M A T E M A T I C K O - F Y Z I K Á L N Í  F A K U L T A

P R A H A

# ENSEMBLE PARSING AND ITS EFFECT ON MACHINE TRANSLATION

Nathan Green, Zdeněk Žabokrtský

ÚFAL Technical Report
**TR-2012-48**

U N I V E R S I T A S  C A R O L I N A  P R A G E N S I S

Copies of ÚFAL Technical Reports can be ordered from:

Institute of Formal and Applied Linguistics (ÚFAL MFF UK)
Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, CZ-11800 Prague 1
Czech Republic

or can be obtained via the Web: http://ufal.mff.cuni.cz/techrep

Charles University in Prague

Faculty of Mathematics and Physics

**Technical Report**

Nathan Green

Zdeněk Žabokrtský

# Ensemble Parsing and its Effect on Machine Translation

Institute of Formal and Applied Linguistics

Prague 2012

# Contents

# 1. Introduction

## 1.1 Problem Definition

**Problem:** The focus of much of dependency parsing is on creating new modeling techniques and examining new feature sets for existing dependency models. Often these new models are lucky to achieve equivalent results with the current state of the art results and often perform worse. These approaches are for languages that are often resource-rich and have ample training data available for dependency parsing. For this reason, the accuracy scores are often quite high. This, by its very nature, makes it quite difficult to create a significantly large increase in the current state-of-the-art. Research in this area is often concerned with small accuracy changes or very specific localized changes, such as increasing accuracy of a particular linguistic construction. With so many modeling techniques available to languages with large resources the problem exists on how to exploit the current techniques with the use of combination, or ensemble, techniques along with this plethora of data.

Dependency parsers are almost ubiquitously evaluated on their accuracy scores, these scores say nothing of the complexity and usefulness of the resulting structures. The structures may have more complexity due to the depth of their coordination or noun phrases. As dependency parses are basic structures in which other systems are built upon, it would seem more reasonable to judge these parsers down the NLP pipeline. The types of parsing errors that cause significant problems in other NLP applications is currently an unknown.

## 1.2 Research Approach

**Research Questions:** There are many questions to be examined when looking at state-of-the-art dependency parsing models and resource-rich languages. Models have similar accuracy but are they significantly different in their approach so that their construction of both correct and incorrect parse structures supply useful knowledge to an ensemble parse structure? Can the differences in parsers be exploited in both a combination, or ensemble, system and in a discrete classification system? Not only can we determine the usefulness of a particular model,

but can these differences be combined or selected in a fashion that still allows a system to construct a legitimate and logical dependency parse.

While errors are treated as equals in unlabeled accuracy scores, the effect on NLP systems maybe be greater. How can we rank the errors that effect Machine Translation and modify an ensemble system to better address those issues? One crucial question is what type of errors are the most egregious when propagated to other NLP systems. Is there any relation between UAS accuracy and performance found in the Treex system? Will an improvement with ensemble systems have an overall positive increase in a machine translation system.

**Research Approach:** To examine whether parser outputs can be combined in an effective manner, we look at creating one ensemble parse from any $N$ amount of parsers, whether they are constituent-based, transition-based, or graph-based parses. Looking at different combinations of these parser types will allow use to see how they differ both structurally and as well the differences in their error types. Given that the part of speech error distribution differs in each dependency technique, the ensemble weights can be learned from these distributions. To do this we will use fuzzy clustering of POS errors per dependency model to obtain our ensemble weights. Similarly these models perform differently depending on sentence length and length of dependencies. While ensemble systems improve the overall performance through a combination of approaches, we examine further whether a classifier can determine the appropriate model to use on a per token level. To do this we implement a meta-classifier using an SVM.

To examine the effects of dependency parsing down the NLP pipeline, we now turn to machine translation. Our dependency models will be evaluated using the Treex system and TectoMT translation system. This system, as opposed to other popular machine translation systems, makes direct use of the dependency structure during the conversion from source to target languages via a tectogrammatical tree translation approach. We will compare UAS accuracy to corresponding NIST and BLEU scores from the start to finish of the machine translation pipeline.

## 1.3 Contributions and Impact

**Contributions:**

- We show that combining dependency parsers of different techniques in an

ensemble framework, in particular constituency to dependency converted and traditional dependency techniques, leads to improved UAS for English dependency parsing.

- The same ensemble framework can be used for non-English languages for a similar improvement in most situations. These languages generally lack the constituency to dependency conversion so the ensemble system needs to be augmented with additional models of the same dependency techniques.
- Part of Speech error distribution can successfully be used via fuzzy clustering to learn the weights of an ensemble system, leading to greater UAS scores and reduced part of speech error.
- An SVM classifier can improve dependency parsing using simple sentence length, POS label, and model agreement features.
- We show the initial results of ensemble dependency parsers when used in the machine translation pipeline.

**Impact:**
The benefit of these techniques do not stop simply with dependency parsing. A dependency parse tree is often an input into other natural language processes. While under-resources languages may not be examining an entire NLP pipeline, resource-rich languages that take advantage of these approaches should additionally measure success in applications further down the NLP pipeline.

Often new annotation or new parsing techniques are tried and abandoned if they don't give an immediate boost to UAS. It is our goal and the hopeful impact of this report that we give further evidence on why parsing models should be evaluated on NLP systems other than just parsing output. Increasing UAS scores is more equated to a learning problem while increasing the results of an NLP system are the result of adding additional detail and information to your early level NLP structures. The latter we find to be a more convincing argument for research.

# 2. Ensemble Parsing

## 2.1 Introduction

Ensemble learning [3] has been used for a variety of machine learning tasks and recently has been applied to dependency parsing in various ways and with different levels of success. [25, 6] showed a successful combination of parse trees through a linear combination of trees with various weighting formulations. To keep their tree constraint, they applied Eisner's algorithm for reparsing [4].

Parser combination with dependency trees has been examined in terms of accuracy [23, 24, 27, 8]. Other methods of parser combinations have shown to be successful such as using one parser to generate features for another parser. This was shown in [17], in which Malt Parser was used as a feature to MST Parser. The result was a successful combination of a transition-based and graph-based parser, but did not address adding other types of parsers into the framework.

We believe our work is the first to examine ensemble parsing across a variety of languages and with both dependency and constituent parsers. In addition to examining the accuracy and improvements across these ensemble parsers we also examine their accuracy by POS and how these parse trees effect machine translation further in the NLP pipeline (see Chapter 3).

In this chapter we will discuss three ensemble approaches. First an fixed weight ensemble approach in which edges are added together in a weighted graph. Second, We will describe a way to learn these weights through fuzzy clustering based on POS errors. Third, we will describe a meta-classifier that uses an SVM to predict the correct model for edge using only model agreements without any linguistic information added. We will demonstrate these for a variety of languages and data sizes.

## 2.2 Fixed Weight Ensemble Parsing

Ensemble methods are sometimes used when you want models trained on different data to avoid a single domain output. We will only be looking at ways to combine models trained on the same or similar data in this report. We have created an

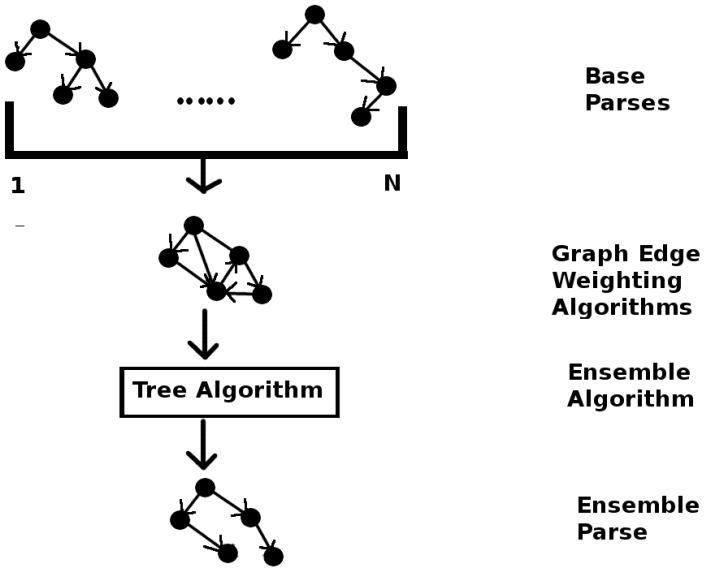


Figure 2.1: General flow to create an ensemble parse tree

ensemble class in Treex that collects all analytical trees (A-trees) present and combines their structure into a weighted graph.

To generate a single ensemble parse tree, our system takes $N$ parse trees as input. The inputs are from a variety of parsers as described in 2.2.1. All edges in these parse trees are combined into a graph structure. This graph structure accepts weighted edges. So if more than one parse tree contains the same tree edge, the graph will be weighted appropriately according to a chosen weighting algorithm. The weighting algorithms used in our experiments are described in 2.2.1.

Once the system has a weighted graph, the system then uses an algorithm to find a corresponding tree structure so there are no cycles. In this set of experiments we constructed a tree by finding the maximum spanning tree using Chu–Liu/Edmonds' algorithm, which is a standard choice for MST tasks. Figure 2.1 graphically shows the decisions one needs to make in this framework to create an ensemble parse.

### 2.2.1 Methodology

#### Evaluation

Made a standard in the CoNLL shared tasks competition, two standard metrics for comparing dependency parsing systems are typically used. *L*abeled *a*ttachment *s*core (LAS) and *u*nlabeled *a*ttachment *s*core (UAS). UAS studies the structure of a dependency tree and assesses how often the output has the correct head and

dependency arcs. In addition to the structure score in UAS, LAS also measures the accuracy of the dependency labels on each arc [1]. Since we are mainly concerned with the structure of the ensemble parse, we report only UAS scores in this paper on section 23 of the WST.

**Parsers**

For English we use 5 of the most commonly used parsers which enables us to have a wide scope for ensemble learning. They range from graph-based approaches to transition-based approaches to constituent parsers. Constituency output is converted to dependency structures using a converter [9]. All parsers are integrated in the Treex framework [26, 20] using the publicly released parsers from the respective authors but with Perl wrappers to allow them to work on a common tree structure.

- **Graph-Based:** A dependency tree is a special case of a weighted edge graph that spawns from an artificial root and is acyclic. Because of this we can look at a large history of work in graph theory to address finding the best spanning tree for each dependency graph. In this paper we use MST Parser version 0.5 using a second order model [15] as an input to our Ensemble Parser.

- **Transition-Based:** Transition-based parsing creates a dependency structure that is parameterized over the transitions used to create a dependency tree. This is closely related to shift-reduce constituency parsing algorithms. The benefit of transition-based parsing use greedy algorithms which have a linear time complexity. However, due to the greedy algorithms, longer arc parses can cause error propagation across each transition [11]. We make use of Malt Parser [16], which in the shared tasks was often tied with the best performing systems. With Maltparser version 1.5 we are using th nivreeager training algorithm. Additionally we will use Zpar 0.5 [28] which is based on Malt Parser but with a different set of non-local features.

