Unlexicalized Transition-based Discontinuous Constituency Parsing

Maximin Coavoux[1*], **Benoît Crabbé**[2,3], **Shay B. Cohen**[1]

[1]University of Edinburgh, ILCC
[2]Université Paris Diderot, Université Sorbonne Paris Cité, LLF
[3]Institut Universitaire de France (IUF)
{mcoavoux,scohen}@inf.ed.ac.uk
bcrabbe@linguist.univ-paris-diderot.fr

Abstract

Lexicalized parsing models are based on the assumptions that (i) constituents are organized around a lexical head and (ii) bilexical statistics are crucial to solve ambiguities. In this paper, we introduce an *unlexicalized* transition-based parser for discontinuous constituency structures, based on a structure-label transition system and a bi-LSTM scoring system. We compare it with lexicalized parsing models in order to address the question of lexicalization in the context of discontinuous constituency parsing. Our experiments show that unlexicalized models systematically achieve higher results than lexicalized models, and provide additional empirical evidence that lexicalization is not necessary to achieve strong parsing results. Our best unlexicalized model sets a new state of the art on English and German discontinuous constituency treebanks. We further provide a per-phenomenon analysis of its errors on discontinuous constituents.

1 Introduction

This paper introduces an unlexicalized parsing model and addresses the question of lexicalization, as a parser design choice, in the context of transition-based discontinuous constituency parsing. Discontinuous constituency trees are constituency trees where crossing arcs are allowed in order to represent long-distance dependencies, and in general phenomena related to word order variations (e.g., the left dislocation in Figure 1).

Lexicalized parsing models (Collins, 1997; Charniak, 1997) are based on the assumptions that (i) constituents are organized around a lexical head and (ii) bilexical statistics are crucial to solve ambiguities. In a lexicalized Probabilistic Context-Free Grammar (PCFG), grammar rules involve nonterminals annotated with a terminal element that represents their lexical head, for example:

$$\text{VP[saw]} \longrightarrow \text{VP[saw]}\ \ \text{PP[telescope]}.$$

The probability of such a rule models the likelihood that *telescope* is a suitable modifier for *saw*.

In contrast, unlexicalized parsing models renounce modeling bilexical statistics, based on the assumptions that they are too sparse to be estimated reliably. Indeed, Gildea (2001) observed that removing bilexical statistics from Collins' (1997) model lead to at most a 0.5 drop in F-score. Furthermore, Bikel (2004) showed that bilexical statistics were in fact rarely used during decoding, and that when used, they were close to that of back-off distributions used for unknown word pairs.

Instead, unlexicalized models may rely on grammar rule refinements to alleviate the strong independence assumptions of PCFGs (Klein and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006; Narayan and Cohen, 2016). They sometimes rely on structural information, such as the boundaries of constituents (Hall et al., 2014; Durrett and Klein, 2015; Cross and Huang, 2016b; Stern et al., 2017; Kitaev and Klein, 2018).

Although initially coined for chart parsers, the notion of lexicalization naturally transfers to transition-based parsers. We take *lexicalized* to denote a model that (i) assigns a lexical head to each constituent and (ii) uses heads of constituents as features to score parsing actions. Head assignment is typically performed with REDUCE-RIGHT and REDUCE-LEFT actions. Most proposals in transition-based constituency parsing since Sagae and Lavie (2005) have used a lexicalized transition system, and features involving heads to score

[*]Work partly done at Université Paris Diderot.

Transactions of the Association for Computational Linguistics, vol. 7, pp. 73–89, 2019. Action Editor: Stephen Clark.
Submission batch: 9/2018; Revision batch: 11/2018; Published 4/2019.

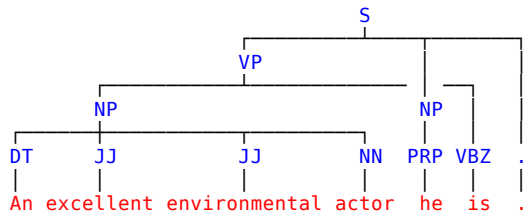


Figure 1: Tree from the Discontinuous Penn Treebank (Evang and Kallmeyer, 2011).

actions (Zhu et al., 2013; Zhang and Clark, 2011, 2009; Crabbé, 2014; Wang et al., 2015, among others), including proposals for discontinuous constituency parsing (Versley, 2014a; Maier, 2015; Coavoux and Crabbé, 2017a). A few recent proposals use an unlexicalized model (Watanabe and Sumita, 2015; Cross and Huang, 2016b; Dyer et al., 2016). Interestingly, these latter models all use recurrent neural networks (RNN) to compute constituent representations.

Our contributions are the following. We introduce an unlexicalized discontinuous parsing model, as well as its lexicalized counterpart. We evaluate them in identical experimental conditions. Our main finding is that, in our experiments, unlexicalized models consistently outperform lexicalized models. We assess the robustness of this result by performing the comparison of unlexicalized and lexicalized models with a second pair of transition systems. We further analyze the empirical properties of the systems in order to better understand the reasons for this performance difference. We find that the unlexicalized system oracle produces shorter, more incremental derivations. Finally, we provide a per-phenomenon error analysis of our best model and identify which types of discontinuous constituents are hard to predict.

2 Related Work

Several approaches to discontinuous constituency parsing have been proposed. Hall and Nivre (2008) reduces the problem to non-projective dependency parsing, via a reversible transformation, a strategy developed by Fernández-González and Martins (2015) and Corro et al. (2017). Chart parsers are based on probabilistic Linear Context-Free Rewriting Systems (LCFRS) (Evang and Kallmeyer, 2011; Kallmeyer and Maier, 2010), the Data-Oriented Parsing (DOP) framework (van Cranenburgh and Bod, 2013; van Cranenburgh et al., 2016), or pseudo-projective parsing (Versley, 2016).

Some transition-based discontinuous constituency parsers use the *swap* action, adapted from dependency parsing (Nivre, 2009) either with an easy-first strategy (Versley, 2014a,b) or with a shift-reduce strategy (Maier, 2015; Maier and Lichte, 2016; Stanojević and Garrido Alhama, 2017). Nevertheless, the swap strategy tends to produce long derivations (in number of actions) to construct discontinuous constituents; as a result, the choice of an oracle that minimizes the number of swap actions has a substantial positive effect in accuracy (Maier and Lichte, 2016; Stanojević and Garrido Alhama, 2017).

