Artificial Intelligence

Algorithms and Applications with Python

Chapter 9

Dr. Dominik Jung
dominik.jung42@gmail.com

Outline

9 Language and Image Processing

9.1 Applied Natural Language Processing

9.2 Speech Processing and Communication

9.3 Perception and Image Processing

9.4 Generative Modelling

Lectorial 7: Building NLP and Machine Learning Pipelines for Webservices

▶ **What we will learn:**

- We will discuss how to make use of the copious knowledge that is expressed in natural language
- And how to process and prepare language data for machines to build state-of-the-art machine learning pipelines for language processing
- We learn a new type of models, so termed generative models, which allow to generate new instances based on the knowledge of your model

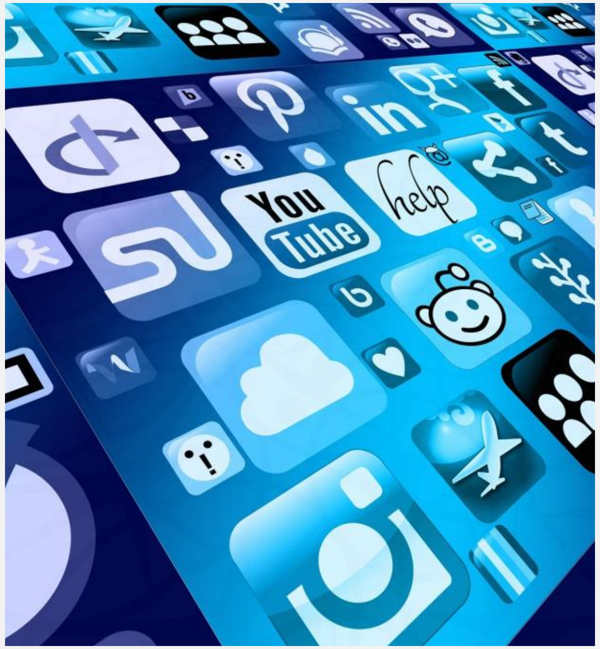


Image source: ↗ Pixabay (2021) / ↗ CC0

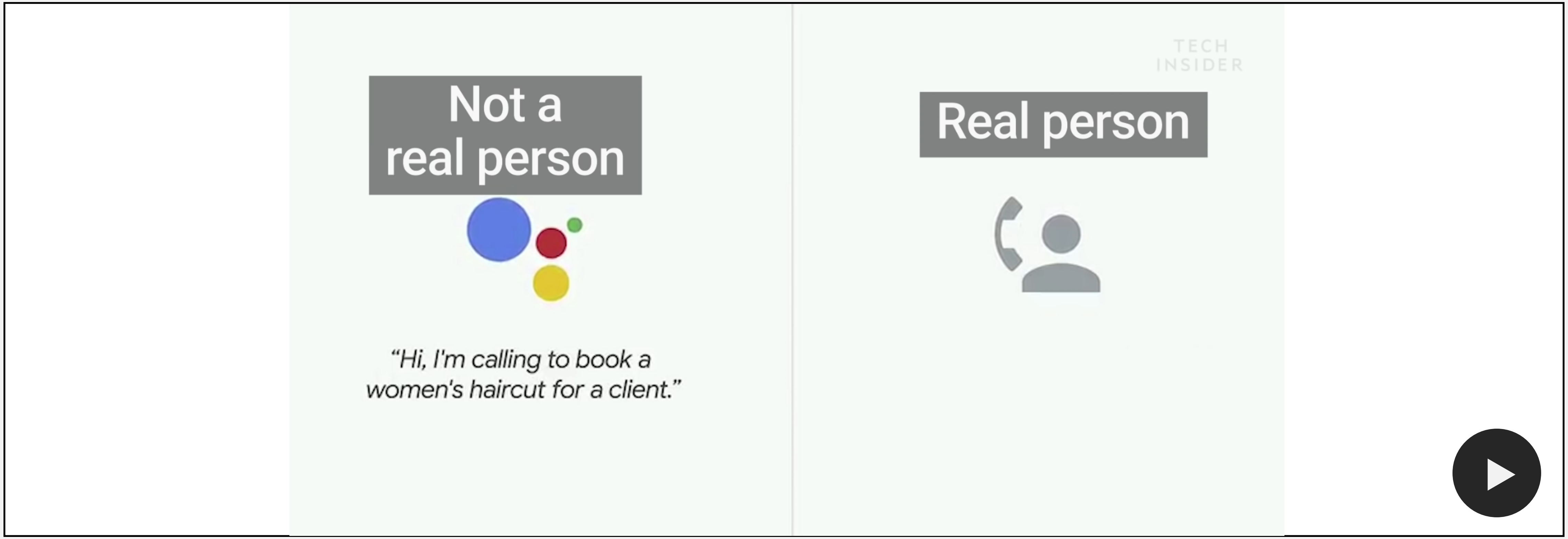
▶ **Duration:**

- 135 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 9.1, 9.4

9.1 Natural Language Processing



D **Natural Language Processing** is the branch of computer science focused on developing systems that allow computers to communicate with people using everyday language

9.1 Text Processing Example I


```
…
<p><b>Dr.-Ing. h.c. F. Porsche AG</b>,
usually shortened to <b>Porsche</b>
(<small>German
pronunciation:&#32;</small><span
title="Representation in the International
Phonetic Alphabet (IPA)" class="IPA"><a
href="/wiki/Help:IPA/Standard_German"
title="Help:IPA/Standard
German">[ˈpɔʁʃə]</a></span>&#32;<span
class="nowrap" style="font-size:85%">(<span
class="unicode haudio"><span class="fn"><span
style="white-space:nowrap;margin-
right:.25em;"><a href="/wiki/File:De-
Porsche.ogg" title="About this sound"><img
alt="audio speaker icon"
src="//upload.wikimedia.org/wikipedia/commons
/thumb/8/8a/Loudspeaker.svg/11px-
Loudspeaker.svg.png" decoding="async"
width="11" height="11
…
```

Preprocessing

- Extract plain text
- Reduce complexity
- prepare transformation

9.1 Text Processing Example II

- Remove html tags

```
Dr.-Ing. h.c. F. Porsche AG, usually shortened to Porsche (German pronunciation: [ˈpɔʁʃə] (audio speaker iconlisten); see below), is a German automobile manufacturer specializing in high-performance sports cars, SUVs and sedans, headquartered in Stuttgart, Baden-Württemberg, Germany.
```

- Remove vocal information (mostly done by html tags, so do it before you remove them)

```
Dr.-Ing. h.c. F. Porsche AG, usually shortened to Porsche (German pronunciation: [ˈpɔʁʃə] (audio speaker iconlisten); see below), is a German automobile manufacturer specializing in high-performance sports cars, SUVs and sedans, headquartered in Stuttgart, Baden-Württemberg, Germany.
```

- Remove special characters

```
Dr.-Ing. h.c. F. Porsche AG, usually shortened to Porsche (:();), is a German automobile manufacturer specializing in high-performance sports cars, SUVs and sedans, headquartered in Stuttgart, Baden-Württemberg, Germany.
```

9.1 Text Processing Example III

- Convert to same case

```
Dr Ing h c F Porsche AG usually shortened to Porsche German is a German automobile manufacturer specializing in high-performance sports cars SUVs and sedans headquartered in Stuttgart Baden-Württemberg Germany
```

- Remove single characters, and unimportant words

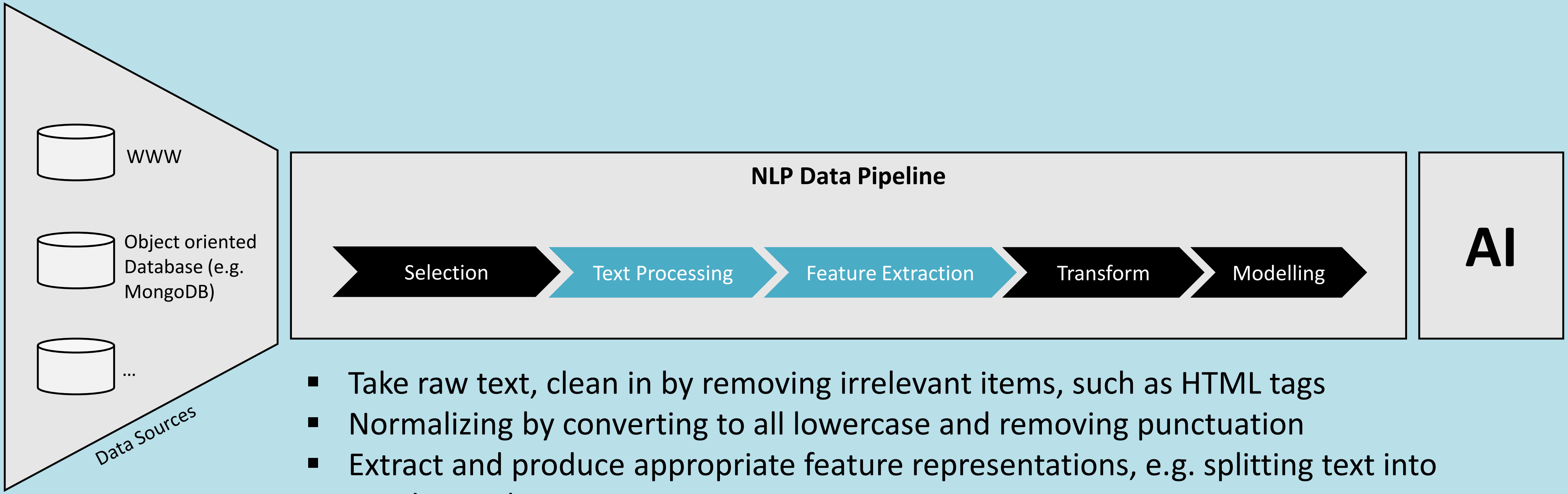
```
dr ing h c f porsche ag usually shortened to porsche german is a german automobile manufacturer specializing in high-performance sports cars suvs and sedans headquartered in stuttgart baden-württemberg germany
```

- Remove special characters

```
dr ing porsche ag shortened porsche german german automobile manufacturer specializing in high-performance sports cars suvs sedans headquartered stuttgart baden-württemberg germany
```

➡ Transform to usable schemata like e.g. table

9.1 NLP Data Pipelines



- Take raw text, clean in by removing irrelevant items, such as HTML tags
- Normalizing by converting to all lowercase and removing punctuation
- Extract and produce appropriate feature representations, e.g. splitting text into words or tokens
- Removing words that are too common, also known as stop words
- Identifying different parts of speech and named entities
- Converting words into their dictionary forms, using stemming and lemmatization
- Design your machine learning model, and then use it in your planned AI system

9.1 NLP Cleaning Tools

Beautiful Soup


Beautiful Soup 4.9.0 documentation » Beautiful Soup Documentation

Table of Contents

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.10.0. The examples in this documentation were written for Python 3.8.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed and that all support for it was dropped on December 31, 2020. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

This documentation has been translated into other languages by Beautiful Soup users:

- 这篇文档当然还有中文版.
- このページは日本語で利用できます(外部リンク)
- 이 문서는 한국어 번역도 가능합니다.
- Este documento também está disponível em Português do Brasil.
- Эта документация доступна на русском языке.

Getting help

If you have questions about Beautiful Soup, or run into problems, send mail to the discussion group. If your problem involves parsing an HTML document, be sure to mention what the diagnose() function says about that document.

Quick Start

Here's an HTML document I'll be using as an example throughout this document. It's part of a story from *Alice in Wonderland*:

```
html_doc = """<html><head><title>The Dormouse's story</title></head>
<body>
```

Regular Expressions

```
/((\d{3})(?:\.|-))?(\d{3})(?:\.|-)(\d{4})/
```

9.1 NLP Normalization

Sport-car

Sportcar

Sport Car

sport-car

…

```
text = "sport-Car"

# convert to lowercase
text = text.lower()

# Remove punctuation characters
import re
text = re.sub(r"[^a-zA-Z0-9]", " ", text)
```

```
>>> print(text)
```

! Do not mix up this concept with statistical normalization we discussed in lecture 4

9.1 NLTK – Python package for Tokenization

```
import nltk
nltk.download()
```

NLTK

Documentation

Search

NLTK Documentation

API Reference
Example Usage
Module Index
Wiki
FAQ
Open Issues
NLTK on GitHub

Installation

Installing NLTK
Installing NLTK Data

More

Release Notes
Contributing to NLTK
NLTK Team

nltk.tokenize package

NLTK Tokenizer Package

Tokenizers divide strings into lists of substrings. For example, tokenizers can be used to find the words and punctuation in a string:

```
>>> from nltk.tokenize import word_tokenize
>>> s = '''Good muffins cost $3.88\nin New York.  Please buy me
... two of them.\n\nThanks.'''
>>> word_tokenize(s)
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

This particular tokenizer requires the Punkt sentence tokenization models to be installed. NLTK also provides a simpler, regular-expression based tokenizer, which splits text on whitespace and punctuation:

```
>>> from nltk.tokenize import wordpunct_tokenize
>>> wordpunct_tokenize(s)
['Good', 'muffins', 'cost', '$', '3', '.', '88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

If you want to save storage, you can download only specific functions like “punkt”, with `nltk.download('punkt')`. You can learn more about nltk data installation on their ↗ website

9.1 Tokenization

- You can use the in-build tokenization methods from NLTK to perform straight forward tokenization of your texts

```
>>> from nltk.tokenize import word_tokenize
>>> text = "I like this lecture. It has many practical examples."
>>> # Split text into words using NLTK
>>> words = word_tokenize(text)
>>> print(words)

['I', 'like', 'this', 'lecture', '.', 'It', 'has', 'many', 'practical',
'examples', '.']
```

? There are many more build-in tokenization functions. What will e.g. `sent_tokenize(text)` return?

9.1 Stop Word Removal

- You can also use this package to remove noise like stopwords in different languages:

```
from nltk.corpus import stopwords
words = [w for w in words if w not in stopwords.words("english")]
```

```
>>> print(stopwords.words("english"))
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their', 'theirs …
```

9.1 Motivation for Part-of-Speech Tagging (POS)

- Identifiying how words are being used in a sentence helps us better understand what is being said:

pronoun verb noun

Because she was so fast, Marie was always the first at the restaurant.

- Helps us to find relationships between words

- Get full list of word classes with `nltk.help.upenn_tagset()`

9.1 Part-of-Speech Tagging (POS)

- You can also use this package to remove noise like stopwords in different languages:

```
from nltk import pos_tag, ne_chunk
from nltk.tokenize import word_tokenize
text = "If ever there was proof that going electric needn’t be a
compromise, the Porsche Taycan is it"
```

```
>>> sentence = word_tokenize(text) # tokenize text
>>> pos_tag(sentence) # tag each word with pos

[('If', 'IN'),                 IN: preposition or conjunction, subordinating
 ('ever', 'RB'),               RB: adverb
 ('there', 'EX'),              EX: existential there
 ('was', 'VBD'),               VBD: verb, past tense
 ('proof', 'NN’),…             NN: noun, common, singular or mass
```

9.1 Named Entity Recognition (NER)

- Identifiying how words are being used in a sentence helps us better understand what is being said:

2 words, 1 entity

The $\boxed{\text{Porsche AG}}$ builds sport-cars

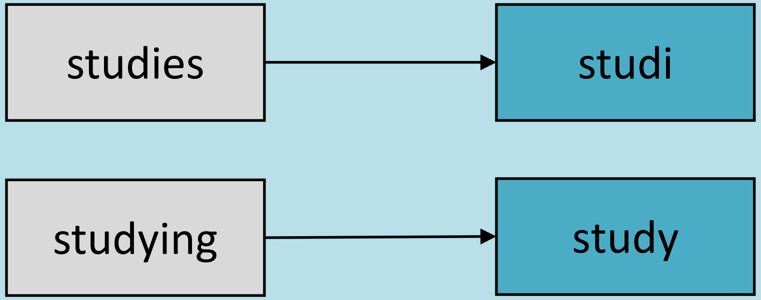
- Tokenize your text, make POS, then recognize named entities in text:

```
text = "The Porsche AG builds sport-cars"
tree = ne_chunk(pos_tag(word_tokenize(text)))
print(tree)
```

```
(S
  The/DT
  (ORGANIZATION Porsche/NNP)
  AG/NNP
  builds/VBZ
  sport-cars/NNS)
```

