

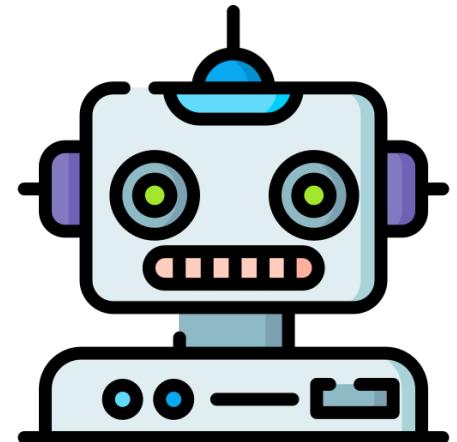
M4: Text Mining Basics

CAS Data Engineering and Applied Data Science 1

Fabian Gröger

Algorithmic Business Research Lab (ABIZ)
Hochschule Luzern
www.abiz.ch

fabian.groeger@hslu.ch



Upcoming Lectures

- 26.05: Supervised and Unsupervised Learning
- 02.06: Supervised and Unsupervised Learning II
- **03.06: Text Mining Basics**

Agenda

- Introduction to Text Analysis
- Text Basics
- Text Similarity
- What is ChatGPT?
- Prompt Engineering
 - Group Work
- Extension + Presentations from "ML in the Wild"

Recap :)

- Regression
 - Motivation
 - How to measure errors?
- Classification
 - Motivation
 - How to measure errors?
 - What to be careful about?
- Evaluation workflows
 - Motivation + Simple workflow
- Clustering
 - Motivation
 - K-Means + Agglomerative clustering
- Dimensionality reduction

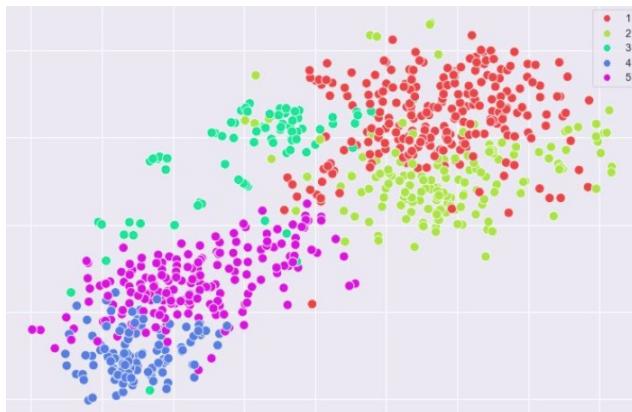
Introduction

What is Text Analysis?

- Series of operations completed by one or more pieces of software on a sample of written human language (text), with the **goal of extracting useful information**
- These analyses can be performed by humans, but in a (much) longer period of time
- Part of the larger field of human language technology (HLT) or **natural language processing (NLP)**
 - Text Analysis: from language to other meaningful information
 - HLT / NLP also include text generation and interactive systems

Teaser

Are you looking for an opportunity to not just develop software, but to help build a system that will utilize **artificial intelligence SKILL** (**AI SKILL**) and **machine learning SKILL** to provide personalized human performance and training feedback for a US Air Force customer? As a **full-stack developer JOBTITLE**, you can solve a problem with a complete end-to-end solution in a fast-paced, **agile environment SKILL**. We're looking for a **developer JOBTITLE** with the skills needed to build an **AI SKILL** and **deep learning SKILL** (DL) solution that will interact with individuals to persist safety, training, proficiency, and tactics for pilots and **intelligence SKILL** professionals, from vision to production-ready system.



Applications of Text Analysis

ANALYSIS

Spell checkers, incl. grammar
Keyword extraction
Authorship attribution
Document retrieval
Text classification
Text mining
Sentiment analysis
Content-based recommendation

