Using an SVM Ensemble System for improved Tamil Dependency Parsing

Abstract

Dependency parsing has gained interest of late due to its ability to help parse languages with a free word order. Dependency parsing has been shown to improve NLP systems in certain languages and in many cases is considered the state of the art in the field. The use of dependency parsing has mostly been limited to free word order languages. Morphologically rich languages often times are 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 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 5.44% and 0.52%, statistically significant percent improvement with 95% confidence, increase over the average dependency model and the best individual system respectively.

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. These competitions gave the community a common training/testing framework along with many open source systems. These systems have, for certain languages, achieved fairly high accuracy. 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 in Tamil using the Tamil Treebank (REF).

Morphologically Rich and Under-resourced Languages (Loganathan) something about why Tamil is Morphologically rich

Manual dependency annotation is very time consuming and costly. While these annotations exist for some of the larger treebanks, the cost of annotation is prohibitive for under-resourced languages. 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 empirically means on a tuning set but as the data grows in the future this model may no longer be the best choice. If such a model was used for semi supervised or self training this could lead to reduced annotation accuracy. For this reason, we believed ensemble combinations are an appropriate direction for lesser resourced languages, often a large portion of morphologically rich languages. Ensemble methods allow us the added robustness as data sizes grow since the SVM can easily be re-tuned with additional data and the ensemble model chooses the best model on an edge by edge basis. As one model improves this can be integrated into the system easily without
retraining.

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; Zeaman 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.

Tamil Dependency Treebanks and Parsing (Loganathan)

Parsing under-resourced languages

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 8 data size splits for training/tuning/testing and for each split we sample and rerun the experiment 100 times. This allows us to better show the overall effect on the accuracy metric as well as 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 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 non-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 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 (Kübler et al., 2009). We make use of

![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.](image)
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 and use them as input into our ensemble system along with the 2 Graph-based models described above.

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.

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.

To assign weights to our ensemble tree we use our classifier to predict which model will be correct by again creating the model agreement feature set on the predicted edge of the unknown test data. The predicted model will then get a uniform weight of one in the graph. At the end of all the tokens in a sentence the graph may not be connected. Using ChuLiu/Edmonds’ algorithm’s Perl implementation we obtain a minimum spanning forest, in which each subgraph is then connected in order to achieve a well formed dependency structure. Figure 2 gives a graphical representation of how the SVM weighting 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).

3.4 Data Sets

( Loganathan)

3.5 Evaluation

As an artifact of the CoNLL shared tasks competition, two standard metrics for comparing dependency parsing systems emerged. Labeled attachment score (LAS) and unlabeled attachment score (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 are mainly concerned with the structure of the ensemble parse, we report only UAS scores in this paper.

To test statistical significant 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

For each of the data splits Table 1 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 1 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 2, shows that the decrease in the 90-5-5 split is not statistically significance 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 6-20-20 data split.

<table>
<thead>
<tr>
<th>Data Split</th>
<th>Average SVM UAS</th>
<th>% Increase over Avg</th>
<th>% Increase over Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>70-20-10</td>
<td>76.50%</td>
<td>5.13%</td>
<td>0.52%</td>
</tr>
<tr>
<td>60-20-20</td>
<td>76.36%</td>
<td>5.68%</td>
<td>0.72%</td>
</tr>
<tr>
<td>60-30-10</td>
<td>75.42%</td>
<td>5.44%</td>
<td>0.52%</td>
</tr>
<tr>
<td>60-10-30</td>
<td>75.66%</td>
<td>4.83%</td>
<td>0.10%</td>
</tr>
<tr>
<td>85-5-10</td>
<td>75.33%</td>
<td>3.10%</td>
<td>-1.21%</td>
</tr>
<tr>
<td>90-5-5</td>
<td>75.42%</td>
<td>3.19%</td>
<td>-1.10%</td>
</tr>
<tr>
<td>80-10-10</td>
<td>76.44%</td>
<td>4.84%</td>
<td>0.48%</td>
</tr>
</tbody>
</table>

Table 1: 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.

It appears that size of the tuning and training data matter more than the size of the test data. Given that 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 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 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 score 76.25% and the SVM had a UAS of 75.42%. However, 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.

5 Conclusion

We have shown a new SVM based ensemble parse tree 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. Using this Ensemble model we show a 5.44% and 0.52%, statistically significance percent improvement with 95% confidence, increase over the average dependency model and the best individual system respectively.

In the future we would like to examine how our training/tuning/testing data splits’ results change as more data is added. This might be a prime use for self training. Seeing how the tuning data size for the SVM seems most important, the UAS scores may be improved by only adding self training data to our tuning sets. This would have the additional benefit of eliminating the need to retraining the individual parsers, thus saving computation time.

References


<table>
<thead>
<tr>
<th>Model</th>
<th>70-20-10</th>
<th>60-20-20</th>
<th>60-30-10</th>
<th>60-10-30</th>
<th>85-5-10</th>
<th>90-5-5</th>
<th>80-10-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2planar</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>mstnonproj</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>mstproj</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>nivereager</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>x</td>
</tr>
<tr>
<td>nivrestandard</td>
<td>*</td>
<td>*</td>
<td>**</td>
<td>x</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>planar</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>stackeager</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>stacklazy</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>stackproj</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>x</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
</tbody>
</table>

Table 2: 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 ≥ 0.05


