Regular Approximation as a Heuristics for A* Parsing[⋆]

Daniel Quernheim and Christoph Teichmann

[1] Daniel.Quernheim@uni-potsdam.de
Universität Potsdam, Institut für Linguistik
Karl-Liebknecht-Str. 24-25, D-14476 Potsdam
[2] rightlinear@googlemail.com
Topic Maps Lab; Johannisgasse 26, D-04103 Leipzig

Abstract. Parsing probabilistic context-free grammars generated from treebanks can be made more efficient by employing heuristics to reduce the search space. Klein and Manning (2003) applied A* search to parsing and achieved a huge efficiency gain using several search estimates which rely on grammar transformation and context summaries. We review ideas that have been published and propose a new estimate based on regular approximation. Since recognition in finite automata is linear in time, the estimate can be used exhaustively. We prove that this method is a (theoretically) valid (admissible and monotonic) estimate. This is work in progress as it has not yet been implemented.

1 Introduction

Context-free grammars (CFG) generated from treebanks are usually probabilistic. As for ordinary CFGs, there are parsing algorithms with worst-case cubic time bounds for probabilistic CFGs as well. As this might not be practical for long sentences, strategies for speeding up parsing are invaluable. A variety of different heuristics has been proposed. Among them are greedy strategies which prune the search space, either only processing a fixed number of parse candidates or processing the candidates in an order determined by a *figure-of-merit* (FOM). The former is called *beam search*, while the latter is called *best-first parsing*. However, both of these approaches fail to guarantee that the first parse to be found is the *Viterbi parse* (the actual best parse).

Considering this, Klein and Manning [1] applied *A* search*, a method used in a lot of other contexts, to chart parsing. A* search always finds the Viterbi parse provided the estimate used is *admissible* and *monotonic*. The crucial problem is to find estimates which are valid but sophisticated enough to reduce the amount of computation. Klein and Manning proposed two such estimates which are based on *grammar transformation* and *context summaries* and report a reduction to 3% of edges to be processed, compared to exhaustive search.

[⋆] We are thankful to the anonymous reviewers for their helpful comments, and to Concha Höfler and Joseph DeVeaugh-Geiss for proofreading.

We propose a new estimate based on approximating the grammar by weighted finite automata. We show how the accuracy can be improved by equipping the automata with a bag and prove our estimate to be valid.

2 Preliminaries

This section will introduce basic concepts of language and automata theory that are used in this paper [2, 3].

Definition 1. *A* context-free grammar *(CFG) is a* 4*-tuple* $\langle \Sigma, N, P, S\rangle$*, where* Σ *is a finite set of* terminals*,* N *is a finite set of* nonterminals *and* $\Sigma \cap N = \emptyset$*. We call* $V = N \cup \Sigma$ *the* vocabulary*.* $S \in N$ *is the* start symbol*, and* P *is a finite set of* rules $A \to \alpha$*, where* $A \in N$ *and* $\alpha \in V^*$*.*

The relation $\to$ *is extended to* $V^* \times V^*$ *and then written* $\Rightarrow$*. Its reflexive and transitive closure is denoted* $\Rightarrow^*$ (derivation)*, and the* language *generated by a grammar* G *is* $L(G) = \{w \in \Sigma^* : S \Rightarrow^* w\}$*. We call a CFG* separated *if every rule is of the form* $A \to B$ *with* $B \in N^*$ *or* $C \to a$ *with* $a \in \Sigma$*.*

Some conventions: we usually write $a, b, c, \ldots$ for terminals (also called *alphabet symbols*); $\alpha, \beta, \gamma, \ldots$ for elements of V and $\ldots x, y, z$ for elements in Σ^*.

Definition 2. *A* probabilistic CFG (PCFG) *is a CFG with a* weight function $w : P \to \{x \in \mathbb{R} : 0 \le x \le 1\}$ *such that alternative rules for the same nonterminal have a combined weight (the sum of their weights) of* 1*.*

We will denote a weighted rule by $A \to \alpha \,/\, w$ where w is the associated weight. Along derivations, weights are multiplied, while they are added for alternative derivations. The weight of a derivation represents its probability; the Viterbi parse is defined for a word as the derivation with the highest probability. It is useful to transform probabilities into their negative logarithm and call them *costs*. With this terminology, the Viterbi parse is the derivation with least cost.

