Language Technology (2023)

Course introduction

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

This session

- What is language technology?
- Course organisation and examination
- Text segmentation

What is language technology?

What is language technology?

- **Language technology** is technology for the analysis and interpretation of natural language.

  not programming languages!

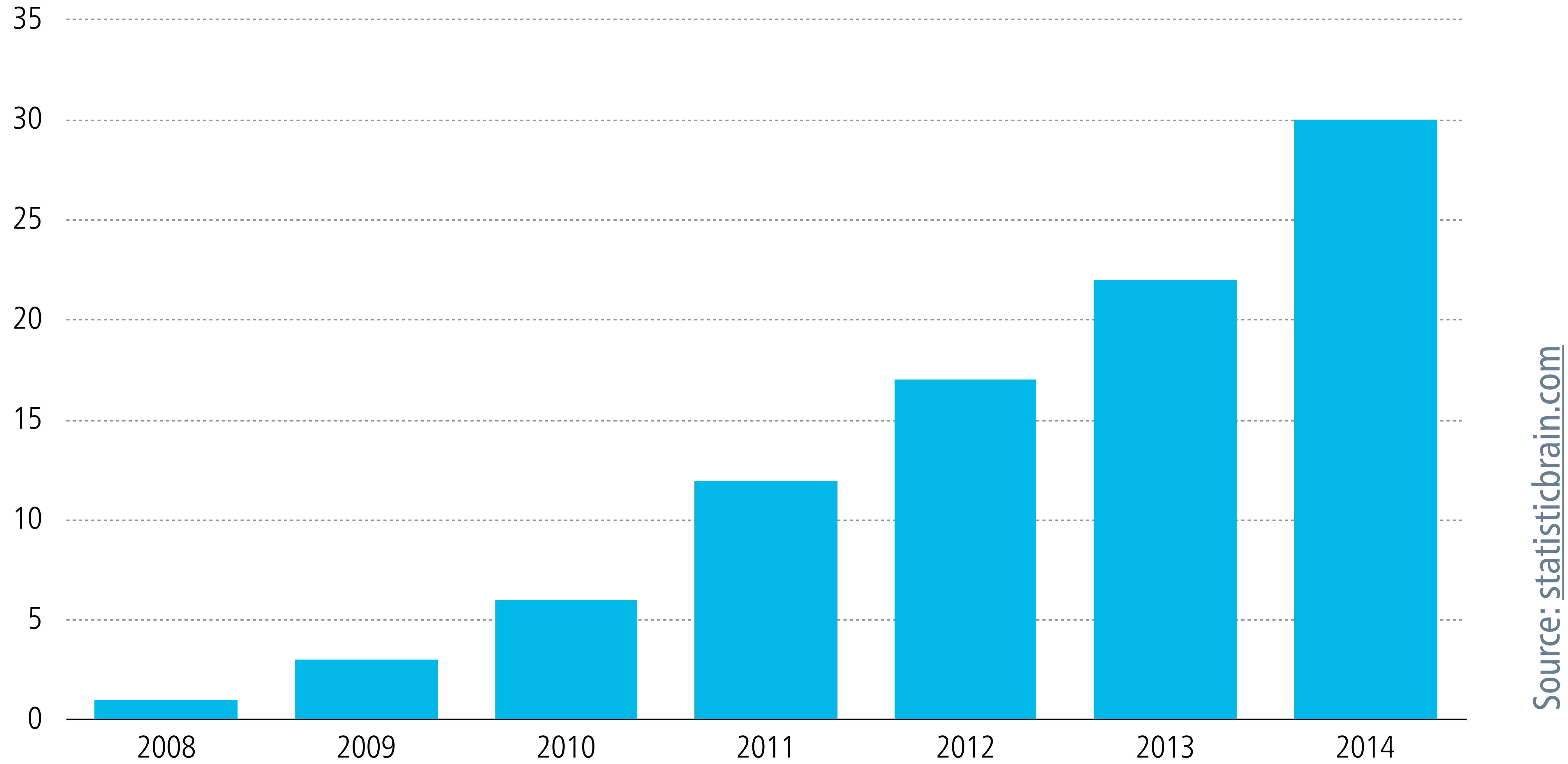
- Language technology is an interdisciplinary research area involving computer science, linguistics, and cognitive science.

