

Dependency Parsing and Semantic Role Labeling as a Single Task

Roser Morante, Vincent Van Asch
CLiPS - Computational Linguistics
University of Antwerp
Prinsstraat 13
B-2000 Antwerpen, Belgium
Roser.Morante, Vincent.VanAsch@ua.ac.be

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

Abstract

We present a comparison between two systems for establishing syntactic and semantic dependencies: one that performs dependency parsing and semantic role labeling as a single task, and another that performs the two tasks in isolation. The systems are based on local memory-based classifiers predicting syntactic and semantic dependency relations between pairs of words. In a second global phase, the systems perform a deterministic ranking procedure in which the output of the local classifiers is combined per sentence into a dependency graph and semantic role labeling assignments for all predicates. The comparison shows that in the learning phase a joint approach produces better-scoring classifiers, while after the ranking phase the isolated approach produces the most accurate syntactic dependencies, while the joint approach yields the most accurate semantic role assignments.

Keywords

Joint learning, dependency parsing, semantic role labeling

1 Introduction

In their currently popular definitions, dependency parsing and semantic role labeling are partly overlapping tasks. In their standard definitions they map to differently structured output spaces: dependency graphs span over sentences, while semantic role assignments center around individual predicates. Yet, the spaces overlap; in a dependency graph verbal predicates will tend to have dependency relations with the same modifiers that have a semantic role as argument of that predicate. In general, even though the labels are different, syntactic dependencies between two words often co-occur with the existence of certain semantic roles. Although they do not signify the same, the “subject” dependency relation, for example, often co-occurs with the “A0” label that denotes the agent role in the PropBank annotation scheme [14]. Overlaps such as these naturally suggest the possibility of jointly learning the two labeling tasks as if they were one.

In this paper we present a system that performs dependency parsing and semantic role labeling jointly,

which we submitted to the CoNLL Shared Task 2009 [8]. The task combines the identification and labeling of syntactic dependencies and semantic roles for seven languages. Details about the task setting and the data sets used can be found in the web page of the task¹. Additionally, we present a comparison of the joint system with another version of the system (“*isolated*” system) that processes semantic and syntactic dependencies separately. In this way, we are able to evaluate whether and where the joint learning approach is more efficient and successful than the isolated approach. As far as we know, this is the first time that such a comparison is performed.

In the joint system, the two labeling tasks are learned jointly by merging the syntactic and semantic dependencies, which implies that the number of labels increases, and the average number of examples per label decreases. This does not rule out the application of a machine learning classifier to the joint task, but the classifier should not be too sensitive to a fragmented class space with many labels. This is the main reason our system relies on local memory-based classifiers: they are largely insensitive in terms of training and processing efficiency to the number of class labels [4].

Memory-based algorithms have been previously applied to processing semantic and syntactic dependencies separately. As for semantic role labeling, [10] describes a memory-based semantic role labeling system for Spanish based on gold standard dependency syntax; [11] report on a semantic role labeling system for English based on syntactic dependencies produced by the MaltParser system of Nivre *et al.* [13]. As for dependency parsing, MaltParser uses memory-based learning as one of its optional local classifiers. Canisius *et al.* [2] present another type of memory-based dependency parser, extended later in [3] to a constraint satisfaction-based dependency parser. The latter parser combines local memory-based classification with a global optimization method based on soft weighted constraint-satisfaction inference, where the local classifiers estimate syntactic relations between pairs of words, the direction of the relation from children to parents, and the relations that parents have with children. Our current joint system adopts a similar strategy, but uses ranking rather than weighted constraint satisfaction inference.

¹ <http://ufal.mff.cuni.cz/conll2009-st/>

We briefly discuss the issue of joint learning of two tasks in Section 2. The two versions of the system are 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. For example, Sejnowski and Rosenberg [15] train a back-propagation multi-layered perceptron network on the joint task of mapping a letter in its context within an English word onto a joint label representing its phonemic mapping and a marker indicating stress on that phoneme. Van den Bosch [16] demonstrates that the joint learning of these two tasks indeed produces superior generalization performance as compared to learning the letter-phoneme task and the stress marker assignment task separately. Buchholz [1] transposes this idea to shallow parsing, and shows that POS tagging and base phrase chunking could be learned as a single task without any significant performance loss. Wang *et al.* [17] jointly learn Chinese word segmentation, named entity recognition, and part-of-speech tagging, outperforming a pipeline architecture baseline. Recently, Finkel and Manning show that joint learning of parsing and named entity recognition produce mildly improved performance for both tasks [6].

