Ensemble Technique Utilization for Indonesian Dependency Parser

Arief Rahman
Institut Teknologi Bandung
Indonesia
23516008@std.stei.itb.ac.id

Ayu Purwarianti
Institut Teknologi Bandung
Indonesia
ayu@stei.itb.ac.id

Abstract

Two of the main problems in creating an Indonesian parser with high accuracy are the lack of sentence diversity in treebank used for training and suboptimal uses of parsing techniques. To resolve these problems, we build an Indonesian dependency treebank of 2098 sentences (simple and complex sentences) and use ensemble techniques to maximize the usage of available dependency parsers. We compare the combination of seven parsing algorithms provided by MaltParser and MSTParser, which provides both transition-based and graph-based models. From our experiments, we found that the graph-based model performs better than the transition-based model for Indonesian sentences. We also found no significant accuracy difference in models between several simple ensemble models and reparsing algorithms.

1 Introduction

Text parsing is one of the major tasks in natural language text processing (NLP). Text parsing is the process of determining the syntactic structure of a sentence. The result of text parsing is a syntactical tree, which is mostly used for higher-level NLP tasks, like sentiment analysis (Di Caro and Grella, 2013) and semantic role labeling (Johansson and Nugues. 2008).

There are two kinds of text parsing to date: constituent parsing and dependency parsing. Constituent parsing parses a sentence by determining the constituent phrases of the sentence hierarchically, usually by using a grammar (Aho, 2003). Dependency parsing, on the other hand, parses a sentence by determining a dependency relation for each word in a sentence. In this research, we use dependency parsing, because it is suited for analyzing languages with free word order, such as Indonesian (Nivre, 2007). Figure 1 shows an example of a parsed Indonesian sentence using dependency structure.

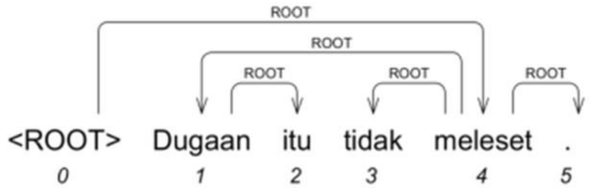

Figure 1. Example of a parsed Indonesian sentence (TL: *That allegation does not miss*) with dependency structure

Up until now, there have been only a few studies regarding Indonesian dependency parsing (Sulaeman, 2012; Green et.al, 2012). Most of the previous researches focused on rule-based parsing (Purwarianti et.al, 2013), which yielded quite a low accuracy, compared to other languages. Based on these researches, we use ensemble parsing techniques (Surdeanu and Manning, 2010) in our works. We also built a dependency Treebank corpus used for the model training with 2098 sentences.

In the following sections, we describe the relevant studies and some basic concepts about dependency parsing and its models. We then describe the corpus used in this research, our experiment settings, and finally the results and analysis.

2 Related Works

There are two studies that are related to ensemble dependency parsing, which is Surdeanu & Manning's work for English (Attardi and Dell'Orletta, 2009), and Green et al.'s work for Indonesian (Green et.al, 2012). Surdeanu & Manning created an ensemble dependency parser using parsing algorithms from both MaltParser and MSTParser for English. This research used CoNLL 2008 shared task corpus as the treebank for training and testing. There are two types of ensemble models used in this research: ensemble model at learning (using stacking) and ensemble model at runtime (using voting mechanism). The ensemble system at runtime used both weighted and unweighted voting scheme. The system also used a reparsing algorithm (Attardi and Dell'Orletta, 2009) to ensure the resulting dependency graphs always form a tree. The employed reparsing algorithms are Eisner's algorithm (Eisner, 1996) and Attardi's algorithm (Attardi and Dell'Orletta, 2009).

There are three conclusions that can be inferred from this research. First, an ensemble model that combines several base parsers at runtime performs significantly better than an ensemble model that combines two parsers at learning time. Second, well-formed dependency trees can be guaranteed without significant performance loss by linear-time approximate reparsing algorithms. Lastly, unweighted voting performs as well as weighted voting for the re-parsing of candidate dependencies.

