Natural Language Processing
Lecture 01 Introduction; Grammars; Morphology; Word Frequency; Collocations

Qun Liu, Valentin Malykh
Huawei Noah's Ark Lab

Spring 2020
A course delivered at MIPT, Moscow

Content

Content

Logistics

- Instructors: Prof. Qun Liu, Valentin Malykh
- Time: 18.30 each Thursday
- Location: Klimentovsky lane, 1 bld. 1, auditorium 308
- Slides: will be uploaded to the course website before each class.
- Mailing List:
- Website:

Course description

Natural Language Processing (NLP) is a domain of research whose objective is to analyze and understand human languages and develop technologies to enable human machine interactions with natural languages. NLP is an interdisciplinary field involving linguistics, computer sciences and artificial intelligence. The goal of this course is to provide students with comprehensive knowledge of NLP. Students will be equipped with the principles and theories of NLP, as well as various NLP technologies, including rule-based, statistical and neural network ones. After this course, students will be able to conduct NLP research and develop state-of-the-art NLP systems.

Grading policy

Assignments

Projects

Examinations

Content

Natural language processing in Wikipedia

Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.

Synonyms of NLP

- **Computational Linguistics**
- **Natural Language Processing**
- **Natural Language Understanding**
- **Human Language Processing**

- Subtleties

  Computational Linguistics is more regarded as a branch of Linguistics, whose main purpose is to understand the mechanism of human languages by means of computing

Synonyms of NLP

- **Computational Linguistics**
- **Natural Language Processing**
- **Natural Language Understanding**
- **Human Language Processing**

- Subtleties

  Natural Language Processing is a branch of computer sciences and artificial intelligent, whose main purpose is to develop technologies to enable human-computer interactions using human languages

Synonyms of NLP

- **Computational Linguistics**
- **Natural Language Processing**
- **Natural Language Understanding**
- **Human Language Processing**

- Subtleties

  Natural Language Understanding is one of the two main challenges in Natural Language Processing, while the other is **Natural Language Generation.**

