JMLR: Workshop and Conference Proceedings 21:224–236, 2012 The 11th ICGI

Model Merging versus Model Splitting Context-Free Grammar Induction

Menno van Zaanen M.M.vanZaanen@uvt.nl
Nanne van Noord N.J.E.vanNoord@uvt.nl
Tilburg University, Tilburg, The Netherlands

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

When comparing different grammatical inference algorithms, it becomes evident that generic techniques have been used in different systems. Several finite-state learning algorithms use state-merging as their underlying technique and a collection of grammatical inference algorithms that aim to learn context-free grammars build on the concept of substitutability to identify potential grammar rules. When learning context-free grammars, there are essentially two approaches: model merging, which generalizes with more data, and model splitting, which specializes with more data. Both approaches can be combined sequentially in a generic framework. In this article, we investigate the impact of different approaches within the first phase of the framework on system performance.

Keywords: Context-free grammars, model merging, model splitting, evaluation

1. Introduction

Formal grammatical inference deals with formal learnability of families of languages. Results in this field are typically mathematical proofs of (non-)learnability of classes of languages under certain learning settings. Often, the learnability proofs are based on algorithms, presented in pseudo-code, that can learn any grammar from the family under consideration.

In contrast, empirical grammatical inference deals with the learning of specific languages. These languages come from a family that is known to be learnable efficiently, but the formal description of the family of languages is unknown. An example of such a family is the class of natural languages. We know these are learnable efficiently, as humans can do this within practical limitations (Adriaans and van Zaanen, 2006).

The algorithms that are used in both formal and empirical grammatical inference often follow a similar approach or build on a common underlying mechanism. The details or specific settings change the bias of the algorithm which allows for learning of different, but sometimes related families of languages.

Several algorithms that focus on learning languages that are (sub-sets of) regular languages use state-merging as the underlying principle. Starting from a finite-state machine that accepts only the sample training sequences, states are merged according to some evaluation criteria. Merging states leads to generalization over the input sequences, which means that a more general language is created (Carrasco and Oncina, 1994; Lang et al., 1998).

In the area of learning context-free languages, a similar underlying technique or principle can be identified. Many context-free grammar (CFG) learning systems are based on the

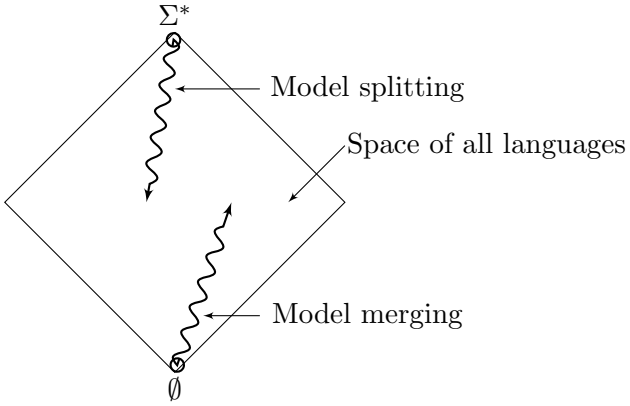

Figure 1: Model merging and model splitting in the space of possible languages.

notion of substitutability. This notion corresponds well with the idea of context-freeness. Non-terminals in a CFG may be rewritten according to the available grammar rules regardless of their context. Systems based on the notion of substitutability reverse this idea and search for regularities in the data that may have been generated through the application of CFG rules (van Zaanen and de la Higuera, 2011).

Another way of looking at identifying the underlying grammar in the collection of all possible languages (of a particular class) is the direction of the search path in the space of potential languages. One can identify two general directions: model merging (Stolcke and Omohundro, 1994) or model splitting (Belz, 2002). The model merging approach starts from a model that describes exactly the example training sequences. The search algorithm then tries to make the model more general. In the context of learning finite-state machines, this corresponds to merging nodes, which may make the accepted language more general.

