Indonesian Dependency Treebank: Annotation and Parsing

Nathan Green[1], Septina Dian Larasati[1,2] and Zdeněk Žabokrtský[1]

[1]Charles University in Prague
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Prague, Czech Republic

[2]SIA Tilde
Vienibas gatve 75a
LV-1004
Riga, Latvia

{green,larasati,zabokrtsky}@ufal.mff.cuni.cz

Abstract

We introduce and describe ongoing work in our Indonesian dependency treebank. We described characteristics of the source data as well as describe our annotation guidelines for creating the dependency structures. Reported within are the results from the start of the Indonesian dependency treebank.

We also show ensemble dependency parsing and self training approaches applicable to under-resourced languages using our manually annotated dependency structures. We show that for an under-resourced language, the use of tuning data for a meta classifier is more effective than using it as additional training data for individual parsers. This meta-classifier creates an ensemble dependency parser and increases the dependency accuracy by 4.92% on average and 1.99% over the best individual models on average. As the data sizes grow for the the under-resourced language a meta classifier can easily adapt. To the best of our knowledge this is the first full implementation of a dependency parser for Indonesian. Using self-training in combination with our Ensemble SVM Parser we show aditional improvement. Using this parsing model we plan on expanding the size of the corpus by using a semi-supervised approach by applying the parser and correcting the errors, reducing the amount of annotation time needed.

1 Introduction

Treebanks have been a major source for the advancement of many tools in the NLP pipeline from sentence alignment to dependency parsers to an end product, which is often machine translation. While useful for machine learning as well and linguistic analysis, these treebanks typically only exist for a handful of resource-rich languages. Treebanks tend to come in two linguistic forms, dependency based and constituency based each with their own pros and cons. Dependency treebanks have been made popular by treebanks such as the Prague dependency treebank (Hajic, 1998) and constituency treebanks by the Penn treebank (Marcus et al., 1993). While some linguistic phenomena are better represented in one form instead of another, the two forms are generally able to be transformed into one another.

While many of the world's 6,000+ languages could be considered under-resourced due to a limited number of native speakers and low overall population in their countries, Indonesia is the fourth most populous country in the world with over 23 million native and 215 million non-native Bahasa Indonesia speakers. The development of language resources, treebanks in particular, for Bahasa Indonesia will have an immediate effect for Indonesian NLP.

Further development of our Indonesian dependency treebank can affect part of speech taggers, named entity recognizers, and machine translation systems. All of these systems have technical benefits to the 238 million native and non-native Indonesian speakers ranging for spell checkers, improved information retrieval, to improved access to more of the Web due to better page translation.

Some other NLP resources exist for Bahasa Indonesia as described in Section 2. While these are a nice start to language resources for Indonesian, dependency relations can have a positive effect on

26th Pacific Asia Conference on Language,Information and Computation pages 137–145

word reordering, long range dependencies, as well as anaphora resolution. Dependency relations have also been shown to be integral to deep syntactic transfer machine translation systems (Žabokrtský et al., 2008).

2 Related Work

There was research done on developing a rule-base Indonesian constituency parser applying syntactic structure to Indonesian sentences. It uses a rule-based approach by defining the grammar using PC-PATR (Joice, 2002). There was also research that applied the above constituency parser to create a probabilistic parser (Gusmita and Manurung, 2008). To the best of our knowledge no dependency parser has been created and publicly released for Indonesian.

Semi-supervised annotation has been shown to be a useful means to to increase the amount of annotated data in dependency parsing (Koo et al., 2008), however typically for languages which already have plentiful annotated data such as Czech and English. Self-training was also shown to be useful in constituent parsing as means of seeing known tokens in new context (McClosky et al., 2008). Our work differs in the fact that we examine the use of ensemble collaborative models' effect on the self-training loop as well as starting with a very reduced training set of 100 sentences. The use of model agreement features for our SVM classifier is useful in its approach since under-resourced languages will not need any additional analysis tools to create the classifier.

Ensemble learning (Dietterich, 2000) 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. (Surdeanu and Manning, 2010; Haffari et al., 2011) showed a successful combination of parse trees through a linear combination of trees with various weighting formulations. Parser combination with dependency trees have been examined in terms of accuracy (Sagae and Lavie, 2006; Sagae and Tsujii, 2007; Zeman and Žabokrtský, 2005). POS tags were used in parser combination in (Hall et al., 2007) for combining a set of Malt Parser models with an SVM classifier with success, however we believe our work is novel in its use an SVM classifier solely on model agreements.