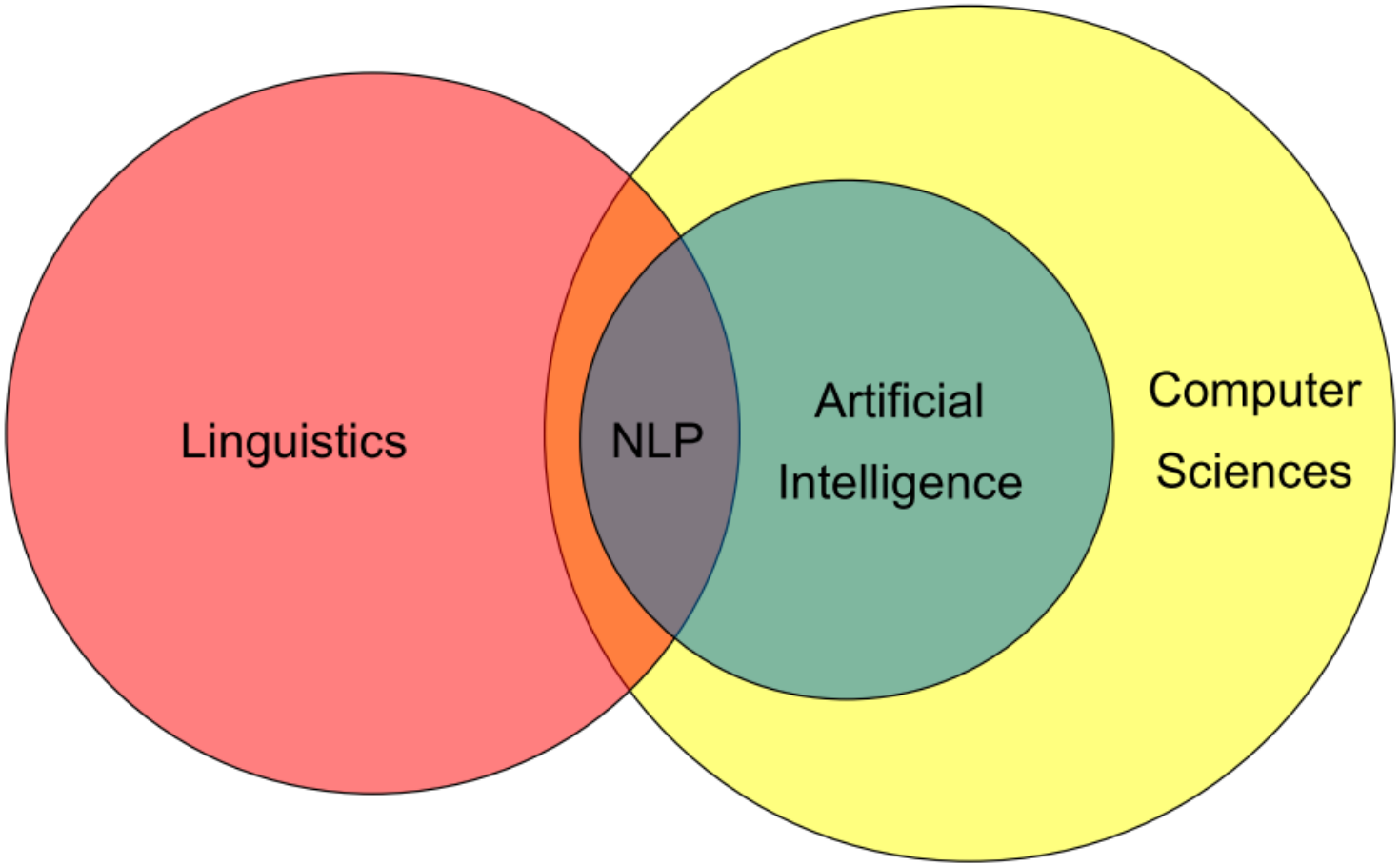
Synonyms of NLP

- **Computational Linguistics**
- **Natural Language Processing**
- **Natural Language Understanding**
- **Human Language Processing**

- Subtleties

  Human Language Technologies mainly refer to NLP technologies, but may also include other language related technologies, include speech technologies, optical character recognition (OCR), computer typesetting, etc.

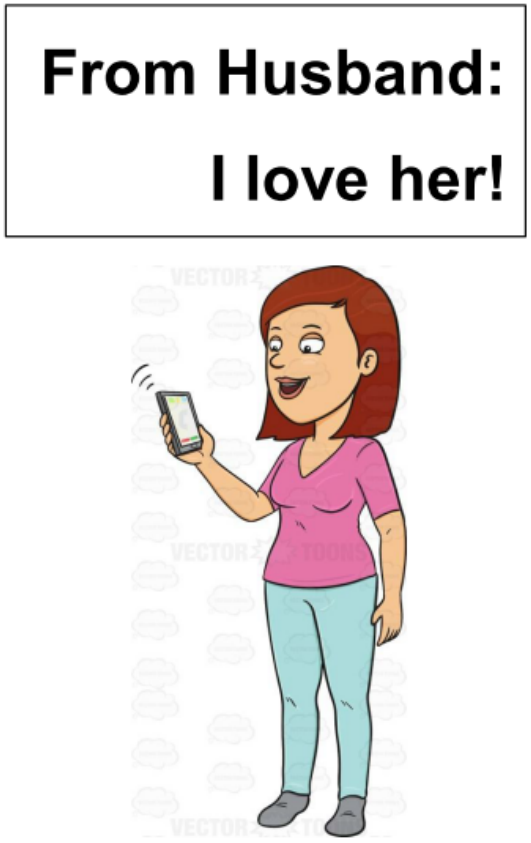
NLP as an interdisciplinary study



Understanding human languages is not easy

- We are getting used to the fact that human beings can understand each other using language communication.
- Although it is a natural result of evolution for human to obtain the language competence.
- It seems to be a miracle, due to its complexity.
- No other species in this planet can use languages at the degree as humans do.
- The mechanism behind human languages is not fully discovered.
- Understanding human languages by computer is difficult.

Understanding human languages is not easy



Research questions

- How humans understand each other by using language communication?
- Is it possible to simulate human language behaviors without understanding language mechanisms?

The way of NLP research

- Unlike linguists who develop numerous theories to explain the language mechanisms, NLP researchers try to simulate human language behaviors by computing, not necessary to understand the language mechanisms.

