A Derivational Model of Discontinuous Parsing

Mark-Jan Nederhof[1] and Anssi Yli-Jyrä[2]

[1] School of Computer Science, University of St Andrews, UK
[2] Department of Modern Languages, University of Helsinki, Finland

Abstract. The notion of latent-variable probabilistic context-free derivation of syntactic structures is enhanced to allow heads and unrestricted discontinuities. The chosen formalization covers both constituent parsing and dependency parsing. The derivational model is accompanied by an equivalent probabilistic automaton model. By the new framework, one obtains a probability distribution over the space of all discontinuous parses. This lends itself to intrinsic evaluation in terms of perplexity, as shown in experiments.

Keywords: parsing, grammars, weighted automata

1 Introduction

Much of traditional parsing theory considers a syntactic structure to be a tree in which siblings are linearly ordered, and a sentence is formed by the labels of the leaves from left to right. Whereas most English sentences can be given syntactic analyses that satisfy these constraints, other languages, especially those with more flexible word order such as German or Czech, do not let themselves be described easily, if at all, by using trees of this form. At the very least, these languages require types of syntactic trees with 'crossing edges', a phenomenon which is known formally as *discontinuity*.

In the theory of constituent parsing, leaf nodes in a parse tree are commonly words and punctuation tokens, and non-leaf nodes represent categories; one may also assign a special role to the nodes one level above the words, to represent parts of speech. In the theory of dependency parsing however, each node corresponds to a word or punctuation token, and can also be tagged with a part of speech; moreover, the parent-child edges are typically labeled by dependency relations. Discontinuity in dependency parsing is more specifically called *non-projectivity*.

One approach to obtaining discontinuous structures is to use formalisms that distinguish between derived trees and derivation trees, with discontinuity introduced through the interaction between the two kinds of trees. This approach has been explored in particular for tree adjoining grammars (TAGs) and linear context-free rewriting systems (LCFRSs). For TAG, see [10]. For LCFRS applied to dependency parsing see [12] and for LCFRS applied to constituent parsing see [5]. In all of these cases, probability models may remain attached to grammar rules, as in the case of traditional probabilistic context-free parsing [8].

Another approach takes shift-reduce automata as starting point, with an additional mechanism for swapping elements in the stack [18, 13]. Typically, there is a unique next parser action determined by a discriminative method, or there may be a probability distribution over a set of possible next actions. In the latter case, beam-search may be used to reduce the computational costs, by restricting attention to a bounded number of partial parses that locally seem most promising. In either case, parsing tends to be very fast.

However, it was shown for continuous parsing that models of syntax that rely on a probability distribution over actions of a shift-reduce parser are incomplete, in the sense that some probability distributions that can be expressed by probabilistic grammars cannot be expressed in terms of the corresponding automata [16]. Moreover, shift-reduce models tend to have many more parameters than grammatical models. Under certain conditions, this may lead to more accurate models [24] but when little training material is available, accurate estimation of the larger number of parameters may be infeasible. One way to deal with this is to derive the probabilities of parser actions directly from an underlying grammatical model [15].

The purpose of the current paper is to explore avenues towards similar theory for the discontinuous case. Concretely, we propose a grammatical, probabilistic model of discontinuous syntax, and show how this relates to an automaton model. This approach differs from an approach using TAG or LCFRS in that it retains a notion of immediate dominance that is context-free. Added to this is an independent model of discontinuity.

Our work has elements in common with, for example, [22, 11, 4], which also investigated discontinuity through a redefinition of context-free derivations. An important difference is that we aim to characterize 'typical' discontinuity in terms of a probabilistic model rather than in terms of a system of boolean constraints.

Other approaches, such as hybrid parsing [17], pseudo-projectivity [9, 14, 20], and the reversible splitting conversion of [2], are incomplete, in that allowed discontinuity is bounded by properties of the trained grammar, whereas our model provably allows for any discontinuity.

2 Trees

In the following, we define a type of trees that is able to represent both constituent structures and dependency structures with discontinuities. We assume that each substructure has a *head*. Heads are an inherent component of most definitions of dependency structures [7, 6]. Most older definitions of constituent structures avoid the notion of heads altogether, whereas some more recent literature tends to at least involve heads in some way [3].

Let $\mathbb{N}^+ = \{1, 2, \dots\}$, and let $[n] = \{1, 2, \dots, n\}$ for $n \in \mathbb{N}^+$. Let Σ be a finite set of *terminals* and let N be a finite set of *labels*. Terminals correspond naively to tokens, although in reality they can represent open classes of distributionally

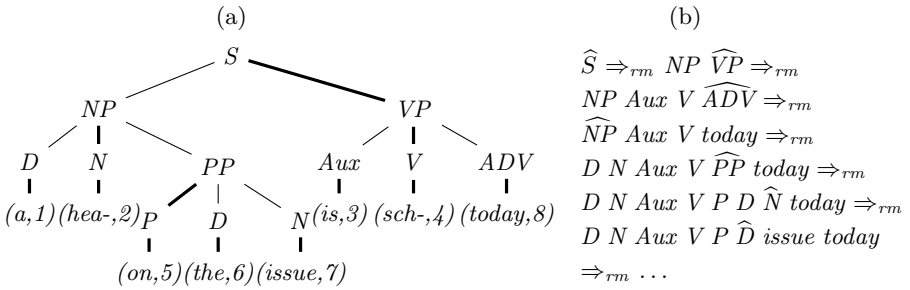

Fig. 1. Complete ph-tree for "*a hearing is scheduled on the issue today*" and corresponding rightmost derivation, assuming π is the identity function. Thick edges lead to heads and thin edges to dependents. Examples of headed rules used here are $NP \to D\langle N\rangle PP$ and $PP \to \varepsilon\langle P\rangle D\ N$ with ε here indicating absence of left dependents.

similar tokens. Labels could represent categories, parts of speech, semantic roles, or even a combination of these.