Definition 3. *A* semiring *is a* 5*-tuple* $\langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}\rangle$*, where* $\langle \mathbb{K}, \oplus, \overline{0}\rangle$ *is a commutative monoid;* $\langle \mathbb{K}, \otimes, \overline{1}\rangle$ *is a monoid;* $\otimes$ *distributes over* $\oplus$*:* $\forall a, b, c \in \mathbb{K} : (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ *and* $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$*;* $\overline{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K} : a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$*.*

Definition 4. *A* weighted finite automaton *over the semiring* $\mathbb{K}$ *is defined as the* 6*-tuple* $\mathcal{A} = \langle \Sigma, Q, E, q_0, \lambda, \rho\rangle$*, with* Σ *as its (finite)* alphabet*,* Q *as the set of* states*,* $E \subseteq Q \times (\Sigma \cup \epsilon) \times \mathbb{K} \times Q$ *as the finite set of* transitions*,* $q_0 \in Q$ *as the* initial state*,* $\lambda \in \mathbb{K}$ *as the* initial weight *and* $\rho : Q \to \mathbb{K}$ *as the* final weight *function. Given a transition* $e = \langle p, a, k, q\rangle$*, we denote the* source state $s[e] = p$*, the* target state $t[e] = q$*, the* weight $w[e] = k$ *and the* label $l[e] = a$*.*

A *path* $\pi = e_1 \ldots e_n$ in $\mathcal{A}$ is an element of E^* with adjacent transitions, i.e. $t[e_i] = s[e_{i+1}]$ for $1 \le i < n$. We define $s[\pi] = s[e_1]$, $t[\pi] = t[e_n]$ and $l[\pi] = l[e_1] \ldots l[e_n]$. The *weight of a path* $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_n]$ is then the product of the weights of its constituent transitions; $w[\epsilon] = \overline{1}$.

The output weight assigned to a string x by a weighted automaton $\mathcal{A}$ is defined as follows, where $\Pi(x) = \{\pi \in E^* : s[\pi] = q_0\}$, i.e. the set of paths starting in the initial state.

Definition 5. $[\![\mathcal{A}]\!](x) = \bigoplus_{\pi \in \Pi(x)} \lambda \otimes w[\pi] \otimes \rho(t[\pi])$

If the set of paths for x is empty, its weight is defined as $\overline{0}$. This generalises finite automata, since the unweighted case can be treated as weighted automata over the Boolean semiring $\mathbb{B} = \langle \{1, 0\}, \vee, \wedge, 0, 1\rangle$. The languages generated by finite automata are called *regular languages*.

We will use the *tropical semiring* $\langle \mathbb{R}^+ \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ as weighting structure in this paper, as we will represent probabilities by their negative logarithm. Note that numbers are multiplied by adding their logarithms, thus the multiplicative monoid of the tropical semiring is isomorphic to the probabilistic monoid $\langle [0, 1], \cdot, 1\rangle$. The min function always picks the weight of the best path.

2.1 Regular approximation

For a CFG to generate a non-regular language, it has to be *self-embedding* [3]:

Definition 6. *A CFG is self-embedding if there is $A \in N$ such that $A \Rightarrow^* \alpha A \beta$ with $\alpha \neq \epsilon \wedge \beta \neq \epsilon$.*