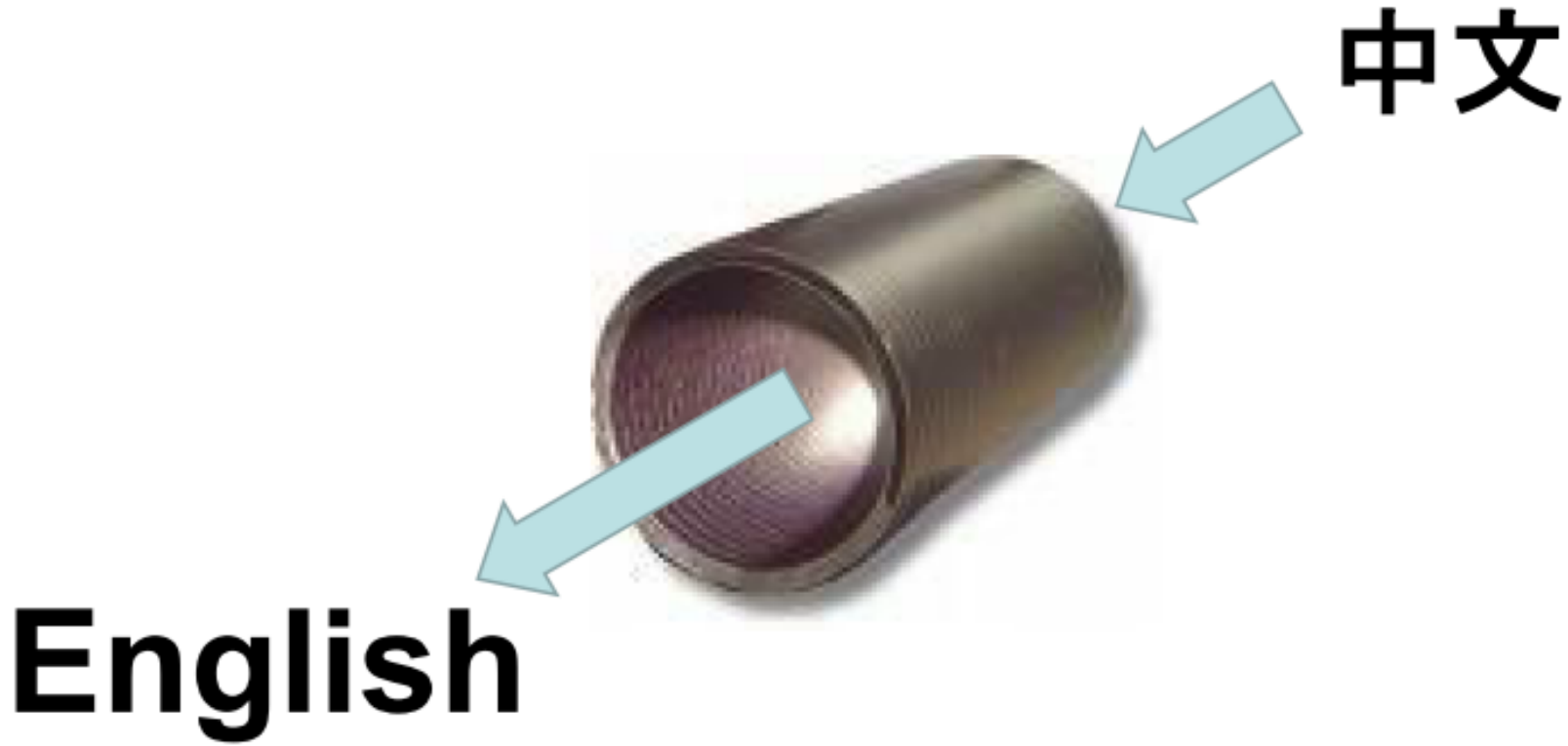
A brief history of NLP

- 1960s-1990s: Rule-based approaches
- 1990s-2010s: Statistical approaches
- 2010s-present: Neural network (deep learning) approaches

Holy grails of NLP

- Accurate machine translation between human languages
- Free conversation between humans and computers

