Learning Structure using Alignment Based Learning

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

Abstract

In this paper we will introduce a new algorithm that learns structure in the form of bracketed sentences using unstructured, untagged sentences. The algorithm is based on the idea of Harris [Har51] stating that constituents of the same type can be replaced within a sentence. The algorithm consists of two phases. In the first phase, called *alignment learning*, the algorithm learns possible constituents. In the second phase, the algorithm selects the best constituents out of the possible constituents that were learned. This phase is called *bracket selection*. The learning phase finds possible constituents by comparing all sentences in memory in pairs. Two sentences from memory are aligned to each other using the Edit Distance algorithm [WF74]. This algorithm finds the parts that are similar or dissimilar in both sentences. The dissimilar parts of the sentences are assumed to be possible constituents. All of these possible constituents are stored. At some point the algorithm might have learned overlapping constituents. The bracket selection phase selects constituents that do not overlap. Three systems, two probabilistic and one non-probabilistic, have been implemented. We applied the algorithm to the ATIS corpus and found a non-crossing brackets precision of 86.04%, non-crossing brackets recall of 89.39% and a percentage of zero-crossing sentences of 29.05%.

1 Introduction

This paper introduces a new type of learning algorithm, which is based on finding similar parts in sentences by aligning them. It is applied to a corpus of unstructured sentences, yielding a grammar in the form of tree structures.

The alignment part of the algorithm aligns sentences in pairs. This results in a partition of the sentences with parts similar in both sentences and parts dissimilar in both sentences. The dissimilar parts of the sentences are deemed to be interchangeable, since interchanging all dissimilar parts in one sentence, results in the other sentence. The dissimilar parts are remembered as possible constituents.

The alignment part of the algorithm generates a number of possibly overlapping or ambiguous constituents. The selection phase in the algorithm selects the most probable non-overlapping constituents from all generated possible constituents.

The rest of this paper is organized as follows. Firstly, we will start by describing previous work in language learning and then we will give a description of the ABL algorithm. Some results of the algorithm applied to the ATIS corpus will be given, followed by a discussion of the algorithm and future research.

2 Previous work in learning structure

This section contains a brief overview of previous work in grammar acquisition systems. We take a closer look at several different methods and investigate the relation between ABL and these methods.

2.1 Language learning methods

Language learning algorithms can be roughly divided into two groups, *supervised* and *unsupervised* learning algorithms, based on the type of information that is used. All learning algorithms use a *teacher* that gives examples of (unstructured) sentences in the language. In addition, some algorithms use a *critic*. A critic may be asked if a certain sentence (possibly including structure) is a well-formed sentence in the language.[1] Supervised language learning methods use a teacher and a critic, whereas the unsupervised methods only use a teacher. [Pow97]

Supervised language learning methods typically generate better results. These methods receive knowledge of the structure of the language (by initialisation or querying a critic) and therefore can tune their output using this knowledge. In contrast, unsupervised language learning methods do not know at all what the output should look like. The last two systems we will describe are supervised. The other systems are all unsupervised, like the ABL algorithm.

There have been many different approaches to language learning. We will describe some of these approaches here briefly. This section is merely meant to relate the ABL system to other language learning methods; we do not claim to give a complete overview of this field.

Most recent language learning methods are based on some probabilistic or counting theories. Examples of these methods are Memory-Based Learning (MBL) which keeps track of the distribution of contexts of words and assigns word types based on that information [Dae95]. [MM90] describes a system that can find constituent boundaries using mutual information of n-grams within sentences. In [FC92] and [RCF98] models are proposed that use distributional information to acquire syntactical categories.

Systems based on the Minimum Description Length (MDL) principle compress the input corpus by minimizing the description length needed to express the corpus. The compression results in a grammar that can describe the corpus. Examples of these systems can be found in [Grü94] and [dM96].

Pereira and Schabes [PS92] describe a system that uses a partially bracketed corpus to infer parameters of a Stochastic Context-Free Grammar (SCFG) using inside-outside reestimation.

The final system we mention here is Transformation-Based Learning [Bri93]. This system differs from other systems in that it is a non-probabilistic system that learns transformations to improve a naive parse (for example a right branching parse).

2.2 ABL in relation to other methods

ABL consists of two distinct phases, alignment learning and bracket selection. Both phases can be roughly compared to different language learning methods.

The alignment learning phase can be seen as compressing the corpus. Similar techniques can be found in systems that are based on the MDL principle [Grü94]. MDL systems compress the corpus by looking for the minimal description of the corpus. The MDL principle results in grouping reoccurring parts of sentences yielding a reduction of the corpus. The alignment learning phase finds constituents based on the idea of interchangeability, which effectively compresses the corpus.