The set $H(\Sigma, N)$ of *headed trees* over Σ and N is defined inductively as follows. We have a *leaf* $(a, i) \in H(\Sigma, N)$ for each $a \in \Sigma$ and $i \in \mathbb{N}^+$. We also have $A(s_1 \cdots s_k, h, t_1 \cdots t_\ell) \in H(\Sigma, N)$ for each $A \in N$ and $s_1, \ldots, s_k, h, t_1, \ldots, t_\ell \in H(\Sigma, N)$. Nothing else is in $H(\Sigma, N)$.

In a leaf (a, i), the number i indicates an (input) position of the occurrence of a in a string of terminals. The set of all positions in a headed tree t is denoted by $pos(t)$. All of $s_1, \ldots, s_k$, h, $t_1, \ldots, t_\ell$ in a headed tree $t = A(s_1 \cdots s_k, h, t_1 \cdots t_\ell)$ will be referred to as *immediate subtrees* of t. We call subtree h the *head* of t, we call $s_1, \ldots, s_k$ its *left dependents* and $t_1, \ldots, t_\ell$ its *right dependents*.

The *head leaf* of $t = (a, i)$ is t itself and the head leaf of $t = A(s_1 \cdots s_k, h, t_1 \cdots t_\ell)$ is the head leaf of h. If (a, i) is the head leaf of t, then a is called the *head terminal* of t and i is called the *head position* of t.

A headed tree is a *positioned headed tree* (ph-tree) if no two leaves share the same position and if for every subtree $t = A(s_1 \cdots s_k, h, t_1 \cdots t_\ell)$, the sequence of the head positions of $s_1, \ldots, s_k$, h, $t_1, \ldots, t_\ell$ is strictly increasing. A ph-tree t is *complete* if $pos(t) = [n]$ for some $n \in \mathbb{N}^+$; see Figure 1(a) for an example. The set of ph-trees is denoted by $P(\Sigma, N)$, and the set of complete ph-trees by $C(\Sigma, N)$. The *yield* of a ph-tree whose leaves are (a_1, k_1), ..., (a_m, k_m), arranged such that $k_1 < \ldots < k_m$, is the string $a_1 \cdots a_m$.

We call a ph-tree *unilexical* if each subtree is either a leaf or of the form $A(s_1 \cdots s_k, h, t_1 \cdots t_\ell)$, where h is a leaf and none of $s_1, \ldots, s_k$, $t_1, \ldots, t_\ell$ are leaves. A complete ph-tree that is unilexical would more commonly be called a dependency structure.

3 Headed Context-free Grammars

In this section, we give a formalization of headed grammars that differs somewhat from related definitions in the literature, for example those of [1, 3]. This is

motivated by the need for a streamlined presentation that allows formulation of both discontinuous constituent parsing and non-projective dependency parsing. We also wish to incorporate the notion of latent variables, as this is an essential component of some state-of-the-art parsers [21].

A *headed context-free grammar* (HCFG) is a 6-tuple $(\Sigma, Q, q_{init}, R, N, \pi)$, where Σ is a finite set of terminals as before, Q is a finite set of *states*, $q_{init} \in Q$ is the *initial state*, and R is a set of *headed rules*. Also as before, N is a finite set of labels. The function π maps states to labels. Several states may be mapped to the same label, to accommodate for latent variables. However, in the examples and in the current experiments (Section 7), π is always the identity function.

A headed rule has the form $q \to \alpha\langle Z\rangle\beta$, where $q \in Q$, $Z \in \Sigma \cup Q$ and $\alpha, \beta \in (\Sigma \cup Q)^*$. Here q is called the *left-hand side*, and $\alpha\langle Z\rangle\beta$ the *right-hand side*, in which Z is the *head*, and the symbols in α and β are the left and right dependents, respectively. The set R of headed rules is potentially infinite, in which case we assume finite descriptions for the sets of strings α and β that may appear in rules of the form $q \to \alpha\langle Z\rangle\beta$, for given q and Z. Such descriptions would typically be finite automata.

Assuming a fixed HCFG, the binary relation $\Rightarrow$ has $\gamma_0\gamma_1 \cdots \gamma_k q \delta_0 \delta_1 \cdots \delta_\ell \Rightarrow \gamma_0 X_1 \gamma_1 X_2 \cdots X_k \gamma_k Z \delta_0 Y_1 \delta_1 Y_2 \cdots Y_\ell \delta_\ell$ if $q \to X_1 \cdots X_k\langle Z\rangle Y_1 \cdots Y_\ell$ is a rule. Note that if we were to restrict $\gamma_1 \cdots \gamma_k$ and $\delta_0 \cdots \delta_{\ell-1}$ to be always ε, then we obtain the familiar notion of (continuous) context-free derivation. The reflexive, transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$.