  related names: natural language processing, computational linguistics

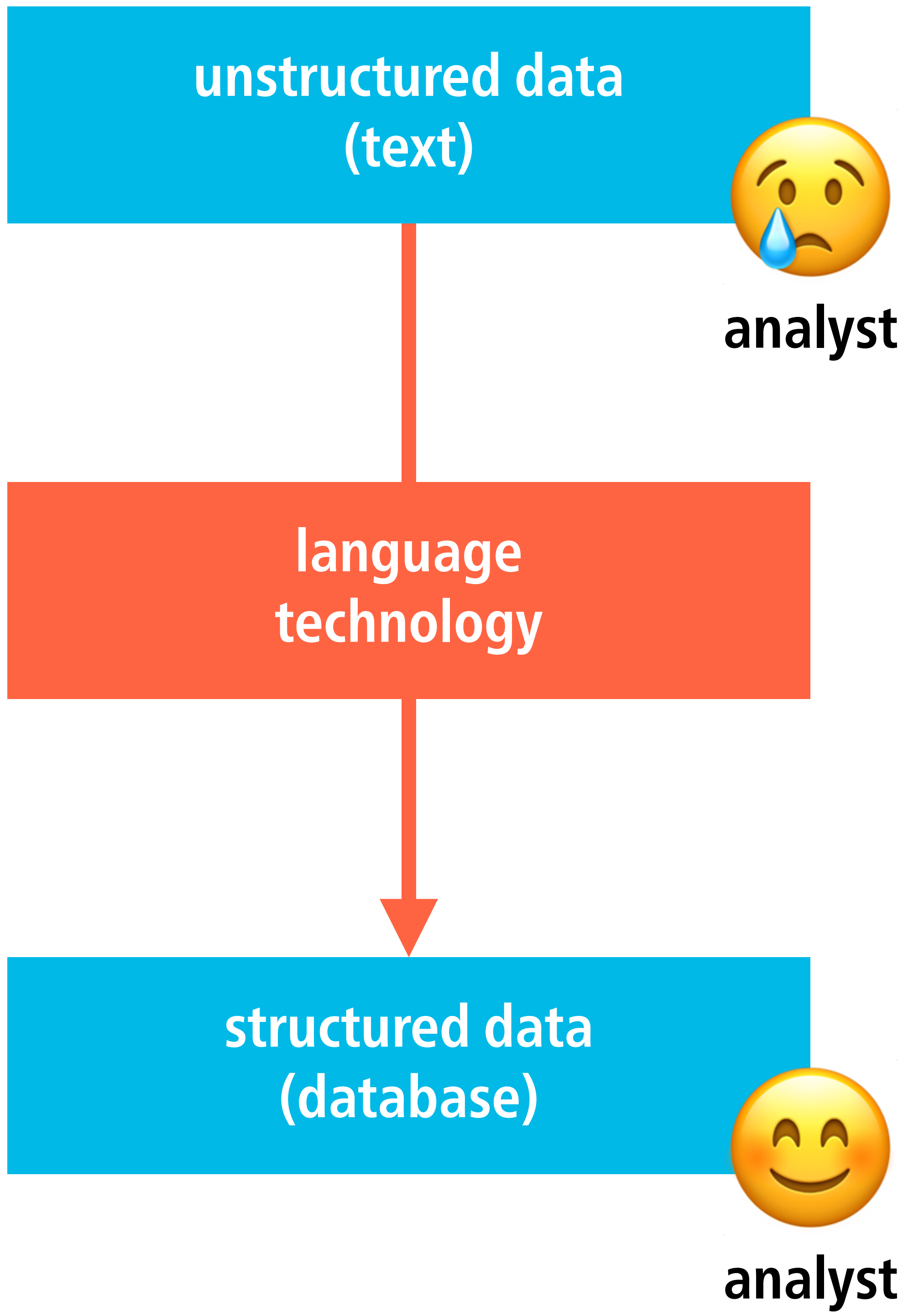
‘We are drowning in information
but starved for knowledge.’