The bracket selection phase selects constituents based on the probability of the possible constituents. This is in some way similar to systems that use distributional information to select the most probable syntactic types as in [FC92] or [RCF98]. On the other hand, ABL assigns a probability to the different constituents, which can be seen as a SCFG.

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 initialise, all information is gathered from the unstructured

[1] When an algorithm uses a treebank or structured corpus to initialise, it is said to use a teacher and a critic. The structure of the sentences in the corpus can be seen as the critic.

What is a dual carrier
What is the payload of an African Swallow

What is (a dual carrier)$_{X_1}$
What is (the payload of an African Swallow)$_{X_1}$

Figure 1: Example bootstrapping structure

(Book Delta 128)$_X$ *from Dallas to Boston*
(Give me all flights)$_X$ *from Dallas to Boston*

Give me (all flights from Dallas to Boston)$_Y$
Give me (help on classes)$_Y$

Figure 2: Overlapping constituents

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

The algorithm builds on Harris's idea [Har51] which states that *constituents of the same type can be replaced.* Consider the sentences as shown in figure 1.[2] The constituents *a dual carrier* and *the payload of an African Swallow* both have the same syntactic type (they are both noun phrases), so they can be replaced. This means that when *a dual carrier* in the first sentence is replaced by *the payload of an African Swallow*, the result is the second sentence (which is also a well-formed sentence).

The main goal of the algorithm is to establish that *a dual carrier* 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, they are constituents and have the same type.* So the algorithm now has to find word groups that can be replaced (as in figure 1) and after replacement still generate well-formed sentences.

The algorithm consists of two phases: *alignment learning* and *bracket selection.* Both phases will be described in more detail in the next sections.

3.1 Alignment learning

The system learns by comparing all sentences in memory to each other in pairs. Sentences that have words in common are used to learn structure. Similar words in two sentences are "linked". Adjacent linked words are then grouped. This process reveals the groups of similar words, but also uncovers the groups of dissimilar words in the sentences. In figure 1 *What is* is a group of similar words and *a dual carrier* and *the payload of an African Swallow* are dissimilar word groups. The dissimilar groups are the ones that are interchangeable, so they are determined to be constituents of the same type. In the next three sections we describe how the dissimilar groups are found and stored.

3.1.1 Edit distance

We use the edit distance algorithm by Wagner and Fischer [WF74] to find the similar word groups in the sentences. The algorithm finds the minimum number of edit operations (insertion, deletion and substitution) to change one sentence into the other.

The algorithm builds a matrix in which the words in sentence s_1 index the rows and the words in sentence s_2 index the columns. The value in the matrix at row r and column c denotes the minimum edit cost needed to change the first r words in one sentence to the first c words in the other sentence. This matrix is built incrementally.

The minimum edit cost of the two sentences can now be found at (l_1, l_2) where $l_1 = |s_1|$ and $l_2 = |s_2|$. There is a path in the matrix from $(0, 0)$ to (l_1, l_2) with the minimum edit cost as its maximum value. This path describes how s_1 can be changed into s_2.

Where edit operations were necessary on the path in the matrix, the minimum edit cost was increased. On places where the minimum edit cost did *not* increase, no edit operations were necessary, so at these points similar words in both sentences can be found. These are exactly the words we are looking for.

[2] All example sentences in this paper are taken from the ATIS corpus.

3.1.2 Grouping

An edit distance algorithm links similar 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* would be a group like this.)

The rest of the sentences can also be grouped. The words in these groups are words that are dissimilar in the two sentences. When all of these groups from one sentence are replaced by the respective groups of the other sentence, this results in the latter sentence. (*a dual carrier* and *the payload of an African Swallow* are of this type of group.) Each pair of these dissimilar groups consists of possible constituents of the same type.[3] It is possible that empty groups are found.

3.1.3 Existing constituents

At some point it may be possible that the system learns a constituent that was already stored in memory. This may happen when a new sentence is compared to a partially structured sentence in memory. 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 memory.

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 memory, is compared to another (partially structured) sentence in memory. When the comparison of these two sentences yields a constituent that was already present in both sentences, the types of these constituents are then said to be the same. All constituents of these types in memory are updated, so they have the same type.

