Multiword Expression-Aware A* TAG Parsing Revisited

Jakub Waszczuk
LIFO
Université d'Orléans
6, rue Léonard de Vinci
45067 Orléans, France

Agata Savary
Laboratoire d'Informatique
Université François-Rabelais Tours
3, place Jean-Jaurès
41000 Blois, France

Yannick Parmentier
LIFO
Université d'Orléans
6, rue Léonard de Vinci
45067 Orléans, France

firstname.lastname@univ-{orleans|tours}.fr

Abstract

A* algorithms enable efficient parsing within the context of large grammars and/or complex syntactic formalisms. Besides, it has been shown that promoting multiword expressions (MWEs) is a beneficial strategy in dealing with syntactic ambiguity. The state-of-the-art A* heuristic for promoting MWEs in tree-adjoining grammar (TAG) parsing has certain drawbacks: it is not monotonic and it composes poorly with grammar compression techniques. In this work, we propose an enhanced version of this heuristic, which copes with these shortcomings.

1 Introduction

In the domain of syntactic parsing, a growing interest is dedicated to A* parsing strategies (Klein and Manning, 2003) which allow to compute the most plausible parse tree(s) without having to generate the space of all the grammar-compliant solutions in advance. Such strategies enable efficient parsing within the context of large grammars and/or complex syntactic formalisms (Angelov and Ljunglöf, 2014). They were also shown to finely combine with probabilistic supertagging, which can be used to pre-score the individual solutions and, hence, guide the parser to quickly find the most probable one(s) (Lewis and Steedman, 2014). Once supertagging is correctly performed, determining the corresponding syntactic structure is greatly simplified (Bangalore and Joshi, 1999), and A* parsing allows to backtrack and correct the possible misestimations of the supertagger.

Lewis and Steedman (2014) showed that it is possible to obtain a highly accurate and fast CCG parser by (i) adopting a simple probabilistic model factored on lexical category assignments and (ii) using supertagging techniques to obtain relatively reliable probability estimations in a sentence-dependent manner. However, it seems that this proposal, even though quite generic, is not easy to extend to the context of a lexicalized grammar in which various categories of multiword expressions (MWEs) are represented as elementary grammar units, which is a standard approach to modeling MWEs in tree-adjoining grammars, TAGs (Abeillé and Schabes, 1989; Abeillé, 1995). While adapting the (Lewis and Steedman, 2014) approach to continuous MWEs, such as *all of a sudden* or *a hot dog*, could be achieved by using word lattices to represent ambiguous input segmentations (Constant et al., 2013), non-continuous MWEs (as in *he is making no good decisions*) seem harder to account for.

At the same time, Wehrli (2014) showed that promoting collocation-based derivations in syntactic parsing – MWEs often are statistical collocations – is potentially beneficial in that it helps to deal with syntactic ambiguity. MWEs account for up to 40% of words in a corpus (Gross and Senellart, 1998) and, due to their lexicalized nature, should be easy to spot before parsing, which suggests the usefulness of enriching A* parsing strategies with MWE-dedicated mechanisms.

We previously proposed such a mechanism (Waszczuk et al., 2016b), extending the (Lewis and Steedman, 2014) heuristic to a variant allowing multi-anchored TAG elementary trees (ETs). We showed that considerable speed-up gains can be achieved by promoting MWE-based analyses with virtually no loss in syntactic parsing accuracy. This is in contrast to purely statistical approaches where the idiosyncratic properties of MWEs are hard to capture, and systematically promoting MWEs may decrease the parser's accuracy due to the lack of control of the underlying grammar which would certify the plausibility of the re-

Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+13), pages 84–93,
Umeå, Sweden, September 4–6, 2017.

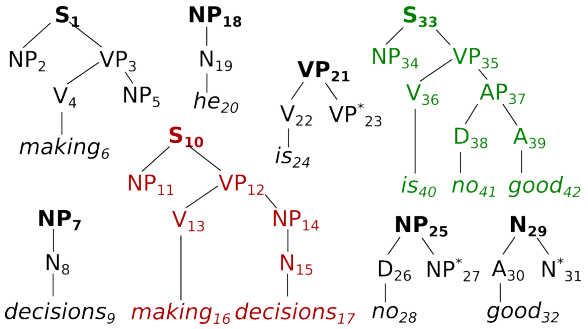

Figure 1: A toy TAG grammar

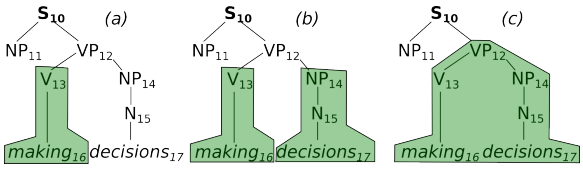

Figure 2: Traversal configurations

sulting derivations.

However, the MWE-based A* heuristic proposed in (Waszczuk et al., 2016b) has three important drawbacks. Firstly, it lacks the proof of correctness and, therefore, it is not clear whether the first derivation found by the parser is indeed the most probable one, a property normally assured by a well-defined A* heuristic. Secondly, it produces sub-optimal estimations – the bottom-up Early-style parser the heuristic relies on (Waszczuk et al., 2016a) performs predictions in order to identify the spans over which adjunction can be potentially performed, but the inside probabilities related to such spans are ignored. Finally, computing the values of the heuristic is relatively easy (it can be performed in close to constant time), but it composes badly with grammar compression. In a real-world context, different parsing speed-up optimizations should ideally combine to provide an optimal solution. In this work, we propose an enhanced version of this heuristic, which copes with the three drawbacks mentioned above.

We first describe the baseline parser and its heuristic, and give a motivating example demonstrating its drawbacks (Sec. 2). Then we formalize a compressed grammar representation (Sec. 3) and our new parser adapted to it (Sec. 4). We introduce the enhanced heuristic and sketch a proof of its monotonicity (Sec. 5). Finally we present some experimental results (Sec. 6), as well as conclusions and future work (Sec. 7).