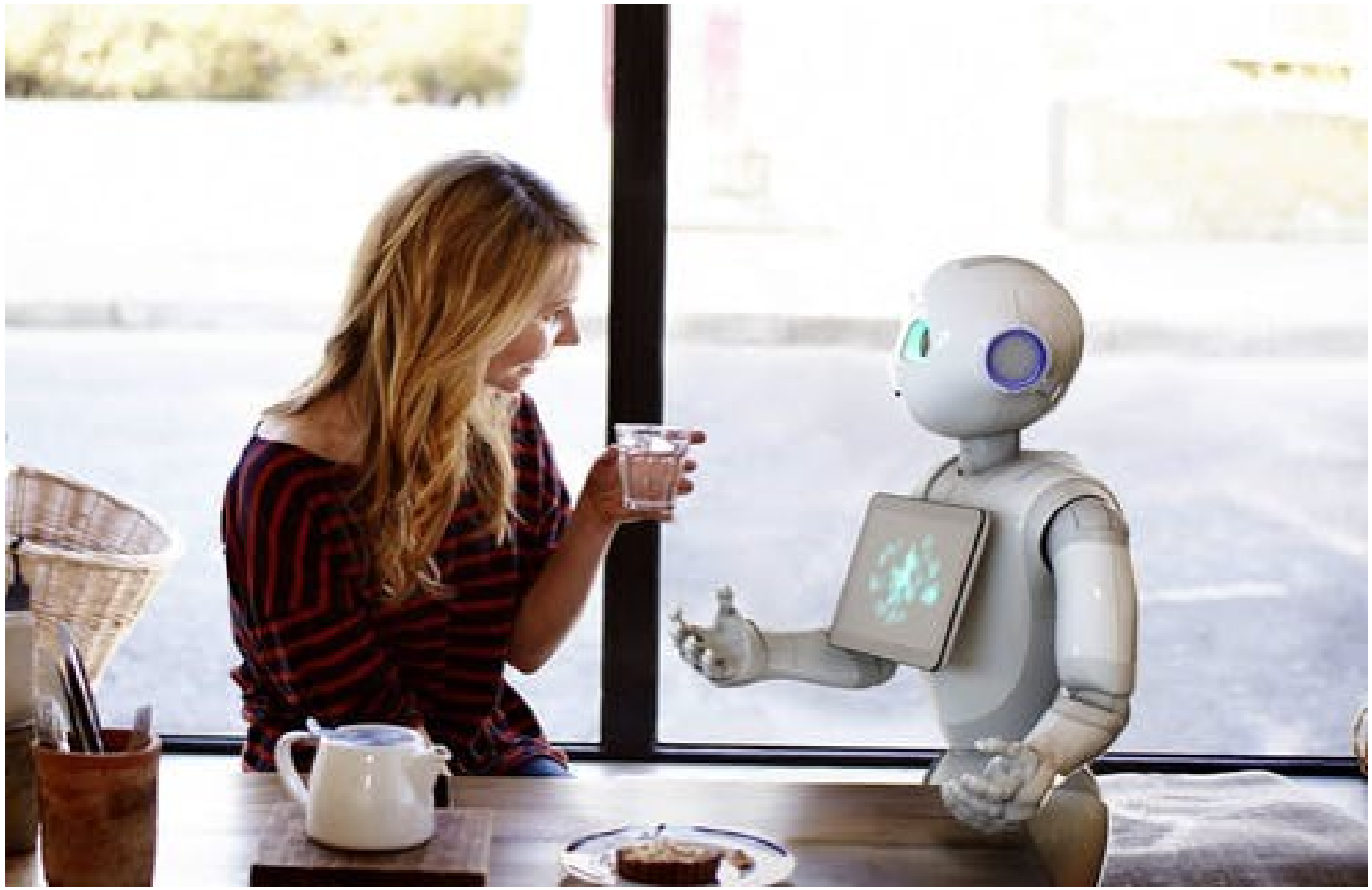
Accurate machine translation



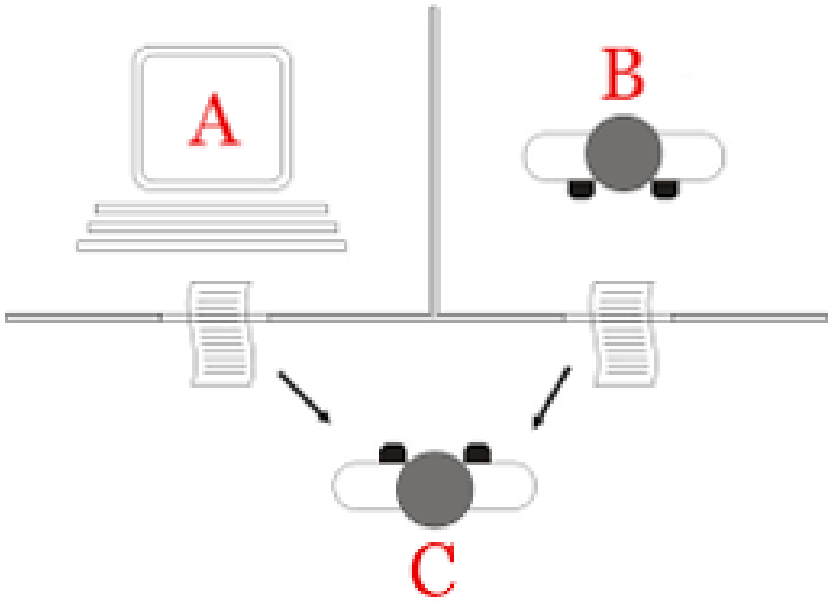
The Tower of Babel

Oil painting by Pieter Bruegel the Elder, 1563, from Wikipedia

Free human machine conversation



Turing test

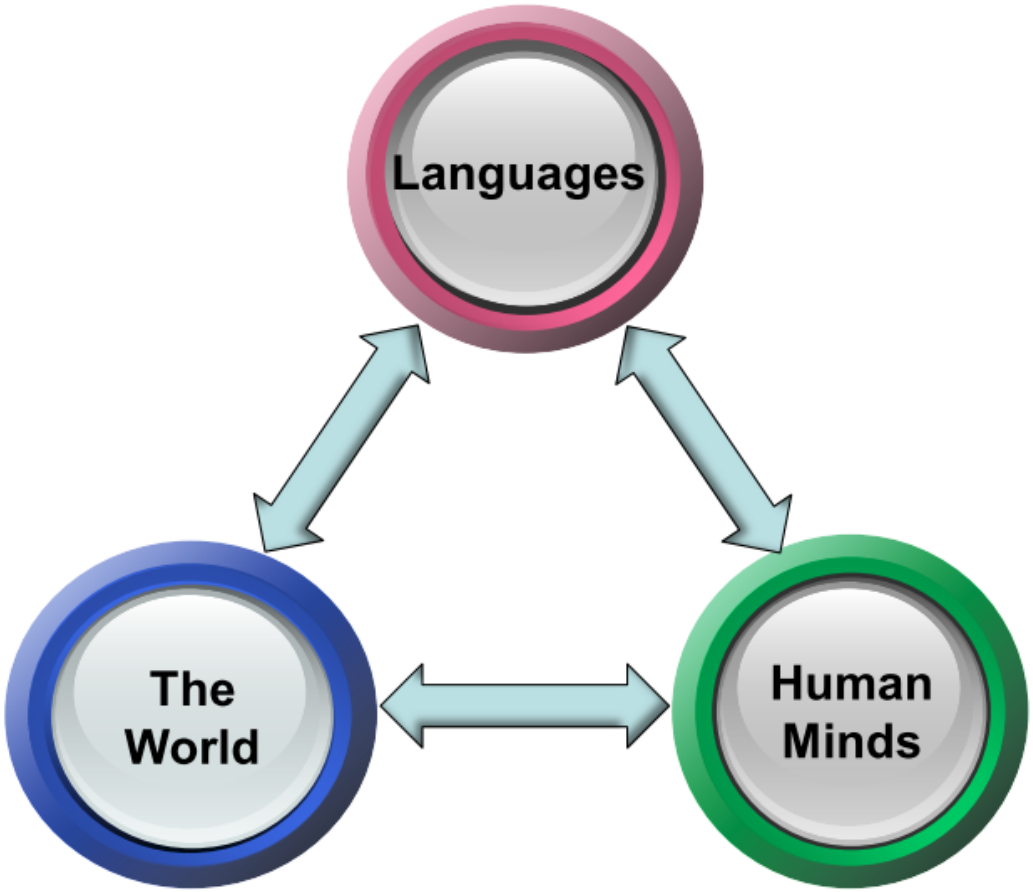


By Juan Alberto Sánchez Margallo, CC BY 2.5, from Wikipedia

NLP Tasks

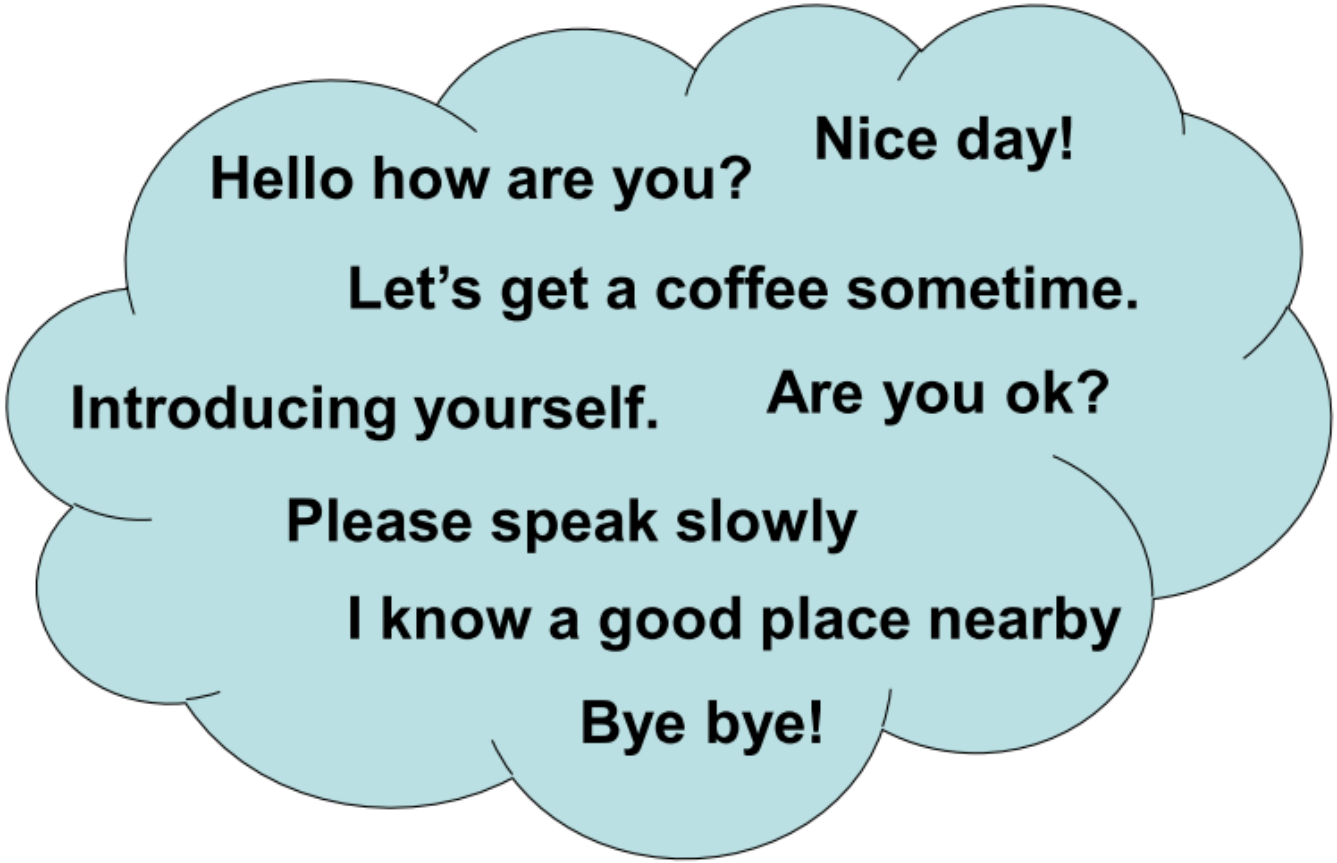
	Word / Phrase	**Sentence**	**Document**
Features / Expressions	word frequency, colocations, one-hot vectors, word embeddings	sentence embeddings, language models	bag of word / n-grams, word frequency vectors, tf-idfs, key words / phrases extraction, topic distributions, document embeddings
Classification	part-of-speech tagging, word sense disambiguation	sentiment analysis	text classification, sentiment analysis
Sequence labeling / Segmentation	stemming	word segmentation, part-of-speech tagging, named entity recognition	sentence segmentation, coreference resolution
