Earley intersection

Wilker Aziz

May 4, 2017

Earley parsers (Earley, 1970) can deal with arbitrary context-free grammars (CFGs), that is, they impose no constraints on:

- left-recursion;
- length of RHS sequence;
- epsilon rulesl

We will be working with a generalised view of parsing corresponding to intersection between discrete (potentially infinite) sets Bar-Hillel et al. (1961); Billot and Lang (1989). One set is finite-state, another set is context-free, and the result (the intersection) is also context-free. To draw a parallel with standard parsing, the FSA corresponds to the input sentence, the CFG corresponds to the grammar, the intersection corresponds to a parse forest (or chart).

This generalisation is interesting because when dealing with probability distributions over complex discrete spaces (such as those in parsing and hierarchical MT), it is convenient to be able to parse finite-state automata that can represent more than single sentences.

1 FSA

An FSA is a 5-tuple $A = \langle \Sigma, Q, E, I, F \rangle$ where

- Σ is a finite set of labels (e.g. words);
- Q is a finite set of states;
- E is a finite set of arcs;
- I is a finite set of initial states;
- F is a finite set of final/accepting states.

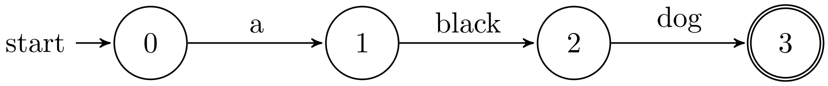


Figure 1: A *linear-chain* FSA is convenient for representing single sentences. Accepting states are denoted by double circles. In this case $L(A) = \{$"a black dog"$\}$.

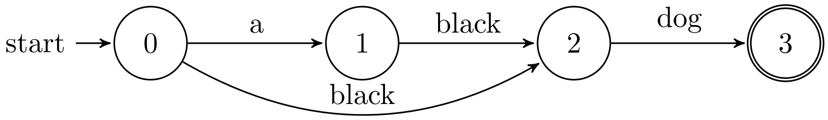


Figure 2: An FSA can define larger sets, for example, in this case $L(A) = \{$"a black dog", "black dog"$\}$.

An arc is a tuple $\langle q, x, s\rangle$ with $(q, s) \in Q \times Q$ and $x \in \Sigma$. An FSA defines a *language* $L(A)$ which correspond to the set of sentences that is accepted by A. A sentence $\mathbf{x}$ is accepted if there is a path from a state in I (an initial state) to a state in F (an accepting state) whose label sequence corresponds to $\mathbf{x}$. See Figures 1, 2 and 3 for examples.

2 CFG

A CFG is a 4-tuple $G = \langle\Sigma, N, S, R\rangle$ where

- Σ is a finite set of terminal symbols (e.g. words);
- N is a finite set of nonterminal symbols (e.g. variables);
- $S \in N$ is a distinguished *start* symbol;
- R is a finite set of context-free rules/productions.

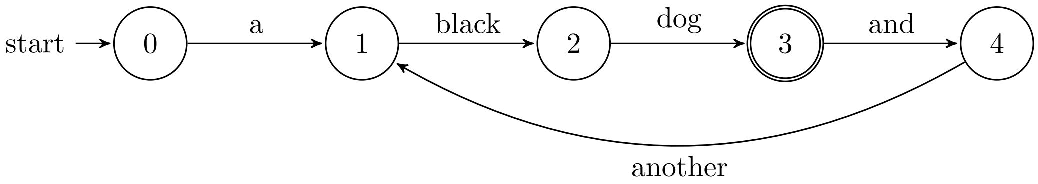
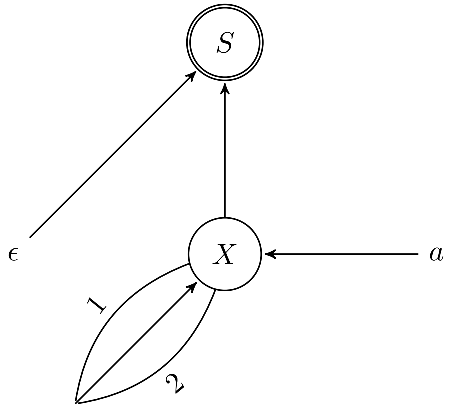


Figure 3: An FSA can also definite infinitely large sets, for example, in this case $L(A)$ contains "a black dog" and infinitely many sentences made of "a black dog" concatenated with an arbitrary number of copies of "and another black dog".

$$\begin{aligned} S &\to \epsilon \\ S &\to X \\ X &\to X\ X \\ X &\to a \end{aligned}$$

(a) Set of rules of G.