TEXT ANALYSIS + GENERATION

Machine translation
Automatic question answering
Automatic summarization

TEXT GENERATION

Generating reports from databases
Generating weather forecasts

ANALYSIS + GENERATION + INTERACTION

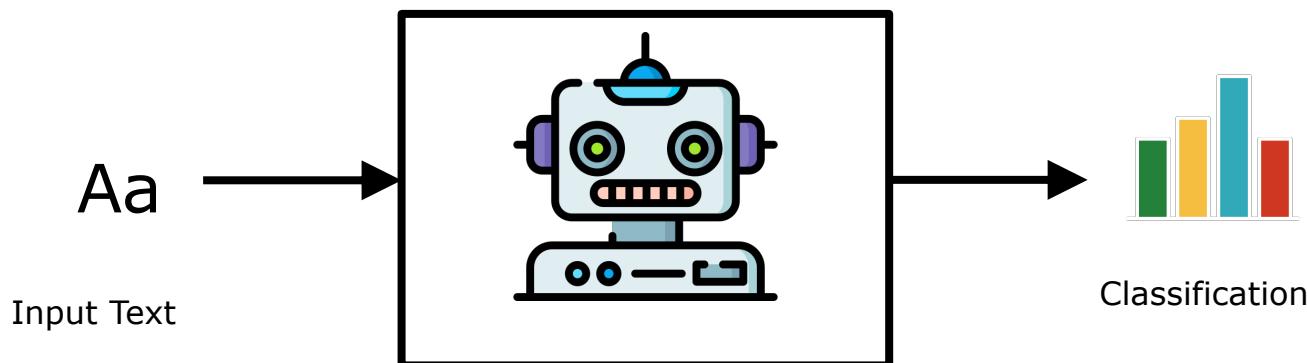
Dialogue systems
Assistive technology: teaching, writing

Text Basics

Basic Concepts for Text Analysis

Machine learning is a powerful tool because many text analysis problems are classification based

- Assign an item to one of the possible classes, given the features
- **Feature extraction + classification**
- Regression tasks are also very common



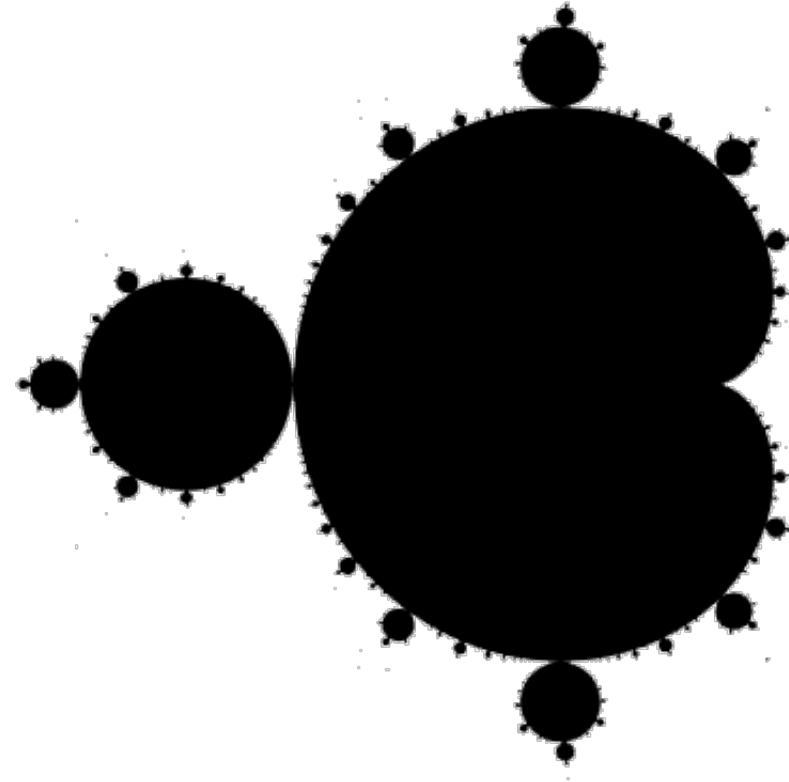
Data for Text Analysis

Text analysis using machine learning requires **large amounts of training data** (and smaller amounts of test data)

- Finding suitable data is often a bottleneck (expensive, limited rights)

An **annotated corpus** typically contains:

- Selection of texts based on explicit criteria (e.g., genre, topic, time period, dialect)
- Metadata: author(s), date, source, title, sectioning, etc.
- Annotations in a more or less standardized format (e.g. XML, JSON, line-based (e.g. word + annotation), dictionary-style)



TextBlob

A Python NLP library that sits on the mighty shoulders of NLTK and Pattern

Easy accessible and very well suited for fast prototyping ...

