ABL: Alignment-Based Learning

Menno van Zaanen
School of Computer Studies
University of Leeds
LS2 9JT Leeds
UK
menno@scs.leeds.ac.uk

Abstract

This paper introduces a new type of grammar learning algorithm, inspired by string edit distance (Wagner and Fischer, 1974). The algorithm takes a corpus of flat sentences as input and returns a corpus of labelled, bracketed sentences. The method works on pairs of unstructured sentences that have one or more words in common. When two sentences are divided into parts that are the same in both sentences and parts that are different, this information is used to find parts that are interchangeable. These parts are taken as possible constituents of the same type. After this alignment learning step, the selection learning step selects the most probable constituents from all possible constituents.

This method was used to bootstrap structure on the ATIS corpus (Marcus et al., 1993) and on the OVIS[1] corpus (Bonnema et al., 1997). While the results are encouraging (we obtained up to 89.25 % non-crossing brackets precision), this paper will point out some of the shortcomings of our approach and will suggest possible solutions.

1 Introduction

Unsupervised learning of syntactic structure is one of the hardest problems in NLP. Although people are adept at learning grammatical structure, it is difficult to model this process and therefore it is hard to make a computer learn structure.

We do not claim that the algorithm described here models the human process of language learning. Instead, the algorithm should, given unstructured sentences, find the best structure. This means that the algorithm should assign structure to sentences which are similar to the structure people would give to sentences, but not necessarily in the same time or space restrictions.

The algorithm consists of two phases. The first phase is a constituent generator, which generates a motivated set of possible constituents by aligning sentences. The second phase restricts this set by selecting the best constituents from the set.

The rest of this paper is organized as follows. Firstly, we will start by describing previous work in machine learning of language structure and then we will give a description of the ABL algorithm. Next, some results of applying the ABL algorithm to different corpora will be given, followed by a discussion of the algorithm and future research.

2 Previous Work

Learning methods can be grouped into supervised and unsupervised methods. Supervised methods are initialised with structured input (i.e. structured sentences for grammar learning methods), while unsupervised methods learn by using unstructured data only.

In practice, supervised methods outperform unsupervised methods, since they can adapt their output based on the structured examples in the initialisation phase whereas unsupervised methods cannot. However, it is worthwhile to investigate unsupervised grammar learning methods, since "the costs of annotation are prohibitively time and expertise intensive, and the resulting corpora may be too susceptible to restriction to a particular domain, application, or genre". (Kehler and Stolcke, 1999)

There have been several approaches to the *unsupervised* learning of syntactic structures. We will give a short overview here.

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

Memory based learning (MBL) keeps track of possible contexts and assigns word types based on that information (Daelemans, 1995). Redington et al. (1998) present a method that bootstraps syntactic categories using distributional information and 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.

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 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. Chen (1995) presents a Bayesian grammar induction method, which is followed by a post-pass using the inside-outside algorithm (Baker, 1979; Lari and Young, 1990).

Most work described here cannot learn complex structures such as recursion, while other systems only use limited context to find constituents. However, the two phases in 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. ABL remembers all possible constituents, building a search space. The selection learning phase searches this space, directed by a probabilistic evaluation function.

3 Algorithm

We will describe an algorithm that learns structure using a corpus of plain (unstructured) sentences. It does not need a structured training set to initialize, all structural information is gathered from the unstructured sentences.

The output of the algorithm is a labelled, bracketed version of the input corpus. Although the algorithm does not generate a (context-free) grammar, it is trivial to deduce one from the structured corpus.

The algorithm builds on Harris's idea (1951) that states that *constituents of the same type can be replaced by each other*. Consider the sentences as shown in figure 1.[2] The constituents *a family fare* and *the payload of an African Swallow* both have the same syntactic type (they are both NPs), so they can be replaced by each other. This means that when the constituent in the first sentence is replaced by the constituent in the second sentence, the result is a valid sentence in the language; it is the second sentence.

