What are the Productive Units of Natural Language Grammar? A DOP Approach to the Automatic Identification of Constructions.

Willem Zuidema
Institute for Logic, Language and Computation
University of Amsterdam
Plantage Muidergracht 24, 1018 TV, Amsterdam, the Netherlands.
jzuidema@science.uva.nl

Abstract

We explore a novel computational approach to identifying "constructions" or "multi-word expressions" (MWEs) in an annotated corpus. In this approach, MWEs have no special status, but emerge in a general procedure for finding the best statistical grammar to describe the training corpus. The statistical grammar formalism used is that of stochastic tree substitution grammars (STSGs), such as used in Data-Oriented Parsing. We present an algorithm for calculating the expected frequencies of arbitrary subtrees given the parameters of an STSG, and a method for estimating the parameters of an STSG given observed frequencies in a tree bank. We report quantitative results on the ATIS corpus of phrase-structure annotated sentences, and give examples of the MWEs extracted from this corpus.

1 Introduction

Many current theories of language use and acquisition assume that language users store and use much larger fragments of language than the single words and rules of combination of traditional linguistic models. Such fragments are often called constructions, and the theories that assign them a central role "construction grammar" (Goldberg, 1995; Kay and Fillmore, 1999; Tomasello, 2000; Jackendoff, 2002, among others). For construction grammarians, multi-word expressions (MWEs) such as idioms, collocations, fixed expressions and compound verbs and nouns, are not so much exceptions to the rule, but rather extreme cases that reveal some fundamental properties of natural language.

In the construction grammar tradition, co-occurrence statistics from corpora have often been used as evidence for hypothesized constructions. However, such statistics are typically gathered on a case-by-case basis, and no reliable procedure exists to automatically identify constructions. In contrast, in computational linguistics, many automatic procedures are studied for identifying MWEs (Sag et al., 2002) – with varying success – but here they are treated as exceptions: identifying multi-word expressions is a pre-processing step, where typically adjacent words are grouped together after which the usual procedures for syntactic or semantic analysis can be applied. In this paper I explore an alternative formal and computational approach, where multi-word constructions have no special status, but emerge in a general procedure to find the best statistical grammar to describe a training corpus. Crucially, I use a formalism known as "Stochastic Tree Substitution Grammars" (henceforth, STSGs), which can represent single words, contiguous and noncontiguous MWEs, context-free rules or complete parse trees in a unified representation.

My approach is closely related to work in statistical parsing known as Data-Oriented Parsing (DOP), an empirically highly successful approach with labeled recall and precision scores on the Penn Tree Bank that are among the best currently obtained (Bod, 2003). DOP, first proposed in (Scha, 1990),

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 29–36, New York City, June 2006.

can be seen as an early formalization and combination of ideas from construction grammar and statistical parsing. Its key innovations were (i) the proposal to use fragments of trees from a tree bank as the symbolic backbone; (ii) the proposal to allow, in principle, trees of arbitrary size and shape as the elementary units of combination; (iii) the proposal to use the occurrence and co-occurrence frequencies as the basis for structural disambiguation in parsing.

The model I develop in this paper is true to these general DOP ideals, although it differs in important respects from the many DOP implementations that have been studied since its first inception (Bod, 1993; Goodman, 1996; Bod, 1998; Sima'an, 2002; Collins and Duffy, 2002; Bod et al., 2003, and many others). The crucial difference is in the estimation procedure for choosing the weights of the STSG based on observed frequencies in a corpus. Existing DOP models converge to STSGs that either (i) give all subtrees of the observed trees nonzero weights (Bod, 1993; Bod, 2003), or (ii) give only the largest possible fragments nonzero weights (Sima'an and Buratto, 2003; Zollmann and Sima'an, 2005). The model in this paper, in contrast, aims at finding the smallest set of productive units that explain the occurrences and co-occurrences in a corpus. Large subtrees only receive non-zero weights, if they occur more frequently than can be expected on the basis of the weights of smaller subtrees.