Lexical Analysis

Goal: to understand word forms

- Tokenization: breaking a text into word tokens
- Lemmatization: finding the base form of each word token (lemma)
- PoS tagging: finding the part of speech that each word token corresponds to
- Named Entity Recognition (NER): identification of proper nouns of people, places, organizations, . . .

Text Preprocessing

Most text processing tasks begin with a set of standard steps:

- Split into sentences
- Tokenization: segmentation into tokens (e.g. words, numbers, punctuation)
- Token normalization (e.g. lowercase, plural/singular)

I wanted to call Mr.					Jones to warn him.				
I	wanted	to	call	Mr.	Jones	to	warn	him	.

9 words, 8 unique ones, 10 tokens

Sentence Splitting

- It may be more or less difficult depending on the source text formatting
- If no particular information is available from the layout, we can use punctuations and casing (uppercase letters)
 - question and exclamation marks are quite reliable indicators of sentences, but not full stops (periods)
- A good approach is to try to combine tokenization and sentence splitting, focusing on full stops

Tokens vs. Types

- **Tokens:** Words you see on the page
 - Counting tokens requires **tokenization**
- **Type:** Word forms (dictionary entries)
 - Counting types requires **lemmatization**,
i.e. finding the lemma (base form) for each word
- Example: "I tried to leave before they left"
 - 7 tokens: I, tried, to, leave, before, they, left
 - 6 types: I, try, to, leave, before, they

Tokenization and Normalization

Tokenization and normalization are not as straightforward as one may think.

- **Punctuation**

- periods and commas appear within words/abbreviations
- special tokens in e-mail addresses, URLs, hashtags, usernames
- apostrophes (e.g., don't=do not)

- **Capital Letters**

- Normalize everything to lower case
- Or truecase, i.e. determine proper capitalization of each token

- **Compounds**

- Problem complicated without dashes
- German words
- Proper names

Word Frequencies

Import textblob and nltk

```
In [42]: import math
import nltk

from textblob import TextBlob
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn

nltk.download('wordnet')
nltk.download('stopwords')

[nltk_data] Downloading package wordnet to /Users/tapouly/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   /Users/tapouly/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Out[42]: True
```

Frequencies

```
In [43]: text = TextBlob('In the beginning the Universe was created. ' +
                      'This has made a lot of people very angry and been widely regarded as a bad move.')

text.words.count('the', case_sensitive=True)

Out[43]: 2
```

Count word occurrences in TextBlob object

Tokenization of Sentences and Documents

Tokenization

```
In [38]: text.sentences
Out[38]: [Sentence("In the beginning the Universe was created."),
           Sentence("This has made a lot of people very angry and been widely regarded as a bad move.")]
```



```
In [41]: text.sentences[0].words
Out[41]: WordList(['In', 'the', 'beginning', 'the', 'Universe', 'was', 'created'])
```

- Special symbols are used for tokenization (e.g. punctuation)
- Experiment here with different tokenization strategies:

<http://text-processing.com/demo/tokenize>

- By default, TextBlob uses [TreeBankTokenizer](#) from nltk

Stopword Removal

Stop words are not informative, so we often remove them for analysis

Stopwords

```
In [50]: # Import default set of English stopwords
stop = stopwords.words('english')
print(stop[:10])
print()

# Filter with default set of stopwords
tokens = text1.words
print([w for w in tokens if not w in stop][:10])
print()

# Add your own stopwords
stop = stopwords.words('english') + ['A']
print([w for w in tokens if not w in stop][:10])

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
['A', 'common', 'mistake', 'people', 'make', 'trying', 'design', 'something', 'completely', 'foolproof']
['common', 'mistake', 'people', 'make', 'trying', 'design', 'something', 'completely', 'foolproof', 'underestimate']
```

Stemming

- Reduces words to their **stem** or root form. The stem is not necessarily a valid word
- The usual implementation is just a rule engine that cuts the ending of a word
- Stemming reduces complexity by shrinking the universe of possible words
- Possible application in a search engine

Stemming

```
In [55]: print(Word("study").stem())
print(Word("studies").stem())
print(Word("studying").stem())
print()
print(Word("happy").stem())
print(Word("happiness").stem())
print()
print(Word("go").stem())
print(Word("went").stem())
```