2 Baseline heuristic

In this section, we shortly review the heuristic proposed in (Waszczuk et al., 2016b) and point out its shortcomings. We assume the toy TAG grammar from Fig. 1, where each node is marked with a unique ID. This grammar contains two ETs representing two possibly discontinuous verbal MWEs (*making decisions* and *is no good*). We also assume an input sentence $s = s_1 \dots s_n$ being a sequence of terminal symbols.

The parser in (Waszczuk et al., 2016b) is based on *deduction rules* (Shieber et al., 1995), which serve to infer *chart items*. Each chart item is a pair $\langle x, r \rangle$, where x is a *configuration* and r is a *span*. Each configuration x represents a position in the traversal of the corresponding ET t_x, and each span r represents a fragment of the input sentence. Informally, $\langle x, r \rangle$ asserts that the already traversed part of t_x can be matched against the words in r.

Fig. 2 shows three different configurations corresponding to the traversal of the ET rooted at S_{10}. For instance, Fig. 2 (a) stipulates that the parser has already matched the nodes VP_{13} and $making_{16}$, and that it still needs to match the remaining nodes in the ET. We will refer to the configurations by the nodes at which they are rooted – i.e., to Fig. 2 (a), (b), and (c) by $\{V_{13}\}$, $\{V_{13}, NP_{14}\}$, and $\{VP_{12}\}$, respectively. Any configuration x determines the split of the terminals present in t_x into two parts: (i) the terminals already scanned by the parser, and (ii) the remaining terminals, which the parser still needs to match against the input words. We denote by $inf(x)$ and $sup(x)$ the multiset of terminals (written $\{\}_{ms}$) in part (i) and (ii), respectively. For instance, $inf(\{V_{13}\}) = \{making\}_{ms}$, $sup(\{V_{13}\}) = \{decisions\}_{ms}$, $inf(\{AP_{37}\}) = \{no, good\}_{ms}$, $sup(\{AP_{37}\}) = \{is\}_{ms}$, etc.[1]

Let $pos(s) = \{0, \dots, n\}$ be the set of positions between the words in s, before s_1 and after s_n. In TAGs, the yield of an ET can span two non-adjacent fragments of s, hence a span takes the form of a tuple $r = \langle i, j, k, l \rangle$, where $i, l \in pos(s)$ and $j, k \in pos(s) \cup \{-\}$. $\langle j, k \rangle$, when defined, represents the *gap* between the two non-adjacent fragments $\langle i, j \rangle$ and $\langle k, l \rangle$. For instance, in Fig. 3, $\langle 2, -, -, 6 \rangle$ corresponds to continuous *making no good decisions*, $\langle 2, 3, 5, 6 \rangle$ corresponds to discontinuous *making decisions* with

[1] We extend the standard set operations (sum, difference, etc.) to multisets.

the gap *no good*, etc. We will refer to spans of the form $\langle i, -, -, l\rangle$ as $\langle i, l\rangle$ for short. Each span r determines the split of the terminals present in the input sentence into: (i) the terminals in r, and (ii) the terminals outside r. We denote by $in(r)$ and $out(r)$ the multiset of terminals in part (i) and (ii), respectively. For instance, $in(\langle 2, 6\rangle) = \{making, no, good, decisions\}_{ms}$, $out(\langle 2, 6\rangle) = \{he, is\}_{ms}$, $in(\langle 2, 3, 5, 6\rangle) = \{making, decisions\}_{ms}$, $out(\langle 2, 3, 5, 6\rangle) = \{he, is, no, good\}_{ms}$, etc.

Given a deduced chart item $\langle x, r\rangle$, it holds that $inf(x) \subset in(r)$. Moreover, if $sup(x) \not\subset out(r)$, then the item is a dead-end – no final derivations (i.e. the derivations covering the entire input sentence) based on x can be created (the tokens remaining to parse do not contain all the terminals present in the remaining part of t_x's traversal).

To each derivation constructed via the deduction rules the parser assigns a *weight* which allows to discriminate the more probable (lighter) from the less probable (heavier) derivations. Given a chart item η, the goal of an A* heuristic $h(\eta)$ is to estimate $\alpha(\eta)$, the minimal outside derivation weight (roughly, the cost remaining to parse the entire input sentence). Formally, $\alpha(\eta) = \gamma(\eta) - \beta(\eta)$,[2] where $\gamma(\eta)$ is the minimal weight of a final derivation containing η, and $\beta(\eta)$ is the minimal weight of an inside derivation of η (roughly, the cost of the already parsed part of the sentence).

In (Waszczuk et al., 2016b) we assume a simple weighting scheme in which each ET t comes with its own weight $\omega_t \geq 0$ and the weight of a derivation is the sum of the weights of the participating ETs. Consider Fig. 3, where each ET has weight 1, and two items: $\eta_{gr} = \langle\{VP_{21}\}, \langle 1, 2, 6, 6\rangle\rangle$ corresponding to the derivation marked in green (solid line), and $\eta_{bl} = \langle\{VP_{12}\}, \langle 2, 6\rangle\rangle$ corresponding to the derivation marked in blue (dashed line). Then, $\beta(\eta_{gr}) = 1$ and $\beta(\eta_{bl}) = 2$.

A heuristic should be *admissible*, i.e., never overestimate $\alpha(\eta)$, and *monotonic*. Monotonicity means that, when a chart item η is inferred from another item μ contained in its lowest-weight derivation, $h(\eta) + \beta(\eta)$ should be at least as high as $h(\mu) + \beta(\mu)$ (Klein and Manning, 2002).