- **Constituent Transformation:** While not a true dependency parser, one technique often applied is to take a state-of-the-art constituent parser and transform its phrase based output into dependency relations. This has been shown to also be state-of-the-art in accuracy for dependency parsing in English. In this paper we transformed the constituency structure to

dependencies using the Penn Converter conversion tool [9]. A version of this converter was used in the CoNLL shared task to create dependency treebanks as well. With English we have the additional ability to include constituent parsers. For the following ensemble in English experiments we make use of both Charniak's [2][1] constituent parser and Stanford's [10] version 1.6.9 constituent parser.

For this study we use the English CoNLL data. This data comes from the Wall Street Journal (WSJ) section of the Penn treebank [14]. All parsers are trained on sections 02-21 of the WSJ except for the Stanford parser which uses sections 01-21. Charniak, Stanford and Zpar use pre-trained models *ec50spfinal*, *wsjPCFG.ser.gz*, *english.tar.gz* respectively. For testing we use section 23 of the WSJ for comparability reasons with other papers. This test data contains 56,684 tokens. For tuning we use section 22. This data is used for determining the weighting features for the POS accuracies.

In addition to the UAS score of the enumerated parsers, we also report the accuracy of an Oracle Parser. This parse is simply the best possible parse of all the edges of the combined dependency trees in terms of UAS. If the reference, gold standard, tree has an edge that any of the parsers contain, we include that edge in the Oracle parse. Initially all nodes of the tree are connected to an artificial root. Since only edges that exist in a reference tree are added, the Oracle Parser maintains the acyclic constraint.

### Weighting Schemes

Currently we are applying three weighting algorithms to the graph structure. All three of these are simple weighting techniques but even in their simplicity we can see the benefit of this type of combination.

- **Uniform Weights**: an edge in the graph gets incremented +1 weight for each matching edge in each parser. If an edge occurs in 4 parsers, the weight is 4.
- **UAS Weighted**: Each edge in the graph gets incremented by the value of it's parsers individual accuracy. So in the UAS results in Table 2.2 an edge in Charniak's tree gets .92 added while MST gets .86 added to every edge

[1] ftp://ftp.cs.brown.edu/pub/nlparser/reranking-parserAug06.tar.gz

they share with the resulting graph. This weighting should allow us to add poor parsers with very little harm to the overall score.

- **Plural Voting Weights**: In Plural Voting the parsers are rated and each gets a "vote" based on their quality. With $N$ parsers the best parser gets $N$ votes while the last place parser gets 1 vote. In this paper, Charniak received 5 votes, Stanford received 4 votes, MST Parser received 3 votes, Malt Parser received 2 votes, and Zpar received 1 vote. Votes in this case are added to each edge as a weight.

- **UAS$^{10}$**: For this weighting scheme we took each UAS value to the 10th power. This gave us the desired affect of making the differences in accuracy more apparent and giving more distance from the best to worse parser. As shown later, this value was found empirically on tuning data.

### 2.2.2 Results

Table 2.2 contains the results of different parser combinations of the 5 parsers in Table 2.1. The results seem to indicate that using two parsers will give you an "average" score. Ensemble learning seems to start to have a benefit around 3 parsers with a few combinations having a better UAS score than any of the baseline parsers, these cases are in bold throughout the table. When we add a 4th parser to the mix almost all configurations lead to an improved score when the edges are not weighted uniformly. The only case in which this does not occur is when Stanford's Parser is not used. When all 5 parsers are used together with Plural Voting, the Ensemble Parser improves over the highest individual parser's UAS score. For UAS$^{10}$ voting, the 5 parser combination gives the second highest accuracy score. The top overall score is when we use UAS$^{10}$ weighting with the 4 top individual parsers. For parser combinations that do not feature Charniak's parser, we also find an increase in overall accuracy score compared to each individual parser, although never beating Charniak's individual score.

To see the maximum accuracy an ensemble system could achieve we include an Oracle Ensemble Parser in Table 2.2. The Oracle Parser gives us a ceiling on what ensemble learning can achieve. As we can see in Table 2.2, the ceiling of ensemble learning is 97.41% accuracy. Because of this high value with so few parsers, ensemble learning should be a very prosperous area for dependency parsing research.

| Parser | UAS % |
|---|---|
| Charniak | 92.08% |
| Stanford | 87.88% |
| MST | 86.49% |
| Malt | 84.51% |
| Zpar | 76.06% |

Table 2.1: Our baseline parsers and corresponding UAS used in our ensemble experiments

To discover the best exponential we looked at our combining all parsers at different exponential values. We empirically test difference values on our tuning data. $X^{10}$ is the top scoring weight for English. The results are in Table 2.3. We only discover this weight using the "all" parser setting and only on English. If this setup was used in production it would be wise to relearn this exponential value through new tuning data for the model combination choice and particular language each time.

**Pos Errors**

In [11] the authors confirm that two parsers, MST Parser and Malt parser, give similar accuracy results but with very different errors. MST parser, a maximum spanning tree graph-based algorithm, has evenly distributed errors with respect to sentence length while Malt Parser, a transition based parser, has errors on mainly longer sentences. This result comes from the approaches themselves. MST parser is globally trained so the best mean solution should be found, this is why errors on the longer sentences are about the same as the shorter sentences. Malt Parser on the other hand uses a greedy algorithm with a classifier that chooses a particular transition at each vertex. This leads to the possibility of the propagation of errors further in a sentence. Along this line of research we look at the error distribution for all 5 parsers along without best ensemble parser configuration. Much like the previous work we expect different types of errors given that our parsers are from 3 different parsing techniques. To examine if the Ensemble Parser is substantially changing the parse tree or is just taking the best parse tree and substituting a few edges, we examine the part of speech errors in Table 2.4.

As we can see the range of POS errors varies dramatically depending on which parser we examine. For instance for *CC*, Charniak has 83% while MST is only

| System | Uniform Weighting | UAS Weighted | Plural Voting | $UAS^{10}$ Weighted | Oracle UAS |
|---|---|---|---|---|---|
| Charniak-Stanford | 89.84 | 92.08 | 92.08 | 92.08 | 94.85 |
| Charniak-Mst | 89.14 | 92.08 | 92.08 | 92.08 | 95.33 |
| Charniak-Malt | 88.15 | 92.08 | 92.08 | 92.08 | 95.40 |
| Charniak-Zpar | 84.10 | 92.08 | 92.08 | 92.08 | 94.49 |
| Stanford-Mst | 86.92 | 86.49 | 87.88 | 86.49 | 94.29 |
| Stanford-Malt | 86.05 | 87.88 | 87.88 | 87.88 | 94.09 |
| Stanford-Zpar | 81.86 | 87.88 | 87.88 | 87.88 | 93.02 |
| Mst-Malt | 85.54 | 86.49 | 86.49 | 86.49 | 90.38 |
| Mst-Zpar | 81.19 | 86.49 | 86.49 | 86.49 | 92.03 |
| Malt-Zpar | 80.07 | 84.51 | 84.51 | 84.51 | 91.46 |
| Charniak-Stanford-Mst | 91.86 | **92.27** | **92.28** | **92.25** | 96.48 |
| Charniak-Stanford-Malt | 91.77 | **92.28** | **92.30** | 92.08 | 96.49 |
| Charniak-Stanford-Zpar | 91.22 | 91.99 | 92.02 | 92.08 | 95.94 |
| Charniak-Mst-Malt | 88.80 | 89.55 | 90.77 | 92.08 | 96.30 |
| Charniak-Mst-Zpar | 90.44 | 91.59 | 92.08 | 92.08 | 96.16 |
| Charniak-Malt-Zpar | 88.61 | 91.30 | 92.08 | 92.08 | 96.21 |
| Stanford-Mst-Malt | 87.84 | **88.28** | **88.26** | **88.28** | 95.62 |
| Stanford-Mst-Zpar | **89.12** | **89.88** | **88.84** | **89.91** | 95.57 |
| Stanford-Malt-Zpar | **88.61** | **89.57** | 87.88 | 87.88 | 95.47 |
| Mst-Malt-Zpar | **86.99** | **87.34** | **86.82** | 86.49 | 93.79 |
| Charniak-Stanford-Mst-Malt | 90.45 | **92.09** | **92.34** | **92.56** | 97.09 |
| Charniak-Stanford-Mst-Zpar | 91.57 | **92.24** | **92.27** | **92.26** | 96.97 |
| Charniak-Stanford-Malt-Zpar | 91.31 | **92.14** | **92.40** | **92.42** | 97.03 |
| Charniak-Mst-Malt-Zpar | 89.60 | 89.48 | 91.71 | 92.08 | 96.79 |
| Stanford-Mst-Malt-Zpar | **88.76** | **88.45** | **88.95** | **88.44** | 96.36 |
| All | 91.43 | 91.77 | **92.44** | **92.58** | 97.41 |

Table 2.2: Initial English Results of the minimum spanning tree algorithm on a combined edge graph. Scores are in **bold** when the ensemble system increased the UAS score over all individual systems.

| X | $weight^x$ |
|---|---|
| 0.5 | 91.77 |
| 2 | 91.84 |
| 4 | 91.98 |
| 6 | 92.44 |
| 8 | 92.47 |
| 9 | 92.52 |
| 10 | **92.58** |
| 11 | 92.57 |
| 12 | 92.57 |
| 16 | 92.43 |

Table 2.3: UAS scores of our ensemble parser with all parsers included at different exponential values ($UAS^x$)

71% accurate. There are also POS errors that are almost always universally bad such as the left parenthesis *(*. Given the large difference in POS errors, weighting an ensemble system by POS would seem like a logical choice in future work. As we can see in Figure 2.2, the varying POS accuracies indicate that the parsing techniques we have incorporated into our ensemble parser, are significantly different. In almost every case in Table 2.4, our Ensemble parser achieves the best accuracy for each POS, while reducing the average relative error rate by 8.64%.

The current weighting systems don't simply default to the best parser or to an average of all errors. In the majority of cases our ensemble parser obtains the top accuracy. In all cases, our ensemble parser is never the worst parser. In cases where the POS is less frequent, our ensemble parser seems to average out the error distribution.

We have shown the benefits of using a maximum spanning tree algorithm in fixed weight ensemble learning for dependency parsing, especially for combining constituent parsers with other dependency parsing techniques. This ensemble method shows improvements over the current state of the art for each individual parser. We also show a theoretical maximum oracle parser which indicates that much more work in this field can take place to improve dependency parsing accuracy toward the oracle score of 97.41%.

