Using an SVM Ensemble System for Improved Tamil Dependency Parsing

Nathan Green, Loganathan Ramasamy and Zdeněk Žabokrtský
Charles University in Prague
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Prague, Czech Republic
{green,ramasamy,zabokrtsky}@ufal.mff.cuni.cz

Abstract

Dependency parsing has been shown to improve NLP systems in certain languages and in many cases helps achieve state of the art results in NLP applications, in particular applications for free word order languages. Morphologically rich languages are often short on training data or require much higher amounts of training data due to the increased size of their lexicon. This paper examines a new approach for addressing morphologically rich languages with little training data to start.

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. 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.

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.

1 Introduction

Dependency parsing has made many advancements in recent years. A prime reason for the quick advancement has been the CoNLL shared task competitions, which gave the community a common training/testing framework along with many open source systems. These systems have, for certain languages, achieved high accuracy ranging from on average from approximately 60% to 80% (Buchholz and Marsi, 2006). The range of scores are more often language dependent rather than system dependent, as some languages contain more morphological complexities. While some of these languages are morphologically rich, we would like to additionally address dependency parsing methods that may help under-resourced languages as well, which often overlaps with morphologically rich languages. For this reason, we have chosen to do the experiments in this paper using the Tamil Treebank (Ramasamy and Žabokrtský, 2012).

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*.

When working with small datasets it is often very difficult to determine which dependency model will best represent your data. One can try to pick the model through empirical means on a tuning set but as the data grows in the future this model may no longer be the best choice. The change in the best model may be due to new vocabulary or through a domain shift. If the wrong single model is chosen early on when training is cheap, when the model is applied in semi supervised or self training it could lead to significantly reduced annotation accuracy.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 72–77,
Jeju, Republic of Korea, 12 July 2012.

For this reason, we believe ensemble combinations are an appropriate direction for lesser resourced languages, often a large portion of morphologically rich languages. Ensemble methods are robust as data sizes grow, since the classifier can easily be retrained with additional data and the ensemble model chooses the best model on an edge by edge basis. This cost is substantially less than retraining multiple dependency models.

2 Related Work

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; Søgaard and Rishøj, 2010). (Sagae and Lavie, 2006; Green and Žabokrtský, 2012) differ in part since their method guarantees a tree while our system can, in some situations, produce a forest. 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 of an SVM classifier solely on model agreements. Other methods of parse combinations have shown to be successful such as using one parser to generate features for another parser. This was shown in (Nivre and McDonald, 2008; Martins et al., 2008), in which Malt Parser was used as a feature to MST Parser.

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) (Ramasamy and Žabokrtský, 2012). Development of TamilTB is still in progress and the initial results for TamilTB appeared in (Ramasamy and Žabokrtský, 2011). Previous parsing experiments in Tamil 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 Malt Parser and MST parser.

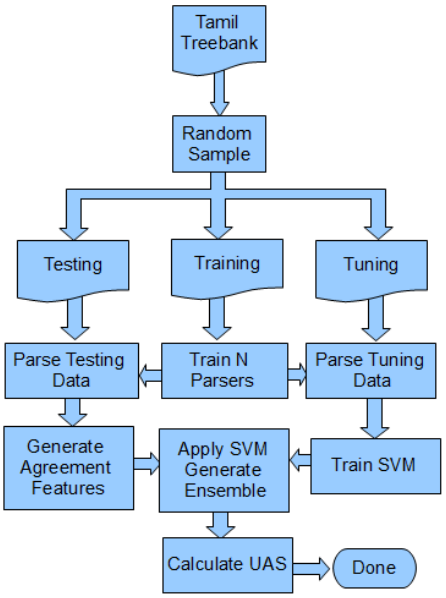

Figure 1: Process Flow for one run of our SVM Ensemble system. This Process in its entirety was run 100 times for each of the 8 data set splits.

When the morphological tags were available during both training and testing, the rule based approach performed better than Malt and MST parsers. For other Indian languages, treebank development is active mainly for Hindi and Telugu. Dependency parsing results for them are reported in (Husain et al., 2010).

3 Methodology

3.1 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 7 training/tuning/testing data split configurations. For each configuration we randomly sample without replacement the training/tuning/testing data and rerun the experiment 100 times. These 700 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 3.5. Figure 1 shows this process flow for one run of this experiment.