Grammars which are not self-embedding are called *strongly regular*. Their languages can be recognised by a finite automaton. An algorithm to convert a strongly regular grammar into a finite automaton is given by [3]. Self-embedding grammars can be transformed into strongly regular grammars approximating the language. Different kinds of approximation have been described in the literature. One of them is superset approximation, creating an automaton which accepts a language which is a superset of the original language. This means that every word in the original language is preserved which is a useful property, as will be seen later. Superset approximations are also called *sound* approximations. Among the superset algorithms are the ones by [3], [4] and [5]. We describe an algorithm by [6]. First, define a relation $\mathcal{R}$ on the set of nonterminals N by:

$$A \,\mathcal{R}\, B \Leftrightarrow (\exists\, \alpha, \beta \in V^* : A \Rightarrow^* \alpha B \beta) \wedge (\exists\, \alpha, \beta \in V^* : B \Rightarrow^* \alpha A \beta)$$

According to [6], this is an equivalence relation which partitions N into subsets of *mutually recursive nonterminals*. We now transform the grammar for each such set M. For each nonterminal $A \in M$ choose $A' \notin N$ and add a rule $A' \rightarrow \epsilon \,/\, 1$ and for each rule $A \rightarrow \alpha_0\, B_1\, \alpha_1\, B_2 \ldots B_n\, \alpha_n \;/\, w$ where $A, B_1, \ldots B_{n-1} \in M$ and $\alpha_1 \ldots \alpha_n \in (V \setminus M)^*$, replace it by the following set of rules:

$$\begin{aligned}
A &\rightarrow \alpha_0\, B_1 \;/\, w_0 \\
B_1' &\rightarrow \alpha_1\, B_2 \;/\, w_1 \\
&\;\vdots \\
B_{n-1}' &\rightarrow \alpha_{n-1} B_n \;/\, w_{n-1} \\
B_n' &\rightarrow \alpha_n\, A' \;/\, w_n
\end{aligned}$$

such that $w_0 \otimes w_1 \otimes \cdots \otimes w_{n-1} \otimes w_n = w$. The easiest choice to distribute the weight would be $w_0 = w$ and $w_1 = w_2 = \cdots = w_{n-1} = w_n = 1$, but machine learning probably allows for much better distributions.

Intuitively speaking, every nonterminal marked with $'$ signifies the end of a derivation of this nonterminal in the original grammar. It is easy to see that a word accepted by the original grammar is still accepted by the new grammar. Furthermore, the new grammar is not self-embedding anymore [6] and can thus be transformed into an equivalent finite automaton.

2.2 Improving regular approximation

Regular approximation often produces bad results. We propose a model which provides greater generative capacity than finite automata: bag-equipped finite automata. The motivation for bags comes from an observation: when a rule is decomposed using the algorithm by [6], nonterminals A and A' correspond to beginning and end of a constituent. However, a sequence need not be used completely; it can be terminated by a rule of the form $B' \to \epsilon$, where B was in the right-hand side of the original rule.

We try to rule out these candidates by introducing a bag (multiset). This bag M is a mapping $N \to \mathbb{N}$, where N is the set of nonterminals, for instance $M_1 = \{\langle \mathrm{S}, 1\rangle, \langle \mathrm{NP}, 2\rangle\}$. The bag may be generalised to allow for negative occurrences of elements: $M : N \to \mathbb{Z}$. Anytime a rule sequence is started, the corresponding nonterminal is pushed into the bag, and everytime a rule of the form $A' \to \epsilon$ is applied, A is popped from the bag. The automaton only accepts with an empty bag. Formally, a *bag-equipped finite automaton* is defined just like an ordinary finite automaton, where every transition carries a set of symbols from a bag alphabet, specifying either to push or pop them from the bag. Push and pop actions are taken along the paths; the automaton starts with an empty bag and can only accept with an empty bag.

Among the languages that can be recognised with bag-equipped finite automata are the (non-context-free) MIX language (any permutation of $a^n b^n c^n$) and $\{a^n b^n : n \in \mathbb{N}\}$ which is context-free. In fact, any number of agreement chains (even beyond context-free) can be recognised. To see why this is, consider the rational semiring $\langle \mathbb{Q}, +, \cdot, 0, 1\rangle$ and the set of prime numbers $P = \{2, 3, 5, 7, \dots\}$ which represent pushing to the bag and $P' = \{1/p : p \in P\}$ which represent popping from the bag. Accept a word only if the (deterministic) automaton assigns it the weight 1. We say, the automaton $\{1\}$-recognises a language L if it assigns weight 1 to each $w \in L$. A sample automaton $\{1\}$-recognising $\{a^n b^n c^n d^n\}$ (context-sensitive) is depicted in Fig. 1. For other constructions that show the power of bags and the corresponding semiring constructions see [7–9].