In contrast, Coavoux and Crabbé (2017a) extended a shift-reduce transition system to handle discontinuous constituents. Their system allows binary reductions to apply to the top element in the stack, and any other element in the stack (instead of the two top elements in standard shift-reduce parsing). The second constituent for a reduction is chosen dynamically, with an action called GAP that gives access to older elements in the stack and can be performed several times before a reduction. In practice, they made the following modifications over a standard shift-reduce system:

1. The stack, that stores subtrees being constructed, is split into two parts S and D;
2. reductions are applied to the respective tops of S and D;
3. the GAP action pops an element from S and adds it to D, making the next element of S available for a reduction.

Their parser outperforms swap-based systems. However, they only experiment with a linear classifier, and assume access to gold part-of-speech (POS) tags for most of their experiments.

All these proposals use a lexicalized model, as defined in the introduction: they assign heads to new constituents and use them as features to inform parsing decisions. Previous work on unlexicalized transition-based parsing models only focused on projective constituency trees (Dyer et al., 2016; Liu and Zhang, 2017). In particular, Cross and Huang (2016b) introduced a system that does not require explicit binarization. Their system decouples the construction of a tree and the labeling of its nodes by assigning types (*structure* or *label*) to each action, and alternating between a structural action for even steps and

Structural actions	Input	Output
SHIFT	$\langle S,\ D,\ i,\ C\rangle$	$\Rightarrow \langle S\|D,\ \{i+1\},\ i+1,\ C\rangle$
MERGE	$\langle S\|I_{s_0},\ D\|I_{d_0},\ i,\ C\rangle$	$\Rightarrow \langle S\|D,\ I_{s_0} \cup I_{d_0},\ i,\ C\rangle$
GAP	$\langle S\|I_{s_0},\ D,\ i,\ C\rangle$	$\Rightarrow \langle S,\ I_{s_0}\|D,\ i,\ C\rangle$
Labeling actions	**Input**	**Output**
LABEL-X	$\langle S,\ I_{d_0},\ i,\ C\rangle$	$\Rightarrow \langle S,\ I_{d_0},\ i,\ C \cup \{(X, I_{d_0})\}\rangle$
NO-LABEL	$\langle S,\ I_{d_0},\ i,\ C\rangle$	$\Rightarrow \langle S,\ I_{d_0},\ i,\ C\rangle$

Table 1: The ML-GAP transition system, an unlexicalized transition system for discontinuous constituency parsing.

labeling action for odd steps. This distinction arguably makes each decision simpler.

3 Transition Systems for Discontinuous Parsing

This section introduces an unlexicalized transition system able to construct discontinuous constituency trees (Section 3.1), its lexicalized counterpart (Section 3.2), and corresponding oracles (Section 3.3).

3.1 The Merge-Label-Gap Transition System

The Merge-Label-Gap transition system (henceforth ML-GAP) combines the distinction between structural and labeling actions from Cross and Huang (2016b) and the SHIFT-REDUCE-GAP (SR-GAP) strategy with a split stack from Coavoux and Crabbé (2017a).

Like the SR-GAP transition system, ML-GAP is based on three data structures: a stack S, a double-ended queue (dequeue) D, and a buffer B. We define a parsing configuration as a quadruple $\langle S, D, i, C\rangle$, where S and D are sequences of index sets, i is the index of the last shifted token, and C is a set of **instantiated discontinuous constituents**. We adopt a representation of instantiated constituents as pairs (X, I), where X is a nonterminal label, and I is a set of token indexes. For example, the discontinuous VP in Figure 1 is the pair (VP, $\{1, 2, 3, 4, 6\}$), because it spans tokens 1 through 4 and token 6.

The ML-GAP transition system is defined as a deduction system in Table 1. The available actions are the following:

- The SHIFT action pushes the singleton $\{i+1\}$ onto D.
- The MERGE action removes I_{s_0} and I_{d_0} from the top of S and D, computes their union $I = I_{s_0} \cup I_{d_0}$, transfers the content of D to S and pushes I onto D. It is meant to construct incrementally subsets of tokens that are constituents.
- The GAP action removes the top element from S and pushes it at the beginning of D. This action gives the system the ability to construct discontinuous trees, by making older elements in S accessible to a subsequent merge operation with I_{d_0}.
- LABEL-X creates a new constituent labeled X whose yield is the set I_{d_0} at the top of D.
- NO-LABEL has no effect.

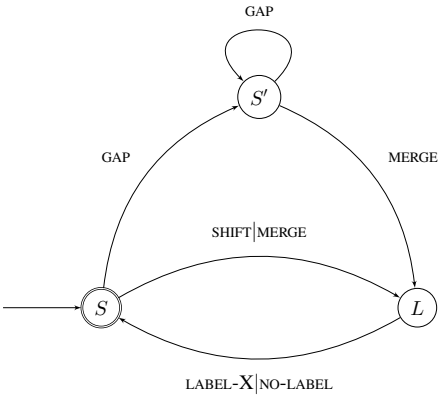


Figure 2: Action sequences allowed in ML-GAP. Any derivation must be recognized by the automaton.

Actions are subcategorized into **structural actions** (SHIFT, MERGE, GAP) and **labeling actions** (NO-LABEL, LABEL-X). This distinction is meant to make each single decision easier. The current state of the parser determines the type of action to be predicted next, as illustrated by the automaton in Figure 2. When it predicts a projective tree, the parser alternates structural and labeling actions (states S and L). However, it must be able to perform several GAP actions in a row to predict some discontinuous constituents. Since the semantics of the GAP action, a structural action,

Structural action	Stack – Dequeue – Buffer	Labeling action
	An excellent environment actor he is	
SH ⇒	$\{$An$\}_{d_0}$ excellent environment actor he is	⇒ NO-LABEL ⇒
SH ⇒	$\{$An$\}_{s_0}$ $\{$excellent$\}_{d_0}$ environment actor he is	⇒ NO-LABEL ⇒
MERGE ⇒	$\{$An excellent$\}_{d_0}$ environment actor he is	⇒ NO-LABEL ⇒
SH ⇒	$\{$An excellent$\}_{s_0}$ $\{$environment$\}_{d_0}$ actor he is	⇒ NO-LABEL ⇒
MERGE ⇒	$\{$An excellent environment$\}_{d_0}$ actor he is	⇒ NO-LABEL ⇒
SH ⇒	$\{$An excellent environment$\}_{s_0}$ $\{$actor$\}_{d_0}$ he is	⇒ NO-LABEL ⇒
MERGE ⇒	$\{$An excellent environment actor$\}_{d_0}$ he is	⇒ LABEL-NP ⇒
SH ⇒	$\{$An excellent environment actor$\}_{s_0}$ $\{$he$\}_{d_0}$ is	⇒ LABEL-NP ⇒
SH ⇒	$\{$An excellent environment actor$\}_{s_1}$ $\{$he$\}_{s_0}$ $\{$is$\}_{d_0}$	⇒ NO-LABEL ⇒
GAP ⇒	$\{$An excellent environment actor$\}_{s_0}$ $\{$he$\}_{d_1}$ $\{$is$\}_{d_0}$	⇒
MERGE ⇒	$\{$he$\}_{s_0}$ $\{$An excellent environment actor is$\}_{d_0}$	⇒ LABEL-VP ⇒
MERGE ⇒	$\{$he An excellent environment actor is$\}_{d_0}$	⇒ LABEL-S