John Naisbitt (1982)

Total number of pages indexed by Google

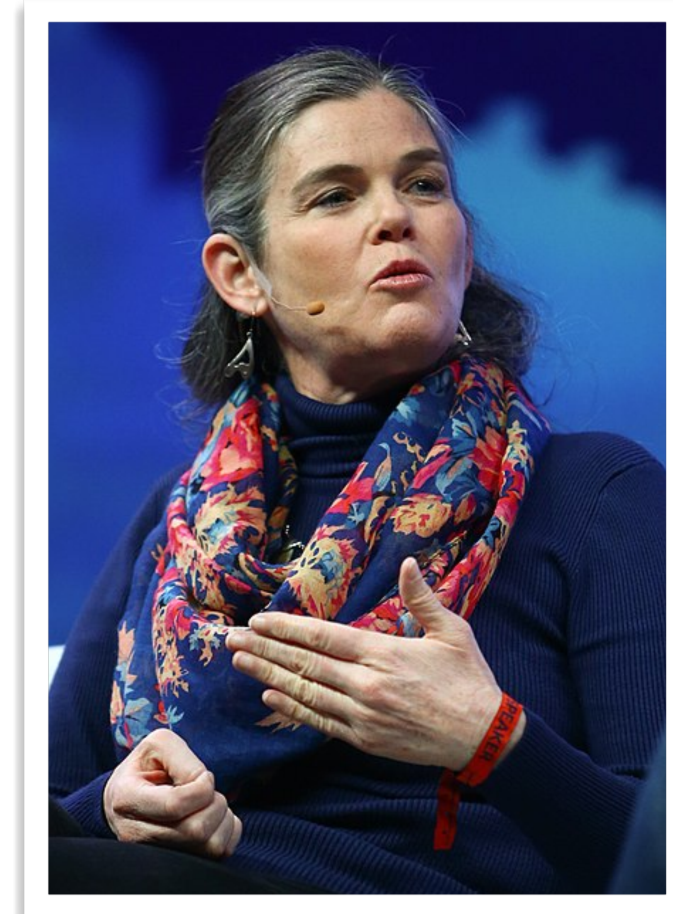


The Knowledge Gap



JEOPARDY!

