arXiv:cs/0104005v1 [cs.LG] 3 Apr 2001

Bootstrapping Structure using Similarity

Menno van Zaanen
University of Leeds

Abstract

In this paper a new similarity-based learning algorithm, inspired by string edit-distance (Wagner and Fischer, 1974), is applied to the problem of bootstrapping structure from scratch. The algorithm takes a corpus of unannotated sentences as input and returns a corpus of bracketed sentences. The method works on pairs of unstructured sentences or sentences partially bracketed by the algorithm that have one or more words in common. It finds parts of sentences that are interchangeable (i.e. the parts of the sentences that are different in both sentences). These parts are taken as possible constituents of the same type. While this corresponds to the basic bootstrapping step of the algorithm, further structure may be learned from comparison with other (similar) sentences.

We used this method for bootstrapping structure from the flat sentences of the Penn Treebank ATIS corpus, and compared the resulting structured sentences to the structured sentences in the ATIS corpus. Similarly, the algorithm was tested on the OVIS corpus. We obtained 86.04 % non-crossing brackets precision on the ATIS corpus and 89.39 % non-crossing brackets precision on the OVIS corpus.

1 Introduction

People seem to learn syntactic structure without great difficulty. Unfortunately, it is difficult to model this process and therefore, it is difficult to make a computer learn structure.

Instead of actually modeling the human process of language learning, we propose a grammar learning algorithm and apply it to a corpus of natural language sentences. The algorithm should assign structure to the sentences which is similar to the structure people give to the sentences.

The algorithm consists of two phases. The first phase, *alignment learning*, builds a large set of possible constituents by aligning sentences. The second phase, which is called *bracket selection*, selects the best constituents from this set.

The rest of the paper is organized as follows. We will start out by describing some previous work in grammar induction. This is followed by a detailed description of the ABL algorithm, where three different implementations of ABL will be described, *ABL:incr*, *ABL:leaf* and *ABL:branch*. We will then discuss some of the properties of the ABL algorithms. First, we claim the ABL algorithms can generate recursive structures and then discuss problems with PP attachment. After that, some results of the algorithms applied to the ATIS corpus (Marcus et al., 1993) and to the OVIS corpus (Bonnema et al., 1997) will be presented, followed by a description of future research on the ABL algorithm.

2 Previous Work

This section contains a brief overview of previous work in grammar acquisition systems. Some differences between supervised and unsupervised acquisition methods will be discussed, followed by a discussion of several different methods. We then relate ABL to some of these methods.

2.1 Supervised versus Unsupervised

Language learning algorithms can be roughly divided into two groups, *supervised* and *unsupervised* learning algorithms based on the amount of information about the language they use. All learning algorithms use a *teacher* that gives examples of (unstructured) sentences in the language. In addition, some algorithms also use a *critic*. A critic may be asked if a certain sentence (possibly including structure) is a valid sentence in the language. The algorithm can use a critic to validate hypotheses about the language.[1] Supervised language learning methods use a teacher and a critic, whereas the unsupervised methods only use a teacher. (Powers, 1997)

Supervised language learning methods typically generate better results. These methods can tune their output, since they receive knowledge of the structure of the language (by initialisation or querying a critic). In contrast, unsupervised language learning methods do not receive these structured sentences, so they do not know at all what the output should look like.

However, it is worthwhile to investigate unsupervised language 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)

2.2 Language Learning Methods

In this section a short overview of several supervised and unsupervised language learning methods will be given. However, the focus will be on on unsupervised language learning methods, since we are interested in what results can be achieved using as little information as possible.

There have been many different approaches to language learning, only some of the different systems will be described 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.

A lot of recent language learning methods are based on probabilistic or counting theories. An example of this type of method is Memory-Based Learning (MBL) which keeps track of the distribution of contexts of words and assigns word types based on that information (Daelemans, 1995). Magerman and Marcus (1990) describe a system that can find constituent boundaries using mutual information of n-grams within sentences, while in (Finch and Chater, 1992) and (Redington et al., 1998) models are proposed that use distributional information to acquire syntactic categories.

