Syntax — II

Anton Alekseev
Steklov Mathematical Institute in St Petersburg

ITMO University, St Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Plan

1. What is parsing and why we need it
2. Phrase structure grammar
3. Dependency grammar
    a. What is DG
    b. Evaluation metrics
    c. Approaches to parsing
    d. Transition-based DP, intuitively
    e. Transition-based DP
    f. Tools and datasets

Dependency grammar

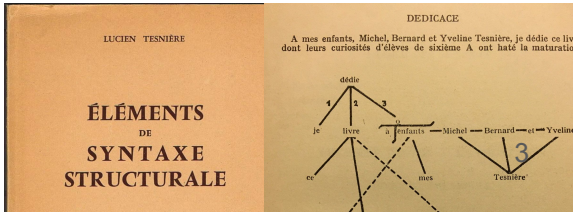
Keypoints

1. Sentence structure is words and relations (connections) between them
2. The relations between words are directed; one word is the head one, the other one is dependent
3. The connections form a tree, there is a path from the ‘root’ to any word

The main advantage: better suits the languages with the relatively “free word order”

...and in PSG one would have to set rules for every possible word/phrase position in a sentence

Lucien Tesnière
1893-1954



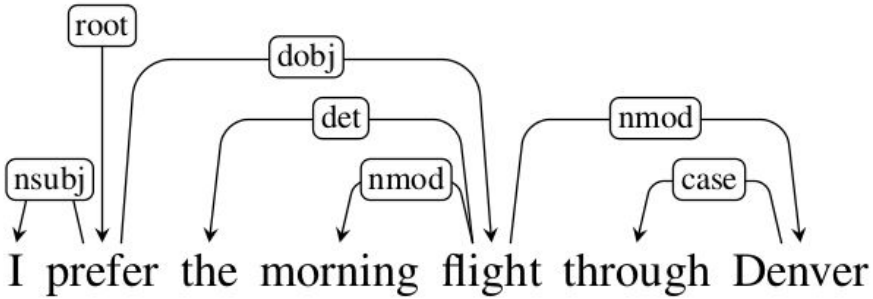
A note on the “free word order”

“If the first linguists had been 'O'odham* speakers, and if they were predisposed to assume that all reasonable languages operate on the basis of the same function-structure mappings as does their native language (as have the majority of scholars of language to date), **then English would be viewed as a "free" word-order language**”

Doris L. Payne. Nonidentifiable mentions and order in 'O'odham.
// Pragmatics of Word Order Flexibility, ed. by Doris L. Payne, Amsterdam: John Benjamins, 1992. Pp. 137-166

*O’odham — Uto-Aztecan language

Example



Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

A tree, not similar to what we've had with PSG: (1) all **flight**'s arguments are connected directly with it, (2) each connection has a type (typed dependency structure) ~~e i e i e~~

Where do we get dependency treebanks from?

Dependency treebanks

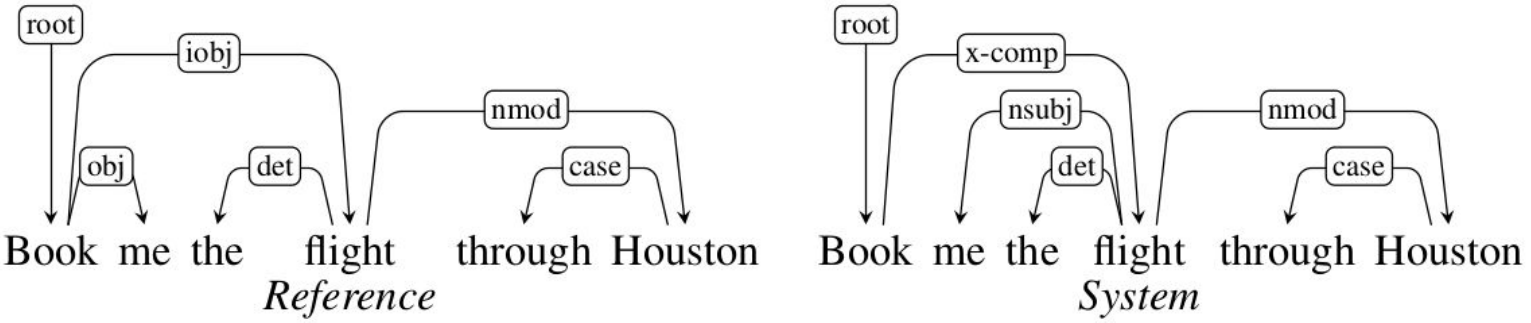
- made by experts
  (possibly by fixing the third-party-tools-based annotation)