9.1 Stemming and Lemmatization

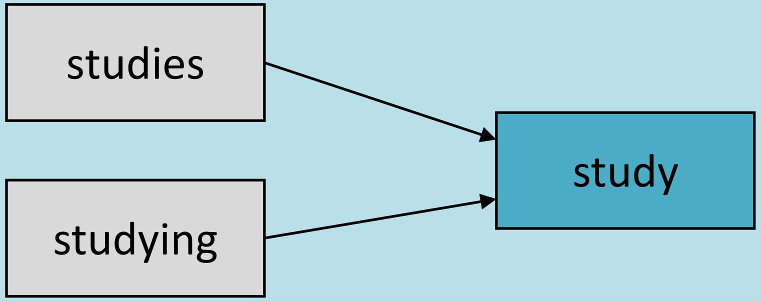
Stemming



- Removes suffixes

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

words = ["driver", "drives", "drive"]
for w in words:
    print(w, " : ", ps.stem(w))
```

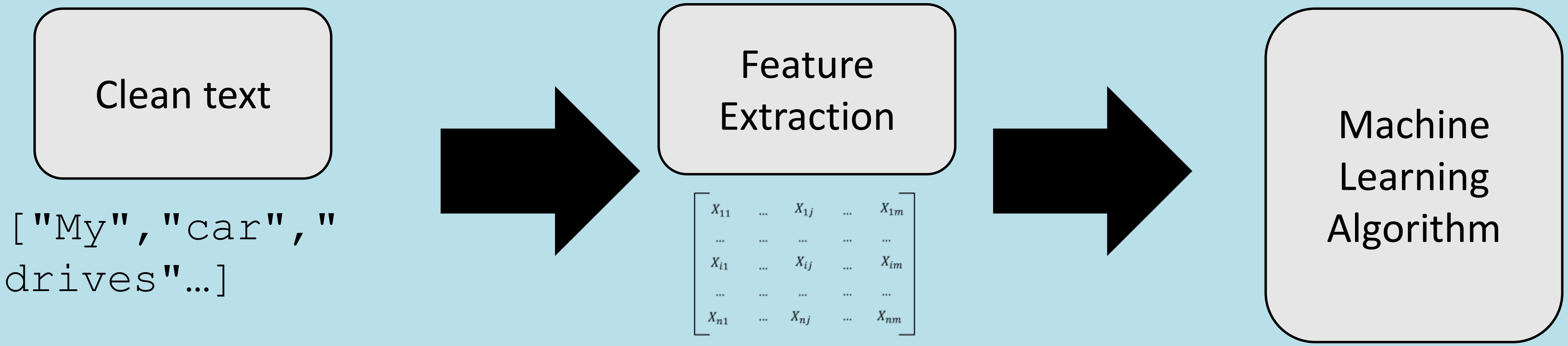
Lemmatization



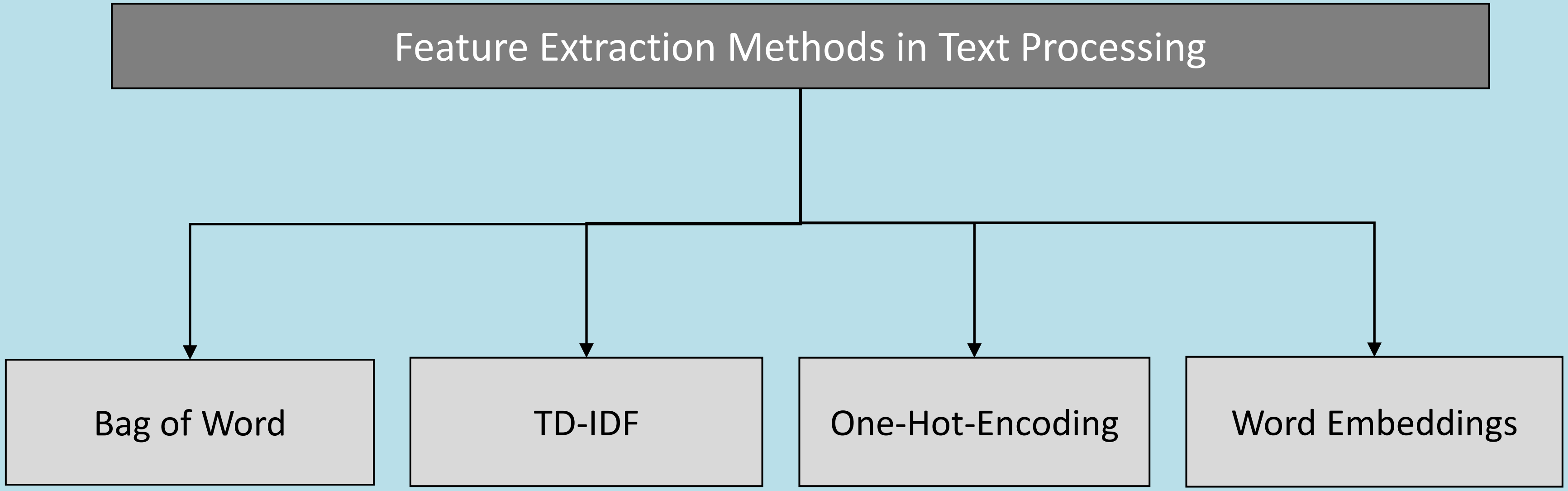
- Uses grammatical informations

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("sport-car :",
lemmatizer.lemmatize("rocks"))
```

9.1 Motivation: Feature Extraction



9.1 Popular Word and Representations for NLP



9.1 Python Code Example for Feature Extraction

- Load Packages and tokenize documents

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize

corpus = ["I like driving my Porsche.",
        "It is a very fast sport-car"
         …]

stop_words = stopwords.words("english")
lemmatizer = WordNetLemmatizer()

def tokenize(text):
    text = re.sub(r"[^a-zA-Z0-9]", " ", text.lower())
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]

    return tokens
```

9.1 Bag of Words

```
Document1 =
["My", "car",
"drives",
"very", "fast"]

Document2 =
["My", "car",
"is", "a",
"Porsche"]

Document3 =
["I", "like",
"my", "911"]
```

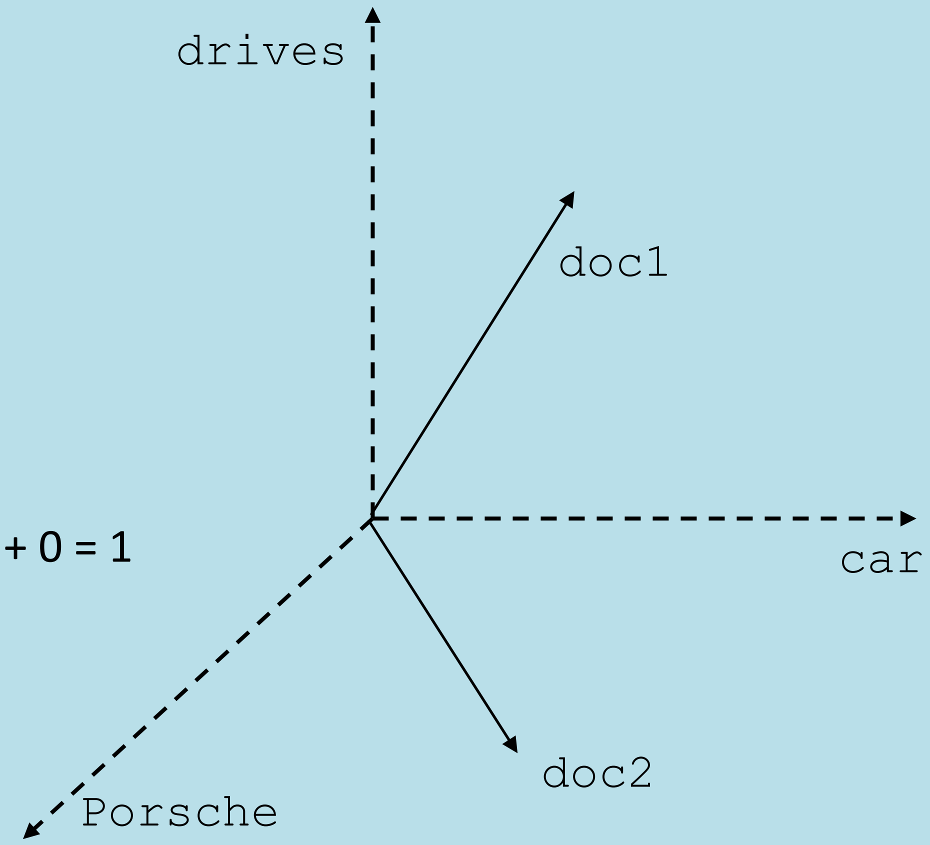
Document-term matrix

Term	Doc1	Doc2	Doc3	Sum
car	1	1	0	2
drives	1	0	0	1
very	1	0	0	1
fast	1	0	0	1
Porsche	0	1	0	1
like	0	0	1	1
911	0	0	1	1

```
Noise = ["a", "is", "my", "I"]
```

9.1 Similarity in Bag of Words

Term	Doc1	Doc2
car	1	1
drives	1	0
very	1	0
fast	1	0
Porsche	0	1



1. Dot product

$$doc1 \cdot doc2 = \sum a_o b_o + a_1 b_1 + \ldots + a_n b_n = 1 + 0 + 0 + 0 + 0 = 1$$

2. Cosine similarity

$$\cos(\theta) = \frac{doc1 \cdot doc2}{\| doc1 \| \cdot \| doc2 \|} = \frac{1}{\sqrt{4} \times \sqrt{2}}$$

9.1 Bag of Words with NLTK

- initialize count vectorizer object

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer(tokenizer=tokenize)
```

- compute counts of each word (token) in our text data

```
>>> X = vect.fit_transform(corpus)
>>> X.toarray()

array([[0, 0, 0, 1, 0,], [0, 0, 0, 1, 0,] ….], dtype=int64)
```

9.1 Term Frequency (TF) – Inverse Document Frequency (IDF)

Term	Doc1	Doc2	Doc3	Freq
engine	3	2	0	5
drives	2	1	0	3
fuel	3	1	0	4
price	0	0	5	5
saldo	0	0	3	3
…	…	…	…	…

Term	Doc1	Doc2	Doc3	Freq
engine				5
drives				3
fuel				4
price				5
saldo				3
…	…	…	…	…

$$tf = \frac{count(t,d)}{|d|}$$

$$idf = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$tfidf = tf \cdot idf$$

?

What will be the new values of the idf table? Can you compute them?

Do you have any idea of the background of doc1, doc2 and doc3?

9.1 TfidfTransformer with sklearn

- Compute tf-idf values based on the counts from count vectorizer

```
from sklearn.feature_extraction.text import TfidfTransformer

transformer = TfidfTransformer(smooth_idf=False)
tfidf = transformer.fit_transform(X)
```

- You can print it if you want

```
>>> tfidf.toarray()

array([[ 0.        ,  0.        ,  0.        ,  0.37       , 0.     ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.     ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.57   ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ]])
```

9.1 Alternative: TfidfVectorizer = CountVectorizer + TfidfTransformer

- compute tf-idf values based on the counts from count vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