What is a family fare
What is the payload of an African Swallow

What is (a family fare)$_X$
What is (the payload of an African Swallow)$_X$

Figure 1: Example bootstrapping structure

```
For each sentence $s_1$ in the corpus:
    For every other sentence $s_2$ in the corpus:
        Align $s_1$ to $s_2$
        Find the identical and distinct parts
            between $s_1$ and $s_2$
        Assign non-terminals to the constituents
            (i.e. distinct parts of $s_1$ and $s_2$)
```

Figure 2: Alignment learning algorithm

The main goal of the algorithm is to establish that *a family fare* and *the payload of an African Swallow* are constituents and have the same type. This is done by reversing Harris's idea: *if (a group of) words can be replaced by each other, they are constituents and have the same type.* So the algorithm now has to find groups of words that can be replaced by each other and after replacement still generate valid sentences.

The algorithm consists of two steps:

1. Alignment Learning

2. Selection Learning

3.1 Alignment Learning

The model learns by comparing all sentences in the input corpus to each other in pairs. An overview of the algorithm can be found in figure 2.

Aligning sentences results in "linking" identical words in the sentences. Adjacent linked words are then grouped. This process reveals

[2] All sentences in the examples can be found in the ATIS corpus.

from *()*$_1$ *San Francisco* *(to Dallas)*$_2$
from *(Dallas to)*$_1$ *San Francisco* *()*$_2$

from *(San Francisco to)*$_1$ *Dallas* *()*$_2$
from *()*$_1$ *Dallas* *(to San Francisco)*$_2$

from *(San Francisco)*$_1$ *to* *(Dallas)*$_2$
from *(Dallas)*$_1$ *to* *(San Francisco)*$_2$

Figure 3: Ambiguous alignments

the groups of identical words, but it also uncovers the groups of distinct words in the sentences. In figure 1 *What is* is the identical part of the sentences and *a family fare* and *the payload of an African Swallow* are the distinct parts. The distinct parts are interchangeable, so they are determined to be constituents of the same type.

We will now explain the steps in the alignment learning phase in more detail.

3.1.1 Edit Distance

To find the identical word groups in the sentences, we use the edit distance algorithm by Wagner and Fischer (1974), which finds the minimum number of edit operations (insertion, deletion and substitution) to change one sentence into the other. Identical words in the sentences can be found at places where no edit operation was applied.

The instantiation of the algorithm that finds the longest common subsequence in two sentences sometimes "links" words that are too far apart. In figure 3 when, besides the occurrences of *from*, the occurrences of *San Francisco* or *Dallas* are linked, this results in unintended constituents. We would rather have the model linking *to*, resulting in a structure with the noun phrases grouped with the same type correctly.

Linking *San Francisco* or *Dallas* results in constituents that vary widely in size. This stems from the large distance between the linked words in the first sentence and in the second sentence. This type of alignment can be ruled out by biasing the cost function using distances between words.

3.1.2 Grouping

An edit distance algorithm links identical words in two sentences. When adjacent words are linked in both sentences, they can be grouped. A group like this is a part of a sentence that can also be found in the other sentence. (In figure 1, *What is* is a group like this.)

The rest of the sentences can also be grouped. The words in these groups are words that are distinct in the two sentences. When all of these groups from sentence one would be replaced by the respective groups of sentence two, sentence two is generated. (*a family fare* and *the payload of an African Swallow* are of this type of group in figure 1.) Each pair of these distinct groups consists of possible constituents of the same type.[3]

As can be seen in figure 3, it is possible that empty groups can be learned.

3.1.3 Existing Constituents

At some point it may be possible that the model learns a constituent that was already stored. This may happen when a new sentence is compared to a sentence in the partially structured corpus. In this case, no new type is introduced, but the constituent in the new sentence gets the same type of the constituent in the sentence in the partially structured corpus.