The A* heuristic presented in (Waszczuk et al., 2016b) is based on the observation that the weight of a particular TAG derivation tree can be reformulated as follows. Firstly, the weights of the individual ETs are projected over the words in the input sentence to which they are attached. There are several possible ways of doing that, and the simplest one is to evenly distribute the weight of a given ET over its terminals, as shown in Fig. 3. Secondly, the weight of a derivation tree is redefined as the sum of the weights projected over the words in the sentence.

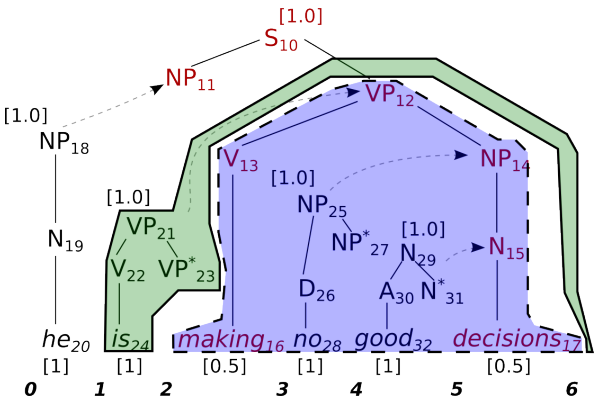

Figure 3: Projecting the weights of the ETs in a TAG derivation on the corresponding terminals. The weights of the ETs are shown, in square brackets, above their roots, while the projected weights are shown, also in square brackets, below the terminals.

When the parser considers a particular item $\langle x, r\rangle$, the weights which can be projected over the words in r are known, but not the weights which can be projected over the words outside r. One can easily find the lower-bound estimates of the latter – i.e., assume that the minimum possible weight, denoted $minw(w)$ (e.g., $minw(\text{is}) = \frac{1}{3}$, $minw(\text{making}) = \frac{1}{2}$, etc.), will be projected over each word w outside r. Consequently, in (Waszczuk et al., 2016b) we define the baseline heuristic as:

$$h(\langle x, r\rangle) = \begin{cases} \mathcal{C}(out(r)), \text{ if } comp(x) \\ \infty, \text{ if } sup(x) \not\subset out(r) \\ \omega_{t_x} + \mathcal{C}(out(r) \setminus sup(x)), \text{ o/w,} \end{cases}$$

where $\mathcal{C}(m)$ is the globally minimal cost of scanning all the words in the multiset m (the sum of their $minw$ values) and $comp(x)$ is true iff the traversal represented by x is complete (all the

[2] This equation holds provided that the weighting function is monotonic in the sense of (Huang and Chiang, 2005).

nodes in t_x have been parsed). For instance:

$$\begin{aligned} h(\eta_{gr}) &= \mathcal{C}(out(\langle 1,2,6,6\rangle)) \\ &= \mathcal{C}(\{\text{he}, \text{making}, \text{no}, \text{good}, \text{decisions}\}_{ms}) \\ &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} + \frac{1}{2} = 2 + \frac{2}{3}, \\ h(\eta_{bl}) &= 1 + \mathcal{C}(\{\text{he}, \text{is}\}_{ms}) = 2 + \frac{1}{3}. \end{aligned}$$

The heuristic does not take the weight ω_{t_x} of the ET t_x being parsed into account if x is complete. This is because ω_{t_x} is already added to the weight of the inside derivation of $\langle x, r\rangle$ in this case. Moreover, $comp(x) \implies sup(x) = \emptyset_{ms}$.

This heuristic presents some important shortcomings, foremostly non-monotonicity. Given the predictive nature of the parser, an item $\langle x, \langle i, j, k, l\rangle\rangle$ such that $\langle j, k\rangle \neq \langle -, -\rangle$ is inferred iff an appropriate derivation, spanning $\langle j, k\rangle$, exists. In Fig. 3, η_{gr} thus relies on η_{bl} and when η_{gr} is inferred, the weights which can be projected over the words in $\langle 2, 6\rangle$ are already known. Nevertheless, $h(\eta_{gr})$ assumes that the globally minimal weights will be projected over $\langle 2, 6\rangle$, thus underestimating the weights projected over the words *no* and *good*. In Fig. 3, these projections are equal to 1, while $minw(no) = minw(good) = \frac{1}{3}$, since both words belong to the MWE *is no good*. As a result, $h(\eta_{gr}) + \beta(\eta_{gr}) = 2\frac{2}{3} + 1$ is smaller than $h(\eta_{bl}) + \beta(\eta_{bl}) = 2\frac{1}{3} + 2$, which shows that the heuristic is not monotonic.

A similar situation occurs in top-down CFG parsing with prediction when the weight of the premise item of the prediction rule is not transferred to the conclusion. Nederhof (2003) shows that a simplified version of the A* algorithm (the Knuth's algorithm, similar to the standard Dijkstra's shortest-path algorithm and relying on no heuristic) still correctly computes the derivations with the lowest weights in this case. While it can be stipulated that his proof applies to the algorithm used in (Waszczuk et al., 2016b), ignoring the weights of the items used for adjunction-related predictions not only makes the heuristic non-monotonic, it also means that the already computed inside weights, which could provide better estimations of $\alpha(x)$, are sometimes ignored.

3 Grammar representation

The complexity of TAG parsing is polynomial in the sentence length and linear in the grammar size ($\mathcal{O}(n^6 * |G|)$) (Gardent et al., 2014). With real-size grammars, especially those containing explicitly encoded MWEs, the latter factor can be prohibitive. Therefore – in order to speed up parsing – an A* algorithm should ideally be combined with grammar compression, as proposed below.

We assume a twofold representation of a TAG grammar. It is first transformed into an equivalent directed acyclic graph with ordered outgoing edges for each node (GDAG). Then, traversals of the ETs and their subtrees are represented as paths in finite-state automata (FSAs). This is a simplified variant of the grammar encoding applied in (Waszczuk et al., 2016a), where we showed that grammar compression alone can greatly speed-up TAG parsing, while using a variant of the standard, bottom-up Earley-style parser (Alonso et al., 1999).