3 Data Description

The 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 (Larasati, 2012). The raw version of the sentences originally were taken from the BPPT articles in economy from the PAN localization (PAN, 2010) project output. The treebank used Parts-Of-Speech tags (POS tags) provided by MorphInd (Larasati et al., 2011). Since the MorphInd output is ambiguous, the tags are also disambiguated and corrected manually, including the unknown POS tag. The distribution of the POS tags can be seen in Table 1.

The annotation is done using the visual tree editor, TreD (Pajas, 2000) and stored in CoNLL format (Buchholz and Marsi, 2006) for compatibility with several dependency parsers and other NLP tools.

4 Annotation Description

Currently the annotation provided in this treebank is the unlabeled relationship between the head and its dependents. We follow a general annotation guidelines as follows:

- The main head node of the sentence is attached to the *ROOT* node.
- Similarly as the main head node, the sentence separator punctuation is also attached to the *ROOT* node.
- The Subordinate Conjunction (with POS tag 'S–') nodes are attached to its subordinating clause head nodes. The subordinating clause head nodes are attached to its main clause head nodes.
- The Coordination Conjunctions (with POS tag 'H–') nodes, that connect between two phrases (using the conjunction or commas), are attached to the first phrase head node. The second phrase head nodes are attached to the conjunction node. It follows this manner when there are more than two phrases.

- The Coordination Conjunctions (with POS tag 'H–') nodes, that connect between two clauses (using the conjunction or commas), are attached to the first clause head node. The second clause head nodes are attached to the conjunction node. It follows this manner when there are more than two clauses.

- The prepositions nodes with the POS tag 'R–' are the head of Prepositional Phrases (PP).

- In Quantitative Numeral Phrases such as "3 thousand", 'thousand' node will be the head and '3' node attached to 'thousand' node.

In general, the trees have the verb of the main clause as the head of the sentence where the Subject and the Object are attached to it. In most cases, the most left noun tokens are the noun phrase head, since most of Indonesian noun phrases are constructed in Head-Modifier construction.

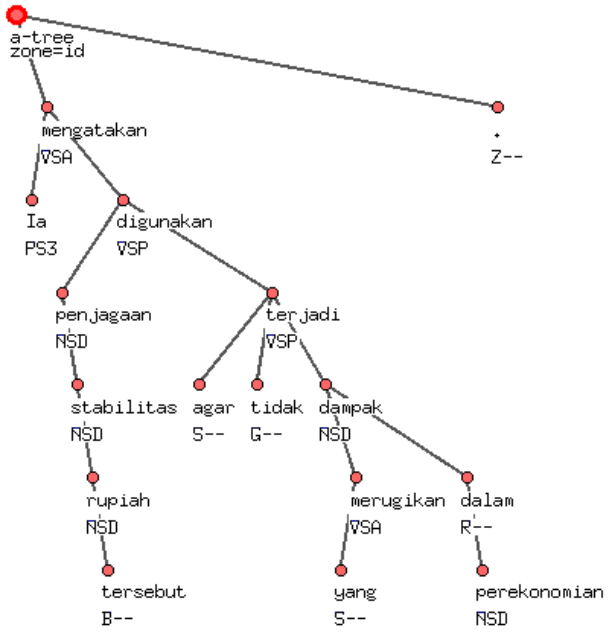

Figure 1: Dependency tree example for the sentence "He said that the rupiah stability protection is used so that there is no bad effect in economy."

POS tag	Description	Freq
NSD	Noun Singular	1037
Z–	Punctuation	278
VSA	Verb Singular Active	248
CC-	Cardinal Number	226
R–	Preposition	205
D–	Adverb	147
ASP	Adjective Singular Positive	127
S–	Subordinating Conjunction	104
VSP	Verb Singular Passiver	91
H–	Coordinating Conjunction	62
F–	Foreign Word	60
B–	Determiner	43
CO-	Ordinal Number	19
G–	Negation	17
PS3	Pronoun Singular 3rdPerson	12
W–	Question	7
O–	Copula	6
PP1	Pronoun Plural 1stPerson	6
ASS	Adjective Singular Superlative	4
PS1	Pronoun Singular 1stPerson	2
APP	Adjective Plural Positive	1
CD-	Colective Number	1
VPA	Verb Plural Active	1
VPP	Verb Plural Passive	1

Table 1: The distribution of the Part-Of-Speech tag occurrence.

