Shift-Reduce Phrase-Structure Parsing

Miguel Ballesteros

7-11

Overview

- Shift-reduce transition based process
- A Stack:
  - holds the partial parse trees already built
- A Buffer (or queue)
  - holds the incoming words with POS
- Actions
  - SHIFT, REDUCE-BINARY-L/R, REDUCE-UNARY

Overview

- The parsing process starts with all words in the buffer.
- In each step, there is a deterministic decision that decides which action to take given the parsing state.

- We use a classifier that decides what to do given the parsing state.
  - Parsing state: Stack and Buffer.

Example

S → NP VP

S → Aux NP VP

S → VP

NP → Det NOM

NOM → Noun

NOM → Noun NOM

VP → Verb

VP → Verb NP

Det → that | this | a | the

Noun → book | flight | meal |man

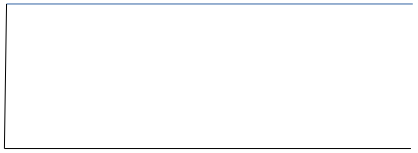
Verb → book | include | read

Aux → does

Example

- Book that flight.

Stack



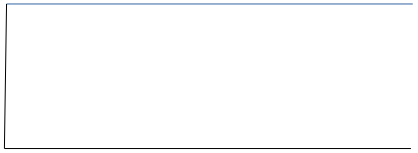
Buffer

Book that flight

Example

- Book that flight.

Stack



Buffer

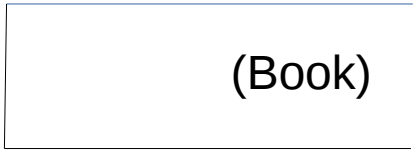
Book that flight

SHIFT

Example

- Book that flight.

Stack



Buffer

that flight

Example

- Book that flight.

Stack

(Book)

Buffer

that flight

REDUCE. Verb → book

Example

- Book that flight.

Stack

(Book)

Buffer

that flight

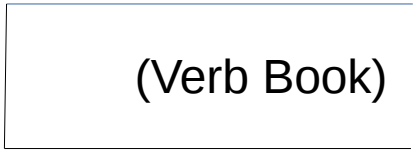
It could also be Noun → Book

REDUCE. Verb → book

Example

- Book that flight.

Stack



Buffer

that flight

Example

- Book that flight.

Stack

(Verb Book)

Buffer

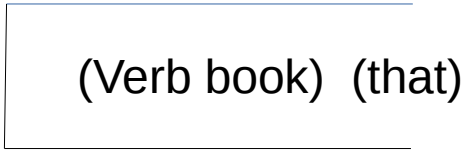
that flight

SHIFT

Example

- Book that flight.

Stack



Buffer

flight

Example

- Book that flight.

Stack

(Verb book) (that)

Buffer

flight

Reduce, Det → that

Example

- Book that flight.

Stack

(Verb book) (Det that)

Buffer

flight

Example

- Book that flight.

Stack

Buffer

flight

(Verb book) (Det that)

SHIFT

Example

- Book that flight.

Stack

(Verb book) (Det that) (flight)

Buffer

Example

- Book that flight.

Stack Buffer

(Verb book) (Det that) (flight)

Reduce. Noun → flight

Example

- Book that flight.

Stack

Buffer

(Verb book) (Det that) (Noun flight)

Example

- Book that flight.

Stack Buffer

(Verb book) (Det that) (Noun flight)

Reduce. NOM → Noun

Example

- Book that flight.

Stack Buffer

(Verb book) (Det that) (NOM (Noun flight))

Example

- Book that flight.

Stack Buffer

(Verb book) (Det that) (NOM (Noun flight))

Reduce. NP → Det NOM

Example

- Book that flight.

Stack Buffer

(Verb book) (NP (Det that) (NOM (Noun flight)))

Example

- Book that flight.

Stack Buffer

(Verb book) (NP (Det that) (NOM (Noun flight)))

Reduce. VP → Verb NP

Example

- Book that flight.

Stack Buffer

(VP (Verb book) (NP (Det that) (NOM (Noun flight))))

Example

- Book that flight.

Stack Buffer

(VP (Verb book) (NP (Det that) (NOM (Noun flight))))

Reduce. S → VP

Example

- Book that flight.

Stack Buffer

(S (VP (Verb book) (NP (Det that) (NOM (Noun flight)))))

Example

- Book that flight.

Stack Buffer

(S (VP (Verb book) (NP (Det that) (NOM (Noun flight)))))

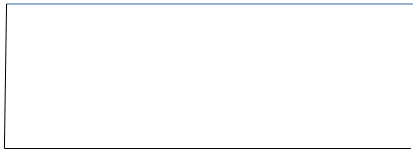
Is S in the stack? YES! Done.

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack



Buffer

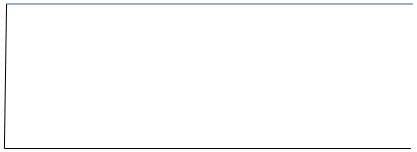
Flight a meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack



Buffer

Flight a meal

SHIFT

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(Flight)

Buffer

a meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(Flight)

Buffer

a meal

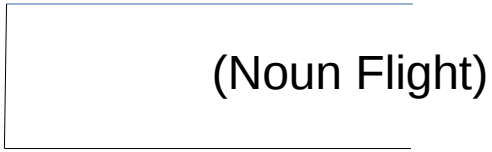
REDUCE. Noun → Flight

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack



Buffer

a meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(Noun Fligh)

Buffer

a meal

Reduce. NOM → Noun

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight))

Buffer

a meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight))

Buffer

a meal

SHIFT

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight)) (a)

Buffer

meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight)) (a)

Buffer

meal

Reduce. Det → a

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight)) (Det a)

Buffer

meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

(NOM (Noun Flight)) (Det a)

Buffer

meal

SHIFT

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (meal)

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (meal)

Reduce
Noun → Meal

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (Noun (meal))

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (Noun (meal))

Reduce
NOM → Noun

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (NOM (Noun (meal)))

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (Det a) (NOM (Noun (meal)))

Reduce
NP → Det NOM

Example

- Flight a meal (?)

Let's try with something “ungrammatical”

Stack

Buffer

(NOM (Noun Flight)) (NP(Det a) (NOM (Noun (meal))))

Now we have:

NP at the top of the stack.

NOM NP at the top of the stack

We are done, but S is not at the top of the stack.

We reject the input.

Shift-Reduce Parser

Start with the sentence to be parsed in an input buffer.

a ”shift” action correponds to pushing the next input symbol from the buffer onto the stack

a ”reduce” action occurs when we have a rule’s RHS on top of the stack. To perform the reduction, we pop the rule’s RHS off the stack and replace it with the terminal on the LHS of the corresponding rule.

- (When either ”shift” or ”reduce” is possible, choose one arbitrarily.)

If you end up with only the Start symbol on the stack, then success!

If you don’t, and you cannot and no ”shift” or ”reduce” actions are possible, backtrack.

Running time

- Linear in the length of the sentence. O(n).
- In terms of memory, it is also cheap.

Problems

- Unable to deal with empty categories: termination problem, unless rewriting empties as constituents is somehow restricted (but then it’s generally incomplete)

- Useless work: locally possible, but globally impossible.

- Repeated work: anywhere there is common substructure.

- Backtracking makes the parsing problem not linear anymore.

  However, in real parsing, this is not the case.