Formally, we define a GDAG as a tuple $D = \langle V_D, E_D, \Sigma_D, N_D, S_D, \ell_D, foot_D\rangle$ such that V_D and E_D are the sets of DAG nodes and (ordered) edges, respectively, Σ_D and N_D are the sets of terminals and non-terminals, respectively, $S_D \in N_D$ is the start symbol, $\ell_D \colon V_D \to \Sigma_D \cup N_D$ is a function assigning non-terminals or terminals to D's nodes, and $foot_D \colon V_D \to \mathbb{B}$ tells whether a given node is a foot node. Also, $R_D, L_D \subset V_D$ are the sets of roots and leaves in D, respectively.

Fig. 1 is a straightforward encoding where each ET is represented by a separate tree. Fig. 4 compares two GDAG representations of three ETs from Fig. 1: (a) the straightforward encoding, and (b) the encoding with common subtrees shared among ETs. Both representations are equivalent, i.e., they entail the same TAG, which can be obtained by the traversals of the sub-DAGs rooted in the GDAG roots. For instance, the traversals starting from S_{10} in (a) and (b) entail the same ET.

We assume that a GDAG satisfies all the relevant, TAG-related well-formedness constraints – for example, that for each $r \in R_D$ there exists at most one $v \in V_D$ reachable from r such that $foot_D(v)$. Moreover, if such $v \in V_D$ exists, then $\ell_D(r) = \ell_D(v) \in N_D$, i.e., the root and the foot of an auxiliary tree must be labeled with the same non-terminal. Each $v \in V_D$ unambiguously determines the corresponding ET subtree (ES) *rooted at* v, denoted $es_D(v)$. Note that, by definition, each ET is also an ES.

Fig. 5 illustrates two possible FSA encodings, without and with prefix sharing, of the GDAG

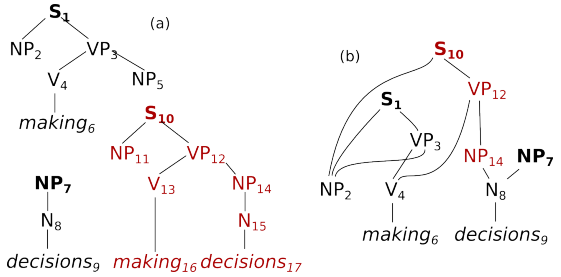

Figure 4: Two possible DAG encodings of three ETs from the grammar in Fig. 1. The values of ℓ_D are put in place of the corresponding nodes, and the node identifiers (the values of V_D) are placed in subscript on the right. All the edges are implicitly oriented downwards.

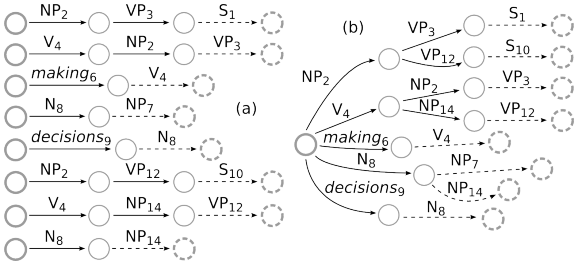

Figure 5: Two possible FSA encodings of the GDAG illustrated in Fig. 4 (b): (a) a simple FSA encoding in which each GDAG traversal is represented by a distinct path, and (b) a prefix-tree-like FSA encoding.

from Fig. 4 (b).[3] From the parsing perspective, each path in an FSA encoding represents the left-to-right traversal performed by the parser while matching, against the input words, a particular ES of height greater than 0.

Formally, we define an FSA encoding of a GDAG D as a tuple $M = \langle Q_M, \delta_M, S_M, heads_M \rangle$ such that Q_M is the set of FSA states, $\delta_M \colon Q_M \times V_D \rightarrow Q_M$ is the transition function consuming DAG nodes (V_D thus constitutes the set of the FSA alphabet symbols), $S_M \subset Q_M$ is the set of start states, and $heads_M \colon Q_M \rightarrow 2^{V_M}$ is a function which returns the final symbols outgoing from a given state. For convenience, we represent the particular states in the FSA encoding as dotted rules, e.g., $S_1 \rightarrow NP_2 \bullet VP_3$ corresponds to the state reached by following the symbol NP_2 on the path (NP_2, VP_3, S_1). Informally, $heads_M$ represent the left-hand-sides of such dotted rules.

Each $x \in V_D \cup Q_M$ represents a particular traversal configuration (cf. Sec. 2): $x \in V_D$ stipulates that the parser has matched $es_D(x)$, while $x \in Q_M$ stipulates that it has matched all the ESs on the path from one of the states in S_M to $x \in Q_M$. For instance, $\text{VP}_3 \rightarrow \text{V}_4\text{NP}_2\bullet$ stipulates that the parser has matched both $es_D(V_4)$ and $es_D(NP_2)$.

As mentioned in Sec. 2, when no grammar compression (i.e. no subtree or prefix sharing) occurs, each traversal configuration corresponds to exactly one ET t_x. With grammar compression, however, there can be many ETs corresponding to a given $x \in V_D \cup Q_M$. We denote their set by T_x. For instance, in Fig. 4 (b) and 5 (b), $T_{V_4} = T_{\text{VP}_3 \rightarrow V_4 \bullet \text{NP}_2} = T_{\text{VP}_{12} \rightarrow V_4 \bullet \text{NP}_{14}} = \{es_D(S_1), es_D(S_{10})\}$, $T_{\text{NP}_7 \rightarrow \bullet N_8} = \{es_D(v) \colon v \in R_D\}$, etc.