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

Another type of language learning system is based on the Minimum Description Length (MDL) principle. These systems compress the input corpus by finding the minimum description needed to express the corpus. The compression results in a grammar that can describe the corpus. Examples of these systems can be found in (Grünwald, 1994) and (de Marcken, 1996).

Wolff (1982) describes a system that 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 to direct search for a grammar. The grammar induction method found in (Stolcke and Omohundro, 1994) 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 and Young, 1990), while Pereira and Schabes (1992) apply the inside-outside algorithm to a partially structured corpus.

The final system mentioned here is Transformation-Based Learning (Brill, 1993), which is also a supervised system. This system differs from others in that it is a non-probabilistic system that learns transformations to improve a naive parse (for example a right branching parse).

2.3 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 step is a way of compressing the corpus. Similar techniques can be found in systems that are based on the MDL principle (Grünwald, 1994). 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 step 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 (Finch and Chater, 1992) or (Redington et al., 1998). On the other hand, ABL assigns a probability to the different constituents, which is similar to an SCFG (Baker, 1979).

3 Algorithm

In this section we describe an unsupervised grammar learning algorithm that learns using a corpus of plain sentences, so neither pre-tagged nor pre-labelled or bracketed sentences are used. 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 output treebank.

The algorithm consists of two distinct phases: *alignment learning* and *bracket selection*. Alignment learning finds possible constituents by aligning sentences from the corpus. The bracket selection phase selects constituents from the possibly overlapping constituents that were found in the alignment learning phase.

Both phases will be described in more detail in the next two sections.

3.1 Alignment Learning

The alignment learning phase is based on Harris's idea of interchangeability (Harris, 1951) that states that *constituents of the same type can be replaced by each other*. This means that for example in the sentence *What is a dual carrier* the noun phrase *a dual carrier* may be replaced by another noun phrase.[2] Replacing the noun phrase with *the payload of an African Swallow* yields the sentences *What is the payload of an African Swallow*, which is another well-formed sentence.

ABL uses the replacement feature to find constituents by reversing the idea. Instead of replacing parts of sentences that have the same type, this idea is used to find constituents of the same type by looking for parts of sentences that can be replaced by each other. In the example this means the algorithm looks for parts as *a dual carrier* and *the payload of an African Swallow*, which then are remembered as possible constituents.

[2]Most example sentences used in this paper can be found in the ATIS corpus.

Finding replaceable parts of sentences is done by finding parts that are identical in two sentences and parts that are distinct. The distinct parts of the sentences are parts that can be interchanged. See for example the sentences in figure 1. The part *What is* is identical in both sentences, while the noun phrases are distinct parts. ABL now assumes the distinct parts of the sentences are constituents of the same type.

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: Bootstrapping structure

An instance of the string edit distance algorithm (Wagner and Fischer, 1974) that finds the longest common subsequences in two sentences is used. These longest common subsequences are parts that are identical in both sentences. The *distinct* parts of the sentences are exactly these parts of the sentences that are not part of the longest common subsequences. In figure 1, *What is* is the longest common subsequence (and the only one), while *a dual carrier* and *the payload of an African Swallow* are the remaining (distinct) parts of the sentences.

The distinct parts of the sentences, found by the string edit distance algorithm, are then grouped and labelled. Respective distinct parts receive the same non-terminal. Since the algorithm does not know any linguistically motivated names for the non-terminals, it assigns names $X_1, X_2, \ldots$ to the different constituents.

More structure is learned when aligning more than two sentences. Each new sentence is com-

```
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

pared to all (partially structured) sentences in the partially structured corpus. An overview of the algorithm can be found in figure 2.

Sentences are always aligned without looking at the structure that is already known; all newly learned structure is added to the old structure, which may yield overlapping constituents.

(Book Delta 128)$_{X_1}$ *from Dallas to Boston*
(Give me all flights)$_{X_1}$ *from Dallas to Boston*

Give me (all flights from Dallas to Boston)$_{X_2}$
Give me (information on reservations)$_{X_2}$

Figure 3: Overlapping constituents

In figure 3 two overlapping constituents are learned on the sentence *Give me all flights from Dallas to Boston*. Constituent X_1 is learned when aligning to the sentence *Book Delta 128 from Dallas to Boston* and the other constituent (X_2) is learned when aligning to the sentence *Give me information on reservations*. The bracket selection phase selects brackets until no more overlapping brackets are found.

An unstructured sentence is sometimes aligned to a partially structured sentence (which was already in the partially structured corpus). Aligning these sentences might yield a constituent that was already present in the partially structured sentence. The new constituent in the unstructured sentence then receives the same non-terminal as the constituent in the partially structured sentence. An example of this can be found in figure 4. Sentences 1 and 2 are compared, resulting in sentences 3 and 4.

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*

Figure 4: Learning with a partially structured sentence and an unstructured sentence

It may even be the case that two partially structured sentences are aligned. This occurs when a new sentence has been aligned to a sentence (and has received some structure) and is then aligned to another partially structured sentence in memory. If this combination of sentences yields a constituent with two distinct non-terminals (a different one in each sentence), the two non-terminals are merged. All occurrences of these non-terminals are updated in the corpus. In figure 5 sentences 1 and 2 are compared, resulting in sentences 3 and 4.

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)*$_{X_3}$

Figure 5: Learning with two partially structured sentences

Merging non-terminals (as shown in figure 4 reduce the number of different non-terminals. By merging non-terminals, we assume that constituents in a certain context can only have one type. In section 6.3 we discuss the consequences and propose a method that loosens this assumption.

3.2 Bracket Selection

The alignment learning phase may generate unwanted overlapping constituents as can be seen in figure 3. Since we assume the underlying grammar of the corpus is context-free and we want to know the most appropriate disambiguated structure of the sentences of the corpus, overlapping constituents have to be eliminated. The bracket selection phase does exactly this.

Three different approaches to the selection of constituents have been implemented. Note that the selected constituents are kept (i.e. they are *not* removed).

ABL:incr Assume the constituent learned earlier is correct. This means that when new constituents overlap with older ones, they are ignored.

ABL:leaf Constituents are selected based on their probability. The system computes the probability of a constituent by counting the number of times the sequence of words in the constituent occurs as a constituent, normalized by the total number of constituents. The probability of a constituent c can be computed as follows:

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

where C is the entire set of constituents.

ABL:branch This method is similar to the ABL:leaf method. The probability of a constituent is now computed by counting the number of times the sequence of words in a constituent *together with its root non-terminal* occur, normalized by the number of constituents with that root non-terminal:

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

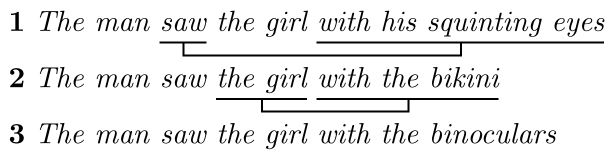
The ABL:incr method may be applied during the alignment learning phase. When a constituent is found that overlaps with an existing constituent, the new constituent will not be stored. ABL:leaf and ABL:branch bracket selection methods are applied after the alignment learning phase, since more specific constituent counts are available then.

The probabilistic methods, ABL:leaf and ABL:branch, both compute the probability of constituents. Since more than just two constituents can overlap, the methods need to consider the probability of all possible combinations of constituents, which is the product of the probabilities of the separate constituents as in SCFGs (cf. (Booth, 1969)). Viterbi style algorithm optimization (Viterbi, 1967) is used to efficiently select the best combination of constituents.