Green et al.'s (2012) research consists of making treebank for Indonesian and analyzing ensemble technique effectivity on Indonesian dependency parser using self-training. This research used four out of five parsing algorithms provided by MaltParser (Nivre, Stack, Planar, and 2-Planar) as its base parsers. This research used 100 Indonesian sentences from IDENTIC (Larasati, 2012) as the treebank. The treebank was split into three parts: one for training, one for self-training tuning, and one for testing. The ensemble techniques used was Chu-Liu Edmonds reparsing algorithm with the unweighted voting scheme.

From this research, Green et al. (2012) concluded that self-training and ensemble parsing can be used to increase overall accuracy for Indonesian dependency parsing. Our work differs from Green et al.'s work by using base parsers from two different parsing models (transition-based and graph-based model), where Green et al.'s and only use one parsing model (transition-based model); and also the treebank size which is 20 times larger than Green et al.'s. Our experiment scheme is also different since we conducted a cross validation scheme in calculating the accuracy.

3 MaltParser and MSTParser

Both MaltParser and MSTParser are data-driven dependency parsers, which use treebank as training data for making parsing models. Both of these parsers are language-independent, which allows any language to be used in the parser without any compromise in accuracy. However, these parsers have different ways to parse sentences. Both of these parsers will be explained in the next sections.

3.1 MaltParser

MaltParser was introduced by Nivre et al. (2007). It is a data-driven and language-independent dependency parser. MaltParser uses transition-based model during parsing. This model uses transition machine, which contains four main components: a set of parsing states, a set of parsing transitions, the initial parsing state, and a set of terminating parsing states. The parsing result of a transition-based model is a transition sequence that can be used to transform the initial parsing state into a terminating parsing state. The learning problem comes from determining the best action to make at each state. This can be achieved learning an “oracle” function.

There are five parsing algorithms available in MaltParser, which can be seen in Table 1. Each of these algorithms differs on the data structures used to represent the parsing states and the set of transitions available for every parsing state.

Algorithm	Parsing Mode	Data Structure	Complexity	Projective?
Nivre	Arc-eager	Stack	O(n)	Yes
	Arc-standard	Stack	O(n)	Yes
Covington	Projective	Two lists	$O(n^2)$	Yes
	Non-projective	Two lists	$O(n^2)$	No
Stack	Projective	Stack	O(n)	Yes
	Non-projective lazy	Stack	O(n)	No
	Non-projective eager	Stack	O(n)	No
Planar		Stack	O(n)	Yes
2-Planar		Two stacks	O(n)	Yes

Table 1. Transition-based Algorithm Used by MaltParser

3.2 MSTParser

MSTParser is a data-driven and language-independent dependency parser that uses graph-based model. The graph-based model adds a weight to each directed edge in a dependency graph, which is determined by the dot product of the feature weight vector and the score vector based on the current dependency relation. The overall graph is scored, which equals to the product of all weights of all directed edges. The graph-based model will be able to determine the best dependency tree for a sentence by finding the spanning tree of the dependency graph created with maximum score.

There are two parsing algorithms available in MSTParser: Eisner and Chu-Liu Edmonds algorithm. The first one is Eisner algorithm, which uses dynamic programming (memoization) to find the maximum spanning trees. It has a complexity of $O(n^3)$ and can only build projective trees. The second one is Chu-Liu Edmonds algorithm, which uses recursive greedy selection to find the maximum spanning tree. It has a complexity of $O(n^2)$ and can build both projective and non-projective trees.

4 Ensemble Technique

In NLP, ensemble technique is a parsing technique that uses a collaboration of several unique parsing models to parse sentences better than individually. Ensemble technique can be applied during learning and during parsing. Ensemble technique can be applied during learning by having a parsing model parse a test data, and then uses another parsing model to repair the mistakes made by the previous parser. These steps are repeated until all parsers are used. Several examples of ensemble during learning are stacked parsing and guided model (Fan et.al, 2008; Nivre and McDonald, 2008).

