragmentation of the class space, and thus to less training examples per class label. Joint learning can therefore only lead to positive results if the data sparsity effect of the fragmentation of the class space is counter-balanced by an improved learnability.

Here, we treat the syntactic and semantic tasks as one and the same task. At the word level, we merge the class labels of the two tasks into single labels, and present the classifiers with these labels. Further on in our system, as we describe in the next section, we do make use of the compositionality of the labels, as the semantic and syntactic output spaces represented two different types of structure.

3 System description

The joint system that we submitted works in two phases: a classification phase in which three memory-based classifiers predict different aspects of joint syntactic and semantic labeling, and a ranking phase in which the output of the classifiers is combined. Additionally, a memory-based classifier is used for predicate sense disambiguation. As a first step, before generating the instances of the classifiers we merge the semantic and syntactic dependencies into single labels. The merged version of the dependencies from an example sentence is shown in Table 1, where column MERGED DEPs contains all the dependencies of a token separated by a blank space expressed in labels with the following format: PHEAD::PDEPREL:APRED.

3.1 Phase 1: Classification

In the classification phase, three classifiers predict different local aspects of the global output structure. The classifiers have been optimized for English, by training on the full training set and testing on the development set; these optimized settings were then used for the other six languages. We experimented with manually selected parameters and with parameters selected by a genetic algorithm, but the parameters found by the genetic algorithm did not yield better results than the manually selected parameters.

N	Token	Merged Dependencies
1	Housing	2::NMOD:A1
2	starts	2::_:A2 3::SBJ:_ 4::_:A1 6::_:A1 13::_:A0
3	are	0::ROOT:_
4	expected	3::VC:_
5	to	4::OPRD:C-A1
6	quicken	5::IM:_
7	a	8::NMOD:_
8	bit	6::OBJ:A2
9	from	6::ADV:A3
10	August	13::NMOD:AM-TMP
11	's	10::SUFFIX:_
12	annual	13::NMOD:AM-TMP
13	pace	9::PMOD:_
14	of	13::NMOD:A2
15	1,350,000	16::NMOD:_
16	units	14::PMOD:_
17	.	3::P:_

Table 1: Example sentence with merged dependency labels.

3.1.1 Classifier 1: Pairwise semantic and syntact dependencies

Classifier 1 predicts the merged semantic and syntactic dependencies that hold between two tokens. Instances represent combinations of pairs of tokens within a sentence. Each token is combined with all other tokens in the sentence. The class predicted is the PDEPREL:APRED label. The amount of classes per language is shown in Table 2 ("Classifier 1").

	Number of classes	
Lang.	Classifier 1	Classifier 2
Cat	111	111
Chi	309	1209
Cze	395	1221
Eng	351	1957
Ger	152	300
Jap	103	505
Spa	124	124

Table 2: Number of classes per language predicted by Classifiers 1 and 2.

We use an IB1 memory–based algorithm as implemented in TiMBL (version 6.1.2) [1], a memory-based classifier based on the k-nearest neighbor

[1] TiMBL: `http://ilk.uvt.nl/timbl`

rule. The IB1 algorithm was parameterised by using modified value difference as the similarity metric, gain ratio for feature weighting, using 11 k-nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance. Because of time limitations we used TRIBL for Czech and Chinese to produce the official results, although we also provide postevaluation results produced with IB1. TRIBL is a hybrid combination of IB1 and IGTREE, a fast decision-tree approximation of k-NN (Daelemans and van den Bosch, 2005), trading off fast decision-tree lookup on the most important features (in our experiments, five) with slower k-NN classification on the remaining features.

The features[2] used by this classifier are:

- The word, lemma, POS and FILLPRED[3] of the token, the combined token and of two tokens before and after token and combined token.
- POS and FILLPRED of the third token before and after token and combined token.
- Distance between token and combined token, location of token in relation to combined token.

Because data are skewed towards the NONE class, we downsampled the training instances so that there would be a negative instance for every positive instance. Instances with the NONE class to be kept were randomly selected.

