NLP Data Structures

PU Tools, Ressourcen, Infrastruktur

Nils Reiter,
nils.reiter@uni-koeln.de

January 7, 2021
(Winter term 2020/21)

Section 1

Recap

Exercise 7

https://github.com/idh-cologne-tools-ressourcen-infra/exercise-07

Section 2

NLP Data Structures

Data Structures

- Storing information digitally
- Atomic values: `int`, `String`, `boolean`, ...
- More complex data: Combinations of multiple atomic values

Data Structures

- Storing information digitally
- Atomic values: `int`, `String`, `boolean`, ...
- More complex data: Combinations of multiple atomic values

Example (Address)

$$\begin{bmatrix} \text{STREET} & \textit{String} \\ \text{HOUSE} & \textit{int} \\ \text{ZIP} & \textit{int} \\ \text{CITY} & \textit{String} \end{bmatrix}$$

Data Structures

- Storing information digitally
- Atomic values: `int`, `String`, `boolean`, ...
- More complex data: Combinations of multiple atomic values

Example (Address)

$$\begin{bmatrix} \text{STREET} & \textit{String} \\ \text{HOUSE} & \textit{int} \\ \text{ZIP} & \textit{int} \\ \text{CITY} & \textit{String} \end{bmatrix}$$

Example (Music)

$$\begin{bmatrix} \text{ARTIST} & \textit{String} \\ \text{TITLE} & \textit{String} \\ \text{RELEASE} & \textit{int} \\ \text{TRACKS} & \left\langle \begin{bmatrix} \text{TITLE} & \textit{String} \\ \text{LENGTH} & \textit{time} \\ \text{RATING} & \textit{int} \end{bmatrix}, \ \ldots \right\rangle \\ \text{RATING} & \textit{int} \\ \text{COMPILATION} & \textit{boolean} \end{bmatrix}$$

Data Structures

Collections

- Storing individual objects is often clear
- Collections of values of the same type
  - Impact on programming experience
  - Impact on performance

Data Structures

Collections

- Storing individual objects is often clear
- Collections of values of the same type
  - Impact on programming experience
  - Impact on performance
- Commonly used collection data structures (e.g., in interviews)
  - Arrays, stacks, queues, linked lists, trees, graphs, ...
  - Sometimes hidden and/or merged by programming libraries

NLP Data

- Text and annotations
- Annotations: Meta data associated with specific locations in the text

Storing Text

- Texts can be stored as strings
- Beware: Unicode
- Limitations on length

NLP Tasks

- Tokenization, sentence splitting
- Morphological analysis
- Part of speech-tagging, lemmatisation
- Named entity recognition
- Phrase structure and dependency syntax
- Coreference chains

NLP Tasks

Tokenization

Example

Sentence: The dog is fed by Mr. Parker, who wears a yellow hat.
Tokens: [The] [dog] [is] [fed] [by] [Mr.] [Parker] [,] [who] [wears] [a] [yellow] [hat] [.]

NLP Tasks

Tokenization

Example

Sentence: The dog is fed by Mr. Parker, who wears a yellow hat.
Tokens: [The] [dog] [is] [fed] [by] [Mr.] [Parker] [,] [who] [wears] [a] [yellow] [hat] [.]

- Split the text into ›meaningful‹ units (e.g., words)
- Two storage approaches

A list of strings / token objects

$$\left\langle \left[\text{SURFACE } \textit{The}\right], \left[\text{SURFACE } \textit{dog}\right], \ldots \right\rangle$$

NLP Tasks

Tokenization

Example

Sentence: The dog is fed by Mr. Parker, who wears a yellow hat.
Tokens: [The] [dog] [is] [fed] [by] [Mr.] [Parker] [,] [who] [wears] [a] [yellow] [hat] [.]

