Parsing

A* Parsing

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Winter 2017/18

Table of contents

Introduction (1)

Idea of weighted deductive parsing (Nederhof, 2003):

- Give a deductive definition of the probability of a parse tree.
- Use Knuth's algorithm to compute the best parse tree for category S and a given input w.

Idea of A* parsing (Klein & Manning, 2003): Incorporate an estimate of the outside viterbi score of the parse items into the weights in order to reduce the number of generated items.

Advantage:

- Yields the best parse without exhaustive parsing.
- Weights are more precise than only inside viterbi scores, therefore less items are produced.

Introduction (2)

Extension of a deductive parsing system to a weighted deduction system:

- Each item has an additional weight. Intuition: weight = costs to build an item.
- The deduction rules specify how to compute the weight of the consequent item form the weights of the antecedent items.

Example (CYK) with probability of best parse tree (inside viterbi score) for every item:

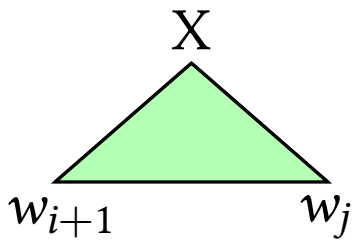
Scan: $\dfrac{}{|log(p)| : [A, i-1, i]}\ p : A \to w_i$

Complete: $\dfrac{x_1 : [B, i, j], x_2 : [C, j, k]}{x_1 + x_2 + |log(p)| : [A, i, k]}\ p : A \to B\,C$

A* parsing (1)

Shortcoming:

- The weight of an item $[X, i, j]$ is based only on the probability of its parse tree, i.e., on its inside viterbi score.

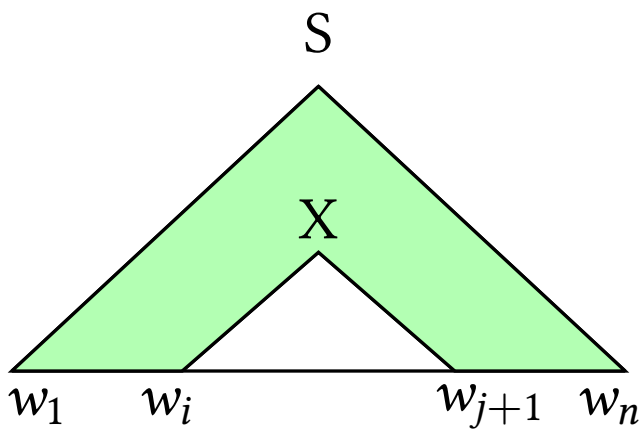


- Shorter derivations are often preferred (even if they do not lead to the best parse).
- Sometimes the gain arising from the reduction of the item number is less than the costs of the management of the priority queue.

Solution: Incorporate an estimate of the outside viterbi score in the weights (A* parsing, Klein & Manning, 2003).

A* parsing (2)

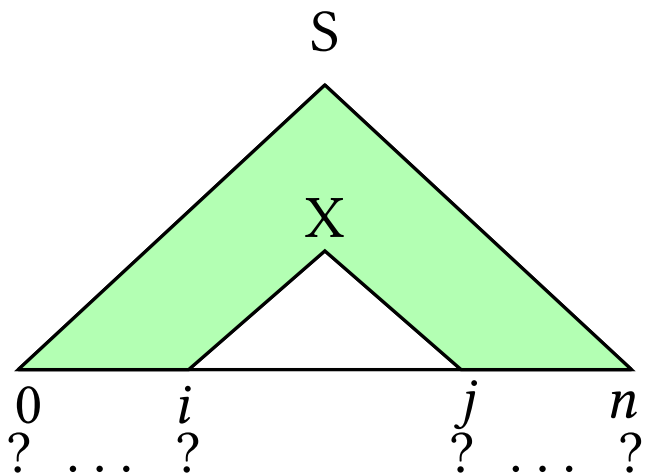
Outside viterbi score of $[X, i, j]$: maximal probability of parse tree with root S and leaves $w_1 \cdots w_i X w_{j+1} \cdots w_n$:



A* parsing (3)

Different context summary estimates for $[X, i, j]$:

