Bootstrapping Syntax and Recursion using Alignment-Based Learning

Menno van Zaanen MENNO@SCS.LEEDS.AC.UK
School of Computer Studies, University of Leeds, Woodhouse Lane, LS2 9JT Leeds, UK

Abstract

This paper introduces a new type of unsupervised learning algorithm, based on the alignment of sentences and Harris's (1951) notion of interchangeability. The algorithm is applied to an untagged, unstructured corpus of natural language sentences, resulting in a labelled, bracketed version of the corpus. Firstly, the algorithm aligns all sentences in the corpus in pairs, resulting in a partition of the sentences consisting of parts of the sentences that are similar in both sentences and parts that are dissimilar. This information is used to find (possibly overlapping) constituents. Next, the algorithm selects (non-overlapping) constituents. Several instances of the algorithm are applied to the ATIS corpus (Marcus et al., 1993) and the OVIS[1] corpus (Bonnema et al., 1997). Apart from the promising numerical results, the most striking result is that even the simplest algorithm based on alignment learns recursion.

1. Introduction

This paper introduces a new type of grammar learning algorithm, which uses the alignment of sentences to find possible constituents in the form of labelled brackets. When all possible constituents are found, the algorithm selects the best constituents. We call this type of algorithm Alignment-Based Learning (ABL).

The main goal of the algorithm is to automatically find constituents in plain sentences in an unsupervised way. The only information the algorithm uses stems from these sentences; no additional information (for example POS-tags) is used.

The underlying idea behind our algorithm is Harris's notion of interchangeability; *two constituents of the same type can be replaced.* ABL finds constituents by looking for parts of sentences that can be replaced and assumes that these parts of the sentences are probably constituents, which is Harris's notion reversed.

At some point the algorithm may have learned possible constituents that overlap. Since generating results is done by comparing a learned structure to the structure in the corpus, the algorithm needs to disambiguate conflicting constituents. This process continues one tree structure covering the sentence remains.

This paper is organised as follows. We start out by describing the algorithm in detail. We then report experimental results from various instances of the algorithm. We discuss the algorithm in relation to other grammar learning algorithms, followed by description of some future research.

2. Algorithm

In this section we describe an algorithm that learns structure in the form of labelled brackets on a corpus of natural language sentences. This corpus is a selection of plain sentences containing no brackets or labels.

The algorithm was developed on several small corpora. These corpora indicated some problems when simply applying Harris's idea to learn structure. These problems were solved by introducing two phases: *alignment learning* and *selection learning*, which will now be described in more detail.

2.1 Alignment Learning

The first phase of the algorithm is called *alignment learning*. It finds possible constituents by aligning all plain sentences from memory in pairs. Aligning uncovers parts of the sentences that are similar in both sentences and parts that are dissimilar. Finally, the dissimilar parts are stored as possible constituents of the same type. This is shown by grouping the parts and labelling them with a non-terminal.

Finding constituents like this is based on Harris's notion of interchangeability. Harris (1951) states that

[1] Openbaar Vervoer Informatie Systeem (OVIS) stands for Public Transport Information System.

two constituents of the same type can be replaced. The alignment learning algorithm tries to find parts of sentences that can be replaced, indicating that these parts might be constituents.

We have included a simple example taken from the ATIS corpus to give a visualisation of the algorithm in Table 1. It shows that that *Show me* is similar in both sentences and *flights from Atlanta to Boston* and *the rates for flight 1943* are dissimilar. The dissimilar parts are then taken as possible constituents of the same type. In this example there are only two dissimilar parts, but if there were more dissimilar parts, they would also be grouped. However, a different nonterminal would be assigned to them (as can be seen in sentences 3 and 4 in Table 2).

Table 1. Bootstrapping structure

Show me flights from Atlanta to Boston
Show me the rates for flight 1943
Show me (flights from Atlanta to Boston)$_X$
Show me (the rates for flight 1943)$_X$

Note that if the algorithm tries to align two completely dissimilar sentences, no similar parts can be found at all. This means that no inner structure can be learned. The only constituents that can be learned are those on sentence level, since the entire sentences can be seen as dissimilar parts.

2.1.1 Aligning

The alignment of two sentences can be accomplished in several ways. Three different algorithms have been implemented, which will be discussed in more detail here.