3.1.2 Classifier 2: Per-token relations

Classifier 2 predicts the labels of the dependency relations of a token with its syntactic and/or semantic head(s). Instances represent a token. As an example, the instance that represents token 2 in Table 1 would have as class: _:A2-SBJ:_-_:A1-_:A1-_:A0. The amount of classes per language is shown in Table 2 under "Classifier 2". The number of classes exceeds 1,000 for Chinese, Czech, and English.

The features used by the classifier are the word, lemma, POS and FILLPRED of the token and two tokens before and after the token. We use the IB1 memory–based algorithm parameterised in the same way as Classifier 1.

[2] *POS* refers to predicted part-of-speech and *lemma* to predicted lemma in the description of features for all classifiers.

[3] The FILLPRED column has value Y if a token is a predicate.

3.1.3 Classifier 3: Pairwise detection of a relation

Classifier 3 is a binary classifier that predicts whether two tokens have a dependency relation. Instance representation follows the same scheme as with Classifier 1. We use the IGTREE algorithm as implemented in TiMBL. The data are also skewed towards the NONE class, so we downsampled the training instances so that there would be a negative instance for every four positive instances.

The features used by this classifier are:

- The word, lemma, POS and FILLPRED of the token, of the combined token, and of two tokens before and after the token.
- Word and lemma of two tokens before and after combined token.
- Distance between token and combined token.

3.1.4 Results

The results of the Classifiers are presented in Table 3. The performance of Classifiers 1 and 3 is similar across languages, whereas the scores for Classifier 2 are lower for Chinese, Czech and English. This can be explained by the fact that the number of classes that Classifier 2 predicts for these languages is significantly higher.

Lang.	C1	C2	C3
Cat	94.77	86.30	97.96
Chi	92.10	70.11	95.47
Cze	87.33	67.87	93.88
Eng	94.17	76.16	95.37
Ger	92.76	83.23	93.77
Jap	91.55	81.22	96.75
Spa	94.76	84.40	96.39

Table 3: Micro F1 scores per classifier (C) and per language.

Training times for the three classifiers were reasonably short, as is to be expected with memory-based classification. With English, C2 takes just over two minutes to train, and C3 half a minute. C1 takes 8 hours and 18 minutes, due to the much larger amount of examples and features.

3.2 Phase 2: Ranking

The classifier that is at the root of generating the desired output (dependency graphs and semantic

role assignments) is Classifier 1, which predicts the merged semantic and syntactic dependencies that hold between two tokens (PDEPREL:APRED labels). If this classifier would be able to predict the dependencies with 100% accuracy, no further processing would be necessary. Naturally, however, the classifier predicts incorrect dependencies to a certain degree, and does not provide a graph in wich all tokens have at least a syntactic head. It achieves 51.3% labeled macro F1. The ranking phase improves this performance. This is done in three steps: (i) ranking the predictions of Classifier 1; (ii) constructing an intermediate dependency tree, and (iii) adding extra semantic dependencies to the tree.

3.2.1 Ranking predictions of Classifier 1

In order to disambiguate between all possible dependencies predicted by this classifier, the system applies ranking rules. It analyses the dependency relations that have been predicted for a token with its potential parents in the sentence and ranks them. For example, for a sentence with 10 tokens, the system would make 10 predictions per token. The predictions are first ranked by entropy of the class distribution for that prediction, then using the output of Classifier 2, and next using the output of Classifier 3.

Ranking by entropy In order to compute entropy we use the (inverse-linear) distance-weighted class label distributions among the nearest neighbors that Classifier 1 was able to find. For example, the prediction for an instance can be: { NONE (2.74), NMOD:_ (0.48) }. We can compute the entropy for this instance using the formula in (1):

$$-\sum_{i=1}^{n} P(label_i) log_2(P(label_i)) \qquad (1)$$

with
- n: the total number of different labels in the distribution, and
- $P(label_i)$: $\frac{\text{the weight of label i}}{\text{the total sum of the weights in the distribution}}$

The system ranks the prediction with the lowest entropy in position 1, while the prediction with the highest entropy is ranked in the last position. The rationale behind this is that the lower the entropy, the more certain the classifier is about the predicted dependency. Table 4 lists the first six heads for the predicate word 'starts' ranked by entropy (cf. Table 1).