- converted from PSG treebanks
  (note: due to that the corresponding grammars are **projective**, i.e. all the words between any pair of the head and dependent words are reachable by arcs from the head one; **sometimes, however, it is not so!**)

Plan

1. What is parsing and why we need it
2. Phrase structure grammar
3. ~~Dependency grammar~~
    a. ~~What is DG~~
    b. Evaluation metrics
    c. Approaches to parsing
    d. Transition-based DP, intuitively
    e. Transition-based DP
    f. Tools and datasets

Quality evaluation



Labeled Attachment Score (LAS) = 4/6
(correct labeled pairs)

Unlabeled Attachment Score (UAS) = 5/6
(correct pairs (we don't care about labels))

Precision and recall of certain relations can be computed as well

Parsing

dependency tree construction given a sentence

Two main approaches

- **transition-based**
  building the tree in a greedy fashion (discussed further)

- **graph-based**
  search in the space of all parse trees, assigning scores to subtrees (non-projectivity + achieves better results for long sentences)

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}} score(t, S)$$

Parsing

dependency tree construction given a sentence

Two main approaches

- **transition-based**
  building the tree in a greedy fashion (discussed further)

- ~~**graph-based**~~
  ~~*search in the space of all parse trees, assigning scores to subtrees (non-projectivity + achieves better results for long sentences)*~~

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}} score(t, S)$$

Plan

1. What is parsing and why we need it
2. Phrase structure grammar
3. ~~Dependency grammar~~
    a. ~~What is DG~~
    b. ~~Evaluation metrics~~
    c. ~~Approaches to parsing~~
    d. Transition-based DP, intuitively
    e. Transition-based DP
    f. Tools and datasets

Transition-Based DP: intuitively

Let us imagine we play a ***Wheel of Fortune***-like quiz* with words, not letters: we are shown one word at a time (consecutively) and we have to guess the correct parse tree

The third player!
Roll the wheel!

Book...

Something about the book, obviously; the word ‘book’ could e.g. be a subject of the sentence

* the images come from the oldschool videogame based on the memetic TV quiz “Поле чудес” (1990 - up to now)

Transition-Based DP: intuitively

Let us imagine we play a *Wheel of Fortune*-like quiz with words, not letters: we are shown one word at a time (consecutively) and we have to guess the correct parse tree

Book me...

Not a subject! It's either a request to book one, or to book something else for someone, but **me** clearly depends on the verb **book**

Say the word now?

Transition-Based DP: intuitively

Let us imagine we play a *Wheel of Fortune*-like quiz with words, not letters: we are shown one word at a time (consecutively) and we have to guess the correct parse tree

Book me the...

not clear yet, but looks like a **request** to book SOMETHING

Transition-Based DP: intuitively

Let us imagine we play a *Wheel of Fortune*-like quiz with words, not letters: we are shown one word at a time (consecutively) and we have to guess the correct parse tree

Book me the morning...

Booking mornings sounds a bit strange, however, **morning** can depend on **book** in some situations: “book me the morning at my dentist’s”

Random prize
or do we move on?

Transition-based dependency parsing: intuitively

Let us imagine we play a *Wheel of Fortune*-like quiz with words, not letters: we are shown one word at a time (consecutively) and we have to guess the correct parse tree

Book me the morning flight.