2 Formalism, Notation and Definitions

2.1 Stochastic Tree Substitution Grammars

STSGs are a simple generalization of Stochastic Context Free Grammars (henceforth, SCFGs), where the productive units are elementary trees of arbitrary size instead of the rewrite rules in SCFGs (which can be viewed as trees of depth 1). STSGs form a restricted subclass of Stochastic Tree Adjoining Grammars (henceforth, STAGs) (Resnik, 1992; Schabes, 1992), the difference being that STSGs only allow for substitution and not for adjunction (Joshi and Sarkar, 2003). This limits the generative capacity to that of context-free grammars, and means STSGs cannot be fully lexicalized. These limitations notwithstanding, the close relationship with STAGs is an attractive feature with extensions to the class of mildly context-sensitive languages (Joshi et al., 1991) in mind. Most importantly, however, STSGs are already able to model a vast range of statistical dependencies between words and constituents, which allows them to rightly predict the occurrences of many constructions (Bod, 1998).

For completeness, we include the usual definitions of STSGs, the substitution operation and derivation and parse probabilities (Bod, 1998), using our own notation. An STSG is a 5-tuple $\langle V_n, V_t, S, T, w\rangle$, where V_n is the set of non-terminal symbols; V_t is the set of terminal symbols; $S \in V_n$ is the start symbol; T is a set of elementary trees, such that for every $t \in T$ the unique root node $r(t) \in V_n$, the set of internal nodes $i(t) \subset V_n$ and the set of leaf nodes $l(t) \subset V_n \cup V_t$; finally, $w : T \rightarrow [0, 1]$ is a probability (weight) distribution over the elementary trees, such that for any $t \in T$, $\sum_{t' \in R(t)} w(t') = 1$, where $R(t)$ is the set of elementary trees with the same root label as t. It will prove useful to also define the set of all possible trees θ over the defined alphabets (with the same conditions on root, internal and leaf nodes as for T), and the set of all possible *complete* parse trees Θ (with $r(t) = S$ and all leaf nodes $l(t) \subset V_t$). Obviously, $T \subset \theta$ and $\Theta \subset \theta$.

The substitution operation $\circ$ is defined if the *leftmost nonterminal leaf* in t_1 is identical to the root of t_2. Performing substitution $t_1 \circ t_2$ yields t_3, if t_3 is identical to t_1 with the leftmost nonterminal leaf replaced by t_2. A derivation is a sequence of elementary trees, where the first tree $t \in T$ has root-label S and every next tree combines through substitution with the result of the substitutions before it. The probability of a derivation d is defined as the product of weights of the elementary trees involved:

$$P(d = t_1 \circ \ldots \circ t_n) = \prod_{i=1}^{n} \left(w\left(t_i\right)\right). \qquad (1)$$

A *parse tree* is any tree $t \in \Theta$. Multiple derivations can yield the same parse tree; the probability of a parse tree p equals the sum of the probabilities of the different derivations that yield that same tree:

$$P(p) \;=\; \sum_{d:\hat{d}=p} \left(P\left(d\right)\right), \qquad (2)$$

where $\hat{d}$ is the tree derived by derivation d.

In this paper, we are only concerned with grammars that define proper probability distributions over

trees, such that the probability of all derivations sum up to 1 and no probability mass gets lost in derivations that never reach a terminal yield. We require:

$$\sum_{p \in \Theta} P(p) = \sum_{d : \hat{d} \in \Theta} P(d) = 1. \quad (3)$$

2.2 Usage Frequency and Occurrence Frequency

In addition to these conventional definitions, we will make use in this paper of the concepts "usage frequency" and "occurrence frequency". When we consider an arbitrary subtree t, the usage frequency $u(t)$ describes the relative frequency with which elementary tree t is involved in a set of derivations. Given a grammar $G \in$ STSG, the expected usage frequency is:

$$u(t) = \sum_{d : t \in d} \left(P(d) \, C(t, d) \right), \quad (4)$$

where $C(t, d)$ gives the number of occurrences of t in d. The set of derivations, and hence usage frequency, is usually considered hidden information.

The occurrence frequency $f(t)$ describes the relative frequency with which t occurs as a subtree of a set of parse trees, which is usually assumed to be observable information. If grammar G is used to generate trees, it will create a tree bank where each parse tree will occur with an expected frequency as in equation (2). More generally, the expected occurrence frequency $f(t)$ (relative to the number n of complete trees in the tree bank) of a subtree t is:

$$\mathbf{E}[f(t)] = \sum_{p : t \in p^*} \left(P(p) \, C(t, p^*) \right), \quad (5)$$

where p^* is the multiset of all subtrees of p.