Computing the probability of a combination of constituents by taking the product of the probabilities leads to a "thrashing" effect. Since the product of two probabilities is always smaller than or equal to the two single probabilities, the system has a preference to single constituents over a combination of constituents. Therefore, the geometric mean is used to compute the probability of a combination of constituents instead (Caraballo and Charniak, 1998).[3] When more combinations of constituents have the same probability, one is chosen at random.

4 Discussion

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



1 *The man saw the girl with his squinting eyes*

2 *The man saw the girl with the bikini*

3 *The man saw the girl with the binoculars*

Figure 7: PP attachment

4.1 Recursion

All ABL algorithms generate recursive structures. Figure 6 shows some real examples of recursive structures generated by the ABL systems compared to their structure in the ATIS corpus.

In the first example ABL finds a recursive structure, but there is no recursion in the original sentence. The context of the constituents is quite similar (both follow the word "on"). The second example contains recursive structures that are very alike. The original corpus contains two recursive noun phrases, while the learned structure describes recursive noun phrases without the determiner. The last example contains recursive noun phrases with conjunction. Again similar structures are found in the learned and original corpora.

Intuitively, a recursive structure is formed first by building the constituents that form the structure of the recursion but with different root non-terminals. Now the non-terminals need to be merged. This happens when two partially structured sentences are compared to each other yielding a constituent that already existed in both sentences with the non-terminals present in the "recursive" structure (see figure 5). The non-terminals are then merged, resulting in a recursive structure.

4.2 PP attachment

Most unsupervised grammar learning systems have difficulty regarding prepositional phrase attachments. In figure 7 the first two sentences seem similar, but the prepositional phrase *with his squinting eyes* indicates the man *saw* the girl using his eyes, while *with the bikini* determines *the girl* to have the bikini, since it is difficult to see using a bikini.[4] On the other hand, the third sentence in figure 7 is ambiguous.

[4]The phrase *with his squinting eyes* could also mean that the girl has squinting eyes, just like the man, but this reading is less likely.

learned *Book reservations for five from Dallas to Baltimore on (flight 314 on (May 12th)*$_{X_{15}}$*)*$_{X_{15}}$
original *Book reservations for five from Dallas to Baltimore on (flight 314)*$_{NP}$ *on (May 12th)*$_{NP}$
learned *What is the (name of the (airport in Boston)*$_{X_{18}}$*)*$_{X_{18}}$
original *What is (the name of (the airport in Boston)*$_{NP}$*)*$_{NP}$
learned *Explain classes QW and (QX and (Y)*$_{X_{52}}$*)*$_{X_{52}}$
original *Explain classes ((QW)*$_{NP}$ *and (QX)*$_{NP}$ *and (Y)*$_{NP}$*)*$_{NP}$

Figure 6: Recursion in the ATIS corpus

To solve the problem of PP attachment, several types of information may be needed. The PP attachment in the first sentence can be solved using syntactic information only, since *his* clearly indicates the man is looking. However, the second sentence can only be solved using semantic information. Without knowing that a bikini does not help enhancing sight, it is difficult if not impossible to attach the prepositional phrase to the correct constituent.

The third sentence shows that sometimes discourse information is needed to solve attachment. The prepositional phrase *with the binoculars* should be attached to *saw* or *the girl* depending on the who has the binoculars.

Since the ABL system builds structures based on a context-free grammar and does not use any other information than the input sentences, ABL might seem unable to solve PP attachments correctly. However, the bracket selection phase might find a difference in distribution between *the girl with his squinting eyes* and *the girl with the bikini.* This difference in distribution might resolve in the different PP attachments.

5 Results

This section describes the results of applying the three ABL algorithms and two baseline systems to the Penn Treebank ATIS (Air Travel Information System) corpus (Marcus et al., 1993) and to the OVIS corpus (Bonnema et al., 1997).[5] First the test environment is described, followed by a discussion of the test results.

5.1 Test Environment

The three ABL algorithms, ABL:incr, ABL:leaf and ABL:branch have been applied to the ATIS and OVIS corpus. The ATIS corpus contains 716 sentences and 11,777 constituents. The larger OVIS corpus contains 10,000 sentences. When single word sentences are removed, this results in a corpus of 6,797 sentences containing 48,562 constituents.

The sentences of the corpora are stripped of their structure. The resulting plain sentences are used in the learning algorithms and the resulting structured sentences are then compared to the structure of the sentences in the original corpora.