Firstly, we implemented the edit distance algorithm by Wagner and Fischer (1974) to find the similar word groups in the sentences. It finds the minimum edit cost to change one sentence into the other based on a predefined cost function γ. The possible edit operations are insertion, deletion and substitution, which are used to change one sentence in the other. It is possible to find the words in the sentences that match (i.e. no edit operation). These words combined are the similar parts of the two sentences.

The cost function of the edit distance algorithm can be defined to find the longest common subsequences in two sentences. The cost function γ returns 1 for an insert or delete operation, 0 if the two arguments are the same and 2 if the two arguments are different. We will call this algorithm the *default* γ.

Unfortunately, this approach has the disadvantage depicted in Table 2. Here, the algorithm aligns sentences 1 and 2. Default γ finds that *San Francisco* is the longest common subsequence. This is correct, but results in an unwanted syntactic structure as can be seen in sentences 3 and 4.

Table 2. Ambiguous alignments

1 *from San Francisco to Dallas*
2 *from Dallas to San Francisco*
3 *from ( )$_{X_1}$ San Francisco (to Dallas)$_{X_2}$*
4 *from (Dallas to)$_{X_1}$ San Francisco ( )$_{X_2}$*
5 *from (San Francisco to)$_{X_3}$ Dallas ( )$_{X_4}$*
6 *from ( )$_{X_3}$ Dallas (to San Francisco)$_{X_4}$*
7 *from (San Francisco)$_{X_5}$ to (Dallas)$_{X_6}$*
8 *from (Dallas)$_{X_5}$ to (San Francisco)$_{X_6}$*

The problem is that aligning *San Francisco* results in constituents that differ greatly in length. In other words, the position of *San Francisco* in both sentences differs significantly. Similarly, aligning *Dallas* results in unintended constituents (see sentences 5 and 6 in Table 2), but aligning *to* would not (as can be seen in sentences 7 and 8), since *to* resides more "in the middle" of both sentences.

This problem is solved by redefining the cost function of the edit distance algorithm to prefer matches between words that have similar offsets in the sentences. When two words have similar offsets, the cost will be low, but when the words are far apart, the cost will be higher. We will call this algorithm *biased* γ. The biased γ is similar to the default γ, only in case of a match, the biased γ returns

$$\left| \frac{i_1}{s_1} - \frac{i_2}{s_2} \right| * \frac{s_1 + s_2}{2}$$

where i_1 and i_2 are the indices of the considered words in sentence 1 and sentence 2 while s_1 and s_2 are the lengths of sentence 1 and sentence 2 respectively.

Although biased γ solves the problem in Table 2, one may argue if this solution is always valid. It may be the case that sometimes a "long distance" alignment is preferable. Therefore, we implemented a third algorithm, which does not use the edit distance algorithm. It finds all possible alignments. In the example of Table 2 it finds all three mutually exclusive alignments.

2.1.2 Grouping

The previous section described algorithms that align two sentences and find parts of the sentences that are

similar. The dissimilar parts of the sentences, i.e. the rest of the sentences, are considered possible constituents. Every pair of new possible constituents introduces a new non-terminal.[2]

Table 3. Learning with a partially structured sentence and an unstructured sentence

1	What does (AP57 restriction)$_{X_1}$ mean
2	What does aircraft code D8S mean
3	What does (AP57 restriction)$_{X_1}$ mean
4	What does (aircraft code D8S)$_{X_1}$ mean

At some point the system may find a constituent that was already present in one of the two sentences. This may occur when a new sentence is compared to a partially structured sentence in memory. No new type is introduced, instead the type of the new constituent will be the same type of the constituent in memory. (See Table 3 for an example.)

Table 4. Learning with two partially structured sentences

1	Explain the (meal code)$_{X_1}$
2	Explain the (restriction AP)$_{X_2}$
3	Explain the (meal code)$_{X_3}$
4	Explain the (restriction AP)$_{X_3}$

A more complex case may occur when two partially structured sentences are aligned. This happens when a new sentence that contains some structure, which was learned in a previous step, is compared to a sentence in memory. When the alignment of these two sentences yields a constituent that was already present in both sentences, the types of these constituents are then merged. All constituents of these types in memory are updated so they have the same type. This reduces the number of non-terminals in memory as can be seen in Table 4.