We demonstrated that using parsers of different techniques, especially including transformed constituent parsers, can lead to the best accuracy within this ensem-

| POS | Charniak | Stanford | MST | Malt | Zpar | Best Ensemble | Relative Error Reduction |
|---|---|---|---|---|---|---|---|
| PDT | 88.890 | 77.78 | 83.33 | 88.89 | 77.78 | 88.89 | 0.00 |
| CC | 83.540 | 74.73 | 71.16 | 65.84 | 20.39 | **84.63** | 6.64 |
| NNP | 94.590 | 92.16 | 88.04 | 87.17 | 73.67 | **95.02** | 7.81 |
| , | 84.450 | 78.02 | 63.13 | 60.12 | 65.64 | **85.08** | 3.99 |
| WP$ | 90.480 | 71.43 | 85.71 | 90.48 | 0.00 | 90.48 | 0.00 |
| VBN | 91.720 | 89.81 | 90.35 | 89.17 | 88.26 | **93.81** | 25.27 |
| WP | 83.780 | 80.18 | 80.18 | 82.88 | 2.70 | 81.08 | -16.67 |
| RBR | 77.680 | 62.50 | 75.00 | 76.79 | 68.75 | **78.57** | 4.00 |
| CD | 94.910 | 92.67 | 85.19 | 84.46 | 82.64 | **94.96** | 1.02 |
| RP | 96.150 | 95.05 | 97.25 | 95.60 | 94.51 | **97.80** | 42.86 |
| JJ | 95.410 | 92.99 | 94.47 | 93.90 | 89.45 | 95.85 | 0.00 |
| PRP | 97.820 | 96.21 | 96.68 | 95.64 | 95.45 | **98.39** | 26.09 |
| TO | 94.520 | 89.44 | 91.29 | 90.73 | 88.63 | 94.35 | -2.94 |
| EX | 96.490 | 98.25 | 100.00 | 100.00 | 96.49 | **98.25** | 50.00 |
| WRB | 63.910 | 60.90 | 68.42 | 73.68 | 4.51 | 63.91 | 0.00 |
| RB | 86.260 | 79.88 | 81.49 | 81.44 | 80.61 | **87.19** | 6.74 |
| FW | 55.000 | 45.00 | 60.00 | 25.00 | 35.00 | 55.00 | 0.00 |
| WDT | 97.140 | 95.36 | 96.43 | 95.00 | 9.29 | **97.50** | 12.50 |
| VBP | 91.400 | 83.29 | 80.92 | 75.81 | 50.87 | 91.27 | -1.45 |
| JJR | 88.380 | 80.81 | 74.75 | 70.20 | 68.18 | 87.37 | -8.70 |
| VBZ | 91.970 | 87.35 | 83.86 | 80.78 | 57.91 | **92.46** | 6.06 |
| NNPS | 97.620 | 95.24 | 100.00 | 95.24 | 69.05 | 100.00 | 100.00 |
| ( | 73.610 | 75.00 | 54.17 | 58.33 | 15.28 | 73.61 | 0.00 |
| UH | 87.500 | 62.50 | 75.00 | 37.50 | 37.50 | 87.50 | 0.00 |
| POS | 98.180 | 96.54 | 98.54 | 98.72 | 0.18 | **98.36** | 10.00 |
| $ | 82.930 | 80.00 | 67.47 | 66.40 | 52.27 | **84.27** | 7.81 |
| “ | 83.990 | 79.66 | 76.08 | 58.95 | 74.01 | **84.37** | 2.35 |
| : | 77.160 | 72.53 | 45.99 | 44.44 | 53.70 | **79.63** | 10.81 |
| JJS | 96.060 | 90.55 | 88.19 | 86.61 | 82.68 | 93.70 | -60.00 |
| LS | 75.000 | 50.00 | 100.00 | 75.00 | 75.00 | 75.00 | 0.00 |
| . | 96.060 | 93.48 | 91.07 | 84.89 | 87.56 | **97.08** | 25.81 |
| VB | 93.040 | 88.48 | 91.33 | 90.95 | 84.37 | **94.24** | 17.27 |
| MD | 89.550 | 82.02 | 83.05 | 78.77 | 51.54 | **89.90** | 3.28 |
| NNS | 93.100 | 89.51 | 90.68 | 88.65 | 78.93 | **93.67** | 8.26 |
| NN | 93.620 | 90.29 | 88.45 | 86.98 | 83.84 | **94.00** | 6.00 |
| VBD | 93.250 | 87.20 | 86.27 | 82.73 | 64.32 | **93.52** | 4.03 |
| DT | 97.610 | 96.47 | 97.30 | 97.01 | 92.19 | **97.97** | 14.78 |
| # | 100.000 | 80.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 |
| ’ | 88.280 | 83.79 | 81.84 | 69.92 | 79.88 | **90.04** | 15.00 |
| RBS | 90.000 | 76.67 | 93.33 | 93.33 | 86.67 | 90.00 | 0.00 |
| IN | 87.800 | 78.66 | 83.45 | 80.78 | 73.08 | 87.48 | -2.66 |
| SYM | 100.000 | 100.00 | 100.00 | 0.00 | 0.00 | 100.00 | 0.00 |
| PRP$ | 97.640 | 96.07 | 47.22 | 96.86 | 93.12 | 97.45 | -8.33 |
| ) | 70.830 | 77.78 | 96.46 | 55.56 | 12.50 | **72.22** | 4.76 |
| VBG | 85.190 | 82.13 | 82.74 | 82.25 | 81.27 | **89.35** | 28.10 |
| Average | | | | | | | **7.79** |

Table 2.4: POS errors for each of our systems that are used in the ensemble system. We also include the POS error distribution for our best Fixed Weight Ensemble System for English, which is the combination of all parsers using $UAS^{10}$. All POS errors are calculated using the testing data provided by section 23 of the WST. The ensemble system that generated these errors was parameterized on tuning data, section 22 of the WSJ.

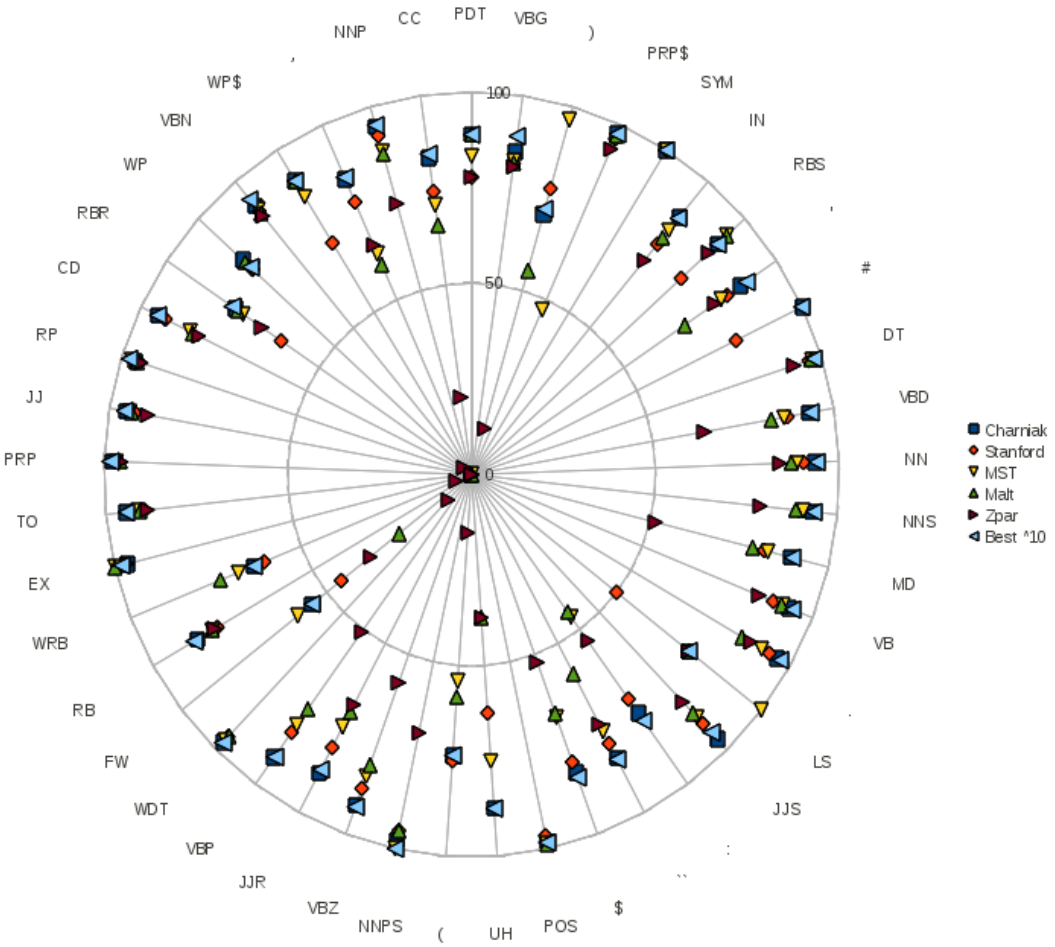


Figure 2.2: POS errors of parsers and the best ensemble system

ble framework. The improvements in accuracy are not simply due to a few edge changes but can be seen to improve the accuracy of the majority of POS tags over all individual systems.

To amplify the effect of POS error reduction further we now look to learn the ensemble weights through our POS error distribution. To do this we will cluster the POS accuracies of our parsers and combine in a similar fashion. These experiments are detailed in Section 2.3.

## 2.3 Fuzzy Clustering

Each dependency parsing technique described thus far can achieves state-of-the-art results, however each technique achieves this result via different error distribution. To minimize these errors and to increase state-of-the-art parsing accuracy, we now examine ensemble techniques that weight graph edges based on part of

speech errors. To do this, we cluster all dependency parsing models on part of speech error counts. This leads us to have a different weighting scheme between dependency parsers for each individual part of speech.

### 2.3.1 Weighting Schemes

We apply three weighting algorithms to the graph structure. First we give each parser uniform weight as we did in the previous section. Second we weight each particular edge by a combination of models weights determined by the part of speech error distribution. Finally we apply exponential scaling to the POS weights ( $\text{POS}^{10}$) to amplify the differences between models. Even in their simplicity, we can see the benefit of this type of combination in an ensemble parser for all three weighting schemes.

- **Uniform**: an edge in the graph gets incremented +1 weight for each matching edge in each parser. If an edge occurs in 4 parsers, the weight is 4.
- **POS**: Each edge of the graph is weighted by a combination of weighting schemes determined by the particular part of speech. This is described in more detail in Section 2.3.2.
- **$\text{POS}^{10}$**: For this weighting scheme we took each POS model score and took it to the 10th power at run time.