Model splitting starts from a general language and the learning algorithm searches for a language that is more specific, given the example sequences. To allow the algorithm to describe more specific languages, the existing model may need to be split to indicate sequences that are part of the language or not. According to Starkie (2002, p. 4): "[the grammar] can be improved by simply removing those rules that generate the negative examples." However, this can also be achieved by giving preference to highly likely structures in the model (and pruning unlikely structures).

The ideas on model merging and model splitting are illustrated in Figure 1. The curved lines indicate the learning process for both model merging and model splitting within the space of possible languages (delimited by the pre-defined bias of the search algorithm).

If algorithms "overshoot" their mark, model merging algorithms will over generate (meaning that too many sequences are accepted as valid), whereas model splitting algorithms under generate (meaning that too few sequences are accepted as valid).

Many CFG learning algorithms consist of two phases. The first phase generates potential structures and the second phase prunes unwanted structures. The generation phase is typically in the form of a model merging algorithm. The selection phase performs model splitting, starting from the end point of the generation phase.

If the generation phase over generates, the selection phase has to prune more unwanted structure to reach the correct underlying language. More over generation means that more potential languages may still be considered during the selection phrase. On the other hand, if the generation phase does not generate enough structure, the selection phase cannot reach the correct language and this may only lead to an approximation.

The difficulty of the selection phase increases as the generation phase over generates more. However, increasing the amount of over generation increases the likelihood that the correct language is still reachable during the selection phase. Hence, reducing the amount of over generation makes sense if we know that the correct grammar is still likely to be reachable. This increases the complexity of the generation phase.

In this article, we will investigate the impact of the choice of the generation model (and hence the amount of over or under generation) with respect to the optimal result that can still be reached given a perfect selection phase.

2. Background

In the last two decades, several empirical grammatical inference systems that aim to learn CFGs have been developed. Even though most of these systems rely on the same underlying principle, namely that of substitutability, there are remarkable differences. In this section we will discuss some of the better known systems and describe their approach in the context of model merging and model splitting.

2.1. EMILE

EMILE is a grammatical inference approach developed by Adriaans (1992). A working implementation is also available (Vervoort, 2000; Adriaans and Vervoort, 2002). EMILE relies heavily on the notion of substitutability. Examples sequence are converted in recurring *expressions* and *contexts*. Expressions are subsequences that occur within particular contexts. When different expressions are found in the same context, this corresponds to constituents according to the notion of substitutability. EMILE introduces grammar rules once enough evidence is available. Exactly when new rules are introduced can be adjusted (using parameters). Additionally, if an expression is identified as the right hand side of a grammar rule, all occurrences of this expression in the data collection are structured according to that rule.

This approach is a clear model merging approach. At first, only the example sentences are recognized as valid. The system searches for recurring patterns and based on these patterns, grammar rules are introduced that allow for generalization.

2.2. ADIOS

ADIOS (Edelman et al., 2004; Solan et al., 2005) stands for "Automatic Distillation of Structure". This systems also makes direct use of the idea of substitutability, although the implementation is quite different from that of EMILE. To start with, a graph, quite similar to a finite state machine, is created with a common start and end node. Each example sequence forms a path from the start to the end node. Next, the learning phase starts. ADIOS searches for significant patterns, where significance is computed using the Motif

Extraction (MEX) procedure. This procedure considers the in and out degrees of the node in the graph. In other words, MEX computes P_L for a sub-path $(e_j; e_i)$ (with e_x being a node and $(e_x; e_y)$ being a path from node e_x to e_y), which is the fraction of the number of sub-paths $(e_j; e_i)$ over the number of sub-paths $(e_j; e_{i+1})$. The algorithm searches for nodes for which there are large decreases in P_L between consecutive nodes in the graph. Similarly, P_R is computed by dividing the number of sub-paths $(e_i; e_j)$ and $(e_i; e_{j-1})$. P_L and P_R essentially model a Markov probability over sub-paths in a graph. The nodes with large P_L and P_R changes are used as start and end points for a pattern. Since the begin and end points of these patterns are found by taking the left and right context into account, the patterns are inherently based on the notion of substitutability.