Table 2: Example derivation for the tree in Figure 1 with the ML-GAP transition system.

is not to modify the top of D, but to make an index set in S available for a MERGE, it must not be followed by a labeling action. Each GAP action must be followed by either another GAP or a MERGE action (state S' in the automaton). We illustrate the transition system with a full derivation in Table 2.

3.2 Lexicalized Transition System

In order to assess the role of lexicalization in parsing, we introduce a second transition system, ML-GAP-LEX, which is designed (i) to be lexicalized and (ii) to differ minimally from ML-GAP.

We define an instantiated **lexicalized discontinuous constituent** as a triple (X, I, h) where X is a nonterminal label, I is the set of terminals that are in the yield of the constituent, and $h \in I$ is the lexical head of the constituent. In ML-GAP-LEX, the dequeue and the stack contains pairs (I, h), where I is a set of indices and $h \in I$ is a distinguished element of I.

The main difference of ML-GAP-LEX with ML-GAP is that there are two MERGE actions, MERGE-LEFT and MERGE-RIGHT, and that each of them assigns the head of the new set of indexes (and implicitly creates a new directed dependency arc):

- MERGE-LEFT: $\langle S|(I_{s_0}, h_{s_0}),\ D|(I_{d_0}, h_{d_0}),\ i,\ C\rangle \Rightarrow \langle S|D,\ (I_{s_0} \cup I_{d_0}, h_{s_0}),\ i,\ C\rangle$;
- MERGE-RIGHT: $\langle S|(I_{s_0}, h_{s_0}),\ D|(I_{d_0}, h_{d_0}),\ i,\ C\rangle \Rightarrow \langle S|D,\ (I_{s_0} \cup I_{d_0}, h_{d_0}),\ i,\ C\rangle$.

3.3 Oracles

In this work, we use deterministic static oracles. We briefly describe an oracle that builds constituents from their head outwards (head-driven oracle) and an oracle that performs merges as soon as possible (eager oracle). The latter can only be used by an unlexicalized system.

Head-driven Oracle The head-driven oracle can be straightforwardly derived from the oracle for SR-GAP presented by Coavoux and Crabbé (2017a). A derivation in ML-GAP-LEX can be computed from a derivation in SR-GAP by (i) replacing REDUCE-LEFT-X (resp. REDUCE-RIGHT-X) actions by a MERGE-LEFT (resp. MERGE-RIGHT), (ii) replacing REDUCE-UNARY-X actions by LABEL-X, and (iii) inserting LABEL-X and NO-LABEL actions as required. This oracle attaches the left dependents of a head first. In practice, other oracle strategies are possible as long as constituents are constructed from their head outward.

Eager Oracle For the ML-GAP system, we use an oracle that builds every n-ary constituent in a left-to-right fashion, as illustrated by the derivation in Table 2.[1] This implicitly corresponds to a left-branching binarization.

4 Neural Architecture

The statistical model we used is based on a Long Short-Term Memory network (bi-LSTM) transducer that builds context-aware representations for each token in the sentence (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016a). The token representations are then fed as input to (i) a tagging component for assigning POS tags and (ii) a parsing component for scoring parsing

[1] The systems exhibit spurious ambiguity for constructing n-ary ($n > 2$) constituents. We leave the exploration of nondeterministic oracles to future work.

