Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing

Kilian Gebhardt
Department of Computer Science
Technische Universität Dresden
D-01062 Dresden, Germany
kilian.gebhardt@tu-dresden.de

Abstract

We formulate a generalization of Petrov et al. (2006)'s split/merge algorithm for interpreted regular tree grammars (Koller and Kuhlmann, 2011), which capture a large class of grammar formalisms. We evaluate its effectiveness empirically on the task of discontinuous constituent parsing with two mildly context-sensitive grammar formalisms: *linear context-free rewriting systems* (Vijay-Shanker et al., 1987) as well as *hybrid grammars* (Nederhof and Vogler, 2014).

1 Introduction

Probabilistic grammars are a standard model for language processing tasks. Their fundamental principle is a rewriting process in which nonterminals are repeatedly unfolded in accordance to rewrite rules until a structure consisting solely of terminals is obtained. Context-free independence assumptions imply that the applicability as well as the probability of a rewrite step depend only on the nonterminal that is unfolded but not on the context or history in which the nonterminal occurs.

The independence assumptions allow for tractable algorithms when processing data with these grammars. Then again, the expressiveness of such a grammar is constrained by the number of its nonterminals. This is why it was found beneficial to refine naturally emerging sets of nonterminals (such as syntactic categories). Strategies of refinement of *context-free grammars* (CFGs) involve for instance *Markovization*, i.e., the encoding of limited context into the nonterminals (Collins, 1999; Klein and Manning, 2003), and automatic state-splitting by means of latent annotations (Matsuzaki et al., 2005; Petrov et al., 2006). An important observation is that these refinements are *latent*, i.e., they are not observed in the predictions that the CFG is supposed to provide. Automatic state-splitting has been successfully applied also to tree-substitution grammars (Shindo et al., 2012) and tree-adjoining grammars (Ferraro et al., 2012).

Koller and Kuhlmann (2011) proposed *interpreted regular tree grammars* (IRTGs) as a uniform way to describe a large class of grammar formalisms that share the context-free rewriting mechanism. IRTGs decouple the derivational process, in which derivation trees are generated by a (probabilistic) *regular tree grammar* (RTG), from the interpretation of derivation trees in one or multiple algebras. IRTGs enable the development of generic algorithms for binarization (Büchse et al., 2013), parsing and decoding (Groschwitz et al., 2016; Teichmann et al., 2017), and estimation techniques (Teichmann et al., 2016).

The central hypothesis of this article is that Petrov et al. (2006)'s *split/merge algorithm* (a) can be transferred from CFGs to a large class of grammar formalisms and (b) that its application improves the probabilistic behavior of a given grammar for parsing and decoding tasks.

The first contribution of our paper is a generic version of Petrov et al. (2006)'s *split/merge algorithm* in the IRTG framework. We choose IRTGs because the separation of the derivational process in an RTG from the interpretation in algebras implies that (i) only one algorithm needs to be formulated to capture a large class of grammar formalisms and that (ii) the nonterminals cannot be observed in the generated structures. Because of (ii) nonterminals of IRTGs may be viewed as already being latent, i.e., with IRTGs latent annotations *come for free*. We also transfer objectives for efficient parsing and decoding as proposed

Proceedings of the 27th International Conference on Computational Linguistics, pages 3049–3063
Santa Fe, New Mexico, USA, August 20-26, 2018.

by Matsuzaki et al. (2005) and Petrov and Klein (2007) into the IRTG framework. An implementation of the generic algorithms is provided.

Then, for a case study of (b) we apply the generalized split/merge algorithm and the different parsing objectives to both *linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al., 1987; Kallmeyer and Maier, 2013) and *hybrid grammars* (Nederhof and Vogler, 2014) on the task of discontinuous constituent parsing. This choice is relevant because the application of the split/merge algorithm to either grammar formalism has been supposed by Evang and Kallmeyer (2011) and Gebhardt et al. (2017), respectively, but to our knowledge not yet been performed. We find that the split/merge algorithm improves the parsing accuracy of the grammars by up to 14.5 points in labeled F1. However, the grammars do not reach the accuracy of recent transition-based discriminative parsers.

2 Preliminaries

Let A, B, and C be sets and $f\colon A \to B$ and $g\colon B \to C$ be functions. The *powerset of* A is denoted by $\mathcal{P}(A)$. We extend f in the natural way to $f\colon \mathcal{P}(A) \to \mathcal{P}(B)$ and $f^{-1}\colon \mathcal{P}(B) \to \mathcal{P}(A)$. We call f *surjective*, if for each $b \in B$, there exists $a \in A$ with $f(a) = b$. We let $g \circ f\colon A \to C$ denote the composition of f and g. We identify a singleton set $\{a\}$ with its element a.

2.1 Interpreted regular tree grammars

An interpreted regular tree grammar generates an object a of some domain A in two phases: firstly a derivation tree ξ is generated and secondly ξ is interpreted in an algebra $\mathcal{A}$ to a. Figure 1 shows two derivation trees ξ_1 and ξ_2 over *operator symbols* f_0, f_1 and f_2. Each operator symbol admits a fixed number of arguments called its *rank*, e.g., the ranks of f_0, f_1, and f_2 are 0, 1, and 2, respectively. A finite, non-empty set of operator symbols constitutes a *signature* Σ. The set of *derivation trees over* Σ, denoted by T_Σ, is the smallest set U where for any $f \in \Sigma$ of rank k and $\xi_1, \ldots, \xi_k \in U$ we have $f(\xi_1, \ldots, \xi_k) \in U$. Let $\xi = f(\xi_1, \ldots, \xi_k)$ be in T_Σ. The set of *positions* of ξ is $\mathrm{pos}(\xi) = \{\varepsilon\} \cup \{i\pi \mid 1 \le i \le k, \pi \in \mathrm{pos}(\xi_i)\}$. The *operator symbol at the position* π *in* ξ, denoted by $\xi(\pi)$, is f if $\pi = \varepsilon$ and $\xi_i(\pi')$ if $\pi = i\pi'$.

Next, we describe the interpretation of a derivation tree from T_Σ by a Σ-algebra. A Σ-*algebra* $\mathcal{A}$ consists of a set A (*domain*) and, for each operator symbol f in Σ of rank k, an *operation* $f^{\mathcal{A}}\colon A^k \to A$. Each derivation tree $f(\xi_1, \ldots, \xi_n)$ in T_Σ can be *evaluated in* $\mathcal{A}$ to an element $[\![f(\xi_1, \ldots, \xi_k)]\!]_{\mathcal{A}} = f^{\mathcal{A}}([\![\xi_1]\!]_{\mathcal{A}}, \ldots, [\![\xi_k]\!]_{\mathcal{A}})$ in A. Let $\Sigma_{\mathrm{ex}} = \{f_0, f_1, f_2\}$. Figure 1 shows the operations of the Σ_{ex}-algebras $\mathcal{A}_s$ and $\mathcal{A}_t$ with the set of strings and parse trees as domains, respectively. Also, it shows the evaluation of $\xi_1, \xi_2 \in T_{\Sigma_{\mathrm{ex}}}$ in $\mathcal{A}_s$ to the string $s = bbb$ and the evaluation of ξ_1 and ξ_2 in $\mathcal{A}_t$ to the parse trees t_1 and t_2, respectively.

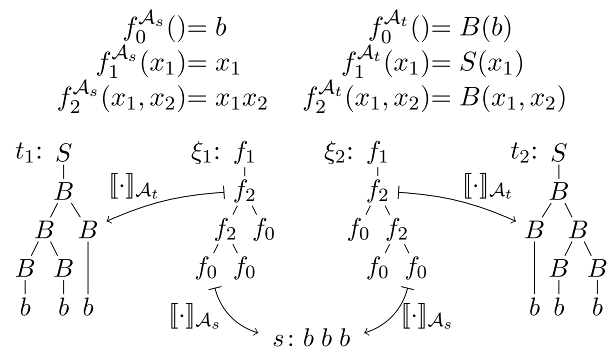

Figure 1: Operations of the Σ_{ex}-algebras $\mathcal{A}_s$ and $\mathcal{A}_t$, derivation trees ξ_1 and ξ_2, and evaluation of ξ_1 and ξ_2 in $\mathcal{A}_s$ and $\mathcal{A}_t$.

