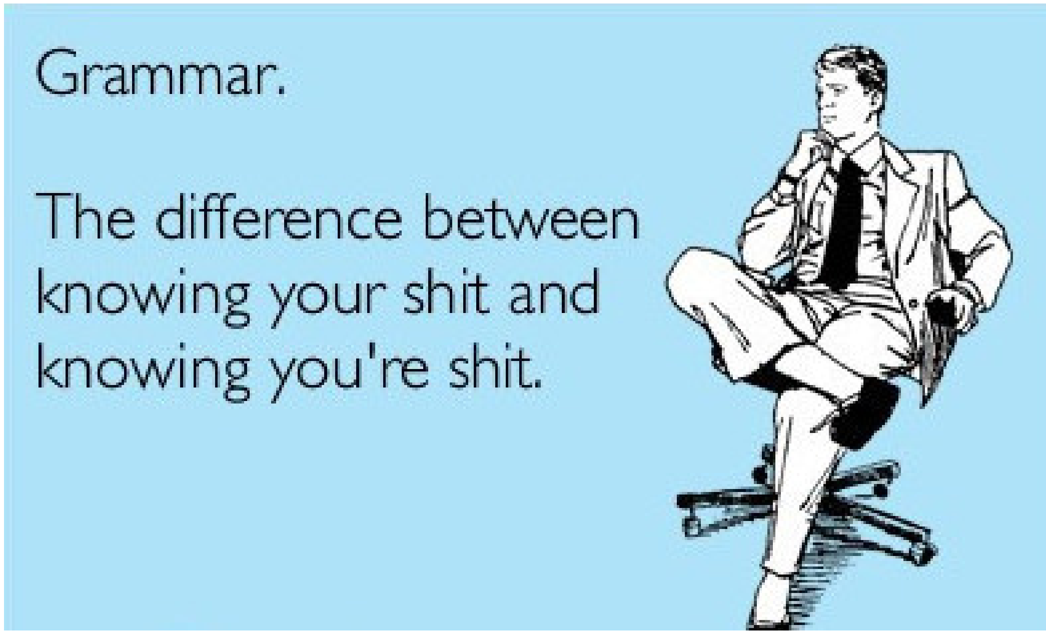
Chapter NLP:IV

IV. Syntax

- Introduction
- Regular Grammars
- Probabilistic Context-Free Grammars
- Parsing based on a PCFG
- Dependency Grammars

Introduction



Introduction

Syntax: an Example

Problem: Given a set of word forms (or words), what sequences are grammatical or meaningful sentences?

A bag of words: `Sun, Leipzig, the, shining, warm, in`

- The Leipzig shining warm the sun.
- In Leipzig warm sun the shining.
- warm the in sun shining.
- The sun is shining in Leipzig.

Introduction

Definition 1

Syntax is the study of principles and processes by which sentences are constructed in particular languages *Chomsky*

1. Syntactic structure of a single language (e.g. German, English, . . .)
   - Which elements?
   - How combined?

2. Grammar of a single language (e.g. German, English, . . .)
   - Syntactic features of a single language
   - Formalization of grammar
   - Conditions of grammaticality

3. Universal grammar (conditions of human language comprehension)
   - Features of human linguistic ability
   - Basics and limits of its formalization

Introduction

Requirements on the syntactic description

1. Linear sequence of words (time linearity)
   - Only one word after the other can be uttered and perceived

2. Sentences contain syntactic ambiguities
   - “Flying planes can be dangerous” (Chomsky)

3. Hierarchic structuring of phrases
   - Some expressions are "closer" to each other than others, e.g. "flying planes", "can be" rather than "planes can"
   - Empirical tests for determining constituents
     e.g. *substitution, permutation, coordination*

Introduction

What is a grammar?