To be able to compare the results, we have also computed the results of two baseline systems, a left-branching and a right-branching system on both corpora.

The results of the ABL:incr system depend on the order of the sentences in the corpus and the two probabilistic ABL systems sometimes choose constituents at random (i.e. when more combinations of constituents have the same probability). Therefore, the ABL:incr system is applied to ten differently ordered versions of the corpora. Likewise, we applied the probabilistic ABL systems ten times to the corpora to account for the random selection of constituents. The mean results are shown in table 1 and the standard deviations are shown in brackets.

Three different metrics are used to compare the results of the different algorithms:

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.

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

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

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

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

$NonLaBr(T)$ denotes the set of non-labelled brackets of the non-terminal nodes of T.
$Cross(U, V)$ denotes the subset of brackets from U that cross at least one bracket in V.
O_i represents a tree in the learned corpus.
T_i is a tree in $TEST$, the original corpus. (Sima'an, 1999)

Figure 8: Formulas of the different metrics

The formulas describing the metrics can be found in figure 8.

5.2 Test Results

The results of applying the ABL algorithms and the baseline systems on the ATIS and OVIS corpora can be found in table 1. The ATIS corpus is predominantly right branching as can be seen in the results, while the OVIS corpus is right branching to a lesser degree.

The ABL:branch method performs significantly better on all metrics on both the ATIS and OVIS corpora compared to the other ABL methods. The only deviating ABL result is the ZCS of the ABL:incr system in the OVIS corpus.

It is interesting to see that the ABL:incr system outperforms ABL:leaf. In the ABL:incr system, incorrectly learned constituents will never be corrected. The idea behind the probabilistic methods (including ABL:leaf) is that incorrect constituents will be corrected because they will receive lower probabilities. Apparently, the statistics used in the ABL:leaf method do not provide enough information to make a correct bracket selection, whereas ABL:branch does.

The standard deviation of the results of the ABL:incr system is quite large, which indicates that the order of the sentences in the corpus is important. On the other hand, selecting random constituents (when multiple constituents have the same probability) in the probabilistic ABL systems yields less varying results. When a more specific probabilistic method is used (ABL:branch), the variance in results is almost zero.

The right branching system outperforms ABL systems on several metrics (the ABL:branch method almost keeps up however). This could be expected, since the ATIS is predominantly right branching (the OVIS corpus to a lesser degree). Therefore, the right branching system should perform well. The ABL systems do not have a directional branching method built-in; furthermore, the ABL systems have no adjustable parameters that can be used to tune the algorithms to the specific corpora. This means that the right branching system does not perform well on a corpus of a predominantly left branching language (for example Japanese), while we expect that ABL does.

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 % non-crossing brackets precision, while the *unsupervised* ABL significantly outperforms this method, reaching 86.04 % precision. Only their *supervised* version results in a slightly higher precision of 90.36 %.

	Results ATIS corpus			Results OVIS corpus		
	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	82.55 (0.80)	82.98 (0.78)	17.15 (1.17)	88.69 (1.11)	83.90 (1.61)	45.13 (4.12)
ABL: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)
ABL: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)

Table 1: Results ATIS and OVIS corpora

6 Future Work

Although the overall result of ABL algorithm is slightly disappointing, some of the results of the different ABL algorithms are encouraging. We expect future extensions of the ABL system to improve performance.

This section contains some future extensions to the ABL implementations described in this paper. These extensions solve certain problems the current ABL systems have. First, we take a look at different alignment schemes, that may be used as an alternative to the string edit distance algorithm. Then an alternative probabilistic model for bracket selection is considered. Finally, we discuss two methods that generate more possible constituents by adding context or weakening exact match.

6.1 Different Alignment Schemes

The edit distance algorithm that is used to find identical parts in sentences sometimes finds alignments that generate unintended constituents. If we consider the sentences 1 and 2 in figure 9, the edit distance aligns these sentences as in sentences 3 and 4. Unfortunately, the alignment in sentences 3 and 4 generate unintended constituents. The most preferred alignment is shown in sentences 7 and 8, since *San Francisco* and *Dallas* are grouped and receive the same type.