Hence, $w(t)$, $u(t)$ and $f(t)$ all assign values (the latter two not necessarily between 0 and 1) to trees. An important question is how these different values can be related. For STSGs which have only elementary trees of depth 1, and are thus equivalent to SCFGs, these relations are straightforward: the usage frequency of an elementary tree simply equals its expected frequency, and can be derived from the weights by multiplying inside and outside probabilities (Lari and Young, 1990). Estimating the weights of an (unconstrained and untransformed) SCFG from an tree bank is straightforward, as weights, in the limit, simply equal the relative frequency of each depth-1 subtree (relative to other depth-1 subtrees with the same root label).

When elementary trees can be of arbitrary depth, however, many different derivations can yield the same tree, and a given subtree t can emerge without the corresponding elementary tree ever having been used. The expected frequencies are sums of products, and – if one wants to avoid exhaustively enumerating all possible parse trees – surprisingly difficult to calculate, as will become clear below.

2.3 From weights to usage frequencies and back

Relating usage frequencies to weights is relatively simple. With a bit of algebra we can work out the following relations:

$$u(t) = \begin{cases} w(t) & \text{if } r(t) = S \\ w(t) \displaystyle\sum_{t' : r(t) \in l(t')} u(t') C_t^{t'} & \text{otherwise} \end{cases} \quad (6)$$

where $C_t^{t'}$ gives the number of occurrences of the root label $r(t)$ of t among the leaves of t'. The inverse relation is straightforward:

$$w(t) = \frac{u(t)}{\sum_{t' \in R(t)} u(t')}. \quad (7)$$

2.4 From usage frequency to expected frequency

The two remaining problems – calculating expected frequencies from weights and estimating the weights from observed frequencies – are surprisingly difficult and heretofore not satisfactorily solved. In (Zuidema, 2006) we evaluate existing estimation methods for Data-Oriented Parsing, and show that they are ill-suited for learning tasks such as studied in this paper. In the next section, we present a new algorithm for estimation, which makes use of a method for calculating expected frequencies that we sketch in this section. This method makes use of sub- and supertree relations that we explain first.

We define two types of subtrees of a given tree t, which, for lack of better terminology, we will call "twigs" and "prunes" of t. Twigs are those subtrees headed by any of t's internal nodes and everything

below. Prunes are those subtrees headed by t's root-node, pruned at any number (≥ 0) of internal nodes. Using $\circ$ to indicate left-most substitution, we write:

- t_1 is a *twig* of t_2, if either $t_1 = t_2$ or $\exists t_3$, such that $t_3 \circ t_1 = t_2$;
- t_1 is a *prune* of t_2, if either $t_1 = t_2$ or $\exists t_3 \ldots t_n$, such that $t_1 \circ t_3 \ldots \circ t_n = t_2$;
- $t' = pr_x(t)$, if x is a set of nodes in t, such that if t is pruned at each $i \in x$ it equals t'.

Thus defined, the set of all subtrees $st(t)$ of t corresponds to the set of all prunes of all twigs of t: $st(t) = \{t''|\exists t'(t' \in tw(t) \wedge t'' \in pr(t'))$.

We further define the sets of supertwigs, superprunes and supertrees as follows:

- $\widehat{tw}(t) = \{t'|t \in tw(t')\}$
- $\widehat{pr_x}(t) = \{t'|t = pr_x(t')\}$
- $\widehat{st}(t) = \{t'|t \in st(t')\}$.

Using these sets, and the set of derivations $D(t)$ of the fragment t, a general expression for the expected frequency of t is:

$$\begin{aligned} \mathbf{E}[f(t)] &= \sum_{d \in D(t)} \alpha\beta \\ \alpha &= \sum_{\tau \in \widehat{tw}(d_1)} \sum_{\tau' \in \widehat{pr_{x(t)}}(\tau)} u(\tau') \\ \beta &= \prod_{\substack{t' \in \\ \langle d_2, \ldots, d_n \rangle}} \sum_{\tau' \in \widehat{pr_{x(t)}}(t')} w\left(\tau'\right) \end{aligned} \quad (8)$$

where $\langle d_1, \ldots, d_n \rangle$ is the sequence of elementary trees in derivation d. A derivation of this equation is provided on the author's website[1]. Note that it will, in general, be computationally extremely expensive to calculate $\mathbf{E}[f(t)]$. We will come back to computational efficiency issues in the discussion.