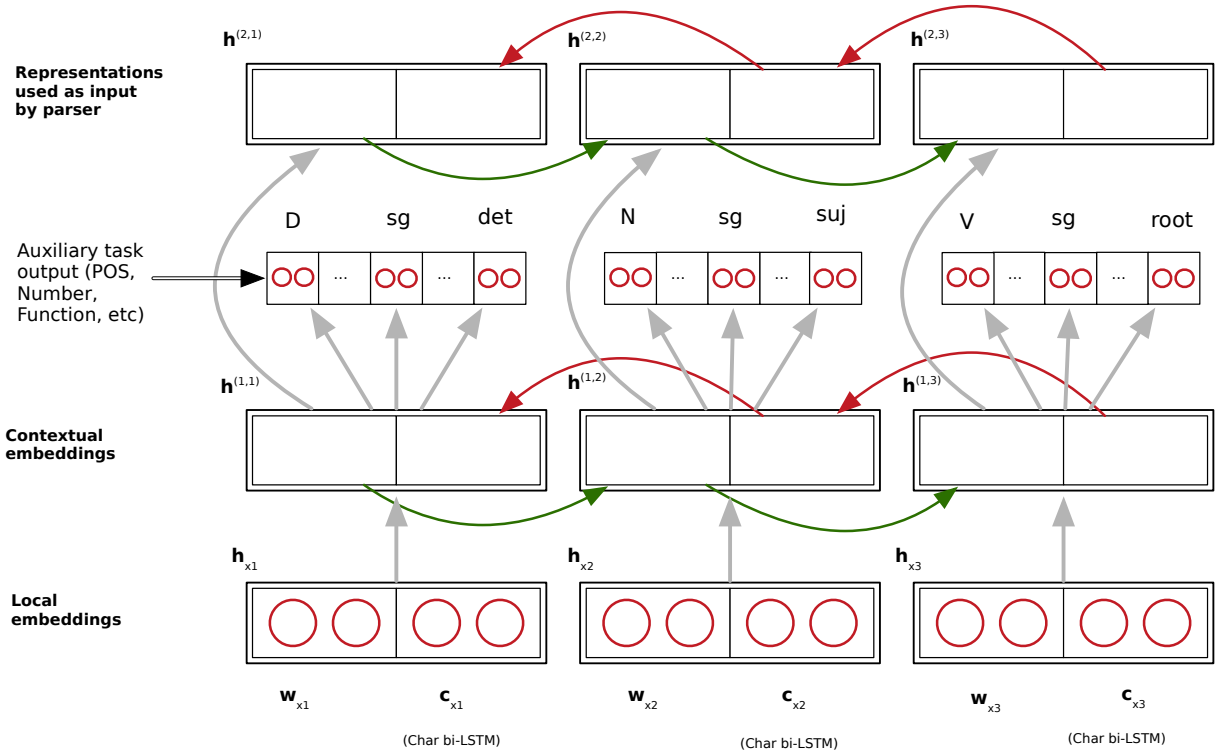


Figure 3: Bi-LSTM part of the neural architecture. Each word is represented by the concatenation of a standard word-embedding $\mathbf{w}$ and the output of a character bi-LSTM $\mathbf{c}$. The concatenation is fed to a two-layer bi-LSTM transducer that produces contextual word representations. The first layer serves as input to the tagger (Section 4.2), whereas the second layer is used by the parser to instantiate feature templates for each parsing step (Section 4.3).

actions.[2] The whole architecture is trained end-to-end. We illustrate the bi-LSTM and the tagging components in Figure 3.

In the following paragraphs, we describe the architecture that builds shared representations (Section 4.1), the tagging component (Section 4.2), the parsing component (Section 4.3), and the objective function (Section 4.4).

4.1 Building Context-aware Token Representations

We use a hierarchical bi-LSTM (Plank et al., 2016) to construct context-aware vector representations for each token. A lexical entry x is represented by the concatenation $\mathbf{h}_x = [\mathbf{w}_x; \mathbf{c}_x]$, where $\mathbf{w}_x$ is a standard word embedding and $\mathbf{c}_x = \text{bi-LSTM}(x)$ is the output of a character bi-LSTM encoder, i.e., the concatenation of its last forward and backward states.

We run a sentence-level bi-LSTM transducer over the sequence of local embeddings $(\mathbf{h}_{x_1}, \mathbf{h}_{x_2}, \ldots, \mathbf{h}_{x_n})$, to obtain vector representations that depend on the whole sentence:

$$(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(n)}) = \text{bi-LSTM}(\mathbf{h}_{x_1}, \mathbf{h}_{x_2}, \ldots, \mathbf{h}_{x_n}).$$

In practice, we use a two-layer bi-LSTM in order to supervise parsing and tagging at different layers, following results by Søgaard and Goldberg (2016). In what follows, we denote the i^{th} state of the j^{th} layer with $\mathbf{h}^{(j,i)}$.

4.2 Tagger

We use the context-aware representations as input to a softmax classifier to output a probability distribution over part-of-speech (POS) tags for each token:

$$P(t_i = \cdot | x_1^n; \boldsymbol{\theta}_t) = \text{Softmax}(\mathbf{W}^{(t)} \cdot \mathbf{h}^{(1,i)} + \mathbf{b}^{(t)}),$$

where $\mathbf{W}^{(t)}, \mathbf{b}^{(t)} \in \boldsymbol{\theta}_t$ are parameters.

In addition to predicting POS tags, we also predict other morphosyntactic attributes when they are available (i.e., for the Tiger corpus) such as the case, tense, mood, person, and gender, since the POS tagset does not necessarily contain this information. Finally, we predict the syntactic

[2] A more involved strategy would be to rely on Recurrent Neural Network Grammars (Dyer et al., 2016; Kuncoro et al., 2017). However, the adaptation of this model to discontinuous parsing is not straightforward and we leave it to future work.

functions of tokens, since this auxiliary task has been shown to be beneficial for constituency parsing (Coavoux and Crabbé, 2017b).

For each type of label l, we use a separate softmax classifier, with its own parameters $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$:

$$P(l_i = \cdot | x_1^n; \boldsymbol{\theta}_t) = \text{Softmax}(\mathbf{W}^{(l)} \cdot \mathbf{h}^{(1,i)} + \mathbf{b}^{(l)}).$$

For a given token, the number and types of morphosyntactic attributes depend on its POS tag. For example, a German noun has a gender and number but no tense nor mood. We use a default value ('undef') to make sure that every token has the same number of labels.

4.3 Parser

We decompose the probability of a sequence of actions $a_1^m = (a_1, a_2, \ldots, a_m)$ for a sentence x_1^n as the product of probability of individual actions:

$$P(a_1^m | x_1^n; \boldsymbol{\theta}_p) = \prod_{i=1}^{m} P(a_i | a_1^{i-1}, x_1^n; \boldsymbol{\theta}_p).$$

The probability of an action given a parsing configuration is computed with a feedforward network with two hidden layers:

$$\mathbf{o}^{(1)} = g(\mathbf{W}^{(1)} \cdot \mathbf{\Phi}_f(a_1^{i-1}, x_1^n) + \mathbf{b}^{(1)}),$$
$$\mathbf{o}^{(2)} = g(\mathbf{W}^{(2)} \cdot \mathbf{o}^{(1)} + \mathbf{b}^{(2)}),$$
$$P(a_i | a_1^{i-1}, x_1^n) = \text{Softmax}(\mathbf{W}^{(3)} \cdot \mathbf{o}^{(2)} + \mathbf{b}^{(3)}),$$

where

- g is an activation function (rectifier);
- $\mathbf{W}^{(i)}, \mathbf{b}^{(i)} \in \boldsymbol{\theta}_p$ are parameters;
- $\mathbf{\Phi}_f$ is a function, parameterized by a feature template list f, that outputs the concatenation of instantiated features, for the configuration obtained after performing the sequence of action $a_1^{(i-1)}$ to the input sentence x_1^n.

Feature templates describe a list of positions in a configuration. Features are instantiated by the context-aware representation of the token occupying the position. For example, token i will yield vector $\mathbf{h}^{(2,i)}$, the output of the sentence-level bi-LSTM transducer at position i. If a position contains no token, the feature is instantiated by a special trained embedding.

Feature Templates The two feature template sets we used are presented in Table 3. The BASE templates form a minimal set that extracts 7 indexes from a configuration, relying only on constituent boundaries. The +LEX feature set adds information about the heads of constituents at the top of S and D, and can only be used together with a lexicalized transition system.

Configuration: $\langle S\|(I_{s_1}, h_{s_1})\|(I_{s_0}, h_{s_0}), D\|(I_{d_1}, h_{d_1})\|(I_{d_0}, h_{d_0}), i, C\rangle$	
Template set	Token indexes
BASE	$\max(I_{s_1}), \min(I_{s_0}), \max(I_{s_0}),$ $\max(I_{d_1}), \min(I_{d_0}), \max(I_{d_0}), i$
+LEX	BASE+ $h_{d_0}, h_{d_1}, h_{s_0}, h_{s_1}$

Table 3: Feature template set descriptions.