The merging of two tasks will typically lead to an increase in the number of class labels, and generally a more complex class space. In the worst case, the number of classes in the new class space is the product of the number of classes in the original tasks. In practice, if two combined tasks are to some extent related, the increase will tend to be limited, as class labels from the original tasks will tend to correlate. For instance, the POS tag for “determiner” will typically co-occur with the chunk marker for “beginning of noun phrase”, and less so, or not at all with other chunk markers. Yet, even a mild increase of the number of classes leads to a further separation 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 separation of the class space is counter-balanced by an unharmed, or even improved learnability. The latter is the primary reason for doing joint learning in the first place: certain parts of either of the combined tasks may be learned with more ease and with better success when it is co-learned with a part of another tasks. Learning this new separate part of the class space may in theory be easier than learning that particular part of the the larger unseparated class space of either of the composing tasks.

Here, in the joint system, we treat the syntactic and semantic tasks as one and the same task. For example, given a pair of words A and B, where B would be a verbal predicate, we train a local classifier to assign the label “SBJ:A0”, signifying that A is the modifier in a subject dependency relation with its head B, as well as that A is the argument with the A0 role of predicate B. Thus, we merge the class labels of the two

tasks into single labels, and present the classifiers with examples with these labels. Further on the system, as we describe in the next section, we do make use of the compositionality of the labels, as in the end we have to produce syntactic dependency graphs and semantic role assignments separately.

Apart from our system, three more joint systems participated in the CoNLL Shared Task 2009. The system described by [9] extends the Eisner parser to accommodate semantic dependencies. The system of [5] decomposes the joint learning task in four subtasks: semantic dependency identification and labeling, and syntactic dependency identification and labeling. A pipeline approach is set up in order to use the output of one task as input of another, and features not available at a certain step are incorporated iteratively. The system described by [7] is based on an incremental parsing model with synchronous syntactic and semantic derivations and a joint probability model for both types of dependency structures.

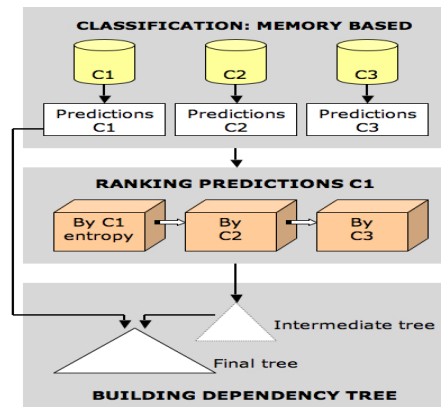


Fig. 1: Architecture of the joint system for dependency parsing and semantic role labeling

3 System description

In this section we describe the joint system, and compare it to the isolated version of the system. The joint system operates in three phases (see Figure 1): a classification phase in which three memory-based classifiers predict different aspects of joint syntactic and semantic labeling; a ranking phase in which the output of the classifiers is combined per sentence; and a phase in which the syntactic and semantic dependency graphs are built.

As a first step, before generating the instances of the joint classifiers, we merge the semantic and syntactic dependencies into single labels. Table 1 lists the merged versions of the dependencies in an example sentence. The “Merged” column contains for each token its one or more merged dependencies, separated by blank spaces; each dependency is expressed in labels with the following format: $\langle headID \rangle :: \langle dependencylabel \rangle : \langle semanticrolelabel \rangle$. If a syntactic or semantic label is absent, it is encoded by an underscore, “_”.