Regular tree grammars are useful to describe formal languages of derivation trees. A *regular tree grammar* (RTG) (Brainerd, 1969) over Σ is a tuple $G = (N_G, S_G, R_G)$. The finite set N_G is disjoint from Σ and contains *nonterminals*. S_G in N_G is the *start nonterminal*. The set R_G is a finite subset of the set of prototypical rules $R[N_G, \Sigma]$. $R[N_G, \Sigma]$ contains each *rule* of the form $B \to f(B_1, \ldots, B_k)$ where $f \in \Sigma$ is of rank k and $B, B_1, \ldots, B_k$ are in N_G. Figure 2a shows the rules of $G_{\mathrm{ex}} = (\{S, B\}, S, R_G)$ over Σ_{ex}.

An RTG G generates a derivation tree $\xi \in T_\Sigma$ if it has a valid run on it: A *run of* G *on* ξ is a mapping $r\colon \mathrm{pos}(\xi) \to N_G$. The *rule at position* π *of* r is $\mathrm{rule}_r^\pi = r(\pi) \to \xi(\pi)(r(\pi 1), \ldots, r(\pi k))$ where k is the rank of $\xi(\pi)$. We call r *valid* if $r(\varepsilon) = S_G$ and r is *consistent with* R_G, i.e., for every $\pi \in \mathrm{pos}(\xi)$, we require that $\mathrm{rule}_r^\pi \in R_G$. We denote the set of all valid runs of G on ξ by $\mathrm{runs}_G^v(\xi)$. The *language of* G is $L(G) = \{\xi \in T_\Sigma \mid \mathrm{runs}_G^v(\xi) \neq \emptyset\}$. Moreover, we let $\mathrm{runs}_G^v = \{(\xi, r) \mid \xi \in T_\Sigma, r \in \mathrm{runs}_G^v(\xi)\}$.

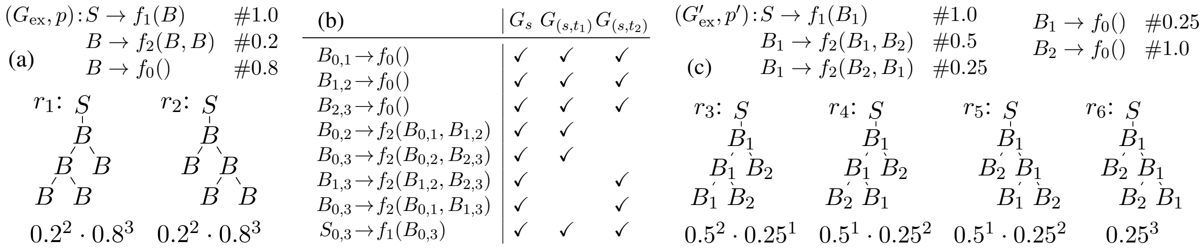

Figure 2: (a) and (c): Prob. RTGs (G_{ex}, p) and (G'_{ex}, p') with runs and probabilities (below). (b): Charts.

Figure 2a shows the only valid runs r_1 and r_2 of G_{ex} on the derivation trees ξ_1 and ξ_2, respectively, in tree-like form.

Interpreted RTGs. An *interpreted RTG* (IRTG) (Koller and Kuhlmann, 2011) is a tuple $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$ where G is an RTG over Σ and each $\mathcal{A}_j$ is a Σ-algebra[1]. Given an object $a = (a_1, \ldots, a_l)$ from the *domain* $A = A_1 \times \ldots \times A_l$, the parsing problem for IRTG is to compute $\text{parses}_{\mathbb{G}}(a) = \{\xi \in L(G) \mid \forall j : [\![\xi]\!]_{\mathcal{A}_j} = a_j\}$. This set is equal to the intersection of all sets $D_{a_j} = \{\xi \in T_\Sigma \mid [\![\xi]\!]_{\mathcal{A}_j} = a_j\}$ and $L(G)$. If, for each $a_j \in A_j$, there is an RTG whose language is D_{a_j}, then $\mathcal{A}_j$ is called *regularly decomposable*. In the following we consider only such algebras. Since the languages of RTGs are closed under intersection, we can construct RTGs G_{a_j} with $L(G_{a_j}) = D_{a_j} \cap L(G)$ and the RTG G_a with $L(G_a) = \bigcap_j L(G_{a_j}) = \text{parses}_{\mathbb{G}}(a)$. We call G_{a_j} and G_a the *chart of* a_j and the *chart of* a, respectively.

As a running example we extend the RTG G_{ex} to an IRTG $\mathbb{G}_{\text{ex}} = (G_{\text{ex}}, \mathcal{A}_s, \mathcal{A}_t)$. Then $\mathbb{G}_{\text{ex}}$ is equivalent to a CFG where $\mathcal{A}_s$ specifies the generation of strings and $\mathcal{A}_t$ describes the corresponding parse trees. Figure 2b depicts the rules of the charts G_s, $G_{(s,t_1)}$, and $G_{(s,t_2)}$ where the nonterminals were annotated with spans as usual and the start nonterminal is $S_{0,3}$.

Grammar morphisms. In order to relate two RTGs G' and G (e.g., G' could be a chart G_a), we consider mappings $\varphi\colon N_{G'} \to N_G$ between the respective sets of nonterminals. We lift φ to $\varphi\colon R[N_{G'}, \Sigma] \to R[N_G, \Sigma]$ by setting $\varphi(B' \to f(B'_1, \ldots B'_n)) = \varphi(B') \to f(\varphi(B'_1), \ldots, \varphi(B'_n))$. If $\varphi^{-1}(S_G) = \{S_{G'}\}$ and $\varphi(R_{G'}) \subseteq R_G$, then we call φ a *grammar morphism from G' to G*. Intuitively, φ has to establish a correspondence between the grammars' start nonterminals and rules.

For instance, we can choose any set M and any mapping $\varphi\colon N_G \to M$ that satisfies $\varphi^{-1}(\varphi(S_G)) = \{S_G\}$ and construct a new RTG $\varphi(G)$ from G: we set $\varphi(G) = (\varphi(N_G), \varphi(S_G), \varphi(R_G))$. Obviously, φ is a grammar morphism from G to $\varphi(G)$.

For each chart G_a we assume a grammar morphism φ_a from G_a to G that gives rise to a one-to-one correspondence between the valid runs of G_a and the valid runs of G for derivations trees of a. Formally, for each $\xi \in \text{parses}_{\mathbb{G}}(a)$ we require that $\bar{\varphi_a}\colon \text{runs}^v_{G_a}(\xi) \to \text{runs}^v_G(\xi)$ where $\bar{\varphi_a}(r) = \varphi_a \circ r$ is a bijection. For instance, φ_s, $\varphi_{(s,t_1)}$, and $\varphi_{(s,t_2)}$ strip the spans from each nonterminal, e.g., $\varphi_s(B_{i,j}) = B$.

2.2 Extending IRTGs with probabilities

RTGs and IRTGs can be equipped with probabilities in the *standard* way: A *weight assignment for an RTG* G is a mapping $p\colon R_G \to [0, 1]$. We define the weight of a run r on a derivation tree ξ as $W_{(G,p)}(\xi, r) = \prod_{\pi \in \text{pos}(\xi)} p(\text{rule}^\pi_r)$ and the weight of a derivation tree ξ as $W_{(G,p)}(\xi) = \sum_{r \in \text{runs}^v_G(\xi)} W_{(G,p)}(\xi, r)$. We call p *proper* if, for every $A \in N_G$, we have $1 = \sum_{\varrho = (A \to f(B_1, \ldots, B_k)) \text{ in } R_G} p(\varrho)$. If $1 = \sum_{\xi \in T_\Sigma} W_{(G,p)}(\xi)$, then we call p *consistent*. In this case we may write $P(\xi, r \mid G, p)$ and $P(\xi \mid G, p)$ for $W_{(G,p)}(\xi, r)$ and $W_{(G,p)}(\xi)$, respectively, and call (G, p) *probabilistic RTG*. Let G' be an RTG and φ be a grammar morphism from G' to G. Then $p \circ \varphi\colon R_{G'} \to [0, 1]$ is a weight assignment for G'.