4.4 Objective Function

The objective function for a single sentence x_1^n decomposes in a tagging objective and a parsing objective. The tagging objective is the negative log-likelihood of gold labels for each token:

$$\mathcal{L}_t(x_1^n; \boldsymbol{\theta}_t) = -\sum_{i=1}^{n} \sum_{j=1}^{k} \log P(t_{i,j} | x_1^n; \boldsymbol{\theta}_t),$$

where k is the number of types of labels to predict.

The parsing objective is the negative log-likelihood of the gold derivation, as computed by the oracle:

$$\mathcal{L}_p(x_1^n; \boldsymbol{\theta}_p) = -\sum_{i=1}^{m} \log P(a_i | a_1^{i-1}, x_1^n; \boldsymbol{\theta}_p).$$

We train the model by minimizing $\mathcal{L}_t + \mathcal{L}_p$ over the whole corpus. We do so by repeatedly sampling a sentence, performing one optimization step for $\mathcal{L}_t$ followed by one optimization step for $\mathcal{L}_p$. Some parameters are shared across the parser and the tagger, namely the word and character embeddings, the parameters for the character bi-LSTM, and those for the first layer of the sentence bi-LSTM.

5 Experiments

The experiments we performed aim at assessing the role of lexicalization in transition-based constituency parsing. We describe the data (Section 5.1) and the optimization protocol (Section 5.2). Then, we discuss empirical runtime efficiency (Section 5.3), before presenting the results of our experiments (Section 5.4).

5.1 Data

To evaluate our models, we used the Negra corpus (Skut et al., 1997), the Tiger corpus (Brants et al., 2002), and the discontinuous version of the Penn Treebank (Evang and Kallmeyer, 2011; Marcus et al., 1993).

For the Tiger corpus, we use the Statistical Parsing of Morphologically Rich Languages (SPMRL) split (Seddah et al., 2013). We obtained the dependency labels and the morphological information for each token from the dependency treebank versions of the SPMRL release. We converted the Negra corpus to labeled dependency trees with the DEPSY tool[3] in order to annotate each token with a dependency label. We do not predict morphological attributes for the Negra corpus (only POS tags) since only a small section is annotated with a full morphological analysis. We use the standard split (Dubey and Keller, 2003) for this corpus, and no limit on sentence length. For the Penn Treebank, we use the standard split (sections 2-21 for training, 22 for development and 23 for test). We retrieved the dependency labels from the dependency version of the Penn Treebank (PTB), obtained by the Stanford Parser (de Marneffe et al., 2006).

We used the relevant module of discodop[4] (van Cranenburgh et al., 2016) for evaluation. It provides an F1 measure on labeled constituents, as well as an F1 score computed only on discontinuous constituents (Disc. F1). Following standard practice, we used the evaluation parameters included in discodop release (`proper.prm`). These parameters ignore punctuation and root symbols.

5.2 Optimization and Hyperparameters

We optimize the loss with the Averaged Stochastic Gradient Descent algorithm (Polyak and Juditsky, 1992; Bottou, 2010) using the following dimensions for embeddings and hidden layers:

- Feedforward network: 2 layers of 128 units with rectifiers as activation function;
- The character bi-LSTM has 1 layer, with states of size 32 (in each direction);
- The sentence bi-LSTM has 2 layers, with states of size 128 (in each direction);
- Character embedding size: 32;
- Word-embedding size: 32.

We tune the learning rate ($\{0.01, 0.02\}$) and the number of iterations ($\{4, 8, 12, \ldots, 28, 30\}$) on the development sets of each corpus. All parameters, including embeddings, are randomly initialized. We use no pretrained word embeddings nor any other external data.[5]

Finally, following the method of Kiperwasser and Goldberg (2016) to handle unknown words, each time we sample a sentence from the training set, we stochastically replace each word by an UNKNOWN pseudoword with a probability $p_w = \left(\frac{\alpha}{\#\{w\}+\alpha}\right)$, where $\#\{w\}$ is the raw number of occurrences of w in the training set and α is a hyperparameter set to 0.8375, as suggested by Cross and Huang (2016b).

5.3 Runtime efficiency

For each experiment, we performed both training and parsing on a single CPU core. Training a single model on the Tiger corpus (i.e., the largest training corpus) took approximately a week. Parsing the 5,000 sentences of the development section of the Tiger corpus takes 53 seconds (1,454 tokens per second) for the ML-GAP model and 40 seconds (1,934 tokens per second) for the SR-GAP-UNLEX model, excluding model initialization and input-output times (Table 5).

Although indicative, these runtimes compare well with other neural discontinuous parsers, e.g., Corro et al. (2017), or to transition-based parsers using a linear classifier (Maier, 2015; Coavoux and Crabbé, 2017a).

5.4 Results

First, we compare the results of our proposed models on the development sets, focusing on the effect of lexicalization (Section 5.4.1). Then, we present morphological analysis results (Section 5.4.2). Finally, we compare our best model to other published results on the test sets (Section 5.4.3).

[3] `https://nats-www.informatik.uni-hamburg.de/pub/CDG/DownloadPage/cdg-2006-06-21.tar.gz` We modified DEPSY to keep the same tokenization as the original corpus.

[4] `https://github.com/andreasvc/disco-dop`

[5] We leave to future work the investigation of the effect of pretrained word embeddings and semi-supervised learning methods, such as tritraining, that have been shown to be effective in recent work on projective constituency parsing (Choe and Charniak, 2016; Kitaev and Klein, 2018).

Transition System	Features	Oracle	English		German (Tiger)		German (Negra)	
			F	Disc. F	F	Disc. F	F	Disc. F
ML-GAP	BASE	eager	**91.2**	72.0	**87.6**	**60.5**	83.7	**53.8**
ML-GAP	BASE	head-driven	91.1	**73.7**	87.2	59.7	83.7	**53.8**
ML-GAP-LEX	BASE	head-driven	91.1	68.2	86.5	56.3	82.4	47.0
ML-GAP-LEX	+LEX	head-driven	90.9	68.2	86.5	57.0	82.7	52.3
SR-GAP-UNLEX	BASE	eager	91.0	72.9	87.1	**60.5**	**84.1**	52.0
SR-GAP-LEX	BASE	head-driven	90.8	70.3	86.0	56.2	82.1	44.9
SR-GAP-LEX	+LEX	head-driven	90.8	71.3	86.5	55.5	82.8	50.6

Table 4: Discontinuous parsing results on the development sets.