- A grammar is a description of the valid structures of a language.
- Formal grammars are one of the most central concepts of linguistics. (e.g. detection, generation)

Formal grammars

- A formal grammar is defined by a set of rules that consist of terminal and non-terminal symbols.
- Terminal symbols ($\approx$ words) cannot be rewritten any further.
- Non-terminals express clusters or generalizations of terminals.

Grammar (Σ, N, S, R)

Σ An alphabet (i.e., a finite set of terminal symbols).

N A finite set of non-terminal symbols.

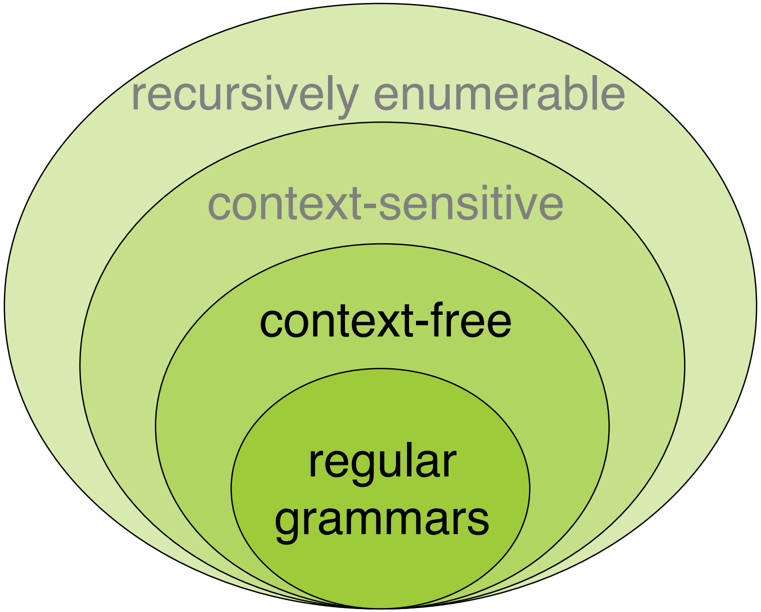
S A start non-terminal symbol, $S \in N$.

R A finite set of production rules, $R \subseteq (\Sigma \cup N)^+ \setminus \Sigma^* \times (\Sigma \cup N)^*$.

Introduction

Chomsky Grammars: Four types of formal grammars

- Chomsky-0 (recursively enumerable). Any (Σ, N, S, R) as defined.
- Chomsky-1 (context-sensitive). Only rules $U \to V$ with $|U| \leq |V|$.
- Chomsky-2 (context-free). Only rules $U \to V$ with $U \in N$.
- Chomsky-3 (regular). Only rules $U \to V$ with $U \in N$ and $V \in \{\varepsilon, v, vW\}$, $v \in \Sigma$, $W \in N$.



In NLP most commonly used are regular and context-free grammars.

Introduction

Chomsky-1 (context-sensitive).: Only rules $U \to V$ with $|U| \leq |V|$.

Example: $xAy \to xvy; A \in N, \;\; x, y \in \Sigma^*, v \in \Sigma^+$

Description: Replace A by v in the Context x, y

Example 1: A context-sensitive grammar can be used to derive the set $\{a^n b^n c^n\}$.

$$
\begin{aligned}
S &\to abc/aAbc \\
Ab &\to bA \\
Ac &\to Bbcc \\
bB &\to Bb \\
aB &\to aa/aaA
\end{aligned}
$$

$$
\begin{aligned}
S &\to aAbc \\
&\to abAc \\
&\to abBbcc \\
&\to aBbbcc \\
&\to aaAbbcc \\
&\to aabAbcc \\
&\to aabbAcc \\
&\to aabbBbccc \\
&\to aabBbbccc \\
&\to aaBbbbccc \\
&\to aaabbbccc
\end{aligned}
$$

Introduction

Chomsky-2 (context-free). Only rules $U \to V$ with $U \in N$.

Example: $A \to v; A \in N, v \in \Sigma^*$

Example 1: A context-free grammar can be used to derive the set $\{a^n b^n\}$.