Given an IRTG $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$ and a proper and consistent weight assignment p for G, we define a probability distribution on A by $P(a \mid \mathbb{G}, p) = \sum_{\xi \in \text{parses}_{\mathbb{G}}(a)} W_{(G,p)}(\xi)$. Using the chart G_a the same quantity can be obtained: $P(a \mid \mathbb{G}, p) = \sum_{\xi \in L(G_a)} W_{(G_a, p \circ \varphi_a)}(\xi)$ (see Appendix A.1).

[1] For ease of notation, we omit the homomorphism that is originally associated to each algebra. We note that the methods we develop in the following are agnostic of the algebras and, thus, applicable to the original definition as well.

Considering $\mathbb{G}_{\text{ex}}$ and the probability assignment p for G in Figure 2a, we get that $P((s,t_1) \mid \mathbb{G}_{\text{ex}}, p) = P((s,t_2) \mid \mathbb{G}_{\text{ex}}, p)$ because r_1 and r_2 have the same probability. In contrast, G'_{ex} in Figure 2c has two runs (r_3/r_4 and r_5/r_6, respectively) on each of ξ_1 and ξ_2. Taking the sum of their probabilities yields $P((s,t_1) \mid \mathbb{G}'_{\text{ex}}, p') > P((s,t_2) \mid \mathbb{G}'_{\text{ex}}, p')$ for the IRTG $\mathbb{G}'_{\text{ex}} = (G'_{\text{ex}}, \mathcal{A}_s, \mathcal{A}_t)$.

Inside and outside weights are a tool for efficient calculation of probabilities during training and parsing. The *inside weight* $\beta(B)$ and the *outside weight* $\alpha(B)$ of a nonterminal $B \in N_G$ are defined as

$$\beta(B) = \sum_{\varrho = B \to f(B_1,\dots,B_k) \in R_G} p(\varrho) \cdot \beta(B_1) \cdot \ldots \cdot \beta(B_k) \quad \text{and} \quad \alpha(B) = \delta_B^S + \sum_{\substack{\varrho = C \to f(B_1,\dots,B_k) \text{ in } R_G \\ 1 \le i \le k\colon B_i = B}} \alpha(C) \cdot p(\varrho) \cdot \prod_{j \ne i} \beta(B_j) \ ,$$

respectively, where $\delta_B^S = 1$ if $B = S$ and 0 otherwise.

The sum of the probabilities of all runs of an RTG G equals $\beta(S_G)$. Hence, an efficient way to obtain $P(a \mid \mathbb{G}, p)$ is computing $\beta(S_{G_a})$. The expected frequency with which a nonterminal B occurs in a run of G is obtained by $\alpha(B) \cdot \beta(B)/\beta(S_G)$ (Nederhof and Satta, 2004). For any rule ϱ of the form $B \to f(B_1, \dots, B_k)$, we let $\alpha(\varrho) = \alpha(B)$ and $\beta(\varrho) = \beta(B_1) \cdot \ldots \cdot \beta(B_k)$.

2.3 A parsing or decoding problem

Suppose we want to employ $\mathbb{G}_{\text{ex}}$ for syntactic parsing (for clarity we write G instead of G_{ex}). This can be framed as a decoding problem: for a given string s, the chart G_s is computed, from which we obtain the set $T = [\![L(G_s)]\!]_{\mathcal{A}_t}$ of parse trees of s (i.e., $T = \{t_1, t_2\}$ in our example). If the IRTG is equipped with a probability assignment, then, alternatively, one can ask for the *best* parse tree $\hat{t}$, which may be formalized as

$$\hat{t} = \arg\max_{t \in T} \quad \textstyle\sum_{\xi \in L(G_s)\colon [\![\xi]\!]_{\mathcal{A}_t} = t} \quad P(\xi \mid G, p) \ . \tag{1}$$

Computing this expression turns out to be infeasible in general because maximizing the sum over the potentially infinite set of derivation trees and the sum over the exponential number of runs over each derivation tree resists dynamic programming. A tractable option is to compute the Viterbi run $\hat{r}$ of the grammar (Knuth, 1977; Nederhof, 2003), defined as

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi, r) \in \text{runs}^v_{G_s}} P(\xi, r \mid G, p) \tag{2}$$

or, with a small overhead, the n-best runs (Huang and Chiang, 2005).

For a given derivation tree $\xi \in T_\Sigma$, we can efficiently compute or approximate $P(\xi \mid G, p)$ by restricting G to ξ, which yields an RTG G_ξ, and computing $\beta(S_{G_\xi})$. Likewise, for a given parse tree t, we can compute $P(t \mid s, G, p) = \beta(G_{(s,t)})/\beta(G_s)$.

2.4 Expectation/Maximization training

The expectation/maximization (EM) algorithm (Dempster et al., 1977) in the inside-outside variant (Lari and Young, 1990) carries over to IRTGs. We recall it for sake of completeness. During the *expectation* step, we compute a corpus $c\colon R_G \to \mathbb{R}_{\ge 0}$ over rules given a corpus $c_A\colon A \to \mathbb{R}_{\ge 0}$ over the domain such that

$$c(\varrho) = \textstyle\sum_{a \in A} c_A(a) \cdot \sum_{\varrho' \in \varphi_a^{-1}(\varrho)} \dfrac{\alpha(\varrho') \cdot p(\varrho') \cdot \beta(\varrho')}{\beta(S_{G_a})} \ .$$

In the *maximization* step the probability assignment p is updated to match the empirical distribution of $c(\varrho)$. Both steps are iterated until p changes only slightly or until the likelihood of a validation set drops.

3 Refinement of IRTGs with the split/merge algorithm

Probabilistic context-free grammars with latent annotation (PCFG-LAs) were introduced by Matsuzaki et al. (2005) as a way to tackle the too strong independence assumptions of probabilistic CFG. Instead of assigning each CFG rule just a single probability, different probabilities are assigned depending on the substate which is annotated to each nonterminal. These substates are *latent*, i.e., when calculating the probability of a parse tree, any assignment of substates to nonterminals is considered. The work of Petrov et al. (2006) extends the PCFG-LA approach by a procedure that adaptively refines the latent state set. Petrov et al. (2006) start from a binarized PCFG-LA where each nonterminal has just one substate. In multiple cycles each such substate is split in two and the resulting grammar is trained with the EM algorithm. Then 50% of the splits are undone depending on how much likelihood gets lost and the grammar is trained with EM again. Finally, the rule weights are smoothed and trained once more.

The idea of latent annotated grammar states can be easily formalized in the IRTG framework: The probabilistic behavior of the IRTG depends only on the nonterminals and the applied rules of its RTG G. However, in the derivation trees in $L(G)$ and their interpretations, the nonterminals of G are no longer visible. To calculate the probability of a derivation tree ξ, we have to consider any valid run r on ξ and its weight. Thus, the latent states of a PCFG-LA naturally correspond to the nonterminals of the RTG G.

We reformulate the split/merge algorithm by Petrov et al. (2006) for an arbitrary IRTG $\mathbb{G} = (G, \Sigma, \mathcal{A}_1, \dots, \mathcal{A}_l)$. In the process a (fine) probabilistic RTG (G', p') is constructed from the (coarse) probabilistic RTG (G, p), while Σ and the algebras are not changed. The refinement from N_G to $N_{G'}$ allows defining a more subtle probabilistic behavior in (G', p'). Thus, $L(G) = L(G')$, however $W_{(G,p)}$ and $W_{(G',p')}$ may differ. In analogy to the original algorithm, each nonterminal is split in two by an inverse grammar morphism μ_{sp}^{-1} yielding an intermediate grammar G^f. The probabilities of the rules of G^f are tuned by EM training. Afterwards, splits that turn out less useful are reversed by a grammar morphism μ_Δ yielding the grammar G'. The probabilities of this grammar are trained again and smoothed.

The grammar morphisms between the grammars G, G^f, and G', also imply the existence of grammar morphisms between the charts of these grammars. Consequently, charts can be easily transformed (sparing recomputation from scratch) which increases the efficiency of EM training and parsing (cf. Section 3.4).