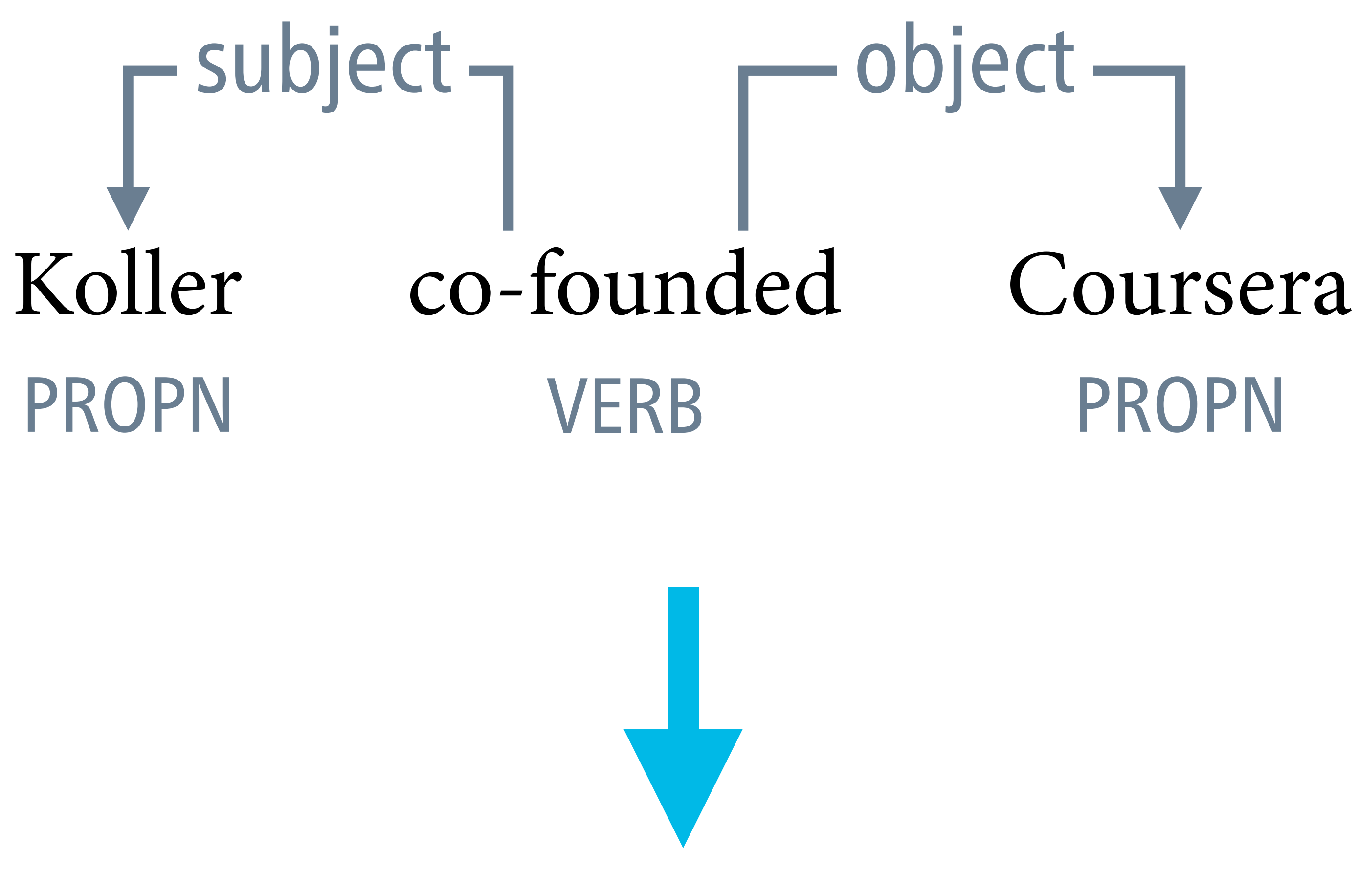
This Stanford University alumna co-founded educational technology company Coursera.



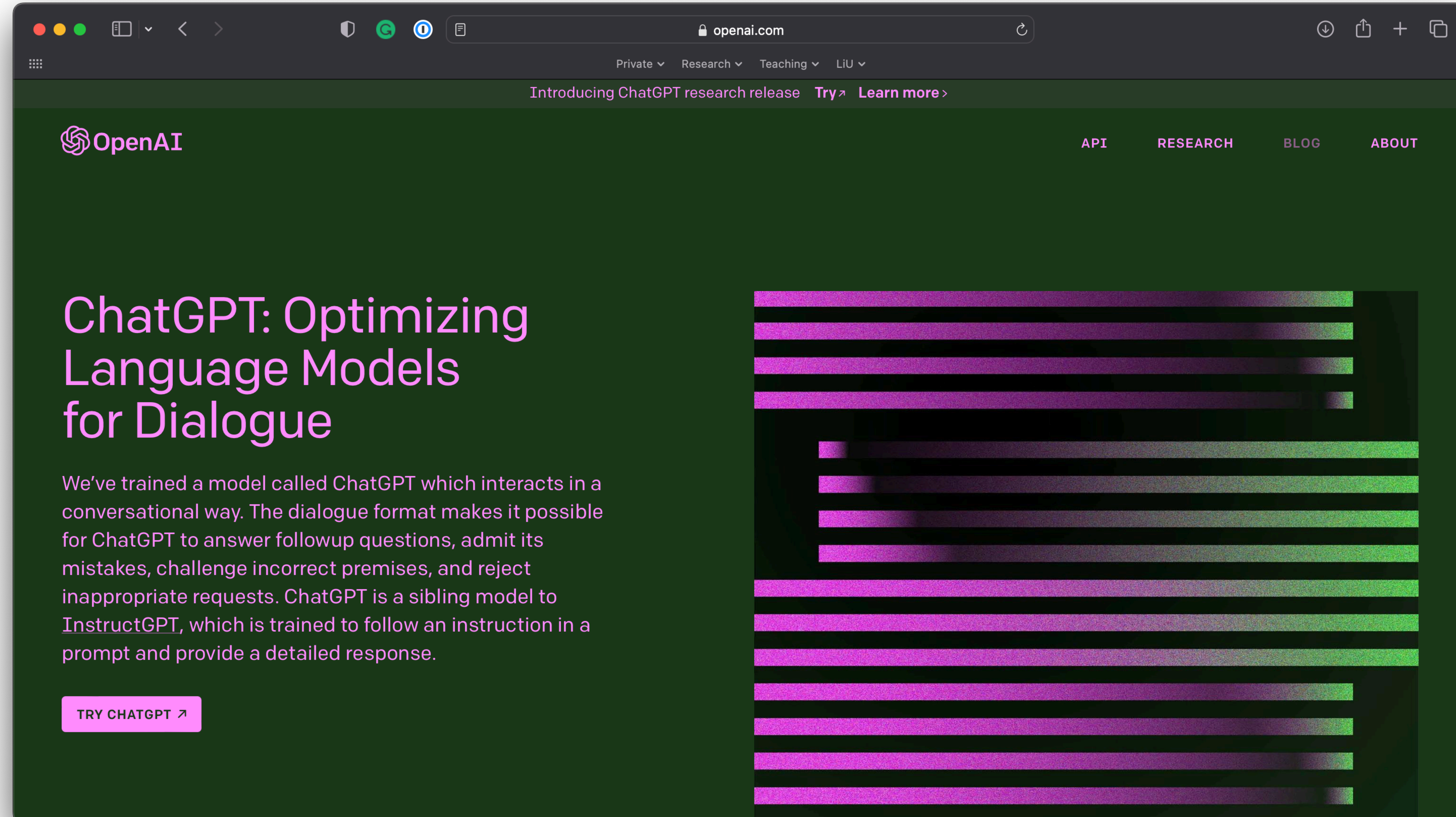
SPARQL query against DBPedia

```
SELECT DISTINCT ?x WHERE {
  ?x dbp:education dbr:Stanford_University.
  dbr:Coursera dbp:founder ?x.
}
```