Ensemble technique can also be applied during training by having several base parsers parse the same test data. The base parsers are trained using the same training data. After that, the result from each base parser will be used to determine one final dependency graph that considers all of the base parsers' results. There are three kinds of ensemble during parsing to date: meta-classifier, voting system, and reparsing algorithm. We will only discuss the voting system and the reparsing algorithm in this paper.

In voting system, every token in a sentence will have a dependency relation that was determined by majority voting. Every dependency relation from all of the base parsers will be tallied according to a voting scheme (weighted or unweighted). After that, the best dependency relation for each token will be used for the final dependency graph. In practice, voting scheme is simpler than meta-classifier and performs at the same level as meta-classifier.

There two types of voting that can be used for voting system: weighted and unweighted. Unweighted voting makes all base parsers give the same score for all dependency relations. On the other hand, weighted voting makes base parsers with better accuracy give bigger score for particular dependency relations. When using voting system, the dependency relation with the biggest score for a particular token will be used by the ensemble parser to create the final dependency graph. Voting is done until every token has a dependency relation.

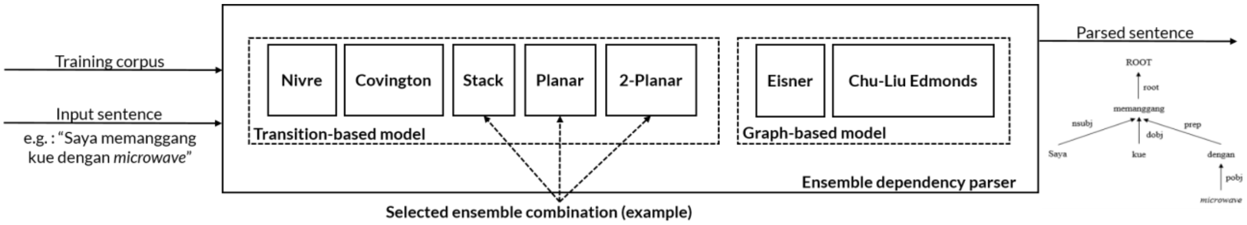

Figure 2. Overall ensemble parsing process

Sometimes, the dependency graphs that are created by the voting system does not make a dependency tree. To resolve this, a reparsing algorithm can be used to parse the dependency graph by finding the maximum spanning tree of the graph. The weight of each directed edge is calculated by tallying the dependency relations from all of the base parsers using a weighting scheme (weighted or unweighted). Three of the most used reparsing algorithms are Eisner algorithm, Chu-Liu Edmonds algorithm, and Attardi algorithm. Our work uses voting system with unweighted voting scheme and all of the reparsing algorithms (all with unweighted weighting scheme).

There are three main steps on doing ensemble parsing. The first step is training all of the base parsers with parsing algorithms and learning algorithm provided by MaltParser and MSTParser. The base parsers are trained using the treebanks that will be listed in the next section. The second step is parsing the test sentences using a particular base parsers combination. The parsing result is in CoNLL. The last step is using a particular ensemble technique to create an ensemble tree. The whole process of ensemble parsing can be seen in Figure 2.

5 Experiments

5.1 Experimental Settings

Our treebank statistic is shown in Table 2. We performed the experiments using our treebank that contains 2098 sentences. We used Kuncoro's treebank (2013), which contains 2018 sentences, and added 80 sentences, which we manually parsed from news sites like Kompas and Tempo to include in our treebank.

There are three main scenarios in our research. In the first scenario, we compared the performances of the base parsers in parsing Indonesian sentences. There were eleven single parsers that were compared: Nivre eager, Nivre standard, Covington projective, Covington non-projective, Stack projective, Stack eager, Stack lazy, Planar, 2-Planar, Eisner, and Chu-Liu Edmonds. The parsers were tested using 10-fold cross validation and used the same learning algorithm (SVM).