Where we speak of '*a derivation* $q_{init} \Rightarrow^* w$', we implicitly assume a certain sequence of rule applications that leads us from q_{init} to w, via intermediate *sentential forms*. This includes not only the identity of each applied rule, but also the occurrence of the state on which it is applied (a sentential form may contain several occurrences of the same state), and the locations where the left and right dependents are placed among the existing elements in the sentential form.

A derivation $q_{init} \Rightarrow^* w$ maps to a complete ph-tree as follows. First we enhance $w = a_1 \cdots a_n$ to $w' = (a_1, 1) \cdots (a_n, n)$, so that each terminal is coupled to its position in the derived string. In the same way, we construct a set of enhanced rules, on the basis of the rules that occur in the derivation. Concretely, for a rule ρ of the form $q \to X_1 \cdots X_k\langle Z\rangle Y_1 \cdots Y_\ell$, the set of enhanced rules contains all rules of the form $(q, i) \to (X_1, i_1) \cdots (X_k, i_k)\langle(Z, i)\rangle(Y_1, j_1) \cdots (Y_\ell, j_\ell)$, where $1 \leq i_1 < \ldots < i_k < i < j_1 < \ldots < j_\ell \leq n$. The set of such enhanced rules for given rule ρ will be denoted by $\rho^{(n)}$. Examples of enhanced rules relevant for Figure 1 are $(S, 4) \to (NP, 2)\langle(VP, 4)\rangle\varepsilon$ and $(D, 6) \to \varepsilon\langle(the, 6)\rangle\varepsilon$. We can now extend the derivation $q_{init} \Rightarrow^* w$ in a unique way to a derivation $(q_{init}, i_0) \Rightarrow^* w'$, for some i_0, where we replace an application of a rule ρ by some rule $\rho' \in \rho^{(n)}$. In the enhanced derivation, we have made explicit in which position each terminal occurrence will end up, as well as made explicit the head position belonging to each state occurrence.

Next, we interpret the enhanced derivation as a tree structure, with the right-hand side elements of an enhanced rule being the children of the left-hand side.

On this tree structure, we apply π, which amounts to replacing each (q, i) by $\pi(q)$. In particular, if π is the identity function, as in the case of Figure 1, then each (q, i) is simply replaced by q. Hereby, a derivation maps to a unique string w as well as to a unique complete ph-tree. For a given HCFG G, the string language $SL(G)$ generated by G is the set of all strings that can be derived and the tree language $TL(G)$ is the corresponding set of complete ph-trees.

The *permutation closure* of a string language $L \subseteq \Sigma^*$ is defined to be the language $perm(L)$ of strings that are permutations of a string in L. We extend permutation closure to rules as follows. If ρ is a rule $q \to \alpha\langle Z\rangle\beta$, then $perm(\rho)$ is the set of rules of the form $q \to \alpha'\langle Z\rangle\beta'$, where $\alpha' Z \beta'$ is a permutation of $\alpha Z \beta$; note that α' and β' need not be of the same length as α and β. The permutation closure of a HCFG G, denoted by $perm(G)$, is obtained by replacing its set of rules by the union of their permutation closures. It is easy to see that a language is a permutation closure of an epsilon-free context-free language (and thereby of an epsilon-free regular language) if and only if it is $SL(perm(G))$ for some HCFG G.

In practice, it is undesirable for a model of natural language syntax to allow indiscriminate permutation of left dependents or of right dependents, let alone indiscriminate swapping of left and right dependents. This motivates considering the following weaker alternative to permutation closure. It involves shuffling the descendents of a node with descendents of other nodes, while preserving the relative order of immediate subtrees. Formally, a complete ph-tree u_2 is a *shuffling* of a complete ph-tree u_1 if their yields, both of length n, are equal under some permutation f of positions in $[n]$ and if for every subtree of u_1 of the form $A(s_1 \cdots s_k, h, t_1 \cdots t_\ell)$, whose immediate subtrees have head positions $i_1, \ldots, i_k, i, j_1, \ldots, j_\ell$, there is a corresponding subtree of u_2 of the form $A(s'_1 \cdots s'_k, h', t'_1 \cdots t'_\ell)$, whose immediate subtrees have head positions $f(i_1), \ldots, f(i_k), f(i), f(j_1), \ldots, f(j_\ell)$. The *shuffle closure* of a set T of complete ph-trees, denoted by $shuffle(T)$, is obtained by replacing every tree $t \in T$ by the set of its shufflings. It is easy to see that $TL(G) = shuffle(TL_c(G))$ for every HCFG G, where $TL_c(G)$ is the tree language that results if $\Rightarrow$ is restricted to continuous derivation.

We have thus seen two ways of relating HCFG to continuous context-free grammar, one in terms of string languages, using the permutation closure, and one in terms of tree languages, using the shuffle closure.