Structural prediction / Parsing	morphological analysis	constituent parsing, dependency parsing, semantic role labeling, semantic parsing	discourse parsing
Sequence Generation /	machine translation, text summarization, dialog, style transfer, question answering, machine reading comprehension		

Languages, human minds and the world



Content

How can we define a language?



How can we define a language?

- A language can be defined as the set of sentences which can be accepted by the speakers of that language.
- It is not possible to define a natural language by enumerate all the sentences, because the number of sentences in a natural languages is infinite.
- Two feasible ways to define a language with infinite sentences:
  - By a Grammar
  - By an Automaton

Define a language by a grammar

- A grammar G is defined as:
  - A finite set of rules, and,
  - A mechanism to generate word sequences by applying the rules in G in a finite number of time steps.
- A sentence of a language is defined as:
  - A word sequence S is called a sentence of a language L if and only if S belongs to L.
- Given a grammar G, a language L could be defined by G as:
  - A word sequence S is a sentence of L if and only if S can be generated by G.

Define a language by an automaton (1)

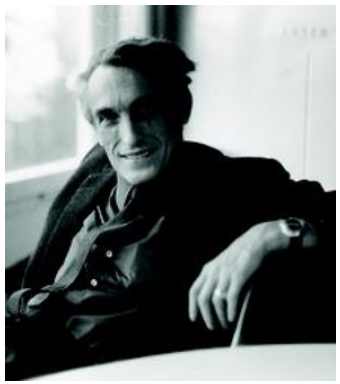
- An automaton A is a abstract machine which:
  - Takes a symbol sequence S as input, and determines if A will *accept* or *reject* S.
  - Has a finite number of states and a finite number of actions.
  - At each time step, S is in a state, and points to a position in S.
  - The current state and current symbol determines the action which A will execute, which determines the next state of A and the next position of S where A will point to.
  - Given a input S, A will run until it stops, and the final state of A determines if A will *accept* or *reject* S.