3.1 Splitting and merging

We define the splitting and merging of nonterminals in a generic way by (inverse) grammar morphisms. Let (G, p) be a probabilistic RTG. For splitting the nonterminals of (G, p) we consider a surjective mapping $\mu\colon N' \to N_G$ where N' is a finite set (fine nonterminals) and $\mu^{-1}(S_G)$ is a singleton set. Splitting corresponds to applying the *inverse* of μ to G, i.e., $\mu^{-1}(G) = (N', \mu^{-1}(S_G), R')$ where $R' = \mu^{-1}(R) = \{\varrho' \in R[N', \Sigma] \mid \mu(\varrho) \in R\}$. The corresponding weight assignment for $\mu^{-1}(G)$ is $p \circ \mu$, which may be normalized to obtain a genuine probability assignment.

For merging the nonterminals of (G, p) we consider a surjective mapping $\mu\colon N_G \to M$ where M is a finite set (merged nonterminals) and $\mu^{-1}(\mu(S_G)) = \{S_G\}$. Merging is as simple as applying μ to G, i.e., computing $\mu(G) = (M, \mu(S_G), \mu(R))$. In order to construct a probability assignment $\mu(p)$ for $\mu(G)$, we let for every $\hat{\varrho} \in \mu(R)$:

$$(\mu(p))(\hat{\varrho}) = \textstyle\sum_{\varrho \in \mu^{-1}(\hat{\varrho})} p(\varrho) \cdot \dfrac{\alpha(\varrho) \cdot \beta(\varrho)}{\sum_{\varrho' \in \mu^{-1}(\hat{\varrho})} \alpha(\varrho') \cdot \beta(\varrho')} \ ,$$

where α and β are computed with respect to (G, p).

Two instances. We give two instances for grammar morphisms in reminiscence of Petrov et al. (2006). For splitting we consider a grammar morphism μ_{sp} that splits every nonterminal B of G but the start nonterminal into B_1 and B_2. Formally, $\mu_{\text{sp}}\colon N' \to N_G$ where $N' = \{B_q \mid B \in N_G, B \neq S, q \in \{1, 2\}\} \cup \{S\}$ and

$$\mu_{\text{sp}}(B') = \begin{cases} B & \text{if } B' = B_1 \text{ or } B' = B_2 \\ B' & \text{if } B' = S \end{cases}$$

In order to partially reverse the split of μ_{sp}, we define the grammar morphism μ_Δ that merges each pair B_1 and B_2 back to B based on a utility measure $\Delta(B_1, B_2)$. Formally, $\mu_\Delta\colon N' \to M$ where

$$\mu_\Delta(B') = \begin{cases} B & \text{if } B' = B_q \text{ for } B \in N_G,\ q \in \{1, 2\},\ \text{and } \Delta(B_1, B_2) > \eta \\ B' & \text{otherwise.} \end{cases}$$

We chose M to be the largest subset of $N_G \cup N'$ such that μ_Δ is surjective.

The function Δ is meant to approximate the quotient of likelihood after and before merging. This approximation, introduced by Petrov et al. (2006), uses inside and outside weights of charts, which were precomupted during EM training, to simulate merging of single instances of B_1 and B_2 in one chart. A generalization of Δ can be defined for arbitrary IRTG as long as each nonterminal of some chart occurs at most once in a run (App. A.2). The parameter η is set dynamically such that 50% of the splits are merged.

3.2 The complete split/merge cycle

Splitting, merging, the EM training of Section 2.4, and a tie-breaking and smoothing step, yet to be defined, is composed to a complete *split/merge cycle* in Algorithm 3.1. Multiple split/merge cycles are iteratively applied to a *base grammar* G_0 until the resulting grammar G_i reaches the desired level of refinement. For every refined grammar G_i, there exists a grammar morphism μ_i from G_i to G_0. During *tie-breaking* each rule's probability obtains a small random perturbation. During *smoothing* the probability of a rule ϱ is set proportional to $\gamma \cdot p(\varrho) + (1-\gamma) \cdot u$ where u is the sum of probabilities of rules $\varrho' \in \mu_i^{-1}(\mu_i(\varrho))$. Intuitively, the probability of different refinements of the same rule from the base grammar is slightly aligned. Following Petrov et al. (2006), we set γ to 0.9 for rules without nonterminals on the right-hand side. Otherwise, we set γ to 0.99.

Algorithm 3.1 Split/merge cycle

Input: IRTG $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$, prob. ass. p
corpus $c_A \colon A \to \mathbb{R}_{\geq 0}$
Output: IRTG $\mathbb{G}'$, prob. assignment p'

```
(G^f, p^f) ← (μ_sp^{-1}(G), p ∘ μ_sp)
p^f ← BreakTies(p^f)
p^f ← EM-Training(G^f, p^f, A_1, ..., A_l, c_A)
(G', p') ← (μ_Δ(G^f), μ_Δ(p^f))
p' ← EM-Training(G', p', A_1, ..., A_l, c_A)
p' ← smooth(G', p')
p' ← EM-Training(G', p', A_1, ..., A_l, c_A)
output (G', A_1, ..., A_l), p'
```

3.3 Efficient refinement of a chart

Let G^c be a coarse grammar, $a \in A$ be in the domain, and the chart G^c_a be already computed. We assume that G^c was refined to $G^f = \mu^{-1}(G^c)$ and that we want to compute the chart G^f_a. Due to the definition of splitting and merging, we do not need to compute G^f_a from scratch. Instead we construct (a grammar that is isomorphic to) G^f_a via the grammar morphisms φ'_a and μ' such that the diagram on the right commutes. Let $N' = \{(B, q) \mid B \in N_{G^c_a}, q \in \mu^{-1}(\varphi_a(B))\}$, and for every $(B, q) \in N'$, set $\varphi'_a(B, q) = q$ and $\mu'(B, q) = B$. We define $G^f_a = \mu'^{-1}(G^c_a)$.

$$
\begin{array}{ccc}
G^f_a & \xrightarrow{\varphi'_a} & G^f \\
{\scriptstyle \mu'}\downarrow & & \downarrow{\scriptstyle \mu} \\
G^c_a & \xrightarrow[\varphi_a]{} & G^c
\end{array}
$$

3.4 Parsing objectives and weight projections

In order to apply a refined IRTG to parsing, we return to the question on how to substitute Equation 1. We consider alternative parsing objectives inspired by Matsuzaki et al. (2005). In the following we refer to the base RTG and the one resulting from several iterations of Algorithm 3.1 by G^c and G^f, respectively. Also, we consider charts of a string s and assume grammar morphisms φ_s, φ'_s, μ, and μ' as in Section 3.3.

The $\hat{t}$ of Equation 1 is sometimes called *most probable parse*. A subproblem that is in general still infeasible is finding the parse tree corresponding to the *most probable derivation tree*, i.e.,

$$\hat{t} = [\![\arg\max_{\xi \,\in L(G^f_s)} P(\xi \mid G^f, p^f)]\!]_{\mathcal{A}_t} \ .$$

For usual PCFG(-LA) this objective coincides with the most probable parse because derivation trees and parse trees are in a one-to-one correspondence.

The parse corresponding to the *viterbi derivation tree* is $\hat{t} = [\![\hat{\xi}]\!]$ where $(\hat{\xi}, \hat{r})$ is computed according to Equation 2 for $(G^f_s, p^f \circ \varphi'_s)$. This objective is tractable but reported to yield suboptimal parses for PCFG-LA in terms of the usual bracket scoring metric (Matsuzaki et al., 2005; Petrov et al., 2006).

