Lecture 10
Syntactic parsing
(and some other tasks)

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
21.04.2021
Plan for today

● Why syntax?
● Syntactic parsing
  ● Shallow parsing
  ● Constituency parsing
  ● Dependency parsing
● Neural graph-based dependency parsing
● Coreference resolution
Why syntax?
Syntax

- Internal structure of words

```
  S
  / \  /
 NP  VP
 /   / /
John V NP
     /  /
    hit Det N
     /  /
    the ball
```
Syntactic ambiguity

Sherlock saw the man using binoculars.
Sherlock saw the man using binoculars.
The chicken is ready to eat.
“One morning I shot an elephant in my pajamas.

How he got in my pajamas, I don’t know.”

Groucho Marx
Exercise

How many different meanings has the sentence:

Time flies like an arrow.
The role of syntax in NLP

- Text generation/summarization/machine translation
- Can be helpful in grammatical error correction
- Useful features for various information extraction tasks
- Syntactic structure also reflects the semantic relations between the words
Syntactic analysis/parsing

- Shallow parsing
- Phrase structure / constituency parsing
- Dependency parsing
Shallow parsing
Shallow parsing

- Also called **chunking** or **light parsing**
- Split the sentence into non-overlapping syntactic phrases

**Diagram:**
- S
- NP: John
- VP
- V: hit
- NP: the ball

**Sentence:** John hit the ball.

**Analysis:**
- NP – Noun phrase
- VP – Verb phrase
Shallow parsing

The morning flight from Denver has arrived.

NP – Noun phrase
PP – Prepositional Phrase
VP – Verb phrase
BIO tagging

A labelling scheme often used in information extraction problems, treated as a sequence tagging task

The    morning flight from    Denver has    arrived.

B_NP   I_NP   I_NP B_PP  B_NP  B_VP  I_VP

B_NP – Beginning of a noun phrase
I_NP – Inside a noun phrase
B_VB – Beginning of a verb phrase etc
Sequence classifier

- Need annotated data for training: POS-tagged, phrase-annotated

- Use a sequence classifier of your choice
  - CRF with engineered features
  - Neural sequence tagger
  - BERT-based
Constituency Parsing
Constituency parsing

- Full constituency parsing helps to resolve structural ambiguities
Context-free grammars

S → NP VP
VP → V NP
VP → V NP PP
NP → NP NP
NP → NP PP
NP → N
NP → e
PP → P NP

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with

\[ G = (T, N, S, R) \]
- T – set of terminal symbols
- N – set of non-terminal symbols
- S – start symbol (\( S \in N \))
- R – set of rules/productions of the form \( X \rightarrow \gamma \)
  - \( X \in N \)
  - \( \gamma \in (N \cup T)^* \)

A grammar G generates a language L
Probabilistic PCFG

\[ G = (T, N, S, R, P) \]

- \( T \) – set of terminal symbols
- \( N \) – set of non-terminal symbols
- \( S \) – start symbol (\( S \in N \))
- \( R \) – set of rules/productions of the form \( X \rightarrow \gamma \)
- \( P \) – probability function
  - \( P: R \rightarrow [0, 1] \)
  - \( \forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1 \)

A grammar \( G \) generates a language model \( L \)
A PCFG

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>1.0</td>
<td>N → people</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>0.6</td>
<td>N → fish</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → V NP PP</td>
<td>0.4</td>
<td>N → tanks</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → NP NP</td>
<td>0.1</td>
<td>N → rods</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>0.2</td>
<td>V → people</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → N</td>
<td>0.7</td>
<td>V → fish</td>
<td>0.6</td>
</tr>
<tr>
<td>PP → P NP</td>
<td>1.0</td>
<td>V → tanks</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P → with</td>
<td>1.0</td>
</tr>
</tbody>
</table>
The probability of strings and trees

- $P(t)$ – the probability of a tree $t$ is the product of the probabilities of the rules used to generate it.