- Split the text into ›meaningful‹ units (e.g., words)
- Two storage approaches

A list of strings / token objects

$$\left\langle \left[\text{SURFACE } \textit{The}\right], \left[\text{SURFACE } \textit{dog}\right], \ldots \right\rangle$$

A list of character positions (preferred)

$$\begin{bmatrix} \text{TEXT} & \textit{»The dog is fed ...«} \\ \text{TOKS} & \left\langle \begin{bmatrix} \text{BEG} & \textit{0} \\ \text{END} & \textit{3} \end{bmatrix}, \begin{bmatrix} \text{BEG} & \textit{4} \\ \text{END} & \textit{7} \end{bmatrix}, \ldots \right\rangle \end{bmatrix}$$

NLP Tasks

Sentence splitting

- Split a text into sentences
- Storing sentences as strings
  - Easy to pass into the next component
  - Difficult to access token-based information

NLP Tasks

Sentence splitting

- Split a text into sentences
- Storing sentences as strings
  - Easy to pass into the next component
  - Difficult to access token-based information
- Storing sentences as character positions
  - Easy to combine with token-based information
  - Not complicated to extract the sentence string
    - `text.substring(begin, end)`

NLP Tasks

Sentence splitting

- Split a text into sentences
- Storing sentences as strings
  - Easy to pass into the next component
  - Difficult to access token-based information
- Storing sentences as character positions
  - Easy to combine with token-based information
  - Not complicated to extract the sentence string
    - `text.substring(begin, end)`

$$\begin{bmatrix} \text{TEXT} & \text{»}\textit{The dog is fed ...}\text{«} \\ \text{TOKS} & \left\langle \begin{bmatrix} \text{BEG} & 0 \\ \text{END} & 3 \end{bmatrix}, \begin{bmatrix} \text{BEG} & 4 \\ \text{END} & 7 \end{bmatrix}, \ldots \right\rangle \\ \text{SENT} & \left\langle \begin{bmatrix} \text{BEG} & 0 \\ \text{END} & 53 \end{bmatrix}, \ldots \right\rangle \end{bmatrix}$$

NLP Tasks

Part of speech-tagging and lemmatization

- PoS-tagging: Assign word categories (noun, verb, ...) to each token
- Lemmatization: Determine the base form of each word

NLP Tasks

Part of speech-tagging and lemmatization

- PoS-tagging: Assign word categories (noun, verb, ...) to each token
- Lemmatization: Determine the base form of each word

$$\left\langle \begin{bmatrix} \text{SURF} & \textit{The} \\ \text{POS} & \textit{det} \\ \text{LEM} & \textit{the} \end{bmatrix}, \begin{bmatrix} \text{SURF} & \textit{dog} \\ \text{POS} & \textit{NN} \\ \text{LEM} & \textit{dog} \end{bmatrix}, \ldots \right\rangle$$

NLP Tasks

Part of speech-tagging and lemmatization

- PoS-tagging: Assign word categories (noun, verb, ...) to each token
- Lemmatization: Determine the base form of each word

$$\left\langle \begin{bmatrix} \text{SURF} & \textit{The} \\ \text{POS} & \textit{det} \\ \text{LEM} & \textit{the} \end{bmatrix}, \begin{bmatrix} \text{SURF} & \textit{dog} \\ \text{POS} & \textit{NN} \\ \text{LEM} & \textit{dog} \end{bmatrix}, \ldots \right\rangle$$

$$\begin{bmatrix} \text{TXT} & \textit{»The dog is fed ...«} \\ \text{TOK.} & \left\langle \begin{bmatrix} \text{BEG} & \textit{0} \\ \text{END} & \textit{3} \\ \text{POS} & \textit{det} \\ \text{LEM} & \textit{the} \end{bmatrix}, \begin{bmatrix} \text{BEG} & \textit{4} \\ \text{END} & \textit{7} \\ \text{POS} & \textit{NN} \\ \text{LEM} & \textit{dog} \end{bmatrix}, \ldots \right\rangle \end{bmatrix}$$

- Straightforward in any case

Accessing Information

- Extract all nouns: Iterate over each token, collect nouns

Accessing Information

- Extract all nouns: Iterate over each token, collect nouns
- Alternative: PoS-Tagger produces a separate list
  - $\left[\text{NOUNS}\ \left\langle \left[\text{SURF}\ \textit{dog}\right], \left[\text{SURF}\ \textit{hat}\right] \right\rangle\right]$
  - $\left[\text{NOUNS}\ \left\langle \begin{bmatrix}\text{BEG} & \textit{4}\\ \text{END} & \textit{7}\end{bmatrix}, \begin{bmatrix}\text{BEG} & \textit{49}\\ \text{END} & \textit{52}\end{bmatrix} \right\rangle\right]$