4 Leftmost and Rightmost Derivations

Just as in established theory of context-free grammars, there can be several derivations for the same string and the same tree that differ only in the order in which states are rewritten. For the purpose of designing effective parsers and formulating consistent probability distributions, one traditionally restricts derivations to be either leftmost or rightmost. The behavior of top-down parsers most closely matches leftmost derivations, whereas bottom-up parsers typically match (reversed) rightmost derivations.

Restricting our discontinuous derivations to be either leftmost or rightmost is more involved than in established theory, because of the potentially non-local behavior of derivation steps. Although we could define leftmost and rightmost derivations for arbitrary HCFGs, the definitions become simpler if we restrict ourselves to HCFGs that are separated; a HCFG is called *separated* if terminals only occur in rules of the form $q \to \varepsilon\langle a\rangle\varepsilon$, where $a \in \Sigma$, and all other rules are of the form $q \to \alpha\langle r\rangle\beta$, where $r \in Q$ and $\alpha, \beta \in Q^*$. In a rightmost derivation, every sentential form is then split into a prefix and a suffix. The suffix consists entirely of terminals, whereas the prefix is in the set $\{\varepsilon\} \cup Q^*\hat{Q}Q^*$, where $\hat{Q} = \{\hat{q} \mid q \in Q\}$. In words, if the prefix is not the empty string, then it contains only states, of which exactly one has a hat.

The binary relation $\Rightarrow_{rm}$ ('rm' for 'rightmost') has $\gamma\hat{q}\delta w \Rightarrow_{rm} \gamma_0 q'_1\gamma_1 q'_2 \cdots q'_k\gamma_k r'\delta_0 s'_1\delta_1 s'_2 \cdots s'_\ell\delta_\ell w$ if $q \to q_1 \cdots q_k\langle r\rangle s_1 \cdots s_\ell$ is a rule, $\gamma = \gamma_0 \cdots \gamma_k \in Q^*$, $\delta = \delta_0 \cdots \delta_\ell \in Q^*$, $w \in \Sigma^*$, $q, q_1, \ldots, q_k, r, s_1, \ldots, s_\ell \in Q$, and $q'_1 \cdots q'_k r' s'_1 \cdots s'_\ell$ is obtained from $q_1 \cdots q_k r s_1 \cdots s_\ell$ by placing the hat on exactly one of these states.

The relation $\Rightarrow_{rm}$ further has $\gamma\hat{q}w \Rightarrow_{rm} \gamma' aw$ if $q \to \varepsilon\langle a\rangle\varepsilon$ is a rule, where $a \in \Sigma$, and $\gamma' = \gamma$ if $\gamma = \varepsilon$ and otherwise γ' is obtained from γ by placing the hat on exactly one of the states. A derivation starts from the sentential form $\widehat{q_{init}}$.

Perhaps counter-intuitively at first sight, our rightmost derivations do not necessarily rewrite the rightmost state. Instead, a rightmost derivation can be decomposed into several chains of rewrites, each of which ends in a step that is rightmost in the sense of adding one more terminal at the front of the suffix of terminals. After the first step in a single chain, each step rewrites a state introduced by the previous step. This constraint is enforced by placing a hat on a state to be rewritten next.

An example of a chain starts in the third line in Figure 1(b), with $\widehat{NP}$. In the fourth line, NP has been rewritten to $D\ N\ PP$, of which the last obtains the hat. The chain ends when *issue* is added at the front of the terminal suffix, upon which another state from the sentential form obtains the hat, in this case the rightmost D, which starts a new chain.

Leftmost derivations are analogous. We can define SL_{lm}, TL_{lm}, SL_{rm} and TL_{rm} much as we defined SL and TL, now restricting the derivations to be leftmost or rightmost, respectively. Of central importance to later sections is:

Theorem 1 *For each HCFG G, we have $SL(G) = SL_{lm}(G) = SL_{rm}(G)$, and $TL(G) = TL_{lm}(G) = TL_{rm}(G)$.*

A proof can be given in terms of enhanced derivations, which can be rearranged to become rightmost (or leftmost for the symmetric case). For a string of length n, there are chains of rewrites, one for each $i = n, \ldots, 1$. In the chain for a certain i, only state occurrences are rewritten whose corresponding subtrees have yields that include the i-th terminal but not the j-th terminal in the string, for any $j > i$. Theorem 1 can be refined to formulate a surjective mapping from an arbitrary derivation to a rightmost derivation for the same tree. Moreover, if

π is the identity function, then there is a bijective mapping from rightmost derivations to $TL(G)$.

5 The Automaton Model

This section defines a *discontinuous* shift-reduce parser, which computes (reversed) rightmost derivations of a separated HCFG. A *configuration* is a pair consisting of a *stack*, which is a string in $\{\varepsilon\} \cup Q^*\hat{Q}Q^*$, and a *remaining input*, which is a string in Σ^*.