A combination of coarse-to-fine parsing with n-best parsing yields the *base-n-rerank* objective: n-best runs of the base grammar are computed and the corresponding derivation trees are reranked according to the refined grammar. Formally, we compute $\hat{t} = [\![\hat{\xi}]\!]_{\mathcal{A}_t}$ with

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi, r) \,\in\, n\text{-best-runs}(G^c_s, p^c \,\circ\, \varphi_s)} P(\xi \mid G^f, p^f) \ .$$

SQ → yf_1(MD, VP, NP)	$\mathrm{yf}_1^{\mathcal{A}_s}(x_1,(x_2,x_3),x_4) = (x_2 x_1 x_3 x_4)$	$\mathrm{yf}_1^{\mathcal{A}_t}(x_1,x_2,x_3) = \mathrm{SQ}(x_1,x_2,x_3)$
VP → yf_2(WHNP, VB)	$\mathrm{yf}_2^{\mathcal{A}_s}((x_1),(x_2)) = (x_1, x_2)$	$\mathrm{yf}_2^{\mathcal{A}_t}(x_1,x_2) = \mathrm{VP}(x_1,x_2)$
WHNP → $\mathrm{yf}_3()$	$\mathrm{yf}_3^{\mathcal{A}_s}() = (\text{What})$	$\mathrm{yf}_3^{\mathcal{A}_t}() = \mathrm{WHNP}(\text{What})$
MD → $\mathrm{yf}_4()$	$\mathrm{yf}_4^{\mathcal{A}_s}() = (\text{should})$	$\mathrm{yf}_4^{\mathcal{A}_t}() = \mathrm{MD}(\text{should})$
NP → $\mathrm{yf}_5()$	$\mathrm{yf}_5^{\mathcal{A}_s}() = (\text{I})$	$\mathrm{yf}_5^{\mathcal{A}_t}() = \mathrm{NP}(\text{I})$
VB → $\mathrm{yf}_6()$	$\mathrm{yf}_6^{\mathcal{A}_s}() = (\text{do})$	$\mathrm{yf}_6^{\mathcal{A}_t}() = \mathrm{VB}(\text{do})$

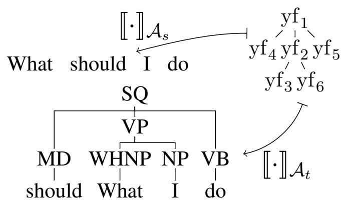

Figure 4: Rules of an LCFRS and the evaluation of a derivation tree in $\mathcal{A}_s$ and $\mathcal{A}_t$.

Parsing by weight projection. Matsuzaki et al. (2005) propose an alternative parsing objective, where a new weight assignment q for G_s^c is computed based on $(G_s^f, p^f \circ \varphi_s')$ such that the KL-divergence of $P(\xi \mid G_s^c, q)$ to $P(\xi \mid s, G^f, p^f)$ is minimized. Precisely, $q = \mu'(p^f \circ \varphi_s')$. Subsequently the parse tree $\hat{t}$ corresponding to the Viterbi derivation tree is computed using q, i.e., $\hat{t} = [\![\hat{\xi}]\!]_{\mathcal{A}_t}$ with

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi,r)\,\in \mathrm{runs}^v_{G_s^c}} P(\xi, r \mid G_s^c, q) \ .$$

An empirically superior way to define q called *max-rule-product* is due to Petrov and Klein (2007). They intent to optimize for "the tree with greatest chance of having all rules correct, under the (incorrect) assumption that the rules correctness are independent." To this end, each rule is assigned the sum of the expected frequencies of its refinements, i.e., $q(\varrho)$ is set to $\sum_{\varrho' \in \mu'^{-1}(\varrho)} \alpha(\varrho') \cdot (p^f \circ \varphi_s')(\varrho') \cdot \beta(\varrho')/\beta(S_{G_s^f})$. Hence, the value $q(\varrho)$ does not have to be in the interval $[0, 1]$ and max-rule-product (as well as *max-rule-sum* – where the weight of a run is the sum of rule weights rather than the product) is in theory a potentially non-monotonic and, thus, ill-defined objective (cf. Appendix A.3).

4 Grammars for discontinuous parsing

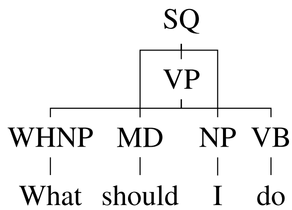

Figure 3: A discontinuous phrase structure.

String-rewriting *linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al., 1987) generalize CFGs and can account for discontinuous constituent trees such as the one in Figure 3. Each nonterminal B derives instead of a single string an m_B-tuple of strings. The integer $m_B \geq 1$ is called *fanout* of B. Each rule of an LCFRS consists of a left-hand side nonterminal B, any number of right-hand side nonterminals $B_1, \ldots, B_k$, and a *yield function* yf. The latter specifies how the components of the string tuples generated by $B_1, \ldots, B_k$ are concatenated with new terminal symbols to form the string tuple of B. The rewriting is *linear*, that is, each input to yf is used at most once. Parsing of binary LCFRS is in $\mathcal{O}(n^{3m})$ where n is the sentence length and m is the maximum fanout of nonterminals of the LCFRS (Seki et al., 1991).

We use the following straightforward representation of LCFRSs as IRTGs: Σ is the set of yield function symbols, A_s is the set of tuples over strings, and the string algebra $\mathcal{A}_s$ interprets each yield function symbol by the respective yield function. To obtain the corresponding parse tree, the derivation tree is interpreted in an algebra $\mathcal{A}_t$. Figure 4 depicts an LCFRS with start nonterminal SQ and the interpretation of a derivation tree.

In a *LCFRS/sDCP-hybrid grammar* (Nederhof and Vogler, 2014; Gebhardt et al., 2017) (short: hybrid grammar), an LCFRS is coupled with a tree generating device called *simple definite clause program* (sDCP) (Deransart and Małuszynski, 1985). An object in the domain of a hybrid grammar is a *hybrid tree*, i.e., a string and a tree together with a linking between sentence positions and tree positions. Hybrid trees (and hybrid grammars) are suitable to model (formal languages of) discontinuous constituent structures. Figure 5 depicts a hybrid grammar in IRTG notation and the evaluation of the derivation tree ξ_3 in the algebra $\mathcal{A}_h$ to the hybrid tree h. The algebra $\mathcal{A}_s$ is a copy of the string component of $\mathcal{A}_h$ and evaluates ξ_3 to "What should I do". We observe that the structure of ξ_3 deviates notably from the structure of the tree component of h. In fact, the hybrid grammar generates a discontinuous tree although its string component is equivalent to a CFG.

Nederhof and Vogler (2014) present an algorithm that induces an LCFRS/sDCP-hybrid grammar $\mathbb{G}$ given a corpus of phrase structure trees. The algorithm is parametrized by an integer $k \geq 1$ that limits the maximum fanout of the LCFRS component of $\mathbb{G}$ to k. A second parameter of the algorithm is one of two nonterminal labeling schemes called *child labeling* and *strict labeling* of which the former is coarser. Drewes et al. (2016) present an algorithm that computes the chart G_h in time polynomial in the size of some hybrid tree h. Computing G_s, given a string s, inherits the parsing complexity of LCFRSs.

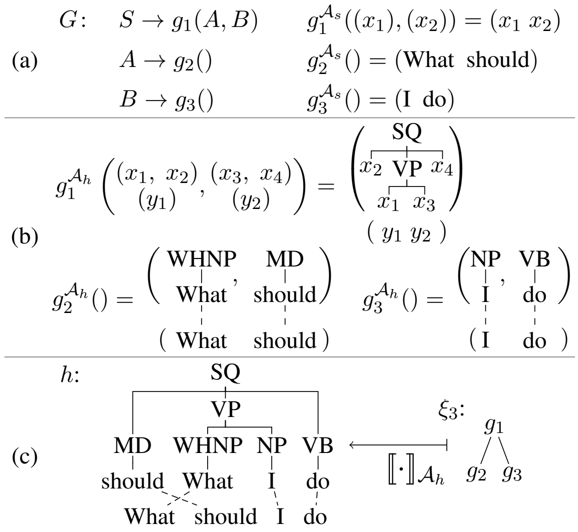

Figure 5: (a,b): An LCFRS/sDCP-hybrid grammar $(G, \mathcal{A}_s, \mathcal{A}_h)$ in IRTG notation. (b): $\mathcal{A}_h$ interprets each function symbol g_i by a function on synchronized tuples over trees (top) and tuples over strings (bottom). (c): Interpretation of ξ_3 in $\mathcal{A}_h$ to a hybrid tree h.

5 Experimental evaluation

We implemented[2] the generic split/merge algorithm and the parsing objectives in C++ with bindings to python3. We evaluate it with LCFRSs and hybrid grammars for discontinuous phrase structure parsing. We use the TiGer corpus (Brants et al., 2004) and employ the split of the SPMRL shared task (Seddah et al., 2014) (TiGerSPMRL) and the one of Hall and Nivre (2008) (TiGerHN08). TiGer contains ca. 50k annotated sentences of German news text, exhibits discontinuity, and is predominantly used for evaluation in recent literature on discontinuous parsing. Evaluation with other languages is subject of further research. Part-of-speech (POS) tags for the TiGerHN08 test set are predicted using the MATE tagger (Björkelund et al., 2010), which we trained on the training and development section of TigerHN08.

