Introduction to Computational Linguistics

PD Dr. Frank Richter

fr@sfs.uni-tuebingen.de.

Seminar für Sprachwissenschaft

Eberhard-Karls-Universität Tübingen

Germany

Incremental Linguistic Analysis

- tokenization
- morphological analysis (lemmatization)
- part-of-speech tagging
- named-entity recognition
- partial chunk parsing
- full syntactic parsing
- semantic and discourse processing

Potential Tasks

- Tokenize arbitrary text
- Subtask: Recognize date expressions
- Assign correct suffixes respecting vowel harmony
- Given an inflected verb: Find a base form of verbs and their agreement features
- Given a a base form of verbs and their agreement features: find the appropriate inflected form
- Morphology: derivation: English verbs + suffix *-able* (yields an adjective: desirable, printable, readable, etc.)
- Assign syntactic categories to tokens in preprocessed text
- Bracketing of syntactic chunks in arbitrary text

Formal Languages & Computation

The language perspective

1. Type 3: regular expression languages
2. Type 2: context free languages
3. Type 1: context sensitive languages
4. Type 0: recursively enumerable languages

Formal Languages & Computation

The language perspective

1. Type 3: regular expression languages
2. Type 2: context free languages
3. Type 1: context sensitive languages
4. Type 0: recursively enumerable languages

The automata perspective

1. Finite automata
2. Pushdown automata
3. Linear automata (Turing machines with finite tapes)
4. Turing machines

Form of Grammars of Type 0–3

For $i \in \{0, 1, 2, 3\}$, a grammar $\langle N, T, \Pi, s \rangle$ of Type i, with N the set of non-terminal symbols, T the set of terminal symbols (N and T disjoint, $\Sigma = N \cup T$), Π the set of productions, and s the start symbol ($s \in N$), obeys the following restrictions:

Type 3: Every production in Π is of the form $A \rightarrow aB$ or $A \rightarrow \epsilon$, with $B, A \in N$, $a \in T$.

Type 2: Every production in Π is of the form $A \rightarrow x$, with $A \in N$ and $x \in \Sigma^*$.

Type 1: Every production in Π is of the form $x_1 A x_2 \rightarrow x_1 y x_2$, with $x_1, x_2 \in \Sigma^*$, $y \in \Sigma^+$, $A \in N$ and the possible exception of $C \rightarrow \epsilon$ in case C does not occur on the righthand side of a rule in Π.

Type 0: No restrictions.

An Example of a Type 2 Grammar

Let $\langle N, T, \Pi, S \rangle$ be a grammar with N, T and Π as given below:

- $N = \{S, NP, VP, V\}$
- $T = \{\text{John}, \text{walks}\}$
- $\Pi = \{S \rightarrow NP\ VP, NP \rightarrow \text{John}, VP \rightarrow V, V \rightarrow \text{walks}\}$

Finite State Technology

Regular languages and finite state automata

- deterministic finite state automata,

Finite State Technology

Regular languages and finite state automata

- deterministic finite state automata,
- nondeterministic finite state automata,

Finite State Technology

Regular languages and finite state automata

- deterministic finite state automata,
- nondeterministic finite state automata,
- finite state automata, and

Finite State Technology

Regular languages and finite state automata

- deterministic finite state automata,
- nondeterministic finite state automata,
- finite state automata, and
- regular expressions

Finite State Technology

Regular languages and finite state automata

- deterministic finite state automata,
- nondeterministic finite state automata,
- finite state automata, and
- regular expressions

characterize the same class of languages, *viz.* Type 3 languages