Like EMILE, ADIOS is a model merging approach. Starting from the sample sequences, slowly the grammar is generalized. The main difference with EMILE is the way the patterns are found. Here, the MEX procedure identifies the start and end points of the interesting patterns.

2.3. CCM

The Constituent-Context Model (CCM) (Klein and Manning, 2002) is an empirical grammatical inference model based on classic linguistic constituency tests (Radford, 1988), which corresponds with the notion of substitutability. The focus of the algorithm lies on parameter searching where the parameters describe the likelihood of structure on the sequences. The first step of the algorithm places a uniform distribution of the set of possible binary trees on the sequences.[1] Next, the expectation maximization (EM) algorithm (Dempster et al., 1977) is applied to the potential trees.

There are several differences between CCM and the previous systems. Firstly, CCM works on sequences of part-of-speech tags, whereas EMILE, ADIOS and, discussed in Section 2.5, ABL start from sequences of words. Secondly, the output of CCM is by choice a collection of binary tree structures. The systems discussed so far are all able to generate n-ary tree structures. Finally, CCM is a model splitting approach. It starts from all possible (binary) tree structures and selects the best fitting structure. The notion of substitution is used only in the selection of structure, not in the generation of structure.

2.4. U-DOP

U-DOP (Bod, 2006a,b) is an unsupervised approach that relies heavily on the statistical model of Data-Oriented Parsing (DOP) (Bod, 1995, 1998; Bod et al., 2003). Similarly, to CCM, all potential binary tree structures are generated. Next, the trees are converted into a practically usable form of the DOP formalism. This allows for a statistically stronger representation (even though the structural representation of U-DOP is equivalent to that of CCM). Identifying the structure of a sequence can now be done by parsing the sequences using the generated grammar. Since some symbols are more likely to occur in a subsequence, some elementary sub-trees will be more likely. U-DOP makes use of this to give preference to these structures.

1. This limits the potential structure of the output in the form of binary trees or grammar rules in Chomsky Normal Form.

The main difference between U-DOP and CCM is the statistical model used to identify the most likely structure. Apart from that, U-DOP is in many ways similar to CCM. It also starts from sequences of part-of-speech, it generates binary tree structures and it is also a model splitting approach.

2.5. ABL

Alignment-Based Learning (ABL) (van Zaanen, 2000a,b, 2002) is another grammatical inference system that is explicitly based on the notion of substitutability. The first phase aligns sample sequences in pairs. Unequal parts of the sequences are stored as hypotheses, or potential constituents. A second phase identifies the most likely of these hypotheses. This step removes overlapping brackets (which makes sure the result is a tree structure).

ABL is similar to EMILE and ADIOS. The first phase in ABL is a model merging phase, which starts from accepting only the sample sequences. Slowly, more structure is added, which therefor generalizes over the training data. However, the second phase is a model splitting step. This phase prunes the available structure, similarly to CCM and U-DOP.

In van Zaanen (2002), ABL is proposed as a general framework. All systems described so far fit this framework. The first step in the framework generates potential structure. In the case of EMILE and ADIOS, the introduction of structure is done carefully, whereas U-DOP and CCM over generate. The second step prunes the introduced structure. In EMILE and ADIOS, this step is not explicitly required. However, U-DOP and CCM rely heavily on this step.

3. Experiments

As described in the previous section, we can map existing grammatical inference systems into the ABL framework. This allows us to investigate the effectiveness of the generation and selection (or model merging and model splitting) steps of the algorithms in a common framework. In this paper we focus on the effectiveness of the generation phase only.

In this section, we will describe experimental results that aim to measure the effectiveness and potential of the generation step in the ABL framework. We do this by comparing a selection of generation algorithms that have been used in previous work.