$N = \{S, X\}, \Sigma = \{a, b\}$

$S \to ab$

$S \to aXb$

$X \to ab$

$X \to aXb$

Example 2: A context-free grammar can be used to derive the set $\{a^n b^k c^n\}$.

$N = \{S, X, Y\}, \Sigma = \{a, b, c\}$

$S \to ac$

$S \to aXc$

$X \to aXc$

$X \to aYc$

$Y \to Yb$

$Y \to b$

Introduction

Regular grammars in NLP

- Regular grammars are particularly useful in inferring information whose language follows clear sequential patterns.
- To this end, texts are matched against regular expressions (more later)
- Tasks. Numeric entity recognition, extraction of structured entities (e.g., email addresses), lexico-syntactic relations (e.g., “<NN> is a <NN>”), ...

Parsing linguistic units with regular grammars

- Numeric (and alphanumeric) entities
  - Values, quantities, proportions, ranges, or similar.
  - E.g. time and date expressions, phone numbers, monetary values, ...

    “9:15 am”, “2018-10-18” “$ 100 000” “60-68 44”

- Numeric entity recognition
  - Text analysis that mines numeric entities from text.
  - Used in many information extraction tasks.

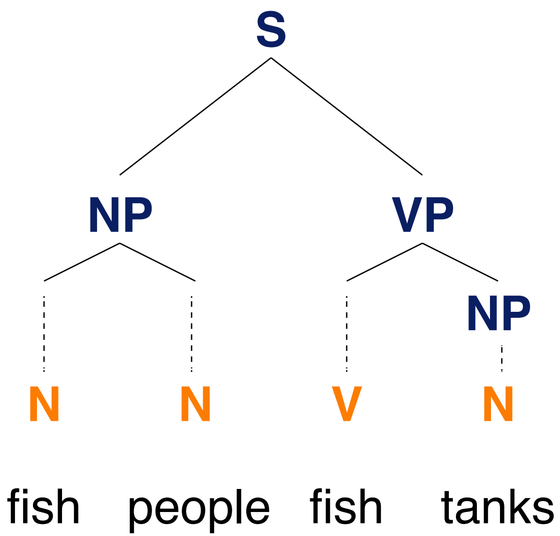
Introduction

Context-free grammars (CFGs) in NLP

- CFGs are particularly useful for hierarchical structures of language.
- Probabilistic extensions (PCFGs) capture the likeliness of structures.
- CFGs usually define the basis of syntactic parsing.

Syntactic parsing

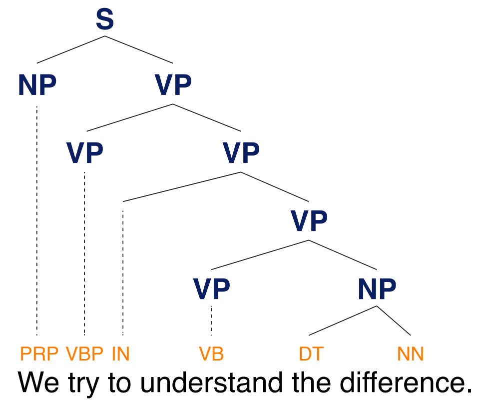
- Text analysis that determines the syntactic structure of a sentence.
- Used in NLP as a standard processing step, e.g., as preprocessing for tasks such as relation extraction.



Introduction

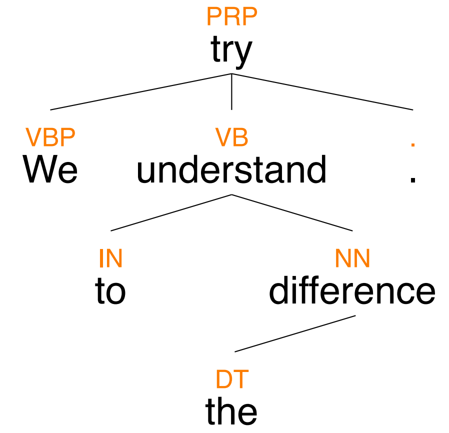
Constituency

- A constituent is a word or group of words that function as a single unit within a hierarchical structure.
- Constituent structure models the constituents (phrases) of a sentence and how they are composed of each other.
- Constituency parsing infers the phrase structure of a sentence.