Accessing Information

- Extract all nouns: Iterate over each token, collect nouns
- Alternative: PoS-Tagger produces a separate list
  - $\left[\text{NOUNS}\ \left\langle \left[\text{SURF}\ \textit{dog}\right], \left[\text{SURF}\ \textit{hat}\right] \right\rangle\right]$
  - $\left[\text{NOUNS}\ \left\langle \begin{bmatrix}\text{BEG}\ \textit{4}\\ \text{END}\ \textit{7}\end{bmatrix}, \begin{bmatrix}\text{BEG}\ \textit{49}\\ \text{END}\ \textit{52}\end{bmatrix} \right\rangle\right]$
- Often: Trade-off between storage and time
  - I.e., storing more information makes access faster, but consumes more space (even if you only store pointers)

NLP Tasks

Named Entity Recognition

- Detection of proper names in texts
- Classification according to entity type
  - Person, Organisation, Location, ...

NLP Tasks

Named Entity Recognition

- Detection of proper names in texts
- Classification according to entity type
  - Person, Organisation, Location, ...
- Multi-word expressions: »Donald Trump«, »the European Union«, ...

NLP Tasks

Named Entity Recognition

- Detection of proper names in texts
- Classification according to entity type
  - Person, Organisation, Location, ...
- Multi-word expressions: »Donald Trump«, »the European Union«, ...
- Overlapping: »Die [[Oswald von [Wolkenstein]$_{\text{LOC}}$]$_{\text{PER}}$-Gesellschaft]$_{\text{ORG}}$ ist ...«

NLP Tasks

Named Entity Recognition

- Detection of proper names in texts
- Classification according to entity type
  - Person, Organisation, Location, ...
- Multi-word expressions: »Donald Trump«, »the European Union«, ...
- Overlapping: »Die [[Oswald von [Wolkenstein]$_{\text{LOC}}$]$_{\text{PER}}$-Gesellschaft]$_{\text{ORG}}$ ist ...«

Data Structures

- Lists of token sequences:

$$\left[\text{NE}\ \left\langle \begin{bmatrix} \textit{named-entity} \\ \text{TYPE} \quad \textit{ORG} \\ \text{TOKENS}\ \langle \boxed{0}, \boxed{1}, \boxed{2}, \boxed{3} \rangle \end{bmatrix}, \begin{bmatrix} \textit{named-entity} \\ \text{TYPE} \quad \textit{PER} \\ \text{TOKENS}\ \langle \boxed{0}, \boxed{1}, \boxed{2} \rangle \end{bmatrix}, \begin{bmatrix} \textit{named-entity} \\ \text{TYPE} \quad \textit{LOC} \\ \text{TOKENS}\ \langle \boxed{2} \rangle \end{bmatrix} \right\rangle\right]$$

NLP Tasks

Syntax

Phrase Structure Syntax (HPSG, LFG, Montague)

- Parse tree consists of phrases
- A phrase covers multiple words or other phrases
- Context-free grammar

NLP Tasks

Syntax

Phrase Structure Syntax (HPSG, LFG, Montague)

- Parse tree consists of phrases
- A phrase covers multiple words or other phrases
- Context-free grammar

Dependency Syntax

- Parse tree consists of words
- Edges in the tree represent dependency relation (instead of constituency)

NLP Tasks

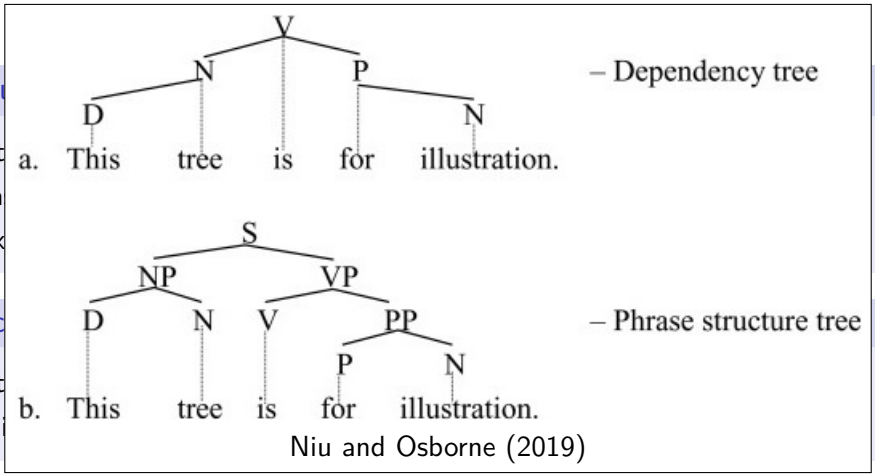
Syntax

Phrase Stru

- Parse t
- A phra
- Contex

Dependenc

- Parse t
- Edges i



Niu and Osborne (2019)