2.2 Selection Learning

The algorithm so far may generate constituents that overlap with other constituents. In Table 5 sentence 2 receives one structure when aligned with sentence 1 and a different structure when sentence 3 (which is the same as sentence 2) is aligned with sentence 4. The constituents in sentence 2 and 3 are overlapping.

This is solved by adding a selection method that selects constituents until no overlaps remain. (During the alignment learning phase all possible constituents are remembered, even if they overlap.) We have implemented three different methods, although other implementations may be considered. Note that only one of the methods is used at a time.

[2] In our implementation we used natural numbers to denote the different types.

Table 5. Overlapping constituents

1	*(Book Delta 128)$_X$ from Dallas to Boston*
2	*(Give me all flights)$_X$ from Dallas to Boston*
3	*Give me (all flights from Dallas to Boston)$_Y$*
4	*Give me (help on classes)$_Y$*

2.2.1 Incremental Method of Constituent Selection

The first selection method is based on the assumption that once a constituent is learned and remembered, it is correct. When the algorithm finds a possible constituent that overlaps with an older constituent, the new constituent is considered incorrect. We call this method *incr* (after incremental).

The main disadvantage of this method is that once an incorrect constituent has been learned, it will never be corrected. The incorrect constituent always remains in memory.

2.2.2 Probabilistic Methods of Constituent Selection

To solve the disadvantage of the *incr* method, two additional (probabilistic) constituent selection methods have been implemented.

The second selection method computes the probability of a constituent counting the number of times the words in the constituent have occurred as a constituent in the learned text, normalized by the total number of constituents.

$$P_{leaf}(c) = \frac{|c' \in C : yield(c') = yield(c)|}{|C|}$$

where C is the entire set of constituents. This method is called *leaf* since we count the number of times the leaves (i.e. the words) of the constituent co-occur in the corpus as a constituent.

The third method computes the probability of a constituent using the occurrences of the words in the constituent *and* its non-terminal (i.e. it is a normalised probability of *leaf*).

$$P_{branch}(c|root(c) = r) = \frac{|c' \in C : yield(c') = yield(c) \wedge root(c') = r|}{|c'' \in C : root(c'') = r|}$$

The probability is based on the root node and the terminals of the constituent, which can be seen as a

branch (of depth one) in the entire structure of the sentence, hence the name *branch*.

These two methods are probabilistic in nature. The system computes the probability of the constituent using the formula and then selects constituents with the highest probability. These methods are accomplished after alignment, since more specific information (in the form of better counts) can be found at that time.

2.2.3 Combination Probability

Two methods to determine the probability of a constituent have been described. Since more than two constituents can overlap, a combination of non-overlapping constituents has to be selected. Therefore, we need to know the probability of a combination of constituents. The probability of a combination of constituents is the product of the probabilities of the constituents as in SCFGs (cf. Booth, 1969).

Using the product of the probabilities of constituents results in a *trashing* effect, since the product of probabilities is always smaller than or equal to the separate probabilities. Instead, we use a normalised version, the geometric mean[3] (Caraballo & Charniak, 1998).

However, the geometric mean does not have a preference for richer structures. When there are two (or more) constituents that have the same probability, the constituents have the same probability as their combination and the algorithm selects one at random.

To let the system prefer more complex structure when there are more possibilities with the same probability, we implemented the *extended geometric mean*. The only difference with the (standard) geometric mean is that when there are more possibilities (single constituents or combinations of constituents) with the same probability, this system selects the one with the most constituents. To distinguish between systems that use the geometric mean and those that use the extended geometric mean, we add a + to the name of the methods that use the extended geometric mean.

Instead of computing the probabilities of all possible combinations of constituents, we have used a Viterbi (1967) style algorithm optimization to efficiently select the best combination of constituents.

3. Test Environment

In this section we will describe the systems we have tested and the metrics we used.

[3]The geometric mean of a set of constituents $c_1, \ldots, c_n$ is $P(c_1 \wedge \ldots \wedge c_n) = \sqrt[n]{\prod_{i=1}^{n} P(c_i)}$

3.1 System Variables

The ABL algorithm consists of two phases, alignment learning and selection learning. For both phases, we have discussed several implementations.