The binary relation $\vdash$ is defined by two allowable steps. A *shift* is $(\gamma, aw) \vdash (\gamma'\hat{q}, w)$ if $q \to \varepsilon\langle a\rangle\varepsilon$ is a rule, and γ' results from γ by removing the occurrence of the hat if there is one (if not, then $\gamma = \varepsilon$). A *reduction* is $(\gamma_0 q'_1 \gamma_1 q'_2 \cdots q'_k \gamma_k r' \delta_0 s'_1 \delta_1 s'_2 \cdots s'_\ell \delta_\ell, w) \vdash (\gamma_0 \cdots \gamma_k \hat{q} \delta_0 \cdots \delta_\ell, w)$ if $q \to q_1 \cdots q_k \langle r\rangle s_1 \cdots s_\ell$ is a rule, and $q'_1 \cdots q'_k r' s'_1 \cdots s'_\ell$ contains exactly one hat, which is removed to give $q_1 \cdots q_k r$ $s_1 \cdots s_\ell$. It differs from the usual definition of reduction in continuous parsing by the fact that the occurrences of symbols in the right-hand side of the used rule can be arbitrarily deep in the stack (but in the same relative order as in the rule). The location in the stack where the head is found determines the location of the left-hand side of the rule after the reduction.

A *computation* recognizing a string w is a sequence of steps $(\varepsilon, w) \vdash^* (\widehat{q_{init}}, \varepsilon)$. As $\vdash$ can be seen as the reversal of $\Rightarrow_{rm}$, it is not difficult to see that a string can be recognized if and only if it is in $SL(G)$, using Theorem 1. Further, computations can be enhanced to construct corresponding trees.

For the example from Figure 1, the computation is:

$$
\begin{array}{lll}
(\varepsilon & , \textit{a hearing is scheduled on the issue today}) \vdash \\
(\widehat{D} & , \textit{hearing is scheduled on the issue today}) \vdash \\
(D\ \widehat{N} & , \textit{is scheduled on the issue today}) \vdash \ldots \vdash \\
(D\ N\ Aux\ V\ P\ \widehat{D} & , \textit{issue today}) \vdash \\
(D\ N\ Aux\ V\ P\ D\ \widehat{N} & , \textit{today}) \vdash \\
(D\ N\ Aux\ V\ \widehat{PP} & , \textit{today}) \vdash \\
(\widehat{NP}\ Aux\ V & , \textit{today}) \vdash \\
(NP\ Aux\ V\ \widehat{ADV} & , \varepsilon) \vdash \\
(NP\ \widehat{VP} & , \varepsilon) \vdash \\
(\widehat{S} & , \varepsilon)
\end{array}
$$

6 Probabilities

There may be many derivations for a given string, each of which determines one tree. In order to disambiguate, and choose one out of those derivations, and thereby one tree, one may impose a probability model, either on steps of the automaton, or on derivation steps. For reasons explained in Section 1, we here want to investigate the latter, as before for a fixed HCFG that is separated.

The task ahead is to define a probability distribution over the derivation steps that are possible for a sentential form $\gamma\hat{q}\delta w$. A derivation step is characterized first by a choice of a rule $q \to \alpha\langle r\rangle\beta$ or a rule $q \to \varepsilon\langle a\rangle\varepsilon$; the latter is only possible if $\delta = \varepsilon$. When a non-lexical rule $q \to \alpha\langle r\rangle\beta$ is applied, we also need to choose the way in which γ and δ are broken up, to accommodate for the placement of the elements of α and β, and we need to choose the state among those in $\alpha r\beta$ that will have the hat next. When a lexical rule $q \to \varepsilon\langle a\rangle\varepsilon$ is applied, we need to choose the state among those in the sentential form that will have the hat next.

In order to be able to estimate parameters effectively, we need to make a number of independence assumptions, which will become clear in the following. The probability of obtaining $\alpha_2 = \gamma_0 q'_1 \gamma_1 \cdots q'_k \gamma_k r' \delta_0 s'_1 \delta_1 \cdots s'_\ell \delta_\ell w$ in one step from $\alpha_1 = \gamma_0 \cdots \gamma_k \hat{q} \delta_0 \cdots \delta_\ell w$ through application of $q \to q_1 \cdots q_k \langle r\rangle s_1 \cdots s_\ell$, where the m-th state in $q'_1 \cdots q'_k r' s'_1 \cdots s'_\ell$ is the one with the hat, and B denoting the truth value of $\delta_0 \cdots \delta_\ell = \varepsilon$, can be approximated by:

$$\begin{aligned} p(\alpha_2 \mid \alpha_1) \approx\ & p_{rule}(q \to q_1 \cdots q_k \langle r\rangle s_1 \cdots s_\ell \ \mid\ q, B) \ \cdot \\ & p_{left}(|\gamma_0|, \ldots, |\gamma_k| \ \mid\ k, |\gamma_0 \cdots \gamma_k|) \ \cdot \\ & p_{right}(|\delta_0|, \ldots, |\delta_\ell| \ \mid\ \ell, |\delta_0 \cdots \delta_\ell|) \ \cdot \\ & p_{rule\_hat}(m \ \mid\ k, \ell) \end{aligned} \tag{1}$$

The use of p_{rule} embodies the independence assumption that the probability of applying a non-lexical rule for state q does not depend on the context, except for the question whether q is the rightmost state in the sentential form. This is because probability mass may be shared with application of a lexical rule in case $B = \mathbf{true}$, which is to be discussed later.