[1] http://staff.science.uva.nl/∼jzuidema. The intuition behind it is as follows. Observe first that there are many ways in which an arbitrary fragment t can emerge, many of which do not involve the usage of the *elementary tree* t. It is useful to partition the set of all derivations of complete parse trees according to the substitution sites inside t that they involve, and hence according to the corresponding derivations of t. The first summation in (8) simply sums over all these cases.

Each derivation of t involves a first elementary tree d_1, and possibly a sequence of further elementary trees $\langle d_2, \ldots, d_n \rangle$. Roughly speaking, the α-term in equation (8) describes the frequency with which a d_1 will be generated. The β-term then describes the probability that d_1 will be expanded as t. The equation simplifies considerably for those fragments that have no nonterminal leaves: the set $\widehat{pr_{x(t)}}$ then only contains t, and the two summations over this set disappear. The equation further simplifies if only depth-1 elementary trees have nonzero weights (i.e. for SCFGs): α and β then essentially give outside and inside probabilities (Lari and Young, 1990). However, for unconstrained STSGs we need all sums and products in (8).

3 Estimation: push-n-pull

The goal of this paper is an automatic discovery procedure for finding "constructions" based on occurrence and co-occurrence frequencies in a corpus. Now that we have introduced the necessary terminology, we can reformulate this goal as follows: *What are the elementary trees with multiple words with the highest usage frequency in the STSG estimated from an annotated corpus?* Thus phrased, the crucial next step is to decide on an estimation procedure for learning an STSG from a corpus.

Here we develop an estimation procedure we call "push-n-pull". The basic idea is as follows. Given an initial setting of the parameters, the method calculates the expected frequency of all complete and incomplete trees. If a tree's expected frequency is higher than its observed frequency, the method subtracts the difference from the tree's score, and distributes ("pushes") it over the trees involved in its derivations. If it is lower, it "pulls" the difference from these same derivations. The method includes a bias for moving probability mass to smaller elementary trees, to avoid overfitting; its effects become smaller as more data gets observed.

Because the method for calculating estimated frequency works with usage-frequencies, the push-n-pull algorithm also uses these as parameters. More precisely, it manipulates a "score", which is the product of usage frequency and the total number of parse trees observed. Implicit here is the assumption that by shifting usage frequencies between different derivations, the relation with weights remains as in equation (6). Simulations suggest this is reasonable.

In the current implementation, the method starts with all frequency mass in the longest derivations, i.e. in the depth-1 elementary trees. Finally, the current implementation is incremental. It keeps track of the frequencies with which it observes subtrees in a corpus. For each tree received, it finds all derivations and all probabilities, updates frequencies and scores according to the rules sketched above. In pseudo-code, the push-n-pull algorithm is as follows:

for each observed parse tree p

for each depth-1 subtree t in p
    update-score$(t, 1.0)$
for each subtree t of p
    $\Delta =$min$(sc(t), B + \gamma(\mathbf{E}[f(t)] - f(t)))$
    $\Delta' = 0$
    for each of n derivations d of t
        let $t' \ldots t''$ be all elementary trees in d
            $\delta =$min$(sc(t'), \ldots, sc(t''), -\Delta/n)$
        $\Delta' -= \delta$
        for each elementary tree t' in d
            update-score(t', δ)
    update-score (t, Δ')

where $sc(t)$ is the score of t, B is the bias towards smaller subtrees, γ is the learning rate parameter and $f(t)$ is the observed frequency of t. Δ' thus gives the actual change in the score of t, based on the difference between expected and observed frequency, bias, learning rate and how much scores can be pushed or pulled[2]. For computational efficiency, only subtrees with a depth no larger than $d = 3$ or $d = 4$ and only derivations involving 2 elementary trees are considered.

4 Results

We have implemented the algorithms for calculating the expected frequency, and the push-n-pull algorithm for estimation. We have evaluated the algorithms on a number of simple example STSGs and found that the expected frequency algorithm correctly predicts observed frequencies. We have further found that – unlike existing estimation methods – the push-n-pull algorithm converges to STSGs that closely model the observed frequencies (i.e. that maximize the likelihood of the data) without putting all probability mass in the largest elementary trees (i.e. whilst retaining generalizations about the data).