In the second scenario, we compared the performances of four ensemble techniques: voting system with unweighted scheme, Eisner reparsing algorithm, Chu-Liu Edmonds reparsing algorithm, and Attardi reparsing algorithm. All of the reparsing algorithms used unweighted weighting scheme. The ensemble combination used is 2-Planar, Eisner, and Chu-Liu Edmonds parsing algorithms. The parsers were tested using 10-fold cross validation and used the same learning algorithm (SVM).

In the third scenario, we compared the performances of ensemble parsers that use different algorithm combination. There were six ensemble combinations that were compared: all parsing algorithms (both from MaltParser and MSTParser), all algorithms from MaltParser, all algorithms from MSTParser, all projective parsing algorithms, all non-projective algorithms, and three algorithms with the highest accuracy (according to the first scenario). The parsers used Eisner reparsing algorithm with unweighted weighting scheme and were tested using 10-fold cross validation and used the same learning algorithm (SVM).

5.2 Results and Analysis

The results of the four experiments are shown in Table 3, Table 4, and Table 5. The metric used in this work is UAS (unlabeled attachment score). We don't use LAS (labeled attachment score) since we have no dependency label in our treebank yet.

Sentence Type		Number of Sentences (Percentage)
Number of clauses	Simple sentence	1067 (50.86%)
	Compound sentences	349 (16.63%)
	Complex sentence	527 (25.12%)
	Complex-compound sentence	155 (7.39%)
Presence of gerund	Present	50 (2.38%)
	Not present	2048 (97.62%)
POS tag of central dependency	Transitive verb	1017 (48.47%)
	Intransitive verb	989 (47.14%)
	Adjective	69 (3.29%)
	Noun	8 (0.38%)
	Others	15 (0.71%)
Deletion type	None	1630 (77.69%)
	Anaphoric	312 (14.87%)
	Cataphoric	89 (4.24%)
	Structural	67 (3.19%)

Table 2. Indonesian Treebank Statistic

The result from Table 4 shows that Chu-Liu Edmonds algorithm is the best parsing algorithm to be used for Indonesian sentences. One of the main factors that contribute to Chu-Liu Edmonds' high accuracy is the fact that graph-based model can handle long distance dependency well, which most Indonesian sentences have. We can see from the results that Chu-Liu Edmonds dominated both the accuracy on parsing the long sentences and the short sentences. Theoretically, transition-based models should have been able to parse short sentences better than graph-based model. However, the results showed the opposite. This could be caused by Indonesian sentences tendency to use long distance dependencies, even in short sentences.

Another interesting thing that can be inferred from these results is the fact that transition-based models generally performed better when parsing sentences with outlier predicates (like adjectives and nouns). This is most likely because of the rich feature representations that transition-based model has, which depends on the data structures used to represent the parsing state. Figure 3 and 4 shows the example of this occurrence.

The result from Table 5 shows that there is no significant accuracy difference on the ensemble technique used. However, voting system with unweighted scheme has a little higher accuracy than others (0.01%), because the resulting graphs are not reparsed, which make the individual dependency accuracy better than those that use reparsing algorithm. The accuracy indifference may be caused by the fact that all of the reparsing algorithms used unweighted voting scheme, which would make the weight of many dependency relations to be the same, regardless of the algorithm.

The result from Table 6 shows that the parser that uses the combination of the top three base parsers (2-Planar, Eisner, and Chu-Liu Edmonds) has the highest accuracy. This is because of the ensemble property itself. Most of the correct majority decisions (from the best parsers) were able to repair the best parser's mistakes. We can also see that parsers combining all algorithms have lower accuracy than others. This is because of the fact that most of the parsing algorithms created the same dependency trees, especially for the same variants (like Nivre's standard and eager mode). This resulted in most majority decisions to come from the algorithms with several variants.