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

Table 1: *Example sentence with isolated and merged dependency labels*

3.1 Phase 1: Classification

In the classification phase, three classifiers predict different local aspects of the global output structure. All three operate at the word level. Two classifiers consider pairs of words, and predict the identity or the presence, respectively, of a joint semantic and syntactic dependency between them. The third classifier focus on single words only, and predicts the relations one word has with other words, without making reference to these other words. The hyperparameters of the classifiers were optimized on 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 as well. The hyperparameters and features used per classifier can be found in [12].

Classifier 1: Pairwise semantic and syntactic 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 a joint $\langle dependencyrelation \rangle$: $\langle semanticrole \rangle$ label, or NONE if no relation is present between the tokens. The amount of occurring classes for all seven languages is shown in Table 2.

| | Cat | Chi | Cze | Eng | Ger | Jap | Spa |
|----|-----|------|------|------|-----|-----|-----|
| C1 | 111 | 309 | 395 | 551 | 152 | 103 | 124 |
| C2 | 111 | 1209 | 1221 | 1957 | 300 | 505 | 124 |

Table 2: *Number of classes per language predicted by Classifiers 1 (C1) and 2 (C2)*

The three most frequent merged class labels in the case of English carry a syntactic dependency only: NMOD:_ (16.2% of all joint dependency labels), P:_ (10.1%), and PMOD:_ (8.7%). The syntactic dependency components of these three labels also co-occur with semantic roles in other joint labels, such as NMOD:A1, which is the sixth-most frequent joint label (3.9%). The fourth most frequent class is a semantic-role-only label: _:A0 (5.5%). The fifth most frequent class is the most frequent example of a joint syntactic and semantic dependency: OBJ:A1 (4.5% of all joint

dependency labels). The joint label SBJ:A0 ranks number 12 in the frequency list, covering 2.5% of all labels. At the other end of the frequency list, many joint labels occur only rarely; for English, 287 classes of the 551 occur only once.

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 single tokens. For 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 “C2”. The number of classes exceeds 1,000 for Chinese, Czech, and English. These numbers are higher than those for Classifier 1, as single tokens can have several semantic heads, along with always one syntactic head.

Classifier 3: Pairwise detection of a relation. Classifier 3 is a binary variant of Classifier 1 that predicts whether two tokens have a dependency relation. Instance representation follows the same scheme as with Classifier 1.

3.1.1 Results

The results of the Classifiers in terms of micro-averaged F-scores (with $\beta = 1$) over all class labels are presented in Table 3. The performance of Classifiers 1 and 3 is mostly above 90%, which is promising, leaving a clear margin of error nonetheless. The micro-averaged F-scores for Classifier 2 are lower, especially for Chinese, Czech, and English. There appears to be a correlation with the high number of Classifier 2 class labels (more than one thousand) for these three languages in particular, as witnessed by Table 2. The data sparsity induced by this high fragmentation of the class space may be hampering performance of Classifier 2 for these three languages.

| Lang. | C1 | C2 | C3 |
|-------|-------|-------|-------|
| Cat | 94.77 | 86.30 | 97.96 |
| Chi | 92.97 | 70.11 | 95.47 |
| Cze | 91.49 | 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-averaged F-scores of the joint system per classifier (C) and per language on the test corpora*

The isolated version of the system consists of six classifiers: each of the three classifiers described above, applied separately to syntax and semantics. These classifiers learn the content of the columns “Synt.” and “Sem.” in Table 1 in isolation, instead of learning it jointly.

| English Dependencies | C1 | | C2 | | C3 | |
|----------------------|-------|-------|-------|-------|-------|-------|
| | ISO | JOINT | ISO | JOINT | ISO | JOINT |
| Synt | 94.65 | 95.19 | 87.32 | - | 95.88 | - |
| Sem | 96.29 | 97.87 | 80.42 | - | 95.43 | - |
| Synt/Sem | 92.24 | 94.17 | - | 76.16 | 94.42 | 95.37 |

Table 4: *Comparison of Micro-averaged F-scores per classifier (C) on English test data in the joint and the isolated systems*

Table 4 compares the results per classifier for the joint (“JOINT”) and the isolated (“ISO”) systems. For Classifier 1 we compute the results for syntax, semantics and for the combination. To compute the combined results of the isolated system we recombine the results of the syntax and semantics classifiers. For Classifier 2 we can only compare the results of the combined syntactic and semantic labels in the joint system with the results that we obtain separately for syntax and semantics in the isolated system. For Classifier 3 it is not possible to compute the results of syntax and semantics separately for the joint system, as its binary labels do not distinguish between syntax and semantics.