Here we report first quantitative results on the ATIS3 corpus (Hemphill et al., 1990). Before processing, all trees (train and test set) were converted to a format that our current implementation requires (all non-terminal labels are unique, all internal nodes have two daughters, all preterminal nodes have a single lexical daughter; all unary productions and all traces were removed). The set of trees was randomly split in a train set of 462 trees, and a test set of 116 trees. The push-n-pull algorithm was then run in 10 passes over the train set, with $d = 3$, $B = 0$ and $\gamma = 0.1$. By calculating the most probable parse[3] for each yield of the trees in test set, and running "evalb" we arrive at the following quantitative results: a string set coverage of 84% (19 failed parses), labeled recall of 95.07, and labeled precision of 95.07. We obtained almost identical numbers on the same data with a reimplementation of the DOP1 algorithm (Bod, 1998).

method	# rules	Cov.	LR	LP	EM
DOP1	77852	84%	95.07	95.07	83.5
p-n-p	58799	84%	95.07	95.07	83.5

Table 1: Parseval scores of DOP1 and push-n-pull on the same 462-116 random train-testset split of a treebank derived from the ATIS3 corpus (we emphasize that all trees, also those of the test-set, were converted to Chomsky Normal Form, whereby unary production and traces were removed and top-nodes relabeled "TOP". These results are thus not comparable to previous methods evaluated on the ATIS3 corpus.) EM is "exact match".

method	# rules	Cov.	LR	LP	EM
$sc > 0.3$	8593	77%	80.8	80.8	46.3
$sc > 0.1$	98443	77%	81.9	81.9	48.8

Table 2: Parseval scores using a p-n-p induced STSG on the same treebank as in table 1, using a different random 525-53 train-testset split. Shown are results were only elementary trees with scores higher than 0.3 and 0.1 respectively are used.

However, more interesting is a qualitative analysis of the STSG induced, which shows that, unlike DOP1, push-n-pull arrives at a grammar that gives high weights (and scores) to those elementary

[2] An important topic for future research is to clarify the relation between push-n-pull and Expectation Maximization.

[3] We approximated the most probable parse as follows (following (Bod, 2003)). We first converted the induced STSG to an isomorph SCFG, by giving the internal nodes of every elementary tree t unique address-labels, and reading off all CFG productions (all with weight 1.0, except for the top-production, which receives the weight of t). An existing SCFG parser (Schmid, 2004) was then used, with a simple unknown word heuristic, to generate the Viterbi n-best parses with $n = 100$, and, after removing the address labels, all equal parses and their probabilities were summed, and the one with highest probability chosen.

trees that best explain the overrepresentation of certain constructions in the data. For instance, in a run with $d = 4, \gamma = 1.0, B = 1.0$, the 50 elementary trees with the highest scores, as shown in figure 1, are all exemplary of frequent formulas in the ATIS corpus such as "show me X", "I'd like to X", "which of these", "what is the X", "cheapest fare" and "flights from X to Y". In short, the push-n-pull algorithm – while starting out considering all possible subtrees – converges to a grammar which makes linguistically relevant generalizations. This allows for a more compact grammar (58799 rules in the SCFG reduction, vs. 77852 for DOP1), whilst retaining DOP's excellent empirical performance.

5 Discussion

Calculating $\mathbf{E}[f(t)]$ using equation (8) can be extremely expensive in computational terms. One will typically want to calculate this value for all subtrees, the number of which is exponential in the size of the trees in the training data. For each subtree t, we will need to consider the set of all its derivations (exponential in the size of t), and for each derivation the set of supertwigs of the first elementary trees and, for incompletely lexicalized subtrees, the set of superprunes of all elementary trees in their derivations. The latter two sets, however, need not be constructed for every time the *expected* frequency $\mathbf{E}[f(t)]$ is calculated. Instead, we can, as we do in the current implementation, keep track of the two sums for every change of the weights.

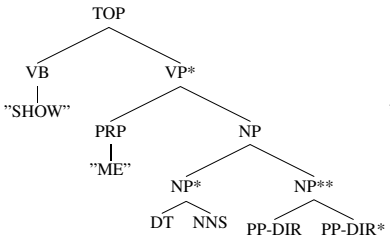
However, there are many further possibilities for improving the efficiency of the algorithm that are currently not implemented. Equation (8) remains valid under various restrictions on the elementary trees that we are willing to consider as productive units. Some of these will remove the exponential dependence on the size of the trees in the training data. For instance, in the case where we restrict the productive units (with nonzero weights) to depth-1 trees (i.e. CFG rules), equation (8) collapses to the product of inside and outside probabilities, which can be calculated using dynamical programming in polynomial time (Lari and Young, 1990). A major topic for future research is to define linguistically motivated restrictions that allow for efficient computation.