$$P(t) = \prod_{X \rightarrow \gamma \in t} P(X \rightarrow \gamma)$$

- $P(s)$ – The probability of the string $s$ is the sum of the probabilities of the trees which have that string as their yield

$$P(s) = \sum_j P(s, t_j) = \sum_j P(t_j)$$

- where $t$ is a parse of $s$
Constituency parsing

- There are dynamic programming algorithms (CKY) to find the most likely tree
  - Proceeds bottom-up and performs Viterbi on trees
  - The span scores can be computed by a neural model

- Constituency parsing displays the whole structure of the sentences
- Constituency parsing is more useful for languages, where the word order is relatively fixed (like English)
Dependency Parsing
Dependency parsing

- Dependency parsing is another syntactic formalism
- It expresses the dependency relations between pairs of words
  - Constituency parsing expressed the structure of the whole sentence.
- Suitable for language with free word order
Dependency parsing

• Dependency parse is a directed graph $G = (V, A)$
  • $V$ – the set of vertices corresponding to words
  • $A$ – the set of nodes corresponding to dependency relations

```
PRP  nsubj  VBP  DT  NN  dobj  nmod  det  compound
I    prefer  the  morning
```

```
NN  det  IN  case  NNP
flight  through  Denver
```
Dependency parsing

```
prefer
I

flight

the morning Denver

through

S

NP

Pro

Verb

I prefer

NP

Det

the Nom

Nom

Noun flight through

P

NP

Noun morning

Pro Denver
```
Dependency relations

- The arrows connect heads and their dependents
- The main verb is the head or the root of the whole sentence
- The arrows are labelled with grammatical functions/dependency relations
Properties of a dependency graph

A dependency tree is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs
   - Typically the main verb of the sentence

2. Except for the root node, each node has exactly one incoming arc
   - Each dependent has a single head

3. There is a unique path from the root node to each vertex in V
   - The graph is acyclic and connected
Projectivity

- Projective trees – there are no arc crossings in the dependency graphs
- Non-projective trees - crossings due to free word order
## Dependency relations

<table>
<thead>
<tr>
<th>Clause argument relations</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUBJ</td>
<td>Nominal subject</td>
<td>United canceled the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>Direct object</td>
<td>United canceled the flight.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>Indirect object</td>
<td>We booked her the flight to Paris.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nominal modifier relations</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMOD</td>
<td>Nominal modifier</td>
<td>We took the morning flight.</td>
</tr>
<tr>
<td>AMOD</td>
<td>Adjectival modifier</td>
<td>Book the cheapest flight.</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Numeric modifier</td>
<td>Before the storm JetBlue canceled 1000 flights</td>
</tr>
<tr>
<td>APPOS</td>
<td>Appositional modifier</td>
<td>United, a unit of UAL, matched the fares.</td>
</tr>
<tr>
<td>DET</td>
<td>Determiner</td>
<td>Which flight was delayed?</td>
</tr>
<tr>
<td>CASE</td>
<td>Prepositions, postpositions</td>
<td>Book the flight through Houston.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other notable relations</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONJ</td>
<td>Conjunct</td>
<td>We flew to Denver and drove to Steamboat</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
</tbody>
</table>
Universal dependencies

- [http://universaldependencies.org/](http://universaldependencies.org/)
- Annotated treebanks in many languages
- Uniform annotation scheme across all UD languages:
  - Universal POS tags
  - Universal morphological features
  - Universal dependency relations
- All data in CONLL-U tabulated format
<table>
<thead>
<tr>
<th></th>
<th>Nominals</th>
<th>Clauses</th>
<th>Modifier words</th>
<th>Function Words</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core arguments</strong></td>
<td>nsubj</td>
<td>csubj</td>
<td>advmod*</td>
<td>aux</td>
</tr>
<tr>
<td></td>
<td>obj</td>
<td>ccomp</td>
<td>discourse</td>
<td>cop</td>
</tr>
<tr>
<td></td>
<td>obj</td>
<td>xcomp</td>
<td></td>
<td>mark</td>
</tr>
<tr>
<td><strong>Non-core dependents</strong></td>
<td>obl</td>
<td>advcl</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>vocative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>expl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dislocated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nominal dependents</strong></td>
<td>nmod</td>
<td>acl</td>
<td>amod</td>
<td>det</td>
</tr>
<tr>
<td></td>
<td>appos</td>
<td></td>
<td></td>
<td>clf</td>
</tr>
<tr>
<td></td>
<td>nummod</td>
<td></td>
<td></td>
<td>cage</td>
</tr>
<tr>
<td><strong>Coordination</strong></td>
<td>MWE</td>
<td>Loose</td>
<td>Special</td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td>conj cc</td>
<td>fixed</td>
<td>list</td>
<td>orphan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flat</td>
<td>parataxis</td>
<td>goeswith</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compound</td>
<td></td>
<td>reparandum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>punct</td>
</tr>
</tbody>
</table>
CONLL-U format