Model	Tiger		DPTB	
	tok/s	sent/s	tok/s	sent/s
This work, ML-GAP, BASE	1,454	95	1,450	61
This work, SR-GAP-UNLEX, BASE	1,934	126	1,887	80
Maier (2015), beam=8		80		
Coavoux and Crabbé (2017a), beam=4	4,700	260		
Corro et al. (2017)				≈7.3

Table 5: Running times on development sets of the Tiger and the DPTB, reported in tokens per second (tok/s) and sentences per second (sent/s). Runtimes are only indicative; they are not comparable with those reported by other authors, since they use different hardware.

5.4.1 Effect of Lexicalization

Lexicalized vs. Unlexicalized Models We first compare the unlexicalized ML-GAP system with the ML-GAP-LEX system (Table 4). The former consistently obtains higher results. The F-score difference is small on English (0.1 to 0.3) but substantial on the German treebanks (more than 1.0 absolute point) and in general on discontinuous constituents (Disc. F).

In order to assess the robustness of the advantage of unlexicalized models, we also compare our implementation of SR-GAP (Coavoux and Crabbé, 2017a)[6] with an unlexicalized variant (SR-GAP-UNLEX) that uses a single type of reduction (REDUCE) instead of the traditional REDUCE-RIGHT and REDUCE-LEFT actions. This second comparison exhibits the same pattern in favor of unlexicalized models.

These results suggest that lexicalization is not necessary to achieve very strong discontinuous parsing results. A possible interpretation is that the bi-LSTM transducer may implicitly learn latent lexicalization, as suggested by Kuncoro et al. (2017), which is consistent with recent analyses of other types of syntactic information captured by LSTMs in parsing models (Gaddy et al., 2018) or language models (Linzen et al., 2016).

Effect of Lexical Features For lexicalized models, information about the head of constituents (+LEX) have a mixed effect and brings an improvement in only half the cases. It is even slightly detrimental on English (ML-GAP-LEX).

Controlling for the Oracle Choice The advantage of unlexicalized systems could be due to the properties of its eager oracle, in particular its higher incrementality (see Section 6 for an analysis). In order to isolate the effect of the oracle, we trained ML-GAP with the head-driven oracle, i.e., the oracle used by the ML-GAP-LEX system. We observe a small drop in F-measure on English (-0.1) and on the Tiger corpus (-0.4) but no effect on the Negra corpus. However, the resulting parser still outperforms ML-GAP-LEX, with the exception of English. These results suggest

[6] This is not the same model as Coavoux and Crabbé (2017a) since our experiments use the statistical model presented in Section 4, with joint morphological analysis, whereas they use a structured perceptron and require a POS-tagged input.

Corpus	Attribute	Acc.	F1	Cov.
PTB	POS	97.2	-	100
Negra	POS	98.1	-	100
Tiger (ours)	POS	98.4	-	100
	Complete match	92.9	-	100
	Case	96.9	96.9	48.2
	Degree	99.7	98.0	7.5
	Gender	96.9	96.8	47.7
	Mood	99.9	99.1	7.8
	Number	98.4	98.7	57.8
	Person	99.9	99.5	9.5
	Tense	99.9	99.3	7.8
Tiger (Björkelund et al., 2013; Mueller et al., 2013)				
	POS	98.1		
	Complete match	91.8		

Table 6: Morphological analysis results on development sets.

that the oracle choice definitely plays a role in the advantage of ML-GAP over ML-GAP-LEX, but is not sufficient to explain the performance difference.

Discussion Overall, our experiments provide empirical arguments in favor of unlexicalized discontinuous parsing systems. Unlexicalized systems are arguably simpler than their lexicalized counterparts—since they have no directional (left or right) actions—and obtain better results. We further hypothesize that derivations produced by the eager oracle, which cannot be used by lexicalized systems, are easier to learn. We provide a quantitative and comparative analysis of derivations from both transition systems in Section 6.

5.4.2 Tagging and Morphological Analysis

We report results for morphological analysis with the selected models (ML-GAP with BASE features for the Penn Treebank and Tiger, SR-GAP-UNLEX with BASE features for Negra) in Table 6. For each morphological attribute, we report an accuracy score computed over every token. However, most morphological attributes are only relevant for specific part-of-speech tags. For instance, TENSE is only a feature of verbs. The accuracy metric is somewhat misleading, since the fact that the tagger predicts correctly that a token does not have an attribute is considered a correct answer. Therefore, if only 5% of tokens bore a specific morphological attribute, a 95% accuracy is a most frequent baseline score. For this reason, we also report a coverage metric (Cov.) that indicates the proportion of tokens in the corpus that possess an attribute, and an F1 measure.

The tagger achieves close to state-of-the-art results on all three corpora. On the Tiger corpus, it slightly outperforms previous results published by Björkelund et al. (2013) who used the MarMoT tagger (Mueller et al., 2013). Morphological attributes are also very well predicted, with F1 scores above 98%, except for case and gender.

5.4.3 External Comparisons

The two best performing models on the development sets are the ML-GAP (DPTB, Tiger) and the SR-GAP-UNLEX (Negra) models with BASE features. We report their results on the test sets in Table 7. They are compared with other published results: transition-based parsers using a SWAP action (Maier, 2015; Stanojević and Garrido Alhama, 2017) or a GAP action (Coavoux and Crabbé, 2017a), the pseudo-projective parser of Versley (2016), parsers based on non-projective dependency parsing (Fernández-González and Martins, 2015; Corro et al., 2017), and finally chart parsers based on probabilistic LCFRS (Evang and Kallmeyer, 2011; Gebhardt, 2018) or data-oriented parsing (van Cranenburgh et al., 2016). Note that some of these publications report results in a gold POS-tag scenario, a much easier experimental setup that is not comparable to ours (bottom part of the table). In Table 7, we also indicate models that use a neural scoring system with a '*'.

Our models obtain state-of-the-art results and outperform every other system, including the LSTM-based parser of Stanojević and Garrido Alhama (2017) that uses a SWAP action to predict discontinuities. This observation confirms in another setting the results of Coavoux and Crabbé (2017a), namely that GAP transition systems have more desirable properties than SWAP transition systems.