This brings us to the second issue with the heuristic defined in Sec. 2. One of the first steps of computing the value of $h(\langle x, r \rangle)$ is to check whether $sup(x) \subset out(r)$. This step allows to obtain better estimations of $\alpha(\langle x, r \rangle)$ – namely, to exclude the dead-end chart items, which cannot possibly lead to final derivations. However, with grammar compression, sup depends not only on the configuration x, but also on the corresponding ET $t \in T_x$. Hence, we define a generalized version of sup as a two-argument function $sup(x, t)$ which determines the multiset of words remaining to parse on t's traversal starting from x.

When estimating $\alpha(\langle x, r \rangle)$, the parser needs to consider all the different ET traversals containing x, leading to different ETs, and with different multisets of the words remaining to complete the traversal. Assuming the prefix-tree grammar compression and that q_0 is the root of the prefix-tree FSA encoding, T_{q_0} is the set of all grammar ETs. Therefore, when grammar compression is used, computing h's values is $\mathcal{O}(N)$, where N is the number of ETs.

4 Enhancing the baseline A* TAG parser

In (Waszczuk et al., 2016b,a) we put forward a version of the bottom-up, Earley-like parser described in (Alonso et al., 1999) to test the idea of promoting MWEs in A* parsing. We now formalize an enhanced version of this parser, adapted to

[3] While a minimal FSA encoding, including both prefix and suffix sharing, might seem optimal, we empirically showed in (Waszczuk et al., 2016a) that it does not bring significant improvements over the prefix-tree encoding in TAG parsing.

AX (axiom):	$\frac{}{(0,0):\langle q_0,\langle i,i\rangle\rangle_a}$	$i\in pos(s)\backslash\{n\}$ $q_0\in S_M$
SC (scan):	$\frac{(w,w'):\langle q,\langle i,j,k,l\rangle\rangle_a}{(w,w'):\langle \delta_M(q,v),\langle i,j,k,l+1\rangle\rangle_a}$	$v\in L_D:\ \ell_D(v)=s_{l+1}$ $\delta_M(q,v)$ defined
DE (deactivate):	$\frac{(w,w'):\langle q,\langle i,j,k,l\rangle\rangle_a}{(w+[\omega_{es_D(v)}\|v\in R_D],w'):\langle v,\langle i,j,k,l\rangle\rangle_p}$	$v\in heads_M(q)$
PS (pseudo-subst.):	$\frac{(w_1,w_1'):\langle q,\langle i,j,k,l\rangle\rangle_a \quad (w_2,w_2'):\langle v,\langle l,j',k',l'\rangle\rangle_p}{(w_1+w_2,w_1'+w_2'):\langle \delta_M(q,v),\langle i,j\cup j',k\cup k',l'\rangle\rangle_a}$	$\delta_M(q,v)$ defined
SU (substitution):	$\frac{(w_1,w_1'):\langle q,\langle i,j,k,l\rangle\rangle_a \quad (w_2,0):\langle v,\langle l,l'\rangle\rangle_p}{(w_1+w_2,w_1'):\langle \delta_M(q,v'),\langle i,j,k,l'\rangle\rangle_a}$	$v'\in L_D:\ \ell_D(v')=\ell_D(v)\wedge\neg foot_D(v')$ $\delta_M(q,v')$ defined $v\in R_D$
FA (foot-adjoin):	$\frac{(w_1,0):\langle q,\langle i,l\rangle\rangle_a \quad (w_2,w_2'):\langle v,\langle l,j',k',l'\rangle\rangle_p}{(w_1,w_2+w_2'+[A(v)\|v\notin R_D]):\langle \delta_M(q,v'),\langle i,l,l',l'\rangle\rangle_a}$	$v'\in L_D:\ \ell_D(v')=\ell_D(v)\wedge foot_D(v')$ $\delta_M(q,v')$ defined $v\in R_D \implies (j',k')=(-,-)$
RA (root-adjoin):	$\frac{(w_1,w_1'):\langle w,\langle i,j,k,l\rangle\rangle_p \quad (w_2,w_2'):\langle v,\langle j,j',k',k\rangle\rangle_p}{(w_1+w_2,w_2'):\langle v,\langle i,j',k',l\rangle\rangle_p}$	$w\in R_D\wedge(j,k)\neq(-,-)$ $\ell_D(w)=\ell_D(v)$ $v\in R_D \implies (j',k')=(-,-)$

Table 1: Weighted inference rules of the Earley-style, bottom-up parser, where: (in PS) $i \cup j$ is equal to i if $j = -$ and j otherwise, and (in DE and FA) $[x|p] = x$ if p is true and $[x|p] = 0$ otherwise.

the grammar representation introduced in Sec. 3, and to the new heuristic defined in Sec. 5.

We say that a chart item $\langle x, r\rangle$ is *active* if $x \in Q_M$, and that it is *passive* if $x \in V_D$. We also write $\langle q, r\rangle_a$ or $\langle v, r\rangle_p$ to refer to an active or a passive item, respectively.

Tab. 1 specifies the inference rules of the parser, using the weighted deductive framework (Shieber et al., 1995; Nederhof, 2003). Each of the inference rules takes zero, one or two chart items on input (*premises*, presented above the horizontal line) and yields a new item (*conclusion*, presented below the line) to be added to the chart if the conditions given on the right-hand side are met. Besides, to each item η in each rule a pair of weights (w, w'), given before the colon, is assigned, thus specifying how to compute the weights corresponding to the conclusion based on the weights corresponding to the premises. The meaning of w' and A (the latter used to compute w's values in the FA rule), both specific to the enhanced heuristic, will be detailed in Sec. 5. The value w, on the other hand, represents the weight of η's inside derivation. The idea of computing pairs of weights using inference rules comes from Nederhof (2003), who showed that such a solution can be used to overcome the non-monotonicity issue in top-down CFG parsing with prediction.