Parsing Algorithm	**Accuracy**			
	Overall	**Outlier Predicates**	**Sentence with > 15 tokens**	**Sentence with ≤ 15 tokens**
Nivre-eager (Malt)	83.5%	**60.00%**	77.16%	85.81%
Nivre-standard (Malt)	82.9%	55.71%	75.51%	85.54%
Covington projective (Malt)	82.4%	51.43%	75.25%	85.01%
Covington non-projective (Malt)	82.6%	50.00%	75.40%	85.29%
Stack projective (Malt)	83.3%	55.71%	76.23%	85.81%
Stack eager (Malt)	83.7%	57.14%	77.58%	85.86%
Stack lazy (Malt)	83.9%	57.14%	78.17%	85.90%
Planar (Malt)	84.1%	57.14%	77.85%	86.30%
2-Planar (Malt)	84.7%	54.29%	78.79%	86.82%
Eisner (MST)	85.8%	54.29%	80.68%	87.51%
Chu-Liu-Edmonds (MST)	**86.1%**	52.86%	**80.89%**	**87.86%**

Table 3. Accuracy of Single Dependency Parsers

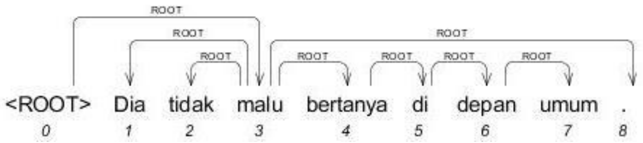

Figure 3. Correct dependency tree for sentence *Dia tidak malu bertanya di depan umum* (He is not ashamed of asking questions in public)

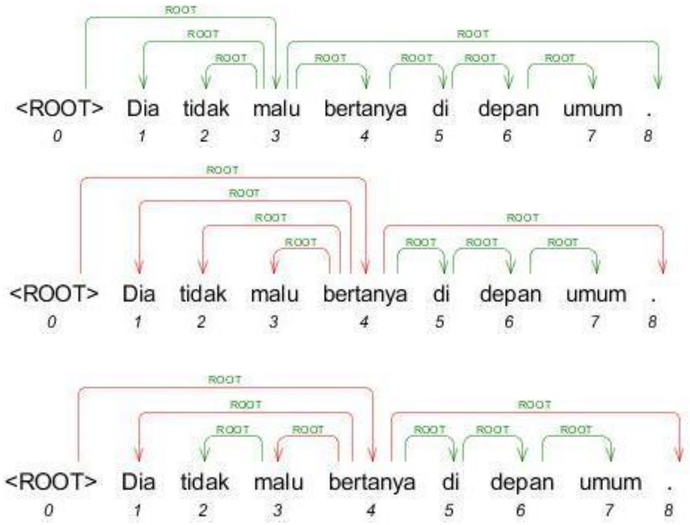

Figure 4. Parsing result for sentence *Dia tidak malu bertanya di depan umum* (He is not ashamed of asking questions in public) using 2-Planar, Eisner, and Chu-Liu Edmonds parsing algorithm respectively

Ensemble Technique	Accuracy
Unweighted majority	**86.6%**
Eisner	86.5%
Chu-Liu-Edmonds	86.5%
Attardi	86.5%

Table 4. Accuracy of Parsers with Different Ensemble Technique

Ensemble Technique	Accuracy
All parsing algorithms (MaltParser + MSTParser)	85.5%
All parsing algorithms from MaltParser	85.1%
All parsing algorithms from MSTParser	86.0%
All projective parsing algorithms	85.6%
All non-projective parsing algorithms	85.3%
Top three parsers (2-Planar, Eisner, and Chu-Liu Edmonds)	**86.5%**

Table 5. Accuracy of Parsers with Different Ensemble Combination

6 Problems While Creating Indonesian Treebank

During the making of our Indonesian Treebank, we encountered several problems that should be solved in the future works. Most of the problems revolve around labeling standards. The first problem is the POS-tags standards. Our current treebank uses proprietary standards for both the coarse-grained and fine-grained POS-tags. While our standards are adequate to cover most word types, the lack of standards for POS-tags makes it difficult to merge several treebanks to create a larger data set for future studies. INACL has issued a POS-tags standard for Indonesian[1], however, there is still a matter of mapping the old POS-tags standards to the new POS-tags standards.