It may even be the case that a partially structured sentence is compared to another partially structured sentence. This occurs when a sentence that contains some structure, which was learned by comparing to a sentence in the partially structured corpus, is compared to another (partially structured) sentence. When the comparison of these two sentences yields a constituent that was already present in both sentences, the types of these constituents are merged. All constituents of these types are updated, so they have the same type.

By merging types of constituents we make the assumption that constituents in a certain context can only have one type. In section 5.2 we discuss the implications of this assumption and propose an alternative approach.

3.2 Selection Learning

The first step in the algorithm may at some point generate constituents that overlap with other constituents. In figure 4 *Give me all flights from Dallas to Boston* receives two overlapping structures. One constituent is learned

[3] Since the algorithm does not know any (linguistic) names for the types, the algorithm chooses natural numbers to denote different types.

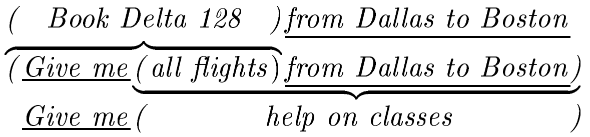
(Book Delta 128) from Dallas to Boston
(Give me (all flights) from Dallas to Boston)
Give me (help on classes)

Figure 4: Overlapping constituents

by comparing against *Book Delta 128 from Dallas to Boston* and the other (overlapping) constituent is found by aligning with *Give me help on classes.*

The solution to this problem has to do with selecting the correct constituents (or at least the better constituents) out of the possible constituents. Selecting constituents can be done in several different ways.

ABL:incr Assume that the first constituent learned is the correct one. This means that when a new constituent overlaps with older constituents, it can be ignored (i.e. they are not stored in the corpus).

ABL:leaf The model computes the probability of a constituent counting the number of times the particular words of the constituent have occurred in the learned text as a constituent, 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.

ABL:branch In addition to the words of the sentence delimited by the constituent, the model computes the probability based on the part of the sentence delimited by the words of the constituent *and* its non-terminal (i.e. a normalised probability of ABL: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 first method is non-probabilistic and may be applied every time a constituent is found that overlaps with a known constituent (i.e. while learning).

The two other methods are probabilistic. The model computes the probability of the constituents and then uses that probability to select constituents with the highest probability. These methods are applied after the alignment learning phase, since more specific information (in the form of better counts) can be found at that time.

In section 4 we will evaluate all three methods on the ATIS and OVIS corpus.

3.2.1 Viterbi

Since more than just two constituents can overlap, all possible combinations of overlapping constituents should be considered when computing the best combination of constituents, which is the product of the probabilities of the separate constituents as in SCFGs (cf. (Booth, 1969)). A Viterbi style algorithm optimization (1967) is used to efficiently select the best combination of constituents.

When computing the probability of a combination of constituents, multiplying the separate probabilities of the constituents biases towards a low number of constituents. Therefore, we compute the probability of a set of constituents using a normalized version, the geometric mean[4], rather than its product. (Caraballo and Charniak, 1998)

4 Results

The three different ABL algorithms and two baseline systems have been tested on the ATIS and OVIS corpora.

The ATIS corpus from the Penn Treebank consists of 716 sentences containing 11,777 constituents. The larger OVIS corpus is a Dutch corpus containing sentences on travel information. It consists of exactly 10,000 sentences. We have removed all sentences containing only one word, resulting in a corpus of 6,797 sentences and 48,562 constituents.

The sentences of the corpora are stripped of their structures. These plain sentences are used in the learning algorithms and the resulting structure is compared to the structure of the original corpus.