6 Model Analysis

In this section, we investigate empirical properties of the transition systems evaluated in the previous section. A key difference between lexicalized and unlexicalized systems is that the latter are arguably simpler: they do not have to assign heads to new constituents. As a result, they need fewer

Model	English (DPTB)		German (Tiger)		German (Negra)	
	F	Disc. F	F	Disc. F	F	Disc. F
Predicted POS tags						
Ours*, ML-GAP (SR-GAP-UNLEX for Negra), BASE features	**91.0**	**71.3**	**82.7**	**55.9**	**83.2**	**54.6**
Stanojević and Garrido Alhama (2017)*, SWAP, stack/tree-LSTM			77.0			
Coavoux and Crabbé (2017a), SR-GAP, perceptron			79.3			
Versley (2016), pseudo-projective, chart-based			79.5			
Corro et al. (2017)*, bi-LSTM, Maximum Spanning Arborescence	89.2					
van Cranenburgh et al. (2016), DOP, ≤ 40	87.0				74.8	
Fernández-González and Martins (2015), dependency-based			77.3			
Gebhardt (2018), LCFRS with latent annotations			75.1			
Gold POS tags						
Stanojević and Garrido Alhama (2017)*, SWAP, stack/tree-LSTM			81.6		82.9	
Coavoux and Crabbé (2017a), SR-GAP, perceptron			81.6	49.2	82.2	50.0
Maier (2015), SWAP, perceptron			74.7	18.8	77.0	19.8
Corro et al. (2017)* bi-LSTM, Maximum Spanning Arborescence	90.1		81.6			
Evang and Kallmeyer (2011), PLCFRS, < 25	79†					

Table 7: Discontinuous parsing results on the test sets.
*Neural scoring system. †Does not discount root symbols and punctuation.

types of distinct transitions, and they have simpler decisions to make. Furthermore, they do not run the risk of error propagation from wrong head assignments.

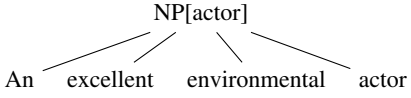
We argue that an important consequence of the simplicity of unlexicalized systems is that their derivations are easier to learn. In particular, ML-GAP derivations have a better incrementality than those of ML-GAP-LEX (Section 6.1) and are more economical in terms of number of GAP actions needed to derive discontinuous trees (Section 6.2).

Average length of stack (S+D)		
Corpus	ML-GAP-LEX	ML-GAP
English (DPTB)	5.62	4.86
German (Negra)	3.69	2.88
German (Tiger)	3.56	2.98

Table 8: Incrementality measured by the average size of the stack during derivations. The average is calculated across all configurations (not across all sentences).

6.1 Incrementality

We adopt the definition of incrementality of Nivre (2004): an incremental algorithm minimizes the number of connected components in the stack during parsing. An unlexicalized system can construct a new constituent by incorporating each new component immediately, whereas a lexicalized system waits until it has shifted the head of a constituent before starting to build the constituent. For example, to construct the following head-final NP,



a lexicalized system must shift every token before starting reductions in order to be able to predict the dependency arcs between the head *actor* and its three dependents.[7] In contrast, an unlexicalized

[7] SH(IFT), SH, SH, SH, SH, M(ERGE)-R(IGHT), M-R, M-R, M-R, LABEL-NP. (NO-LABEL actions are omitted.)

		ML-GAP-LEX	ML-GAP
English (DPTB)	Max number of consecutive GAPS	9	8
	Average number of consecutive GAPS	1.78	1.34
	Total number of GAPS	33,341	18,421
German (Tiger)	Max number of consecutive GAPS	10	5
	Average number of consecutive GAPS	1.40	1.12
	Total number of GAPS	40,905	25,852
German (Negra)	Max number of consecutive GAPS	11	5
	Average number of consecutive GAPS	1.47	1.11
	Total number of GAPS	20,149	11,181

Table 9: GAP action statistics in training sets.

system can construct partial structures as soon as there are two elements with the same parent node in the stack.[8]

We report the average number of connected components in the stack during a derivation computed by an oracle for each transition system in Table 8. The unlexicalized transition system ML-GAP has a better incrementality. On average, it maintains a smaller stack. This is an advantage since parsing decisions rely on information extracted from the stack and smaller localized stacks are easier to represent.

6.2 Number of GAP Actions

The GAP actions are supposedly the most difficult to predict, because they involve long distance information. They also increase the length of a derivation and make the parser more prone to error propagation. We expect that a transition system that is able to predict a discontinuous tree more efficiently, in terms of number of GAP actions, to be a better choice.

We report in Table 9 the number of GAP actions necessary to derive the discontinuous trees for several corpora and for several transition systems (using oracles). We also report the average and maximum number of consecutive GAP actions in each case. For English and German, the unlexicalized transition system ML-GAP needs much fewer GAP actions to derive discontinuous trees (approximately 45% fewer). The average number of consecutive GAP actions is also smaller (as well as the maximum for German corpora). On average, the elements in the stack (S) that need to combine with the top of D are closer to the top of S with the ML-GAP transition system than with lexicalized systems. This observation is not surprising; since ML-GAP can start constructing constituents before having access to their lexical head, it can construct larger structures before having to GAP them.

7 Error Analysis

In this section, we provide an error analysis focused on the predictions of the ML-GAP model on the discontinuous constituents of the discontinuous PTB. It is aimed at understanding which types of long-distance dependencies are easy or hard to predict and providing insights for future work.

7.1 Methodology

We manually compared the gold and predicted trees from the development set that contained at least one discontinuous constituent.

Out of 278 sentences in the development set containing a discontinuity (excluding those in which the discontinuity is only due to punctuation attachment), 165 were exact matches for discontinuous constituents and 113 contained at least one error. Following Evang (2011), we classified errors according to the phenomena producing a discontinuity. We used the following typology,[9] illustrated by examples where the main discontinuous constituent is highlighted in bold:

- *Wh*-extractions: ***What** should I **do**?*
- Fronted quotations: ***‘‘Absolutely’’**, he **said**.*
- Extraposed dependent: *In April 1987, **evidence** surfaced **that commissions were paid.***

[8] SH, SH, M(ERGE), SH, M, SH, M, SH, M, LABEL-NP.

[9] These categories cover all cases in the development set.