The base LCFRS is induced from the treebank following Maier and Søgaard (2008). For each rule we remember the grammatical function symbol of the left-hand side nonterminal to its parent in the algebra $\mathcal{A}_t$. We binarize the LCFRS either right-to-left ($\text{LCFRS}_{\text{r2}\ell}$) or head-outward ($\text{LCFRS}_{\text{ho}}$) (cf. Kallmeyer and Maier, 2013) and apply Markovization ($v = 1$, $h = 1$). The base LCFRS/sDCP-hybrid grammar is induced according to a modified version of the algorithm by Nederhof and Vogler (2014) where we include only the POS tag in the sDCP component of a lexical rule but no additional unary categories. Also, we include syntactic function labels into the rules' sDCP component. We restrict the fanout to 2 and use strict and child labeling (abbreviated $\text{hybrid}_{\text{strict}}$ and $\text{hybrid}_{\text{child}}$, respectively). The numbers of nonterminals and all/lexical rules of either grammar, and the coverage of trees from the development set are given in the table on the right for TiGerHN08.

	nont.	rules / lex.	cover.
LCFRS_{ho}	767	50,153 / 28,080	78.3%
$\text{LCFRS}_{\text{r2}\ell}$	817	49,297 / 28,080	76.5%
$\text{hybrid}_{\text{child}}$	288	39,123 / 28,080	82.9%
$\text{hybrid}_{\text{strict}}$	32,281	108,957 / 28,080	50.0%

The LCFRSs and hybrid grammars are refined by 5 and 4 split/merge cycles, respectively. We stop EM training if the likelihood of the covered trees in the development set decreases. Then we apply the grammars in some of the ways outlined in Section 3.4 for parsing unseen sentences. Each sentence is a tokenized string s over pairs of words and (gold) POS tags. For both LCFRSs and hybrid grammars we have to carry out an LCFRS parsing step on s, for which we employ the implementation[3] of van Cranenburgh et al. (2016, see Sec. 6.4). Precisely, a pruned version of the chart G_s is computed via a coarse-to-fine pipeline that utilizes a PCFG approximation of the probabilistic LCFRS. Once a best parse has been selected, we compare it to the gold tree by computing F1 and exact match (EM) for labeled brackets, F1 for discontinuous labeled brackets, and F1 including function tags (F1-fun) using *disco-dop*[3].

[2] The implementation is freely available at https://github.com/kilian-gebhardt/panda-parser.

[3] *disco-dop* can be obtained from https://github.com/andreasvc/disco-dop. For evaluation on TiGerHN08 we use the proper.prm parameters. For TiGerSPMRL we also report F1 with spmrl.prm.

Objective	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun
	LCFRS head-outward			LCFRS right-to-left			$\text{hybrid}_{\text{child}}$			$\text{hybrid}_{\text{strict}}$		
base-Viterbi	68.29 (22.55)	28.21	41.73	70.36 (23.00)	30.06	43.15	63.19 (15.04)	23.89	39.22	69.86 (29.34)	29.63	43.24
fine-Viterbi	76.59 (29.01)	35.87	63.45	77.32 (30.94)	36.83	65.48	76.56 (29.66)	39.27	65.03	73.34 (34.47)	33.95	61.06
variational	79.09 (33.17)	41.30	67.23	79.04 (34.32)	40.85	68.74	77.48 (30.53)	40.79	66.96	73.94 (33.75)	35.28	62.34
max-rule-prod.	**79.44** (33.74)	**41.73**	67.51	79.21 (**34.54**)	40.95	**68.83**	77.69 (30.45)	41.18	67.05	73.99 (34.02)	35.48	62.37
base-500-rerank	74.09 (29.31)	36.77	55.65	74.52 (28.82)	36.49	56.15	69.30 (25.61)	31.82	52.16	72.53 (32.98)	33.68	55.41

Table 1: Results on the TiGerHN08 development set with gold POS tags for sentences up to length 40.

The results on the TiGerHN08 development set are depicted in Table 1. The refined grammars notably improve over the respective base grammars by up to 14.50 points in F1. If we fix a base grammar but alter the parsing objectives, then we observe the following variations in F1 and EM: Reranking the 500-best derivations of the base grammar gives worse results than the Viterbi objective on the refined grammar. The weight projection approaches again improve over the Viterbi objective by up to 2.85 and 1.13 points in F1 for LCFRSs and hybrid grammars, respectively, where max-rule-product is consistently superior to variational. We do not observe an instance where max-rule-product is ill-defined in our experiments. Figure 6 shows that the F1 decreases for longer sentences by the example of LCFRS_{ho}/max-rule-product. $\text{LCFRS}_{\text{r2}\ell}$ is superior to LCFRS_{ho} with the Viterbi objective but the opposite holds for the projection-based objectives. $\text{Hybrid}_{\text{strict}}$, which suffers from 165 parse failures, produces worse results than $\text{hybrid}_{\text{child}}$ (12 parse failures) except for the reranking objective. For the discontinuous F1 and the F1-fun we find that $\text{LCFRS}_{\text{r2}\ell}$ performs better than LCFRS_{ho} in all cases but one. Also $\text{hybrid}_{\text{strict}}$ outperforms $\text{hybrid}_{\text{child}}$ with respect to discontinuous F1 but not for F1-fun. Overall, LCFRSs outperform hybrid grammars in terms of F1. The best F1 of 79.44 and 77.69 are obtained by LCFRS_{ho} and $\text{hybrid}_{\text{child}}$, respectively, with max-rule-product.

We parse the test set with LCFRS_{ho} and $\text{hybrid}_{\text{child}}$ using the max-rule-product objective. We present the results and compare to other discontinuous constituent parsers of the recent literature in Table 2. Most of these parsers are discriminative. The parsers by Hall and Nivre (2008), Fernández-González and Martins (2015), and Corro et al. (2017) employ different forms of dependency representation which are converted into constituent structures (dep2const). In contrast, Maier (2015), Maier and Lichte (2016), Coavoux and Crabbé (2017), and Stanojević and Garrido Alhama (2017) employ transition systems (SR-swap, SR-gap, SR-adj-gap) that can produce discontinuous constituent structures directly. Lastly, the chart-based parser of van Cranenburgh et al. (2016) is a generative model that enhances LCFRSs to discontinuous tree substitution grammars where tree fragments are learned according to the *data-oriented parsing* (DOP) paradigm. The results on the HN08 test set are close to the one on the development set. The F1 on the SPMRL test set is more than 4 points lower than in the HN08 split. Other parsers exhibit the same phenomenon that is probably caused by a shift in the distribution to longer sentences. Using predicted POS tags decreases the F1 by 2.38 and 2.00 points for the LCFRS and the hybrid grammar, respectively.

Discussion. The results indicate a strong influence of the granularity of the base grammar's nonterminals. A low granularity results in a higher coverage but also decreases the performance of the base grammar. For instance, for hybrid grammars we see that, despite many parse failures, strict labeling outperforms child labeling with the reranking objective. The drastically lower scores of the reranking objective for $\text{LCFRS}_{\text{r2}\ell}$, LCFRS_{ho} and $\text{hybrid}_{\text{child}}$ in light of the small difference with $\text{hybrid}_{\text{strict}}$ are likely caused by the base grammars being too coarse and, thus, assigning higher probabilities to bad candidates. Moreover, we suppose that including some context in the base grammars' nonterminals helps to guide the split/merge algorithm and avoids overfitting: For $\text{hybrid}_{\text{child}}$ the accuracy drops if we run more than 4 split/merge cycles. Also, in early experiments we observed worse performance if the conditioning context of Markovization for LCFRSs is further restricted. It will be interesting to study hybrid grammars whose base grammars have slightly finer nonterminals than with child labeling.

The EM algorithm is prone to overfitting to the training corpus. In fact, in our experiments we observed that the validation likelihood decreased after some epochs of training whereas the smoothing step counteracted this trend. To improve robustness, we plan to investigate changes in the training regime,

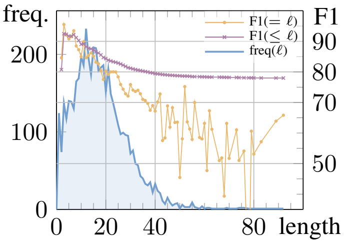

Figure 6: Frequency of sentence length ℓ, F1 for sentences of length ℓ, and F1 of sentences of length $\leq \ell$ on TiGerHN08 dev. set with LCFRS_{ho}/max-rule-product.