Another concern is the size of the grammar the estimation procedure produces, and hence the time and space efficiency of the resulting parser. Table 1 already showed that push-n-pull leads to a more concise grammar. The reason is that many potential elementary trees receive a score (and weight) 0. More generally, push-n-pull generates extremely tilted score distributions, which allows for even more compact but highly accurate approximations. In table 2 we show, for the $d = 4$ grammar of figure 1, that a 10-fold reduction of the grammar size by pruning elementary trees with low scores, leads only to a small decrease in the LP and LR measures.

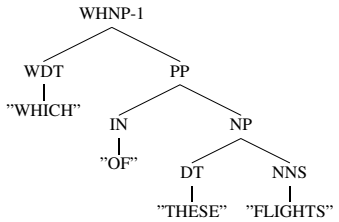
Another interesting question is if and how the current algorithm can be extended to the full class of Stochastic Tree-Adjoining Grammars (Schabes, 1992; Resnik, 1992). With the added operation of adjunction, equation (8) is not valid anymore. Given the computational complexities that it already gives rise to, however, it seems that issue of linguistically motivated restrictions (other than lexicalization) should be considered first. Finally, given that the current approach is dependent on the availability of a large annotated corpus, an important question is if and how it can be extended to work with unlabeled data. That is, can we transform the push-n-pull algorithm to perform the unsupervised learning of STSGs? Although most work on unsupervised grammar learning concerns SCFGs (including some of our own (Zuidema, 2003)) it is interesting to note that much of the evidence for construction grammar in fact comes from the language acquisition literature (Tomasello, 2000).

6 Conclusions

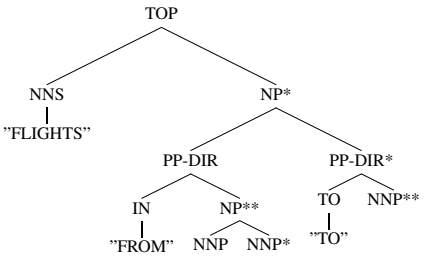
Theoretical linguistics has long strived to account for the unbounded productivity of natural language syntax with as few units and rules of combination as possible. In contrast, construction grammar and related theories of grammar postulate a heterogeneous and redundant storage of "constructions". If this view is correct, we expect to see statistical signatures of these constructions in the distributional information that can be derived from corpora of natural language utterances. How can we recover those signatures? In this paper we have presented an approach to identifying the relevant statistical correlations in a corpus based on the assumption that the



(a) The "show me NP PP" frame, which occurs very frequently in the training data and is represented in several elementary trees with high weight.



(b) The complete parse tree for the sentence "Which of these flights", which occurs 16 times in training data.