(b) A graphical representation (a cyclic hypergraph) for the simple CFG in 4. The start symbol is denoted with double circles.

Figure 4: A simple CFG: terminals are represented by lowercase letters. $L(G)$ contains the empty sentence and sentences made of arbitrarily many copies of "a".

A rule has the form $X \to \alpha$ where $X \in N$ (the rule's *left-hand side*—LHS) and $\alpha \in (\Sigma \cup N)^*$ (the rule's *right-hand side*—RHS). In other words, the rule's LHS is a single variable and the rule's RHS is a (possibly empty) sequence of terminal and nonterminal symbols.[1]

3 Earley intersection

Our version of Earley intersection (Dyer et al., 2008; Dyer, 2010) will take an ϵ-free FSA $A = \langle \Sigma, Q, E, I, F \rangle$ and an arbitrary CFG $G = \langle \Sigma, N, S, R \rangle$ and compute another CFG $G' = \langle \Sigma, N', S', R' \rangle$ such that $L(G') = L(A) \cap L(G)$.

In this presentation we will make use of deductive systems, thus we start by defining our item form

$$[q, X \to \alpha_{\blacksquare} \bullet \beta_{\square}, r] \tag{1}$$

where

- q and r are states in Q: together they state that this item concerns all paths in A from q to r;
- $X \to \alpha\ \beta$ is a rule in R: note that α and β are possibly empty sequences of terminals and nonterminals;

[1] An empty RHS is denoted by an empty string (ϵ), for example, $X \to \epsilon$, we also call such rules *epsilon rules*.

- $\bullet$ represents how far down the rule's RHS the algorithm has progressed, in particular, it states that the first part of the rule (α) has already been intersected with FSA states, and that the second part (β) is yet to be dealt with;
- $\blacksquare$ is a shorthand for the FSA states that have already been intersected (in dealing with the symbols in α);
- $\square$ is a there just to remind us that we still need to intersect the symbols in β with FSA states.

For a concrete example, consider the FSA in Figure 1 and the following fragment of a grammar

$$\begin{array}{lcl} S & \rightarrow & NP \\ S & \rightarrow & NP\ VP \\ NP & \rightarrow & NN \\ NP & \rightarrow & DT\ NN \\ NP & \rightarrow & DT\ JJ\ NN \\ DT & \rightarrow & \text{the} \\ JJ & \rightarrow & \text{black} \\ NN & \rightarrow & \text{dog} \end{array}$$

then the item

$$[0, NP \rightarrow DT_{0,1}\ JJ_{1,2} \bullet NN, 2]$$

states that we have intersected paths from 0 to 2 in the automaton and this has allowed us to progress along the RHS of rule $NP \rightarrow DT\ JJ\ NN$ all the way to just before the last symbol (NN). We can interpret this as: every path from 0 to 2 in A corresponds to an NP missing an NN. Can you extrapolate this to the case where between 0 and 2 we have multiple paths?

NOTE ON TERMINOLOGY in parsing literature the subscripts in the symbols of the item correspond to *spans* and storing the in the item corresponds to keeping *back-pointers*.

Now we are ready to present a complete intersection algorithm

$$\text{Item form} \quad [q, X \rightarrow \alpha_{\blacksquare} \bullet \beta_{\square}, r] \tag{2}$$

$$\text{Axioms} \quad \frac{}{[q, S' \rightarrow \bullet S_{\square}]\,, q} \qquad q \in I \tag{3}$$

$$\text{Goal} \quad [q, S' \rightarrow S_{\blacksquare} \bullet, r] \qquad q \in I \text{ and } r \in F \tag{4}$$

$$\text{Predict} \quad \frac{[q, X \rightarrow \alpha_{\blacksquare} \bullet Y_{\square} \beta_{\square}, r]}{[r, Y \rightarrow \bullet \gamma_{\square}, r]} \qquad Y \rightarrow \gamma \in R \tag{5}$$

$$\Sigma\text{-Scan} \quad \frac{[q, X \rightarrow \alpha_{\blacksquare} \bullet x \beta_{\square}, r]}{[q, X \rightarrow \alpha_{\blacksquare} x \bullet \beta_{\square}, s]} \qquad x \neq \epsilon \text{ and } \langle r, x, s \rangle \in E \tag{6}$$

$$\epsilon\text{-Scan} \quad \frac{[q, X \rightarrow \alpha_{\blacksquare} \bullet \epsilon \beta_{\square}, r]}{[q, X \rightarrow \alpha_{\blacksquare} \epsilon \bullet \beta_{\square}, r]} \tag{7}$$

$$\text{Complete} \quad \frac{[q, X \rightarrow \alpha_{\blacksquare} \bullet Y_{\square} \beta_{\square}, r] \quad [r, Y \rightarrow \gamma_{\blacksquare} \bullet, s]}{[q, X \rightarrow \alpha_{\blacksquare} Y_{r,s} \bullet \beta_{\square}, s]} \tag{8}$$

Axioms state that we can intersect sentences generated by the start symbol of G as long as they project onto paths in A that start from one of A's initial states. We also introduce a new symbol S' which will be the start symbol of the intersection G'.

Goal is to have all derivations from S project paths from an initial state of A to a final state of A (this is basically the condition that a sentence in the intersection must be in the language of A and in the language of G).

Predict introduces new items in the program by predicting that we will be able to intersect a rule $Y \rightarrow \gamma$ with paths from $r \in Q$.

Scan deals separately with actual terminals and the empty string. The first variant simply moves the dot along the terminal if that terminal labels an FSA arc from r (the position of the dot in the antecedent) to s (the position of the dot in the consequent). The second variant, moves the dot without progressing in the FSA, that's because the empty string does not have to be matched against anything.[2]

Complete advances the dot by merging an incomplete antecedent with a complete one. The incomplete antecedent currently misses at least an Y projecting onto FSA paths from r, the complete antecedent offers a Y projecting onto FSA paths from r to s, thus in the consequent we can advance the dot over $Y_{\square}$ filling the empty square with the states r, s.

[2] In some presentations you will see a rule to scan ϵ arcs, here we are dealing with ϵ-free FSA and instead have ϵ in the grammar rules. This is sufficient for most uses including our project 2.

4 Implementation

We typically design items as immutable objects that hold a CFG rule and a sequence of integers representing intersected states. For example, $[0, NP \rightarrow DT_{0,1}\ JJ_{1,2} \bullet NN, 2]$ would be represented by Item(Rule($NP \rightarrow DT\ JJ\ NN$), (0, 1, 2)). We instantiate axioms and initialise a queue. Then, for as long as the queue is not empty we proceed by (1) popping an active item from the queue, (2) calling inference rules and gathering new items, (3) pushing new items into the queue, (4) storing the active item in a set of passive items. In pushing items back into the queue we make sure we do not push the same item more than once (to avoid spurious computations). At the end of this process, there will be no active items in the queue and our set of passive items will contain incomplete and complete items. The complete items can then be interpreted as our intersected grammar (or parse forest). At this point, we typically convert the items back to a CFG format. For example, the item Item(Rule($NP \rightarrow DT\ JJ\ NN$), (0, 1, 2, 3)) would motivate a CFG rule $NP_{0,3} \rightarrow DT_{0,1}\ JJ_{1,2}\ NN_{2,3}$. Note that in the intersection G' nonterminals will be decorated with a pair of FSA states. These states represent the boundaries of paths in A. For the traditional case where A contains a single sentence, these pairs of states correspond to spans (in parsing terminology).

Make sure to check our notebooks

https://github.com/uva-slpl/nlp2/tree/gh-pages/resources/notebooks

and if you need more on Earley intersection, you can check Section 5.1 of my thesis

http://rgcl.wlv.ac.uk/papers/aziz-thesis.pdf

which contains a lengthy discussion with examples and drawings.

References

Bar-Hillel, Y., Perles, M. A., and Shamir, E. (1961). On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, (14):143–172.

Billot, S. and Lang, B. (1989). The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Dyer, C. (2010). Two monolingual parses are better than one (synchronous parse). In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–266, Los Angeles, California. Association for Computational Linguistics.

Dyer, C., Muresan, S., and Resnik, P. (2008). Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio. Association for Computational Linguistics.

Earley, J. (1970). An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102.