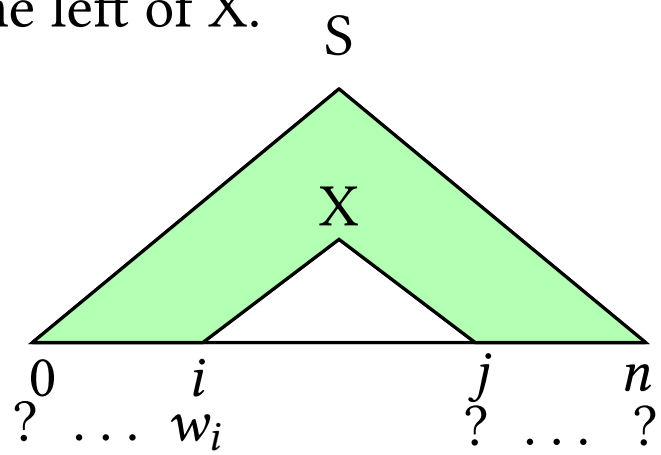
- SX: maximal probability of a parse tree with root S, a leaf X and i terminal leaves to the left and $n - j$ terminal leaves to the right of X.



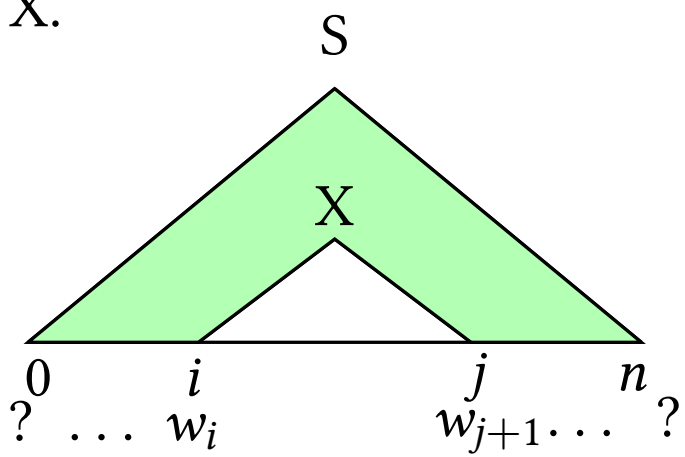
A* parsing (4)

More lexicalized estimates:

- SXL: extends SX, estimate depends also on first terminal (often first POS tag) to the left of X.



- SXLR: extends SXL, estimate depends also on first terminal/POS tag to the right of X.



A* parsing (5)

Priority (= weight) of an item: inside weight + estimated outside weight.

Conditions that ensure that the best parse gets found:

- The estimate must be admissible, i.e., it must not underestimate the actual probability required to complete the parse.
- It must be monotonic, i.e., when applying a rule, the priority never decreases.

Savings in the number of items for different estimates (Klein & Manning, 2003):

NULL	11.2%
SX	80.3%
SXL	83.5%

Computation of the SX estimates (1)

The SX estimates are precompiled, up to a certain maximal sentence length n_{max}:

For every non-terminal A, every possible sentence length $n \leq n_{max}$, every possible number n_l of terminals to the left of the span of A and every possible number n_r of terminals to the right of the span of A, and for $l = n - (n_l + n_r)$ we precompute

$$outside(A, n_l, l, n_r)$$

In this case, the span of A starts with the $(n_l + 1)$th input symbol and ends with the $n - n_r$th input symbol.

Note that $outside(A, n_l, l, n_r)$ is not an estimate of the outside probability of an A with span from w_{n_l+1} to w_{n-n_r} (this would require a sum over all possible parse trees) but it is an estimate of the probability of the best outside parse tree.

Computation of the SX estimates (2)

We need an estimate of the inside score *in* of an A with a span of length l. $in(A, l)$ is then $|log(p)|$ where p is the maximal probability of a parse tree with root label A and span length l.

Computation of $in(A, l)$ for all n, $1 \leq n \leq n_{max}$:

Computation of inside score

For all $A \in N, l \leq n$: $in(A, l) = \infty$.
  For all l, $1 \leq l \leq n$ and all non-terminals A:
  If $l == 1$ and p max. prob. with $p : A \rightarrow a \in P$
    for some $a \in T$, then $in(A, l) = |log(p)|$.
  Else:
    For all l_1, $1 \leq l_1 \leq l - 1$, and all $A \rightarrow BC$:
      $new = |log(p(A \rightarrow BC))| + in(B, l_1) + in(C, l - l_1)$.
      If $new < in(A, l)$, then $in(A, l) = new$.