The way the inside weights are computed by the parser corresponds to the assumption that the probability of a derivation is the product of the probabilities of the participating ETs, hence the weight computed for the conclusion item is the sum of the weights of the premise items, with two exceptions. The DE inference rule, responsible for matching full ESs, adds the weight $es_D(v)$, provided that $es_D(v)$ is an ET. The FA rule, used to predict that adjunction over a particular span r is possible, does not transfer the weight of the premise item spanned over r. In the baseline parser, the same behavior was precisely the reason of the non-monotonicity in Fig. 3 discussed in Sec. 2. However, now this weight is accounted for in w', which preserves monotonicity, as shown below.

5 Enhanced heuristic

We now propose an enhanced version of the heuristic described in Sec. 2, with the goal of overcoming the issues related to non-monotonicity and grammar compression. However, for the sake of clarity, we start by assuming that no grammar compression is performed.

As mentioned before, w represents the weight of an *inside* derivation of the corresponding item η. The weight w', on the other hand, represents (roughly) the weight of η's *witness* derivation, i.e., the previously obtained derivation which can fill η's gap. Consider Fig. 6 (b) and three chart items: $\mu_{gr} = \langle S_8, \langle 1, 2, 4, 5\rangle\rangle$, $\mu_{bl} = \langle S_3, \langle 2, 2, 3, 4\rangle\rangle$, and $\mu_{rd} = \langle S_6, \langle 2, 3\rangle\rangle$. Then, the derivation delimited by the green solid line is μ_{gr}'s inside derivation, while the derivation delimited by the blue dashed line is μ_{gr}'s witness derivation. Sim-

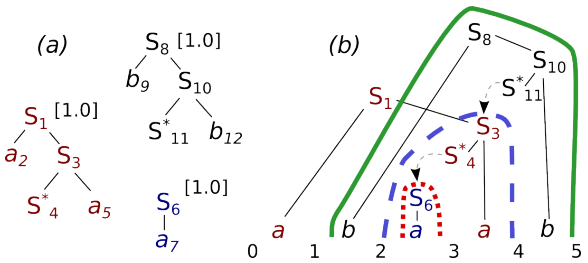

Figure 6: Copy language

ilarly, the derivation delimited by the blue line is μ_{bl}'s inside derivation, while the derivation delimited by the red dotted line is μ_{bl}'s witness. Finally, the derivation delimited by the red line is μ_{rd}'s inside derivation, but μ_{rd} has no witness derivation, since it is not gapped. In general, for any non-gapped item η, $w' = 0$.

Given a configuration $x \in V_D \cup Q_M$, we define x's *amortized weight* as:

$$A(x) = \omega_{t_x} - \mathcal{C}(sup(x)).$$

For instance, in Fig. 6 (a), $A(S_3) = A(S_{10}) = \frac{1}{2}$ (since $minw(a) = minw(b) = \frac{1}{2}$), and $A(S_6) = A(S_1) = 1$. $A(x)$ accounts for the weight of the ET t_x, and for the fact that t_x may still contain some terminals which need to be consumed (w.r.t. to the position of x in t_x's traversal). Thus, $A(x)$ can be intuitively understood as the weight of the already parsed part of t_x.

A version of h which performs no dead-end detection, i.e. never takes the ∞ value unlike h in Sec. 2, but, otherwise, provides identical estimations as h, can be defined in terms of A as:

$$h_{spl}(\langle x, r\rangle) = \begin{cases} \mathcal{C}(out(r)), \text{ if } x \in R_D \\ A(x) + \mathcal{C}(out(r)), \text{ o/w.} \end{cases}$$

We also define the *amortized weight* of a derivation δ as the sum of the amortized weights of the ESs (more precisely, their roots) present in δ. For instance, in Fig. 6 (b), the amortized weight of μ_{bl}'s inside derivation is $A(S_6) + A(S_3) = 1 + \frac{1}{2}$. The weight w', attached to each chart item η, is precisely the amortized weight of η's witness derivation (if η is gapped, otherwise $w' = 0$).

Then, given a chart item $\eta = \langle x, r\rangle$ such that $r = \langle i, j, k, l\rangle$ and the corresponding weight w', we define the enhanced A* heuristic as:

$$h_{adj}(\eta) = \begin{cases} \mathcal{C}(rest(r)) + w', \text{ if } x \in R_D \\ A(x) + \mathcal{C}(rest(r)) + w', \text{ o/w,} \end{cases}$$

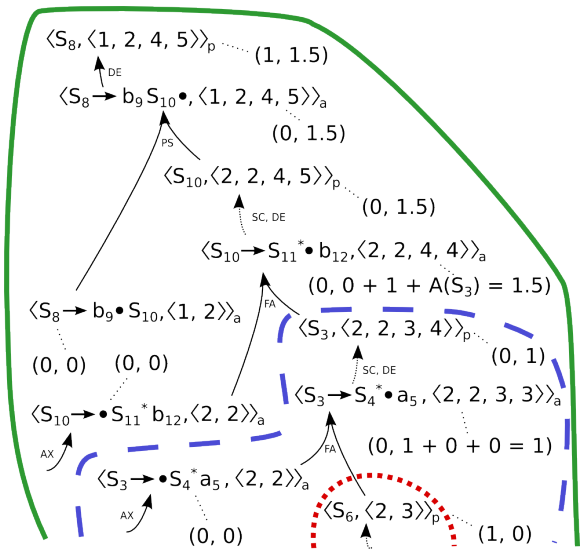

Figure 7: A fragment of an inside derivation of $\langle S_8, \langle 1, 2, 4, 5\rangle\rangle_p$, represented as a hyperpath.

where $rest(r) = out(r) \setminus in(\langle j, k\rangle)$, i.e. the multiset of words outside r but not in the gap, whose cost is already accounted for in w'. The advantage over the baseline heuristic is precisely that, instead of assuming that the lowest possible weights will be projected over all words in the gap, the weights of the existing derivations spanning the gap are considered. The disadvantage is that, by not checking that the remaining part of the sentence contains all the tokens required by t_x's traversal, h_{adj} does not detect the dead-end items, which is the price to pay for better integration with grammar compression techniques (see below).