3.1. Datasets

We will investigate the effectiveness of the generation phase by applying different generation methods to natural language data collections. In particular, we will use the Air Travel Information System (ATIS) and Wall Street Journal (WSJ) sections of the Penn Treebank 3 (Marcus et al., 1993). The ATIS section contains 578 sentences on air travel. These are relatively short sentences, with an average length of 7.5 (sd=3.9) words.

A more extensive collection is the WSJ part of the Penn Treebank. We follow Klein and Manning (2002); Bod (2006a) by selecting all sentences in the corpus of at most ten words (discarding traces and punctuation). This results in a collection of 7,092 sentences with an average length of 7.0 (sd=2.5) words. We will call this subset WSJ10.

In addition to the results on the WSJ10 collection we are also interested in the effects when larger datasets are used. As a comparison, we use a second selection of the entire WSJ

part of the Penn Treebank consisting of all sentences of at most 20 words (again discarding traces and punctuation). This results in a second WSJ collection (WSJ20), which contains 25,017 sentences with an average length of 13.2 (std=4.8) words.

All results presented here are computed using 10-fold cross validation. The entire dataset is divided into ten folds. Each fold is used as testing data once and the remaining nine folds are used for training.

3.2. Metrics

Before we can show the actual results on the datasets, there are several choices that have to be made. Firstly, we have to decide exactly what structure to use when computing metrics. Obviously, the learned structure should be evaluated, which is done by comparing it against a gold standard structure. However, some structure is "trivial". For instance, structure that indicates that a sequence is a sequence (shown by a pair of brackets around the entire sequence) is trivial. Similarly, brackets around single symbols are trivial and empty structure (brackets that do not span any word) may also be considered trivial.

We have analyzed the impact of trivial structure on the results. There are significant ($p < .001$) differences between removing word or sentence spanning brackets or both. Even though removing empty brackets (such as traces, that do not span any words) does not have a significant impact, single word spanning brackets or sentence spanning brackets do influence the results significantly.

Given that adding sentence spanning and word spanning brackets can be inserted trivially, these structures do not add to the knowledge of the language and as such should not be taken into account during the evaluation. All results shown in this article are based on data without trivial structure. The choice of removing all trivial structure (compared against keeping all structure) leads to significantly lower scores.

Secondly, we need to choose metrics to show the results of the different systems. Following most publications in the area of grammatical inference on natural language data, we will be using precision, recall and f-score metrics as introduced by van Zaanen and Adriaans (2001). These are standard metrics taking from the field of information retrieval (van Rijsbergen, 1979). Precision describes the correctness of the induced structure, whereas recall measures the completeness of the learned structure. The f-score is the geometric mean between precision and recall. The metrics are formally defined as:

$$\text{Precision} = \frac{\sum_{s \in structure} |correct(gold(s),\ learned(s))|}{\sum_{s \in structure} |learned(s)|}$$

$$\text{Recall} = \frac{\sum_{s \in structure} |correct(gold(s),\ learned(s))|}{\sum_{s \in structure} |gold(s)|}$$

$$\text{F-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Two versions of these metrics exist: micro and macro averaged. We have analyzed the impact of both versions on the results. As could be expected, there are significant differences ($p < .001$) between the different variants. Since the different metrics are significantly

different, the choice of metrics is important. Here, we show micro averaged performance measures. Choosing the micro averaged metrics leads to significantly lower scores.[2]

Finally, to get a complete image, we have looked at the metrics themselves. We found significant differences ($p < .001$) between precision and recall. Also, the f-score is statistically significantly different ($p < .001$) from precision and recall. This indicates that as the f-score is the geometric mean of precision and recall, the f-score is in between the precision and recall values and the values are relatively distant from each other.

3.3. Systems

In this article we report on the results for seven different alignment systems on the different treebanks. The different systems can be grouped into three classes. The first class contains systems that generate structure in a simple heuristic way. This class contains the systems: *left*, *right*, and *both*, which generate left branching (from the start non-terminal, only the left-most non-terminals are expanded), right branching (only the right-most non-terminals are expanded), and a random choice between either the left or right branching tree structure on a given sentence, respectively.