The alignment learning phase builds on the alignment algorithm. We have implemented three algorithms: *default* γ, *biased* γ and *all* alignments.

After the alignment learning phase, the selection learning phase takes place, which can be accomplished in different ways: *incr* (the first constituent is correct), *leaf* (based on the probability of the words in the constituent) and *branch* (based on the probability of the words *and* label of the constituent).

There are two ways of combining the probabilities of constituents in the probabilistic methods: *geometric mean* and *extended geometric mean*. A + is added to the systems using the extended geometric mean.

The alignment and selection methods can be combined into several ABL systems. The names of the algorithms are in the form of: *alignment:selection*, where *alignment* and *selection* represent an alignment and selection method respectively.

3.2 Metrics

To see how well the different systems perform, we use the three following metrics:

$$NCBP = \frac{\sum_i |O_i| - |Cross(O_i, T_i)|}{\sum_i |O_i|}$$

$$NCBR = \frac{\sum_i |T_i| - |Cross(T_i, O_i)|}{\sum_i |T_i|}$$

$$ZCS = \frac{\sum_i Cross(O_i, T_i) = 0}{|TEST|}$$

$Cross(U, V)$ denotes the subset of constituents from U that cross at least one constituent in V. O_i and T_i represent the constituents of a tree in the learned corpus and in $TEST$, the original corpus, respectively. (Sima'an, 1999)

NCBP stands for Non-Crossing Brackets Precision, which denotes the percentage of *learned* constituents that do not overlap with any constituents in the *original* corpus. *NCBR* is the Non-Crossing Brackets Recall and shows the percentage of constituents in the *original* corpus that do not overlap with any constituents in the *learned* corpus. Finally, *ZCS* stands for 0-Crossing Sentences and represents the percentage of *sentences* that do not have any overlapping constituents.

Table 6. Results of the ATIS corpus and OVIS corpus

	Results ATIS corpus			Results OVIS corpus		
	NCBP	NCBR	ZCS	NCBP	NCBR	ZCS
DEFAULT:INCR	82.55 (0.80)	82.98 (0.78)	17.15 (1.17)	88.69 (1.11)	83.90 (1.61)	45.13 (4.12)
BIASED:INCR	82.64 (0.76)	83.90 (0.74)	17.82 (1.01)	88.71 (0.79)	84.36 (1.10)	45.11 (3.22)
ALL:INCR	83.55 (0.63)	83.21 (0.64)	17.04 (1.19)	89.24 (1.23)	84.24 (1.82)	**46.84** (5.02)
DEFAULT:LEAF	82.20 (0.30)	82.65 (0.29)	21.05 (0.76)	85.70 (0.01)	79.96 (0.02)	30.87 (0.07)
BIASED:LEAF	81.42 (0.30)	82.75 (0.29)	21.60 (0.66)	85.32 (0.02)	79.96 (0.03)	30.87 (0.09)
ALL:LEAF	82.55 (0.31)	82.11 (0.32)	20.63 (0.70)	85.84 (0.02)	79.58 (0.03)	30.74 (0.08)
DEFAULT:LEAF+	82.31 (0.32)	83.10 (0.31)	22.02 (0.76)	85.67 (0.02)	79.95 (0.03)	30.90 (0.08)
BIASED:LEAF+	81.43 (0.32)	83.11 (0.31)	22.44 (0.70)	85.25 (0.02)	79.88 (0.03)	30.89 (0.08)
ALL:LEAF+	82.55 (0.35)	82.42 (0.35)	21.51 (0.69)	85.83 (0.02)	79.56 (0.03)	30.83 (0.08)
DEFAULT:BRANCH	86.04 (0.10)	87.11 (0.09)	29.01 (0.00)	89.39 (0.00)	84.90 (0.00)	42.05 (0.02)
BIASED:BRANCH	85.31 (0.11)	**87.14** (0.11)	**29.71** (0.00)	89.25 (0.00)	**85.04** (0.01)	42.20 (0.01)
ALL:BRANCH	**86.47** (0.08)	86.78 (0.08)	29.57 (0.00)	**89.63** (0.00)	84.76 (0.00)	41.98 (0.02)
DEFAULT:BRANCH+	86.04 (0.10)	87.10 (0.09)	29.01 (0.00)	89.39 (0.00)	84.90 (0.00)	42.04 (0.02)
BIASED:BRANCH+	85.31 (0.10)	87.13 (0.09)	**29.71** (0.00)	89.25 (0.00)	**85.04** (0.00)	42.19 (0.01)
ALL:BRANCH+	**86.47** (0.07)	86.78 (0.07)	29.57 (0.00)	**89.63** (0.00)	84.76 (0.00)	41.98 (0.02)