5 Ensemble SVM Dependency Parsing

5.1 Methodology

5.1.1 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. For this reason our experiments are conducted over 18 training/tuning/testing data split configurations which enumerates possible configurations for testing sizes of 5%,10%,20% and 30%. For each configuration we randomly sample without replacement the training/tuning/testing data and rerun the experiment 100 times, each time sampling new sets for training,tuning, and testing. These 1800 runs, each on different samples, allow us to better show the overall effect on the accuracy metric as

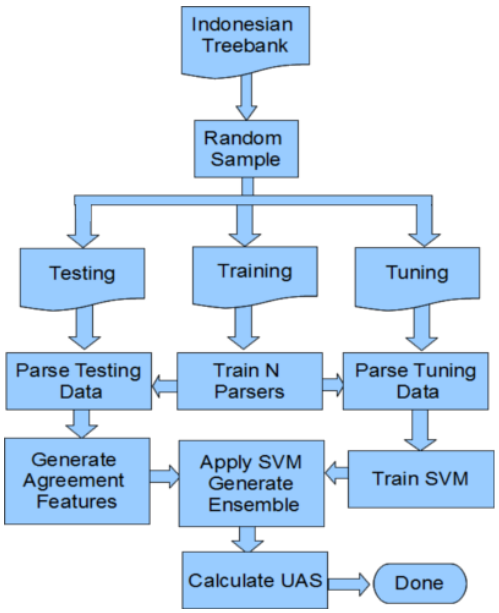

Figure 2: Process Flow for one run of our SVM Ensemble system. This Process in its entirety was run 100 times for each of the 18 data set splits.

well as the statistically significant changes as described in Section 5.1.5. Figure 2 shows this process flow for one run of this experiment.

5.1.2 Parsers

Dependency parsing systems are often optimized for English or other major languages. This optimization, along with morphological complexities, leads other languages toward lower accuracy scores in many cases. The goal here is to show that while the corpus is not the same in size as most CoNLL data, a successful dependency parser can still be trained from the annotated data and provide semi-supervised annotation to help increase the corpus size.

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 is the use of greedy algorithms which have a linear time complexity. However, due to the greedy algorithms, longer arc parses can cause error propagation across each transition (Kübler et al., 2009). We make use of Malt Parser (Nivre et al., 2007), which in the CoNLL shared tasks was often tied with the best performing systems.

For the experiments in this paper we only use Malt Parser, but we use different training parameters to create various parsing models. For Malt Parser we use a total of 7 model variations as shown in Table 2.

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: 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

5.1.3 Ensemble SVM System

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 have cycles. Using a Perl implementation of minimum spanning tree, 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 3 gives a graphical

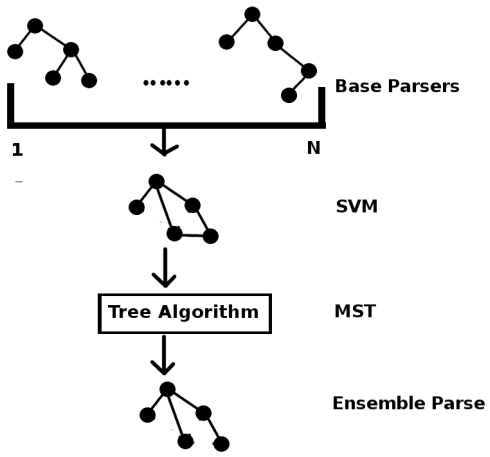

Figure 3: General flow to create an Ensemble parse tree

representation of how the SVM decision and MST algorithm create a final Ensemble parse tree which is similar to the construction used in (Hall et al., 2007; Green and Žabokrtský, 2012). Future iterations of this process could use a multi-label SVM or weighted edges based on the parser's accuracy on tuning data.

5.1.4 Data Set Split Configurations

Since this is a relatively small treebank and in order to confirm that our experiments are not heavily reliant on one particular sample of data we try a variety of data splits. To test the effects of the training, tuning, and testing data we try 18 different data split configurations, each one being sampled 100 times. The data splits in Section 5.2 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.