This problem occurs every time the algorithm aligns words that are “too far apart”. The relative distance between the two *San Francisco*s in the two sentences is much larger than the relative distance between the word *to* in both sentences.

There are two possible solutions to this problem. The first solution is to change the cost function of the edit distance algorithm to let it select the better alignment. Another solution is to simply generate all possible alignments (using a completely different alignment algorithm) and let the bracket selection phase of the algorithm select the best alignment.

1 *from San Francisco to Dallas*
2 *from Dallas to San Francisco*

---

3 *from* $()_{X_1}$ *San Francisco* $(\textit{to Dallas})_{X_2}$
4 *from* $(\textit{Dallas to})_{X_1}$ *San Francisco* $()_{X_2}$

---

5 *from* $(\textit{San Francisco to})_{X_3}$ *Dallas* $()_{X_4}$
6 *from* $()_{X_3}$ *Dallas* $(\textit{to San Francisco})_{X_4}$

---

7 *from* $(\textit{San Francisco})_{X_5}$ *to* $(\textit{Dallas})_{X_6}$
8 *from* $(\textit{Dallas})_{X_5}$ *to* $(\textit{San Francisco})_{X_6}$

Figure 9: Unintended constituents

A better cost function should be biased towards alignments with a small relative distance. This can be accomplished by letting the cost function return a high cost when the difference of the relative offsets of the words is large. The relative distance between the two *San Francisco*s in sentences 3 and 4 in figure 9 is larger compared to the relative distance between the two *to*s in sentences 7 and 8. Therefore the total edit cost of sentences 7 and 8 will be less than the edit cost of sentences 3 and 4 or sentences 5 and 6.

When all possible alignments are generated, there is a large probability the intended alignment will also be found. Unfortunately, it is not known which of the possible alignments is the intended. The bracket selection phase should select the best constituents from all possible constituents that are generated by the alignment learning phase. When the probabilistic bracket selection methods are used, we just have to assume that the intended alignment generates more probable constituents, or equivalently that the intended alignment contains constituents that occur more often in the corpus than the unintended constituents.

6.2 Alternative Probabilistic Models

The results of the application of the algorithm on the ATIS corpus show that the probabilistic methods generate less fluctuating results (i.e. they have a smaller standard deviation) than the non-probabilistic method. The results of the ABL:branch method show less deviation than the results of the ABL:leaf method. Furthermore, the ABL:branch method generates the best results. These results indicate that a more specific probabilistic method works better in bracket selection.

As future research, a DOP-like probabilistic method (Bod, 1998) of bracket selection will be implemented. In this approach, all possible constituents are broken into fragments (cf. elementary subtrees in DOP). The probability of such a fragment f is:

$$P(f|root(f) = r) = \frac{|f|}{\sum_{f':root(f')=r} |f'|}$$

When combining these fragments, the structure of constituent c can be generated. This can be seen as a derivation of the structure of the constituent. The probability of a derivation $f_1 \circ \ldots \circ f_n$ is:

$$P(f_1 \circ \ldots \circ f_n) = \prod_i P(f_i)$$

Usually, there is more than one way of deriving the structure of the constituent. The probability of the constituent is now the combination of all derivations that yield the structure of c:

$$P(c) = \sum_{d \text{ derives } c} P(d)$$

The probabilistic model of DOP does not only use the counts of terminals or non-terminals like in the ABL:leaf and ABL:branch methods, but also uses the internal structure of constituents. This yields a much more specific stochastic model.

The ABL:branch bracket selection method is comparable to the probabilistic model of SCFGs (cf. (Booth, 1969)), where probabilities of context-free grammar rules (terminals and their root non-terminal) are used. Since DOP clearly outperforms SCFGs (as shown in (Bod, 1995)), a DOP model in bracket selection is expected to increase performance.

6.3 Adding Context

Some problematic cases exist where ABL might seem unable to learn the correct syntactic type. Consider sentences as in figure 10. The ABL algorithm finds that *morning* and *nonstop* are of the same type, since the rest of the two sentences is identical. However, *morning* is tagged as *NN* (a noun) and *nonstop* as *JJ* (an adjective).

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

Figure 10: Inconsistent syntactic types

On the other hand, one might argue these words *are* of the same type, exactly because they occur in the same context. Both words might be seen as some sort of adjective phrase.

This is a difference between syntactic type and functional type. *Morning* and *nonstop* have a different syntactic type, a noun and an adjective respectively, but both modify the noun *flights*, i.e. they have the same functional type. ABL finds the functional type, while the words are tagged according to their syntactic type.