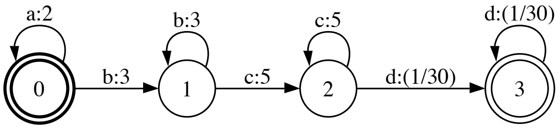

Fig. 1. Sample automaton $\{1\}$-recognising the context-sensitive language $a^n b^n c^n d^n$.

The linguistic justification is that this enables us to filter out some derivations where brackets are unmatched; i.e. where a full tree was not created. Of course, when brackets are matching but also crossing, bags will not help. This will happen when rules from other rule sequences are used in a derivation.[3]

3 A* parsing

The *A* algorithm*, also known as *stack decoder*, is widely used in AI. It relies on a heuristics to estimate the weight a partly processed candidate will have when it is completed. According to [10], the algorithm can be characterised as follows:

1. Initialise a priority queue with the empty hypothesis.
2. Execute the following steps until a complete hypothesis is found:
   (a) Pop the best hypothesis from the queue;
   (b) Expand the hypothesis yielding a list of candidates;
   (c) Evaluate every candidate and place them onto the queue accordingly.
3. For *n-best search*, go to 2 and proceed with the next-best hypothesis.

3.1 Probabilistic Chart parsing

Let $G = \langle \Sigma, N, P, S, w \rangle$ be a separated PCFG. Recall the following algorithm which is based on Earley's algorithm [11] to determine the minimal cost of a word x in G. Initialise two structures called the *chart* and the *agenda*. On the chart, we keep track of *passive edges* $X : [i, j]$ where the substring between position i and j of x represents the nonterminal X. On the agenda, we place *active edges* represented by *dotted rules* $X : \alpha \bullet \beta[i, j]$ for $X \to \alpha\beta \in P$ and $\alpha, \beta \in (\Sigma \cup N)^*$. Put $S' : \bullet S[0, 0]$ on the agenda and perform the following in a loop; if a passive edge $S' : [0, |x|]$ is found on the chart, accept x.

1. **predict:** For each edge on the agenda of the form $A : \alpha \bullet B\beta[i, j]$ and each rule $B \to \gamma$ put a new edge $B : \bullet\gamma[j, j]$ on the agenda.
2. **scan:** For each edge on the agenda of the form $A : \bullet x_j[i, j]$ put a new edge $A : x_j \bullet [i, j + 1]$ on the agenda (x_j being the j-th symbol of the word, counting from 0). Move passive edges to the chart.
3. **complete:** For each edge on the chart of the form $A : \alpha \bullet [j, k]$ and each edge on the agenda of the form $B : \alpha \bullet A\beta[i, j]$ put a new edge $B : \alpha A \bullet \beta[i, k]$ on the agenda. Move passive edges to the chart.

In order to find the Viterbi parse, costs have to be associated to the edges. We represent costs as the negative logarithm of the probability; lower scores represent higher probabilities ($-\log 1 = 0$ and $-\log 0 := \infty$). Addition of costs

[3] A more general model is the *deterministic pushdown automaton* (DPDA). Its generative capacity is equivalent to the deterministic context-free languages (DCFL), a proper subset of the CFLs but a proper superset of the regular languages [2]. Parsing time is linear. Unfortunately, the authors know of no algorithm which approximates a CFG. Also, DPDA's equipped with stacks have been suggested by Søgaard [9].

corresponds to multiplication of probabilities. In the **complete** step costs have to be added, and when moving edges to the chart, only the least cost is kept. This is an approximation $b(e)$ of what is called *Viterbi inside score* of an edge e and written $\beta(e)$. Over time, $b(e)$ will eventually equal $\beta(e)$. Analogously, the *Viterbi outside score* $\alpha(e)$ is defined as the least cost to complete the parse with edge e. The *Viterbi score* is the product of inside and outside score.