Further, $p_{left}(i_0, \ldots, i_k \mid k, i)$ is the probability that k left dependents are distributed over $i = i_0 + \ldots + i_k$ states, leaving i_0 states before the first left dependent, i_k after the last left dependent, and i_1, …, i_{k-1} states between the corresponding pairs of consecutive left dependents. The meaning of $p_{right}(j_0, \ldots, j_\ell \mid \ell, j)$ is analogous. The assumption made here is that the probability of how the left and right dependents are interspersed with the existing states of the sentential form is independent of the identities of the involved states, and only their numbers matter. The motivation behind this assumption is to keep the model simple. Investigation of more refined models is a matter for future research.

Lastly, $p_{rule\_hat}(m \mid k, \ell)$ is the probability that with k left dependents, one head, and ℓ right dependents, the hat is placed on the m-th element, with $m \in [k + 1 + \ell]$. Once more, there is the independence assumption that the identities of the involved states do not matter.

The probability of obtaining $\alpha_2 = \gamma' a w$ from $\alpha_1 = \gamma \hat{q} w$ using a rule $q \to \varepsilon\langle a\rangle\varepsilon$, where the m-th state in γ' is the one with the hat, can be approximated by:

$$p(\alpha_2 \mid \alpha_1) \approx p_{rule}(q \to \varepsilon\langle a\rangle\varepsilon \ \mid\ q) \cdot p_{lex\_hat}(m \ \mid\ |\gamma|) \tag{2}$$

Here p_{rule} is implicitly conditional on $B = \mathbf{true}$, as lexical rules can only rewrite states that occur rightmost in the sentential form. Further, $p_{lex\_hat}(m \mid k)$ is the

probability that the hat is placed on the m-th state of a string of k states. Once again, there is the independence assumption that the identities of the involved states do not matter.

We define p_{left} and p_{right} recursively, motivated by the assumption that the probability decreases exponentially with the number of existing states from the sentential form that are interspersed with the dependents. For $i \geq 0$ and $k \geq 0$:

$$p_{left}(i_0 \mid 0, i) = p_{left}(i_0, \ldots, i_k \mid k, 0) = 1 \tag{3}$$

$$p_{left}(i_0, \ldots, i_k, i_{k+1} + 1 \mid k + 1, i + 1) = P_{left} \cdot p_{left}(i_0, \ldots, i_k, i_{k+1} \mid k + 1, i) \tag{4}$$

$$p_{left}(i_0, \ldots, i_k, 0 \mid k + 1, i + 1) = (1 - P_{left}) \cdot p_{left}(i_0, \ldots, i_k \mid k, i + 1) \tag{5}$$

Here P_{left} is the probability that the next dependent is *not* placed rightmost among the available existing states in the sentential form, provided there are any. We have probability 1 if there are no more available states in the sentential form that can be skipped, or no more dependents. We define p_{right} in the same way, with constant P_{right}. By the same reasoning, we define $p_{rule\_hat}(m \mid k, \ell) = P_{rule\_hat}^{k+\ell+1-m} \cdot (1 - P_{rule\_hat})$ for $m > 1$ and $p_{rule\_hat}(m \mid k + \ell + 1) = P_{rule\_hat}^{k+\ell+1-m}$ for $m = 1$. Here P_{rule_hat} can be seen as the probability that the hat is *not* placed on the next available state of a rule, from right to left, if there are at least two more available states. We have probability 1 if there is only one more state. We define p_{lex_hat} similarly, with constant P_{lex_hat}, which can be seen as the probability that the hat is *not* placed on the next available state in the sentential form, from right to left.

With the usual assumption of absence of useless rules, a sufficient condition for the above equations to specify a consistent probability model is that there is at least one rule $q \to \alpha\langle r\rangle\beta$ with non-empty β for each q. To illustrate the problem that is potentially caused if this requirement is not satisfied, assume the hat is placed on a state q in the sentential form that is not rightmost, and assume only lexical rules exist that have q as left-hand side. Then no rules at all are applicable, and probability mass is lost.

Such a problem in fact had to be solved for our experiments in Section 7 with dependency grammars, which are formalized in terms of rules $q_{init} \to \varepsilon\langle\overline{A}\rangle\varepsilon$, $\overline{A} \to \alpha\langle A\rangle\beta$, and $A \to \varepsilon\langle a\rangle\varepsilon$, where state A represents a part of speech and $\overline{A}$ is an auxiliary state for the same part of speech, α and β are strings of such auxiliary states for parts of speech, and a is a word. Dependency relations are ignored. For each A there is one smoothed bigram model p_{ld} for possible choices of left dependents α and one such model p_{rd} for right dependents β. To avoid problems caused by out-of-vocabulary words and inflection, probabilities of rules $A \to \varepsilon\langle a\rangle\varepsilon$ are ignored, so that conceptually the input consists of a string of parts of speech.

In order to then obtain a probability distribution over derivations, one needs to ensure that the hat cannot be given to a part of speech that is non-rightmost in the sentential form. This is achieved by adjusting the definitions of p_{rule_hat} and p_{lex_hat} to ignore parts of speech unless they are right-most in the sentential form. We further ensure that an auxiliary state $\overline{A}$ that occurs non-rightmost in the sentential form can only be rewritten using $\overline{A} \to \alpha\langle A\rangle\beta$ if $\alpha\beta \neq \varepsilon$,

Table 1. Perplexity of the likelihood p, and some estimated parameters, for German, Norwegian, English and Hindi, for leftmost and for rightmost derivations.

