An Earley Parsing Example

Shay Cohen

Inf2a

November 5, 2018

The sentence we try to parse:

“book that flight”

Whenever we denote a span of words by [i,j], it means it spans word i+1 through j, because i and j index, between 0 and 3, the *spaces* between the words:

0 book 1 that 2 flight 3

Grammar rules:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Prep NP
Verb → *book* | *include* | *prefer*
Noun → *book* | *flight* | *meal*
Det → *that* | *this* | *these*

Start with Prediction for the S node:

S → . NP VP [0,0]
S → . Aux NP VP [0,0]
S → . VP [0,0]

All of these elements are created because we just started parsing the sentence, and we expect an S to dominate the whole sentence

NP → . Pronoun [0,0]
NP → . Proper-Noun [0,0]
NP → . Det Nominal [0,0]
VP → . Verb [0,0]
VP -> . Verb NP [0,0]
VP → . Verb NP PP [0,0]
VP →. Verb PP [0,0]
VP → . VP PP [0,0]

Now we can apply PREDICTOR on the above S nodes! Note that PREDICTOR creates endpoints [i,j] such that i=j and i and j are the right-end points of the state from which the prediction was made

NOTE: For a PREDICTOR item, the dot is always in the beginning!

In the previous slide we had states of the following form:

VP -> . Verb NP [0,0]
VP → . Verb NP PP [0,0]
VP →. Verb PP [0,0]

Note that we now have a dot before a terminal.

We look at the right number of [i,j], and we see that it is 0, so we will try to match the first word in the sentence being a verb. This is the job of the Scanner operation.

CHECK! We have a rule Verb → book, so therefore, we can advance the dot for the above Verb rules and get the following new states:

VP -> Verb . NP [0,1]
VP → Verb . NP PP [0,1]
VP →Verb . PP [0,1]

Great. What does that mean now?

We can call PREDICTOR again, we have new nonterminals with a dot before them!

In the previous slide we had states of the following form:

VP -> Verb . NP [0,1]
VP → Verb . NP PP [0,1]
VP →Verb . PP [0,1]

We said we can now run PREDICTOR on them. What will this create?

For NP:

NP → . Pronoun [1,1]
NP → . Proper-Noun [1,1]
NP → . Det Nominal [1,1]

Note that now we are expecting a NP at position 1!

And also for PP:

PP → . Prep Nominal [1,1]

In the previous slide we created the following states:

NP → . Pronoun [1,1]
NP → . Proper-Noun [1,1]
NP → . Det Nominal [1,1]
PP → . Prep Nominal [1,1]

Now we have an opportunity to run SCANNER again on the second word in the sentence!
Question: for which item above would we do that?

We would do that for NP → . Det Nominal [1,1]. *"that"* can only be a Det. So now we create a new item:

NP → Det . Nominal [1,2]

Note that now [i,j] is such that it spans the second word (1 and 2 are the "indexed spaces" between the words before and after the second words)

In the previous slide, we added the state: NP → Det . Nominal [1,2]

Now PREDICTOR can kick in again, because Nominal is a nonterminal in a newly generated item in the chart.

What will PREDICTOR create? (Hint: PREDICTOR takes an item and adds new rules for all rules that have LHS like the nonterminal that appears after the dot.)

These are the new states that PREDICTOR will generate:

Nominal → . Noun [2,2]
Nominal → .Nominal Noun [2,2]
Nominal → .Nominal PP [2,2]

Note that again, predictor always starts with i=j for the [i,j] spans.

Its interpretation is: there might be a Nominal nonterminal spanning a substring in the sentence that starts with the third word.

In the previous slide, we created an element of the form:

Nominal → . Noun [2,2]

Its interpretation is: "there might be a use of the rule Nominal → Noun, starting at the third word. To see if you can use it, you need to first check whether there is a Noun at the third position in the sentence." We do! So SCANNER can kick in.

What will we get?

We now scanned the Noun, because we the word "flight" can be a Noun.

Nominal → Noun . [2,3]

That's nice, now we have a complete item. Can COMPLETER kick now into action?

We have to look for all items that we created so far that are expecting a Nominal starting at the second position.

In the previous slide, we created Nominal → Noun . [2,3] which is a complete item. Now we need to see whether we can apply completer on it.

Remember we created this previously?

NP → Det . Nominal [1,2]

Now we can apply COMPLETER on it in conjunction with Nominal → Noun . [2,3] and get:

NP → Det Nominal . [1,3]

Nice! This means we completed another item, and it means that we can create an NP that spans the second and the third word ("*that flight*") – that's indeed true if you take a look at the grammar.

In any case, now that we have completed an item, we need to see if we can complete other ones. The question we ask: is there any item that expects an NP (i.e. the dot appears before an NP) and the right-hand side of [i,j] is 1?

We actually had a couple of those:

VP → Verb . NP [0,1]
VP → Verb . NP PP [0,1]

They are waiting for an NP starting at the second word.

So we can use COMPLETER on them with the item NP → Det Nominal [1,3] that we created in the previous slide.

So now we will have new items:

VP → Verb NP . [0,3]
VP → Verb NP . PP [0,3]

The first one is also a complete one! So maybe we can apply COMPLETE again? We need an item that expects a VP at position 0.

Let's try to remember if we had one of those…

We had one indeed:

S → . VP [0,0]

That was one of the first few items we created, which is a good sign, it means we are creating now items that span the tree closer to the top node.

So now we can COMPLETE this node with the item VP → Verb NP . [0,3] that we created in the previous slide.

What do we get?

We get the item:

S → VP . [0, 3]

and that means we managed to create a full parse tree, we have an S that spans all words in the sentence.

How do we get a parse tree out of this? Back pointers…

Let's consider the "back-pointers" we created.

We created the node S → VP . [0, 3] as a result of a COMPLETER on the item VP → Verb NP . [0,3].

We created VP → Verb NP . [0,3] as a result of a COMPLETER on VP → Verb . NP [0,1] when we had an NP → Det Nominal [1,2].

That means the tree has to look like: