Self-training a Constituency Parser using N-gram Trees

Arda Çelebi and Arzucan Özgür

Department of Computer Engineering
Boğaziçi University
Bebek, 34342 Istanbul, Turkey
{ arda.celebi arzucan.ozgur } @boun.edu.tr

Abstract

In this study, we tackle the problem of self-training a feature-rich discriminative constituency parser. We approach the self-training problem with the assumption that while the full sentence parse tree produced by a parser may contain errors, some portions of it are more likely to be correct. We hypothesize that instead of feeding the parser the guessed full sentence parse trees of its own, we can break them down into smaller ones, namely n-gram trees, and perform self-training on them. We build an n-gram parser and transfer the distinct expertise of the n-gram parser to the full sentence parser by using the Hierarchical Joint Learning (HJL) approach. The resulting jointly self-trained parser obtains slight improvement over the baseline.

Keywords: Self-training, Constituency Parsing, n-gram Trees

1. Introduction

While statistical approaches for the supervised parsers reach their highs, semi-supervised approaches like self-training of parsers is starting to emerge as a next challenge in the field. A self-trained parser starts its training with a seed set. At some point during its training, it uses what it has learned so far to process newly given sentences and adds the outputted parse trees into its existing training set. It continues to train on an extended training set. If it exceeds the accuracy of the original model, which is trained on the initial seed training set, self-training, that is learning from its own output, is achieved.

We approach the self-training problem with the assumption that while the full sentence parse tree produced by a parser may contain errors, some portions of it are more likely to be correct. We hypothesize that instead of feeding the parser the guessed full sentence parse trees of its own, we can break them down into smaller subtrees and perform self-training on them. We refer to these subtrees as n-gram trees.

Due to their "not-complete" nature, we consider the n-gram parsing task to be different from the full sentence parsing task. In our recent study, we proposed an approach for n-gram parsing and showed that jointly training a discriminative full sentence parser with an n-gram parser improves the performance of the full sentence parser (Çelebi and Özgür, 2013). In this paper, unlike the prior work that self-train full sentence parsers, we self-train n-gram parsers to boost the accuracy of a full parser. We employ the Hierarchical Joint Learning approach, which is described by Finkel et al. (2008), to train the n-gram parser and the full sentence parser with each other simultaneously. This allows the full sentence and self-trained n-gram parsers to *"transfer"* their knowledge to each other during training. This also includes the transfer of the updated knowledge of the n-gram parser gained from its own output.

2. Related Work

In the syntactic parsing literature, almost all of the studies have been based on supervised or semi-supervised methods, with a couple of exceptions of unsupervised approaches, such as (Klein and Manning, 2004). Even though supervised methods achieve the best results, in the absence of sufficient annotated data, they can be outperformed by semi-supervised methods. Self-training is one approach for semi-supervised learning. In self-training, small amount of labelled data is used to annotate unlabelled data, which becomes the labelled data for the learner at the next cycle of its training.

One of the first studies on using self-training for parsing was proposed by Charniak (1997), who first trained a parser on the Penn treebank (Marcus et al., 1993) and then used automatically parsed 30 million words from Wall Street Journal (WSJ) to extent the initial training data. However, the self-trained parser failed to outperform the original model. Roark and Bacchiani (2003) trained Roark's parser on the Brown treebank, then self-trained and tested it on data from the WSJ. While they show some improvement, their parsing results were lower than the state-of-the-art levels. In another study, Steedman et al. (2003) observed that self-training did not yield a significant gain unless the baseline results were sufficiently bad, supporting (Roark and Bacchiani, 2003). Reichart and Rappoport (2007) also showed that one can self-train with a generative parser only if the seed training data size is small.

In (McClosky et al., 2006), they used a generative parser and a discriminative reranker. They trained their system on the Penn treebank and used that trained parser to parse the North American News Text Corpus. They obtained an absolute 1.1 F_1 score improvement over the previous best result on the Penn treebank. Despite this encouraging success, to our knowledge, there hasn't been any research that achieved self-training of a parser that runs on a discriminative approach with no reranker.