	$\Rightarrow_{lm}$					$\Rightarrow_{rm}$				
	p	P_{lex_hat}	P_{rule_hat}	P_{left}	P_{right}	p	P_{lex_hat}	P_{rule_hat}	P_{left}	P_{right}
Ge	1.88	$4.0*10^{-3}$	$2.0*10^{-3}$	$1.3*10^{-1}$	$4.2*10^{-2}$	1.89	$6.0*10^{-4}$	$1.1*10^{-2}$	$1.0*10^{-2}$	$2.8*10^{-1}$
No	1.96	$5.8*10^{-4}$	$3.4*10^{-3}$	$5.0*10^{-1}$	$8.5*10^{-2}$	1.97	$2.1*10^{-3}$	$1.7*10^{-2}$	$5.1*10^{-3}$	$2.1*10^{-1}$
En	1.99	$4.8*10^{-4}$	$3.8*10^{-3}$	$5.2*10^{-1}$	$3.7*10^{-2}$	2.00	$1.1*10^{-3}$	$6.5*10^{-3}$	$4.5*10^{-3}$	$2.3*10^{-1}$
Hi	1.72	$2.3*10^{-3}$	$3.6*10^{-3}$	$5.7*10^{-1}$	$4.1*10^{-2}$	1.71	$3.9*10^{-4}$	$2.6*10^{-2}$	$9.6*10^{-3}$	$4.6*10^{-1}$

with probability $p_{ld}(\alpha \mid A) \cdot p_{rd}(\beta \mid A) \cdot C_{norm}(A)$, with the normalization factor $C_{norm}(A) = 1 \,/\, (1 - p_{ld}(\varepsilon \mid A) \cdot p_{rd}(\varepsilon \mid A))$. For rightmost occurrences of $\overline{A}$ there is no such restriction on α and β and the normalization factor is not needed.

7 Evaluation

The number of derivations is exponential in the length of a sentence. This makes it infeasible to compute the most probable among all rightmost (or leftmost) derivations. One may use beam search or related pruning techniques to reduce running time. However, an extrinsic evaluation that determines the usual F1 score (combining precision and recall) would then say as much about the used techniques of pruning as it does about the underlying model.

Because our model defines a probability distribution, one may instead perform an intrinsic evaluation in terms of perplexity, which is the negative log likelihood of a test corpus, normalized by the size of that corpus, i.e. $\frac{-\sum_{t \in T} \log_2 p(t)}{\sum_{t \in T} |t|}$. Here T is the set of trees for the test corpus, p is the trained probability model of leftmost or rightmost derivations, and $|t|$ is the number of nodes in t. Perplexity was shown by [23] to be a good indicator of parsing accuracy.

An obvious question to investigate is whether there is a difference in perplexity between leftmost and rightmost derivations. For this, we considered four corpora from the Universal Dependencies treebank [19], taking the first 13000 trees from each training section and the first 950 trees from each testing section. All punctuation was removed and sentences consisting entirely of punctuation were ignored altogether. Table 1 presents perplexity and some of the parameters obtained by maximum likelihood estimation.

Note that the probabilities of p_{rule}, as determined by p_{ld} and p_{rd} and the probabilities of rules $q_{init} \to \varepsilon\langle\overline{A}\rangle\varepsilon$, are not affected by the direction of the derivations. They make the biggest contribution to the perplexity, so that total values for leftmost and right derivations become very similar. Table 2 therefore looks at the decomposition of the perplexity, into the contributions from p_{rule}, p_{lex_hat}, p_{rule_hat}, from p_{left} and p_{right} together, as well as the (negative) contribution from the normalization factor C_{norm}. Also this more detailed view does not reveal a clear preference for leftmost or rightmost derivations.

Table 2. Perplexity of the likelihood p decomposed.

		$\Rightarrow_{lm}$					$\Rightarrow_{rm}$				
	p_{rule}	p	p_{lex_hat}	p_{rule_hat}	p_{left}*p_{right}	C_{norm}	p	p_{lex_hat}	p_{rule_hat}	p_{left}*p_{right}	C_{norm}
Ge	1.86	1.88	$5.4*10^{-3}$	$2.9*10^{-3}$	$1.5*10^{-2}$	$-3.0*10^{-4}$	1.89	$8.3*10^{-3}$	$1.3*10^{-2}$	$1.2*10^{-2}$	$-1.6*10^{-3}$
No	1.92	1.96	$4.6*10^{-3}$	$7.0*10^{-3}$	$2.9*10^{-2}$	$-3.4*10^{-4}$	1.97	$1.8*10^{-2}$	$1.8*10^{-2}$	$2.1*10^{-2}$	$-5.5*10^{-3}$
En	1.98	1.99	$1.7*10^{-3}$	$1.1*10^{-3}$	$8.9*10^{-3}$	$-1.8*10^{-4}$	2.00	$4.6*10^{-3}$	$6.7*10^{-3}$	$5.5*10^{-3}$	$-1.4*10^{-3}$
Hi	1.67	1.72	$1.1*10^{-2}$	$6.4*10^{-3}$	$3.4*10^{-2}$	$-5.3*10^{-4}$	1.71	$1.7*10^{-3}$	$2.3*10^{-2}$	$2.0*10^{-2}$	$-1.7*10^{-3}$

8 Conclusion

Motivated by the ultimate aim of developing more accurate and robust parsers that can handle discontinuity, we have introduced a model of syntax that captures discontinuous derivation. Unlike previous models, it explicitly defines a probability distribution over the space of all discontinuous parses. We have shown that this allows evaluation in terms of perplexity.