# initialize tf-idf vectorizer object
vectorizer = TfidfVectorizer()

# compute bag of word counts and tf-idf values
X = vectorizer.fit_transform(corpus)
```

9.1 Application of TF-IDF score

- **Information retrieval**: Match TF-IDF score of a user query against the whole document set to figure out how relevant a document is to that given query

- **Keywords extraction**: The highest ranking words for a document in terms of TF-IDF score can very well represent the keywords of that document

Outline

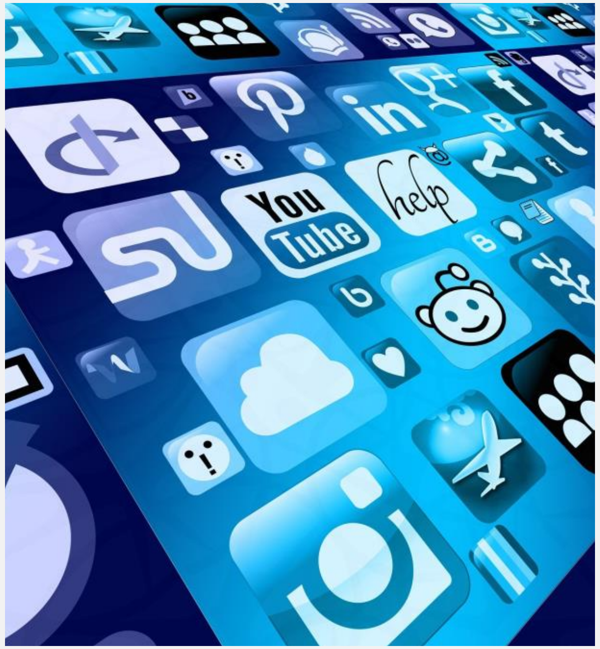


Image source: ↗ Pixabay (2021) / ↗ CC0

▶ **What we will learn:**
- We will discuss how to make use of the copious knowledge that is expressed in natural language
- And how to process and prepare language data for machines to build state-of-the-art machine learning pipelines for language processing
- We learn a new type of models, so termed generative models, which allow to generate new instances based on the knowledge of your model

▶ **Duration:**
- 135 min + 90 (Lectorial)

▶ **Relevant for Exam:**
- 9.1, 9.4

9.2 Communication

Speaker

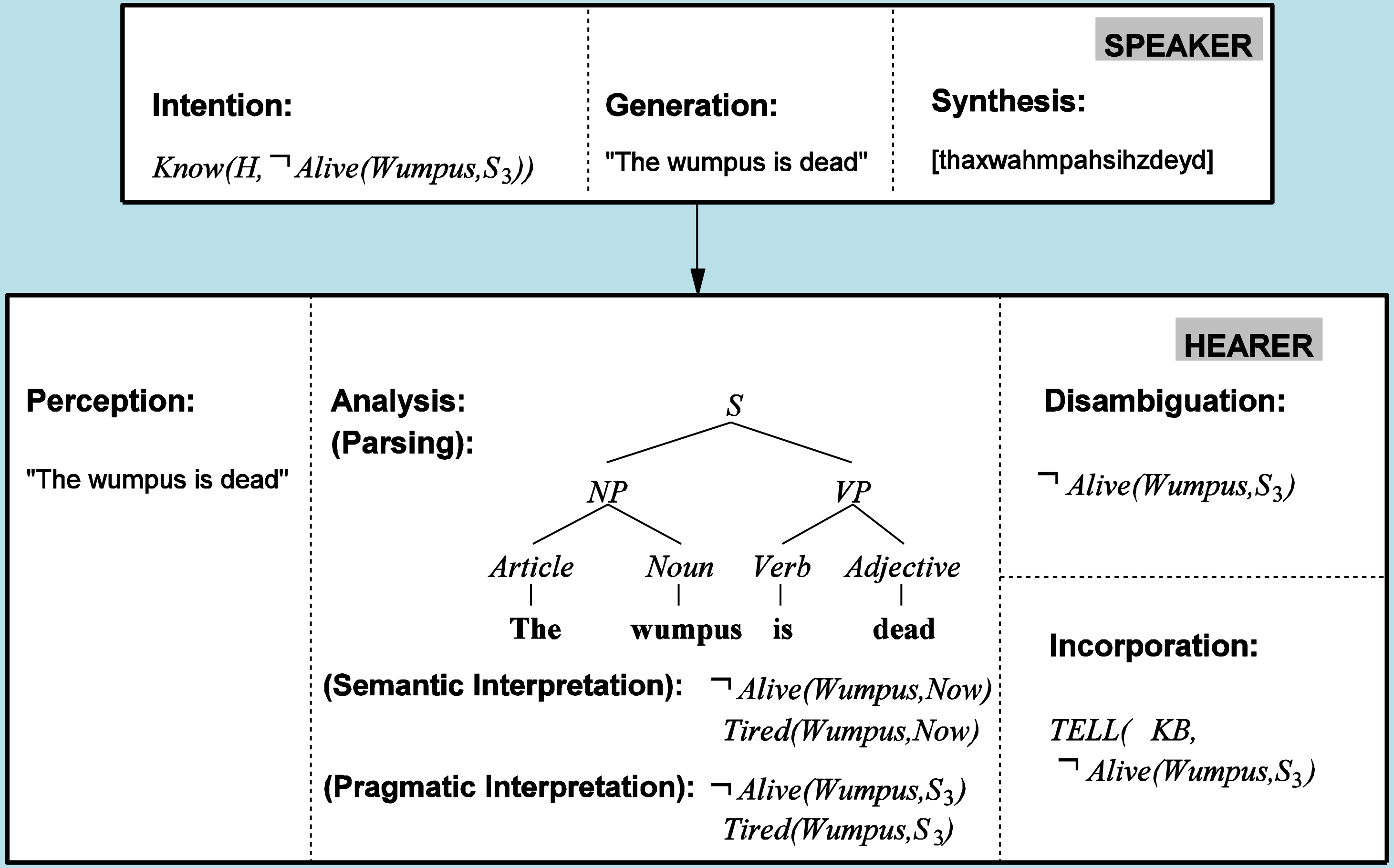
- **Intention**: Decide when and what information should be transmitted (a.k.a. strategic generation). May require planning and reasoning about agents' goals and beliefs.
- **Generation**: Translate the information to be communicated (in internal logical representation or "language of thought") into string of words in desired natural language (a.k.a. tactical generation).
- **Synthesis**: Output the string in desired modality, text or speech.

Adapted from Rusell, S., & Norvig, P. (2016)

Supervised Learning

- **Perception**: Map input modality to a string of words, e.g. optical character recognition (OCR) or speech recognition.
- **Analysis**: Determine the information content of the string.
  - Syntactic interpretation (parsing): Find the correct parse tree showing the phrase structure of the string.
  - Semantic Interpretation: Extract the (literal) meaning of the string (logical form).
  - Pragmatic Interpretation: Consider effect of the overall context on altering the literal meaning of a sentence.
- **Incorporation**: Decide whether or not to believe the content of the string and add it to the KB.

9.2 Communication (cont)



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Syntax, Semantic, Pragmatics

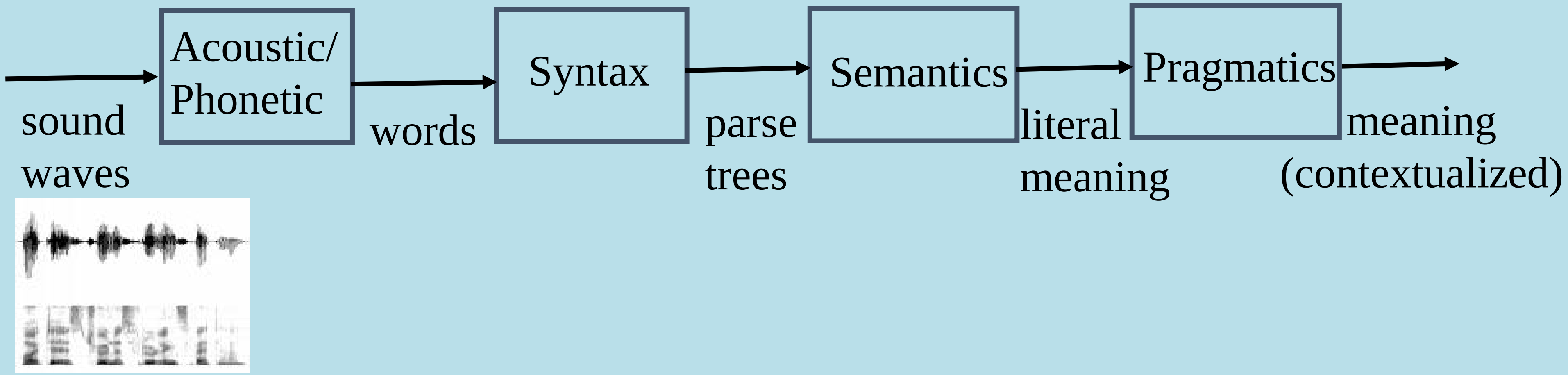
- Syntax concerns the proper ordering of words and its affect on meaning.
  - The dog bit the boy.
  - The boy bit the dog.
  - Colorless green ideas sleep furiously.

- Semantics concerns the (literal) meaning of words, phrases, and sentences.
  - “plant” as a photosynthetic organism
  - “plant” as a manufacturing facility
  - “plant” as the act of sowing

- Pragmatics concerns the overall communicative and social context and its effect on interpretation.
  - The ham sandwich wants another beer. (co-reference, anaphora)
  - John thinks vanilla. (ellipsis)

Adapted from Rusell, S., & Norvig, P. (2016)

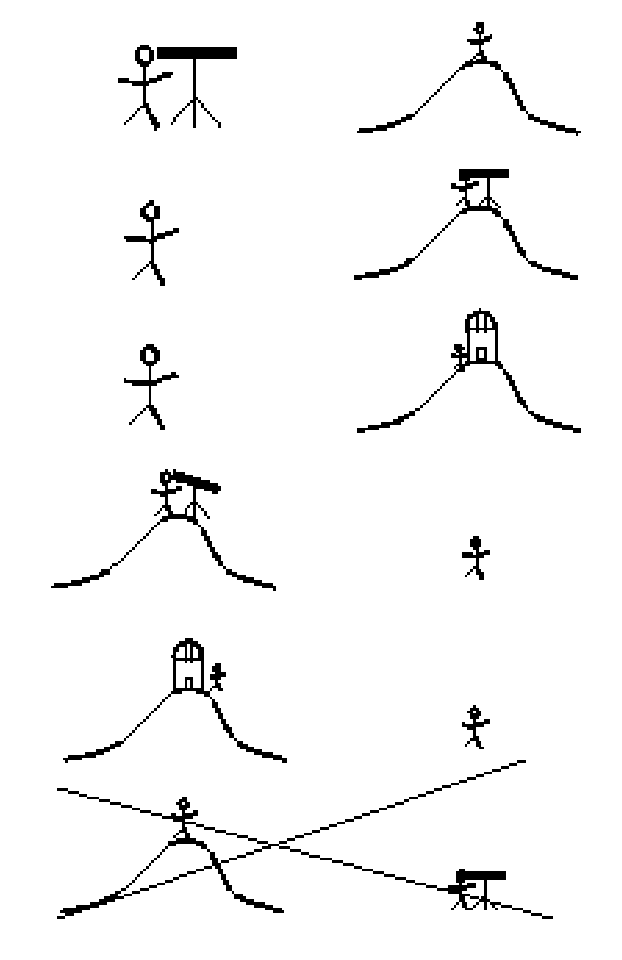
9.2 Modular Comprehension



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Ambiguity

- Natural language is highly ambiguous and must be *disambiguated*.
  - I saw the man on the hill with a telescope.
  - I saw the Grand Canyon flying to LA.
  - Time flies like an arrow.
  - Horse flies like a sugar cube.
  - Time runners like a coach.
  - Time cars like a Porsche.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Ambiguity is Ubiquitous I

- Speech Recognition
  - “recognize speech” vs. “wreck a nice beach”
  - “youth in Asia” vs. “euthanasia”
- Syntactic Analysis
  - “I ate spaghetti with chopsticks” vs. “I ate spaghetti with meatballs.”
- Semantic Analysis
  - “The dog is in the pen.” vs. “The ink is in the pen.”
  - “I put the plant in the window” vs. “Ford put the plant in Mexico”

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Ambiguity is Ubiquitous II

- Pragmatic Analysis
  - From “The Pink Panther Strikes Again”:

Clouseau: Does your dog bite?
Hotel Clerk: No.
Clouseau: [bowing down to pet the dog] Nice doggie.
[Dog barks and bites Clouseau in the hand]
Clouseau: I thought you said your dog did not bite!
Hotel Clerk: That is not my dog.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Ambiguity is Explosive

- Ambiguities compound to generate enormous numbers of possible interpretations.
- In English, a sentence ending in n prepositional phrases has over 2n syntactic interpretations (cf. Catalan numbers).
  - "I saw the man with the telescope": 2 parses
  - "I saw the man on the hill with the telescope.": 5 parses
  - "I saw the man on the hill in Texas with the telescope": 14 parses
  - "I saw the man on the hill in Texas with the telescope at noon.": 42 parses
  - "I saw the man on the hill in Texas with the telescope at noon on Monday" 132 parses

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Humor and Ambiguity

Many jokes rely on the ambiguity of language:

- Groucho Marx: One morning I shot an elephant in my pajamas. How he got into my pajamas, I'll never know.
- She criticized my apartment, so I knocked her flat.
- Noah took all of the animals on the ark in pairs. Except the worms, they came in apples.
- Policeman to little boy: "We are looking for a thief with a bicycle." Little boy: "Wouldn't you be better using your eyes."
- Why is the teacher wearing sun-glasses. Because the class is so bright.

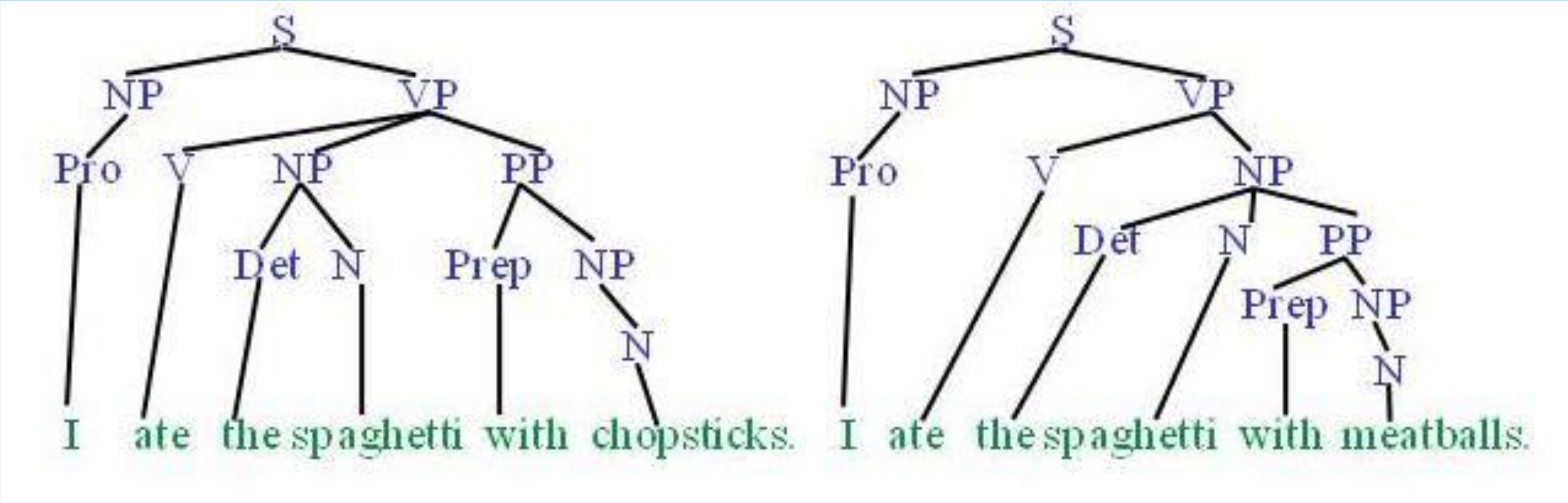
Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Natural Languages vs. Computer Languages

- Ambiguity is the primary difference between natural and computer languages.
- Formal programming languages are designed to be unambiguous, i.e. they can be defined by a grammar that produces a unique parse for each sentence in the language.
- Programming languages are also designed for efficient (deterministic) parsing, i.e. they are deterministic context-free languages (DCLFs).
- A sentence in a DCFL can be parsed in O(n) time where n is the length of the string.