### 2.3.2 Determining Part of Speech Clustering Weights

To automatically learn the weights of our models, we turn to part of speech error analysis. We obtain a POS error distribution from our tuning data. Using fuzzy clustering with the cosine distance metric over 20 iterations we find 3 clusters. For a particular part of speech we get a weight corresponding to each cluster that sum to 1. In N clusters we have M weights corresponding to each M models. So for a particular edge we get it's weight by summing each cluster multiplied by all model weights as seen in the equation below.

$$Weight_{edge} = \sum_{i=1}^{N} w_i \sum_{j=1}^{M} w_j$$

If a part of speech did not occur in the tuning data, the weights are equally split across all clusters.

The English experiments are conducted using the same datasets and parsers from the Fixed Weight Ensemble Parsers. We did however add two additional languages, Italian and Japanese, that have different error distributions and different tag sets. For Italian and Japanese we also use train and test CoNLL releases. For tuning data we remove the last 500 sentence of the training data for estimating POS accuracies.

For English we ran $2^5$ model combinations but only report on combinations of three or more models. We conducted $2^9$ combinations of models for each of three weighting schemes for both Japanese and Italian. This is far too much to fully describe in one paper, so to consolidate the results we only display the top ten results for Italian and Japanese for the $POS^{10}$ weighting.

### 2.3.3 Results

Our clustering algorithm is based on Fuzzy-Cmeans algorithm. This algorithm allows for a "data point" to exist partially in many clusters. The cluster centroid is iteratively calculated. For our data points we will use a count of correctly predicted POS's tags for each parser so for one entry we would have NOUN⇒Parser1⇒10, Parser2⇒20 Parser3⇒5, Parser4⇒6. The clusters will then specify the centroids of different clusters of these data points. We use 3 clusters which gives use 3 combinations of model weights.

| **Models** | **Cluster 1** | **Cluster 2** | **Cluster 3** |
|---|---|---|---|
| Charniak | 21.48% | 26.46% | 31.68% |
| Stanford | 20.47% | 24.91% | 27.62% |
| Mst | 20.29% | 24.25% | 21.57% |
| Malt | 19.38% | 20.47% | 10.43% |
| Zpar | 18.38% | 3.91% | 8.70% |

Table 2.5: Cluster weights for each model when averaged based across centroids for our English models

This weighting system models the POS tag in a fashion similar to the POS error distribution. For instance the POS tags "CC" has high weights for Cluster 1. Cluster 1 gives very little weight to Zpar. If we examine the POS errors in Table

| POS | Cluster 1 | Cluster 2 | Cluster 3 |
|---|---|---|---|
| ( | 0.0070 | 0.9928 | 0.0007 |
| ) | 0.0430 | 0.9508 | 0.0067 |
| CC | 0.0080 | 0.9904 | 0.0019 |
| JJ | 1.0000 | 0.0001 | 0.0000 |
| MD | 0.8040 | 0.1820 | 0.0136 |
| NN | 1.0000 | 0.0000 | 0.0000 |
| NNP | 1.0000 | 0.0001 | 0.0000 |
| NNPS | 0.0070 | 0.9917 | 0.0013 |
| PDT | 0.2100 | 0.7277 | 0.0618 |
| PRP | 1.0000 | 0.0001 | 0.0001 |
| VB | 1.0000 | 0.0000 | 0.0000 |
| VBD | 0.9810 | 0.01530 | 0.0036 |
| VBZ | 0.9710 | 0.0250 | 0.0041 |
| WDT | 0.0030 | 0.9963 | 0.0005 |

Table 2.6: The Weights of each Cluster for each individual POS Tag. The POS list has been reduced for space concerns

2.4, Zpar did very poorly on these tags. Overall, it does appear that the clusters tend towards a more balanced weighting scheme while only pointing out true outliers.

Table 2.7 shows the results of the run on our tuning data. The accuracies are higher but comparable to what is seen with a basic uniform weighting scheme. The weights were a combination of the fuzzy clustering weights based on POS errors shown in Table 2.6. This score, **92.54%**, which occurred when all parsers were used, is the best accuracy of all our ensemble techniques and model combinations with English.

**Fuzzy Clustering w/ various training parameters**

For Italian and Japanese we do not have access to a constituency to dependency transformation which limits use to too few parsers. So for these languages we only use Malt Parser and MST Parser but we use different training parameters to create various parsing models. For Malt Parser we use 7 models and for MST Parser we use 2 models.

| Parser | Uniform | POS | $POS^{10}$ | Oracle |
|---|---|---|---|---|
| Charniak-Stanford-Mst | 91.86 | **92.27** | 92.08 | 96.48 |
| Charniak-Stanford-Malt | 91.77 | **92.28** | 92.08 | 96.49 |
| Charniak-Stanford-Zpar | 91.22 | 92.00 | 92.08 | 95.94 |
| Charniak-Mst-Malt | 88.80 | 89.55 | 92.08 | 96.30 |
| Charniak-Mst-Zpar | 90.44 | 91.59 | 92.08 | 96.16 |
| Charnial-Malt-Zpar | 88.61 | 91.30 | 92.08 | 96.21 |
| Stanford-Mst-Malt | 87.84 | **87.94** | 87.88 | 95.62 |
| Stanford-Mst-Zpar | **89.12** | **89.89** | 87.88 | 95.57 |
| Stanford-Malt-Zpar | **88.61** | **89.60** | **89.58** | 95.47 |
| Mst-Malt-Zpar | **86.99** | **87.34** | 86.49 | 93.79 |
| Charniak-Stanford-Mst-Malt | 90.45 | **92.09** | **92.45** | 97.09 |
| Charniak-Stanford-Mst-Zpar | 91.57 | **92.24** | **92.49** | 96.97 |
| Charniak-Stanford-Malt-Zpar | 91.31 | **92.15** | 92.08 | 97.03 |
| Charniak-Mst-Malt-Zpar | 89.60 | 89.53 | 92.08 | 96.79 |
| Stanford-Mst-Malt-Zpar | **88.76** | **88.40** | 87.88 | 96.36 |
| All | 91.43 | 91.84 | **92.54** | 97.41 |

Table 2.7: UAS scores of our ensemble parser using POS fuzzy clustering weights for English. Values are bolded wherever the result is greater than all individual models within the ensemble system.

| Models | IT-UAS | JA-UAS |
|---|---|---|
| mstnonproj | 72.89% | 78.65% |
| mstproj | 76.03% | 84.04% |
| nivreeager | 82.08% | 92.99% |
| nivrestandard | 81.11% | 92.87% |
| 2planar | 82.58% | 90.01% |
| planar | 81.89% | 90.81% |
| stackeager | 81.44% | 93.25% |
| stacklazy | 81.27% | 93.43% |
| stackproj | 81.57% | 91.87% |

Table 2.8: Our baseline parsers and corresponding UAS used in our ensemble experiments for Italian and Japanese

Table 2.9 shows the scores for Japanese. The scores are taken from the top 10 performing systems for $POS^{10}$ weighting scheme. All of the top 10 systems performed at or better than the best performing individual model. This is a promising result since Japanese already has a relatively high baseline compared to other languages. Extending the results from the English experiments we might see even greater improvement given more diversity of models instead of only Malt Parser and MST Parser.

| **Model Combos** | **Uniform** | **POS** | $POS^{10}$ | **Oracle** |
|---|---|---|---|---|
| mstnonproj-2planar-nivrestandard nivreeager-stacklazy-stackeager | 92.96% | 93.00% | 93.43% | 97.51% |
| mstnonproj-nivrestandard-nivreeager-stackeager | 93.28% | 93.43% | 93.43% | 97.08% |
| mstnonproj-nivrestandard-nivreeager-stacklazy | 93.35% | 93.43% | 93.43% | 97.04% |
| mstnonproj-planar-nivreeager-stacklazy | 92.63% | 93.45% | 93.43% | 97.11% |
| mstnonproj-planar-nivrestandard nivreeager-stacklazy | 92.73% | 93.12% | 93.43% | 97.32% |
| mstnonproj-2planar-nivreeager-stacklazy | 93.24% | 93.43% | 93.45% | 97.22% |
| mstnonproj-nivrestandard-nivreeager | 93.29% | 93.45% | 93.45% | 96.59% |
| mstnonproj-planar-nivrestandard nivreeager-stacklazy-stackeager | 93.38% | 93.45% | 93.45% | 97.46% |
| mstnonproj-2planar-nivrestandard nivreeager-stacklazy | 92.84% | 93.08% | 93.50% | 97.41% |
| mstnonproj-2planar-nivrestandard-nivreeager | 93.26% | 93.59% | 93.59% | 97.11% |
| Average over all Combos | 92.41% | 92.65% | 92.63% | 96.84% |

Table 2.9: UAS scores of our ensemble parser using POS fuzzy clustering weights for Japanese

Table 2.10 shows the scores for Italian. The scores are taken from the top 10 performing systems for $POS^{10}$ weighting scheme as well. Overall none of the combinations were able to achieve as high of a score as the best individual model which was the *2planar* trained Malt Parser. Of all the weighting schemes, $POS^{10}$ performed the best with an average accuracy of 77.38% and a max accuracy of 81.34%. Although this did not beat *planar2* the average POS error reduction, described in Section 2.3.3, shows us another story on how this ensemble system may be used.

| Model Combos | Uniform | POS | $POS^{10}$ | Oracle |
|---|---|---|---|---|
| mstnonproj-2planar-nivrestandard nivreeager-stacklazy-stackeager | 80.77% | 80.57% | 81.04% | 89.82% |
| mstnonproj-nivrestandard-nivreeager-stackeager | 81.06% | 81.10% | 81.04% | 90.36% |
| mstnonproj-nivrestandard-nivreeager-stacklazy | 80.61% | 80.77% | 81.08% | 89.78% |
| mstnonproj-planar-nivreeager-stacklazy | 80.75% | 80.89% | 81.10% | 90.72% |
| mstnonproj-planar-nivrestandard nivreeager-stacklazy | 80.61% | 80.75% | 81.10% | 89.87% |
| mstnonproj-2planar-nivreeager-stacklazy | 80.57% | 80.77% | 81.14% | 89.68% |
| mstnonproj-nivrestandard-nivreeager | 80.69% | 80.99% | 81.16% | 90.62% |
| mstnonproj-planar-nivrestandard nivreeager-stacklazy-stackeager | 80.75% | 81.04% | 81.24% | 90.58% |
| mstnonproj-2planar-nivrestandard-nivreeager-stacklazy nivreeager-stacklazy | 80.95% | 81.14% | 81.32% | 87.40% |
| mstnonproj-2planar-nivrestandard-nivreeager | 80.34% | 80.93% | 81.34% | 88.30% |
| Average over all Combos | 77.33% | 77.36% | 77.38% | 87.98% |

Table 2.10: UAS scores of our ensemble parser using POS fuzzy clustering weights for Italian