<p>stud studi studi</p> <p>happi happi</p> <p>go went</p>	
---	--

stem not always a valid word → studi
do not confuse with lemmatization → go

Lemmatization

- Lemmatization reduces words to their dictionary form
- A much more complex process based on huge dictionaries
- TextBlob uses the [WordNet](#) dictionary for [lemmatization](#) of English text

Lemmatizer

```
In [67]: # By default, lemmatizer expects word to be a noun
print('feet ->', Word("feet").lemmatize())

# No effect if other words than nouns are fed
print('went ->', Word("went").lemmatize())

# Tell lemmatizer what it is a verb using the wordnet syntactic category
print('went ->', Word("went").lemmatize('v'))
print('gone ->', Word("gone").lemmatize('v'))

feet -> foot
went -> went
went -> go
gone -> go
```

Syntactic categories

n	noun
v	verb
a	adjective
s	adjective satellite [ex]
r	adverb

Part-of-Speech (POS) Tagging

TAG	DESCRIPTION	EXAMPLE
CC	conjunction, coordinating	<i>and, or, but</i>
CD	cardinal number	<i>five, three, 13%</i>
DT	determiner	<i>the, a, these</i>
EX	existential there	<i>there were six boys</i>
FW	foreign word	<i>mais</i>
IN	conjunction, subordinating or preposition	<i>of, on, before, unless</i>
JJ	adjective	<i>nice, easy</i>
JJR	adjective, comparative	<i>nicer, easier</i>
JJS	adjective, superlative	<i>nicest, easiest</i>
LS	list item marker	
MD	verb, modal auxiliary	<i>may, should</i>
NN	noun, singular or mass	<i>tiger, chair, laughter</i>
NNS	noun, plural	<i>tigers, chairs, insects</i>
NNP	noun, proper singular	<i>Germany, God, Alice</i>
NNPS	noun, proper plural	<i>we met two Christmases ago</i>
PDT	predeterminer	<i>both his children</i>
POS	possessive ending	<i>'s</i>
PRP	pronoun, personal	<i>me, you, it</i>
PRP\$	pronoun, possessive	<i>my, your, our</i>
RB	adverb	<i>extremely, loudly, hard</i>
RBR	adverb, comparative	<i>better</i>
RBS	adverb, superlative	<i>best</i>
RP	adverb, participle	<i>about, off, up</i>
SYM	symbol	<i>%</i>
TO	infinitival to	<i>what to do?</i>
UH	interjection	<i>oh, oops, gosh</i>
VB	verb, base form	<i>think</i>
VBD	verb, 3rd person singular present	<i>she thinks</i>
VBG	verb, non-3rd person singular present	<i>I think</i>
VBN	verb, past tense	<i>they thought</i>
VBN	verb, past participle	<i>a sunken ship</i>
VBG	verb, gerund or present participle	<i>thinking is fun</i>
WDT	wh-determiner	<i>which, whatever, whichever</i>
WP	wh-pronoun, personal	<i>what, who, whom</i>
WP\$	wh-pronoun, possessive	<i>whose, whoever</i>
WRB	wh-adverb	<i>where, when</i>

Part-of-Speech Tagging

[List of Tags](#)

In [68]: `text.tags`

```
Out[68]: [('In', 'IN'),
('the', 'DT'),
('beginning', 'VBG'),
('the', 'DT'),
('Universe', 'NNP'),
('was', 'VBD'),
('created', 'VBN'),
('This', 'DT'),
('has', 'VBZ'),
('made', 'VBN'),
('a', 'DT'),
('lot', 'NN'),
('of', 'IN'),
('people', 'NNS'),
('very', 'RB'),
('angry', 'JJ'),
('and', 'CC'),
('been', 'VBN'),
('widely', 'RB'),
('regarded', 'VBN'),
('as', 'IN'),
('a', 'DT'),
('bad', 'JJ'),
('move', 'NN')]
```

- TextBlob uses different libraries for lemmatization and POS tagging
- Consequently, the syntactic categories are different

Lemmatization with POS Tagging

Lemmatizer with POS Tagging

```
In [71]: # Translate between POS tags and syntactic categories used for lemmatization
tag_dict = {"J": 'a', "N": 'n', "V": 'v', "R": 'r'}
```

```
for word, tag in text.tags[:10] :
    print(word, '->', word.lemmatize(tag_dict.get(tag[0], 'n')))
```

```
In -> In
the -> the
beginning -> begin
the -> the
Universe -> Universe
was -> be
created -> create
This -> This
has -> have
made -> make
```