Adapted from Rusell, S., & Norvig, P. (2016)

- Produce the correct syntactic parse tree for a sentence.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Context Free Grammars (CFG)

- *N* a set of ***non-terminal symbols*** (or ***variables***)
- Σ a set of ***terminal symbols*** (disjoint from *N*)
- *R* a set of ***productions*** or ***rules*** of the form $A \rightarrow \beta$, where A is a non-terminal and β is a string of symbols from $(\Sigma \cup N)^*$
- S, a designated non-terminal called the ***start symbol***

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Simple CFG for ATIS English

Grammar

S → NP VP

S → Aux NP VP

S → VP

NP → Pronoun

NP → Proper-Noun

NP → Det Nominal

Nominal → Noun

Nominal → Nominal Noun

Nominal → Nominal PP

VP → Verb

VP → Verb NP

VP → VP PP

PP → Prep NP

Lexicon

Det → the | a | that | this

Noun → book | flight | meal | money

Verb → book | include | prefer

Pronoun → I | he | she | me

Proper-Noun → Houston | NWA

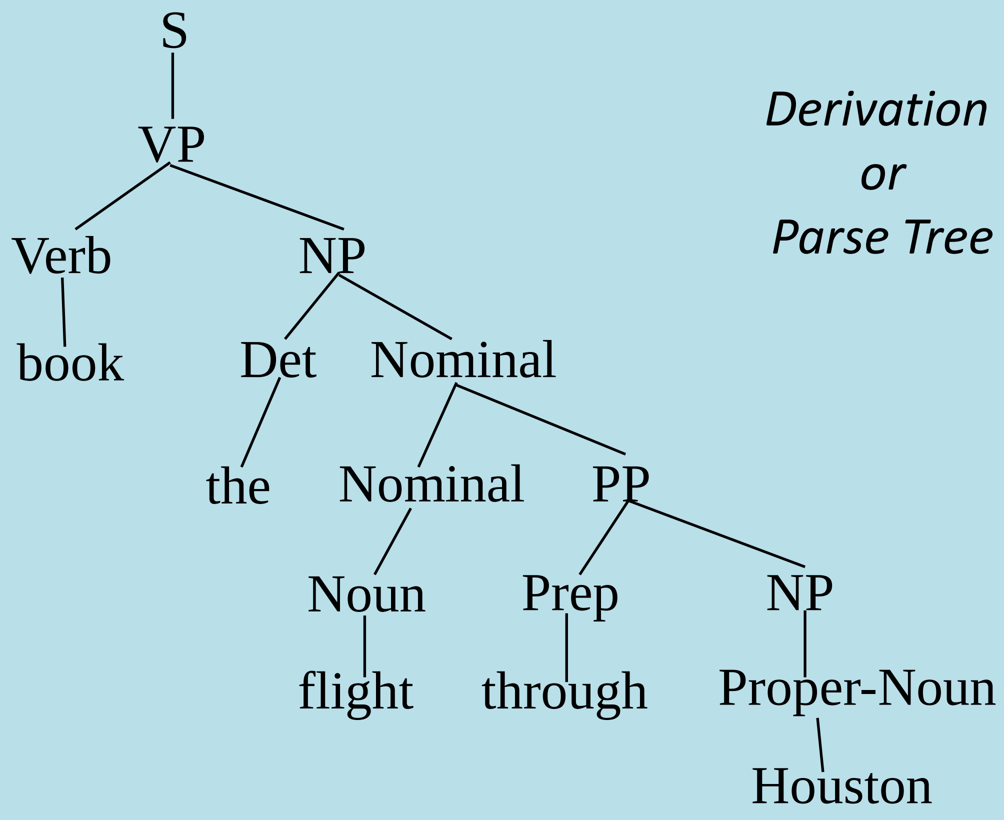
Aux → does

Prep → from | to | on | near | through

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Sentence Generation

- Sentences are generated by recursively rewriting the start symbol using the productions until only terminals symbols remain.

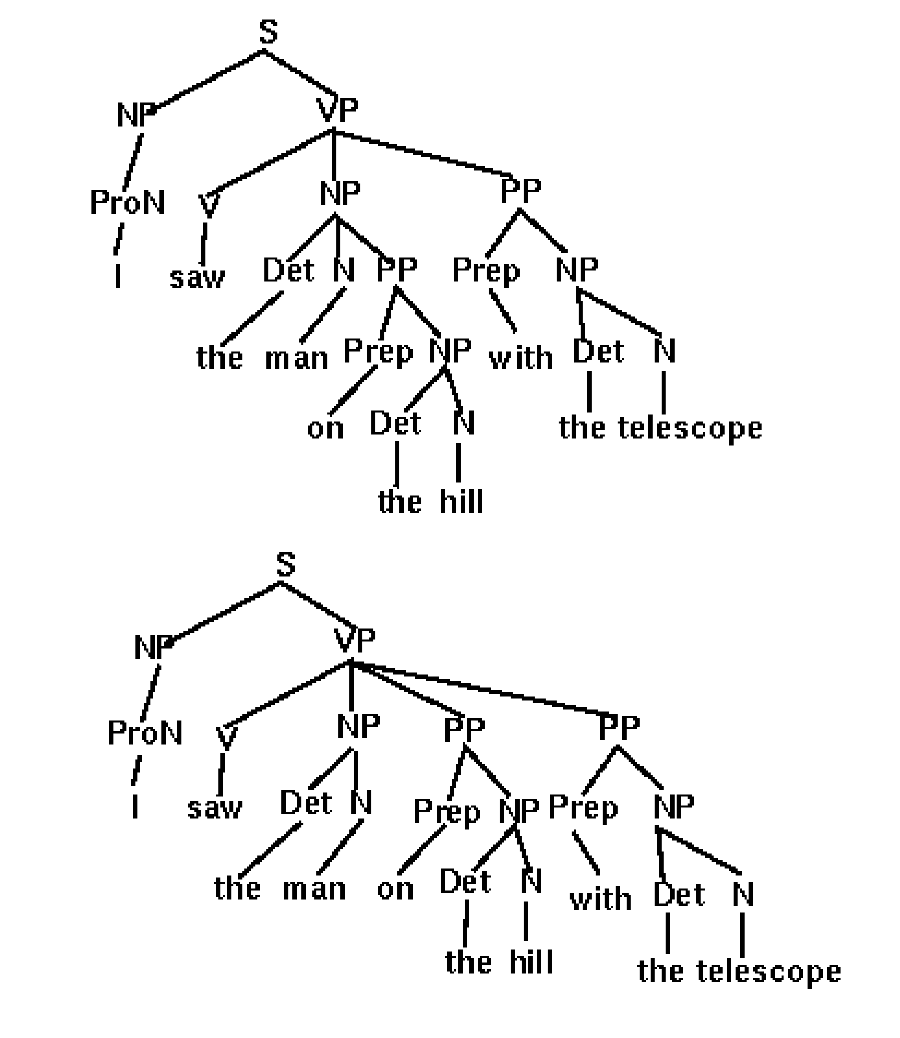


Derivation
or
Parse Tree

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Parse Trees and Syntactic Ambiguity

- If a sentence has more than one possible derivation (parse tree) it is said to be *syntactically ambiguous*.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Prepositional Phrase Attachment Explosion

- A transitive English sentence ending in *m* prepositional phrases has *at least* 2^m parses.

I saw the man on the hill with a telescope on Tuesday in Austin….

- The exact number of parses is given by the *Catalan numbers* (where *n*=*m*+1)

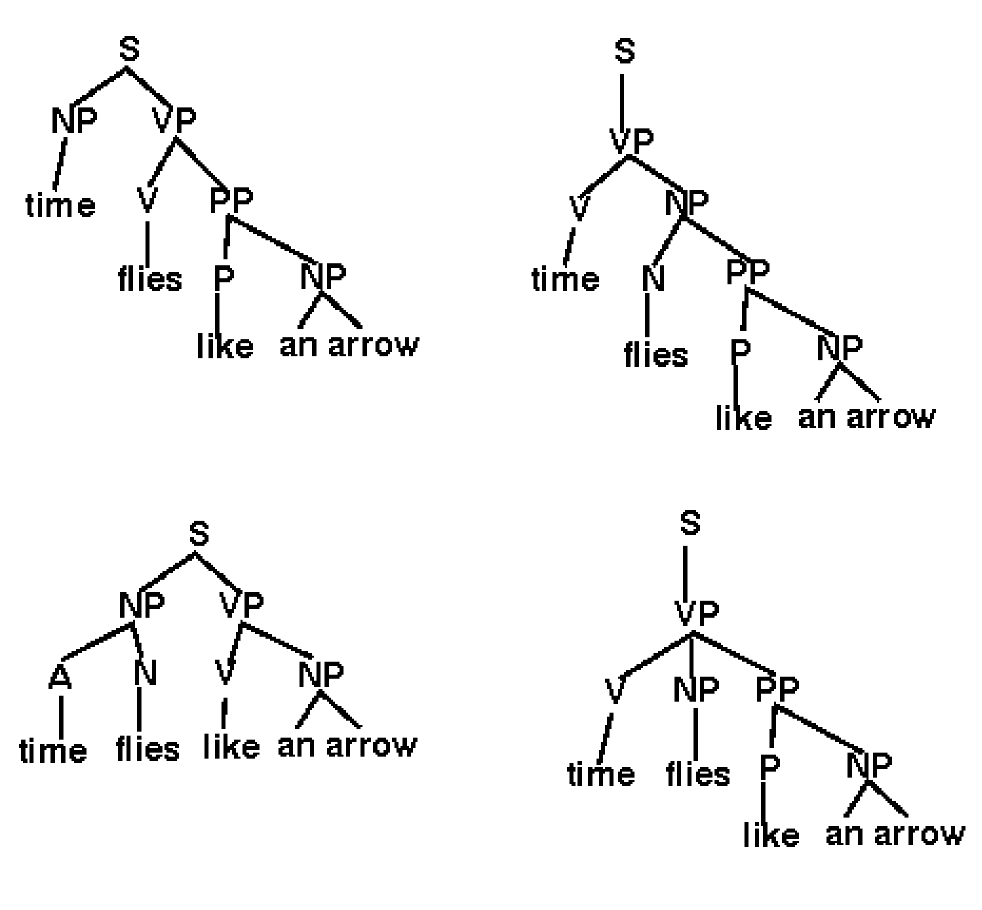
$$\binom{2n}{n} - \binom{2n}{n-1} \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

1, 2, 5, 14, 132, 429, 1430, 4862, 16796, ……

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Spurious Ambiguity

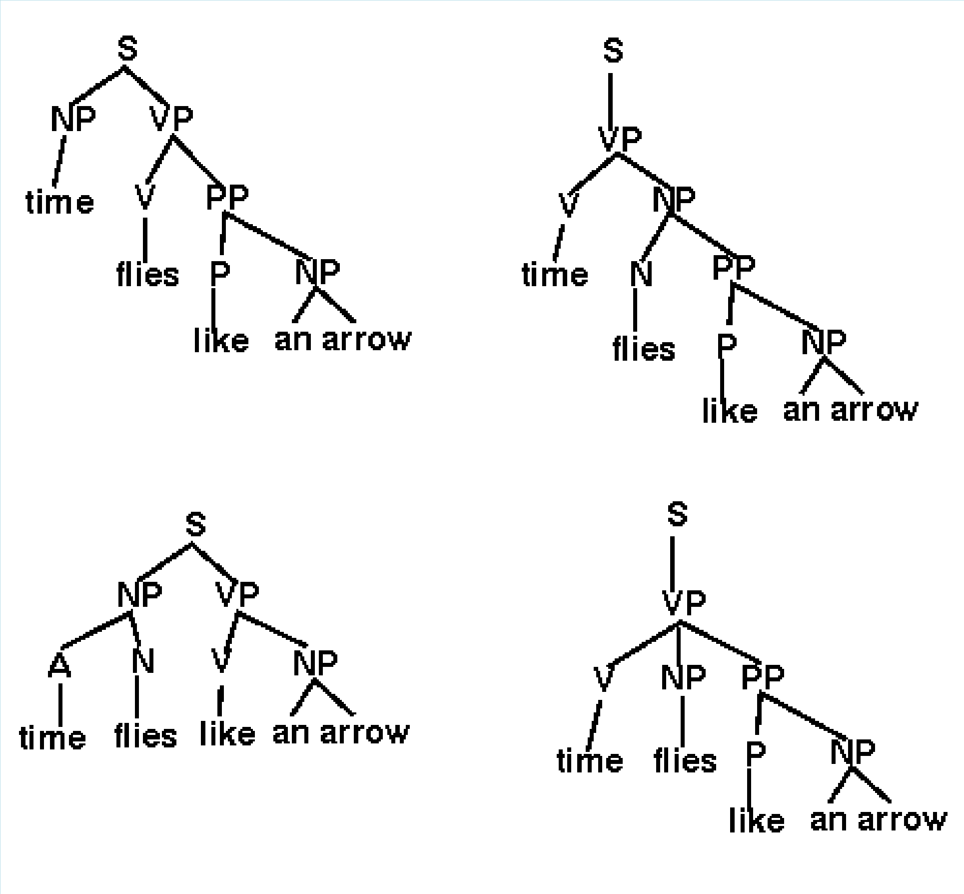
- Most parse trees of most NL sentences make no sense.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Spurious Ambiguity

- Most parse trees of most NL sentences make no sense.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Parsing

- Given a string of non-terminals and a CFG, determine if the string can be generated by the CFG.
  - Also return a parse tree for the string
  - Also return all possible parse trees for the string

- Must search space of derivations for one that derives the given string.
  - **Top-Down Parsing**: Start searching space of derivations for the start symbol.
  - **Bottom-up Parsing**: Start search space of reverse deivations from the terminal symbols in the string.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Top Down vs. Bottom Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Syntactic Parsing & Ambiguity

- Just produces all possible parse trees.
- Does not address the important issue of ambiguity resolution.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Statistical Parsing

- Statistical parsing uses a probabilistic model of syntax in order to assign probabilities to each parse tree.
- Provides principled approach to resolving syntactic ambiguity.
- Allows supervised learning of parsers from tree-banks of parse trees provided by human linguists.
- Also allows unsupervised learning of parsers from unannotated text, but the accuracy of such parsers has been limited.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Probabilistic Context Free Grammar (PCFG)

- A PCFG is a probabilistic version of a CFG where each production has a probability.
- Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal.
- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

Adapted from Rusell, S., & Norvig, P. (2016)

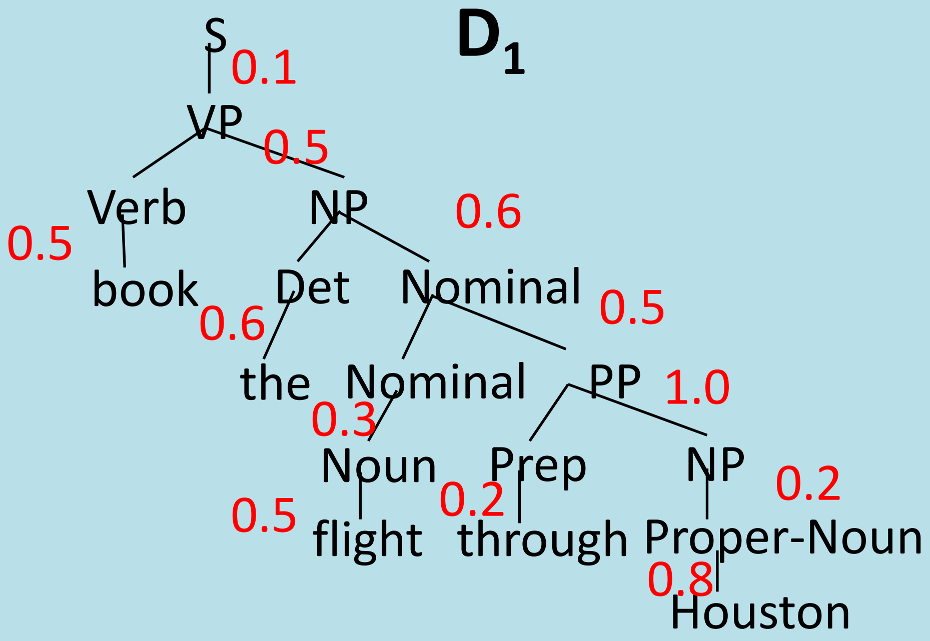
9.2 Simple PCFG for ATIS English

Grammar	Prob		Lexicon
S → NP VP	0.8		Det → the | a | that | this
S → Aux NP VP	0.1	+ 1.0	0.6 0.2 0.1 0.1
S → VP	0.1		Noun → book | flight | meal | money
NP → Pronoun	0.2		0.1 0.5 0.2 0.2
NP → Proper-Noun	0.2	+ 1.0	Verb → book | include | prefer
NP → Det Nominal	0.6		0.5 0.2 0.3
Nominal → Noun	0.3		Pronoun → I | he | she | me
Nominal → Nominal Noun	0.2	+ 1.0	0.5 0.1 0.1 0.3
Nominal → Nominal PP	0.5		Proper-Noun → Houston | NWA
VP → Verb	0.2		0.8 0.2
VP → Verb NP	0.5	+ 1.0	Aux → does
VP → VP PP	0.3		1.0
PP → Prep NP	1.0		Prep → from | to | on | near | through
			0.25 0.25 0.1 0.2 0.2

Adapted from Rusell, S., & Norvig, P. (2016)

- Assume productions for each node are chosen independently.
- Probability of derivation is the product of the probabilities of its productions.

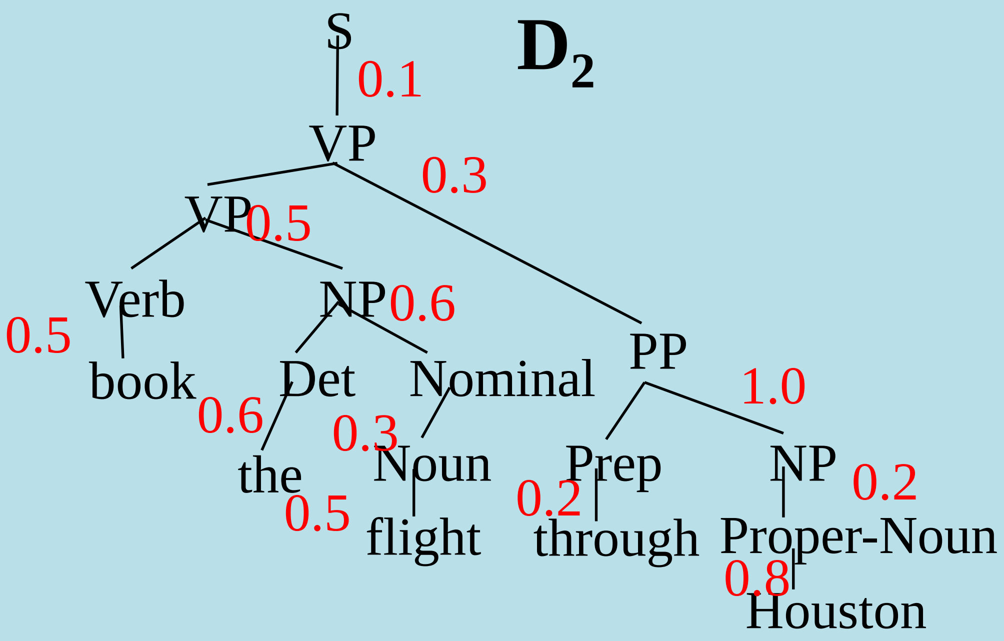
$$P(D_1) = 0.1 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times 0.5 \times 0.3 \times 1.0 \times 0.2 \times 0.2 \times 0.5 \times 0.8 = 0.0000216$$



Adapted from Rusell, S., & Norvig, P. (2016)

- Resolve ambiguity by picking most probable parse tree.

$$P(D_2) = 0.1 \times 0.3 \times 0.5 \times 0.6 \times 0.5 \times 0.6 \times 0.3 \times 1.0 \times 0.5 \times 0.2 \times 0.2 \times 0.8 = 0.00001296$$



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Sentence Probability

- Probability of a sentence is the sum of the probabilities of all of its derivations.

P("book the flight through Houston") =
$P(D_1) + P(D_2) = 0.0000216 + 0.00001296$
$= 0.00003456$

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Three Useful PCFG Tasks

- Observation likelihood: To classify and order sentences.
- Most likely derivation: To determine the most likely parse tree for a sentence.
- Maximum likelihood training: To train a PCFG to fit empirical training data.

Adapted from Rusell, S., & Norvig, P. (2016)

- What is the probability that a given string is produced by a given PCFG

- Can use a PCFG as a language model to choose between alternative sentences for speech recognition or machine translation