Since the overall use of the two words differs greatly, they occur in different contexts. *Morning* may in general occur in places where nouns or noun phrases belong, while *nonstop* may not. This discrepancy can be used to differentiate between the two words. Instead of giving the words only one type, they get two: one type describes the context (i.e. the functional type), which is the same for both words and the other type describes the syntactic type, which is different for both words.

The distribution of syntactic types combined with functional types can be used to find words that belong to the same syntactic type. *Morning* and *nonstop* have the same functional type but different syntactic types, since *morning* normally occurs in other contexts than *nonstop*.

Since the functional type should describe the context of a constituent, merging of constituents as in figure 4 and figure 5 should only apply to functional types. Merging of syntactic types is only done when the distribution of the constituents is similar enough. This effectively loosens the assumption that constituents in a certain context have the same (syntactic *and* functional) type.

6.4 Weakening Exact Match

The algorithm described in this paper is unable to learn any structure when two completely different sentences are compared. (This is not an insurmountable problem, since other sentences can be used to learn structure on the two sentences.)

The algorithms described so far all try to align using exactly matching words. Sometimes this is a too strong demand; it is enough to match similar words. Imagine sentences 1 and 2 in figure 11, which are completely distinct. The standard ABL learning methods would conclude both are sentences, but no more structure will be found. When the algorithm knows *Book* and *List* are words of the same type (representing verbs), it would find the structures in sentences 3 and 4 where the type X_1 represents a noun phrase.

1 *Book Delta flight 430*
2 *List the cost for limousines*

---

3 *Book (Delta flight 430)*$_{X_1}$
4 *List (the cost for limousines)*$_{X_1}$

Figure 11: Sentences without identical words

An obvious way of implementing this is by using *equivalence classes.* (See for example (Redington et al., 1998).) Words that are closely related are grouped together in the same class. Words in the same equivalence class are similar enough to be aligned.

A big advantage of equivalence classes is that they can be learned in an unsupervised way. Even when the algorithm is extended with equivalence classes, it still does not need to be initialised with structured training data.

7 Conclusion

In this paper a new language learning algorithm based on aligning sentences is introduced. It uses distinctions between sentences to find possible constituents during the alignment learning phase and selects the most probable constituents afterwards in the bracket selection phase.

Three instances of the algorithm have been applied to the ATIS corpus (716 sentences) and the OVIS corpus (6,797 sentences). The alignment learning phase in the three systems is the same, but the bracket selection phase differs. ABL:incr assumes earlier constituents learned are correct, ABL:leaf and ABL:branch select brackets based on their probability. The ABL:branch method, which uses the most specific stochastic model, performs best.

The ABL algorithm is an unsupervised grammar learning and bootstrapping system. It uses plain sentences to learn structure, so neither pre-tagged, pre-labelled nor pre-bracketed sentences are used. There is no need to train the system on a structured corpus and the system does not use any parameters that need to be set.

Possible constituents are found by looking at an unlimited context. The algorithm does not use a fixed window size. Alignments (and thus constituents) of arbitrarily large size may be considered.

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

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. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Ph.D. thesis, Universiteit van Amsterdam.

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.

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

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.

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

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.

Steven Finch and Nick Chater. 1992. 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.

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.

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

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

Khalil Sima'an. 1999. *Learning Efficient Disambiguation*. Ph.D. thesis, Institute for Language, Logic and Computation, Universiteit Utrecht.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by bayesain model merging. In *Second International Conference on Grammar Inference and Applications*, pages 106–118. Berlin: Springer Verlag. Alicante, Spain.

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.