5.1.5 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 (Buchholz and Marsi, 2006). Since we are mainly concerned with the structure of the ensemble parse, we report only UAS scores in this paper.

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 $p < 0.01$.

5.2 Results and Discussion

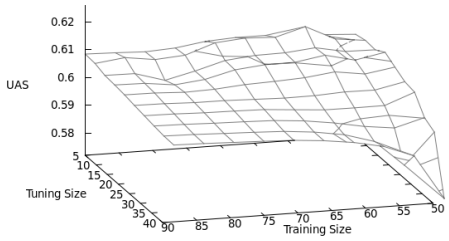

Figure 4: Surface plot of the UAS score for the tuning and training data split.

For each of the data splits, Table 3 shows the percent increase in our SVM system over both the average of the 7 individual models and over the best individual model. As the Table 3 shows, we obtain above average UAS scores in every data split. The increase is statistical significant in all data splits except 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 4. 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.

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 3: Average increases and decreases in UAS score for different Training-Tuning-Test samples. 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

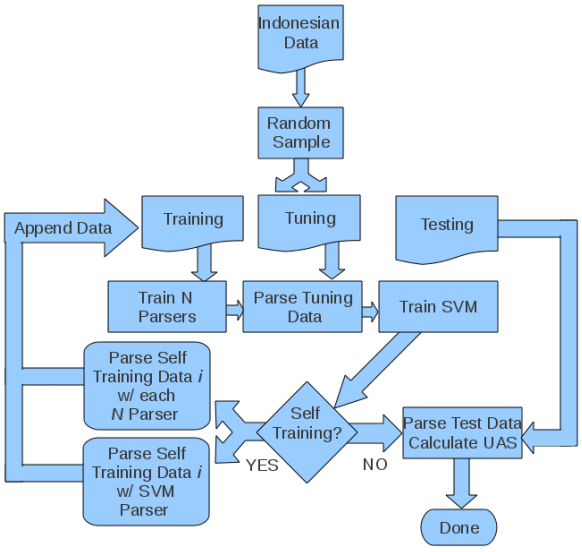

Figure 5: Process Flow for one run of our self-training system. There is one alternative scenario in which the system either does self-training with each N parser or with the ensemble SVM parser. These constitute two different experiments. For all experiments i=10 and N=7

6 Self-training

6.1 Methodology

The following methodology was run 12 independent times. Each time new testing/tuning/and training datasets were randomly selected without replacement. In each iteration the SVM classifier and dependency models were retrained using self-training. Also for each of the 12 experiments, new random self-training datasets were selected from the larger corpus. The results in the next section are averaged amongst these 12 independent runs. Figure 5 shows this process flow for one run of this experiment.

The data for self-training is also taken from IDENTIC and it consists of 45,000 sentences. The data does not have any dependency relation information but it is enriched with POS tags. It is processed with the same morphology tools as the training data described in section 3 but without the manual disambiguation and correction. This data and its annotation information are available on the IDENTIC homepage[1].

For self-training we present two scenarios. First, all parsing models are retrained with their own predicted output. Second, all parsing models are retrained with the output of our SVM ensemble parser. Self-training in both cases is done of 10 iterations of 20 sentences. Sentences are chosen at random from unannotated data. This allows us to examine self-training to a training data size of twice the original set.

The next section examines the differences between these two approaches and the effect on the overall parse.

6.2 Results of Self-training

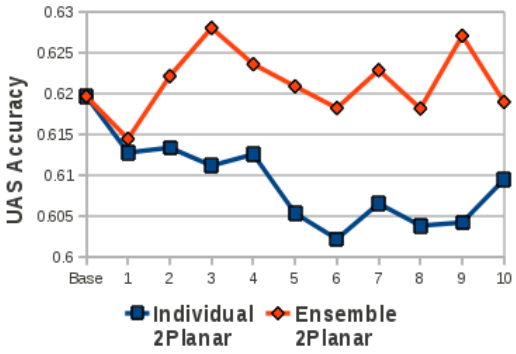

Figure 6: We can see that the self-trained Malt Parser 2Planar model that is trained with the ensemble output consistently outperforms the self-trained model that uses its own output. Results are graphed over the 10 self-training iterations