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

O_1

? The dog big barked.

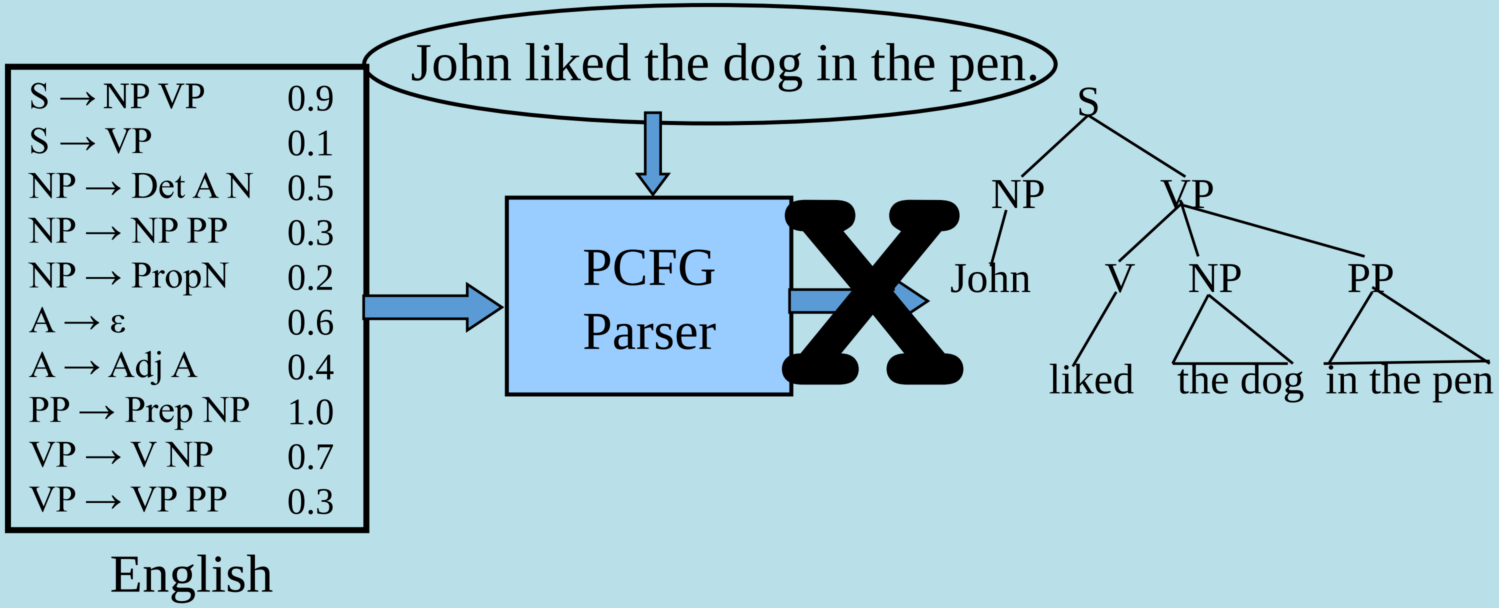
? The big dog barked

O_2

$P(O_2 \mid \text{English}) > P(O_1 \mid \text{English})$?

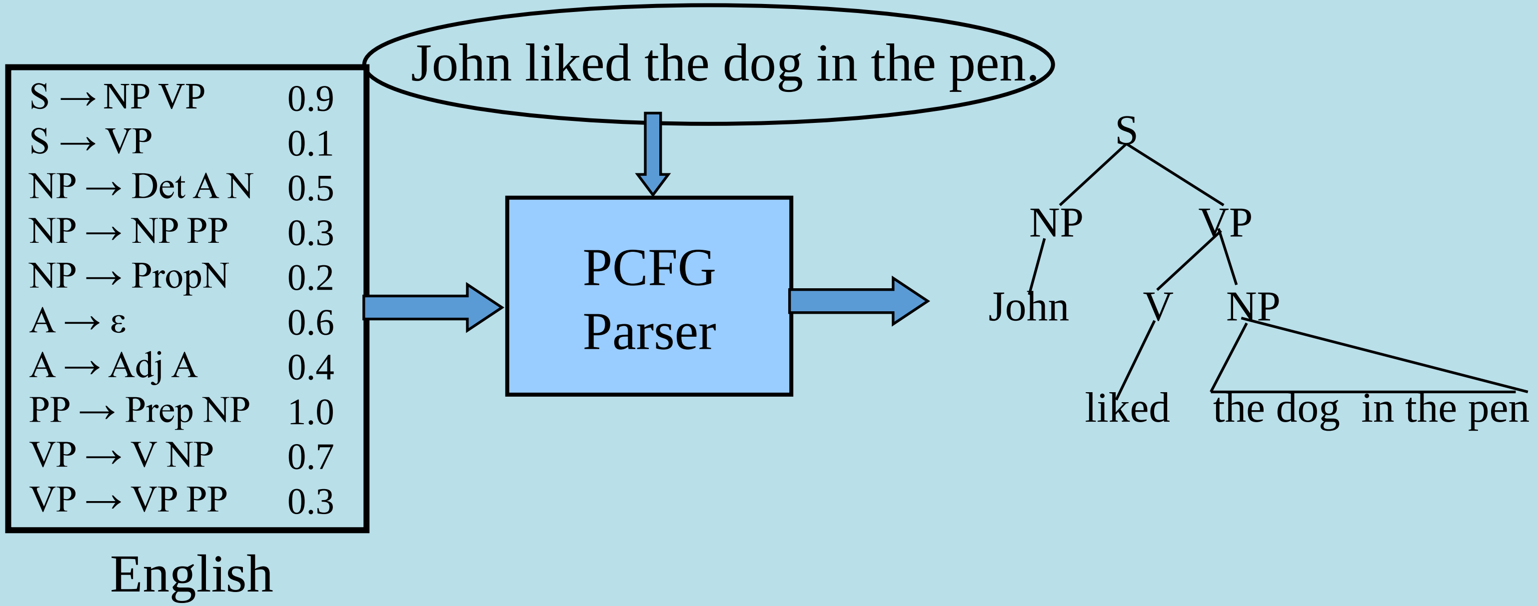
Adapted from Rusell, S., & Norvig, P. (2016)

- What is the most probable derivation (parse tree) for a sentence.



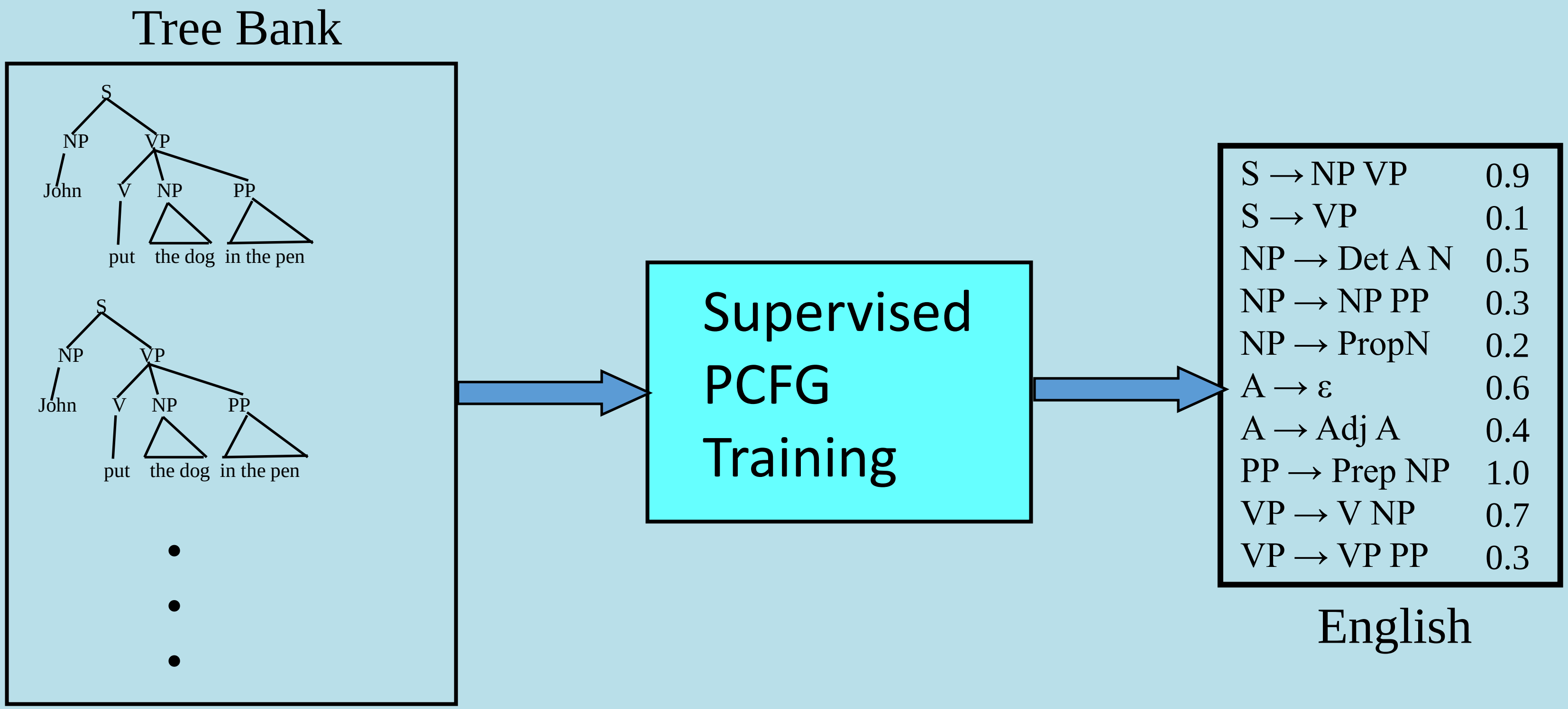
Adapted from Rusell, S., & Norvig, P. (2016)

- What is the most probable derivation (parse tree) for a sentence.



Adapted from Rusell, S., & Norvig, P. (2016)

- If parse trees are provided for training sentences, a grammar and its parameters can be can all be estimated directly from counts accumulated from the tree-bank (with appropriate smoothing).



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Estimating Production Probabilities

- Set of production rules can be taken directly from the set of rewrites in the treebank.
- Parameters can be directly estimated from frequency counts in the treebank.

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

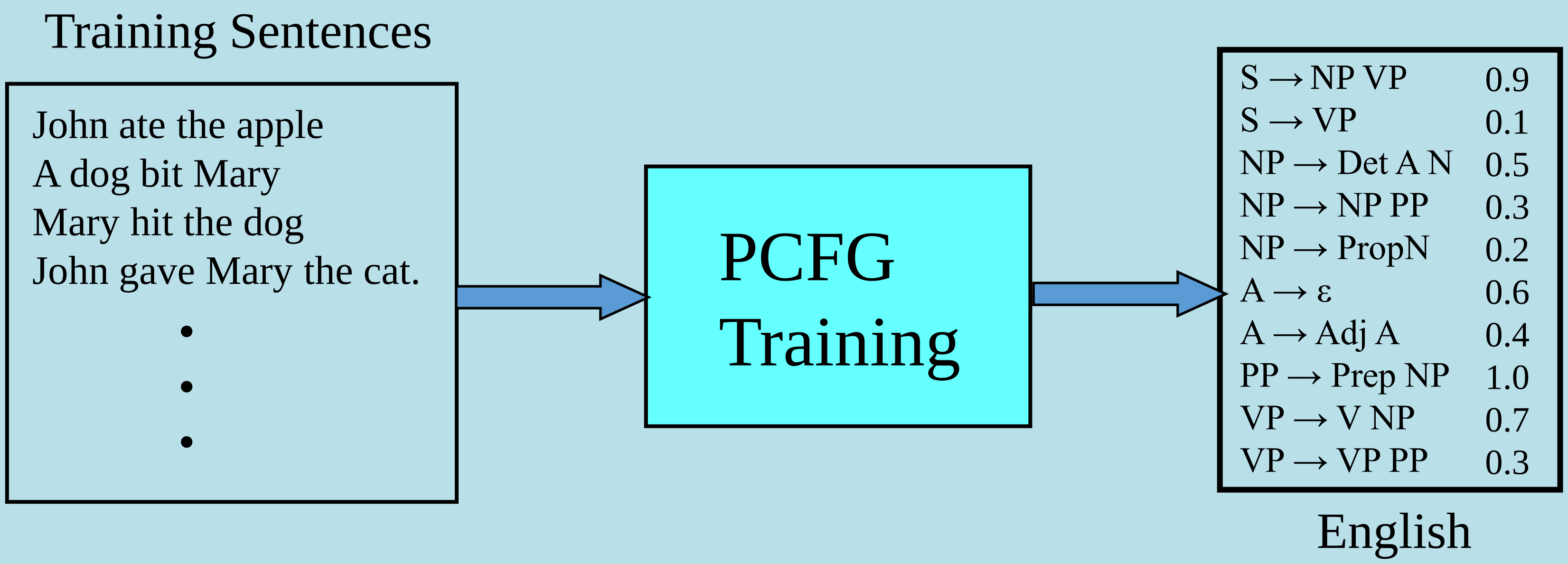
Adapted from Rusell, S., & Norvig, P. (2016)

9.2 PCFG: Maximum Likelihood Training

- Given a set of sentences, induce a grammar that maximizes the probability that this data was generated from this grammar.
- Assume the number of non-terminals in the grammar is specified.
- Only need to have an unannotated set of sequences generated from the model. Does not need correct parse trees for these sentences. In this sense, it is unsupervised.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 PCFG: Maximum Likelihood Training



Adapted from Rusell, S., & Norvig, P. (2016)

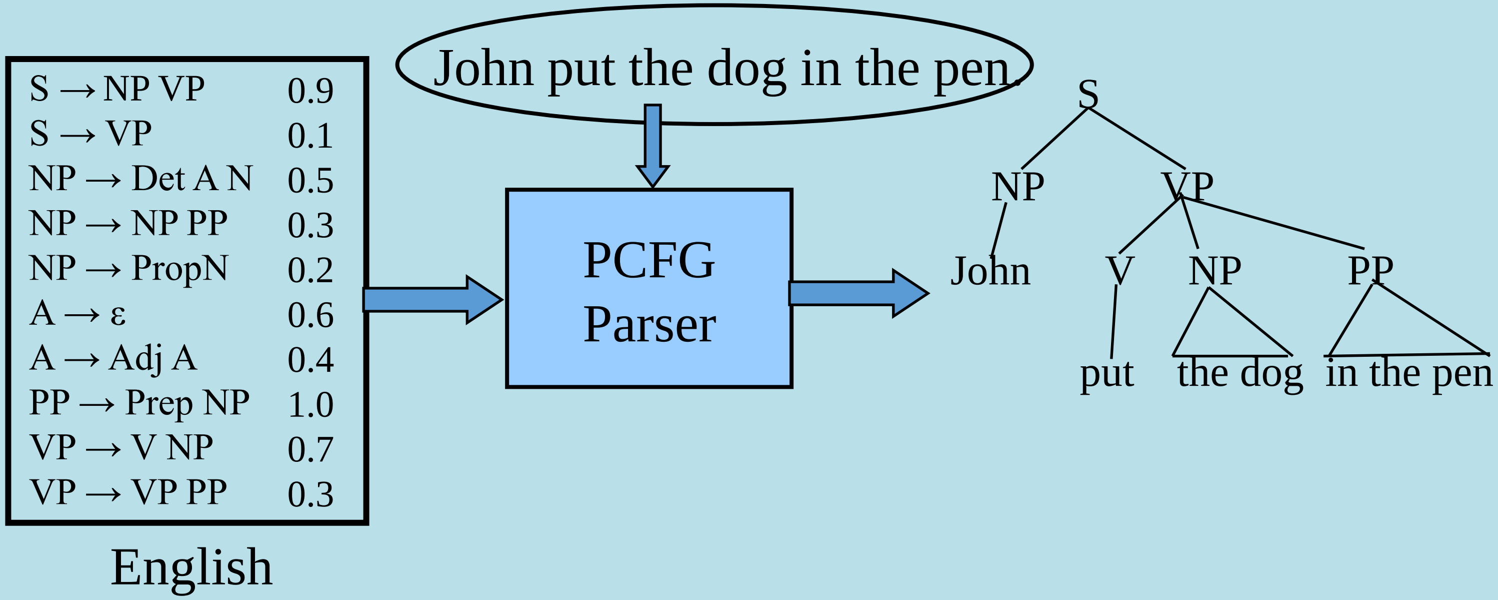
9.2 Vanilla PCFG Limitations

- Since probabilities of productions do not rely on specific words or concepts, only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals).
- Consequently, vanilla PCFGs cannot resolve syntactic ambiguities that require semantics to resolve, e.g. ate with fork vs. meatballs.
- In order to work well, PCFGs must be lexicalized, i.e. productions must be specialized to specific words by including their head-word in their LHS non-terminals (e.g. VP-ate).

Adapted from Rusell, S., & Norvig, P. (2016)

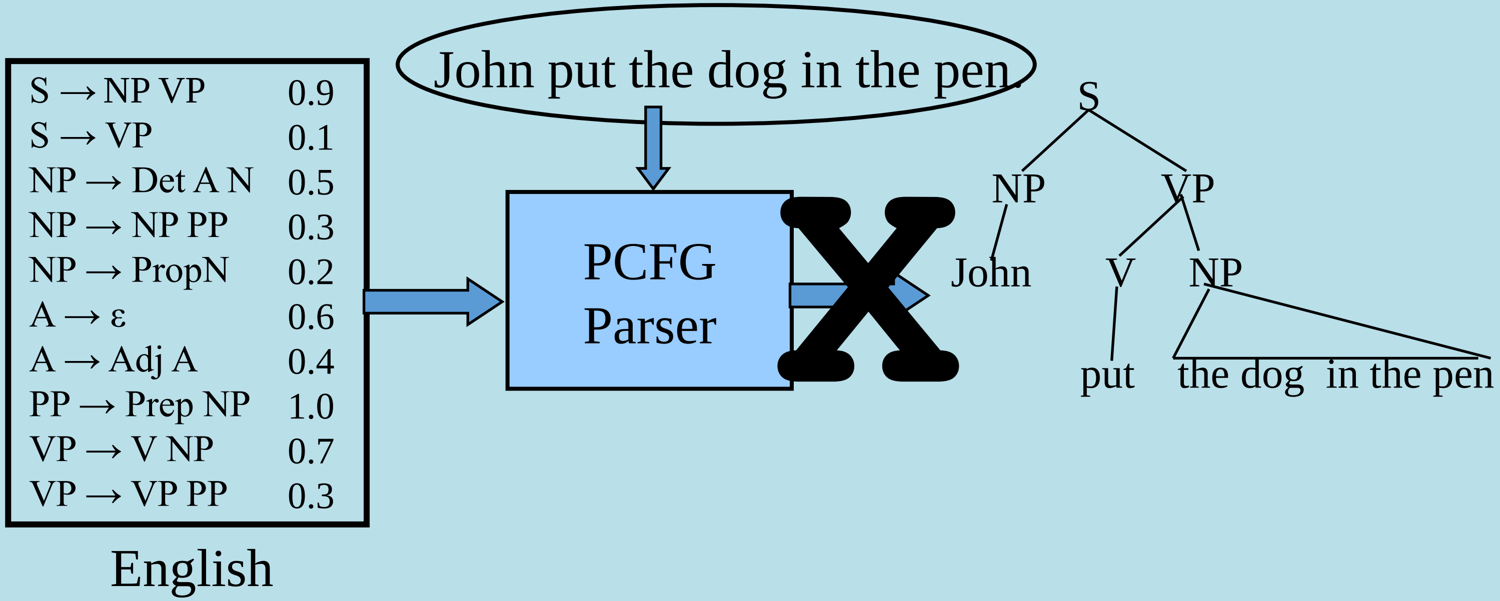
9.2 Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.



Adapted from Rusell, S., & Norvig, P. (2016)

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.



Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Treebanks

- English Penn Treebank: Standard corpus for testing syntactic parsing consists of 1.2 M words of text from the Wall Street Journal (WSJ).
- Typical to train on about 40,000 parsed sentences and test on an additional standard disjoint test set of 2,416 sentences.
- Chinese Penn Treebank: 100K words from the Xinhua news service.
- Other corpora existing in many languages, see the Wikipedia article “Treebank”

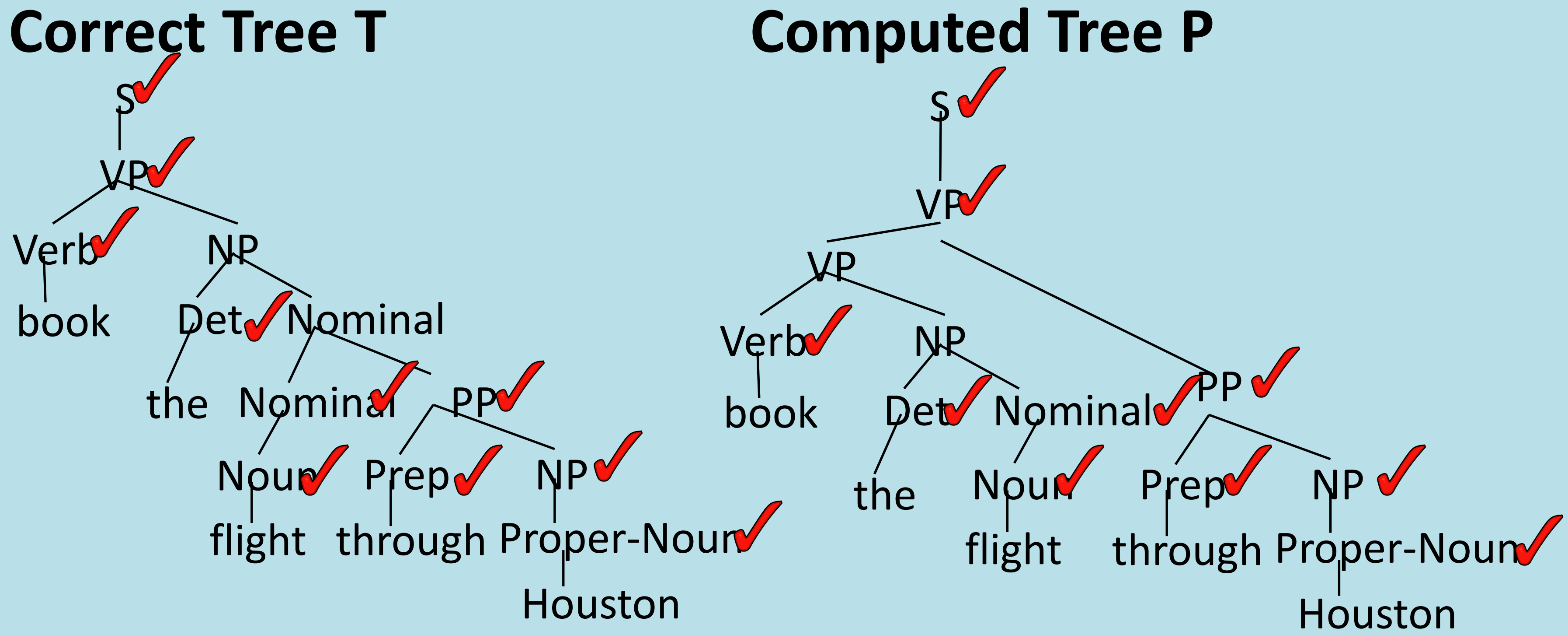
Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Parsing Evaluation Metrics

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If P is the system's parse tree and T is the human parse tree (the "gold standard"):
  - Recall = (# correct constituents in P) / (# constituents in T)
  - Precision = (# correct constituents in P) / (# constituents in P)

- Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.
- F1 is the harmonic mean of precision and recall.

Adapted from Rusell, S., & Norvig, P. (2016)



Constituents: 12

Constituents: 12

Correct Constituents: 10

Recall = 10/12= 83.3%   Precision = 10/12=83.3% F_1 = 83.3%

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Treebank Results

- Results of current state-of-the-art systems on the English Penn WSJ treebank are slightly greater than 90% labeled precision and recall.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Word Sense Disambiguation (WSD)

- Words in natural language usually have a fair number of different possible meanings.
  - Ellen has a strong interest in computational linguistics.
  - Ellen pays a large amount of interest on her credit card.

- For many tasks (question answering, translation), the proper sense of each ambiguous word in a sentence must be determined.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Ambiguity Resolution is Required for Translation

- Syntactic and semantic ambiguities must be properly resolved for correct translation:
  - “John plays the guitar.” → “John toca la guitarra.”
  - “John plays soccer.” → “John juega el fútbol.”

- An apocryphal story is that an early MT system gave the following results when translating from English to Russian and then back to English:
  - “The spirit is willing but the flesh is weak.” → “The liquor is good but the meat is spoiled.”
  - “Out of sight, out of mind.” → “Invisible idiot.”

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Word Sense Disambiguation (WSD) as Text Categorization

- Each sense of an ambiguous word is treated as a category.
  - “play” (verb)
    - play-game
    - play-instrument
    - play-role
  - “pen” (noun)
    - writing-instrument
    - enclosure

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Word Sense Disambiguation (WSD) as Text Categorization

- Treat current sentence (or preceding and current sentence) as a document to be classified.
  - "play":
    - play-game: "John played soccer in the stadium on Friday."
    - play-instrument: "John played guitar in the band on Friday."
    - play-role: "John played Hamlet in the theater on Friday."
  - "pen":
    - writing-instrument: "John wrote the letter with a pen in New York."
    - enclosure: "John put the dog in the pen in New York."

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Learning for WSD

- Assume part-of-speech (POS), e.g. noun, verb, adjective, for the target word is determined.
- Treat as a classification problem with the appropriate potential senses for the target word given its POS as the categories.