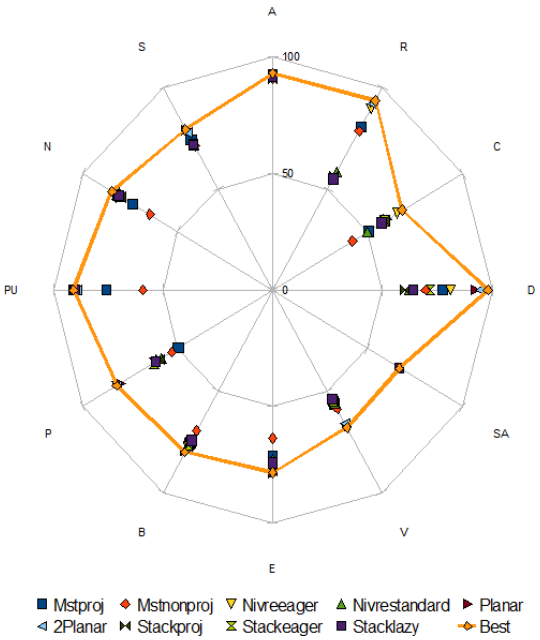


Figure 2.3: We can visually see how the ensemble system reduces POS errors across each POS. The line connects the best ensemble system for for Italian on each of it's POS tags

| POS | Charniak | Stanford | MST | Malt | Zpar | Best Ensemble | Relative Error Reduction |
|---|---|---|---|---|---|---|---|
| PDT | 88.89 | 77.78 | 83.33 | 88.89 | 77.78 | 88.89 | 0.00 |
| CC | 83.54 | 74.73 | 71.16 | 65.84 | 20.39 | **84.63** | 6.64 |
| NNP | 94.59 | 92.16 | 88.04 | 87.17 | 73.67 | **95.02** | 7.81 |
| VBN | 91.72 | 89.81 | 90.35 | 89.17 | 88.26 | **93.81** | 25.27 |
| JJ | 95.41 | 92.99 | 94.47 | 93.90 | 89.45 | 95.85 | 0.00 |
| PRP | 97.82 | 96.21 | 96.68 | 95.64 | 95.45 | **98.39** | 26.09 |
| TO | 94.52 | 89.44 | 91.29 | 90.73 | 88.63 | 94.35 | -2.94 |
| RB | 86.26 | 79.88 | 81.49 | 81.44 | 80.61 | **87.19** | 6.74 |
| FW | 55.00 | 45.00 | 60.00 | 25.00 | 35.00 | 55.00 | 0.00 |
| WDT | 97.14 | 95.36 | 96.43 | 95.00 | 9.29 | **97.50** | 12.50 |
| VB | 93.04 | 88.48 | 91.33 | 90.95 | 84.37 | **94.24** | 17.27 |
| MD | 89.55 | 82.02 | 83.05 | 78.77 | 51.54 | **89.90** | 3.28 |
| NNS | 93.10 | 89.51 | 90.68 | 88.65 | 78.93 | **93.67** | 8.26 |
| NN | 93.62 | 90.29 | 88.45 | 86.98 | 83.84 | **94.00** | 6.00 |
| DT | 97.61 | 96.47 | 97.30 | 97.01 | 92.19 | **97.97** | 14.78 |
| Average | | | | | | | **7.79** |

Table 2.11: POS errors for each of our systems that are used in the ensemble system for English. We also include the POS error distribution for our best Ensemble system. All POS errors are calculated using the testing data, section 23 of the WST. The ensemble system that generated these errors was parameterized on tuning data, section 22 of the WSJ. We only display a reduced set of POS tags for space but the Average is over all POS tags including those not shown.

**POS error reduction**

Figure 2.3 shows visually how the best ensemble system for Italian is at or better than all other parsers in terms of POS errors. This shows that the ensemble system is not just an averaging of errors but actually does reduce error for each individual POS. Similar results can be seen in English as well.

| POS | Mst proj | Mstnon proj | Nivre eager | Nivre standard | Planar | 2Planar | Stack proj | Stack eager | Stack lazy | Best ensemble | Relative err reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 92.33 | 90.67 | 91.67 | 91.67 | 92.00 | 92.00 | 91.00 | 91.67 | 91.67 | 93.00 | 8.70% |
| S | 74.55 | 70.98 | 78.62 | 71.56 | 79.13 | 78.11 | 72.00 | 72.36 | 72.00 | 79.56 | 2.09% |
| N | 73.78 | 64.63 | 82.32 | 80.49 | 81.71 | 83.54 | 79.88 | 82.32 | 81.10 | 84.76 | 7.41% |
| PU | 75.68 | 59.10 | 90.63 | 89.67 | 90.08 | 89.13 | 90.63 | 90.22 | 90.22 | 90.90 | 2.90% |
| P | 49.76 | 53.17 | 81.95 | 58.54 | 80.00 | 81.95 | 59.51 | 62.44 | 61.46 | 81.95 | 0.00% |
| B | 75.63 | 69.54 | 77.66 | 76.14 | 79.70 | 78.17 | 75.13 | 77.16 | 74.11 | 80.20 | 2.50% |
| E | 71.32 | 63.38 | 78.10 | 72.47 | 78.23 | 77.46 | 76.18 | 74.90 | 74.14 | 78.23 | 0.00% |
| V | 56.45 | 58.87 | 67.80 | 56.31 | 67.66 | 66.10 | 55.04 | 55.18 | 54.18 | 67.80 | 0.00% |
| SA | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 66.67 | 0.00% |
| D | 77.36 | 69.81 | 81.13 | 62.26 | 92.45 | 94.34 | 60.38 | 71.70 | 64.15 | 98.11 | 66.67% |
| C | 50.30 | 42.01 | 65.68 | 49.70 | 60.36 | 58.58 | 59.17 | 59.17 | 57.40 | 68.05 | 6.90% |
| R | 80.79 | 78.82 | 89.66 | 58.37 | 92.86 | 91.87 | 56.16 | 54.93 | 54.93 | 93.60 | 10.34% |
| Avg | 70.38 | 65.64 | 79.32 | 69.49 | 80.07 | 79.83 | 70.14 | 71.56 | 70.17 | 81.90 | **8.96%** |

Table 2.12: POS errors for Italian and its relative error reduction

Next in Table 2.12 we look at the relative POS error reduction rate and its average across all parts of speech. Table 2.12 indicates that while the POS clustering ensemble system did not perform better than the best overall system, it did reduce error on an edge by edge level in terms of POS error. This indicates that locally the system makes better decisions but the overall structure of the parse tree is incorrect. To correct this we must look at combining POS clustering with an ensemble method that will favor an overall structure.

## 2.4 Model Classification

This section bridges the gap between using a mixture of prediction for each node/edge relationship to a discrete choice where we let one parser select the head of a node. The parse selection works similarly to the section above in selecting the final tree from a graph structure. However, using a discrete choice brings new complications. If we use a classifier on each node and this allows each node to be selected by a different parser possibly. It is very likely that we will have cycles and more importantly we will have a disconnected graph.

### 2.4.1 Methodology

Morphologically rich and under-resourced languages are often short on training data or require much higher amounts of training data due to the increased size of their lexicon. This section examines a new approach for addressing morphologically rich and under-resourced languages with little training data to start.

We demonstrate this technique on three languages. Tamil, using a combination of different dependency parsers, Indonesian using a combination of models produced from only one parser, and English using a combination of dependency and constituent parsers.

For Tamil we show a statistically significant 5.44% improvement over the average dependency model and a statistically significant 0.52% improvement over the best individual system. For Indonesian we show increases the dependency accuracy by 4.92% on average and 1.99% over the best individual system. For English our model improves on the best individual system by a little more than half a percentage.

#### Languages and Data Sets

Using Tamil as our test language, we create 9 dependency parse models with a limited amount of training data. Using these models we train an SVM classifier using only the model agreements as features (described in Section 2.4.1). We use this SVM classifier on an edge by edge decision to form an ensemble parse tree. Using only model agreements as features allows this method to remain language independent and applicable to a wide range of morphologically rich languages.

Tamil belongs to Dravidian family of languages and is mainly spoken in southern India and also in parts of Sri Lanka, Malaysia and Singapore. Tamil is agglutinative and has a rich set of morphological suffixes. Tamil has nouns and verbs as two major word classes, and hundreds of word forms can be produced by the application of concatenative and derivational morphology. Tamil's rich morphology makes the language free word order except that it is strictly *head final*.

Only few attempts were reported in the literature on the development of a treebank for Tamil. Our experiments are based on the openly available treebank (TamilTB) [22]. Development of TamilTB is still in progress and the initial results for TamilTB appeared in [21]. Previous parsing experiments were done

using a rule based approach which utilized morphological tagging and identification of clause boundaries to parse the sentences. The results were also reported for Maltparser and MST parser. When the morphological tags were available during both training and testing, the rule based approach performed better than Malt and MST parsers.

Table 2.13 shows the statistics of the TamilTB Treebank. The last 2 rows indicate how many word types have unique tags and how many have two tags. The table illustrates that most of the word types can be uniquely identified with single morphological tag; only around 120 word types take more than one morphological tag.

| **Description** | **value** |
|---:|---:|
| #Sentences | 600 |
| #Words | 9581 |
| #Word types | 3583 |
| #Tagset size | 234 |
| #Types with unique tags | 3461 |
| #Types with 2 tags | 112 |

Table 2.13: TamilTB: data statistics

The Indonesian treebank that we use in this work is a collection of manually annotated Indonesian dependency trees. It consists of 100 Indonesian sentences with 2705 tokens and a vocabulary size of 1015 unique tokens. The sentences are taken from the IDENTIC corpus [12]. The raw version of the sentences originally were taken from the BPPT articles in economy from the PAN localization [18] project output. The treebank used Parts-Of-Speech tags (POS tags) provided by MorphInd [13]. Since the MorphInd output is ambiguous, the tags are also disambiguated and corrected manually, including the unknown POS tag.

For English we used the same parsers and data sets described in Section 2.2.

### Process Flow

When dealing with small data sizes it is often not enough to show a simple accuracy increase. This increase can be very reliant on the training/tuning/testing data splits as well as the sampling of those sets.

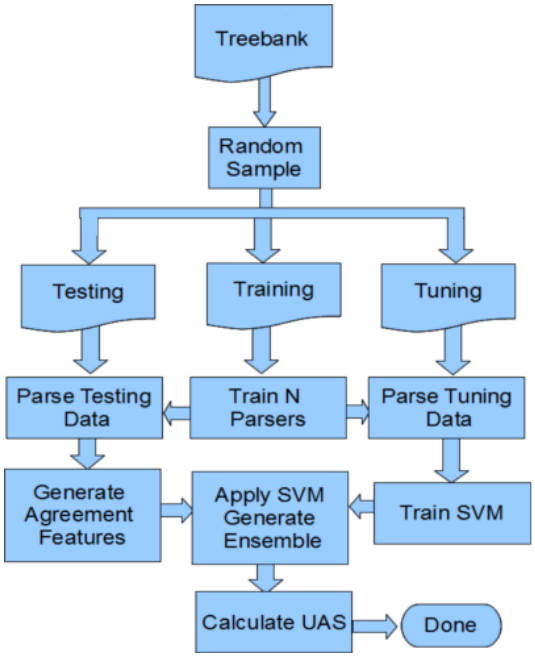


Figure 2.4: Process Flow for one run of our SVM Ensemble system. This Process in its entirety was run 100 times for each data set split.