The second group comes from the original ABL system. The different systems find the longest common subsequences in two sentences. These subsequences are equal parts in both sentences and the remainder of the sentences are the unequal parts. The unequal parts in both sentences are stored as hypotheses, or potential constituents.

All systems in this group are based on the dynamic programming edit distance (Wagner and Fischer, 1974) algorithm. The differences lie in how ambiguity of alignments is handled. Sometimes, there are multiple possible edit transcripts that lead to the same minimal edit distance. The first system, *wm*, which stands for "Wagner Fisher minimal edit distance", identifies a minimal edit distance alignment and introduces structure based on that alignment. The second system, *wb*, which stands for "Wagner Fisher biased edit distance", has a modified cost function, which prefers alignments of words that are relatively "close together" in both sentences. The third system in this group is called *all*. It uses all possible alignments with the minimal edit distance. This introduces more potential structure.

The third group consists of only one system, which is called *binary*. This system corresponds to the structure generation system used in U-DOP and CCM. All possible binary tree structures are generated for a sentence. This comes down to generating all possible pairs of brackets.

3.4. Results

We have applied all seven systems to the ATIS treebank. The structure generated by the systems was then compared against the gold standard treebank structure. The results of this comparison can be found in Table 1. In addition to the precision, recall and f-score results, we have added a column that shows the number of pairs of brackets that have been introduced by each system.

The results on the ATIS dataset show that English is generally a right branching language. The *right* system outperforms all other systems on precision and f-score. In par-

2. We follow Klein (2004) in the choice of metrics, but exclude sentence spanning brackets.

Table 1: Results on the ATIS dataset for a variety of alignment systems.

Alignment	Precision	Recall	F-score	# Brackets
left	6.82	9.90	8.08	3,209
right	**27.49**	39.86	**32.53**	3,209
both	17.54	25.44	20.77	3,209
wb	16.82	55.31	25.79	7,278
wm	16.41	57.34	25.52	7,731
all	15.53	61.00	24.76	8,693
binary	12.25	**100.00**	21.83	18,066

Table 2: Results on the WSJ10 dataset for a variety of alignment systems.

Alignment	Precision	Recall	F-score	# Brackets
left	11.96	16.21	13.77	35,596
right	**45.47**	61.63	**52.33**	35,596
both	28.76	38.98	33.10	35,596
wb	28.11	52.73	36.67	49,279
wm	24.36	76.37	36.94	82,352
all	23.94	77.02	36.52	84,512
binary	15.99	**100.00**	27.57	164,288

ticular, *right* has the highest precision of all systems under consideration. In contrast, *left* performs worst.

The ABL oriented systems outperform the systems in the first group on recall, which indicates that more brackets in the gold standard dataset are found. This is (at least partially) accomplished by introducing more pairs of brackets (as can be seen in the rightmost column). Because the precision metric is lower than that of the *right* system, the f-score of the ABL systems is still lower.

Finally, the *binary* system has a perfect recall result. This is because all possible pairs of brackets are introduced. As a result all structure found in the gold standard is also to be found in the *binary* structure. Since the *binary* system introduces more brackets (about two and a half times as much as the ABL systems and about six times as much as the systems in the first group), the precision is lower.

The results on the WSJ10 dataset can be found in Table 2. These results are very similar to those on the ATIS dataset. Again, the *right* system performs best. The *wm* and *all* systems are a bit more effective on the recall metric, which means that more (correct) brackets identified. Additionally, the ABL systems outperform the *binary* system on precision, which means that those systems introduce fewer incorrect brackets. Obviously, the *binary* system again shows a perfect recall as all possible brackets are introduced.