Results of Classifiers 1 and 3 indicate that learning the tasks jointly produces a moderately better performance. The results of the main classifier, C1, show that syntactic and semantic dependencies are better learned within the joint setting. This might be explained by the fact that merged class labels are more fine-grained, and that certain merged labels can be predicted better separately than when lumped together in the original coarser-grained syntactic or semantic labels. By analysing the scores per class, we find that the scores per syntactic class improve for classes that split into several classes in the joint setting and are frequent. For example, the syntactic class NMOD, which splits into 13 classes and is very frequent, scores 5 points higher in the joint system, whereas the class MNR, which splits into 8 classes and is not frequent, scores 4 points lower. We observe the same trend in the scores per semantic class. For example, class A1, which is the most frequent and splits into 20 combined classes, scores 18 point higher in the joint setting.

3.2 Phase 2: Ranking

The classifier at the root of generating the desired output (dependency graphs and semantic role assignments) is Classifier 1, which predicts the semantic and syntactic dependencies that hold between two tokens. However, the classifier predicts incorrect dependencies to a certain degree, and does not produce a graph in which all tokens have at least a syntactic head. The evaluation of the overall joint syntactic and semantic labeled accuracy based on the output of Classifier 1 produces a baseline score of 51.3% labeled macro F-score. The ranking phase intends to improve over this performance. This is done in two steps: (i) re-ranking alternative predictions of Classifier 1 in order to construct an intermediate dependency tree, and (ii) adding extra semantic dependencies to the tree that do not align with syntactic dependencies.

3.2.1 Ranking predictions of Classifier 1

In order to disambiguate between all possible dependencies predicted by Classifier 1 between tokens, the system applies re-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 is able to find. For example, the prediction for an instance may be: { NONE (2.74), NMOD:_ (0.48) }. 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 confident the classifier is about the predicted dependency. Table 5 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 5: *Output of Classifier 1 for the first six heads of ‘starts’, ranked by entropy*

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 re-ranks the predictions that are not in the set of possible dependencies predicted by Classifier 2 to the bottom of the ranked list. Because this is done after ranking by entropy, the instances with the lowest entropy are still at the top of the list. Table 6 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.

| 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 6: *Output of Classifier 1 for the first six heads of ‘starts’, ranked by entropy and Classifier 2*

Ranking by Classifier 3. The final ranking step makes use of Classifier 3, which predicts the existence of a relation between two tokens. The dependency relations predicted by Classifier 1 that are not confirmed by Classifier 3 are moved to the end of the ranked list. Table 7 lists the resulting ranked list.

| 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 7: *Output of Classifier 1 for the first six heads of ‘starts’ ranked by entropy, Classifier 2, and Classifier 3*

After ranking the predictions of Classifier 1, the system selects as syntactic head for every token the pre-

diction with the best ranking that has a syntactic dependency value different from “_”. This is motivated by the fact that every token has one and only one syntactic head. The tree that results from this step (*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.

The error reduction rate at each step of the ranking process is shown in Table 8.

| Ranking | Labeled Macro F1 | Error Reduction |
|------------|------------------|-----------------|
| No ranking | 53.40 | - |
| C1 Entropy | 68.66 | 32.74 |
| By C2 | 71.48 | 8.99 |
| By C3 | 75.88 | 15.42 |

Table 8: *Effect of the ranking steps in the final results of the joint system on the test data of English*

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 semantic dependencies predicted by Classifier 1 that do not have a syntactic dependency part. The final step, described in Subsection 3.3 adds these relations to the dependency tree. We first describe how ranking in the isolated system is implemented.