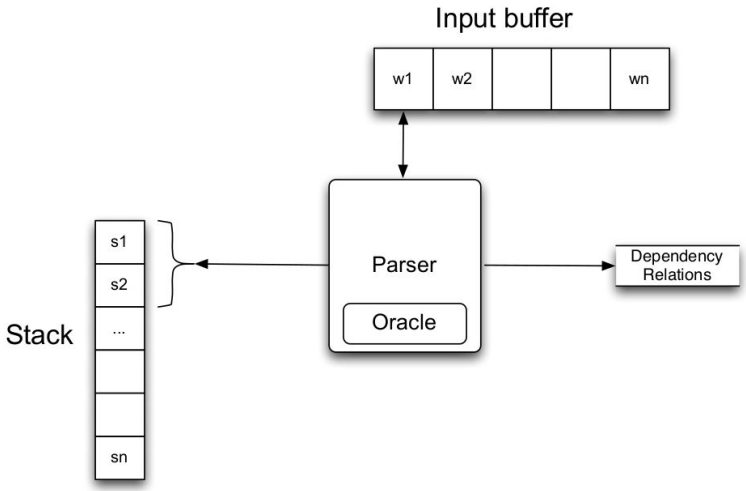
Everything is clear now!

Plan

1. What is parsing and why we need it
2. Phrase structure grammar
3. ~~Dependency grammar~~
    a. ~~What is DG~~
    b. ~~Evaluation metrics~~
    c. ~~Approaches to parsing~~
    d. ~~Transition-based DP, intuitively~~
    e. Transition-based DP
    f. Tools and datasets

Transition-based dependency parsing

Key concept: “**configuration**” = **the state** of parsing process:
input tokens, top of the stack and a set of already saved relations
(what we kept in mind during the ‘quiz’; the analogy is not entirely correct)



This is why it is called **transition-based**: we are going to move **from state to state** of the system using the rules

Aho, A. V. and Ullman, J. D. (1972). The Theory of Parsing, Translation, and Compiling, Vol. 1. Prentice Hall.

Transition-based dependency parsing

In the most simple setting we have three possible steps modifying the configuration:

- **LeftArc** [apply if the second element of the stack is not ROOT]
  saving the dependency between the top token on the stack and the second one + removing the second one from the stack

- **RightArc**
  the same, but with different dependency orientation, and removing the top

- **Shift**
  moving the consecutive word onto the stack

Example

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Pseudocode

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)         ; choose a transition operator to apply
    state ← APPLY(t, state)   ; apply it, creating a new state
  return state
```

LeftArc and RightArc could save **typed** dependencies as well

The approach has its limitations, but it's simple, *greedy* and effective

The most important thing to do here is to **train a good oracle**. But how?

Training = “simulation”

Using the treebank, we construct the configurations and ‘decisions’ the oracle is to make:

1. We go along the sentence
2. We simulate the **LeftArc** action, if it sets the relation (arc)
   that is **present in the parse tree**
3. Else -- **RightArc**, if
   (1) if it sets the relation (arc) that is present in the parse tree and
   (2) all tokens depending from the token on the stack’s top have already
   been ‘added’ to the simulated parse
4. Else -- **Shift**

As a results we have **a set of configurations and corresponding steps** as a training set; the features are to be invented by us

Plan

1. What is parsing and why we need it
2. Phrase structure grammar
3. ~~Dependency grammar~~
    a. ~~What is DG~~
    b. ~~Evaluation metrics~~
    c. ~~Approaches to parsing~~
    d. ~~Transition-based DP, intuitively~~
    e. ~~Transition-based DP~~
    f. Tools and datasets

Tools

- UDPipe
- SyntaxNet (Google)
- Ark Parser (CMU: Noah’s Ark group)
- spaCy.io
- Many more...

Datasets

1. http://universaldependencies.org/
2. National corpora
3. Many more...

Used/recommended literature

1. Martin, Jurafsky. Chapter 14
2. [Russian] Я.Г. Тестелец. Введение в общий синтаксис. М.: РГГУ, 2001. — 798 с.
3. [Russian] Прикладная и компьютерная лингвистика (под ред. И.С. Николаева, О.В. Митрениной, Т.М. Ландо)

Syntax — II

Anton Alekseev
Steklov Mathematical Institute in St Petersburg

ITMO University, St Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Thanks for reading the slides and useful comments go to Denis Kiryanov