Doubling DOP*

Data Oriented Parsing based on Double-DOP and DOP*

Benno Kruit (10576223)
Sara Veldhoen (10545298)

January 16, 2014

1 Introduction

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-DOPand 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 [1]. 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 addition, 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.

After the extraction of the symbolic grammar, the weights are obtained. This is done in a second pass over the treebank, assessing the relative frequencies.

The Double-DOP model has its main focus on determining the symbolic grammar. However, it was implemented with different estimators and maximizing objectives. Empirical results show that

Consistency and bias

2.2 DOP*

In DOP* [2], 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.

Consistency and bias DOP* was claimed to be the first consistent (non-trivial) DOP-estimator, [2] 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 [3] 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 [4]. 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. Whether the corpus is split in two does only influence the size of the symbolic grammar and not its constitution.

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 1 shows a small artificial treebank with four trees. In figure 2 all fragments are displayed that are in the symbolic grammar after applying the maximal overlap or shortest derivation extraction to this corpus. In table 1 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: $p(f) = \frac{count(f)}{\sum_{f' \in F_{root(f)}} count(f')}$[1]. As for the shortest derivation extraction, the weights are determined as the relative frequency of occurring in shortest derivations: $p(f) = \frac{r_c}{\sum_{k \in \{1 \ldots N\}: root(f_k) = root(f_j)} r_k}$[2].

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.

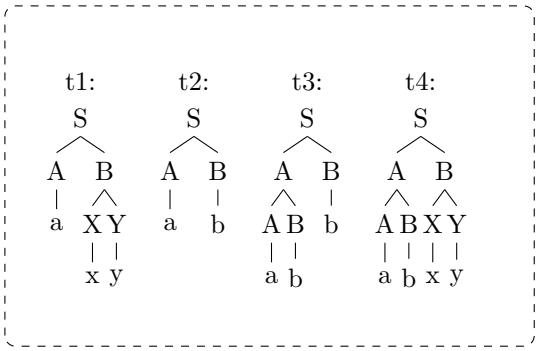

Figure 1: A toy treebank

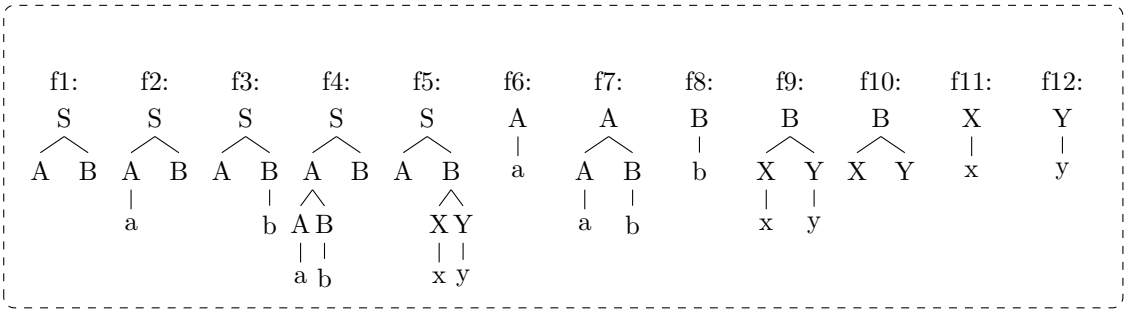

Figure 2: The fragments that get a non-zero weight in*maximal-overlap* or *shortest-derivation* extraction

[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

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

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

3.2 Estimation

Now for estimation, we use the output of the extraction algorithm. In the case of shortest-derivation extraction (algorithm 1), we only need to normalize the counts in M to their relative

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

Data: a treebank TB
Result: a set of tree fragments

```
Initialize M, a map from fragments to weights, empty
for t ∈ TB do
    for t' ∈ TB/{t} do
        f ← maximal overlapping fragment(s) of t and t'
        add f to M if f ∉ M
    end for
end for
```

frequency as compared to fragments that have the same root. In the case of maximal-overlap extraction (algorithm 2), 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

References

[1] F. Sangati and W. Zuidema, "Accurate parsing with compact tree-substitution grammars: Double-dop," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 84–95, Association for Computational Linguistics, 2011.

[2] A. Zollmann and K. Sima'an, "A consistent and efficient estimator for data-oriented parsing," *Journal of Automata, Languages and Combinatorics*, vol. 10, no. 2/3, pp. 367–388, 2005.

[3] W. Zuidema, "Theoretical evaluation of estimation methods for data-oriented parsing," in *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pp. 183–186, Association for Computational Linguistics, 2006.

[4] M. Johnson, "The dop estimation method is biased and inconsistent," *Computational Linguistics*, vol. 28, no. 1, pp. 71–76, 2002.