Computation of the SX estimates (2)

The SX outside estimate depends on the category A, the span length l and the number of terminals n_l, n_r to the left and right of A respectively.

Computation of $out(A, n_l, l, n_r)$ for all n with $1 \leq n \leq n_{max}$:

- We start with the maximal length. The outside score of an S with span length n is 0 (probability 1). Other non-terminals with span length n have score ∞ (probability 0).
- Non-terminals A with smaller spans are estimated via their possible use as a right-hand side element in a production $B \rightarrow AC$ (combination with a sister to the right) or $B \rightarrow CA$ (combination with a sister to the left). In this case, the outside estimate of the mother, the probability of the production and the insde estimate of the sister give a possible value for the outside score of A.

Computation of the SX estimates (3)

Computation of outside score

For all l, $n \geq l \geq 1$ (start with n) and for all n_l, n_r with $n = n_l + l + n_r$ and all non-terminals A:

$out(A, n_l, l, n_r) = \infty$.

If $n_l == n_r == 0$ and $A == S$, then $out(A, n_l, l, n_r) = 0$.

Else:

For all l_C, $1 \leq l_C \leq n_r$, and all $B \to AC$:

$new = |log(p(B \to AC))| + out(B, n_l, l + l_C, n_r - l_C) + in(C, l_C)$.

$out(A, n_l, l, n_r) = \min(new, out(A, n_l, l, n_r))$.

For all l_C, $1 \leq l_C \leq n_l$, and all $B \to CA$:

$new = |log(p(B \to CA))| + out(B, n_l - l_C, l + l_C, n_r) + in(C, l_C)$.

$out(A, n_l, l, n_r) = \min(new, out(A, n_l, l, n_r))$.

Computation of the SX estimates (4)

Algorithm similar to the computation of inside and outside probability (charts β and α), except that here we search for the best probability and not the sum of all probabilities.

Example

Consider the PCFG $G = \langle \{N, A\}, \{camping, car, nice, red, ugly, green, house, bike\}, P, N \rangle$ with productions:

$0.1 : N \rightarrow NN$	$0.2 : N \rightarrow AN$
$0.1 : N \rightarrow red$	$0.1 : N \rightarrow green$
$0.1 : N \rightarrow car$	$0.1 : N \rightarrow bike$
$0.2 : N \rightarrow camping$	$0.1 : N \rightarrow house$
$0.3 : A \rightarrow nice$	$0.25 : A \rightarrow ugly$
$0.2 : A \rightarrow red$	$0.25 : A \rightarrow green$

Computation of the SX estimates (5)

Example continued

$0.1(1) : N \to NN$ $0.2(0.7) : N \to AN$ $0.1(1) : N \to red$ $0.1(1) : N \to green$
$0.1(1) : N \to car$ $0.1(1) : N \to bike$ $0.2(0.7) : N \to camping$ $0.1(1) : N \to house$
$0.3(0.5) : A \to nice$ $0.25(0.6) : A \to ugly$ $0.2(0.7) : A \to red$ $0.25(0.6) : A \to green$

Estimates of inside viterbi scores (up to length 4):

A	0.5	∞	∞	∞	
N	0.7	1.9	3.1	4.3	
	1	2	3	4	l

$in(A,1) = min\{0.5, 0.6, 0.7\}$, $in(N,1) = min\{1, 0.7\}$
$in(A,2) = \infty$, $in(N,2) = min\{1 + 0.7 + 0.7, 0.7 + 0.5 + 0.7\}$
$in(A,3) = \infty$, $in(N,3) = min\{1 + 0.7 + 1.9, 0.7 + 0.5 + 1.9\}$
$in(A,4) = \infty$, $in(N,4) = min\{1+0.7+3.1, 1+1.9+1.9, 0.7+0.5+3.1\}$

Computation of the SX estimates (6)

Example continued

$0.1(1) : N \to NN \quad 0.2(0.7) : N \to AN$

inside scores:

A	0.5	∞	∞	∞
N	0.7	1.9	3.1	4.3
	1	2	3	4

Outside estimates (for $n = 4$, the others are omitted):