However, an edge can only be removed from the agenda if all its subedges have been removed. This is not trivial and crucial for an asymptotical time bound of $O(n^3)$ because the Viterbi inside score must be final when greater spans are built in order to avoid backtracking. Klein and Manning [12, 13] describe an algorithm to parse any PCFG and to find the Viterbi parse in $O(n^3)$. They make use of two ideas: recognising probabilistic parsing as an instance of shortest-path search in a hypergraph and representing the dotted rules in a finite automaton in order to determine the correct sequence in which the edges should be processed. A* parsing tries to minimise the effort by ranking the active edges on the agenda. In order to find a ranking, an evaluation function is introduced:

$$f^*(e) = b(e) + a(e).$$

Here, e is an active edge, and $b(e)$ the best inside score currently known. $a(e)$ is a heuristics which tries to estimate the least cost with which e can be incorporated into a successful parse. This function must comply with three requirements:

1. **Admissibility:** It must not underestimate the actual probability (overestimate the cost). Otherwise, the Viterbi parse might not be found, since the estimate could be inferior to the final cost of some other parse.
2. **Monotony:** Probability must never increase (costs must never decrease) while a path is completed. If this is not fulfilled, again the Viterbi parse could at some point be inferior to the final cost of some other parse.
3. **Efficiency:** It should not overestimate too much since overestimating favors short paths which in the worst case means exhaustive search.

With an estimate like this at hand, it is possible to sort the active edges on a priority queue and perform the expansion and evaluation of candidates (the core loop of the chart parser) only for the first active edge on the agenda.

3.2 Admissible and monotonic estimates

Klein and Manning [1] provide two valid estimates: *context summaries* and *grammar transformation.* Context summaries, based on work by [14], precompute the best Viterbi scores for a finite number of contexts and then find the one that matches. Their other idea is to exhaustively parse with a smaller grammar, as parsing time depends on the grammar size. To this end nonterminals and corresponding rules are collapsed with the minimal cost. Both of these can be parametrised such that they fall between two extremes: the NULL heuristics which represents no information at all and gives everything weight 1 (or log-probability 0), and the TRUE heuristics which provides the actual Viterbi outside score.

What we will present in the next section is a sort of grammar transformation. While it cannot beat context summaries in terms of asymptotical run time, it is asymptotically faster than the exhaustive parse in a transformed grammar which is still $O(n^3)$, but with a much lower factor due to smaller grammar size.

3.3 Regular approximation estimates

We will now turn to our contribution to this subject, i.e. describe how regular approximations can serve as A* heuristics. Not only does the approximation outlined in sec. 2.1 preserve every word in the original grammar, it also preserves every derivation with the corresponding weight [6]. This also holds true if we add a bag. Of course, the resulting grammar is not probabilistic, but this is of no concern. The probability for a word either stays the same or increases (if the new grammar allows for a better derivation). Thus, superset regular approximation is an admissible estimate.

In order to use an approximation of the original grammar represented by a weighted finite automaton $\mathcal{A}$ as a heuristics, a way has to be found to estimate the Viterbi outside score. Given an edge $e = X : [i, j]$, the probability of $\gamma X \zeta$ has to be calculated, where $\gamma = c_1 c_2 \dots c_{i-1}$ and $\zeta = z_{j+1} z_{j+2} \dots z_n$ represent the input string up to the i-th and from the j-th symbol on, which is the left and right context respectively. The easiest way is to add new terminals to the grammar: t_X for every $X \in N$. If we then add a rule for every nonterminal $X \rightarrow t_X X' \,/\, 1$, the automaton resulting from the approximation can skip t_X with weight 1, as there is another rule $X' \rightarrow \epsilon \,/\, 1$ in the ruleset. We now have to let $\mathcal{A}$ recognise $\gamma\, t_X\, \zeta$ which can be done in linear time [7] to get an estimate for $\alpha(e)$. This leads to the following estimate function:

$$f^*(e) = b(e) + [\![\mathcal{A}]\!](\gamma\, t_X\, \zeta),$$

where $b(e)$ is as usual the inside Viterbi score. It might be noteworthy that recognition is linear in time even for non-deterministic finite automata. However, it is not guaranteed that determinisation is possible since a weighted finite automaton can only be determinised if it has the *twins property* [15].