Define a language by an automaton (2)

- A language L can be defined by an automaton A as:
  - A word sequence S is a sentence of L, if and only if: when we input S to A, A will stop in a finite number of time steps at an *accept* state.

Chomsky hierarchy of formal grammars

- The Chomsky hierarchy (occasionally referred to as the Chomsky–Schützenberger hierarchy) is a nested hierarchy of classes of formal grammars.
- This hierarchy of grammars was described by Noam Chomsky in 1956.
- It is also named after Marcel-Paul Schützenberger, who played a crucial role in the development of the theory of formal languages.

Wikipedia - Chomsky hierarchy

Formal grammars

- A formal grammar G is a quadruple $\{N, T, S, R\}$:
  - R: a finite set of production rules ($\text{left-hand side} \rightarrow \text{right-hand side}$, or $\text{LHS} \rightarrow \text{RHS}$), where each side consists of a finite sequence of the symbols from N, T or $\{S\}$
  - N: a finite set of nonterminal symbols (indicating that some production rule can yet be applied)
  - T: a finite set of terminal symbols (indicating that no production rule can be applied)
  - S: a start symbol (a distinguished nonterminal symbol)

Formal grammars and formal languages

- A *formal grammar G* provides an axiom schema for a *formal language L*, which is a set of finite-length sequences of symbols that may be constructed by applying *production rules* to another sequence of symbols, which initially contains just the *start symbol*.
- A *production rule* may be applied by replacing an occurrence of the symbols on its left-hand side (LHS) with those that appear on its right-hand side (RHS).

Formal grammars and formal languages

- A sequence of rule applications is called a *derivation*.
- A formal grammar G defines a formal language L: all sequences of symbols consisting solely of terminal symbols which can be reached by a derivation from the start symbol.

An Example

The language $\{a^n b^n\}$ (i.e. n copies of a followed by n copies of b) can be defined by the following grammars:

Terminals: $\{a, b\}$
Nonterminals:$\{S, A, B\}$
Rules:

$$S \rightarrow AB$$
$$S \rightarrow \epsilon$$
$$S \rightarrow aS$$
$$S \rightarrow b$$

Terminals: $\{a, b\}$
Nonterminals:$\{S\}$
Rules:

$$S \rightarrow aSb$$
$$S \rightarrow \epsilon$$

ϵ is an empty string

Chomsky hierarchy of grammars

Grammar	Languages	Production Rules	Examples
Type 0	Recursively Enumerable	$\alpha A \beta \rightarrow \delta$	
Type 1	Context Sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$	$L = \{ a^n b^n c^n \mid n > 0 \}$
Type 2	Context Free	$A \rightarrow \alpha$	$L = \{ a^n b^n \mid n > 0 \}$
Type 3	Regular	$A \rightarrow a$ or $A \rightarrow aB$	$L = \{ a^n \mid n > 0 \}$

Type 0 grammars

- Also called Unrestricted Phrase Structure Grammars.
- A Type 0 Grammar generates a Recursively Enumerable Language.
- A Recursively Enumerable Language L is semi-decidable by a Turing Machine T:
  - For any sentence S in L, T will accept it in a finite number of time steps.
  - For a sentence S not in L, it is not guaranteed that T can reject it in a finite number of time steps.

Type 1 grammars

- Also called Context Sensitive Grammar.
- A Type 1 Grammar generates a Context Sensitive Language.
- A Context Sensitive Language is decidable by a Linear Bounded Automaton and the complexity of this decision problem is NP-complete.
- It is not practical to use Type 1 Grammars in NLP because of its time complexity.

Type 2 grammars

- Also called Context Free Grammars.
- A Type 2 Grammar generates a Context Free Language.
- A Context Free Language is decidable by a Pushdown Automaton and the complexity of this decision problem is polynomial.
- Type 2 Grammars are the theoretical basis of all programming languages.
- Type 2 Grammars are commonly used in NLP, however, natural languages are not context free languages actually.

Type 3 grammars

- Also called Regular Grammars.
- A Type 3 Grammar generates a Regular Language.
- A Context Free Language is decidable by a Finite State Automaton / Machine and the complexity of this decision problem is linear.
- Type 3 Grammars are the theoretical basis of the lexical analyzers of all programming languages.
- Type 3 Grammars are very broadly used in NLP for many different purposes.