Comparing the WSJ10 results in Table 2 against those of WSJ20 in Table 3, we see that the results of the first group of systems goes down. This is the case on all metrics, so not only are fewer introduced brackets correct (indicated by precision), also fewer correct brackets are found (indicated by recall). The *binary* system keeps a perfect recall, but as can be seen by the precision metric, a larger proportion of the introduced brackets is incorrect.

Table 3: Results on the WSJ20 dataset for a variety of alignment systems.

Alignment	Precision	Recall	F-score	Brackets
left	7.68	10.50	8.87	280,927
right	**35.91**	49.09	**41.48**	280,927
both	21.83	29.84	25.21	280,927
wb	13.76	75.88	23.29	1,133,423
wm	13.26	80.04	22.75	1,240,195
all	12.48	82.19	21.67	1,353,621
binary	9.00	**100.00**	16.52	2,282,623

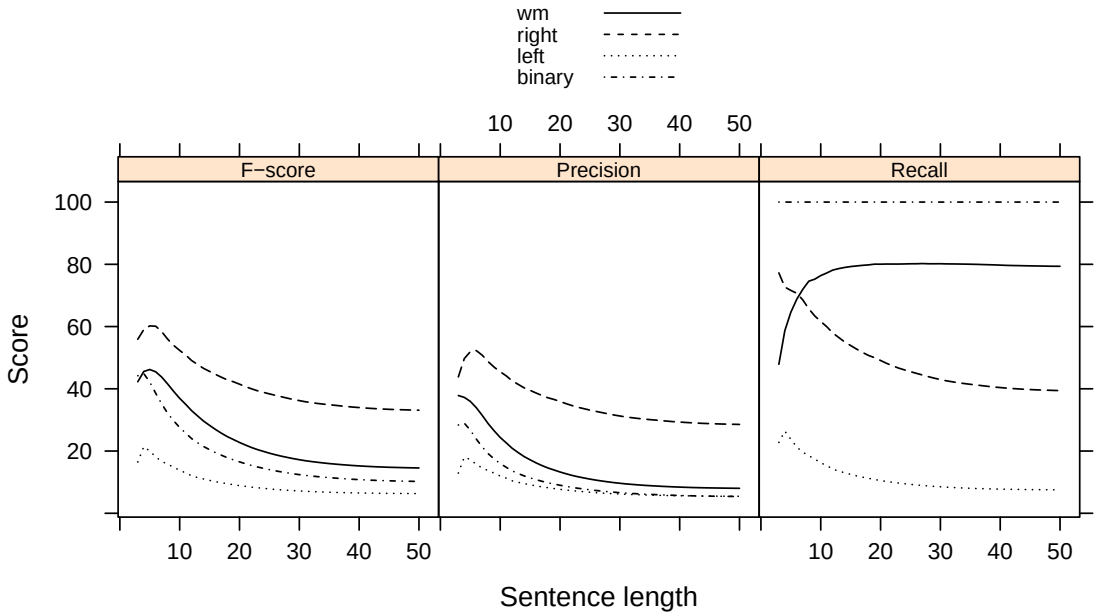

Figure 2: F-score, precision and recall results on subsets of WSJ dataset (the x-axes indicate the maximum sentence length of the subset) for a variety of alignment systems.

In contrast, the ABL metrics have a higher recall, which means that relatively more of the brackets in the gold standard are found. However, similarly to the *binary* system, more incorrect brackets are introduced.

The total number of brackets introduced also grows when the sentences get longer. This is most obvious in the *binary* system that introduces all possible pairs of brackets, but also the ABL systems introduce relatively more brackets when compared against the systems in the first group, which never introduce pairs of brackets that overlap.

The fact that the recall results of the ABL systems increase when they are applied to larger data collections (whereas the systems in the first group have lower results), has led us to investigate the changing of the results when the size of the dataset is modified. In Figure 2 we show the precision, recall and f-score results of the *right*, *left*, *wm*, and *binary*

systems. We have decided to leave out the *both*, *wb* and *all* systems, which would otherwise clutter the figure. The trend of those systems is similar to other systems in their group.

