Doubling DOP*

Data Oriented Parsing based on Double-DOP and DOP*

Benno Kruit (10576223)
Sara Veldhoen (10545298)

January 22, 2014

1 Introduction

In most theories of natural language syntax, parse trees are built up from small, simple rules. When building an empirical model of observed parse trees, these rules are extended with probabilities. This gives the trees that are 'generated' by these rules their own probability, which makes it a statistical model of a distribution over natural language syntax. An alternative approach to these small rules is using larger chunks of parse trees and connecting those together to build the syntactic structures. Just like the simple rules, the chunks have probabilities as well, and connecting them creates probabilities for trees to model the statistical language distribution. The Data-Oriented Parsing approach is the most radical step away from just using small rules. It takes the trees apart in all possible ways and estimates the probabilities of the parts by counting how often they occur compared to the others. The probability of a tree built in a certain way is the product of the probability of the used parts, and the probability of that tree itself is the sum of the probabilities of all the ways it can be put together.

However, this takes too long to calculate, so it's better to only look at the ways a tree can be put together in the smallest number of steps (the shortest derivation).

1.1 The DOP model

Connecting parts of trees together is called *composing*, and building a tree from those pieces is called *deriving* the tree. *fragments subtrees*

1.2 Statistics

In contrast to *competence* models that are the subject of most linguistic study, a *performance* model of language is an estimate of the probability of observing a parse tree. It treats language as a statistical distribution over syntactic structures.

$$\Omega =$$

Using parse tree samples from the language, an estimator builds a statistical model. A parser then uses that statistical model to predict the correct parse tree of sentences. A collection of parse tree samples is called a *corpus* or *treebank*.

In theory, an estimator should make exactly the right estimations of probabilities if it's given an infinite amount of data. That is to say, it should *converge* to the true distribution. If an estimator converges in the limit, that estimator is *consistent.* However, given a finite amount of data, the estimator will probably not generate the correct probabilities. The distance between the true distribution and an estimate is called the *loss* of that estimate. The loss can be defined in different ways, but the most popular is the *mean squared difference*:

$$\mathcal{L}$$

From a true distribution, it's possible to calculate the expected loss of an estimator trained on a treebank of a certain size. This is the *risk* of that estimator given a sample size and a distribution.

$$\mathcal{E}$$

Bias is good.

2 Existing Frameworks: Double-DOP and DOP*

A PTSG consists of the symbolic grammmar, i.e. a set of fragments, and the corresponding weights. In the general case of DOP, all fragments are extracted from all the trees in the treebank. The number of fragments is exponential in the length of the sentences, thus the total number of fragments extracted would be far too large for efficient computation. Later models have therefore restricted the set of fragments in the grammar, thus improving computational efficiency. However, determining a proper subset of fragments to use is not trivial and this choice may negatively influence the performance of the grammar.

In this section, we outline two approaches to constrain the extraction of fragments: Double-DOP and DOP*. Furthermore, we discuss the similarities and dissimilarities for these two approaches.

2.1 Double-DOP

In the following, we discuss Double-DOP as it was presented in [**?**]. In this model, no unique fragments are extracted from the dataset: if a construction occurs in one tree only, it is probably not representative for the language. This is carried out by a dynamic algorithm using tree-kernels. It iterates over pairs of trees in the treebank, looking for fragments they have in common. In fact, only the largest shared fragment is stored.

The symbolic grammar that is the output of this algorithmis not guaranteed to derive each tree in the training corpus. Therefore all one-level fragments, consitutuing the set of PCFG-productions, are also added.

The Double-DOP model describes an extraction method for determining the symbolic grammar. However, it was also implemented with different estimators. The estimation is done in a second pass over the treebank, gathering frequency counts for the fragments in the gramar.

The best maximizing objective appears to be MCP (max consituent parse), which maximizes a weighted average of expected labeled recall and precision. The latter cannot be maximized directly, but is approximated by minimizing the mistake rate. Parameter λ that rules the linear interpolation was empirically found to be optimal at 1.15. Efficient calculation of MPC is possible with dynamic programming, but it can also be approximated with a list of k-best derivations.

In addition, empirical results show that the relative frequency estimate outperforms the other estimates tested, i.e. Equal Weights Estimate (first adjust counts proportional to the size of the symbolic grammar, this value proportional to fragments with the same root), and Maximum Likelihood (optimizing to maximum likelihood for the training data with an Inside-Outside algorithm).