- Encode context using a set of features to be used for disambiguation.
- Train a classifier on labeled data encoded using these features.
- Use the trained classifier to disambiguate future instances of the target word given their contextual features.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 WSD “line” Corpus

- 4,149 examples from newspaper articles containing the word “line.”
- Each instance of “line” labeled with one of 6 senses from WordNet.
- Each example includes a sentence containing “line” and the previous sentence for context.

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Senses of “line”

- Product: “While he wouldn’t estimate the sale price, analysts have estimated that it would exceed $1 billion. Kraft also told analysts it plans to develop and test a line of refrigerated entrees and desserts, under the Chillery brand name.”
- Formation: “C-LD-R L-V-S V-NNA reads a sign in Caldor’s book department. The 1,000 or so people fighting for a place in line have no trouble filling in the blanks.”
- Text: “Newspaper editor Francis P. Church became famous for a 1897 editorial, addressed to a child, that included the line “Yes, Virginia, there is a Santa Clause.”
- Cord: “It is known as an aggressive, tenacious litigator. Richard D. Parsons, a partner at Patterson, Belknap, Webb and Tyler, likes the experience of opposing Sullivan & Cromwell to “having a thousand-pound tuna on the line.”
- Division: “Today, it is more vital than ever. In 1983, the act was entrenched in a new constitution, which established a tricameral parliament along racial lines, whith separate chambers for whites, coloreds and Asians but none for blacks.”
- Phone: “On the tape recording of Mrs. Guba's call to the 911 emergency line, played at the trial, the baby sitter is heard begging for an ambulance.”

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Experimental Data for WSD of “line”

- Sample equal number of examples of each sense to construct a corpus of 2,094.
- Represent as simple binary vectors of word occurrences in 2 sentence context.
  - Stop words eliminated
  - Stemmed to eliminate morphological variation
- Final examples represented with 2,859 binary word features.

Adapted from Rusell, S., & Norvig, P. (2016)

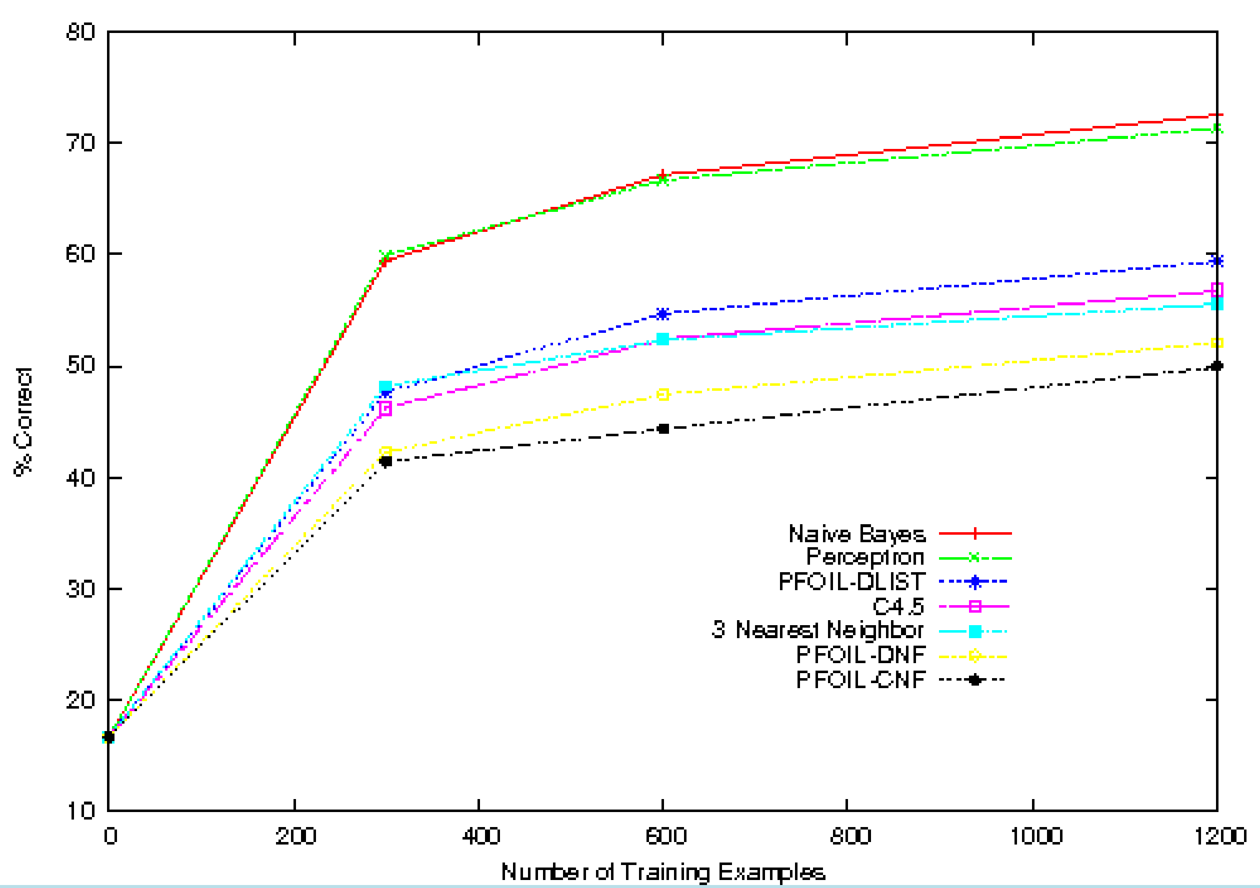
9.2 Machine Learning Algorithms

- Naïve Bayes
  - Binary features
- K Nearest Neighbor
  - Simple instance-based algorithm with k=3 and Hamming distance
- Perceptron
  - Simple neural-network algorithm.
- C4.5
  - State of the art decision-tree induction algorithm
- PFOIL-DNF
  - Simple logical rule learner for Disjunctive Normal Form
- PFOIL-CNF
  - Simple logical rule learner for Conjunctive Normal Form
- PFOIL-DLIST
  - Simple logical rule learner for decision-list of conjunctive rules

Adapted from Rusell, S., & Norvig, P. (2016)

9.2 Learning Curves for WSD of “line”

- Naïve Bayes and Perceptron give the best results.
- Both use a weighted linear combination of evidence from many features.
- Symbolic systems that try to find a small set of relevant features tend to overfit the training data and are not as accurate.
- Nearest neighbor method that weights all features equally is also not as accurate.
- Of symbolic systems, decision lists work the best.



Adapted from Rusell, S., & Norvig, P. (2016)

Outline

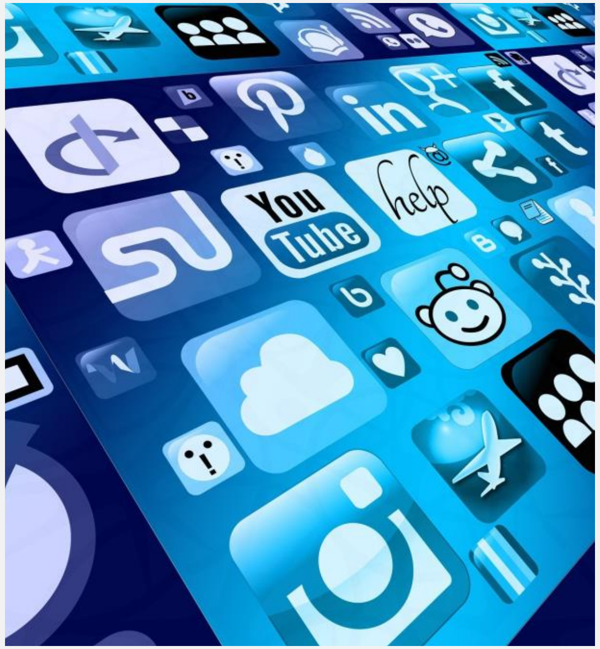


Image source: ↗ Pixabay (2021) / ↗ CC0

▶ **What we will learn:**

- We will discuss how to make use of the copious knowledge that is expressed in natural language
- And how to process and prepare language data for machines to build state-of-the-art machine learning pipelines for language processing
- We learn a new type of models, so termed generative models, which allow to generate new instances based on the knowledge of your model

▶ **Duration:**

- 135 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 9.1, 9.4

Outline

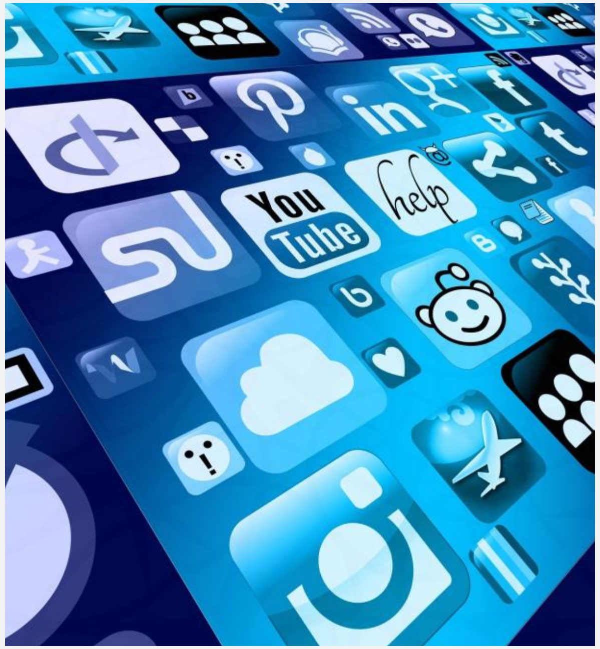


Image source: ↗ Pixabay (2021) / ↗ CC0

▶ **What we will learn:**

- We will discuss how to make use of the copious knowledge that is expressed in natural language
- And how to process and prepare language data for machines to build state-of-the-art machine learning pipelines for language processing
- We learn a new type of models, so termed generative models, which allow to generate new instances based on the knowledge of your model

▶ **Duration:**

- 135 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 9.1, 9.4

9.4 Let us Look behinde to Lecture 1

Autonomous Driving (VW)

Image source: ↗ DB2018AL00555 (VW) | free for editorial purposes

AI model autonomous driving

- **Task**: Driving on public, highway using vision sensors
- **Measure**: Distance before error
- **Training**: sequence of images/video data of human drivers

D

Machine Learning

A computer program is said to learn from experience ‘E’, with respect to some class of tasks ‘T’ and performance measure ‘P’ if its performance at tasks in ‘T’ as measured by ‘P’ improves with experience ‘E’. (Mitchel, 2011)

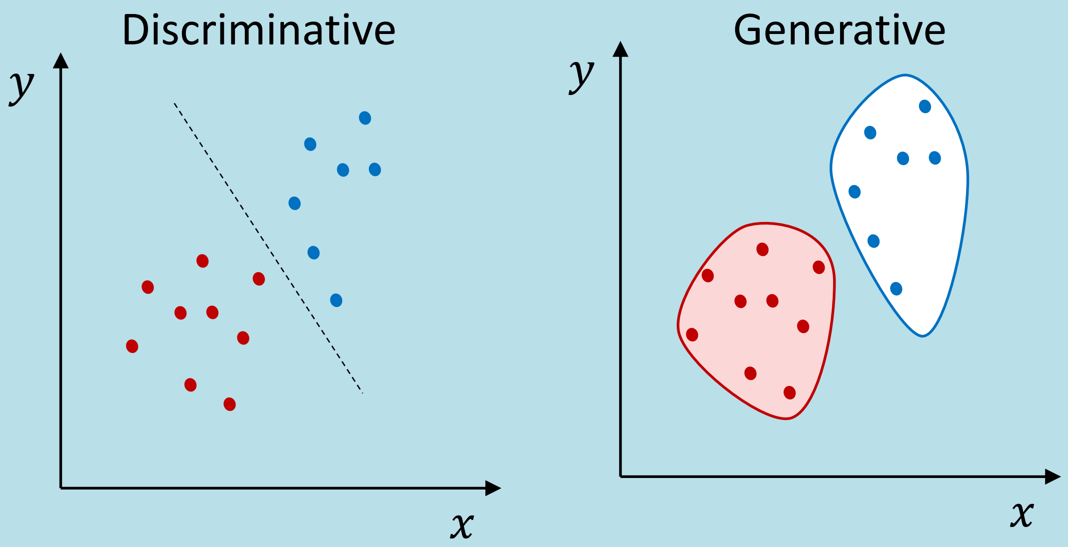
Image source: ↗ Richard Feynman, 1984 (1984) by Tamiko Thiel ↗ CC BY-SA 3.0

„What I cannot create I do not understand“

Richard Feynman

9.4 Problem: High-Performance vs. Understanding

Image source: ↗ DB2018AL00555 (VW) | free for editorial purposes



Example: Pattern recognition vs. generation in autonomous driving

- Sometimes we have to understand/model the true distribution of our dependent variable
- We want to generate new data instances

Adapted from Ng AY & Jordan MI (2002)

9.4 Formalization: Discriminative Models vs. Generative Models

- We defined machine learning as:

$$f: X \rightarrow Y$$

- To predict the class Y from the training example X in machine learning problem (e.g. classification with decision tree or perceptron), we have to solve

$$f(X) = \arg\max_{y} p(Y\,|X)$$

- We try to model this conditional probability by modelling a decision boundy between the classes

Adapted from Ng AY & Jordan MI (2002)

9.4 Formalization: Discriminative Models vs. Generative Models