As can be seen in Figure 6, the base models did better when trained with additional data that was parsed by our SVM ensemble system. The higher UAS accuracy seems to of had a better effect then receiving dependency structures of a similar nature to the current model. We show the 2Planar model in Figure 6 but this was the case for each of the 7 individual models. On an interesting note, the SVM system had least improvement, 0.60%, when the component base models were trained on its own output. This seems warranted as other parser combination papers have shown that ensemble systems prefer models which differ more so that a clearer decision can be made (Hall et al., 2007; Green and Žabokrtský, 2012). The improvements when self-training on our SVM output over the individual parsers' output can be seen in Table 3. Again these are averages over 12 runs of the system, each run containing 10 self-training loops of 20 additional

[1] http://ufal.mff.cuni.cz/ larasati/identic/

sentences.

Model	% Improvement %
2planar	1.10%
nivreeager	0.40%
nivrestandard	1.62%
planar	0.87%
stackeager	2.28%
stacklazy	2.20%
stackproj	1.95%
svm	0.60%

Table 4: The % Improvement of all our parsing models including our ensemble svm algorithm over 12 complete iterations of the experiment.

7 Conclusion

We have shown a successful implementation of self-training for dependency parsing on an under-resourced language. Self-training in order to improve our parsing accuracy can be used to help semi-supervised annotation of additional data. We show this for an initial data set of 100 sentences and an additional self-trained data set of 200 sentences.

We introduce and show a collaborative SVM classifier that creates an ensemble parse tree from the predicted annotations and improves individual accuracy on average of 4.92%. This additional accuracy can release some of the burden on annotators for under-resourced language annotation who would use a dependency parser as a pre-annotation tool. Using these semi-supervised annotation techniques should be applicable to many languages since the SVM classifier is essentially blind to the language and only considers the models' agreement.

The treebank is the first of its kind for the Indonesian language. Additionally all sentences and annotations are being made available publicly online. We have described the beginnings of the Indonesian dependency treebank. Characteristics of the sentences and dependency structure have been described.

8 Acknowledgments

The research leading to these results has received funding from the European Commission's 7th Framework Program under grant agreement n° 238405 (CLARA), by the grant LC536 Centrum Komputační Lingvistiky of the Czech Ministry of Education, and this work uses language resources developed and/or stored and/or distributed by the LINDAT-Clarin project of the Ministry of Education of the Czech Republic (project LM2010013).

References

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK. Springer-Verlag.

Nathan Green and Zdeněk Žabokrtský. 2012. 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*, pages 19–26, Avignon, France, April. Association for Computational Linguistics.

R.H. Gusmita and R. Manurung. 2008. Some initial experiments with indonesian probabilistic parsing. In *Proceedings of the 2nd International MALINDO Workshop*.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. 2011. 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*, pages 710–714, Portland, Oregon, USA, June. Association for Computational Linguistics.

Jan Hajic. 1998. Building a syntactically annotated corpus: The prague dependency treebank. *Issues of valency and meaning*, pages 106–132.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939.

Joice. 2002. Pengembangan lanjut pengurai struktur kalimat bahasa indonesia yang menggunakan constraint-based formalism. undergraduate thesis. Master's thesis, Faculty of Computer Science, University of Indonesia.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*. Synthesis lectures on human language technologies. Morgan & Claypool, US.

Septina Dian Larasati, Vladislav Kuboň, and Dan Zeman. 2011. Indonesian morphology tool (morphind): Towards an indonesian corpus. *Systems and Frameworks for Computational Morphology*, pages 119–129.

Septina Dian Larasati. 2012. Identic corpus:morphologically enriched indonesian-english parallel corpus.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the Penn Treebank. *Comput. Linguist.*, 19:313–330, June.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, Manchester, UK, August. Coling 2008 Organizing Committee.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Petr Pajas. 2000. Tree editor tred, prague dependency treebank, charles university, prague. *See URL http://ufal. mff. cuni. cz/˜ pajas/tred*.

Localization Project PAN. 2010. Pan localization project.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

Mihai Surdeanu and Christopher D. Manning. 2010. 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*, HLT '10, pages 649–652, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. 2008. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL*, pages 167–170.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *In: Proceedings of the 9th International Workshop on Parsing Technologies*.