All ABL methods are tested ten times. The ABL:incr method is applied to random orders of the input corpus. The probabilistic ABL methods select constituents at random when different combinations of constituents have the same probability. The results in table 1 show the

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

	ATIS			OVIS		
	NCBP	NCBR	ZCS	NCBP	NCBR	ZCS
LEFT	32.60	76.82	1.12	51.23	73.17	25.22
RIGHT	82.70	92.91	38.83	75.85	86.66	48.08
ABL:INCR	83.24 (1.17)	87.21 (0.67)	18.56 (2.32)	88.71 (0.79)	84.36 (1.10)	45.11 (3.22)
ABL:LEAF	81.42 (0.11)	86.27 (0.06)	21.63 (0.50)	85.32 (0.02)	79.96 (0.03)	30.87 (0.09)
ABL:BRANCH	85.31 (0.01)	89.31 (0.01)	29.75 (0.00)	89.25 (0.00)	85.04 (0.00)	42.20 (0.01)

Table 1: Results of the ATIS and OVIS corpora

mean and standard deviations (between brackets).

The two baseline systems, left and right, only build left and right branching trees respectively.

Three metrics have been computed. *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 Zero-Crossing Sentences and represents the percentage of *sentences* that do not have any overlapping constituents.

4.1 Evaluation

The *incr* model performs quite well considering the fact that it cannot recover from incorrect constituents, with a precision and recall of over 80 %. The order of the sentences however is quite important, since the standard deviation of the *incr* model is quite high (especially with the ZCS, reaching 3.22 % on the OVIS corpus).

We expected the probabilistic methods to perform better, but the *leaf* model performs slightly worse. The ZCS, however, is somewhat better, resulting in 21.63 % on the ATIS corpus. Furthermore, the standard deviations of the *leaf* model (and of the *branch* model) are close to 0 %. The statistical methods generate more precise results.

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

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

4.2 ABL Compared to Other Methods

It is difficult to compare the results of the ABL model against other methods, since often different corpora or metrics are used. The methods described by Pereira and Schabes (1992) comes reasonably close to ours. The *unsupervised* method learns structure on plain sentences from the ATIS corpus resulting in 37.35 % precision, while the *unsupervised* ABL significantly outperforms this method, reaching 85.31 % precision. Only their *supervised* version results in a slightly higher precision of 90.36 %.

The system that simply builds right branching structures results in 82.70 % precision and 92.91 % recall on the ATIS corpus, where ABL got 85.31 % and 89.31 %. This was expected, since English is a right branching language; a left branching system performed much worse (32.60 % precision and 76.82 % recall). Conversely, right branching would not do very well on a Japanese corpus (a left branching language). Since ABL does not have a preference for direction built in, we expect ABL to perform similarly on a Japanese corpus.

5 Discussion and Future Extensions

5.1 Recursion

All ABL methods described here can learn recursive structures and have been found when applying ABL to the ATIS and OVIS corpus. As can be seen in figure 5, the learned recursive structure is similar to the original structure. Some structure has been removed to make it easier to see where the recursion occurs.

Roughly, recursive structures are built in two steps. First, the algorithm generates the structure with different non-terminals. Then, the two non-terminals are merged as described in section 3.1.3. The merging of the non-terminals may occur anywhere in the corpus, since *all* merged non-terminals are updated.

learned	*Please explain the (field FLT DAY in the (table)*$_{13}$*)*$_{13}$
original	*Please explain (the field FLT DAY in (the table)*$_{NP}$*)*$_{NP}$
learned	*Explain classes QW and (QX and (Y)*$_{52}$*)*$_{52}$
original	*Explain classes ((QW)*$_{NP}$ *and (QX)*$_{NP}$ *and (Y)*$_{NP}$*)*$_{NP}$

Figure 5: Recursive structures learned in the ATIS corpus

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

Figure 6: Wrong syntactic type

5.2 Wrong Syntactic Type

In section 3.1.3 we made the assumption that a constituent in a certain context can only have one type. This assumption introduces some problems.

The sentence *John likes visiting relatives* illustrates such a problem. The constituent *visiting relatives* can be a noun phrase or a verb phrase.