- A standard tabular format for certain type of annotated data
- Each word is in a separate line
- 10 tab-separated columns on each line:
  1. Word index
  2. The word itself
  3. Lemma
  4. Universal POS
  5. Language specific POS
  6. Morphological features
  7. Head of the current word
  8. Dependency relation to the head
  9. Enhanced dependencies
  10. Any other annotation
CONLL-U format: example

# text = They buy and sell books.

1. They  they  PRON  PRP  Case=Nom|Number=Plur
2. buy  buy  VERB  VBP  Number=Plur|Person=3|Tense=Pres
3. and  and  CONJ  CC  _
4. sell  sell  VERB  VBP  Number=Plur|Person=3|Tense=Pres
5. books  book  NOUN  NNS  Number=Plur
6. .  .  PUNCT  .  _

# text = I had no clue.

1. I  I  PRON  PRP  Case=Nom|Number=Sing|Person=1
2. had  have  VERB  VBD  Number=Sing|Person=1|Tense=Past
3. no  no  DET  DT  PronType=Neg
4. clue  clue  NOUN  NN  Number=Sing
5. .  .  PUNCT  .  _
Dependency parsing methods

- Transition-based parsing
  - stack-based algorithms/shift-reduce parsing
  - only generate projective trees

- Graph-based algorithms
  - can also generate non-projective trees
Neural dependency parsers
stanfordnlp graph-based parser

- Inputs are words and their POS tags
- Hidden representations are created with a biLSTM

Dozat et al., 2017. Stanford's Graph-Based Neural Dependency Parser at the CoNLL 2017 Shared Task.
From each word create four vectors for classification:

- for head word
- for head relation
- for dependent word
- for dependent relation

This is achieved using 4 fully connected (FC) layers (+ReLU)

Dozat et al., 2017. Stanford's Graph-Based Neural Dependency Parser at the CoNLL 2017 Shared Task.
stanfordnlp graph-based parser

- Use biaffine classifier to score the dep vector of each word with the head vector of every other word
- The scores for the word $i$ between all possible heads:

$$s_{i}^{(arc)} = H^{(arc-head)}W^{(arc)}h_{i}^{(arc-dep)}$$

- Then find the most probable head:

$$y_{i}^{(arc)} = \arg\max_{j} s_{ij}^{(arc)}$$

- $H^{(arc-head)}$ - the matrix gathering the head vectors for all words
Then use biaffine classifier to score the relation between the head and dependent using the respective relation vectors:

\[
\mathbf{s}_i^{(rel)} = \mathbf{h}_i^{(rel-head)} \mathbf{U}^{(rel)} \mathbf{h}_i^{(rel-dep)} + \mathbf{W}^{(rel)} \mathbf{h}_i^{(rel-dep)} + \mathbf{h}_i^{(rel-head)}
\]

Finally find the relation with maximum score:

\[
y_i^{(rel)} = \text{argmax}_j s_{ij}^{(rel)}
\]
stanfordnlp graph-based parser

- In reality, the parser is a bit more complex
  - The head and dependend relative positions are also modeled:
    - whether the head is to the left or to the right of the depend
    - what is the distance between the head and dependent