$l = 4$: $out(A, 0, 4, 0) = \infty$, $out(N, 0, 4, 0) = 0$

$l = 3$: $out(A, 0, 3, 1) = min\{0.7 + 0.7 + 0\} = 1.4$, $out(A, 1, 3, 0) = \infty$
$out(N, 0, 3, 1) = min\{1 + 0.7 + 0\} = 1.7$,
$out(N, 1, 3, 0) = min\{1 + 0.7 + 0, 0.7 + 0, 5 + 0\} = 1.2$

$l = 2$: $out(A, 2, 2, 0) = \infty$, $out(A, 1, 2, 1) = min\{0.7 + 0.7 + 1.2\} = 2.6$,
$out(A, 0, 2, 2) = min\{0.7 + 0.7 + 1.7, 0.7 + 1.9 + 0\} = 2.6$
$out(N, 0, 2, 2) = min\{1 + 0.7 + 1.7, 1 + 1.9 + 0\} = 2.9$,
$out(N, 1, 2, 1) = min\{0.7 + 0.5 + 1.7, 1 + 0.7 + 1.7, 1 + 0.7 + 1.2\} = 2.9$,
$out(N, 2, 2, 0) = min\{1 + 0.7 + 1.2, 1 + 1.9 + 0\} = 2.9$

Computation of the SX estimates (7)

Example continued

$0.1(1) : N \to NN \quad 0.2(0.7) : N \to AN$

inside scores:

A	0.5	∞	∞	∞
N	0.7	1.9	3.1	4.3
	1	2	3	4

$l = 1$: $out(A, 3, 1, 0) = \infty$,

$out(A, 2, 1, 1) = min\{0.7 + 0.7 + 2.9\} = 4.3$,

$out(A, 1, 1, 2) = min\{0.7 + 0.7 + 2.9, 0.7 + 1.9 + 1.2\} = 3.8$,

$out(A, 0, 1, 3) = min\{0.7+0.7+2.9, 0.7+1.9+1.7, 0.7+3.1+0\} = 3.8$

$out(N, 3, 1, 0) = min\{1 + 0.7 + 2.9, 1 + 1.9 + 1.2, 1 + 3.1 + 0, 0.7 + 0.5 + 2.9\} = 4.1$,

$out(N, 2, 1, 1) = min\{1 + 0.7 + 2.9, 1 + 1.9 + 1.7, 1 + 0.7 + 2.9, 0.7 + 0.5 + 2.9\} = 4.1$,

$out(N, 1, 1, 2) = min\{1 + 0.7 + 2.9, 1 + 0.7 + 2.9, 1 + 1.9 + 1.2, 0.7 + 0.5 + 2.6\} = 3.8$,

$out(N, 0, 1, 3) = min\{1 + 0.7 + 2.9, 1 + 1.9 + 1.7, 0.7 + 3.1 + 0\} = 3.8$

Parsing with the SX estimate (1)

We incorporate the SX estimate into the weights of our deduction rules (n is the sentence length):

Scan: $$\frac{}{|log(p)| + out(A, i-1, 1, n-i) : [A, i-1, i]} \; p : A \to w_i$$

Complete:
$$\frac{x_1 + out(B, i, j-i, n-j) : [B, i, j], \; x_2 + out(C, j, k-j, n-k) : [C, j, k]}{x_1 + x_2 + |log(p)| + out(A, i, k-i, n-k) : [A, i, k]}$$
where $p : A \to B\ C \in P$

Summary of A* parsing (1)

Modular approach:

1. Specify a weighted deductive system.
2. Use Knuth's algorithm to compute the best goal item.
3. Incorporate not only the inside probability but also an estimate of the outside probability in the weights.

Summary of A* parsing (2)

Advantages:

- Guarantee to find the best parse (in contrast to, e.g., beam search methods).
- The combination of inside weight and estimated outside weight helps to reduce the number of items by over 80 %, compared to exhaustive parsing.
- Approach applies to any system that can be characterized with an appropriate weighted deductive system.

k-best parsing (1)

Extension to k-best parsing (Pauls & Klein, 2009).