3.2.2 Ranking in the isolated system

In the isolated system the same ranking process is applied to the syntactic dependency task in order to build a syntactic graph, where every node has only one syntactic head. The ranking algorithm takes as input the output of the classifiers that learn syntactic dependencies in isolation. Table 9 shows the error reduction rates of syntactic dependencies for English at every step of the ranking process, comparing the joint system and the isolated system. The results show that the effect of the ranking process outperforms the scores of the joint system slightly, despite the fact that the individual classifiers produced better scores in the joint setting.

| Ranking | Joint | | Isolated | |
|------------|-------|-------|----------|-------|
| | LAS | ER | LAS | ER |
| No ranking | 51.08 | - | 51.02 | - |
| C1 Entropy | 70.84 | 40.39 | 71.93 | 42.69 |
| By C2 | 74.22 | 11.29 | 74.71 | 9.90 |
| By C3 | 80.35 | 23.77 | 81.08 | 25.18 |

Table 9: *Comparison of the ranking effects in the isolated and joint systems for syntactic dependencies on the test data of English (LAS “Labeled Attachment Score”, ER “Error Reduction”)*

It is not possible to make the same comparison for semantic dependencies in isolation, as the ranking aims to select one syntactic head. In the semantic dependency graph, a token can have more than one head.

3.3 Phase 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 the list of predictions made by Classifier 1, selecting the predictions in which the syntactic part of the label is “_” and the semantic part of the label is not “_”. On the test data for English, applying this rule produces another 9.57% error reduction on labeled macro F1: from 75.88% to 78.19%.

In the isolated system semantic dependencies are processed differently. Classifiers 1, 2 and 3 learn the semantic dependencies in isolation. Then, the predictions of Classifier 1 are ranked by entropy. All AM arguments (e.g. AM-TMP) are kept because a predicate can have more than one, but the redundant basic arguments (such as A0, A1, etc.) are filtered out because each predicate can have only one of them. If there is more than one, we keep the one that occupies the highest position in the ranking. Additionally, some relations are filtered out by using Classifiers 2 and 3. The results obtained for semantics in the isolated and in the joint system are not directly comparable, because we cannot process them in the exact same way.

3.4 Predicate sense disambiguation

In the setting of the CoNLL Shared Task, processing the semantic dependencies of a predicate involves also disambiguating the sense of the predicate. This is performed in the joint and the isolated systems by a classifier for each language that predicts the sense of the predicate. An exception is made for Japanese, as with that language the lemma is taken as the sense. We use the IGTREE algorithm. Instances represent predicates; 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 10. We observe relatively high scores for Chinese and English.

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

Table 10: *Micro-averaged F-score for the predicate sense disambiguation*

4 Overall results

For each language, a full system is developed by training the three classifiers on the training set and testing on the development set. The final results are obtained by processing the test set provided by the CoNLL 2009 Shared Task. Table 11 lists the syntactic and semantic dependency prediction evaluated separately. The labeled attachment score (LA) indicates low scores for Chinese and Czech, and relative success for English and Japanese. In terms of semantic role labeling scores, precision is higher than recall for all languages, and markedly lower scores are obtained with German and Japanese.

The comparison of the final results of the joint and the isolated system presented in Table 12 indicates a moderately better performance of the isolated system

| Lang. | Syntax | Semantics | | |
|-------|--------|-----------|-----------|--------|
| | LAS | F1 | Precision | Recall |
| Cat | 77.33 | 70.14 | 72.49 | 67.94 |
| Chi | 67.92 | 67.63 | 69.48 | 65.86 |
| 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 |

Table 11: *Labeled attachment score (LAS) for syntactic dependencies and F scores of semantic dependencies per language in the joint system*

for syntactic dependencies, and a drop in performance of the isolated system for semantic dependencies. In particular, the isolated system produces considerably lower recall rates. This cannot be caused by the performance of the classifiers, since their results in the isolated setting are less than 2 points lower. Therefore, the ranking process customized to semantic dependencies is suboptimal.

| System | Syntax | Semantics | | |
|----------|--------|-----------|-----------|--------|
| | LAS | F1 | Precision | Recall |
| Joint | 80.35 | 75.97 | 79.04 | 73.13 |
| Isolated | 81.08 | 63.89 | 72.00 | 57.42 |

Table 12: *Comparison of labeled attachment score (LAS) of syntactic dependencies and F scores of semantic dependencies in the joint and the isolated systems for English*

5 Conclusions

In this paper we presented two systems, one that performs dependency parsing and semantic role labeling, based on local classifiers that learn the semantic and syntactic information jointly, and a second that performs the tasks based on local classifiers that learn the semantic and syntactic information in isolation. The isolated system was designed with the purpose of extracting conclusions about the effect of joint learning. By comparing the systems using English data, we found that in the joint learning setting the classifiers achieve slightly better scores.