The second problem is the lack of dependency labels for Indonesian. At the time this research is concluded, there were no dependency label standards that can be used to label each dependency relation in a treebank. This would drastically reduce the usefulness of the parser results for most semantic-related NLP tasks since the dependency label is one of the main features in those tasks. One possible solution is to use the dependency label standards from Universal Dependencies (Nivre et al., 2016), which has a universal dependency labeling scheme.

7 Conclusions and Future Works

From our experiments, we concluded that the graph-based model is better than transition-based models for the Indonesian language. We also concluded that different simple ensemble techniques and ensemble combinations do not give significant accuracy difference between models.

Potential future works lie in using more intricate ensemble techniques (e.g. weighting models by its proficiency in creating dependencies for different POS-tags) or better base parsers (using deep learning or word embedding as features during parsing). Other major future works lie in creating a big and complete dependency treebank, which can be done by merging several treebanks from several studies using one labeling standards for both its POS-tags and dependency labels.

References

Aho, A. V. (2003). Compilers: Principles, Techniques, and Tools (for Anna University), 2nd Edition, Pearson Addison Wesley.

[1] http://inacl.id/inacl/wp-content/uploads/2017/06/INACL-POS-Tagging-Convention-26-Mei.pdf

Attardi, G., and Dell'Orletta, F. (2009). Reverse Revision and Linear Tree Combination for Dependency Parsing. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, pp. 261-264.

Chang, C.-C., and Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology (TIST), Volume 2 Issue 3, April 2011, Article Number 27.

Di Caro, L., and Grella, M. (2013). Sentiment Analysis via Dependency Parsing. Computer Standards & Interfaces, Elsevier, Volume 35, Issue 5, September 2013, pp 442-453.

Eisner, J. M. (1996). Three New Probabilistic Models for Dependency Parsing: An Exploration. Proceedings of the 16th Conference on Computational Linguistics - Volume 1, pp. 340-345.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. The Journal of Machine Learning Research, pp. 1871-1874.

Green, N., Larasati, S. D., and Zabokrtsky, Z. (2012). Indonesian Dependency Treebank: Annotation and Parsing. 26th Pacific Asia Conference on Language, Information, and Computation, pp. 137-145.

Johansson, R., and Nugues, P. (2008). Dependency-Based Syntactic-Semantic Analysis with PropBank and NomBank. Proceedings of the Twelfth Conference on Computational Natural Language Learning, CoNLL '08, (pp. 183-187).

Jurafsky, D., and Martin, J. (2009). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson Prentice Hall.

Jurafsky, Adhiguna. (2013). *Pemanfaatan Pengurai Ensemble dan Teknik Self-Learning untuk Meningkatkan Akurasi Pengurai Bahasa Indonesia* [Ensemble Parsers and Self-Learning Technique Utilization to Increase Indonesian Parser Accuracy]. Institut Teknologi Bandung.

Larasati, S. D. (2012). IDENTIC Corpus: Morphologically Enriched Indonesian-English Parallel Corpus. LREC, pp. 902-906.

Nivre et al. (2007). MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. Natural Language Engineering. 13, pp. 95-135.

Nivre, J., & McDonald, R. T. (2008). Integrating Graph-Based and Transition-Based Dependency Parsers. Proceedings of ACL-08: HLT, pp. 950–958, Columbus, Ohio

Nivre, J., de Marneffe, M. C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., ... & Tsarfaty, R. (2016, May). Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC*.

Purwarianti, A., Saelan, A., Afif, I., Ferdian, F., Wicaksono, A.F. (2013). Natural Language Understanding Tools with Low Language Resource in Building Automatic Indonesian Mind Map Generator. International Journal on Electrical Engineering and Informatics, Vol 5, No. 3, September 2013.

Sulaeman, M. K., and Purwarianti, A. (2012). “Dependency Parsing for Indonesian with GULP”. Proceeding of ICEEI (International Conference of Electrical Engineering and Informatics) 2011. July 2011. Bandung, Indonesia

Surdeanu, M., and Manning, C. D. (2010). Ensemble Models for Dependency Parsing: Cheap and Good? Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 649-652.