Problem with k-best parsing:

- We can no longer abstract away from the concrete parse trees, i.e., use only items $[A, i, j]$.
- Instead, one has to keep track of actual parse trees with their weight, i.e., we need items $[T_A, i, j]$ where T_A is a parse tree yielding the substring of the input between positions i and j.
- There are exponentially many, we need to find a way to efficiently restrict the search space.

Pauls & Klein (2009) extend A* parsing to k-best.

k-best parsing (2)

- The ouside estimate is computed as in A* parsing, yielding values $out(A, n_l, l, n_r)$ as above.
- For the inside viterbi score (best parse tree), we also use A*: (We write all weights as pairs $\langle i, o \rangle$ where the first is the inside component, the second the outside component and the relevant agenda weight is their sum.)

  Scan:

$$\frac{}{\langle |log(p)|, out(A, i-1, 1, n-i) \rangle : I[A, i-1, i]} \; p : A \to w_i$$

  Complete:

$$\frac{\langle x_1, out(B, i, j-i, n-j) \rangle : I[B, i, j], \langle x_2, out(C, j, k-j, n-k) \rangle : I[C, j, k]}{\langle x_1 + x_2 + |log(p)|, out(A, i, k-i, n-k) \rangle : I[A, i, k]}$$

  where $p : A \to B\ C \in P$

k-best parsing (3)

- In addition, we do a delayed computation of the outside viterbi score (weight of items $O[A, i, j]$ and of the score of actual parse trees (weight of $[T_A, i, j]$).
- All these items are handled in a single priority agenda, i.e., in every step, the best item (lowest weight) is popped from the agenda and we produce new items with this one and items from the chart.

k-best parsing (4)

Computation of the outside viterbi score (best parse tree for completing an A spanning the input from i to j to an S spanning the entire input):

$$\frac{\langle x_1, 0\rangle : I[S, 0, n]}{\langle x_1, 0\rangle : O[S, 0, n]}$$

Out-L:

$$\frac{\langle i_1, x_1\rangle : O[A, i, j], \langle x_2, o_2\rangle : I[B, i, k], \langle x_3, o_3\rangle : I[C, k, j]}{\langle x_2, x_1 + x_3 + |log(p)|\rangle : O[B, i, k]}\ p : A \to B\,C \in P$$

Out-R:

$$\frac{\langle i_1, x_1\rangle : O[A, i, j], \langle x_2, o_2\rangle : I[B, i, k], \langle x_3, o_3\rangle : I[C, k, j]}{\langle x_3, x_1 + x_2 + |log(p)|\rangle : O[C, k, j]}\ p : A \to B\,C \in P$$

k-best parsing (5)

Computation of the scores of actual parse trees:

$$\frac{}{\langle |log(p)|, out(A, i-1, 1, n-i)\rangle : [A(w_i), i-1, i]}\ p : A \to w_i$$

$$\frac{\langle i_1, x_1\rangle : O[A, i, j], \langle x_2, o_2\rangle : [T_B, i, k], \langle x_3, o_3\rangle : [T_C, k, j]}{\langle x_2 + x_3 + |log(p)|, x_1\rangle : [A(T_B, T_C), i, j]}\ p : A \to B\,C \in P$$

k-best parsing (6)

- Everything is done in a single agenda.
- Start: Items $I[A, i-1, i]$ and $[A(w_i), i-1, i]$ are put into the agenda.
- Then the inside items $I[A, i, j]$ are computed.
- Once $I[S, 0, n]$ pops from the agenda, we start computing outside items $O[A, i, j]$.
- When the $O[A, i-1, i]$ items for the non-terminals yielding single terminals pop, we start computing parse trees.
- These steps can interleave.

Satisfies the monotonicity requirements of weighted deductive parsing. Consequently, the first k parse trees popped from the agenda are the k best parse trees, i.e., the algorithm can terminate after having popped the first k derivation trees with root S that yield the entire input, $[T_S, 0, n]$.

References

- Dan Klein & Christopher D. Manning (2003). A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of HLT-NAACL 2003 Main Papers*, pp. 40–47. Edmonton, May-June 2003.
- Mark-Jan Nederhof (2003). Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics* 29(1), pp. 135–143 (2003).
- Adam Pauls & Dan Klein (2009). K-Best A* Parsing. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pp. 958–966. Singapore, August 2009.