3.2 Parsers

A dependency tree is a special case of a dependency graph that spawns from an artificial root, is connected, follows a single-head constraint 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. The most common form of this type of dependency parsing is Graph-Based parsing also called arc-factored parsing and deals with the parameterization of the edge weights. The main drawback of these methods is that for projective trees, the worst case scenario for most methods is a complexity of $O(n^3)$ (Eisner, 1996). However, for non-projective parsing Chu-Liu-Edmond's algorithm has a complexity of $O(n^2)$ (McDonald et al., 2005). The most common tool for doing this is MST parser (McDonald et al., 2005). For this parser we generate two models, one projective and one non-projective to use in our ensemble system.

Transition-based parsing creates a dependency structure that is parameterized over the transitions. This is closely related to shift-reduce constituency parsing algorithms. The benefit of transition-based parsing is the 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 (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 this parser we generate 7 different models using different training parameters, seen in Table 1, and use them as input into our ensemble system along with the two Graph-based models described above. Each parser has access to gold POS information as supplied by the TamilTB described in 3.4.

Dependency parsing systems are often optimized for English or other major languages. This optimization, along with morphological complexities, lead 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 or scope of most CoNLL data, a successful dependency parser can still be trained from the annotated data through model combination for morphologically rich languages.

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 1: 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

3.3 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 novel and 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 subgraph is then connected and cycles are eliminated in order to achieve a well formed dependency structure. Figure 2 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 (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.

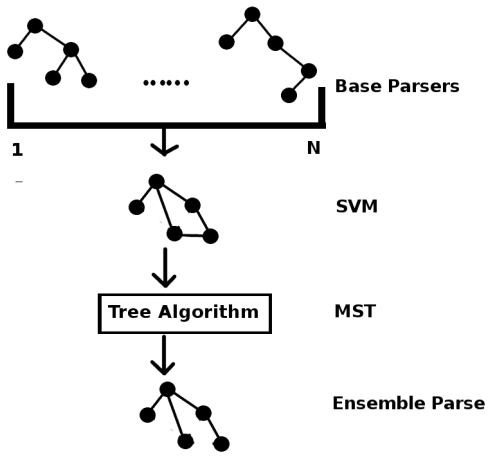

Figure 2: General flow to create an Ensemble parse tree

3.4 Data Sets

Table 2 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. Also, Table 2 illustrates that most of the word types can be uniquely identified with single morphological tag and 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: TamilTB: data statistics

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 7 different data splits. The tuning data in the Section 4 use the format training-tuning-testing. So 70-20-10 means we used 70% of the TamilTB for training, 20% for tuning the SVM classifier, and 10% for evaluation.

3.5 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 whether 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 we have 100 iterations each with different sampling. Each model is compared against the same samples so a paired test is appropriate in this case. We report statistical significance values for $p < 0.01$ and $p < 0.05$.

4 Results and Discussion

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

For each of the data splits, Table 3 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 3 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 Table 4, 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. It appears that

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

the size of the tuning and training data matter more than the size of the test data given the low variance in Table 5. Since the TamilTB is relatively small 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.

Data Splits	SVM Variance
70-20-10	0.0011
60-20-20	0.0005
60-30-10	0.0010
60-10-30	0.0003
85-5-10	0.0010
90-5-5	0.0028
80-10-10	0.0010

Table 5: Variance of the UAS Scores of our Ensemble SVM System over 100 data splits

The traditional approach of using as much data as possible for training does not seem to be as effective as partitioning more data for tuning an SVM. For instance the highest 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 of 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 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.

5 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 languages. 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.

In the future we would like to examine how our data splits' results change as more data is added. This might be a prime use for self training. Since the tuning data size for the SVM seems most important, the UAS may be improved by only adding self training data to our tuning sets. This would have the additional benefit of eliminating the need to retrain the individual parsers, thus saving computation time. The tuning size may have a reduced effect for larger treebanks but in our experiments it is critical to the smaller treebank. Additionally, a full comparison of various ensemble parsing error distributions will be needed.

6 Acknowledgments

This research has received funding from the European Commission's 7th Framework Program (FP7) under grant agreement n° 238405 (CLARA)

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.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

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.

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.

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.

Samar Husain, Prashanth Mannem, Bharat Ram Ambati, and Phani Gadde. 2010. The icon-2010 tools contest on indian language dependency parsing. In *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing*, pages 1–8.

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

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 157–166, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

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.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2011. 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*, CICLing'11, pages 82–95, Berlin, Heidelberg.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2012. Prague dependency style treebank for Tamil. In *Proceedings of LREC 2012*, İstanbul, Turkey.

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.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China, August.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *HLT: 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.

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*.