4. Results

Several ABL algorithms are tested on the ATIS corpus (Marcus et al., 1993) and on the OVIS corpus (Bonnema et al., 1997). The ATIS corpus from the Penn Treebank is a structured, English corpus and consists of 716 sentences containing 11,777 constituents. The OVIS corpus is a structured, Dutch corpus containing sentences on travel information. It consists of exactly 10,000 sentences. From these sentences we have selected all sentences of length larger than one, which results in 6,797 sentences containing 48,562 constituents.

The sentences of the corpora are stripped of their structure and the ABL algorithms are applied to them. The resulting structured sentences are then compared to the structures in the original corpus.

The results of applying the different systems to the ATIS corpus and the OVIS corpus can be found in Table 6. All systems have been tested ten times, since the *incr* system depends on the order of the sentences and the probabilistic systems sometimes select constituents at random. The results in the table show the mean and the standard deviation (in brackets).

4.1 Evaluation

Although we argued that the alignment methods *biased* γ and *all* solve problems of the *default* γ, this can hardly be seen when looking at the results. The main tendency is that the *all* methods generate higher precision (NCBP), with a maximum of 89.63 % on the OVIS corpus, but that the *biased* γ methods result in higher recall (NCBR) with 87.14 % on the ATIS corpus and 0-crossing sentences, 29.71 % on the ATIS corpus (on the OVIS corpus the maximum is reached with the *all* method). The *default* γ method performs worse overall. These differences, however, are slight.

The selection learning methods have a larger impact on the differences in the generated corpora. The *incr* systems perform quite well considering the fact that they cannot recover from incorrect constituents, with a precision and recall of roughly 83 %. The order of the sentences however is quite important, since the standard deviation of the *incr* systems is quite large (especially with the ZCS, reaching 1.19 %).

We expected the probabilistic methods to perform better, but the *leaf* systems perform slightly worse. The ZCS, however, is somewhat better, resulting in 22.44 % for the leaf+ method. Furthermore, the standard deviations of the *leaf* systems (and of the *branch* systems) are close to 0 %. The statistical methods generate more precise results.

The *branch* systems clearly outperform all other systems. Using more specific statistics generate better results.

The systems using the extended geometric mean result in slightly better results on the *leaf* system, but when larger corpora are used, this difference disappears completely.

Although the results of the ATIS corpus and OVIS corpus differ, the conclusions that can be reached are similar.

Table 7. Recursion learned in the ATIS corpus

learned	*What is the (name of the (airport in Boston)$_{18}$)$_{18}$*
original	*What is (the name of (the airport in Boston)$_{NP}$)$_{NP}$*
learned	*Explain classes QW and (QX and (Y)$_{52}$)$_{52}$*
original	*Explain classes ((QW)$_{NP}$ and (QX)$_{NP}$ and (Y)$_{NP}$)$_{NP}$*

4.2 Recursion

All ABL systems learn recursion on the ATIS and OVIS corpora. Two example sentences from the ATIS corpus with the original and learned structure can be found in Table 7. The sentences in the example are stripped of all but the interesting constituents to make it easier to see where the recursion occurs.

The recursion in the first sentence is not entirely the same. The ABL algorithm finds constituents of some sort of noun phrase, while the constituents in the ATIS corpus show recursive noun phrases. Likewise in the second sentence, the ABL algorithm finds a recursive noun phrase while the structure in the ATIS corpus is similar.

5. Previous Work

Existing grammar learning methods can be grouped (like other learning methods) into supervised and unsupervised methods. Unsupervised methods only use plain (or pre-tagged) sentences, while supervised methods are first initialised with structured sentences.

In practice, supervised methods generate better results, since they can adapt their output to the structured examples from the initialisation phase, whereas unsupervised methods do not have any idea what the output should look like. Although unsupervised methods perform worse than supervised methods, unsupervised methods are necessary for the time-consuming and costly creation of corpora for which no corpus nor grammar yet exists.