4 Conclusion

It is hard to say how well the estimate we presented will perform. The only prediction that can be made is in terms of the worst case asymptotical runtime which is obviously not very relevant for a heuristics for two reasons: first, asymptotical runtime says very little about practical use as there could be large constant factors; second, if the heuristics is bad, it does not save time because the search degenerates to an exhaustive search. Therefore, we are actively looking for people to join us in implementing the ideas proposed in this paper so that they can be tested against the ideas of [1] and a baseline model.

In addition, different approaches to regular approximation could be explored. There is a large number of algorithms available which satisfy the conditions on estimates (for a discussion see [3]). Furthermore, some aspects of languages beyond regularity can be captured by weighted finite automata [7], so it is conceivable that the automata could be fine-tuned to prefer structures from the old grammar by assigning them better weights. We also mentioned the idea of achieving better approximation by bag-equipped finite automata and DPDA's. Another interesting question is how well enhanced approximation algorithms work for arbitrary grammars. The authors are planning a detailed survey à la [3].

References

1. Klein, D., Manning, C.D.: A* parsing: Fast exact viterbi parse selection. In: Proceedings of the NAACL. (2003)
2. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages and computation. Addison-Wesley (1979)
3. Nederhof, M.J.: Practical experiments with regular approximation of context-free languages. In: Computational Linguistics, MIT Press for the ACL. Volume 26. (2000)
4. Pereira, F.C.N., Wright, R.N.: Finite-state approximation of phrase structure grammars. In: Meeting of the ACL. (1991) 246–255
5. Grimley-Evans, E.: Approximating context-free grammars with a finite-state calculus. CoRR **cmp-lg/9711002** (1997)
6. Mohri, M., Nederhof, M.J.: Regular approximation of context-free grammars through transformation. In: Robustness in Language and Speech Technology. Kluwer Academic Publishers (2000)
7. Cortes, C., Mohri, M.: Context-free recognition with weighted automata. Grammars **3**(2/3) (2000) 133–150
8. Søgaard, A.: On the weak generative capacity of weighted context-free grammars. In: Coling 2008: Companion volume: Posters, Manchester, UK, Coling 2008 Organizing Committee (August 2008) 99–102
9. Søgaard, A.: A linear time extension of deterministic pushdown automata. In: NODALIDA 2009. Volume 4 of NEALT Proc., Tartu (2009) 182–189
10. Paul, D.B.: Algorithms for an optimal A* search and linearizing the search in the stack decoder. In: HLT '90: Proceedings of the workshop on Speech and Natural Language, Morristown, NJ, USA, ACL (1990) 200–203
11. Earley, J.: An efficient context-free parsing algorithm. Commun. ACM **13**(2) (1970) 94–102
12. Klein, D., Manning, C.D.: Parsing and hypergraphs. In: New developments in parsing technology. Kluwer Academic Publishers, Norwell, MA, USA (2002) 351–372
13. Klein, D., Manning, C.D.: An $O(n^3)$ agenda-based chart parser for arbitrary probabilistic context-free grammars. Technical Report dbpubs/2001-16, Stanford University (2001)
14. Corazza, A., Mori, R.D., Gretter, R., Satta, G.: Optimal probabilistic evaluation functions for search controlled by stochastic context-free grammars. IEEE Trans. Pattern Anal. Mach. Intell. **16**(10) (1994) 1018–1027
15. Mohri, M.: Finite-state transducers in language and speech processing. Computational Linguistics **23**(2) (1997) 269–311