In the figure, we can see that the recall of the systems in the first group goes down when the dataset contains more, and longer sentences. The recall of the *wm* system increases, but seems to level off. The recall of the *binary* system remains at 100%.

The precision of the *right* system peaks when the sentence length is six. For the *left* system, the peak is at sentence length four. The *binary* system has a precision peak when the sentences are of length four and the *wm* system has the highest precision when the sentence length is three. We do not currently know exactly why these differences occur.

The impact of the peaks in precision are also seen in the f-score results. However, due to the increase in recall for the *wm* system, the peak in f-score is at sentence length five.

3.5. Discussion

The results show that the three different groups of systems (based on left/right branching structure, taken from the ABL system, or generating all possible structure) all lead to quite different behavior. The *binary* system always has perfect recall, whereas the *right* system, which has the highest f-score, has decreasing recall when the dataset gets larger. The *wm* system has increasing recall, but it seems that this approach will not lead to perfect recall even with larger datasets.

Recall is an important metric in the overall picture of the systems. The ABL framework consists of two phases, of which we are only analyzing the first here. The second phase selects structure from the structure generated in the first phase. This means that the recall of the complete system cannot be higher than the recall of the first phase. The second phase can only increase precision.

Relating the results to the model merging versus model splitting division, we see that the model merging systems (*wm*, *wb*, and *all*) are incremental learners that perform better with more training data. In contrast, the different model splitting systems (*left*, *right*, and *binary*) essentially show comparable behavior. Removing more structure leads to worse results. *Binary* removes least structure, followed by *right* and *left*.

The importance of recall indicates that the *binary* system is, out of the investigated systems, best suited for this task. Originally, we expected that the ABL-based systems, which are all model merging approaches, would lead to higher recall (close to perfect) if the datasets got larger as well as precision. It turns out that even though precision is higher, recall remains lower, compared to the *binary* system. ABL-based systems are less suited for the task. This is in addition to the limitations on the upper bound of the approach as discussed in Luque and Infante-Lopez (2010).

However, there is more to the results than simply looking at recall. Introducing more structure during the first phase makes selecting the correct structure in the second phase harder. Consider the extremes, if the first phase only introduces correct structure, the second phase can simply select all structure. However, if the first phase introduces many incorrect structures, the second phase has a hard time selecting the correct brackets. The actual impact of this problem cannot really be seen from the results discussed here. The right-most column in the tables indicate that even though the *binary* system has a perfect

recall, about twice the amount of structure, compared against the ABL-based systems, is introduced.

Initial experiments investigating the impact of the generation method indicate that, indeed the structure of the *binary* generation system leads to better overall results after the selection phase. However, further research into the effectiveness of different selection learning systems will need to be performed.

4. Conclusion

We have analyzed seven systems that introduce structure to be used in the first phase of the generic ABL framework for learning of CFGs. The systems were grouped based on their approach: simple systems that introduce left or right branching structures, systems based on alignment learning, and a system that introduces all possible structure.

The model splitting systems perform worse with more data, whereas the model merging systems increase performance. The model merging alignment-based systems outperform the simple systems, but they are unable to reach perfect recall, even with larger datasets. The system that generates all structure leads to perfect recall. Even though more research has to be done on the effect of the amount of structure that is generated by these systems, these results serve as an indication that systems such as U-DOP and CCM can reach better results than ABL, which is based on a sub-optimal structure generation approach.

References

P. Adriaans and M. Vervoort. The Emile 4.1 grammar induction toolbox. In Adriaans et al. (2002), pages 293–295.

Pieter Adriaans, Henning Fernau, and Menno van Zaanen, editors. *Grammatical Inference: Algorithms and Applications (ICGI); Amsterdam, the Netherlands*, volume 2482 of *Lecture Notes in AI*, Berlin Heidelberg, Germany, September 23–25 2002. Springer-Verlag.