Trees

Formal Definition

Definition

1. A rooted tree is a connected, acyclic, undirected graph, with one designated node as root.

Trees

Formal Definition

Definition

1. A rooted tree is a connected, acyclic, undirected graph, with one designated node as root.

2. A tree is some payload data with a (possibly empty) list of children, which are also trees.

Trees

Formal Definition

Definition

1. A rooted tree is a connected, acyclic, undirected graph, with one designated node as root.

2. A tree is some payload data with a (possibly empty) list of children, which are also trees.

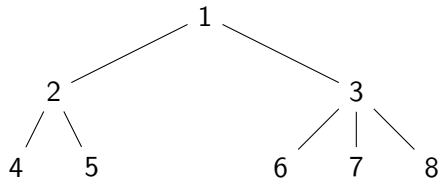


Figure: A tree

Trees

Formal Definition

Definition

1. A rooted tree is a connected, acyclic, undirected graph, with one designated node as root.

2. A tree is some payload data with a (possibly empty) list of children, which are also trees.

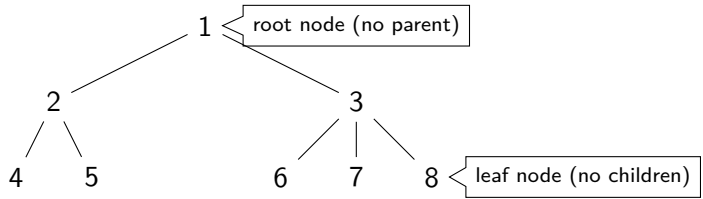
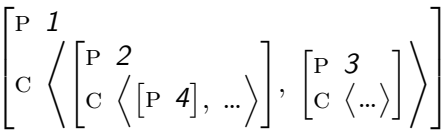


Figure: A tree

Trees

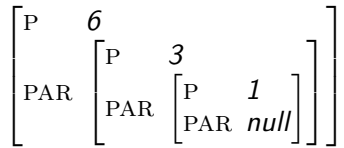
Listing: Top-down tree

```
1 public class Tree {
2   Object payload;
3   Tree[] children;
4 }
```



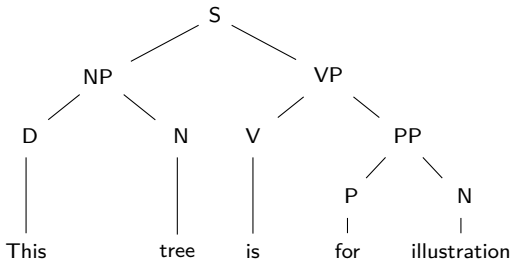
Listing: Bottom-up tree

```
1 public class Tree {
2   Object payload;
3   Tree parent;
4 }
```



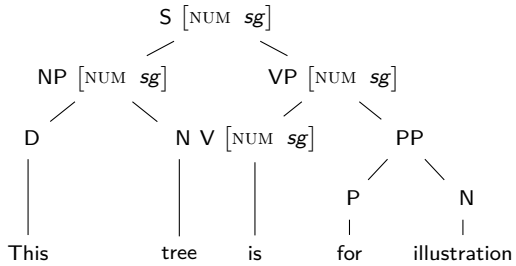
Phrase Structure Syntax

- Phrases: Separate entities in tree



Phrase Structure Syntax

- Phrases: Separate entities in tree
  - Phrases have properties (e.g., agreement in number)



A phrase:

$$\begin{bmatrix} \text{CAT} & S \\ \text{AGR} & [\text{NUM} \quad sg] \\ \text{CHI} & \langle \ldots \rangle \end{bmatrix}$$

Dependency Syntax

- Links between tokens (sometimes typed)
- Each token is thus linked to its governing token
- »This tree is for illustration.«
  - ›This‹ $\xrightarrow{\text{det}}$ ›tree‹, ›tree‹ $\xrightarrow{\text{nsubj}}$ ›is‹, ›for‹ $\xrightarrow{\text{prep}}$ ›is‹, ›illustration‹ $\xrightarrow{\text{pobj}}$ ›for‹, ›.‹ $\xrightarrow{\text{pobj}}$ ›is‹.