	Method	TiGerSPMRL F1 `spmrl/proper`	TiGerHN08 $\ell \leq 40$ F1	EM	F1-fun
Hall and Nivre (2008)	dep2const	- / -	79.93	-	-
Fernández-González and Martins (2015)	dep2const	80.62 / -	**85.53**	-	-
Corro et al. (2017)	dep2const	- / **81.63**	-	-	-
Maier (2015)	SR-swap	- / -	79.52	44.32	-
Maier and Lichte (2016)	SR-swap	- / 76.46	80.02	45.11	-
Coavoux and Crabbé (2017)	SR-gap	**81.50 / 81.60**	85.11	-	-
Stanojević and Garrido Alhama (2017)	SR-adj-swap	- / **81.64**	84.06	-	-
here	LCFRS: head-outward/max-rule-product	75.00 / 75.08	79.29	42.55	67.25
here	hybrid grammar: child/max-rule-product	72.91 / 72.98	77.68	41.28	66.72
† van Cranenburgh et al. (2016)	DOP	- / -	78.2	40.0	68.1
† **here**	LCFRS: head-outward/max-rule-product	- / -	76.91	39.22	64.91
† **here**	hybrid grammar: child/max-rule-product	- / -	75.66	38.40	64.66

Table 2: Evaluation on the test sets. Rows with † use predicted POS tags.

e.g., adding a prior on probability assignments or merging a higher percentage of nonterminal splits.

It is not surprising that the best results are obtained with the projection-based parsing objectives: the loss function of EM training does not guarantee that the probability mass of one derivation is concentrated in a single run. That max-rule-product outperforms the variational approach in terms of F1 and EM is in accordance with the findings and interpretations of Petrov and Klein (2007). For hybrid grammars the improvements of the projection-based methods are smaller than for LCFRSs which may be explained by the additional layer of spurious ambiguity (i.e., multiple derivation trees for one hybrid tree) they exhibit.

The F1 on discontinuous brackets is much lower than the overall F1 where the discontinuous recall is particularly low. Each grammar predicts only between 631 and 989 discontinuous brackets where there are 1837 in the gold standard. This could be affected by the way we approximate the PCFG in the coarse-to-fine pipeline which penalizes discontinuous rules. Most discontinuous brackets are predicted with the reranking objective but, as the lower discontinuous F1 indicates, these brackets are often incorrect.

Table 2 shows that the systems proposed in the literature exhibit a higher F1 than our grammars. This holds in particular for the systems that employ discriminative (and sometimes global) features, which are not available in our system. On the other hand, also van Cranenburgh et al. (2016)'s DOP-based parser is superior to our system in the predicted POS tag scenario. One reason might be error propagation due to wrong POS tags predicted by the external tagger that we use. In contrast, in van Cranenburgh et al. (2016) POS tags are predicted during parsing. Also, the probability assignment that we obtained by EM training may be less robust than the rule probabilities obtained according to the DOP paradigm.

Conclusion. The state refinement method considerably improves over the baseline grammars, which confirms our hypothesis that the usefulness of the split/merge algorithm goes beyond parsing with CFGs. However, at least with the used version of EM training, the initial granularity of the nonterminal set has a decisive impact on the quality of the resulting grammar and should be chosen carefully. When considering the task of discontinuous parsing, the results fall behind recent advances in the literature. This holds in particular for the discriminative deterministic transition-based systems where the representable discontinuity is not restricted by grammar constants, non-local features may be considered, and the parsing is much faster. One remedy to the lower accuracy and speed could be the enhancement of our chart-based method with a discriminative classifier to guide the pruning of the chart (Vieira and Eisner, 2017). In the future one may advance the understanding of the split/merge algorithm by applying it to other IRTGs such as synchronous hyperedge replacement grammars for graph parsing (Peng et al., 2015) or in a task different from parsing like syntax-based machine translation with synchronous grammars (Chiang, 2007).

Acknowledgements

The author thanks the anonymous reviewers for many helpful comments, Markus Teichmann for contributing to the implementation, Heiko Vogler and Tobias Denkinger for feedback on a draft of this paper, and Andreas van Cranenburgh and Toni Dietze for helpful discussions. The implementation utilizes *disco-dop* by Andreas van Cranenburgh, *treetools* by Wolfgang Maier, and the *Eigen* template library.

References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*. Beijing, China, pages 33–36. https://www.aclweb.org/anthology/C10-3009.

Walter S. Brainerd. 1969. Tree generating regular systems. *Information and Control* 14(2):217 – 231. https://doi.org/10.1016/S0019-9958(69)90065-5.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation* 2(4):597–620. https://doi.org/10.1007/s11168-004-7431-3.

Matthias Büchse, Alexander Koller, and Heiko Vogler. 2013. General binarization for parsing and translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 145–154. https://www.aclweb.org/anthology/P13-1015.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics* 33(2):201–228. https://doi.org/10.1162/coli.2007.33.2.201.

Maximin Coavoux and Benoit Crabbé. 2017. 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*. Valencia, Spain, pages 1259–1270. https://www.aclweb.org/anthology/E17-1118.

Michael John Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. http://www.cs.columbia.edu/∼mcollins/papers/thesis.ps.

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*. Copenhagen, Denmark, pages 1644–1654. https://www.aclweb.org/anthology/D17-1172.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38. https://www.jstor.org/stable/2984875.

Pierre Deransart and Jan Małuszynski. 1985. Relating logic programs and attribute grammars. *Journal of Logic Programming* 2(2):119–155. https://doi.org/10.1016/0743-1066(85)90015-9.

Frank Drewes, Kilian Gebhardt, and Heiko Vogler. 2016. EM-training for weighted aligned hypergraph bimorphisms. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. Berlin, Germany, pages 60–69. https://www.aclweb.org/anthology/W16-2407.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland, pages 104–116. https://www.aclweb.org/anthology/W11-2913.

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)*. Beijing, China, pages 1523–1533. https://www.aclweb.org/anthology/P15-1147.

Francis Ferraro, Benjamin Van Durme, and Matt Post. 2012. Toward tree substitution grammars with latent annotations. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*. Montreal, Canada, pages 23–30. https://www.aclweb.org/anthology/W12-1904.

Kilian Gebhardt, Mark-Jan Nederhof, and Heiko Vogler. 2017. Hybrid grammars for parsing of discontinuous phrase structures and non-projective dependency structures. *Computational Linguistics* 43(3):465–520. https://doi.org/10.1162/COLI_a_00291.

Jonas Groschwitz, Alexander Koller, and Mark Johnson. 2016. Efficient techniques for parsing with tree automata. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2042–2051. https://www.aclweb.org/anthology/P16-1192.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In Bengt Nordström and Aarne Ranta, editors, *Advances in Natural Language Processing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 169–180. https://doi.org/10.1007/978-3-540-85287-2_17.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, British Columbia, Canada, pages 53–64. https://www.aclweb.org/anthology/W05-1506.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics* 39(1):87–119. https://doi.org/10.1162/COLI_a_00136.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Sapporo, Japan, pages 423–430. https://www.aclweb.org/anthology/P03-1054.

Donald E. Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters* 6(1):1 – 5. https://doi.org/10.1016/0020-0190(77)90002-3.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland, pages 2–13. https://www.aclweb.org/anthology/W11-2902.

Karim Lari and Steve J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language* 4(1):35 – 56. https://doi.org/10.1016/0885-2308(90)90022-X.

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)*. Beijing, China, pages 1202–1212. https://www.aclweb.org/anthology/P15-1116.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*. San Diego, California, pages 47–57. https://www.aclweb.org/anthology/W16-0906.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*. CSLI Publications, Hamburg, Germany, pages 61–76. https://cslipublications.stanford.edu/FG/2008/maier.pdf.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan, pages 75–82. https://doi.org/10.3115/1219840.1219850.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics* 29(1):135–143. https://doi.org/10.1162/089120103321337467.

Mark-Jan Nederhof and Giorgio Satta. 2004. Kullback-Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland. https://doi.org/10.3115/1220355.1220366.