Pieter W. Adriaans and Menno M. van Zaanen. Computational grammatical inference. In Dawn E. Holmes and Lakhmi C. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, chapter 7. Springer-Verlag, Berlin Heidelberg, Germany, 2006. ISBN 3-540-30609-9.

Pieter Willem Adriaans. *Language Learning from a Categorial Perspective*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, November 1992.

Anja Belz. Pcfg learning by nonterminal partition search. In Adriaans et al. (2002), pages 304–308.

Rens Bod. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, September 1995.

Rens Bod. *Beyond Grammar—An Experience-Based Theory of Language*, volume 88 of *CSLI Lecture Notes*. Center for Study of Language and Information (CSLI) Publications, Stanford:CA, USA, 1998.

Rens Bod. Unsupervised parsing with u-dop. In *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 85–92, Morristown, NJ, USA, 2006a. Association for Computational Linguistics.

Rens Bod. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the Association of Computational Linguistics (ACL); Sydney, Australia*, pages 865–872. Association for Computational Linguistics, 2006b.

Rens Bod, Khalil Sima'an, and Remko Scha, editors. *Data Oriented Parsing*. Center for Study of Language and Information (CSLI) Publications, Stanford:CA, USA, 2003. ISBN: 1-57586-435-5.

Rafael Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In ICGI, pages 139–152.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Shimon Edelman, Zach Solan, Eytan Ruppin, and David Horn. Learning syntactic constructions from raw corpora. In *Proceedings of the 29th Boston University Conference on Language Development, Boston:MA, USA*, 2004.

ICGI. *Proceedings of the Second International Conference on Grammar Inference and Applications; Alicante, Spain*, 1994.

Dan Klein. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *42th Annual Meeting of the Association for Computational Linguistics; Barcelona, Spain*, pages 479–486, 2004.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *40th Annual Meeting of the Association for Computational Linguistics; Philadelphia:PA, USA*, pages 128–135. Association for Computational Linguistics, July 2002.

Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutzki, editors, *Proceedings of the Fourth International Conference on Grammar Inference*, volume 1433 of *Lecture Notes in AI*, pages 1–12, Berlin Heidelberg, Germany, 1998. Springer-Verlag.

Franco Luque and Gabriel Infante-Lopez. Bounding the maximal parsing performance of non-terminally separated grammars. In José Sempere and Pedro García, editors, *Grammatical Inference: Theoretical Results and Applications; Valencia, Spain*, number 6339 in Lecture Notes in AI, pages 135–147, Berlin Heidelberg, Germany, 2010. Springer-Verlag.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

A. Radford. *Transformational grammar: A first course*. Cambridge University Press, Cambridge, UK, 1988.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, August 2005.

Bradford Starkie. Inferring attribute grammars with structured data for natural language processing. In Adriaans et al. (2002), pages 237–248.

Andreas Stolcke and Stephen Omohundro. Inducing probabilistic grammars by bayesian model merging. In ICGI, pages 106–118.

C. J. van Rijsbergen. *Information Retrieval*. University of Glasgow, Glasgow, UK, 2nd edition, 1979. Printout.

M. van Zaanen and C. de la Higuera. Computational language learning. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 765–780. pub-elsevier, pub-elsevier-adr, 2nd edition edition, 2011.

Menno van Zaanen. ABL: Alignment-Based Learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING); Saarbrücken, Germany*, pages 961–967. Association for Computational Linguistics, July 31–August 4 2000a.

Menno van Zaanen. Bootstrapping syntax and recursion using Alignment-Based Learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning; Stanford:CA, USA*, pages 1063–1070, June 29–July 2 2000b.

Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, Leeds, UK, January 2002.

Menno van Zaanen and Pieter Adriaans. Alignment-Based Learning versus EMILE: A comparison. In *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC); Amsterdam, the Netherlands*, pages 315–322, October 2001.

Marco R. Vervoort. *Games, Walks and Grammars*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, September 2000.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.