Head	Predicted label	Distribution	Entropy
Housing	NONE	{ NONE (8.51) }	0.0
expected	_:A1	{ _:A1 (5.64) }	0.0
to	NONE	{ NONE (4.74) }	0.0
quicken	_:A0	{ _:A0 (4.13), _:A1 (0.18), _:A2 (0.31) }	0.56
are	NONE	{ NONE (2.56), SBJ:_ (0.52) }	0.65
starts	_:A0	{ _:A0 (7.90), _:A1 (0.61), _:A2 (1.50) }	0.93

Table 4: Output of Classifier 1 for the first six heads of 'starts', ranked by entropy.

On the development data for English, applying this rule causes a marked error reduction of 26.5% on labeled macro F1: from 51.3% to 64.2%.

Ranking by Classifier 2 The next ranking step is performed by using the predictions of Classifier 2, i.e. the estimated labels of the dependency relations of a token with its syntactic and/or semantic head(s). The system ranks the predictions that are not in the set of possible dependencies predicted by Classifier 2 at the bottom of the ranked list.

Head	Predicted label	Distribution	Entropy
expected	_:A1	{ _:A1 (5.64) }	0.0
Housing	NONE	{ NONE (8.51) }	0.0
to	NONE	{ NONE (4.74) }	0.0
quicken	_:A0	{ _:A0 (4.13), _:A1 (0.18), _:A2 (0.31) }	0.56
are	NONE	{ NONE (2.56), SBJ:_ (0.52) }	0.65
starts	_:A0	{ _:A0 (7.90), _:A1 (0.61), _:A2 (1.50) }	0.93

Table 5: Output of Classifier 1 for the first six heads of 'starts'. Ranked by entropy and Classifier 2.

Because this is done after ranking by entropy, the instances with the lowest entropy are still at the top of the list. Table 5 displays the re-ranked six heads of 'starts', given that Classifier 2 has predicted that possible relations to heads are SBJ:A1 and _:A1, and given that only 'expected' is associated with one of these two relations.

On the development data for English, applying this rule induces a 9.0% error reduction on labeled macro F1: from 64.2% to 67.4%.

Ranking by Classifier 3 The final ranking step makes use of Classifier 3, which predicts the relation that holds between two tokens. The dependency relations predicted by Classifier 1 that are not confirmed by Classifier 3 predicting that a relation exists are moved to the end of the ranked list. Table 6 lists the resulting ranked list. On the development data for English, applying this rule yields another 5.2%

error reduction on labeled macro F1: from 67.4% to 69.1%.

Head	Predicted label	Distribution	Entropy
expected	_:A1	{ _:A1 (5.64) }	0.0
quicken	_:A0	{ _:A0 (4.13), _:A1 (0.18), _:A2 (0.31) }	0.56
starts	_:A0	{ _:A0 (7.90), _:A1 (0.61), _:A2 (1.50) }	0.93
Housing	NONE	{ NONE (8.51) }	0.0
to	NONE	{ NONE (4.74) }	0.0
are	NONE	{ NONE (2.56), SBJ:_ (0.52) }	0.65

Table 6: Output of Classifier 1 for the first six heads of 'starts'. Ranked by entropy, Classifier 2, and Classifier 3.

3.2.2 Construction of the intermediate dependency tree

After ranking the predictions of Classifier 1, the system selects a syntactic head for every token. This is motivated by the fact that every token has one and only one syntactic head. The system selects the prediction with the best ranking that has in the PDEPREL part a value different than "_".

The intermediate tree can have more than one root or no root at all. To make sure that every sentence has one and only one root we apply some extra rules. If the sentence does not have a token with a root label, the system checks the distributions of Classifier 1. The token with the rootlabel in its distribution that is the head of the biggest number of tokens is taken as root. If the intermediate tree has more than one root, the last root is taken as root. The other root tokens get the label with a syntax part (PDEPREL) that has the highest score in the distribution of Classifier 1.