Consistency and bias

2.2 DOP*

In DOP* [?], a rather different approach is taken called held-out estimation. The treebank is split in two parts, the extraction corpus (EC) and a held-out corpus (HC). An initial set of fragments is extracted from the EC, containing all the fragments from its trees. The weights are then determined so as to to maximize the likelihood of HC, under the assumption that this is equivalent to maximizing the joint probability of the *shortest derivations* of the trees in HC. All fragments that do not occur in such a derivation are removed from the symbolic grammar. Note that some trees in HC may not be derivable at all. Furthermore, a tree could have several shortest derivations.

Consistency and bias DOP* was claimed to be the first consistent (non-trivial) DOP-estimator, [?] provides a consistency proof. On the other hand DOP* is biased, but Zollmann shows how bias actually arises from generalization: no non-overfitting DOP estimator could be unbiased. Bias is therefore not prohibited but on the contrary a desirable property of an estimator.

In [?] it is argued that there is a problem with the consistency proof given for DOP*, as well as the non-consistency proof for other DOP-estimators by [?]. Zuidema points out that these proofs use a frequency-distribution test, whereas for DOP a weight-distribution test would be more appropriate.

2.3 Comparison

DOP* and Double-DOP differ both in the set of fragments they extract and their estimation of the weights. To investigate the exact differences, we will view both steps separately.

Note that the DOP* extraction needs another decision: in many cases, there are several shortest derivations possible. From now on, we add all fragments that occur in one of these shortest derivations to the symbolic grammar. Of course we need to adjust the weights (e.g. divide the frequency counts by the number of shortest derivations) so that no full tree gets a higher impact on the PTSG. We will keep to the original formulation of DOP* in case no derivation is possible, i.e. not including any fragments for this tree.

Extraction Double-DOP uses tree kernels to find the maximal overlapping fragments of pairs of trees, which are added to the symbolic grammar. We will call this the *maximal-overlap* method. DOP* iteratively finds the shortest derivation of one tree given all the fragments of a set of trees, herafter the *shortest-derivation* method.

It is easy to see that the DOP* extraction method does not depend on the corpus split: we can also try to find the shortest possble derivation using fragments from all the other trees.

Likewise Double-DOP could be implemented using a split, comparing pairs that consist of a tree from each part of the corpus.

Therefore, we will implement both extraction methods in a 1 vs the rest manner. In this way, we can analyse how the resulting symbolic grammars differ. The analysis will comprise the size of the resulting symbolic grammar and the relative number of fragments of certain depth. Furthermore, we might be able to find interesting patterns by manually looking at the fragments that were extracted by one of the systems only.s

Estimation The next step would be to compare the estimation methods: use either a split or the whole set of trees for both estimators. Double-DOP counts the occurences of fragments in the symbolic grammar, whereas DOP* counts the occurrences in shortest derivations. Therefore, the extraction and estimation are best done simultaneously in the latter case. This comparison would involve a performance measure, such as the F1-score for correctly predicted parses.

Example Figure **??** shows a small artificial treebank with four trees. In figure **??** all fragments are displayed that are in the symbolic grammar after applying the maximal overlap or shortest derivation extraction to this corpus. In table **??** it can be seen where these fragments originate from. Moreover, the table gives the weights assigned by both methods. Note that in the maximal overlap case, all PCFG rules in the treebank are added as well. The weight estimates are the relative frequencies of the fragments in the treebank: $p(f) = \frac{count(f)}{\sum_{f' \in F_{root(f)}} count(f')}$ [**?**]. As for the shortest derivation extraction, the weights are determined as the relative frequency of occurring in shortest derivations: $p(f) = \frac{r_c}{\sum_{k \in \{1 \ldots N\}: root(f_k) = root(f_j)} r_k}$ [**?**].

Note the remarkable differences in the weight distributions. For example, f1 gets a weight of 0.5 in the maximal overlap approach, and zero in the shortest derivation case. Of course, the sparsity of the data contributes much to these extreme variations. However, the observed differences encourage us to investigate these two approaches into more depth.

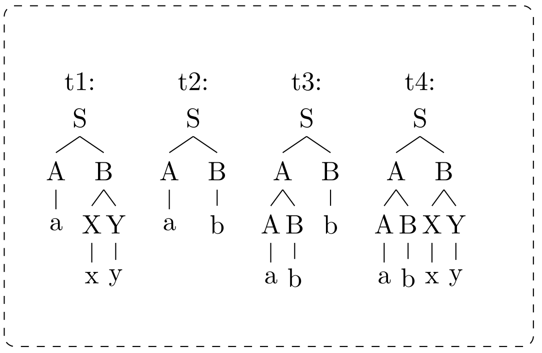

Figure 1: A toy treebank

[1]For this dataset, two shortest derivations exist for each tree. We refer to them with the following variables: 1a = f5, f6; 1b = f2, f9; 2a = f2, f8; 2b = f3, f6; 3a = f4, f8; 3b = f3, f7; 4a = f5, f7; 4b = f4, f9