References

1. Alshawi, H.: Head automata and bilingual tiling: Translation with minimal representations. In: 34th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. pp. 167–176. Santa Cruz, California, USA (Jun 1996)
2. Boyd, A.: Discontinuity revisited: An improved conversion to context-free representations. In: Proceedings of the Linguistic Annotation Workshop, at ACL 2007. pp. 41–44. Prague, Czech Republic (Jun 2007)
3. Collins, M.: Head-driven statistical models for natural language parsing. Computational Linguistics 29(4), 589–637 (2003)
4. Daniels, M., Meurers, W.: Improving the efficiency of parsing with discontinuous constituents. In: Wintner, S. (ed.) Proceedings of NLULP'02: The 7th International Workshop on Natural Language Understanding and Logic Programming. Datalogiske Skrifter, vol. 92, pp. 49–68. Roskilde Universitetscenter, Copenhagen (2002)
5. Evang, K., Kallmeyer, L.: PLCFRS parsing of English discontinuous constituents. In: Proceedings of the 12th International Conference on Parsing Technologies. pp. 104–116. Dublin, Ireland (Oct 2011)
6. Gaifman, H.: Dependency systems and phrase-structure systems. Information and Control 8, 304–337 (1965)
7. Hays, D.: Dependency theory: A formalism and some observations. Language 40(4), 511–525 (1964)
8. Jelinek, F., Lafferty, J., Mercer, R.: Basic methods of probabilistic context free grammars. In: Laface, P., De Mori, R. (eds.) Speech Recognition and Understanding — Recent Advances, Trends and Applications, pp. 345–360. Springer-Verlag (1992)
9. Kahane, S., Nasr, A., Rambow, O.: Pseudo-projectivity, a polynomially parsable non-projective dependency grammar. In: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. vol. 1, pp. 646–652. Montreal, Quebec, Canada (Aug 1998)

10. Kallmeyer, K., Kuhlmann, M.: A formal model for plausible dependencies in lexicalized tree adjoining grammar. In: Eleventh International Workshop on Tree Adjoining Grammar and Related Formalisms. pp. 108–116 (2012)
11. Kathol, A., Pollard, C.: Extraposition via complex domain formation. In: 33rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. pp. 174–180. Cambridge, Massachusetts, USA (Jun 1995)
12. Kuhlmann, M.: Mildly non-projective dependency grammar. Computational Linguistics 39(2), 355–387 (2013)
13. Maier, W.: Discontinuous incremental shift-reduce parsing. In: 53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing. vol. 1, pp. 1202–1212. Beijing (Jul 2015)
14. McDonald, R., Pereira, F.: Online learning of approximate dependency parsing algorithms. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics. pp. 81–88. Trento, Italy (2006)
15. Nederhof, M.J., McCaffery, M.: Deterministic parsing using PCFGs. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics. pp. 338–347. Gothenburg, Sweden (2014)
16. Nederhof, M.J., Satta, G.: An alternative method of training probabilistic LR parsers. In: 42nd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. pp. 551–558. Barcelona, Spain (Jul 2004)
17. Nederhof, M.J., Vogler, H.: Hybrid grammars for discontinuous parsing. In: The 25th International Conference on Computational Linguistics: Technical Papers. pp. 1370–1381. Dublin, Ireland (Aug 2014)
18. Nivre, J.: Non-projective dependency parsing in expected linear time. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. pp. 351–359. Suntec, Singapore (Aug 2009)
19. Nivre, J.: Towards a universal grammar for natural language processing. In: Computational Linguistics and Intelligent Text Processing, 16th International Conference. Lecture Notes in Computer Science, vol. 9041, pp. 3–16. Springer-Verlag, Cairo, Egypt (2015)
20. Nivre, J., Nilsson, J.: Pseudo-projective dependency parsing. In: 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. pp. 99–106. Ann Arbor, Michigan (Jun 2005)
21. Petrov, S., Barrett, L., Thibaux, R., Klein, D.: Learning accurate, compact, and interpretable tree annotation. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. pp. 433–440. Sydney, Australia (Jul 2006)
22. Reape, M.: A logical treatment of semi-free word order and bounded discontinuous constituency. In: Fourth Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference. pp. 103–110. Manchester, England (Apr 1989)
23. Søgaard, A., Haulrich, M.: On the derivation perplexity of treebanks. In: Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories. pp. 223–232. Tartu, Estonia (Dec 2010)
24. Sornlertlamvanich, V., Inui, K., Tanaka, H., Tokunaga, T., Takezawa, T.: Empirical support for new probabilistic generalized LR parsing. Journal of Natural Language Processing 6(3), 3–22 (1999)