General-purpose linguistic representations




```
dbr:Coursera dbo:founder dbr:Daphne_Koller
```


openai.com
Private
Research
Teaching
LiU
Introducing ChatGPT research release Try Learn more
OpenAI
API
RESEARCH
BLOG
ABOUT
ChatGPT: Optimizing Language Models for Dialogue
We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.
TRY CHATGPT

What you will learn in this course

- basic methods and techniques for the analysis and interpretation of words, sentences, and texts
- language technology systems
- validation methods
- tools, software libraries, and data

Commercial interest

Diamond-level sponsors of the ACL 2019 conference

Commercial interest

Doctrin • Ericsson • Etteplan

Findwise • Fodina Language Technology

Gavagai • IamIP • iMatrics

Opera Software • Redeye

Saab • Sectra • Spotify

Storytel • Svenska Dagbladet

A major challenge: Ambiguity

- The term **ambiguity** refers to fact that a linguistic expression can often mean several different things.

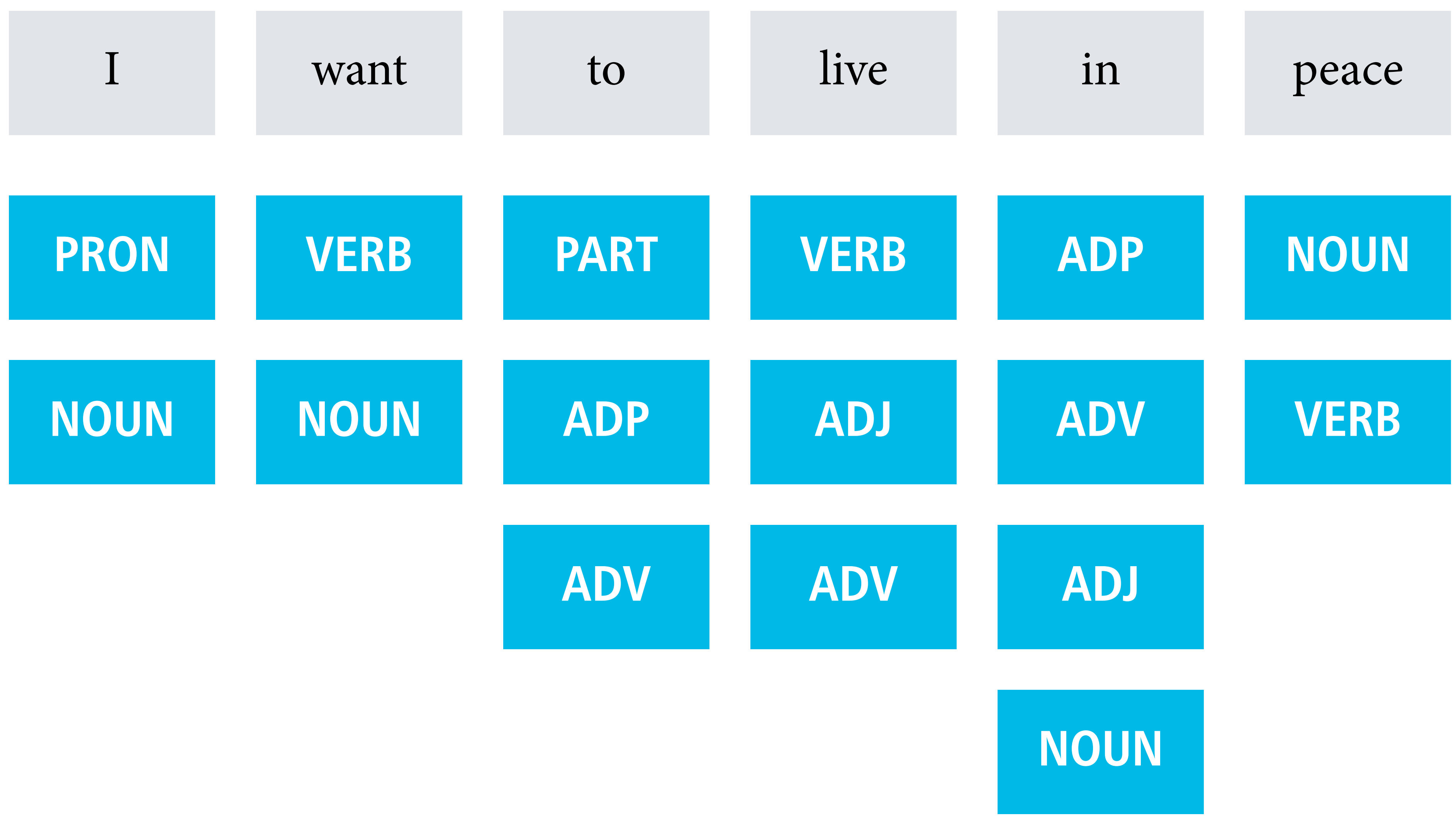
  Time flies like an arrow. Fruit flies like a banana.

- Ambiguity arises at all levels of linguistic description.

  lexical ambiguity, syntactic ambiguity, semantic ambiguity, …

- Humans excel at resolving ambiguities, but for computers, ambiguity poses a major challenge.

Ambiguity causes combinatorial explosion



‘I only want to live in peace, plant potatoes, and dream!’ – Moomin

Ambiguity causes combinatorial explosion

I	want	to	live	in	peace
PRON	VERB	PART	VERB	ADP	NOUN
NOUN	NOUN	ADP	ADJ	ADV	VERB
		ADV	ADV	ADJ	
				NOUN	

'I only want to live in peace, plant potatoes, and dream!' – Moomin

Data to the rescue!

I	want	to	live	in	peace
PRON	VERB	PART	VERB	ADP	NOUN
99.97%	100.00%	63.46%	83.87%	92.92%	100.00%
NOUN	NOUN	ADP	ADJ	ADV	VERB
0.00%	0.00%	35.13%	14.52%	3.61%	0.00%
		ADV	ADV	ADJ	
		0.12%	0.00%	0.03%	
				NOUN	
				0.27%	

Recurring questions

- How does this method work?

  often some kind of algorithm or mathematical formula

- How can we evaluate this method?

  typically some evaluation measure, such as accuracy

- How does this method use data?

  estimate probabilities, learn weights of a neural network, …

This lecture

- What is language technology?
- Course organisation and examination
- Text segmentation

Course organisation and examination

Meet the team!