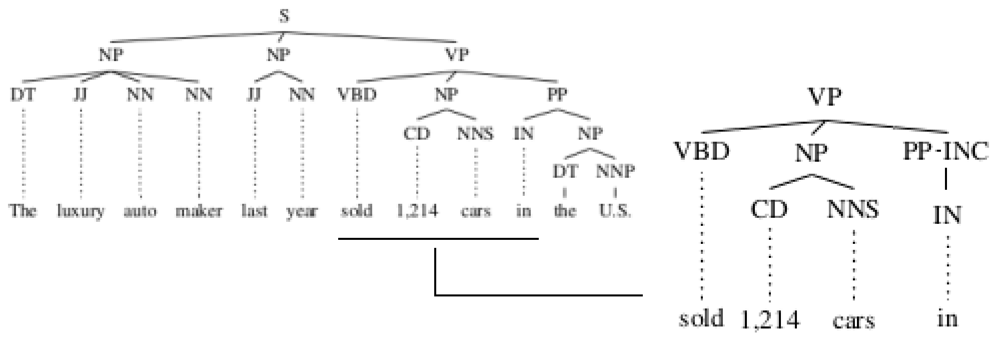

Figure 1: Sample 4-gram tree extracted from a full parse tree.

3. Background

In this section, we briefly describe the model behind our discriminative constituency parser, the concept of n-gram parsing, and our approach for combining full and n-gram parsers using hierarchical joint learning.

3.1. Discriminative Constituency Parsing

Proposed by Lafferty et al. (2001), Conditional Random Fields (CRFs) is a discriminative model which directly optimizes the conditional likelihood of an unobserved variable given the observed data. Discriminative parsers maximize the conditional likelihood of the parse tree given the sentence, that is P(t| s;θ).

$$P(t|s;\theta) = \frac{1}{\mathcal{Z}_s} \prod_{r \in t} \phi(r|s;\theta) \tag{1}$$

Equation 1 describes how to calculate the conditional likelihood of a parse tree t given sentence s. In this equation, CRF-based context-free grammar (CRF-CFG) is represented with local clique potentials ϕ(r|s;θ), where r is one-level subtree of a parse tree t and $\mathcal{Z}_{\text{S}}$ is the partition function. The partition function is calculated over all possible parse trees $\tau(s)$ for a given sentence and then used to normalize the probabilities in Equation 1.

$$\mathcal{Z}_s = \sum_{t \in \tau(s)} \prod_{r \in t} \phi(r|s;\theta) \tag{2}$$

$$\phi(r|s;\theta) = exp \sum_i \theta_i f_i(r,s) \tag{3}$$

Local clique potentials are used to score rules in the Inside-Outside algorithm. Their values are not probabilities but non-negative numbers calculated by taking the exponent of the dot product of the feature vector $f(r,s)$ and parameter vector θ. The feature $f_i(r,s)$ is an indicator function that tells whether feature i is active for given rule r and sentence s.

$$\mathcal{L}(\mathcal{D};\theta) = \sum_{(t,s) \in \mathcal{D}} \left(\sum_{r \in t} \langle f(r,s), \theta \rangle - log Z_{s,\theta} \right) - \sum_i \frac{\theta_i^2}{2\sigma^2} \tag{4}$$

Given a set of training examples, the goal is to choose the parameter values θ such that the conditional log likelihood of these examples, i.e., the objective function $\mathcal{L}$ given in Equation 4, is minimized. In this paper, we use the same implementation and features described in (Çelebi and Özgür, 2013).