Phenomenon	G	PfM	PaM	FN	FP
Wh-extractions	122	87	19	16	8
	100%	71.3	15.6	13.1	NA
Fronted quotations	81	77	3	1	0
	100%	95.1	3.7	1.2	NA
Extrapositions	44	10	1	33	3
	100%	22.7	2.3	75	NA
Circumpositioned quotations	22	11	10	1	3
	100%	50	45.4	4.5	NA
It-extrapositions	16	6	2	8	2
	100%	37.5	12.5	50	NA
Subject-verb inversion	5	4	0	1	1
	100%	80	0	20	NA

Table 10: Evaluation statistics per phenomenon. G: gold occurrences; PfM: perfect match; PaM: partial match; FN: false negatives; FP: false positives.

- Circumpositioned quotations: ***In general****, they say,* ***avoid takeover stocks.***
- It-extrapositions: *''**It**'s better* ***to wait**.''*
- Subject-verb inversion: ***Said*** *the spokeswoman:* ***''The whole structure has changed.''***

For each phenomenon occurrence, we manually classified the output of the parser in one of the following categories: (i) perfect match, (ii) partial match, and (iii) false negative. Partial matches are cases where the parser identified the phenomenon involved but made a mistake regarding the labelling of a discontinuous constituent (e.g., S instead of SBAR) or its scope. The latter case includes, e.g., occurrences where the parser found an extraction, but failed to find the correct extraction site. Finally, we also report false positives for each phenomenon.

7.2 Results

First of all, the parser tends to be conservative when predicting discontinuities: there are in general few false positives. The 72.0 discontinuous F1 (Table 4) indeed decomposes in a precision of 78.4 and a recall of 66.6. This does not seem to be a property of our parser, as other authors also report systematically higher precisions than recalls (Maier, 2015; Stanojević and Garrido Alhama, 2017). Instead, the scarcity of discontinuities in the data might be a determining factor: only 20% of sentences in the Discontinuous Penn Treebank contain at least one discontinuity and 30% of sentences in the Negra and Tiger corpus.

Analysis results are presented in Table 10. For *wh*-extractions, there are two main causes of errors. The first one is an ambiguity on the extraction site. For example, in the relative clause *which many clients didn't know about*, instead of predicting a discontinuous PP, where *which* is the complement of *about*, the parser attached *which* as a complement of *know*. Another source of error (both for false positives and false negatives) is the ambiguity of *that*-clauses, that can be either completive clauses[10] or relative clauses.[11]

Phenomena related to quotations are rather well identified probably due to the fact that they are frequent in newspaper data and exhibit regular patterns (quotation marks, speech verbs). However, a difficulty in identifying circumpositioned quotations arises when there are no quotation marks, to determine what the precise scope of the quotation is.

Finally, the hardest types of discontinuity for the parser are extrapositions. Contrary to previously discussed phenomena, there is usually no lexical trigger (*wh*-word, speech verb) that makes these discontinuities easy to spot. Most cases involve modifier attachment ambiguities, which are known to be hard to solve (Kummerfeld

[10](NP the consensus . . . (SBAR that the Namibian guerrillas were above all else the victims of suppression by neighboring South Africa.))

[11](NP the place (SBAR **that** world opinion has been celebrating **over**))

et al., 2012) and often require some world knowledge.

8 Conclusion

We have introduced an unlexicalized transition-based discontinuous constituency parsing model.[12] We have compared it, in identical experimental settings, with its lexicalized counterpart in order to provide insights on the effect of lexicalization as a parser design choice.

We found that lexicalization is not necessary to achieve very high parsing results in discontinuous constituency parsing, a result consistent with previous studies on lexicalization in projective constituency parsing (Klein and Manning, 2003; Cross and Huang, 2016b). A study of empirical properties of our transition systems suggested explanations for the performance difference, by showing that the unlexicalized system produces shorter derivations and has a better incrementality. Finally, we presented a qualitative analysis of our parser's errors on discontinuous constituents.

Acknowledgments

We thank Kilian Evang and Laura Kallmeyer for providing us with the Discontinuous Penn Treebank. We thank Caio Corro, Sacha Beniamine, *TACL* reviewers, and action editor Stephen Clark for feedback that helped improve the paper. Our implementation makes use of the Eigen C++ library (Guennebaud and Jacob, 2010), `treetools`,[13] and `discodop`.[14] MC and SC gratefully acknowledge the support of Huawei Technologies.

References

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In *Proceedings of EMNLP 2004*, pages 182–189. Association for Computational Linguistics.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (Re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA. Association for Computational Linguistics.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France. Springer.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. Tiger treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories, September 20-21 (TLT02)*. Sozopol, Bulgaria.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pages 598–603. AAAI Press.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017a. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270, Valencia, Spain. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017b. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 331–336, Valencia, Spain. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of*

[12] The source code of the parser is released with pretrained models at `https://github.com/mcoavoux/mtg_TACL`.

[13] `https://github.com/wmaier/treetools`

[14] `https://github.com/andreasvc/disco-dop`

the Association for Computational Linguistics, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1645–1655, Copenhagen, Denmark. Association for Computational Linguistics.

Benoit Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 541–552, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of IWPT*, pages 7–16.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for german using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Kilian Evang. 2011. Parsing discontinuous constituents in English. Ph.D. thesis, Masters thesis, University of Tübingen.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*,

pages 3049–3063, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Gaël Guennebaud and Benoît Jacob. 2010. Eigen v3. http://eigen.tuxfamily.org.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Johan Hall and Joakim Nivre. 2008. *Parsing Discontinuous Phrase Structure with Grammatical Functions*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 537–545, Beijing, China. Coling 2010 Organizing Committee.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics*, 4:521–535.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics, Volume 19, Number 2, June 1993, Special Issue on Using Large Corpora: II*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating

typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA).

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan. Association for Computational Linguistics.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Shashi Narayan and Shay B. Cohen. 2016. Optimizing spectral learning for parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1546–1556, Berlin, Germany. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Joakim Nivre. 2009. 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*, pages 351–359. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. 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*, pages 433–440. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Boris T. Polyak and Anatoli B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing mor phologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington, DC, USA. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Miloš Stanojević and Raquel Garrido Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1667–1677, Copenhagen, Denmark. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Yannick Versley. 2014a. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland. Dublin City University.

Yannick Versley. 2014b. Incorporating semi-supervised features into discontinuous easy-first constituent parsing. *CoRR*, abs/1409.3813v1.

Yannick Versley. 2016. Discontinuity (re)^{2}visited: A minimalist approach to pseudoprojective constituent parsing. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 58–69, San Diego, California. Association for Computational Linguistics.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1138–1147, Beijing, China. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171, Paris, France. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. Association for Computational Linguistics.