Dependency Syntax

- Links between tokens (sometimes typed)
- Each token is thus linked to its governing token
- »This tree is for illustration.«
  - ›This‹ $\xrightarrow{\text{det}}$ ›tree‹, ›tree‹ $\xrightarrow{\text{nsubj}}$ ›is‹, ›for‹ $\xrightarrow{\text{prep}}$ ›is‹, ›illustration‹ $\xrightarrow{\text{pobj}}$ ›for‹, ›.‹ $\xrightarrow{\text{pobj}}$ ›is‹.
- Storing is simple: Each token stores information for the governor and the relation type
  - conceptually, a bottom-up tree

Dependency Syntax

- Links between tokens (sometimes typed)
- Each token is thus linked to its governing token
- »This tree is for illustration.«
  - ›This‹ $\xrightarrow{\text{det}}$ ›tree‹, ›tree‹ $\xrightarrow{\text{nsubj}}$ ›is‹, ›for‹ $\xrightarrow{\text{prep}}$ ›is‹, ›illustration‹ $\xrightarrow{\text{pobj}}$ ›for‹, ›.‹ $\xrightarrow{\text{pobj}}$ ›is‹.
- Storing is simple: Each token stores information for the governor and the relation type
  - conceptually, a bottom-up tree

$$\begin{bmatrix} \text{TXT} & \textit{»This tree is for illustration.«} \\ \text{TOK.} & \left\langle \boxed{0} \begin{bmatrix} \text{BEG} & \textit{0} \\ \text{END} & \textit{4} \\ \text{GOV} & \boxed{1} \\ \text{REL} & \textit{det} \end{bmatrix}, \boxed{1} \begin{bmatrix} \text{BEG} & \textit{5} \\ \text{END} & \textit{9} \\ \text{GOV} & \boxed{3} \\ \text{REL} & \textit{nsubj} \end{bmatrix}, \boxed{2} \begin{bmatrix} \text{BEG} & \textit{10} \\ \text{END} & \textit{12} \\ \text{GOV} & \textit{null} \\ \text{REL} & \textit{null} \end{bmatrix}, \ldots \right\rangle \end{bmatrix}$$

Full Example

$$\begin{bmatrix}
\text{TXT} & \textit{»The dog is fed by Mr. Parker, who wears a yellow hat.«} \\
\text{TOK} & \left\langle \boxed{0} \begin{bmatrix} \text{BEG} & 0 \\ \text{END} & 3 \\ \text{POS} & DT \\ \text{LEM} & the \\ \text{GOV} & \boxed{1} \\ \text{REL} & det \end{bmatrix}, \boxed{1} \begin{bmatrix} \text{BEG} & 4 \\ \text{END} & 7 \\ \text{POS} & NN \\ \text{LEM} & dog \\ \text{GOV} & \boxed{2} \\ \text{REL} & nsubj \end{bmatrix}, \ldots \right\rangle \\
\text{NNS} & \left\langle \boxed{1}, \boxed{12} \right\rangle \\
\text{NES} & \left\langle \begin{bmatrix} \text{TYPE} & PER \\ \text{TOKENS} & \left\langle \boxed{5}, \boxed{6} \right\rangle \end{bmatrix} \right\rangle \\
\text{PHR} & \begin{bmatrix} \text{CAT} & S \\ \text{CHI} & \left\langle \begin{bmatrix} \text{CAT} & NP \\ \text{CHI} & \left\langle \boxed{0}, \boxed{1} \right\rangle \end{bmatrix}, \begin{bmatrix} \text{CAT} & VP \\ \text{CHI} & \langle \ldots \rangle \end{bmatrix} \right\rangle \end{bmatrix}
\end{bmatrix}$$

Summary

- NLP data structures are complex, because language is
- Annotations can be rooted in tokens or character positions
- Many linguistic structures can be represented as trees
  - Trees are a special kind of graph

Section 3

Next Exercise

Exercise 8

Section 4

Appendix

References I


Niu, Ruochen and Timothy Osborne (2019). »Chunks are components: A dependency grammar approach to the syntactic structure of Mandarin«. In: *Lingua* 224, pp. 60 –83. ISSN: 0024-3841. DOI: https://doi.org/10.1016/j.lingua.2019.03.003. URL: http://www.sciencedirect.com/science/article/pii/S0024384118308350.