There have been several different approaches to learn syntactic structures. We will give a short overview here.

Memory based learning (MBL) keeps track of the possible contexts and assigns word types based on that information (Daelemans, 1995). Magerman and Marcus (1990) describe a method that finds constituent boundaries using mutual information values of the part of speech n-grams within a sentence and Redington et al. (1998) present a method that bootstraps syntactic categories using distributional information.

Algorithms that use the minimum description length (MDL) principle build grammars that describe the input sentences using the minimal number of bits. This idea stems from the information theory. Examples of these systems can be found in Grünwald (1994) and de Marcken (1996).

The system by Wolff (1982) performs a heuristic search while creating and merging symbols directed by an evaluation function. Similarly, Cook et al. (1976) describe an algorithm that uses a cost function that can be used to direct search for a grammar. Stolcke and Omohundro (1994) describe a more recent grammar induction method that merges elements of models using a Bayesian framework. Chen (1995) presents a Bayesian grammar induction method, which is followed by a post-pass using the inside-outside algorithm (Baker, 1979; Lari & Young, 1990), while Pereira and Schabes (1992) apply the inside-outside algorithm to a partially structured corpus.

The supervised system described by Brill (1993) takes a completely different approach. It tries to find transformations that improve a naive parse, effectively reducing errors.

The two phases of ABL are closely related to some previous work. The alignment learning phase is effectively a compression technique comparable to MDL or Bayesian grammar induction methods. However, ABL remembers all possible constituents, effectively building a search space. The selection learning phase searches this space, directed by a probabilistic evaluation function.

It is difficult to compare the results of the ABL system against other systems, since different corpora or metrics are used. The system described by Pereira and Schabes (1992) comes reasonably close to ours. That system learns structure on plain sentences from the ATIS corpus resulting in 37.35 % precision, while the *unsupervised* ABL significantly outperforms this method, reaching 86.47 % precision. Only their *supervised* version results in a slightly higher precision of 90.36 %.

A system that simply builds right branching structures results in 82.70 % precision and 92.91 % recall on the

ATIS corpus, where ABL got 86.47 % and 87.14 %. These good results could be expected, since English is a right branching language; a left branching system performed much worse (32.60 % precision and 76.82 % recall. On a Japanese (a left branching language) corpus, right branching would not do very well. Since ABL does not have a preference for direction built in, we expect ABL to perform similarly on a Japanese corpus compared to the ATIS corpus.

6. Discussion and Future Extensions

We will discuss several problems of ABL and suggest possible solutions to these problems.

6.1 Wrong Syntactic Type

There are cases in which the implication "if two parts of sentences can be replaced, they are constituents of the same type", we use in this system, does not hold. Consider the sentences in Table 8. When applying the ABL learning algorithm to these sentences, it will determine that *morning* and *nonstop* are of the same type. However, in the ATIS corpus, *morning* is tagged as an *NN* (a noun) and *nonstop* is a *JJ* (an adjective).

Table 8. Wrong syntactic type

Show me the (morning)$_X$ flights
Show me the (nonstop)$_X$ flights

The constituent *morning* can also be used as a noun in other contexts, while *nonstop* never will. This information can be found by looking at the distribution of the contexts of constituents in the rest of the corpus. Based on that information a correct non-terminal assignment can be made.

6.2 Weakening Exact Match

Aligning two dissimilar sentences yields no structure. However, if we weaken the exact match between words in the alignment phase, it *is* possible to learn structure even with dissimilar sentences.

Instead of linking exactly matching words, the algorithm should match words that are equivalent. One way of implementing this is by using *equivalence classes*. With equivalence classes, words that are closely related are grouped together. (Redington et al. (1998) describe an unsupervised way of finding equivalence classes.)

Words that are in the same equivalence class are said to be sufficiently equivalent and may be linked. Now sentences that do not have words in common, but do have words from the same equivalence class in common, can be used to learn structure.

When using equivalence classes, more constituents are learned since more terminals in constituents may be seen as similar (according to the equivalence classes). This results in structures containing more possible constituents from which the selection phase may choose.