Dependency

- Dependency models the dependencies between the words in a sentence.
- Dependency parsing is based on a dependency grammar, a special case of CFGs where relations are modeled directly between words; the root is nearly always the main verb (of the main clause).



Introduction

Constituency What is the right structure?

- Which elements?
- How Combined?

Tests for constituents: What is the right structure?

Proform substitution: What can be pronominalized (what can be referred to with a proform) is a constituent.

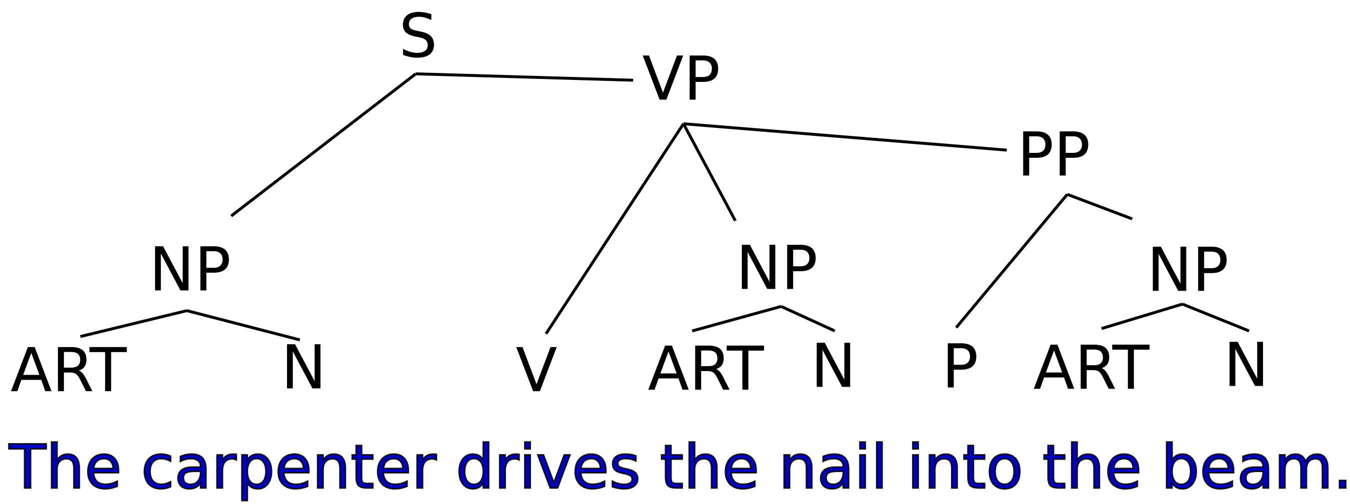
- *The streetcar* **(it)** was stopped *by the collision with a car* **(therethrough)** *on the intersection of Augustusplatz* **(there)**.
- Bill wants to eat a pudding. I want that, too.

[. . .] Removal test, Answer fragments, Topicalization, Coordination

Wikipedia article on constituents

Introduction

Constituency What is the right structure?



Production rules (spoiler):

$S \rightarrow NP\,VP$, $NP \rightarrow ART\ N$

$VP \rightarrow V\ NP\ PP$, $PP \rightarrow P\ NP$

Introduction

Phrase Structure Grammar (PSG) Problems

1. Free word order
2. Thematic roles
3. Congruence and Rection

To (1) and (2): traditional approach: transformation grammar alternatively: dependency grammar

To (3): representation of morphological dependencies (head principle) by a unification grammar (HPSG).