(c) The frame for "flights from NP to NP"

```
1.  ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (NP** PP-DIR PP-DIR*)))) 17.79 0.008 30)
2.  ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) NP**))) 10.34 0.004 46
3.  (TOP (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") (VP** TO VP***)))) 10.02 0.009 20
4.  (WHNP-1 (WDT "WHICH") (PP (IN "OF") (NP (DT "THESE") (NNS "FLIGHTS")))) 8.80 0.078 16
5.  (TOP (WP "WHAT") (SQ (VBZ "IS") (NP-SBJ (DT "THE") (NN "PRICE")))) 8.76 0.005 20
6.  (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* (EX "THERE") SQ**))) 8.25 0.006 36
7.  (VP* (PRP "ME") (NP (NP* (DT "THE") (NNS "FLIGHTS")) (NP** (PP-DIR IN NNP) (PP-DIR* TO NNP*)))) 7.90 0.023 18
8.  (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* (EX "THERE") (SQ** PP-DIR-3 PP-DIR-4)))) 6.64 0.005 26
9.  (TOP (PRP "I") (VP MD (VP* (VB "LIKE") (VP** TO VP***)))) 6.48 0.006 20
10. (TOP (PRP "I") (VP (VBP "NEED") (NP (NP* DT NN) (NP** PP-DIR NP***)))) 5.01 0.004 10
11. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") NNS))) 4.94 0.002 16
12. (TOP WP (SQ (VBZ "IS") (NP-SBJ (DT "THE") (NN "PRICE")))) 4.91 0.0028 20
13. (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* EX (SQ** PP-DIR-3 PP-DIR-4)))) 4.16 0.003 26
14. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NNS "FLIGHTS") NP*))) 4.01 0.001 16
15. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") NP*))) 3.94 0.002 12
16. (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* EX SQ**))) 3.92 0.003 36
17. (TOP (PRP "I") (VP (VBP "NEED") (NP (NP* DT NN) NP**)))) 3.85 0.003 14
18. (TOP (WP "WHAT") (SQ VBZ (NP-SBJ (DT "THE") (NN "PRICE")))) 3.79 0.002 20
19. (WHNP-1 (WDT "WHICH") (PP (IN "OF") (NP (DT "THESE") NNS))) 3.65 0.032 16
20. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (SBAR WDT VP**)))) 3.64 0.002 14
21. (TOP (VB "SHOW") (VP* PRP (NP (NP* DT NNS) (NP** PP-DIR PP-DIR*)))) 3.61 0.002 30
22. (TOP (WHNP (WDT "WHAT") NNS) (SQ (VBP "ARE") (SQ* (EX "THERE") (SQ** PP-DIR-3 PP-DIR-4)))) 3.30 0.002 26
23. (VP (MD "WOULD") (VP* (VB "LIKE") (VP** (TO "TO") (VP*** VB* VP****)))) 3.25 0.012 16
24. (TOP (WDT "WHICH") VP) 3.1460636 0.001646589 12
25. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NP**) NP***))) 3.03 0.001 12
26. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (NP*** PP-DIR PP-DIR*)))) 2.97 0.001 12
27. (PP (IN "OF") (NP* (NN* "FLIGHT") (NP** NNP (NP*** NNP* NP****)))) 2.95 0.015 8
28. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") (NNS "FARES")))) 2.85 0.001 8
29. (VP (VBP "NEED") (NP (NP* (DT "A") (NN "FLIGHT")) (NP** PP-DIR NP***))) 2.77 0.009 12
30. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (NP** PP-DIR PP-DIR*)))) 2.77 0.001 34
31. (TOP (JJS "CHEAPEST") (NN "FARE")) 2.74 0.001 6
32. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NP**) (NP*** PP-DIR PP-DIR*)))) 2.71 0.001 8
33. (TOP (NN "PRICE") (PP (IN "OF") (NP* (NN* "FLIGHT") (NP** NNP NP***)))) 2.69 0.001 6
34. (TOP (NN "PRICE") (PP (IN "OF") (NP* (NN* "FLIGHT") NP**))) 2.68 0.001 8
35. (PP-DIR (IN "FROM") (NP (NNP "WASHINGTON") (NP* (NNP* "D") (NNP** "C")))) 2.67 0.006 6
36. (PP-DIR (IN "FROM") (NP** (NNP "NEWARK") (NP*** (NNP* "NEW") (NNP** "JERSEY")))) 2.60 0.005 6
37. (S* (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") (VP** TO VP***)))) 2.59 0.11 8
38. (TOP (VBZ "DOES") (SQ* (NP-SBJ DT (NN "FLIGHT")) (VP (VB "SERVE") (NN* "DINNER")))) 2.48 0.002 8
39. (TOP (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") VP**))) 2.37 0.002 20
40. (TOP (WP "WHAT") (SQ (VBZ "IS") (NP-SBJ DT (NN "PRICE")))) 2.33 0.001 20
41. (S* (PRP "I") (VP MD (VP* (VB "LIKE") (VP** TO VP***)))) 2.33 0.100 8
42. (WHNP**** (PP-TMP (IN* "ON") (NNP** "FRIDAY")) (PP-LOC (IN** "ON") (NP (NNP*** "AMERICAN") (NNP**** "AIRLINES")))) 2.30 0.086 6
43. (VP* (PRP "ME") (NP (NP* (DT "THE") NNS) (NP** (PP-DIR IN NNP) (PP-DIR* TO NNP*)))) 2.29 0.007 18
44. (TOP (WHNP* (WDT "WHAT") (NNS "FLIGHTS")) (WHNP** (PP-DIR (IN "FROM") NNP) (WHNP*** (PP-DIR* TO NNP*) (PP-TMP IN* NNP**)))) 2.28 0.001 12
45. (SQ (VBP "ARE") (SQ* EX (SQ** (PP-DIR-3 IN NNP) (PP-DIR-4 TO NNP*)))) 2.26 0.015 14
46. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (SBAR WDT VP**)))) 2.22 0.001 8
47. (TOP (NNS "FLIGHTS") (NP* (PP-DIR (IN "FROM") (NP** NNP NNP*)) (PP-DIR* (TO "TO") NNP**))) 2.20 0.001 10)
48. ((VP (VBP "NEED") (NP (NP* (DT "A") (NN "FLIGHT")) (NP** (PP-DIR IN NNP) NP***))) 2.1346128 0.007185978 10)
49. ((NP (NP* (DT "THE") (NNS "FLIGHTS")) (NP** (PP-DIR (IN "FROM") (NNP "BALTIMORE")) (PP-DIR* (TO "TO") (NNP* "OAKLAND")))) 2.1335514 0.00381956 10)
50. ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (NP** PP-DIR NP***)))) 2.09 0.001 8)
```