The product of this step is a tree in which every token is uniquely linked to a syntactic head. Because syntactic and semantic dependencies have been linked, the tree contains also semantic dependencies. However, the tree is missing the purely semantic dependencies. The next step adds these relations to the dependency tree.

3.2.3 Adding extra semantic dependencies

In order to find the tokens that have only a semantic relation with a predicate, the system analyses for each predicate (i.e. tokens marked with Y in FILLPRED) the list of predictions made by Classifier 1 and selects the predictions in which the PDEPREL part of the label is "_" and the APRED part of the label is different than "_". On the development data for English, applying this rule produces a 6.7% error reduction on labeled macro F1: from 69.1% to 71.1%.

3.3 Predicate sense disambiguation

Predicate sense disambiguation is performed by a classifier per language that predicts the sense of the predicate, except for Japanese, as with that language the lemma is taken as the sense. We use the IGTREE algorithm. Instances represent predicates and the features used are the word, lemma and POS of the predicate, and the lemma and POS of two tokens before and after the predicate. The results per language are presented in Table 7.

Lang.	Cat	Chi	Cze	Eng	Ger	Spa
F1	82.40	94.85	87.84	93.64	73.57	81.13

Table 7: Micro F1 for the predicate sense disambiguation.

4 Overall results

The system was developed by training on the training set provided by the task organisers and testing on the development set. The final results were obtained by testing on the testing set. Table 8 shows the global results of the system for syntactic and semantic dependencies.

Lang.	F1	Precision	Recall
Cat	73.75	74.91	72.63
Chi	67.16	68.09	66.26
Chi*	67.79	68.70	66.89
Cze	60.50	62.55	58.58
Cze*	68.68	70.38	67.07
Eng	78.19	79.69	76.74
Ger	67.51	69.52	65.62
Jap	77.75	81.91	73.98
Spa	70.78	71.34	70.22
Av.	70.81	72.57	69.15

Table 8: Macro F1, precision and recall for all dependencies per language. Postevaluation results are marked with *.

Table 9 shows the scores of syntactic and semantic dependencies in isolation.

	Syntax	Semantics		
Lang.	LA	F1	Precision	Recall
Cat	77.33	70.14	72.49	67.94
Chi	67.58	66.71	68.59	64.93
Chi*	67.92	67.63	69.48	65.86
Cze	49.41	71.49	75.68	67.75
Cze*	60.03	77.28	80.73	74.11
Eng	80.35	75.97	79.04	73.13
Ger	73.88	61.01	65.15	57.36
Jap	86.17	68.82	77.66	61.80
Spa	73.07	68.48	69.62	67.38
Av.	72.54	68.95	72.60	65.76

Table 9: Labeled attachment (LA) score for syntactic dependencies and Macro F1, precision and recall of semantic dependencies per language. Postevaluation results are marked with *.

5 Conclusions

In this paper we presented the system that we submitted to the "closed" challenge of the CoNLL Shared Task 2009. We observe fairly low scores, which can be possibly improved for all languages by making use of the available morpho-syntactic features, which we did not use in the present system, by optimising the classifiers per language, and by improving the reranking algorithm. We also observe a relatively low recall on the semantic task as compared to overall recall, indicating that syntactic dependencies are identified with a better precision-recall balance. A logical continuation of this study is to compare joint learning to learning syntactic and semantic dependencies in isolation, using the same architecture. Only then will we be able to put forward conclusions about the performance of a joint learning system versus the performance of a system that learns syntax and semantics independently.

Acknowledgments

This study was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH), and from the Netherlands Organisation for Scientific Research.

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.

S. Canisius and E. Tjong Kim Sang. 2007. A constraint satisfaction approach to dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1124–1128.

W. Daelemans and A. van den Bosch. 2005. *Memory-based language processing.* Cambridge University Press, Cambridge, UK.

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.

R. Morante, W. Daelemans, and V. Van Asch. 2008. A combined memory-based semantic role labeler of english. In *Proc. of the CoNLL 2008*, pages 208–212, Manchester, UK.

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. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakesh, Morroco.