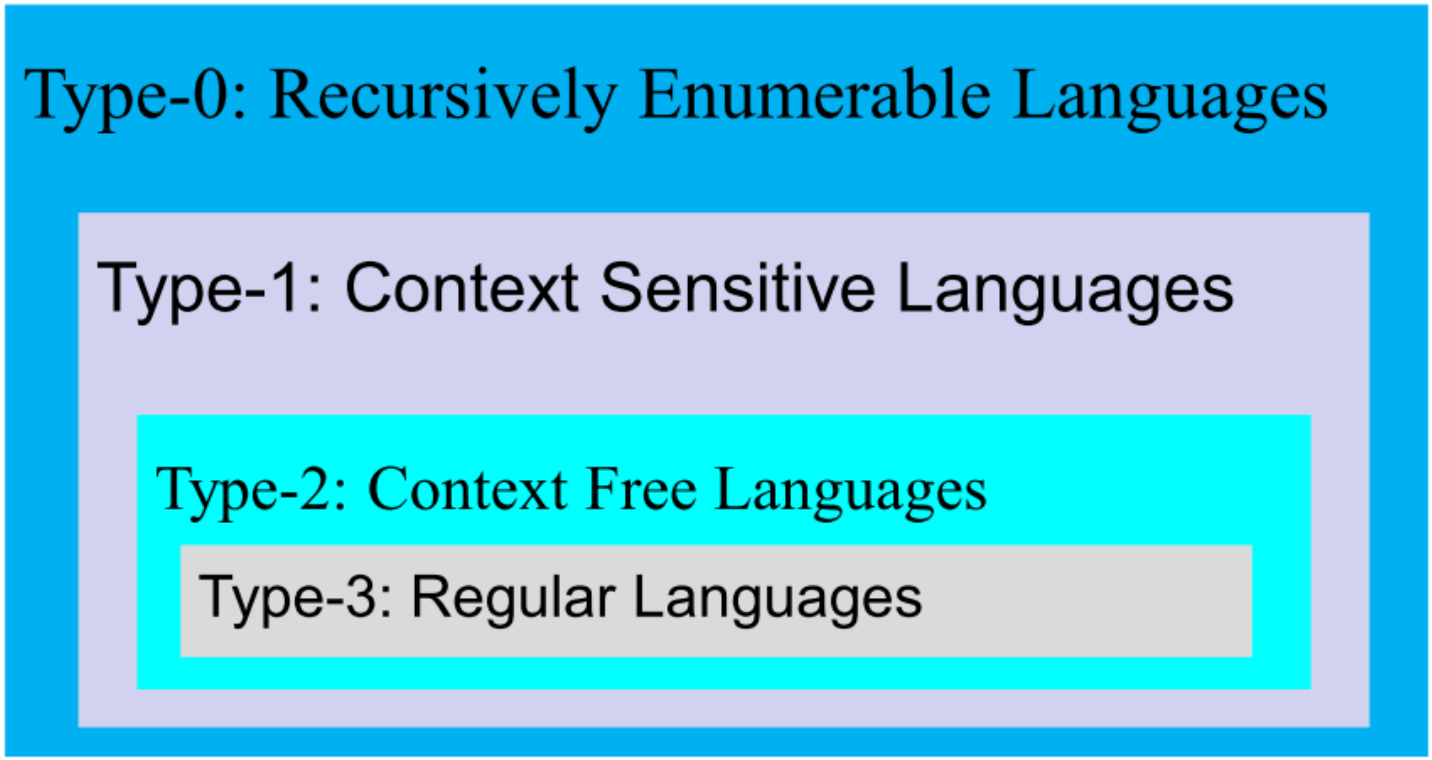
Regular expressions

- Regular Expressions are very useful tools which are supported by most of the modern programming languages and text editors.
- A Regular Expression is equivalent to a Regular Grammar, and vice versa.

Regular expressions in Python

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

Chomsky hierarchy



Set inclusions described by the Chomsky hierarchy

Other grammar classes

- Mild Context Sensitive Grammars:
  - Grmmars classes which define subsets of Context Sensitive Languages, but beyond Context Free Languages.
  - Examples:
    - Index Grammars (IGs)
    - Tree Adjoin Grammars (TAGs)

Grammars and automata

Grammar	Languages	Automaton	Decidability and Complexity
Type 0	Recursively Enumerable	Turing Machine	Semi-Decidable
	Recursive	Decider, or Total Turing Machine	Decidable
Type 1	Context Sensitive	Linear Bounded Automaton (LBA)	NP Complete
Type 2	Context Free	Pushdown Automaton (PDA)	Polynomial
	Deterministic Context Free	Deterministic Pushdown Automaton (PDA)	Linear
Type 3	Regular	Deterministic / Nondeterministic Finite State Machine (FSM)	Linear

Turing machine