Figure 1: Three examples and a list of the first 50 elementary trees with multiple words of an STSG induced using the push-n-pull algorithm on the ATIS3 corpus. For use in the current implementation, the parse trees have been converted to Chomsky Normal Form (all occurrences of $A \rightarrow B$, $B \rightarrow \omega$ are replaced by $A \rightarrow \omega$; all occurrences of $A \rightarrow BC\omega$ are replaced by $A \rightarrow BA*$, $A* \rightarrow C\omega$), all non-terminal labels are made unique for a particular parse tree (address labeling not shown) and all top nodes are replaced by the non-terminal "TOP". Listed are the elementary trees of the induced STSG with for each tree the score, the weight and the frequency with which it occurs in the training set.

corpus is generated by an STSG, and by inferring the properties of that underlying STSG. Given our best guess of the STSG that generated the data, we can start to ask questions like: which subtrees are overrepresented in the corpus? Which correlations are so strong that it is reasonable to think of the correlated phrases as a single unit? We presented a new algorithm for estimating weights of an STSG from a corpus, and reported promising empirical results on a small corpus.

Acknowledgments

The author is funded by the Netherlands Organisation for Scientific Research (Exacte Wetenschappen), project number 612.066.405. Many thanks to Yoav Seginer, Rens Bod and Remko Scha and the anonymous reviewers for very useful comments.

References

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-Oriented Parsing*. CSLI Publications, University of Chicago Press, Chicago, IL.

Rens Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proceedings EACL'93*, pages 37–44.

Rens Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI, Stanford, CA.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings EACL'03*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. *ACL'02*.

Adele E. Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. The University of Chicago Press, Chicago, IL.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings EMNLP'96*, p. 143–152.

C.T. Hemphill, J.J. Godfrey, and G.R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufman, Hidden Valley.

Ray Jackendoff. 2002. *Foundations of Language*. Oxford University Press, Oxford, UK.

Aravind Joshi and Anoop Sarkar. 2003. Tree adjoining grammars and their application to statistical parsing. In Bod et al. (Bod et al., 2003), pages 253–282.

A. Joshi, K. Vijay-Shanker, and D. Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Foundational issues in natural language processing*, pages 21–82. MIT Press, Cambridge MA.

P. Kay and C. Fillmore. 1999. Grammatical constructions and linguistic generalizations. *Language*, 75:1–33.

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.

Philip Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings COLING'92*, p. 418–424.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings CICLing*, pages 1–15.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere. http://iaaa.nl/rs/LeerdamE.html.

Yves Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings COLING'92*, pages 425–432.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings COLING'04*.

Khalil Sima'an and Luciano Buratto. 2003. Backoff parameter estimation for the DOP model. In *Proceedings ECML'03*, pages 373–384.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Michael Tomasello. 2000. The item-based nature of children's early syntactic development. *Trends in Cognitive Science*, 4(4):156–163.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*.

Willem Zuidema. 2003. How the poverty of the stimulus solves the poverty of the stimulus. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 51–58. MIT Press, Cambridge, MA.

Willem Zuidema. 2006. Theoretical evaluation of estimation methods for Data-Oriented Parsing. In *Proceedings EACL'06 (Conference Companion)*, pages 183–186.