Another problem is illustrated in figure 6. When applying the ABL learning algorithm to these sentences, it will determine that *morning* and *nonstop* are of the same type. Unfortunately, *morning* is a noun, while *nonstop* is an adverb.[5]

A future extension will not only look at the type of the constituents, but also at the context of the constituents. In the example, the constituent *morning* may also take the place of a subject position in other sentences, but the constituent *nonstop* never will. This information can be used to determine when to merge constituent types, effectively loosening the assumption.

5.3 Weakening Exact Match

When the ABL algorithms try to learn with two completely distinct sentences, nothing can be learned. If we weaken the exact match between words in the alignment step of the algorithm, it is possible to learn structure even with distinct sentences.

Instead of linking exactly matching words, the algorithm should match words that are equivalent. An obvious way of implementing this is by making use of *equivalence classes.* (See for example (Redington et al., 1998).) The idea behind equivalence classes is that words which are closely related are grouped together.

A big advantage of equivalence classes is that they can be learned in an unsupervised way, so the resulting algorithm remains unsupervised.

Words that are in the same equivalence class are said to be sufficiently equivalent, so the alignment algorithm may assume they are similar and may thus link them. Now sentences that do not have words in common, but do have words in the same equivalence class in common, can be used to learn structure.

When using equivalence classes, more constituents are learned and more terminals in constituents may be seen as similar (according to the equivalence classes). This results in a much richer structured corpus.

5.4 Alternative Statistics

At the moment we have tested two different ways of computing the probability of a constituent: *ABL:leaf* which computes the probability of the occurrence of the terminals in a constituent, and *ABL:branch* which computes the probability of the occurrence of the terminals together with the root non-terminal in a constituent, based on the learned corpus.

Of course, other models can be implemented. One interesting possibility takes a DOP-like approach (Bod, 1998), which also takes into account the inner structure of the constituents.

6 Conclusion

We have introduced a new grammar learning algorithm based on comparing and aligning plain sentences; neither pre-labelled or bracketed sentences, nor pre-tagged sentences are used. It uses distinctions between sentences to find possible constituents and afterwards selects the most probable ones. The output of the algorithm is a structured version of the corpus.

By taking entire sentences into account, the context used by the model is not limited by window size, instead arbitrarily large contexts are

[5] Harris's implication does hold in these sentences. *nonstop* can also be replaced by for example *cheap* (another adverb) and *morning* can be replaced by *evening* (another noun).

used. Furthermore, the model has the ability to learn recursion.

Three different 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 results. Although the OVIS corpus is almost ten times the size of the ATIS corpus, these corpora describe a small conceptual domain. We plan to apply the algorithms to larger domain corpora in the near future.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. In J. J. Wolf and D. H. Klatt, editors, *Speech Communication Papers for the Ninety-seventh Meeting of the Acoustical Society of America*, pages 547–550.

Rens Bod. 1998. *Beyond Grammar — An Experience-Based Theory of Language.* Stanford, CA: CSLI Publications.

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

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

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Stanley F. Chen. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the Association for Computational Linguistics*, pages 228–235.

Walter Daelemans. 1995. Memory-based lexical acquisition and processing. In P. Steffens, editor, *Machine Translation and the Lexicon*, volume 898 of *Lecture Notes in Artificial Intelligence*, pages 85–98. Berlin: Springer Verlag.

Carl G. de Marcken. 1996. *Unsupervised Language Acquisition.* Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, sep.

Peter Grünwald. 1994. A minimum description length approach to grammar inference. In G. Scheler, S. Wernter, and E. Riloff, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language*, volume 1004 of *Lecture Notes in AI*, pages 203–216. Berlin: Springer Verlag.

Zellig Harris. 1951. *Methods in Structural Linguistics.* Chicago, IL: University of Chicago Press.

Andrew Kehler and Andreas Stolcke. 1999. Preface. In A. Kehler and A. Stolcke, editors, *Unsupervised Learning in Natural Language Processing.* Association for Computational Linguistics. Proceedings of the workshop.

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

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

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

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

Martin Redington, Nick Chater, and Steven Finch. 1998. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469.

A. Viterbi. 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.

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

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