The analysis of the results per class of the main classifier in the joint and the isolated setting shows that classes that are frequent and split into several merged classes in the joint setting have the highest increase of scores in the joint setting compared to the isolated setting. This suggests that separation of the class into finer-grained intersections of semantic and syntactic labels space facilitates the learning process, provided that there are enough examples for these finer-grained joint labels.

As for the joint system, the comparatively low scores on most languages (compared to other competing systems in the CoNLL 2009 shared task) can be likely improved (1) by making use of the available morpho-syntactic features, which we did not use in the present system; (2) by optimising the classifiers per language; and (3) by further improving the ranking algorithm.

We also observe a relatively low recall on the semantic task as compared to the overall scores, indicating that syntactic dependencies are identified with a better precision-recall balance than the semantic roles. More detailed tuning of our downsampling strategy may be used to improve the balance for the semantic task.

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

- [1] S. Buchholz. *Memory-Based Grammatical Relation Finding*. PhD thesis, Tilburg University, 2002.
- [2] S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. T. K. Sang. Dependency parsing by inference over high-recall dependency predictions. In *Proc. of CoNLL-X*, pages 3–8, New York City, NY, 2006.
- [3] S. Canisius and E. Tjong Kim Sang. A constraint satisfaction approach to dependency parsing. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1124–1128, 2007.
- [4] W. Daelemans and A. van den Bosch. *Memory-based language processing*. Cambridge University Press, Cambridge, UK, 2005.
- [5] Q. Dai, E. Chen, and L. Shi. An iterative approach for joint dependency parsing and semantic role labeling. In *Proc. of the CoNLL 2009: Shared Task*, pages 19–24, Boulder, CO, 2009.
- [6] J. R. Finkel and C. D. Manning. Joint parsing and named entity recognition. In *Proc. of Human Language Technologies: The 2009 Annual Conference of the NAACL*, pages 326–334, Boulder, Colorado, June 2009. ACL.
- [7] A. Gesmundo, J. Henderson, P. Merlo, and I. Titov. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proc. of the CoNLL 2009: Shared Task*, pages 37–42, Boulder, CO, 2009.
- [8] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Márquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL-2009: Shared Task*, Boulder, Colorado, USA, 2009.
- [9] X. LLuís, S. Bott, and L. Márquez. A second-order joint eisner model for syntactic and semantic dependency parsing. In *Proc. of the CoNLL 2009: Shared Task*, pages 79–86, Boulder, CO, 2009.
- [10] R. Morante. Semantic role labeling tools trained on the Cast3LB-CoNLL-SemRol corpus. In *Proc. of the LREC 2008*, Marrakech, Morocco, 2008.
- [11] R. Morante, W. Daelemans, and V. Van Asch. A combined memory-based semantic role labeler of english. In *Proc. of the CoNLL 2008*, pages 208–212, Manchester, UK, 2008.
- [12] R. Morante, V. Van Asch, and A. van den Bosch. Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proc. of the CoNLL 2009: Shared Task*, pages 25–30, Boulder, CO, 2009.
- [13] J. Nivre. *Inductive Dependency Parsing*. Springer, 2006.
- [14] M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, 2005.
- [15] T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.
- [16] A. Van den Bosch. *Learning to pronounce written words: A study in inductive language learning*. PhD thesis, Universiteit Maastricht, 1997.
- [17] X. Wang, J. Nie, D. Luo, and X. Wu. A Joint Segmenting and Labeling Approach for Chinese Lexical Analysis. In *Proc. of the European conference on Machine Learning and Knowledge Discovery*, pages 538–549, Berlin, 2008. Springer Verlag.