- If we use Bayes‘ rule we can replace $p(Y\,|X)$, and get

$$f(X) = \arg\max_y \frac{p(X\,|Y) \cdot p(Y)}{p(X)} \approx \arg\max_y p(X\,|Y) \cdot p(Y)$$

- With

$$p(X\,|Y) \cdot p(Y) = p(X\,|Y)$$

- Hence, our problem describes now the <u>joint</u> probability distribution $p(X\,|Y)$, which models the actual distribution of each class
- These models are called generative, because they allow to calculate/generate the respective X for each Y

Adapted from Ng AY & Jordan MI (2002)

8.6 Taxonomy: Discriminative vs. Generative in Machine Learning

Supervised

An algorithm uses human-prepared training data to learn the relationship between given inputs and a given outcome.

- **Discriminative**
  - Classification (discrete)
    - Logistic regression
    - SVM
    - Instance based /decision trees
  - Regression (continuous)
    - Linear regression
    - Generalized linear models
- **Generative**
  - Sequence
    - Markov Process, Markov Chain
    - Hidden Markov Model
  - Random
    - Naive Bayes
    - Latent Dirichlet Allocation

Unupervised

An algorithm examines input data without knowing about attributes and possible results.

- **Discriminative**
  - Clustering
    - K-means
  - Dimensionality Reduction
    - Principal Component Analysis
    - Linear Discriminant Analysis
- **Generative**
  - Density Estimation
    - GAN (e.g. LDA)
    - Variational Autoencoder (VAE)

9.4 ThisPersonDoesNotExist.com


ABO STERN PLUS CRIME PLUS

stern

VIDEO SPIELE GUTSCHEINE

Panorama Politik Kultur Lifestyle Digital Wirtschaft Sport Gesundheit Genuss Reise Familie Gutscheine

Home > Neon > Wilde Welt > thispersondoesnotexist.com zeigt Menschen, die es gar nicht gibt

18. Februar 2019 17:23 Uhr

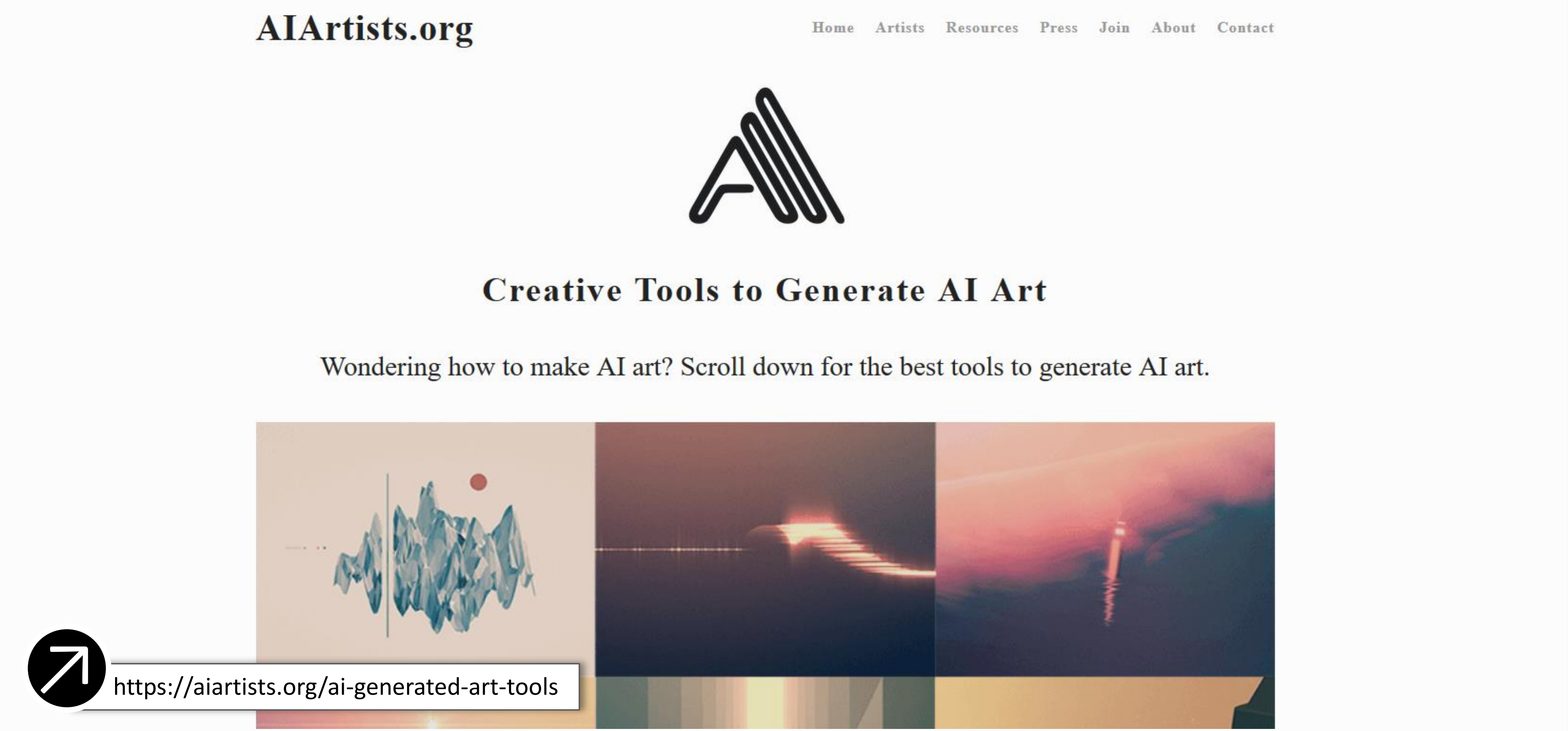
Künstliche Intelligenz: Diese Seite zeigt Menschen, die es gar nicht gibt – und das ist ziemlich gruselig

Achtung: Was ihr seht, ist nicht echt. Mit künstlicher Intelligenz können Maschinen nämlich unendlich viele Fake-Gesichter erstellen. Die Ergebnisse der beeindruckenden und zugleich gruseligen Technologie zeigen die Entwickler jetzt im Netz.

ThisPersonDoesNotExist.com

Adapted from Ng AY & Jordan MI (2002)

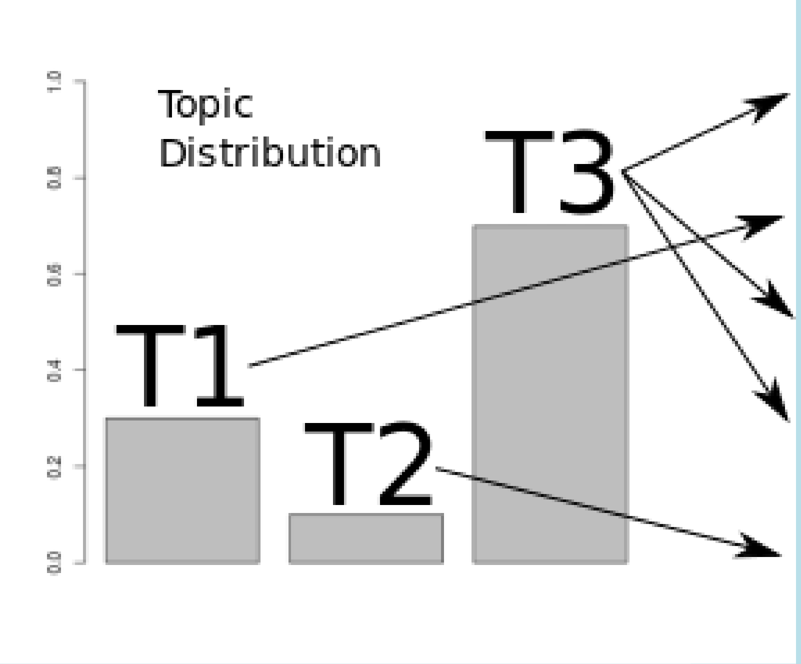
9.4 Create Your own AI Art



https://aiartists.org/ai-generated-art-tools

9.4 Latent Dirichlet Allocation (LDA)

▶ Topic model approaches to analyze unstructured data to find hidden topics



- One of the most popular and simplest topic model approaches to analyze unstructured
- Latent Dirichlet allocation is not the best but todayby far the most popular approach.
- The method is based on the work of Deerwester in latent semantic indexing and of Hofmann in probabilistic latent semantic indexing

Example

- Automated Summarization, Mediarecommender

\+ probabilistic model with interpretable topics.

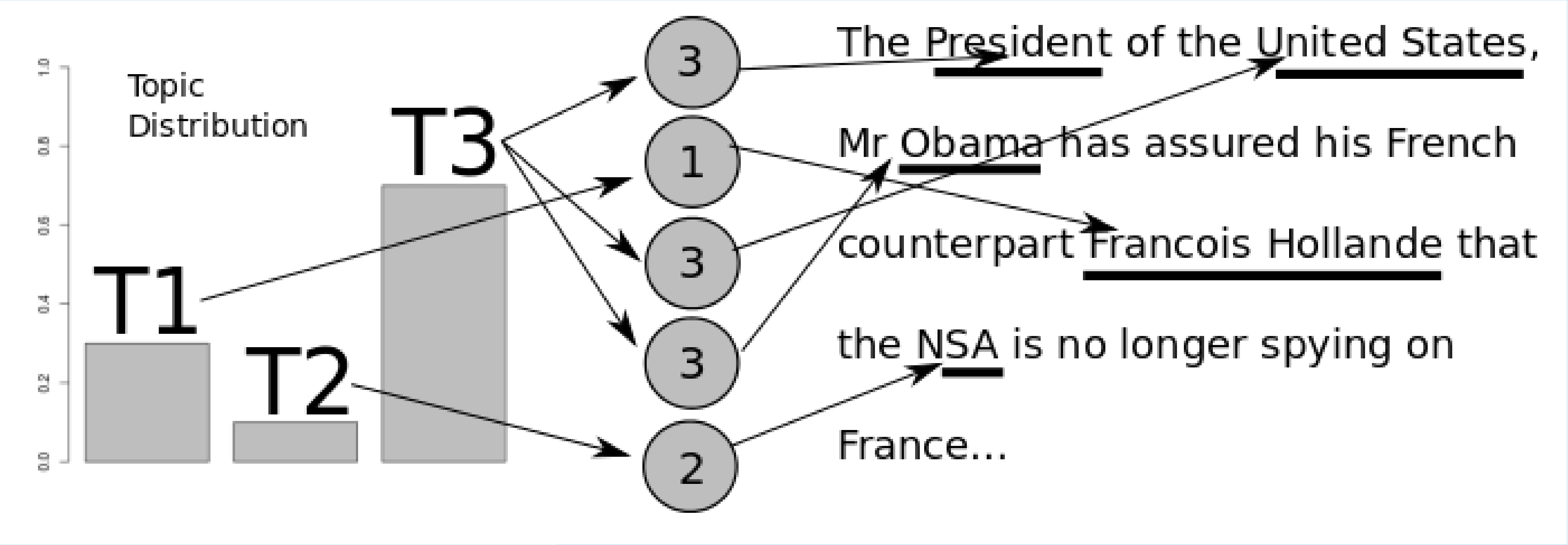
\- hard to know when LDA is working, because topics are soft-clusters so there is no objective metric

9.4 Latent Dirichlet Allocation

- Idea of latent Dirichlet allocation is to understand the document creation as a stochastic process (see chapter 8)
- It is assumed that a document w_j is generated by an author who selects words of a vocabulary V with a given probability from different baskets of words where each basket corresponds to one of k topics
- The created texts are the observed variables, while the topics, the distribution of the topics and the assignments per document and word is hidden. With that assumptions, the purpose is to decompose from a document with $i\ words, w = (w_1; w_2; \dots; w_i)$ the original topic baskets

Adapted from David M. Blei et al. (2006)

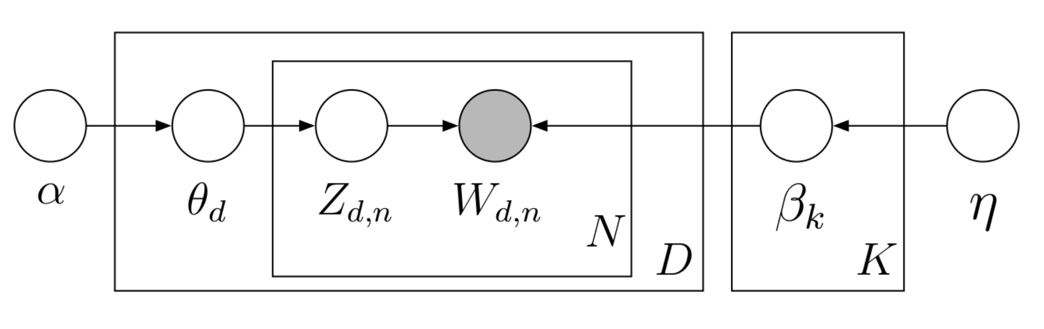
9.4 Topics in LDA



- Each topic contains a individual basket of words and is chosen by his probability (the circles with numbers)
- Each time a topic is chosen, a word ofhis basket is drawn by his probability

Adapted from David M. Blei et al. (2006)

9.4 Latent Dirichlet Allocation



$$p(\vec{\theta}_{1:D}, z_{1:D,1:n}, \vec{\beta}_{1:K} | \omega_{1:D.1:n}, \alpha, \eta) =$$

$$\frac{p(\vec{\theta}_{1:D}, \vec{z}_{1:D,}, \vec{\beta}_{1:K} | \omega_{1:\vec{D}.1:n}, \alpha, \eta) =}{\int_{\vec{\beta}_{1:K}} \int_{\vec{\theta}_{1:D}} p(\vec{\theta}_{1:D}, \vec{z}_{1:D,}, \vec{\beta}_{1:K} | \omega_{1:\vec{D}.1:n})}$$

Algorithm: Generative LDA Process

for each topic in $number\ of\ topics\ k$ **do**
  Draw a distribution over words $\beta_k \sim Dir(\eta)$
end for

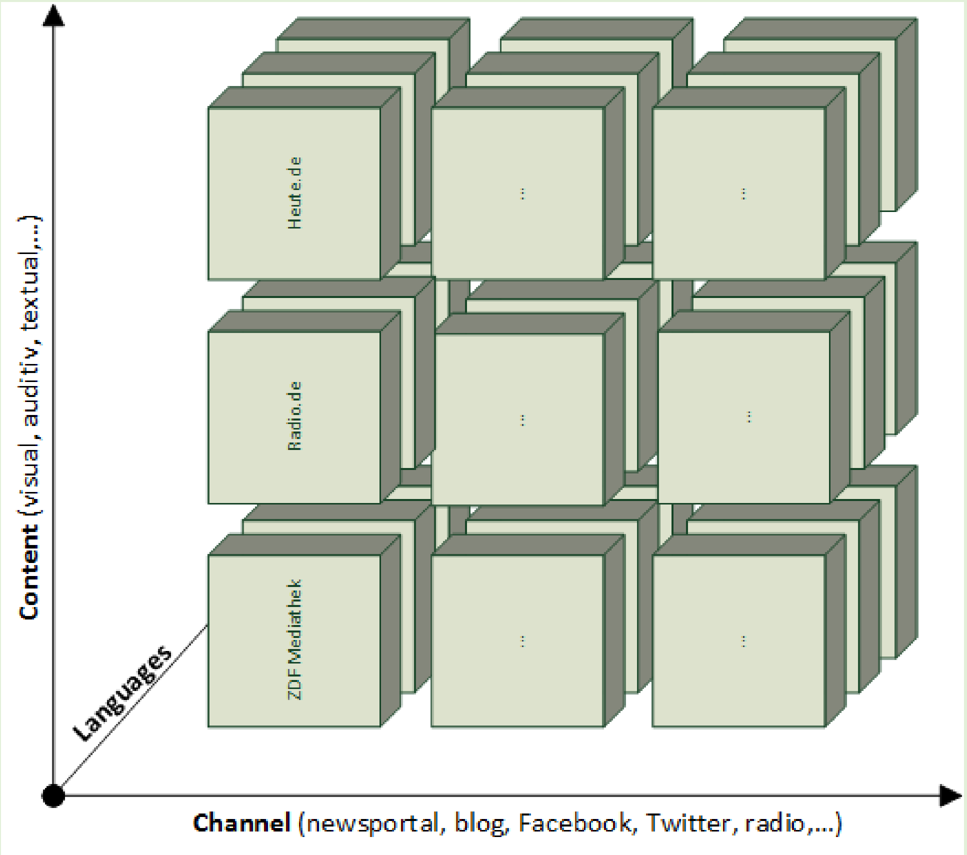
for each document $w_n = d\ in\ corpus\ D$ **do**
  Draw a vector of topic proportions $\vec{\theta}_d \sim Dir(\alpha)$
  for each word w_n^i **do**
  Draw a topic assignment $Z_{d;n}\ Mult(\vec{\theta}_d)_{Z_{d,n} \in [1 \ldots K]}$
  Draw a word $w_{d;n}\ Mult(\vec{\beta}_{Z_{d,n}})_{w_{d,n} \in [1 \ldots K]}$
  end for
end for

Adapted from David M. Blei et al. (2006)

Business Case: Topic Modelling for Multi-Modal Recommendations

Multi-Modal AI is hard

- EU is one of the multilingual markets, with more than 24 official languages
- Today there exists different media items such as email, blog, books, journals, articles in multiple media formats
- Challenge: very heterogeneous media and information. Hence, task as information retrieval, searching, monitoring, recommendations, classication, evaluation, translation, summarization or further media analysis pose a major challenge.



- Mogadala, A., Jung, D., & Rettinger, A. (2017). Linking tweets with monolingual and cross-lingual news using transformed word embeddings. 18th International Conference on Intelligent Text Processing and Computational Linguistics

Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

- Corpus is a collection of n distinct textual media items and is denoted by

$$\mathcal{D} = \{ w^1, w^2, \dots, w^n\}_{n \in \mathbb{N}}$$

- Document of this corpus is a sequence of words:

$$w^j = (w_1^j, w_2^j, \dots, w_i^j)$$