3.2 Bracket selection

The first phase in the algorithm may at some point generate constituents that overlap with other constituents. In figure 2 *Give me all flights from Dallas to Boston* receives two overlapping structures. Comparing the first sentence with the second yields the structure in the second sentence. The third and fourth sentence generate the structure in the third sentence. The structure of the second sentence overlaps with the structure of the third sentence.

To solve this problem, we select constituents until there are no more overlaps. Selecting constituents can be done in several different ways.[4]

ABL:1 Assume that the first constituent learned is the correct one. This means that when new constituents overlap with older constituents, they can be ignored (i.e. they are not stored).

ABL:terms The system selects constituents based on the probability of constituents. The probability of a constituent is the number of times the particular group has occurred in the learned text, normalized by the total number of groups.

$$P(c) = \frac{|c' \in C : yield(c') = yield(c)|}{|C|} \text{ where } C \text{ is the entire set of constituents.}$$

ABL:const This system computes the probability of a constituent based on the count of the words delimited by the constituent *and* its non-terminal.

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

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

[3] The algorithm does not know any (linguistic) names for the types, so the algorithm uses $\{X_1, X_2, \ldots\}$.

[4] The different names of the bracket selection methods will also be used when describing the results.

	NCBP		NCBR		ZCS	
left	32.60		76.82		1.12	
right	82.70		**92.91**		**38.83**	
ABL:1	82.70	(1.73)	86.30	(0.97)	17.39	(3.67)
ABL:terms	82.20	(0.11)	86.20	(0.05)	21.08	(0.67)
ABL:const	**86.04**	(0.01)	89.39	(0.01)	29.05	(0.00)

Table 1: Results ATIS corpus

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

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. [Boo69]). A Viterbi style algorithm optimization [Vit67] 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 the geometric mean[5], rather than its product. [CC98]

4 Test environment

The three different ABL algorithms and two baseline systems, a left and a right branching system, are tested on the Penn Treebank ATIS corpus [MSM93]. The corpus consists of 716 sentences containing 11,777 constituents. The sentences of the corpus were stripped of their structure. These plain sentences are used in the learning algorithms and the resulting structure is compared to the structure of the original corpus.

The different systems are compared against each other using the following metrics:

NCBP Non-Crossing Brackets Precision: the percentage of learned constituents that do not overlap any constituents in the original corpus.

NCBR Non-Crossing Brackets Recall: the percentage of original constituents that do not overlap any constituents in the learned corpus.

ZCS Zero-Crossing Sentences: the percentage of sentences that do not have any overlapping constituents.

Since the ABL:1 system depends on the order of the sentences in the corpus and the two probabilistic methods choose constituents at random (when there are constituents that have the same probability), we have applied the different ABL algorithms 10 times and computed the mean and standard deviation (shown in brackets). The results are shown in table 1.

5 Results

As can be seen from table 1, the ATIS corpus is predominantly right branching. The right branching system performs well on all metrics. It even performs best on recall and 0-crossing sentences.

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

from San Francisco to Dallas
from Dallas to San Francisco

from $()_{X_1}$ *San Francisco (to Dallas)*$_{X_2}$
from (Dallas to $)_{X_1}$ *San Francisco* $()_{X_2}$

Figure 3: Wrong alignment of *San Francisco*

Show me the (morning)$_{X_1}$ *flights*
Show me the (nonstop)$_{X_1}$ *flights*

Figure 4: Wrong syntactic type

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)*$_{NP}$*)*$_{NP}$

Figure 5: Recursion learned in the ATIS corpus

The ABL:const method performs significantly better on all metrics compared to the other ABL methods. ABL:const outperforms even the right branching system on precision and the recall of the ABL:const method is close to that of the right branching system.

It is interesting to see that the ABL:1 system performs slightly better than the ABL:terms method on NCBP and NCBR. The ABL:1 system cannot correct any errors once they have been learned, whereas in the ABL:terms the probability of incorrect constituents can change.

The standard deviation of the results of the ABL:1 system is quite large, which indicates the importance of the order of the sentences in the corpus. The standard deviations of the probabilistic ABL systems are much smaller. When the more specific probabilistic method is used, the variance in results is close to zero.

The right branching system outperforms ABL systems on two metrics. This could be expected, since the ATIS corpus is predominantly right branching. The ABL systems do not have a directional branching method built-in.

6 Discussion and future extensions

6.1 Unintended constituents

The string edit distance algorithm has the problem that words that are too far apart can get "linked", resulting in unintended constituents. In figure 3 *San Francisco* is linked resulting in unintended constituents. We would rather have the system linking *to*, resulting in a structure with San Francisco and Dallas grouped with the same type.