- The resulting graph can contain cycles
- thus post-processing is applied to make sure that the result is a valid dependency graph
Evaluating dependency parsing
Evaluation

Unlabelled attachment score:
● The proportion of correct head attachments

Labelled attachment score:
● The proportion of correct head attachments labelled with the correct relation

Label accuracy
● The proportion of correct incoming relation labels ignoring the head
Evaluation

UAS =
LAS =
LA =
Evaluation

UAS = 5/6
LAS = 4/6
LA = 4/6
Coreference resolution
Discourse model

Discourse Model

Lotsabucks

Megabucks

$ pay

“Victoria”

refer (access)

refer (evoke)

corefer

“she”

antecedent

anaphor

47
Coreference resolution

- The goal is to link together **coreferring mentions** in the text
  - i.e. the mentions that refer to the same entity in the discourse model
  - the result is a set of coreference chains or clusters

Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old also became the company’s president. It is widely known that she came to Megabucks from rival Lotsabucks.

- What kind of entities can we see in this text?
- What kind of mentions for each entity?
Victoria Chen\textsubscript{1}, CFO of Megabucks Banking\textsubscript{2}, saw her\textsubscript{1} pay\textsubscript{3} jump to $2.3 million, as the 38-year-old\textsubscript{1} also became the company\textsubscript{2}’s president. It is widely known that she\textsubscript{1} came to Megabucks\textsubscript{2} from rival Lotsabucks\textsubscript{4}.

1. \{Victoria Chen, her, the 38-year-old, She\}
2. \{Megabucks Banking, the company, Megabucks\}
3. \{her pay\}
4. \{Lotsabucks\}
Coreference resolution approaches

- **Pipeline approach**
  - Mention detection – extract noun phrases, possessive pronouns, named entities
  - Filter out spurious mentions – rules and classifiers
  - Clustering into coreference chains – link anaphors to antecedents using classifiers

- **Joint approach**
  - neural end-to-end models
Joint approach

- Consider all n-gram spans up to certain length as mention candidates
- Assign to each span $i$ an antecedent from the set of spans \{1, ..., $i-1$\}, or a dummy token that denotes no antecedent
- Compute the score between the span pairs. The score takes into account three factors:
  - whether the first span is a mention
  - whether the second span is a mention
  - whether there is a coreference relation between the two mentions
- The representation of each span is computed by a neural encoder:
  - biLSTM, ELMo, BERT, ...

Lee et al., 2017. End-to-end neural coreference resolution
Span representations

Mention score (m)

Span representation (g)

Span head (h_{att})

Bidirectional LSTM (h)

Input word embeddings (ELMo)

General Electric Electric said the the Postal Service Service contacted the the company

Lee et al., 2017. End-to-end neural coreference resolution
Computing scores

\[ \text{Softmax} \ (P(y_i \mid D)) \]

\[ s(\text{the company}, \epsilon) = 0 \]

\[ s(\text{the company, General Electric}) \]

\[ s(\text{the company, the Postal Service}) \]

Coreference score \((s)\)

Antecedent score \((c)\)

Mention score \((m)\)

Span representation \((g)\)

General Electric  the Postal Service  the company

Lee et al., 2017. End-to-end neural coreference resolution
What’s next?
Coming next

● Pre-recorded video lecture next week about speech recognition
● Then two guest lectures during the scheduled lecture time:
  ● 05.05 – Liisa Rätsep about speech synthesis
  ● 12.05 – Hendrik Luuk from AlphaBlues about chatbots
● Then focussing on doing the final homeworks and finishing the projects
In summary

- There are lots of different tasks in NLP
- Some are easier and some are more challenging
  - tasks for inferring simpler grammatical structures (like POS tags) are easier
  - generation and semantic tasks are challenging
- Most tasks can be formalised some kind of classification:
  - text classification
  - token classification
  - span/pair/relation classification
- Typical architecture:
  - encoder: biLSTM (with attention), BERT, CNN, ...
  - and then the suitable classifier on top of that
- Although some models work pretty well on some tasks, most NLP tasks are still far from resolved.