3.2. n-gram Parsing

n-gram parsing refers to the process of predicting the syntactic structure that covers n consecutive words in a sentence. This structure is called an n-gram tree. A sample 4-gram tree extracted from a complete parse tree is shown in Figure 1. In order to fit lengthwise, cutting the left or right hand-side of some constituents might be required, while extracting the subtree covering the consecutive words. That cut may sometimes remove the head of the constituent. The requirement during this extraction is to make sure that every syntactic constituent in the extracted n-gram tree keeps its head in the process. We apply this constraint in order to make sure that every n-gram tree is generatively accurate, following the head-driven constituency production process of Michael Collins (Collins, 1999). Thus, n-gram trees are linguistically motivated. The complete n-gram extraction algorithm is given in our previous work (Çelebi and Özgür, 2013). Compared to the complete parse trees, n-gram trees are smaller parse trees with one distinction. That is, they may include constituents that are trimmed from one or both sides in order to fit the constituent lengthwise within the borders of the n-gram. Such constituents are still accepted in our model but considered incomplete, and denoted with the -INC functional tag. Figure 1 includes an example of an incomplete prepositional phrase denoted with PP-INC.

n-gram parsing is fundamentally no different than the conventional parsing of a complete sentence. However, n-grams, especially the short ones, may have no meanings on their own or can be ambiguous due to the absence of the surrounding context. Even though the relatively smaller size of n-gram trees makes it easier and faster to train on them, their incomplete and ambiguous nature makes the n-gram parsing task difficult. Despite all, n-gram parsing can still be useful for the actual full sentence parser, just like the partial parsing of a sentence used for bootstrapping (Abney, 1999). In (Çelebi and Özgür, 2013), we analyze 3- to 9-gram parsing and investigate how an n-gram parser helps the full sentence parser when they are jointly trained together. Our results showed that n-gram parsing can improve full sentence parsing results.

3.3. Hierarchical Joint Learning

We use an instance of the multi-task learning setup called the Hierarchical Joint Learning (HJL) approach introduced in (Finkel and Manning, 2010). HJL enables multiple models to learn more about their tasks due to the commonality among the tasks. By using HJL, we expect the n-gram parser to help the full parser in cases where the n-gram parser is better.

As described in (Finkel and Manning, 2010), the HJL setup connects all the base models with a top prior, which is set to zero-mean Gaussian in our experiments. All the shared parameters between the base models are connected to each other through this prior. The only requirement for HJL is that the base models need to have some common features in addition to the set of features specific to each task. As both parsers employ the same set of feature templates, they have common features through which HJL can operate.

$$\frac{\partial \mathcal{L}_{hier}(\mathcal{D};\theta)}{\partial \theta_{m,i}} = \frac{\partial \mathcal{L}_{hier}(\mathcal{D}_m,\theta_m)}{\partial \theta_{m,i}} - \frac{\theta_{m,i}-\theta_{*,i}}{\sigma_d^2} \quad (5)$$

The parameter values for the shared features are updated by incorporating the top model feature $\theta_{*,i}$ into the parameter update function as in Equation 5. The first term is the partial derivative of Equation 4 with respect to the model parameter $\theta_{m,i}$. The second term ensures that the base model m is not getting apart from the top model by taking the difference between the top model and the corresponding shared parameter value. The variance σ_d^2 is a parameter to tune this relation.

$$\frac{\partial \mathcal{L}_{hier}(\mathcal{D};\theta)}{\partial \theta_{*,i}} = \left(\sum_{m \epsilon \mathcal{M}} \frac{\theta_{*,i}-\theta_{m,i}}{\sigma_m^2}\right) - \frac{\theta_{*,i}}{\sigma_*^2} \quad (6)$$

Equation 6 shows that updates for the top model parameter values are calculated by summing the parameter value differences divided by the base model variance σ_m^2, and then by subtracting the regularization term to prevent overfitting. As in Equation 5, taking the difference of model parameters keeps the top model and base model close to each other.

4. Self-training with n-gram trees

In this paper, we extend the setup of (Çelebi and Özgür, 2013) where we jointly train a full sentence parser with an n-gram parser. We start with training n-gram and full models for N iterations until pausing the system and using the full sentence parser to parse new sentences. Then, we extract K n-gram trees from the outputted parse trees and add them to the existing training data set of the n-gram parser. The number of parsed sentences depends on how many n-gram trees are required as new additions to the training set of the n-gram model. As the training data set of the n-gram parser expands, the n-gram model starts to have more influence on the full model as they have more data compared to the beginning. After the n-gram training sets are expanded with guessed n-grams, training continues for M more iterations. At the end, we retain the parameter values of the full parser and evaluate it on the development and test sets of the Penn treebank.