Ali Basirat

Ehsan Doostmohammadi

Jenny Kunz

Marcel Bollmann

Marco Kuhlmann

Martin Funkquist

Oskar Holmström

Riley Capshaw

	Monday 8–10	Tuesday 10–12	Wednesday 13–17	Friday 8–10
W03	Self-study	LEC Course introduction	LAB Text segmentation (2h)	UPG Introduction to the project
W04	Self-study	LEC Text classification	LAB Text classification (2h)	LAB Text classification
W05	Self-study	LEC Language modelling	LAB Language modelling (2h)	LAB Language modelling
W06	Self-study	LEC Part-of-speech tagging	LAB Part-of-speech tagging (2h)	LAB Part-of-speech tagging
W07	Self-study	LEC Syntactic analysis	LAB Syntactic analysis (2h)	LAB Syntactic analysis
W08	Self-study	LEC Semantic analysis	LAB Semantic analysis (2h)	LAB Semantic analysis
W09	UPG Project supervision	UPG Project supervision	UPG Project supervision	UPG Project supervision
W10	UPG Project supervision	UPG Project supervision	UPG Project supervision	UPG Project supervision
W11	Self-study	UPG Project presentations	UPG Project presentations	UPG Project presentations
W12	EXA Written digital exam (14–18)	Self-study	Self-study	Course deadline (2023-03-25)

Evaluation of the previous session

- The Spring 2022 session had 78 registered students. Out of these, 26 submitted a course evaluation. (Response rate: 33%)
- Overall, students were quite positive about the course (average overall score 4.39 out of 5).

  729G17: 4.39, TDP030: 4.38

- The main point of criticism was that the examiner did not clearly communicate his expectations for the project.

Changes to the course

- More focus on the project, including a dedicated introduction (Friday) and examples in the teaching sessions
- Optional tests are back after the pandemic.

This lecture

- What is language technology?
- Course organisation and examination
- Text segmentation

Text segmentation

How text is stored on a computer

- Text is stored as a sequence of **bytes**. Each byte consists of 8 bits of information, yielding 256 different values.
- Bytes encode characters according to some **encoding scheme**.
- **Unicode** has been developed with the ambition to specify code points for all naturally occurring characters.

  natural languages (even extinct), mathematical symbols, emoji, …

Sample page from the Unicode specification

1F600 **Emoticons** **1F64E**

The emoticons have been organized by mouth shape to make it easier to locate the different characters in the code chart.

Faces

1F600 😀 GRINNING FACE
1F601 😁 GRINNING FACE WITH SMILING EYES
1F602 😂 FACE WITH TEARS OF JOY
1F603 😃 SMILING FACE WITH OPEN MOUTH
→ 263A ☺ white smiling face
1F604 😄 SMILING FACE WITH OPEN MOUTH AND SMILING EYES
1F605 😅 SMILING FACE WITH OPEN MOUTH AND COLD SWEAT
1F629 😩 WEARY FACE
1F62A 😪 SLEEPY FACE
1F62B 😫 TIRED FACE
1F62C 😬 GRIMACING FACE
• should not be depicted with zipper mouth
→ 1F910 🤐 zipper-mouth face
1F62D 😭 LOUDLY CRYING FACE
1F62E 😮 FACE WITH OPEN MOUTH
1F62F 😯 HUSHED FACE
1F630 😰 FACE WITH OPEN MOUTH AND COLD SWEAT
1F631 😱 FACE SCREAMING IN FEAR
1F632 😲 ASTONISHED FACE

Unicode version 14.0 (September 2021): 144,697 different characters

UTF-8 – 8-bit Unicode Transformation Format

- Unicode has slots for $2^{32} = 4{,}294{,}967{,}296$ different characters.
- To encode Unicode characters into bytes, a single character is represented using more than one byte.

  character 0–127 = 1 byte, 128–2,047 = 2 bytes, 2048–65,535 = 3 bytes, …
- This scheme is called UTF-8 (8-bit Unicode Transformation Format) and is the most widely used encoding scheme today.