For Tamil our experiments are conducted over 8 training/tuning/testing data split configurations with training data ranging from 70-90 percent of the data. For each configuration we randomly sample without replacement the training/-tuning/testing data and rerun the experiment 100 times. These 800 runs, each on different samples, allow us to better show the overall effect on the accuracy metric as well as the statistically significant changes as described in Section 2.2.1. Figure 2.4 shows this process flow for one run of this experiment.

To test the effects of the training, tuning, and testing data for Indonesian we try 18 different data split configurations, each one being sampled 100 times. The data splits in Section 2.4.2 use the format training-tuning-testing. So 70-20-10 means we used 70% of the Indonesian Treebank for training, 20% for tuning the SVM classifier, and 10% for evaluation.

For English we did not retrain the parsers so we use the standard data splits configuration by using WSJ section 22 for tuning and section 23 for testing.

### Parsers

For Tamil we generate two models using MST parser [15], one projective and one non-projective to use in our ensemble system. Additionally we generate many transition-based parsers. We make use of Malt Parser [16], which in the CoNLL

shared tasks was often tied with the best performing systems. For this parser we generate 7 different models using different training parameters and use them as input into our ensemble system along with the two Graph-based models described above.

For Indonesian we test the system using only 1 type of parser, in this case Malt Parser. We use the same 7 models described above.

For English we did not change the parser configurations described in Section 2.2.1

| **Training Parameter** | **Model Description** |
|---|---|
| nivreeager | Nivre arc-eager |
| nivrestandard | Nivre arc-standard |
| stackproj | Stack projective |
| stackeager | Stack eager |
| stacklazy | Stack lazy |
| planar | Planar eager |
| 2planar | 2-Planar eager |

Table 2.14: Table of the Malt Parser Parameters used during training. Each entry represents one of the parsing algorithms used in our experiments. For more information see http://www.maltparser.org/options.html

**Ensemble SVM System and Features**

We train our SVM classifier using only model agreement features. Using our tuning set, for each correctly predicted dependency edge, we create $\binom{N}{2}$ features where $N$ is the number of parsing models. We do this for each model which predicted the correct edge in the tuning data. So for $N = 3$ the first feature would be a 1 if model 1 and model 2 agreed, feature 2 would be a 1 if model 1 and model 3 agreed, and so on. This feature set is widely applicable to many languages since it does not use any additional linguistic tools.

For each edge in the ensemble graph, we use our classifier to predict which model should be correct, by first creating the model agreement feature set for the current edge of the unknown test data. The SVM predicts which model should be correct and this model then decides to which head the current node is attached. At the end of all the tokens in a sentence, the graph may not be connected and will likely

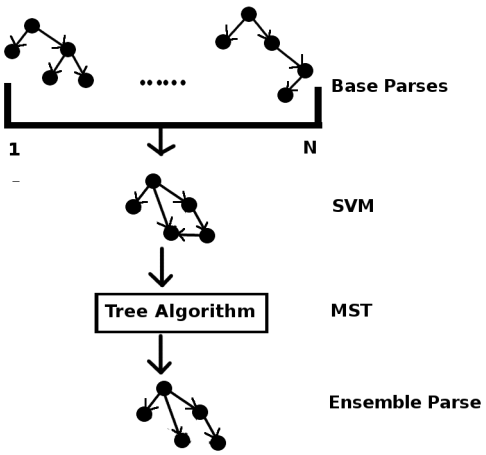


Figure 2.5: General flow to create an Ensemble parse tree with a meta-classifier

have cycles. Using a Perl implementation of minimum spanning tree[2], in which each edge has a uniform weight, we obtain a minimum spanning forest, where each component is then connected and cycles are eliminated in order to achieve a well formed dependency structure. Figure 2.5 gives a graphical representation of how the SVM decision and MST algorithm create a final Ensemble parse tree which is similar to the construction used in [7, 5]. Future iterations of this process could use a multi-label SVM or weighted edges based on the parser's accuracy on tuning data.

#### Evaluation

To test statistical significance we use Wilcoxon paired signed-rank test. For each data split configuration we have 100 iterations of the experiment. Each model is compared against the same samples so a paired test is appropriate in this case. We report statistical significance values for $p < 0.01$.

### 2.4.2 Results and Discussion

#### Tamil Results

For each of the data splits, Table 2.15 shows the percent increase in our SVM system over both the average of the 9 individual models and over the best individual model. As the Table 2.15 shows, our approach seems to decrease in value along with the decrease in tuning data. In both cases when we only used 5% tuning data we did not get any improvement in our average UAS scores. Examining

[2]http://search.cpan.org/ pajas/Graph-ChuLiuEdmonds-0.05/lib/Graph/ChuLiuEdmonds.pm

Table 2.16 shows that the decrease in the 90-5-5 split is not statistically significant however the decrease in 85-5-10 is a statistically significant drop. However, the increases in all data splits are statistically significant except for the 60-20-20 data split.

| **Data Split** | Average SVM UAS | % Increase over Avg | % Increase over Best |
|---|---|---|---|
| 70-20-10 | 76.50% | 5.13% | 0.52% |
| 60-20-20 | 76.36% | 5.68% | 0.72% |
| 60-30-10 | 75.42% | 5.44% | 0.52% |
| 60-10-30 | 75.66% | 4.83% | 0.10% |
| 85-5-10 | 75.33% | 3.10% | -1.21% |
| 90-5-5 | 75.42% | 3.19% | -1.10% |
| 80-10-10 | 76.44% | 4.84% | 0.48% |

Table 2.15: Average increases and decreases in UAS score for different Training-Tuning-Test samples in Tamil. The average was calculated over all 9 models while the best was selected for each data split

It appears that the size of the tuning and training data matters more than the size of the test data. Given that the TamilTB is relatively small (see Table 2.13) when compared to other CoNLL treebanks, we expect that this ratio may shift more when additional data is supplied since the amount of out of vocabulary, OOV, words will decrease as well. As OOV words decrease, we expect the use of additional test data to have less of an effect.

The traditional approach of using as much data as possible for the training does not seem to be as effective as partitioning more data for tuning an SVM. For instance the high test training percentage we use is 90% applied to training with 5% for tuning and testing each. In this case the best individual model had a UAS score 76.25% and the SVM had a UAS of 75.42%. One might think using 90% of the data would achieve a higher overall UAS score than using less training data. On the contrary, we achieve a better UAS score on average using only 60%, 70%, 80%, and 85% of the data towards training. This additional data spent for tuning appears to be worth the cost.

**Indonesian Results**

For each of the data splits, Table 2.17 shows the percent increase in our SVM system over both the average of the 7 individual models and over the best indi-

| **Model** | 70-20-10 | 60-20-20 | 60-30-10 | 60-10-30 | 85-5-10 | 90-5-5 | 80-10-10 |
|---|---|---|---|---|---|---|---|
| 2planar | * | * | * | * | * | * | ** |
| mstnonproj | * | * | * | * | * | * | ** |
| mstproj | * | * | * | * | * | * | ** |
| nivreeager | * | * | * | * | ** | $x$ | * |
| nivrestandard | * | * | ** | $x$ | * | * | * |
| planar | * | * | * | * | * | * | ** |
| stackeager | * | * | * | $x$ | * | ** | * |
| stacklazy | * | * | * | $x$ | * | ** | * |
| stackproj | ** | * | * | $x$ | ** | ** | ** |

Table 2.16: Statistical Significance Table for different Training-Tuning-Test samples for Tamil. Each experiment was sampled 100 times and Wilcoxon Statistical Significance was calculated for our SVM model's increase/decrease over each individual model. $* = p < 0.01$ , $** p =< 0.05$, $x = p \geq 0.05$

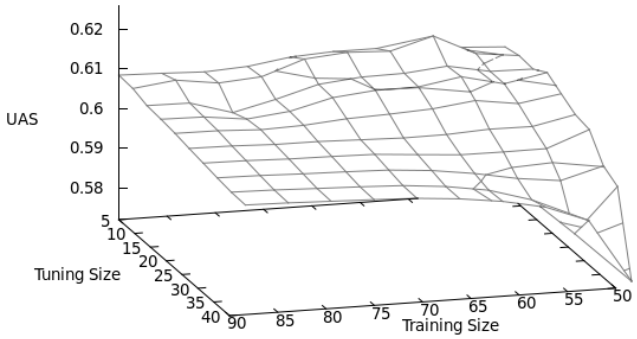


Figure 2.6: Surface plot of the UAS score for the tuning and training data split.

| Data Split | Average SVM UAS | % Increase over Average | % Increase over Best | Statistical Significant |
|---|---|---|---|---|
| 50-40-10 | 60.01% | 10.65% | 4.34% | Y |
| 60-30-10 | 60.28% | 10.35% | 4.41% | Y |
| 70-20-10 | 62.25% | 10.10% | 3.70% | Y |
| 80-10-10 | 60.88% | 8.42% | 1.94% | Y |
| 50-30-20 | 61.37% | 9.73% | 4.58% | Y |
| 60-20-20 | 62.39% | 9.62% | 3.55% | Y |
| 70-10-20 | 62.48% | 7.50% | 1.90% | Y |
| 50-20-30 | 61.71% | 9.48% | 4.22% | Y |
| 60-10-30 | 62.57% | 7.89% | 2.47% | Y |
| 90-5-5 | 60.85% | 0.56% | 0.56% | N |
| 85-10-5 | 61.15% | 0.56% | 0.56% | Y |
| 80-15-5 | 59.23% | 0.54% | 0.54% | Y |
| 75-20-5 | 60.32% | 0.54% | 0.54% | Y |
| 70-25-5 | 59.54% | 0.54% | 0.54% | Y |
| 65-30-5 | 59.76% | 0.54% | 0.54% | Y |
| 60-35-5 | 59.31% | 0.53% | 0.53% | Y |
| 55-40-5 | 57.27% | 0.50% | 0.50% | Y |
| 50-45-5 | 57.72% | 0.51% | 0.51% | Y |

Table 2.17: Average increases and decreases in UAS score for different Training-Tuning-Test samples in Indonesian. The average was calculated over all 7 models while the best was selected for each data split. Each experiment was sampled 100 times and Wilcoxon Statistical Significance was calculated for our SVM model's increase/decrease over each individual model. $Y = p < 0.01$ and $N = p \geq 0.01$ for all models in the data split

vidual model. As the Table 2.17 shows, we obtain above average UAS scores in every data split. The increase is statistically significant in all data splits except for one, the 90-5-5 split. This seems to be logical since this data split has the least difference in training data between systems, with only 5% tuning data. Our highest average UAS score was with the 70-20-10 split with a UAS of 62.48%. The use of 20% tuning data is of interest since it was significantly better than models with 10%-25% more training data as seen in Figure 2.6. This additional data spent for tuning appears to be worth the cost.