Fig. 7 shows a fragment of the inference corresponding to μ_{gr}'s inside derivation in Fig. 6 (b). Each node in Fig. 7 represents a deduced chart item, and each hyperarc (which connects one target node with zero, one or two tail nodes) represents an application of an inference rule. Besides, to each item the corresponding pair of weights (w, w') is attached via an undirected dashed edge. In particular, the pair attached to $\langle S_8, \langle 1, 2, 4, 5\rangle\rangle$ is $(1, 1.5)$, since the amortized weight of its witness derivation is 1.5. Note that the baseline heuristic would assume that the globally minimal weights $minw(a) = 0.5$ are projected over both a in the gap and, thus, underestimate the gap's parsing cost as 1.

The enhanced heuristic composes smoothly with grammar compression techniques. To recall, under compression, each configuration $x \in V_D \cup Q_M$ can belong to several ETs and, conse-

quently, to several ET traversals. To ensure that the heuristic does not overestimate, it is sufficient to calculate the *minimal* amortized weight, i.e., assume that the least-weight traversal will be taken from any given $x \in V_D \cup Q_M$:

$$A(x) = \min\{\omega_t - \mathcal{C}(sup(x,t)) \colon t \in T_x\}. \quad (1)$$

The definition of h_{adj} remains unchanged. The values of A can be pre-computed on a per-grammar basis, just as the values of $\mathcal{C} \circ rest$[4] can be pre-computed on a per-sentence basis,[5] hence the robustness of h_{adj} w.r.t. grammar compression.

To show that h_{adj} is monotonic, it is sufficient to consider each inference rule from Tab. 1 separately and to show that the total weight $(w + h_{adj}(\eta))$ it computes for the conclusion item η is at least as high as the total weight of any of its premise items. As an example, we sketch below the proof of monotonicity of the PS rule.

Proposition 1. *Let $v \in V_D$ and $q \in Q_M$ such that $\delta_M(q,v)$ is defined. Then, $T_{\delta_M(q,v)} \subset T(q)$.*

Proof. Follows from the fact that any FSA path crossing $\delta_M(q,v)$ must also cross v. □

Observation 1. *Let $v \in V_D$, $q \in Q_M$ such that $\delta_M(q,v)$ defined, and $t \in T_{\delta_M(q,v)}$. Then, $\mathcal{C}(sup(\delta_M(q,v),t)) = \mathcal{C}(sup(q,t)) - \mathcal{C}(inf(v))$.*

Proposition 2. *Let $v \in V_D$ and $q \in Q_M$ such that $\delta_M(q,v)$ defined. Then,*

$$A(\delta_M(q,v)) \geq A(q) + \mathcal{C}(inf(v)).$$

Proof. Follows from Pr. 1, Ob. 1, and Eq. 1. □

Proposition 3. *Let $\eta = \langle x, r\rangle$ be a chart item such that $x \notin R_D$, $r = \langle i,j,k,l\rangle$, and (w,w') be the corresponding weights. Then,*

$$w + w' \geq \mathcal{C}(in(\langle i,l\rangle)) - \mathcal{C}(inf(x)).$$

Proof. The LHS is the total weight projected by η's inside derivation over $\langle i,l\rangle$, with the exception of the words in $inf(x)$. The RHS is, by definition, a lower-bound for the projected weight. □

Observation 2. *In the PS rule, it holds that: $\mathcal{C}(rest(\langle i,l'\rangle)) = \mathcal{C}(rest(\langle i,l\rangle)) - \mathcal{C}(in(\langle l,l'\rangle))$.*

Proposition 4. *The PS rule is monotonic.*

Proof. Let η be the conclusion and μ be the active premise. Then:

$$\begin{aligned}
&w_1 + w_2 + h_{adj}(\eta) - w_1 - h_{adj}(\mu) = && (h_{adj})\\
&w_2 + A(\delta_M(q,v)) + \mathcal{C}(rest(\langle i,l'\rangle)) + w'_1 + w'_2\\
&\quad - A(q) - \mathcal{C}(rest(\langle i,l\rangle)) - w'_1 = && \text{(Ob. 2)}\\
&w_2 + A(\delta_M(q,v)) + w'_2 - A(q)\\
&\quad - \mathcal{C}(in(\langle l,l'\rangle)) \geq && \text{(Prop. 2)}\\
&w_2 + \mathcal{C}(inf(v)) + w'_2\\
&\quad - \mathcal{C}(in(\langle l,l'\rangle)) \geq 0. && \text{(Prop. 3)}
\end{aligned}$$

It can be shown that PS is monotonic w.r.t. its passive premise item in a similar fashion. □

6 Experiments

We repeated the experimental evaluation proposed in (Waszczuk et al., 2016b) in order to compare the new heuristic (cf. Sec. 5) with the baseline (cf. Sec. 2). The experiment was based on the version of the Składnica treebank (Świdziński and Woliński, 2010) annotated with MWEs (Savary and Waszczuk, 2017). For each sentence in the corpus, the grammar was first reduced to only those ETs whose terminals occurred in the sentence, and compressed according to the methods described in Sec. 3. Then, the parser (specified in Sec. 4) was run and the search-space-size reductions stemming from promoting MWEs (a behavior obtained by assigning the weight 1 to each ET in the grammar) were measured. We also implemented runtime verification tests which empirically confirmed the monotonicity of (the implementation of) h_{adj}.