We investigate three methods for choosing the n-gram trees for self-training. First method chooses K n-gram trees randomly. In case of the second method, it chooses the top K n-gram trees based on their confidence (model) scores assigned by the trained n-gram parser model. In (Çelebi and Özgür, 2013) we showed that the NP, VP, PP and ADJP constituents help most when the full sentence parser and the n-gram parser are jointly trained. Therefore, the third method uses the n-gram trees which include only NP, VP, PP and ADJP constituents.

5. Experiments

5.1. Data

We use the Penn treebank (PTB) (Marcus et al., 1993) for training and testing. It contains approximately 40K sentences of manually annotated sentences from the WSJ. Among its 23 sections, we use 23rd and 22nd sections for development and testing, respectively and the rest is used for training. Due to performance issues with long sentences at training, we use only those sentences with no more than 15 words (WSJ15), which includes 9753 training and 603 test sentences.

The sentences that we use for self-training come from the Reuters RCV1 corpus, which is chosen due to its content-wise similarity to PTB. It contains newswire articles about finance and economy. As in the case of training data, for self-training data, we pick sentences no longer than 15 words and no shorter than 3 words. While selecting, we also make sure that each contains no more than one unknown word that doesn't occur in PTB.

5.2. Baseline

When we run our baseline full sentence parser on WSJ15, it achieves an F_1 score of 90.2% on the development set and 88.3% on the test set. These are taken with the same parameter settings from (Çelebi and Özgür, 2013) with one improvement, that is the tag dictionary built from the development set of the PTB. However, in order to see the benefit of self-training, we need to compare it with respect to the full parser that is jointly trained with the n-gram parser that uses the seed n-gram training set. In Table 3.3., this parser is denoted as BF and the F_1 scores given under the "1K+0K" and "3K+0K" columns are our actual baseline scores. These are the cases where the n-gram training set is not expanded by its own output, denoted by +0K. When we use only seed n-gram training data (1K or 3K), F_1 score on the test set varies between 88.7 and 89.0, which is higher than our stand-alone parser's test score.

5.3. Self-training Results

Table 3.3. shows the results for the jointly self-trained full sentence parser with different n-gram models from 3 to 6. Our runs show that choosing the initial training iteration count N=3 and the self-training iteration count M=17 give the best results, making the total iteration count 20. The caption of the columns indicate the size of the seed n-gram training set and the number of n-grams added to expand the training set for self-training. For example, "1K+3K" means that the seed training set consists of 1000 n-grams and this

n-gram Selection Method	Models	1K+0K	1K+1K	1K+3K	1K+5K	1K+10K	3K+0K	3K+1K	3K+3K	3K+5K	3K+10K
Randomly	BF+3-gram	88.9	88.7	88.1	88.1	88.1	89.0	**89.1**	88.0	88.0	88.2
	BF+4-gram	88.7	**89.0**	88.7	88.0	88.0	88.9	88.7	88.5	88.4	88.8
	BF+5-gram	88.7	**88.8**	87.7	87.6	87.6	88.7	88.4	87.5	87.6	88.2
	BF+6-gram	88.7	**89.0**	88.4	88.8	88.7	88.8	88.8	87.6	88.3	88.0
Confidence Score	BF+3-gram	88.9	**89.0**	**89.0**	**89.0**	**89.0**	89.0	88.8	88.8	88.8	88.8
	BF+4-gram	88.7	**89.0**	**89.0**	**89.0**	**89.0**	88.9	**89.0**	**89.0**	**89.0**	**89.0**
	BF+5-gram	88.7	**88.9**	**88.9**	**88.9**	**88.9**	88.7	88.7	88.7	88.7	88.7
	BF+6-gram	88.7	**89.0**	**89.0**	**89.0**	**89.0**	88.8	88.8	88.8	88.8	88.8
Constituency Type	BF+3-gram	88.9	88.5	88.1	88.1	88.1	89.0	88.7	88.0	88.3	87.9
	BF+4-gram	88.7	88.6	**89.3**	88.0	88.0	88.9	88.6	88.9	88.7	88.5
	BF+5-gram	88.7	88.7	87.6	87.6	87.6	88.7	88.0	87.5	87.5	87.3
	BF+6-gram	88.7	88.4	**88.8**	**88.8**	**88.9**	88.8	88.7	87.6	87.6	88.4