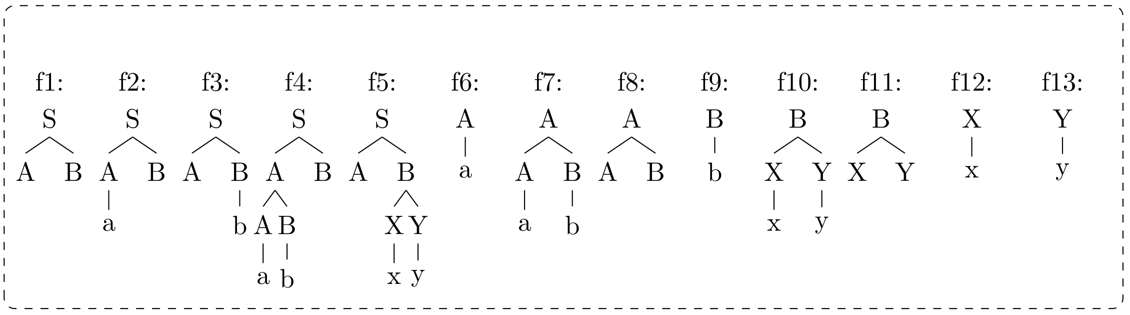

Figure 2: All extracted fragments

	Maximal overlap	Weight	Shortest derivation[1]	Weight
f1	(t1,t3),(t2,t4)	4/12	-	0
f2	(t1,t2)	2/12	1b, 2a	1/4
f3	(t2,t3)	2/12	2b, 3b	1/4
f4	(t3,t4)	2/12	3a, 4b	1/4
f5	(t1,t4)	2/12	1a, 4a	1/4
f6	(t1,t3),(t1,t4),(t2,t3),(t2,t4)	4/6	1a, 2b	1/2
f7	-	0	3b, 4a	1/2
f8	- PCFG rule	2/6	-	0
f9	(t2,t3),(t2,t4),(t3,t4)	4/6	2a, 3a	1/2
f10	-	0	1b, 4b	1/2
f11	- PCFG rule	2/6	-	0
f12	- PCFG rule	2/2	-	0
f13	- PCFG rule	2/2	-	0

Table 1: The weights assignment according to both methods in a one vs. the rest manner

3 Implementation

3.1 Extraction

Algorithm **??** performs shortest-derivation extraction (as in DOP*) in an efficient way, resembling the tree kernel approach in the iterative comparison of one tree with others. Namely, we iterate over the trees in HC (lines 2-8): create a list F of all of its fragments that occur in the rest of the treebank (line 3) and increment the count of the fragments in F that occur in the shortest derivation(s). In this way, we do not need to store all fragments in EC as in the initial step of the original DOP* algorithm.

Algorithm **??** performs the maximal-overlap extraction (as in Double-DOP). For the comparison of trees to find the maximal overlap (line 4), the tree kernel approach from [**?**] is used.

3.2 Estimation

Now for estimation, we use the output of the extraction algorithm. In the case of shortest-derivation extraction (algorithm **??**), we only need to normalize the counts in M to their relative frequency as compared to fragments that have the same root. In the case of maximal-overlap

Algorithm 1 Shortest derivation extraction in a one vs the rest manner

Data: a treebank TB

Result: a map of tree fragments and corresponding counts in shortest derivations

```
1: Initialize $M$, a map from fragments to counts, empty
2: for $t \in TB$ do
3:     $F \leftarrow Frag(t) \cap \bigcup_{t' \in TB/\{t\}} frag(t')$
4:     $D \leftarrow$ the shortest derivation(s) of $t$ using fragments in $F$
5:     for $f \in D$ do
6:         add $f$ to $M$ if $f \notin M$
7:         $M[f] \leftarrow M[f] + 1$                    ▷ Anticipating the estimation step
8:     end for
9: end for
```

Algorithm 2 Maximal overlap extraction in a one vs the rest manner

Data: a treebank TB

Result: a set of tree fragments

```
1: Initialize $M$, a map from fragments to weights, empty
2: for $t \in TB$ do
3:     for $t' \in TB/\{t\}$ do
4:         $f \leftarrow$ maximal overlapping fragment(s) of $t$ and $t'$
5:         add $f$ to $M$ if $f \notin M$
6:     end for
7: end for
```

extraction (algorithm **??**), we need to iterate over the dataset once more to count the occurences of the fragments in the output and then compute their relative frequencies.

4 Results

5 Conclusion