Mark-Jan Nederhof and Heiko Vogler. 2014. Hybrid grammars for discontinuous parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland, pages 1370–1381. https://www.aclweb.org/anthology/C14-1130.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China, pages 32–41. https://www.aclweb.org/anthology/K15-1004.

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 the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 433–440. https://doi.org/10.3115/1220175.1220230.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York, pages 404–411. https://www.aclweb.org/anthology/N07-1051.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin, Ireland, pages 103–109. https://www.aclweb.org/anthology/W14-6111.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88(2):191–229. https://doi.org/10.1016/0304-3975(91)90374-B.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea, pages 440–448. https://www.aclweb.org/anthology/P12-1046.

Miloš Stanojević and Raquel Garrido Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 1666–1676. https://www.aclweb.org/anthology/D17-1174.

Christoph Teichmann, Alexander Koller, and Jonas Groschwitz. 2017. Coarse-to-fine parsing for expressive grammar formalisms. In *Proceedings of the 15th International Conference on Parsing Technologies (IWPT)*. Pisa, Italy, pages 122–127. https://www.aclweb.org/anthology/W17-6317.

Christoph Teichmann, Kasimir Wansing, and Alexander Koller. 2016. Adaptive importance sampling from finite state automata. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. Berlin, Germany, pages 11–20. https://www.aclweb.org/anthology/W16-2402.

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. https://doi.org/10.15398/jlm.v4i1.100.

Tim Vieira and Jason Eisner. 2017. Learning to prune: Exploring the frontier of fast and accurate parsing. *Transactions of the Association for Computational Linguistics* 5:263–278. https://aclweb.org/anthology/Q17-1019.

Krishnamurti Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*. Stanford, California, USA, pages 104–111. https://doi.org/10.3115/981175.981190.

A Appendix

In this appendix we provide auxiliary calculations and definitions, an example in which the max-rule-product parsing objective is ill-defined, and choices of hyperparameters and preprocessing steps used during the experiments.

A.1 Computing the probability of an object using its chart

In Section 2.2 we state that $P(a \mid \mathbb{G}, p) = \sum_{\xi \in L(G_a)} W_{(G_a, p \circ \varphi_a)}(\xi)$. This follows from:

$$\begin{aligned}
\sum_{\xi \in \text{parses}_{\mathbb{G}}(a)} W_{(G,p)}(\xi) &= \sum_{\xi \in \text{parses}_{\mathbb{G}}(a)} \; \sum_{r \in \text{runs}^v_G(\xi)} \; \prod_{\pi \in \text{pos}(\xi)} p(\text{rule}^\pi_r) \\
&= \sum_{\xi \in L(G_a)} \; \sum_{r' \in \text{runs}^v_{G_a}(\xi)} \; \prod_{\pi \in \text{pos}(\xi)} p(\text{rule}^\pi_{\varphi_a \circ r'}) \\
&= \sum_{\xi \in L(G_a)} \; \sum_{r' \in \text{runs}^v_{G_a}(\xi)} \; \prod_{\pi \in \text{pos}(\xi)} (p \circ \varphi_a)(\text{rule}^\pi_{r'}) \\
&= \sum_{\xi \in L(G_a)} W_{(G_a, p \circ \varphi_a)}(\xi)
\end{aligned}$$

A.2 Approximation of the likelihood loss.

Let G^c be the RTG at the beginning of a particular split/merge cycle and $G^f = \mu_{\text{sp}}^{-1}(G^c)$ be the RTG after splitting. We describe how $\Delta(B_1, B_2)$ is computed for a pair B_1 and B_2 of nonterminals in G^f that are candidates for merging. Similar to Section 3.3, for each $a \in A$ we assume grammar morphisms $\mu'_{\text{sp}} \colon G^f_a \to G^c_a$, $\varphi_a \colon G^c_a \to G^c$, and $\varphi'_a \colon G^f_a \to G^f$ which satisfy $\varphi_a \circ \mu'_{\text{sp}} = \mu_{\text{sp}} \circ \varphi'_a$.

Firstly, we compute *merge factors* p_1 and p_2 based on the relative frequency of B_1 and B_2 where

$$p_i = \frac{\text{freq}(B_i)}{\text{freq}(B_1) + \text{freq}(B_2)} \quad .$$

To compute the expected frequency of B_i ($i \in \{1,2\}$), we sum over the expected frequency of each occurrence B'_i of B_i in some chart G^f_a:

$$\text{freq}(B_i) = \sum_{a \in A} c_A(a) \cdot \sum_{B'_i \in {\varphi'_a}^{-1}(B_i)} \frac{\alpha(B'_i) \cdot \beta(B'_i)}{\beta(S_{G^f_a})} \quad .$$

Secondly, we consider pairs (B'_1, B'_2) of occurrences of B_1 and B_2 in some chart G^f_a such that $\mu'_{\text{sp}}(B'_1) = \mu'_{\text{sp}}(B'_2)$. For each such pair, we introduce a hypothetical nonterminal B' which symbolizes merging just B'_1 and B'_2. Its inside and outside weight is obtained by $\alpha(B') = \alpha(B'_1) + \beta(B'_2)$ and $\beta(B') = p_1 \cdot \beta(B'_1) + p_2 \cdot \beta(B'_2)$, respectively.

Using these hypothetical nonterminals, we approximate the loss in likelihood due to merging B_1 and B_2 by accumulating likelihood losses due to merging pairs of occurrences:

$$\Delta(B_1, B_2) = \prod_{\substack{a \in A \\ (B'_1, B'_2) \text{ in } G^f_a}} \left(\frac{\beta(S_{G^f_a}) \; + \; \alpha(B') \cdot \beta(B') - \left(\sum_{i \in \{1,2\}} \alpha(B'_i) \cdot \beta(B'_i) \right)}{\beta(S_{G^f_a})} \right)^{c_A(a)} .$$

In the above formula, the numerator shall express the probability of the chart after merging and the denominator expresses the probability before merging. If a nonterminal A' occurs more than once in a run of the chart, then the sum of the probability of all runs that contain A' is not equal to its expected frequency $\alpha(A') \cdot \beta(A')$. Thus, the above formula is reasonable under the assumption that each nonterminal A' in N_{G_a} occurs at most once in a run on a derivation tree in $L(G^f_a)$. This assumption is violated if, e.g., the chart contains a chain rule $A' \to f(A')$.

A.3 The max-rule-product objective can be ill-defined

We give an example of a CFG with chain rules where max-rule-product is an ill-defined parsing objective. Consider the refined CFG G^f with nonterminals $\{S, A_1, A_2, A_3\}$, terminals $\{a\}$, and rules with probabilities:

$$\begin{aligned} S &\to A_i && \#1.0 \text{ if } i = 1 && \text{else } 0.0 \\ A_i &\to A_j && \#1.0 \text{ if } j = i + 1 && \text{else } 0.0 \\ A_i &\to a && \#1.0 \text{ if } i = 3 && \text{else } 0.0 \end{aligned}$$

Consider the chart G_a^f for the string a which is isomorphic to G^f. The inside and outside weight of each nonterminal of G_a^f is 1.0. If we merge A_1, A_2, and A_3 to A and project weights according to the max-rule-product principle, then we obtain the CFG G_a^c with weights

$$\begin{aligned} S &\to A && \#1.0 \\ A &\to A && \#2.0 \\ A &\to a && \#1.0 \end{aligned}$$

Now the weight of some parse tree of G_a^c is exponential in the number of occurrences of the chain rules $A \to A$ it contains. Consequently, there cannot be a best tree.

A.4 Hyperparameters and preprocessing

The TiGer corpus is preprocessed before grammar induction. Specifically, punctuation tokens are attached to lower nodes in order to reduce the number of discontinuous brackets using *disco-dop*. Also, we replace words with less than 4 occurrences in the training corpus by a fresh "UNKNOWN" symbol. In addition, the training set and the validation set are enriched by copies of the constituent trees where every word is replaced by the "UNKNOWN" symbol. These copies get assigned frequency 0.1 in either set.

During the split/merge algorithm EM training is applied as follows. The probability assignment p_i of the i-th epoch ($m \leq i \leq 20$) with the best validation likelihood is selected as result. We set $m = 2$ after smoothing and $m = 5$ otherwise. Also, we skip all remaining EM epochs, if validation likelihood dropped in 6 consecutive epochs. When computing validation likelihood, we ignore trees with probability 0 except after smoothing.