Table 1: F_1 scores obtained by the jointly self-trained full parser on the test set. BF is the baseline full parser.

training set is expanded using 3000 n-grams extracted from the output of the full parser.

Based on the results in Table 3.3., at certain configurations denoted by bold numbers, the jointly self-trained full parser exceeds the baseline, especially when the seed size is small, that is 1K. The best results are obtained when we order the extracted n-grams to be used for self-training by the confidence scores given by the n-gram parser model. In that case, 4-grams lead to better performance compared to 3-, 5- and 6-grams. However, expanding the training set with more n-grams doesn't help much, except slight improvement observed with the 3K seed in the second row of Table 3.3.

Third set of experiments involve being selective about which type of constituents n-gram tree can have. In (Çelebi and Özgür, 2013), we showed that the n-gram parser helps the jointly trained full parser most for parsing the NP, VP, PP and ADJP constituents. However, in this study, when we pick the n-gram trees that contain only these constituents for self-training, the accuracy of the full parser surprisingly deteriorates in all cases.

6. Conclusion and Future Work

In this study we proposed an approach for self-training a constituency parser using n-gram trees. We investigated three strategies for selecting the n-grams to be used for self-training: random selection, selection by confidence score produced by the n-gram parser model, and selection by the types of the constituents. Our results show that selecting the n-grams by their confidence scores achieves better results in general. However, extending the n-gram training set with n-grams extracted from the output of the full parser only helps when the seed n-gram training set size is small. In addition, when we use more outputted n-gram trees, the accuracy of the jointly trained full parser decreases in most cases. For future work, we plan to expand our experiments and investigate whether using more seed n-gram data improves the results.

7. Acknowledgments

The authors would like to thank Brian Roark for his invaluable feedback. This work was supported by the Boğaziçi University Research Fund 12A01P6.

8. References

Abney, S. (1999). Part-of-speech tagging and partial parsing. *Corpus-Based Methods in Language and Speech Processing, Kluwer Academic Publishers, Dordrecht*, pages 118–136.

Çelebi, A. and Özgür, A. (2013). N-gram parsing for jointly training a discriminative constituency parser. *Proceedings of CICLing*, pages 5–12.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *Proceedings of The Fourteenth National Conference on Artificial Intelligence (AAAI)*, pages 598–603.

Collins, M. (1999). Head-driven statistical models for natural language parsing. *Doctoral Dissertation Department of Computer and Information Science, University of Pennsylvania.*

Finkel, J.R. and Manning, C.D. (2010). Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. *Proceedings of ACL*, pages 720–728.

Finkel, J.R., Kleeman, A., and Manning, C.D. (2008). Efficient, feature-based conditional random field parsing. *Proceedings of ACL-HLT*, pages 959–967.

Klein, D. and Manning, C.D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. *Proceedings of ACL.*

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML*, pages 282–289.

Marcus, M., SantoriniB., and MarcinKiewicz, M.A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. *Proceedings of NAACL-HLT*, pages 152–159.

Reichart, R. and Rappoport, A. (2007). Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. *Proceedings of ACL*, pages 616–623.

Roark, B. and Bacchiani, M. (2003). Supervised and unsupervised pcfg adapation to novel domains. *Proceedings of NAACL-HLT*, pages 205–212.

Steedman, M., Baker, S., Crim, J., Clark, S., HockenmaierJ., Hwa, R., Osborne, M., Ruhlen, P., and Sarkar, A. (2003). CLSP WS-02 Final Report: Semi-supervised training for statistical parsing. *Technical Report, Johns Hopkins University.*