- Aim to find a semantic similarity function $sim()$ or a dissimilarity function $dis()$ to identify relatedness between items. Such a function takes the entities as an input and outputs a score, which indicates the semantic relatedness of the input entity sets.

$$\forall w^j, w^k \in \mathcal{D}: sim\left(w^j, w^k\right) = s_{j,k} \in \mathbb{R}_{[0,1]}$$

Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

- Dissimilarity function $dis()$ can be expressed depending on the similarity function $sim()$

$$dis(w^j, w^k) = 1 - sim(w^j, w^k)$$

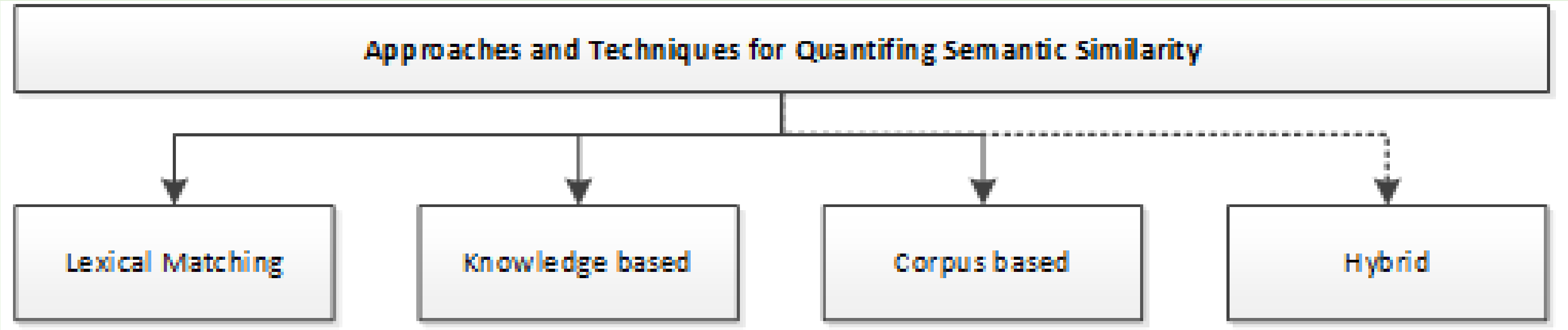
range [0,1], 0: no similarity

- **Multi-Modal Recommender**: Find the entity combination of the corpus with the highest similarity

$$\text{Most similar doc pair } (w^j, w^k)^* = \arg\max_{j,k} sim(w^j, w^k), s.t.\, w^j, w^k \in \mathcal{D}$$

Adapted from Mogadala et al. (201/)

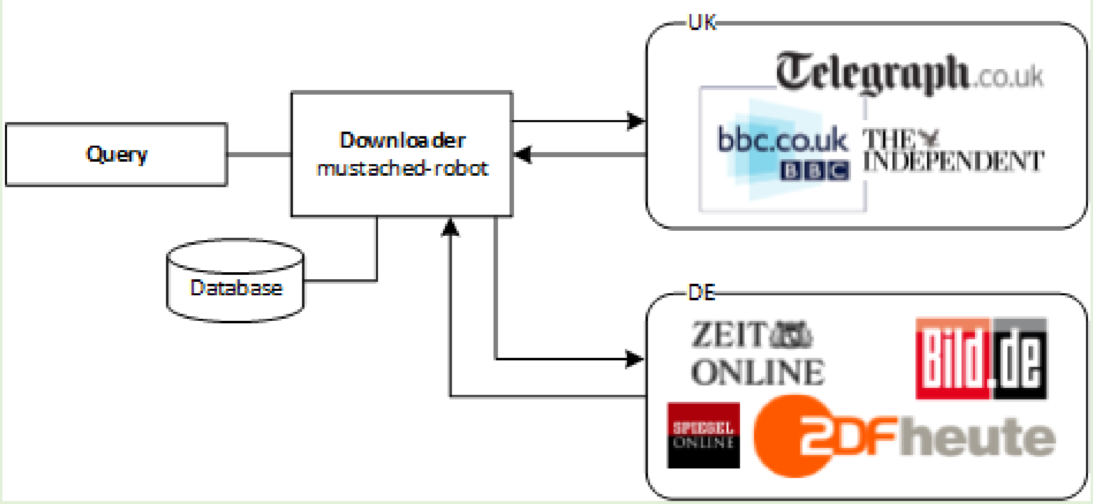
Business Case: Topic Modelling for Multi-Modal Recommendations



- **Lexical matching methods** include methods which aim to calculate a similarity score by counting lexical units that exits in the input texts.
- **Corpus-based algorithms** calculate by means of large corpora information for a similarity score
- **Knowledge-based** measures calculate a similarity score by means of large networks like Word-Net

Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations



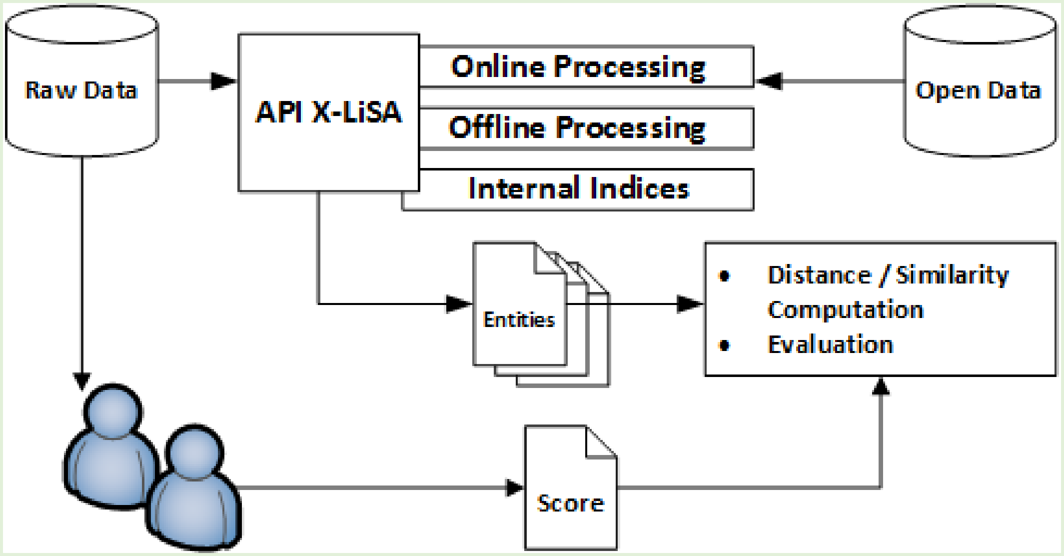
	Dataset 1	Dataset 2
Language of the corpus	Cross-lingual	Cross-lingual
Heterogeneity of the corpus	1 corpus topic	2 corpus topics
Noise of the corpus	Cross-channel	Mono-channel

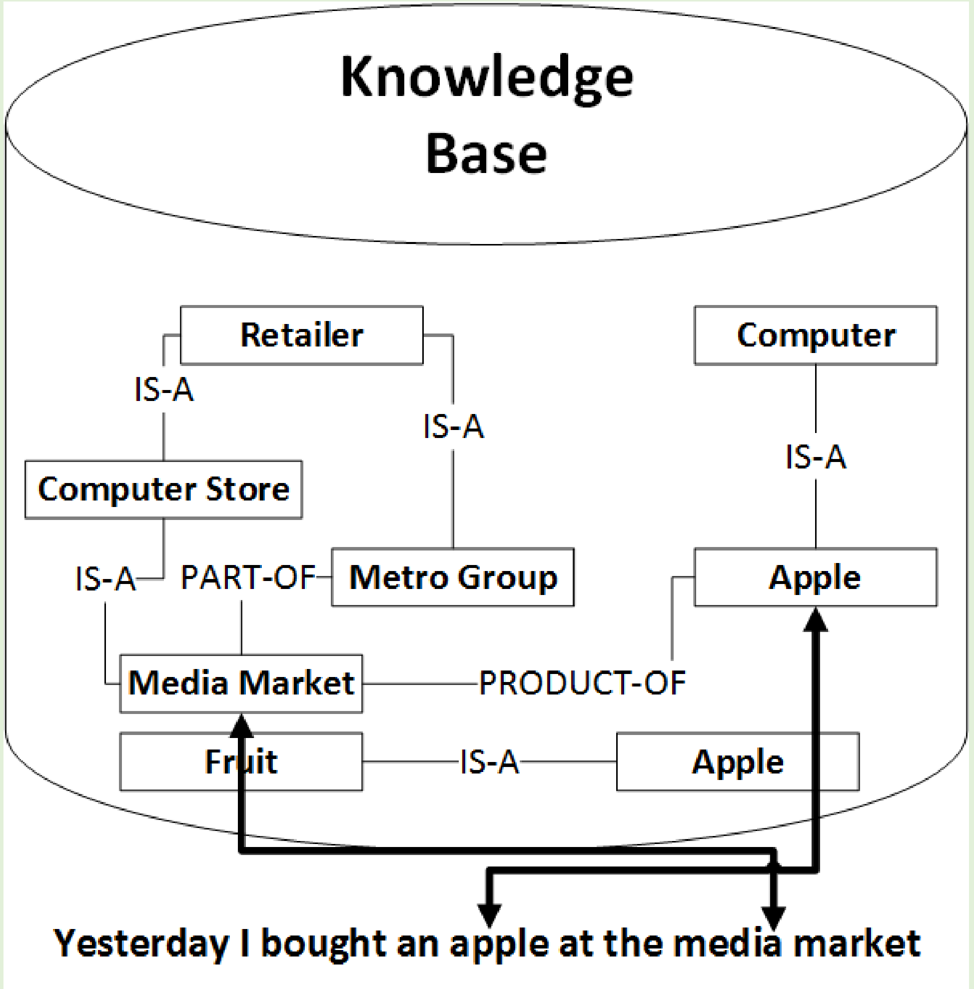
- To have comparable datasets, all the documents were crawled by two dierent keywords "Grexit" (the greek exit of the EU) and "4U9525" (airplain crash in france) from the newsportal from January 2015 until May 2015
- To evaluate our model we conducted a user study. Each participant had to rate all combinations of articles for semantic similarity pairwise

Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

Entity Detection

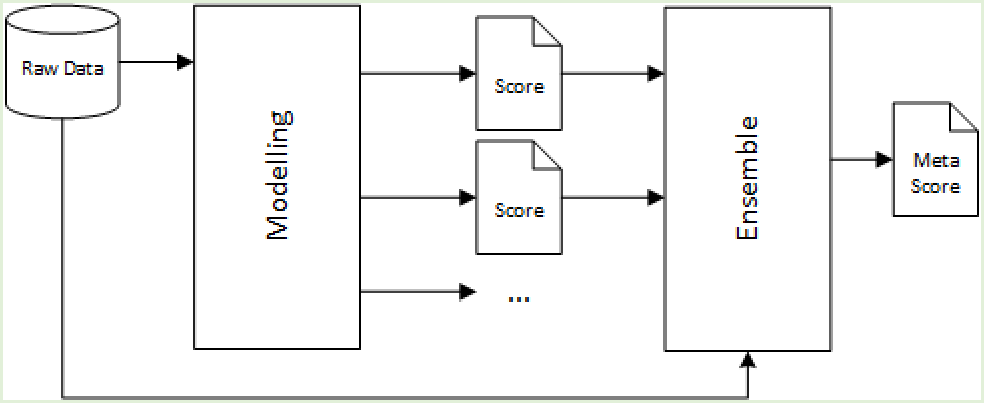




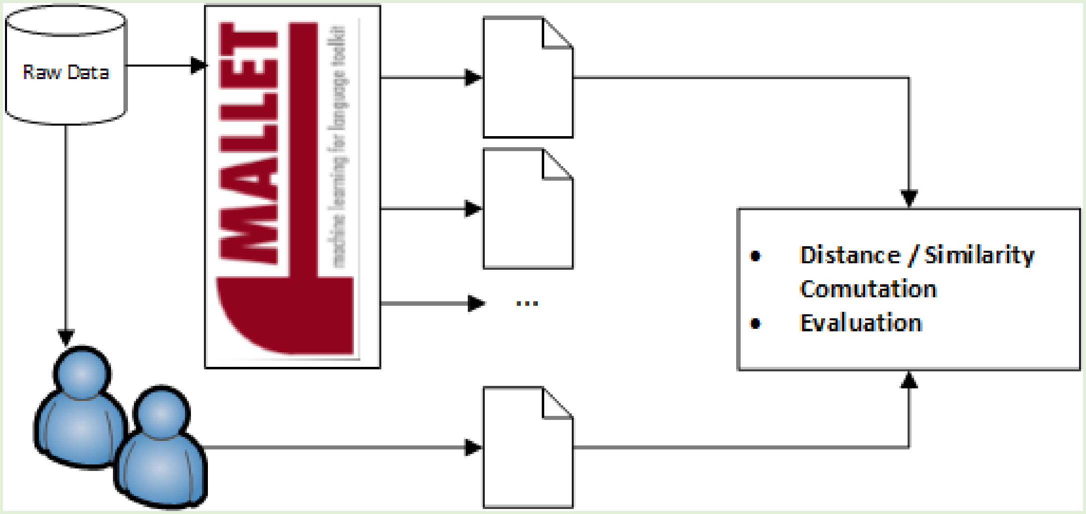
Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

Ensemble



Topic Modelling



Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

	Accuracy	
Method and similarity function	**Dataset 1**	**Dataset 2**
Topic modelling: Euclidean	28.4%	27.72%
Topic modelling: Hellinger	33.64%	37.72%
Entity linking: Subset	33.95%	91.01%
Entity linking: Jaccard	35.18%	94.58%
Ensemble learning algorithm	**Dataset 1**	**Dataset 2**
Ranking learning	66.98%	95.92%
C4.5	64.52%	94.94%
Boosting	92.59%	95.92%
Baseline	**Dataset 1**	**Dataset 2**
Random generated scores	76.23%	95.92%

- Approaches are able to detect the structure of the corpora
- During this study, other approaches in comparable experimental studies were identified and report performances between 51.9% and 82.5%
- Other problems: Noisy data (Twitter!), and high dimensionality of the problem (Transformation)
- New experiments with a more uniform score distribution would be very useful to improve the weaknesses of the models.

Adapted from Mogadala et al. (201/)

Business Case: Topic Modelling for Multi-Modal Recommendations

01 | Executive Summary

Grouping diverse media sources of information that discuss the same topic in varied perspectives are a relevant challenge in online media. But the gap in word usage between informal social media content such as tweets and diligently written content (e.g. news articles) make such assembling difficult. In this paper, we propose a transformation framework to bridge the word usage gap between tweets and online news articles across languages by leveraging their word embeddings. Experimental results show a notable improvement over baselines for monolingual tweets and news articles comparison

	Accuracy	
Method and similarity function	**Dataset 1**	**Dataset 2**
Topic modelling: Euclidean	28.4%	27.72%
Topic modelling: Hellinger	33.64%	37.72%
Entity linking: Subset	33.95%	91.01%
Entity linking: Jaccard	35.18%	94.58%
Ensemble learning algorithm	**Dataset 1**	**Dataset 2**
Ranking learning	66.98%	95.92%
C4.5	64.52%	94.94%
Boosting	92.59%	95.92%
Baseline	**Dataset 1**	**Dataset 2**
Random generated scores	76.23%	95.92%

02 | Solution

- Topic modelling approaches are able to detect the structure of the corpora but best perform in combination with traditional machine learning
- During this study, other approaches in comparable experimental studies were identified and report performances between 51.9% and 82.5%

03 | References

- Mogadala, A., Jung, D., & Rettinger, A. (2017). Linking tweets with monolingual and cross-lingual news using transformed word embeddings. 18th International Conference on Intelligent Text Processing and Computational Linguistics

Take-Aways

- Topic models can be used for similarity ratings and hence, recommendations
- However, the computation of cross-modal similarity is still a challenge

Your turn!

Task

Imagine your task is to classify a text to a language. Which one of the following approaches is a generative one, and which is the discriminative one. Why?

- learning each language, and then classifying it using the knowledge you just gained
- determining the difference in the linguistic models without learning the languages, and then classifying the speech

9. Exercises

Workbook Exercises

- Please read the chapters 22, 23 and 24 of section VI „Communicating, Perceving, and Acting“ from Rusell, S., & Norvig, P. (2016). Then work through the exercises of each chapter.

Coding Exercises

- *There are no coding exercises in this chapter*

9. References

Literature

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, March 2003.
2. Mitchell, T. M. (1997). Machine learning. McGraw Hill.
3. Rusell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach.* Global Edition.

Images

All images that were not marked other ways are made by myself, or licensed ↗CC0 from ↗Pixabay.

9. Glossary


People with no idea about AI, telling me my AI will destroy the world
Me wondering why my neural network is classifying a cat as a dog..
Dog