6.3 Alternative Statistics

At the moment we have tested two different ways of computing the probability of a bracket: *leaf* and *branch*. Of course, other systems can be implemented. One interesting possibility takes a DOP-like approach (Bod, 1998), which takes into account the inner structure of the constituents. As can be seen in the results, the system that uses more specific statistics performs better.

7. Conclusion

We have introduced a new grammar learning algorithm based on aligning plain sentences; neither pre-labelled or bracketed nor pre-tagged sentences are used. It aligns sentences to find dissimilarities between sentences. The alignments are not limited to window-size, instead arbitrarily large contexts are used. The dissimilarities are used to find all possible constituents from which the algorithm selects the most probable ones afterwards.

Three different alignment methods and five different selection methods have been implemented. The instances of the algorithm have been applied to two corpora of different size, the ATIS corpus (716 sentences) and the OVIS corpus (6,797 sentences), generating promising numerical results. Since these corpora are still relatively small, we plan to apply the algorithm to larger corpora.

The results showed that the different selection methods have a larger impact than the different alignment methods. The selection method that uses the most specific statistics performs best. Furthermore, the system has the ability to learn recursion.

Acknowledgements

The author would like to thank Rens Bod and Mila Groot for their suggestions and comments on this paper, Lo van den Berg for his help generating the results and three anonymous reviewers for their useful comments on an earlier draft.

References

Baker, J. K. (1979). Trainable grammars for speech recognition. *Speech Communication Papers for the Ninety-seventh Meeting of the Acoustical Society of America* (pp. 547–550).

Bod, R. (1998). *Beyond grammar — an experience-based theory of language*. Stanford, CA: CSLI Publications.

Bonnema, R., Bod, R., & Scha, R. (1997). A DOP model for semantic interpretation. *Proceedings of the Association for Computational Linguistics/European Chapter of the Association for Computational Linguistics, Madrid* (pp. 159–167). Sommerset, NJ: Association for Computational Linguistics.

Booth, T. (1969). Probabilistic representation of formal languages. *Conference Record of 1969 Tenth Annual Symposium on Switching and Automata Theory* (pp. 74–81).

Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. *Proceedings of the Association for Computational Linguistics* (pp. 259–265).

Caraballo, S. A., & Charniak, E. (1998). New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, *24*, 275–298.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. *Proceedings of the Association for Computational Linguistics* (pp. 228–235).

Cook, C. M., Rosenfeld, A., & Aronson, A. R. (1976). Grammatical inference by hill climbing. *Informational Sciences*, *10*, 59–80.

Daelemans, W. (1995). Memory-based lexical acquisition and processing. In P. Steffens (Ed.), *Machine translation and the lexicon*, vol. 898 of *Lecture Notes in Artificial Intelligence*, 85–98. Berlin: Springer Verlag.

de Marcken, C. G. (1996). *Unsupervised language acquisition*. Doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Grünwald, P. (1994). A minimum description length approach to grammar inference. In G. Scheler, S. Wernter and E. Riloff (Eds.), *Connectionist, statistical and symbolic approaches to learning for natural language*, vol. 1004 of *Lecture Notes in AI*, 203–216. Berlin: Springer Verlag.

Harris, Z. (1951). *Methods in structural linguistics*. Chicago, IL: University of Chicago Press.

Lari, K., & Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, *4*, 35–56.

Magerman, D., & Marcus, M. (1990). Parsing natural language using mutual information statistics. *Proceedings of the National Conference on Artificial Intelligence* (pp. 984–989). Cambridge, MA: MIT Press.

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a large annotated corpus of english: the Penn treebank. *Computational Linguistics*, *19*, 313–330.

Pereira, F., & Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. *Proceedings of the Association for Computational Linguistics* (pp. 128–135). Newark, Delaware.

Redington, M., Chater, N., & Finch, S. (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, *22*, 425–469.

Sima'an, K. (1999). *Learning efficient disambiguation*. Doctoral dissertation, Institute for Language, Logic and Computation, Universiteit Utrecht.

Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by bayesain model merging. *Second International Conference on Grammar Inference and Applications* (pp. 106–118). Berlin: Springer Verlag. Alicante, Spain.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, *13*, 260–269.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, *21*, 168–173.

Wolff, J. G. (1982). Language acquisition, data compression and generalization. *Language & Communication*, *2*, 57–89.