Free word order – permissible permutations in German

Der Bayer stellt die Maß auf den Tisch. / Auf den Tisch stellt der Beyer die Maß!

Whom does Kate love? / Kate loves whom?

Introduction

Traditional Approach: **Transformation**

Base Structure (PSG) $\rightarrow$ *Transformation* $\rightarrow$ **Text Surface**

Transformation is realized by – **Morphology, Syntax**

[The(1) sun(2)]NP [is(3) shining(4) [in(5) Leipzig(6)]PP]VP

Structural Change

- Focus Location: 5, 6, 1, 2, 3, 4
- Question: 3, 1, 2, 4, 5, 6

Theoretical Foundation: Chomsky, Syntactic Structures 1957 - Phrase Structure

Transformations extend PSG (context-free) to an unrestricted language (Turing-equivalent) and are therefore not pursued further in linguistics

The concept of transformations has proven useful in the development of natural language dialog interfaces (e.g., Eliza)

Masswerk Eliza Demo

Introduction

Eliza

`My (1) is (2)`

My <u>father</u> is <u>mad</u>

`How log has your (1) been (2)`

Idea: Transforming inputs into outputs by using input patterns (patterns) and simple morphological transformations.

Pascal Version

- Search by keyword: literal patterns
- Morphological transformations (conjugates): literal patterns
- Composing the answer: by type of open patterns

Introduction

Example Dialogue

Hi! I’m Eliza. I am your personal therapy computer. Please tell me your problem.

```
> I don’t like ASV.
❑ Don’t you really like ASV?
> Yes, it is horrible.
❑ Are you Sure?
> Yes, yes. It is so much work.
❑ I see.
> Well, what do you recommend me to do?
❑ We were discussing you - not me.
> Stupid machine.
❑ How do you...please state your problem.
> Do you see how difficult it is to programme machines to behave
  intelligently?
❑ Oh, see how difficult it is to programme machines to behave
  intelligently?
```

Eliza – Algorithm

Infinite Loop:

- Read input
- Process input
  - Delete punctuation, trim string `Ctrim`
  - transform to upper case `UpCopy`
- Find keyword `FindKey`
- Conjugate string `Conjugate`
- Determine response pattern and insert conjugated string `GetResponse`
- Return Answer

Approx 270 lines of code!

Introduction

Pattern Matching

T. Winograd: "Language as a cognitive process" [Winograd, '82]

- Literal patterns (concrete strings): e.g. `Leipzig, Amazon Inc.`
- open patterns (Wildcards): e.g. "X threatens Y", X was acquired by Y
- Lexical patterns: `N threatens PP, ENTITY was acquired by ENTITY`
- Variable patterns (Same variables .. Same word): `War of the Roses: X threatens X`
- Sentence Structure Patterns: `How much does a ticket from X to Y n cost on a (Mon,Tue,Wed,Thu,Fry,Sat,Sun)?`

Extreme point of view: syntactic or semantic parsers considered as pattern matchers.

Example Implementation: (A)rtificial (I)ntelligence (M)arkup (L)anguage – Interpreters for Java, Node, Python etc. AIML Foundation

Remarks:

- What is the right complexity for natural language?
  - Recursively enumerable? OK but too little structure or restriction, therefore too little explanatory value
  - Regular? Too weak due to recursive structures; Since we have right-linear rules max complexity is $\{a^n b^m\}$

- Context-Free vs. Context-sensitive:
  - Peter resp. Maria resp. Paul hiking resp. swimming resp. cycling to Berlin resp. at the Baltic Sea resp. in Leipzig $\{a^n b^n c^n\}$

- Should we really define patters or should we learn it from data?
  - Today Machine Learning is key to model dialogue by data. – **But we still use morphological and syntactical features**
  - We are using PSG to detect language – Generation is only possible for simple genres. More complex generation is hard with Transformation and HPSG.
  - Language generation had breakthroughs by using a sequence model paradigm – See language models