Note that in the original edit distance algorithm insertion and deletion cost 1 and substitution costs 2. Linking *San Francisco* then costs 4, while linking *to* or *Dallas* costs 6.

At the moment we let the algorithm generate these constituents and hope the bracket selection phase removes the unintended constituents. Other solutions involve biasing the cost function of the edit distance algorithm or generating all possible alignments (using a different alignment algorithm).

6.2 Recursion

All ABL systems learn recursion on the ATIS corpus. Two example sentences from the ATIS corpus with their (learned and original) structure can be found in figure 5. The learned sentences are generated by the ABL system. The sentences in the example are stripped of all but the interesting constituents.

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

corpus. Although these two sentences generate different structures, similar recursion can be found in the ATIS corpus.

6.3 Wrong syntactic type

In some cases 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 figure 4. When the ABL algorithm is applied to these sentences, it will determine that *morning* and *nonstop* are of the same type. In the ATIS corpus, *morning* is tagged as an *NN* (a noun) and *nonstop* is a *JJ* (an adjective).

The constituent *morning* may also be used as a noun in other sentences, but the constituent *nonstop* never will. This information is found by looking at the distribution of the constituents in the rest of the corpus. A future extension of the ABL algorithm that uses distributional information of contexts of constituents should solve this problem.

6.4 Weakening exact match

Throughout this paper we talked about similar words, but actually meant exactly matching words. When the ABL algorithms now try to learn with two completely dissimilar sentences, no possible constituents can be found. If we weaken the exact match between words in the alignment phase of the algorithm, it is possible to learn structure even with completely dissimilar sentences.

Instead of linking exact 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 [RCF98].) The idea behind equivalence classes is that words which are closely related are grouped together.

An additional advantage of equivalence classes is that they can be learned in an unsupervised way, so we do have to not change our algorithm into a supervised method.

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.

6.5 Alternative statistics

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

When these algorithms are applied to the ATIS corpus, the system that uses more specific statistics performs significantly better. We expect to find better results when using even more specific statistics.

One interesting system that uses more specific statistics takes a DOP-like approach [Bod98], which also takes into account the inner structure of the constituents.

7 Conclusion

We have introduced a new unsupervised grammar learning and bootstrapping algorithm based on comparing and aligning sentences. It uses dissimilarities between plain sentences to find possible constituents and afterwards selects the most probable ones.

Three instances of the ABL algorithm have been applied to the unstructured sentences of the ATIS corpus, generating 86.04 % non-crossing brackets precision, 89.39 % non-crossing brackets recall and 29.05 % zero-crossing sentences.

Possible constituents are found by looking at an unlimited context. The algorithm does not use a fixed window size, so arbitrarily large constituents may be found.

Furthermore, the algorithm is able to learn recursion from a finite set of sentences.

References

[Bod98] Rens Bod. *Beyond Grammar — An Experience-Based Theory of Language*. Stanford, CA: CSLI Publications, 1998.

[Boo69] T. Booth. Probabilistic representation of formal languages. In *Conference Record of 1969 Tenth Annual Symposium on Switching and Automata Theory*, pages 74–81, 1969.

[Bri93] Eric Brill. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the Association for Computational Linguistics*, pages 259–265, 1993.

[CC98] Sharon A. Caraballo and Eugene Charniak. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298, 1998.

[Dae95] Walter Daelemans. 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, 1995.

[dM96] Carl G. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, sep 1996.

[FC92] Steven Finch and Nick Chater. Bootstrapping syntactic categories using statistical methods. In Walter Daelemans and David Powers, editors, *Background and Experiments in Machine Learning of Natural Language: Proceedings First SHOE Workshop*, pages 229–235. Institute for Language Technology and AI, Tilburg University, The Netherlands, 1992.

[Grü94] Peter Grünwald. 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, 1994.

[Har51] Zellig Harris. *Methods in Structural Linguistics*. Chicago, IL: University of Chicago Press, 1951.

[MM90] D. Magerman and M. Marcus. Parsing natural language using mutual information statistics. In *Proceedings of the National Conference on Artificial Intelligence*, pages 984–989. Cambridge, MA: MIT Press, 1990.

[MSM93] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of english: the Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Pow97] David M. P. Powers. Machine learning of natural language. Association for Computational Linguistics/European Chapter of the Association for Computational Linguistics Tutorial Notes, Madrid, Spain, 1997.

[PS92] F. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, 1992.

[RCF98] Martin Redington, Nick Chater, and Steven Finch. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469, 1998.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:260–269, 1967.

[WF74] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, jan 1974.