  January 2019: 92.9% of all websites (Source: w3techs.com)

VarfÃ¶r blir det sÃ¥ hÃ¤r?

	s	å		[SPC]	h	ä		r
Unicode	115	229		32	104	228		114
UTF-8	115	195	182	32	104	195	165	114
Latin-1	115	195	182	32	104	195	165	114
	s	Ã	¥	[SPC]	h	Ã	¤	r

Example by Per Starbäck

Text segmentation

- **Text segmentation** refers to the task of segmenting a text into linguistically meaningful units, such as words and sentences.
- In the case where the relevant units are words or word-like units, the task is called **tokenisation**.

  numbers, punctuation marks

A simple tokeniser based on whitespace

```
# tokenise a sequence of lines using whitespace
def tokenize(lines):
    for line in lines:
        for token in line.split():
            yield token

# open “foo.txt” and print all tokens in it
with open(“foo.txt") as fp:
    for token in tokenize(fp):
        print(token)
```

Tokenisation is harder than one may think

- **Undersegmentation:** The tokeniser misses to split.

  we're *should be* we + 're; bl.a. *should be* bl. + a. (?)

- **Oversegmentation:** The tokeniser splits where it shoud not.

  San + Francisco *should be* one token (?)

- Tokenisation is even harder for non-European languages.

  Chinese word segmentation

A more useful tokenisation

Raw text before tokenisation

The gorgeously elaborate
continuation of “The Lord of
the Rings” trilogy is so huge
that a column of words
cannot adequately describe
co-writer/director Peter
Jackson’s expanded vision of
J.R.R. Tolkien’s Middle-earth.

List of tokens after tokenisation

The gorgeously elaborate
continuation of “ The Lord of
the Rings ” trilogy is so huge
that a column of words
cannot adequately describe
co-writer / director Peter
Jackson ’s expanded vision of
J.R.R. Tolkien ’s Middle-earth .

A simple tokeniser based on regular expressions

```
# tokenise a sequence of lines using a regular expression
def tokenize(regex, lines):
    for line in lines:
        for match in re.finditer(regex, line):
            yield match.group(0)

# open “foo.txt” and print all tokens in it
with open(“foo.txt”) as fp:
    for token in tokenize(fp):
        print(token)
```

Word tokens and word types

'Rose is a rose is a rose is a rose.'

Gertrude Stein (1874–1946)

Corpus	Tokens	Types
Shakespeare	ca. 884,000	ca. 31,000
Riksmöte 2012/2013	4,645,560	96,114
Google Ngrams	1,176,470,663	13,588,391

Normalisation

- Lowercasing

  windows vs. Windows

- Harmonisation of spelling variants

  colour, color; gaol, jail; metre, meter

- Stemming (suffix removal)

  wanted → want, wanting → want, happily → happily

Stop words

- A **stop word** is a word that is frequent but does not contribute much value for the application in question.

  Examples from search engines: *a*, *the*, *and*

- Stop words are application-specific – there is no single universal list of stop words, and not all applications use such lists.

Stop words

a about above after again against all am an and any are aren’t as at be because been before being below between both but by can’t cannot could couldn’t did didn’t do does doesn’t doing don’t down during each few for from further had hadn’t has hasn’t have haven’t having he he’d he’ll he’s her here here’s hers herself him himself his how how’s i i’d i’ll i’m i’ve if in into is isn’t it it’s its itself let’s me more most mustn’t my myself no nor not of off on once only or other ought our ours ourselves out over own same shan’t she she’d she’ll she’s should shouldn’t so some such than that that’s the their theirs them themselves then there there’s these they they’d they’ll they’re they’ve this those through to too under until up very was wasn’t we we’d we’ll we’re we’ve were weren’t what what’s when when’s where where’s which while who who’s whom why why’s with won’t would wouldn’t you you’d you’ll you’re you’ve your yours yourself yourselves

Sentence segmentation

- For some applications, we want to identify not only words but also higher-level units such as sentences and paragraphs.
- **Sentence segmentation** refers to the task of dividing a text into individual sentences.
- Sentence segmentation is harder than splitting at periods.

  We visited the U.S. After that, we visited Canada.

This lecture

- What is language technology?
- Course organisation and examination
- Text segmentation