The selection of the test data seems to have caused a difference in our results. While all our ensemble SVM parsings system have better UAS scores, it is a lower increase when we only use 5% for testing. Which in our treebank means we are only using 5 sentences randomly selected per experiment. This does not seem to be enough to judge the improvement.

**English Results**

For English we used only preexisting models. They were trained on the usual Penn treebank split, sections 02-21 for training, tuned with section 22 and tested with section 23. With the SVM system we obtained an accuracy is 92.6% which is higher then all individual systems and all previous ensemble systems described here. We feel this number could be increased by retraining the models and employing multiple Malt and MST models such as was done in the previous experiments for other languages.

### 2.4.3 Conclusion

We have shown a new SVM based ensemble parser that uses only dependency model agreement features. The ability to use only model agreements allows us to keep this approach language independent and applicable to a wide range of morphologically rich and under-resourced languages. For Tamil we show a statistically significant 0.52% improvement over the best individual system. For Indonesian we show increases the dependency accuracy by 1.99% over the best individual system. For English our model improves over the best individual system by a little more than half a percentage.

# 3. Parsing Effects on Machine Translation

## 3.1 Introduction

Dependency parsing is typically evaluated by its labeled and unlabeled accuracy scores for a particular parse tree. The models are rarely used to evaluate performance on other NLP tasks. Although rarely evaluated, they are often used, as parsing is a main component of many NLP tasks. In this chapter we evaluate our three ensemble systems on one particular task, Machine Translation.

## 3.2 Methodology

### 3.2.1 Data Sets

To test our parsers' results in a full machine translation run, we translated newstest2011 data from the Workshop for Machine Translation (WMT). Although we show parsing results for other languages we are only translating text with the English to Czech pairing (en-cs). Newstest2011 dataset contains 2,973 sentences from 1 genre of text.

### 3.2.2 Parsers

We use mostly the parsers described in Chapter 2. Since we are only translating an English to Czech scenario, we are only using the English parsers. We use the different parsers in separate translation runs each time in the same Treex parsing block. So each translation scenario only differs in the parser used and nothing else. The parsers used are as follows:

- MST
- MST with chunking
- Malt

- Malt with chunking
- Zpar
- Charniak
- Stanford
- Fixed Weight Ensemble
- Fuzzy Cluster
- SVM

Both MST and Malt have variants "with chunking", in these cases the parser is applied to a subsegment of the sentences when applicable. This could be when we have a sentence inside parenthesis or when we have an apposition.

### 3.2.3 Evaluation

For Machine Translation we report two automatic evaluation scores, BLEU, and NIST.

The BLEU (*B*i*L*ingual *E*valuation *U*nderstudy) score is an automatic scoring mechanism for machine translation that is quick and can be reused as a benchmark across machine translation tasks. BLEU is calculated as the geometric mean of n-grams multiplied by a brevity penalty, comparing a machine translation and a reference text [19]. This experiment compares the two parsing systems against each other using the above metrics. In both cases the test set data is sampled 1,000 times without replacement to calculate statistical significance using a pairwise comparison.

NIST, from the *N*ational *I*nstitute of *S*tandards and *T*echnology, is based upon the BLEU n-gram approach however it is also weighted towards discovering more "informative" n-grams. The more rare an n-gram is, the higher the weight for a correct translation of it will be. This, in effect, lowers the importance of translating punctuation and common words such as articles.

## 3.3 Results and Discussion

Results of each machine translation run can be found in Table 3.1. The hope of our ensemble system was to form better structures to be used by NLP systems. It appears, by the results, that we are on track. Each of our combined ensemble systems achieves a higher score than any machine translation scenario that uses just one parser.

| **Parser** | **UAS** | **NIST** | **BLEU** |
|---|---|---|---|
| MST | 86.49 | 5.5898 | 13.58 |
| MST with chunking | 86.57 | 5.6346 | 14.00 |
| Malt | 84.51 | 5.5702 | 13.48 |
| Malt with chunking | 87.01 | 5.6025 | 13.80 |
| Zpar | 76.06 | 5.4635 | 12.48 |
| Charniak | 92.08 | 5.6561 | 13.95 |
| Stanford | 87.88 | 5.5970 | 13.63 |
| **Fixed Weight Ensemble** | 92.58 | 5.6831 | 14.04 |
| **Fuzzy Cluster** | 92.54 | 5.6820 | 14.06 |
| **SVM** | 92.60 | 5.6837 | 14.11 |

Table 3.1: Table showing the NIST and BLEU scores for each Treex Machine Translation run along with the UAS score of the parser used in the translation scenario.

All of our ensemble systems currently only use the base Malt and MST parser. The results indicate that the chunking variants do better in translation. Incorporating these into our system should also increase our Fixed Weight Ensemble, Fuzzy Clustering, and SVM systems. The results also point out one additional point of research, in that time spent improving the accuracy of early input NLP tools is well worth the effort as they have a cascading effect down an NLP pipeline.

## 3.4 Conclusion

With these experiments we have shown that is useful to not only characterize parsing results by their accuracy but also show their significance in other NLP

tasks. We showed that by only using combined parsers in a machine translation scenario you can achieve an improved NIST and BLEU score in each of our three combinations.

# 4. Conclusion and Future Work

We have looked at the problem of dependency parsing and machine translation through the lens of ensemble parsing. We have introduced and shown results with three different ensemble parsers. First a fixed weight spanning tree approach that combines many different parsers into a weight graph. Second, a fuzzy clustering approach that chooses dependency edges from an error distribution of different dependency parsers. Third, and most successful, we use a meta classifier support vector machine that chooses the model that is best for a particular edge. In each case we have shown that an ensemble parser can outperform individual parsers. We show this in various languages: English, Italian, Japanese, Tamil, and Indonesian.

While having a slight increase in parser accuracy is nice, we wanted to show that it has further application in the NLP pipeline. We demonstrate this with machine translation using the Treex Framework for English to Czech translation. Changing only the dependency parser in a translation scenario we show increases in the NIST and BLEU scores for each of our ensemble parsers indicating that increases in the source side UAS score will have positive outcomes much further down the translation scenario.

Future work should be focused on showing why this improvement occurs. This will entail a more detailed analysis of the of errors each Ensemble system is correcting. It will also include an analysis of the sentences changed in each machine translation scenario. Additionally, adding more complex feature sets to our meta-classifier, such as neighbor tags and n-grams, should produce improved results.

# Bibliography


[1] Buchholz, S., and Marsi, E. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning* (Stroudsburg, PA, USA, 2006), CoNLL-X '06, Association for Computational Linguistics, pp. 149–164.

[2] Charniak, E., and Johnson, M. Coarse-to-Fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 2005), ACL '05, Association for Computational Linguistics, pp. 173–180.

[3] Dietterich, T. G. Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems* (London, UK, 2000), MCS '00, Springer-Verlag, pp. 1–15.

[4] Eisner, J. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)* (Copenhagen, August 1996), Association for Computational Linguistics, pp. 340–345.

[5] Green, N., and Žabokrtský, Z. Hybrid Combination of Constituency and Dependency Trees into an Ensemble Dependency Parser. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data* (Avignon, France, April 2012), Association for Computational Linguistics, pp. 19–26.

[6] Haffari, G., Razavi, M., and Sarkar, A. An Ensemble Model that Combines Syntactic and Semantic Clustering for Discriminative Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Portland, Oregon, USA, June 2011), Association for Computational Linguistics, pp. 710–714.

[7] Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson, M., and Saers, M. Single Malt or Blended? A Study in Multilingual Parser Optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* (2007), pp. 933–939.

[8] Holan, T., and Žabokrtský, Z. Combining Czech Dependency Parsers. In *Proceedings of the 9th international conference on Text, Speech and Dialogue* (Berlin, Heidelberg, 2006), TSD'06, Springer-Verlag, pp. 95–102.

[9] Johansson, R., and Nugues, P. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA 2007* (Tartu, Estonia, May 25-26 2007), pp. 105–112.

[10] Klein, D., and Manning, C. D. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 2003), ACL '03, Association for Computational Linguistics, pp. 423–430.

[11] Kübler, S., McDonald, R., and Nivre, J. *Dependency parsing*. Synthesis lectures on human language technologies. Morgan & Claypool, US, 2009.

[12] Larasati, S. D. IDENTIC Corpus:Morphologically Enriched Indonesian-English Parallel Corpus. 2012.

[13] Larasati, S. D., Kuboň, V., and Zeman, D. Indonesian Morphology Tool (MorphInd): Towards an Indonesian Corpus. *Systems and Frameworks for Computational Morphology* (2011), 119–129.

[14] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. Building a large annotated corpus of English: the Penn Treebank. *Comput. Linguist. 19* (June 1993), 313–330.

[15] McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (Vancouver, British Columbia, Canada, October 2005), Association for Computational Linguistics, pp. 523–530.

[16] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering 13*, 2 (2007), 95–135.

[17] Nivre, J., and McDonald, R. Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of ACL-08: HLT* (Columbus, Ohio, June 2008), Association for Computational Linguistics, pp. 950–958.

[18] PAN, L. P. PAN Localization Project, 2010.

[19] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Morristown, NJ, USA, 2002), ACL '02, Association for Computational Linguistics, pp. 311–318.

[20] Popel, M., Mareček, D., Green, N., and Žabokrtský, Z. Influence of Parser Choice on Dependency-Based MT. In *Proceedings of the Sixth Workshop on Statistical Machine Translation* (Edinburgh, Scotland, July 2011), Association for Computational Linguistics, pp. 433–439.

[21] Ramasamy, L., and Žabokrtský, Z. Tamil dependency parsing: results using rule based and corpus based approaches. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I* (Berlin, Heidelberg, 2011), CICLing'11, pp. 82–95.

[22] Ramasamy, L., and Žabokrtský, Z. Prague Dependency Style Treebank for Tamil. In *Proceedings of LREC 2012* (İstanbul, Turkey, 2012).

[23] Sagae, K., and Lavie, A. Parser Combination by Reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers* (New York City, USA, June 2006), Association for Computational Linguistics, pp. 129–132.

[24] Sagae, K., and Tsujii, J. Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* (Prague, Czech Republic, June 2007), Association for Computational Linguistics, pp. 1044–1050.

[25] Surdeanu, M., and Manning, C. D. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 2010), HLT '10, Association for Computational Linguistics, pp. 649–652.

[26] Žabokrtský, Z., Ptáček, J., and Pajas, P. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL* (2008), pp. 167–170.