We observed only minor differences between the results obtained with both heuristics. On average (at most, respectively), h_{adj} led to search-space-size reductions of 16.8% (90.6%) vs. 18.1% (90.7%) reductions obtained with h, a drop in performance related to the lack of dead-end detection in h_{adj}. Recall, however, that calculating h's values can be costly (linear w.r.t. the number of ETs) when grammar compression is used, and can be thus infeasible in practical parsing applications. In our experiment, grammar compression alone decreased (on average) the search-space to less than $\frac{1}{3}$ of its original size.

7 Conclusions and future work

Our previous state-of-the-art A* heuristic for promoting MWEs in TAG parsing is not monotonic

[4] The symbol $\circ$ stands for function composition.

[5] In both cases dynamic programming techniques can be used to speed up the process.

and it composes poorly with grammar compression methods. We have presented here an enhanced version of this heuristic, which improves the estimations of the outside derivation weights by making the predictions of the weights related to items' gaps more accurate. The new version is monotonic and combines with our grammar compression techniques with no significant computational overhead. The price to pay is the lack of detection of dead-end items, which partially explains the slight drop in search-space-size reductions in comparison with the baseline heuristic. To the best of our knowledge, this is the first approach explicitly addressing interactions between A* parsing and grammar compression.

For future work, we plan to repeat the experiment with a truly weighted grammar, i.e., with the weights corresponding to single-anchored ETs being estimated from a treebank. We believe that such an experiment will lead to more insightful conclusions as to the pros and cons of the new heuristic. We also plan to optimize the implementation of the parser, in order to verify that the search-space-size reductions transfer to proportional reductions in parsing time.

Acknowledgments

This work has been supported by the French Ministry of Higher Education and Research via a doctoral grant, by the French Centre-Val de Loire Region Council via the APR-AI 2015-1850 ODIL project, by the French National Research Agency (ANR) via the PARSEME-FR[6] project (ANR-14-CERA-0001), and by the European Framework Programme Horizon 2020 via the PARSEME[7] European COST Action (IC1207).

We are grateful to the anonymous reviewers for their insightful comments to the first version of this paper.

References

Anne Abeillé. 1995. The Flexibility of French Idioms: A Representation with Lexicalised Tree Adjoining Grammar. In M. Everaert, E-J. van der Linden, A. Schenk, and R Schreuder, editors, *Idioms: Structural and Psychological Perspectives*, Lawrence Erlbaum Associates, chapter 1.

Anne Abeillé and Yves Schabes. 1989. Parsing idioms in lexicalized tags. In *Proceedings of the Fourth Conference on European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '89, pages 1–9. https://doi.org/10.3115/976815.976816.

Miguel Alonso, David Cabrero, Eric Villemonte de la Clergerie, and Manuel Vilares Ferro. 1999. Tabular algorithms for TAG parsing. In *EACL 1999*. pages 150–157. http://acl.ldc.upenn.edu/E/E99/E99-1020.pdf.

Krasimir Angelov and Peter Ljunglöf. 2014. Fast Statistical Parsing with Parallel Multiple Context-Free Grammars. In *EACL*. volume 14, pages 368–376.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Comput. Linguist.* 25(2):237–265. http://dl.acm.org/citation.cfm?id=973306.973310.

Matthieu Constant, Joseph Le Roux, and Anthony Sigogne. 2013. Combining compound recognition and PCFG-LA parsing with word lattices and conditional random fields. *ACM Trans. Speech Lang. Process.* 10(3):8:1–8:24. https://doi.org/10.1145/2483969.2483970.

Claire Gardent, Yannick Parmentier, Guy Perrier, and Sylvain Schmitz. 2014. Lexical Disambiguation in LTAG using Left Context. In Zygmunt Vetulani and Joseph Mariani, editors, *Human Language Technology. Challenges for Computer Science and Linguistics. 5th Language and Technology Conference, LTC 2011, Poznan, Poland, November 25-27, 2011, Revised Selected Papers*, Springer, volume 8387, pages 67–79. https://doi.org/10.1007/978-3-319-08958-4_6.

Maurice Gross and Jean Senellart. 1998. Nouvelles bases statistiques pour les mots du français. In *Proceedings of JADT'98, Nice 1998*. pages 335–349.

Liang Huang and David Chiang. 2005. *Proceedings of the Ninth International Workshop on Parsing Technology*, Association for Computational Linguistics, chapter Better k-best Parsing, pages 53–64. http://aclweb.org/anthology/W05-1506.

Dan Klein and Christopher D. Manning. 2002. A* Parsing: Fast Exact Viterbi Parse Selection. Technical Report dbpubs/2002-16, Stanford University.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/N03-1016.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 990–1000. http://aclweb.org/anthology/D14-1107.

[6] http://parsemefr.lif.univ-mrs.fr
[7] http://www.parseme.eu

Mark-Jan Nederhof. 2003. Weighted Deductive Parsing and Knuth's Algorithm. *Comput. Linguist.* 29(1):135–143. https://doi.org/10.1162/089120103321337467.

Agata Savary and Jakub Waszczuk. 2017. Projecting multiword expression resources on a polish treebank. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*. Association for Computational Linguistics, Valencia, Spain, pages 20–26. http://www.aclweb.org/anthology/W17-1404.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming* 24(1):3–36.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016a. Enhancing practical TAG parsing efficiency by capturing redundancy. In *21st International Conference on Implementation and Application of Automata (CIAA 2016)*. Séoul, South Korea, Proceedings of the 21st International Conference on Implementation and Application of Automata (CIAA 2016). https://hal.archives-ouvertes.fr/hal-01309598.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016b. Promoting multiword expressions in a* tag parsing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 429–439. http://aclweb.org/anthology/C16-1042.

Eric Wehrli. 2014. The relevance of collocations for parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 26–32. http://www.aclweb.org/anthology/W14-0804.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*. Springer-Verlag, Heidelberg, volume 6231 of *Lecture Notes in Artificial Intelligence*, pages 197–204.