[27] Zeman, D., and Žabokrtský, Z. Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *In: Proceedings of the 9th International Workshop on Parsing Technologies* (2005).

[28] Zhang, Y., and Clark, S. Syntactic Processing Using the Generalized Perceptron and Beam Search. *Computational Linguistics 37*, 1 (2011), 105–151.

# List of Tables

# List of Figures

## The ÚFAL/CKL Technical Report Series

### ÚFAL

ÚFAL (Ústav formální a aplikované lingvistiky; http://ufal.mff.cuni.cz) is the Institute of Formal and Applied linguistics, at the Faculty of Mathematics and Physics of Charles University, Prague, Czech Republic. The Institute was established in 1990 after the political changes as a continuation of the research work and teaching carried out by the former Laboratory of Algebraic Linguistics since the early 60s at the Faculty of Philosophy and later the Faculty of Mathematics and Physics. Together with the "sister" Institute of Theoretical and Computational Linguistics (Faculty of Arts) we aim at the development of teaching programs and research in the domain of theoretical and computational linguistics at the respective Faculties, collaborating closely with other departments such as the Institute of the Czech National Corpus at the Faculty of Philosophy and the Department of Computer Science at the Faculty of Mathematics and Physics.

### CKL

As of 1 June 2000 the Center for Computational Linguistics (Centrum komputační lingvistiky; http://ckl.mff.cuni.cz) was established as one of the centers of excellence within the governmental program for support of research in the Czech Republic. The center is attached to the Faculty of Mathematics and Physics of Charles University in Prague.

#### Technical Reports

The ÚFAL/CKL technical report series has been established with the aim of disseminate topical results of research currently pursued by members, cooperators, or visitors of the Institute. The technical reports published in this Series are results of the research carried out in the research projects supported by the Grant Agency of the Czech Republic, GAČR 405/96/K214 ("Komplexní program"), GAČR 405/96/0198 (Treebank project), grant of the Ministry of Education of the Czech Republic VS 96151, and project of the Ministry of Education of the Czech Republic LN00A063 (Center for Computational Linguistics). Since November 1996, the following reports have been published.

**ÚFAL TR-1996-01** Eva Hajičová, *The Past and Present of Computational Linguistics at Charles University*
Jan Hajič and Barbora Hladká, *Probabilistic and Rule-Based Tagging of an Inflective Language – A Comparison*

**ÚFAL TR-1997-02** Vladislav Kuboň, Tomáš Holan and Martin Plátek, *A Grammar-Checker for Czech*

**ÚFAL TR-1997-03** Alla Bémová at al., Anotace na analytické rovině, *Návod pro anotátory (in Czech)*

**ÚFAL TR-1997-04** Jan Hajič and Barbora Hladká, *Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structural Tagset*

**ÚFAL TR-1998-05** Geert-Jan M. Kruijff, *Basic Dependency-Based Logical Grammar*

**ÚFAL TR-1999-06** Vladislav Kuboň, *A Robust Parser for Czech*

**ÚFAL TR-1999-07** Eva Hajičová, Jarmila Panevová and Petr Sgall, *Manuál pro tektogramatické značkování (in Czech)*

**ÚFAL TR-2000-08** Tomáš Holan, Vladislav Kuboň, Karel Oliva, Martin Plátek, *On Complexity of Word Order*

**ÚFAL/CKL TR-2000-09** Eva Hajičová, Jarmila Panevová and Petr Sgall, *A Manual for Tectogrammatical Tagging of the Prague Dependency Treebank*

**ÚFAL/CKL TR-2001-10** Zdeněk Žabokrtský, *Automatic Functor Assignment in the Prague Dependency Treebank*

**ÚFAL/CKL TR-2001-11** Markéta Straňáková, *Homonymie předložkových skupin v češtině a možnost jejich automatického zpracování*

**ÚFAL/CKL TR-2001-12** Eva Hajičová, Jarmila Panevová and Petr Sgall, *Manuál pro tektogramatické značkování (III. verze)*

**ÚFAL/CKL TR-2002-13** Pavel Pecina and Martin Holub, *Sémanticky signifikantní kolokace*

**ÚFAL/CKL TR-2002-14** Jiří Hana, Hana Hanová, *Manual for Morphological Annotation*

**ÚFAL/CKL TR-2002-15** Markéta Lopatková, Zdeněk Žabokrtský, Karolína Skwarská and Vendula Benešová, *Tektogramaticky anotovaný valenční slovník českých sloves*

**ÚFAL/CKL TR-2002-16** Radu Gramatovici and Martin Plátek, *D-trivial Dependency Grammars with Global Word-Order Restrictions*

**ÚFAL/CKL TR-2003-17** Pavel Květoň, *Language for Grammatical Rules*

**ÚFAL/CKL TR-2003-18** Markéta Lopatková, Zdeněk Žabokrtský, Karolina Skwarska, Václava Benešová, *Valency Lexicon of Czech Verbs VALLEX 1.0*

**ÚFAL/CKL TR-2003-19** Lucie Kučová, Veronika Kolářová, Zdeněk Žabokrtský, Petr Pajas, Oliver Čulo, *Anotování koreference v Pražském závislostním korpusu*

**ÚFAL/CKL TR-2003-20** Kateřina Veselá, Jiří Havelka, *Anotování aktuálního členění věty v Pražském závislostním korpusu*

**ÚFAL/CKL TR-2004-21** Silvie Cinková, *Manuál pro tektogramatickou anotaci angličtiny*

**ÚFAL/CKL TR-2004-22** Daniel Zeman, *Neprojektivity v Pražském závislostním korpusu (PDT)*

**ÚFAL/CKL TR-2004-23** Jan Hajič a kol., *Anotace na analytické rovině, návod pro anotátory*

**ÚFAL/CKL TR-2004-24** Jan Hajič, Zdeňka Urešová, Alevtina Bémová, Marie Kaplanová, *Anotace na tektogramatické rovině (úroveň 3)*

**ÚFAL/CKL TR-2004-25** Jan Hajič, Zdeňka Urešová, Alevtina Bémová, Marie Kaplanová, *The Prague Dependency Treebank, Annotation on tectogrammatical level*

**ÚFAL/CKL TR-2004-26** Martin Holub, Jiří Diviš, Jan Pávek, Pavel Pecina, Jiří Semecký, *Topics of Texts. Annotation, Automatic Searching and Indexing*

**ÚFAL/CKL TR-2005-27** Jiří Hana, Daniel Zeman, *Manual for Morphological Annotation (Revision for PDT 2.0)*

**ÚFAL/CKL TR-2005-28** Marie Mikulová a kol., *Pražský závislostní korpus (The Prague Dependency Treebank) Anotace na tektogramatické rovině (úroveň 3)*

**ÚFAL/CKL TR-2005-29** Petr Pajas, Jan Štěpánek, *A Generic XML-Based Format for Structured Linguistic Annotation and Its application to the Prague Dependency Treebank 2.0*

**ÚFAL/CKL TR-2006-30** Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Annotation on the tectogrammatical level in the Prague Dependency Treebank (Annotation manual)*

**ÚFAL/CKL TR-2006-31** Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Petr Sgall, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Anotace na tektogramatické rovině Pražského závislostního korpusu (Referenční příručka)*

**ÚFAL/CKL TR-2006-32** Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolařová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Petr Sgall,Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský, *Annotation on the tectogrammatical level in the Prague Dependency Treebank (Reference book)*

**ÚFAL/CKL TR-2006-33** Jan Hajič, Marie Mikulová, Martina Otradovcová, Petr Pajas, Petr Podveský, Zdeňka Urešová, *Pražský závislostní korpus mluvené češtiny. Rekonstrukce standardizovaného textu z mluvené řeči*

**ÚFAL/CKL TR-2006-34** Markéta Lopatková, Zdeněk Žabokrtský, Václava Benešová (in cooperation with Karolína Skwarska, Klára Hrstková, Michaela Nová, Eduard Bejček, Miroslav Tichý) *Valency Lexicon of Czech Verbs. VALLEX 2.0*

**ÚFAL/CKL TR-2006-35** Silvie Cinková, Jan Hajič, Marie Mikulová, Lucie Mladová, Anja Nedolužko, Petr Pajas, Jarmila Panevová, Jiří Semecký, Jana Šindlerová, Josef Toman, Zdeňka Urešová, Zdeněk Žabokrtský, *Annotation of English on the tectogrammatical level*

**ÚFAL/CKL TR-2007-36** Magda Ševčíková, Zdeněk Žabokrtský, Oldřich Krůza, *Zpracování pojmenovaných entit v českých textech*

**ÚFAL/CKL TR-2008-37** Silvie Cinková, Marie Mikulová, *Spontaneous speech reconstruction for the syntactic and semantic analysis of the NAP corpus*

**ÚFAL/CKL TR-2008-38** Marie Mikulová, *Rekonstrukce standardizovaného textu z mluvené řeči v Pražském závislostním korpusu mluvené češtiny. Manuál pro anotátory*

**ÚFAL/CKL TR-2008-39** Zdeněk Žabokrtský, Ondřej Bojar, *TectoMT, Developer's Guide*

**ÚFAL/CKL TR-2008-40** Lucie Mladová, *Diskurzní vztahy v češtině a jejich zachycení v Pražském závislostním korpusu 2.0*

**ÚFAL/CKL TR-2009-41** Marie Mikulová, *Pokyny k překladu určené překladatelům, revizorům a korektorům textů z Wall Street Journal pro projekt PCEDT*

**ÚFAL/CKL TR-2011-42** Loganathan Ramasamy, Zdeněk Žabokrtský, *Tamil Dependency Treebank (TamilTB) – 0.1 Annotation Manual*

**ÚFAL/CKL TR-2011-43** Nguy Giang Linh, Michal Novák, Anna Nedoluzhko, *Coreference Resolution in the Prague Dependency Treebank*

**ÚFAL/CKL TR-2011-44** Anna Nedoluzhko, Jiří Mírovský, *Annotating Extended Textual Coreference and Bridging Relations in the Prague Dependency Treebank*

**ÚFAL/CKL TR-2011-45** David Mareček, Zdeněk Žabokrtský, *Unsupervised Dependency Parsing*

**ÚFAL/CKL TR-2011-46** Martin Majliš, Zdeněk Žabokrtský, *W2C – Large Multilingual Corpus*

**ÚFAL TR-2012-47** Lucie Poláková, Pavlína Jínová, Šárka Zikánová, Zuzanna Bedřichová, Jiří Mírovský, Magdaléna Rysová, Jana Zdeňková, Veronika Pavlíková, Eva Hajičová, *Manual for annotation of discourse relations in the Prague Dependency Treebank*

**ÚFAL TR-2012-48** Nathan Green, Zdeněk Žabokrtský, *Ensemble Parsing and its Effect on Machine Translation*