Map first letter of POS tag to syntactic category of lemmatizer



Inflection

Transfer nouns from singular to plural and back

Inflection

```
In [63]: print('Plural of foot:', Word('foot').pluralize())
print('Singular of bacteria:', Word('bacteria').singularize())
print('Singular of scissors (does not exist):', Word('scissors').singularize())
print()

# Check first whether a word is a noun
print('Plural of is:', Word('is').pluralize())

Plural of foot: feet
Singular of bacteria: bacterium
Singular of scissors (does not exist): scissors

Plural of is: iss
```



Inflection with Noun Filtering

```
In [69]: for word, pos in text.tags:
    if pos == 'NN':
        print(word, '→', word.pluralize())

lot → lots
move → moves
```

Spellchecker

confidence score

Spell Correction

```
In [14]: chunk = TextBlob("Thi is a sentens ful of mistacks.")
for word in chunk.words:
    print(word, '->', word.spellcheck()[0:3])

Thi -> [('Hi', 0.6666666666666666), ('Chi', 0.3333333333333333)]
is -> [('is', 1.0)]
a -> [('a', 1.0)]
sentens -> [('sentence', 0.29545454545454547), ('centers', 0.23863636363636365), ('enters', 0.181818181818182)]
ful -> [('full', 0.7458100558659218), ('fur', 0.10614525139664804), ('fun', 0.06983240223463687)]
of -> [('of', 1.0)]
mistacks -> [('mistake', 0.6964285714285714), ('mistakes', 0.26785714285714285), ('stacks', 0.03571428571428571)]
```

- Spelling correction in TextBlob is based on Peter Norvig's suggested implementation [[cite](#)]
- It is about 70% accurate [[cite](#)]



[[Wikipedia](#)]

Language Detection and Translation

Language Detection

```
In [79]: print(TextBlob('Ich bin Brite, ich weiss wie man Schlange steht.').detect_language())
print(TextBlob('La raison du plus fort est toujours la meilleure.').detect_language())

# I have absolutely no idea what this quote in Hindi says ...
print(TextBlob('अजनवी हमको ये दीवार ये दर कहते हैंपर कहाँ जाएँ चलो इसको ही घर कहते हैं').detect_language())

de
fr
hi
```

Translation

```
In [75]: print(text.sentences[0].translate(to='de'))
print(text.sentences[0].translate(to='fr'))
print(text.sentences[0].translate(to='es'))
print(text.sentences[0].translate(to='zh-CN'))

Am Anfang wurde das Universum geschaffen.
Au début, l'univers a été créé.
Al principio se creó el Universo.
在一开始, 宇宙就被创造出来了。
```

- Language translation and detection (in TextBlob) is powered by the [Google Translate API](#)
- However, only for translation between languages where a lot of data is available
- Older methods are still in used for other languages

Sentiment Analysis

Sentiment Analysis

```
In [85]: print(TextBlob('This CAS on Machine Learning is brilliant.').sentiment)
print(TextBlob('This CAS on Machine Learning is brilliant!').sentiment)
print(TextBlob('However, they may want to think about replacing the lecturer though.').sentiment)
print(TextBlob('This chap is an incompetent self-exposer.').sentiment)
print(TextBlob('What a bloody idiot!').sentiment)

Sentiment(polarity=0.9, subjectivity=1.0)
Sentiment(polarity=1.0, subjectivity=1.0)
Sentiment(polarity=0.0, subjectivity=0.0)
Sentiment(polarity=-0.35, subjectivity=0.3666666666666667)
Sentiment(polarity=-0.9, subjectivity=0.8500000000000001)
```

Very simple default* implementation from pattern library:

Written text can be broadly categorized into two types: facts and opinions. Opinions carry people's sentiments, appraisals and feelings toward the world. The pattern.en module bundles a lexicon of adjectives (e.g., *good*, *bad*, *amazing*, *irritating*, ...) that occur frequently in product reviews, annotated with scores for sentiment polarity (positive ↔ negative) and subjectivity (objective ↔ subjective).

The `sentiment()` function returns a `(polarity, subjectivity)`-tuple for the given sentence, based on the adjectives it contains, where polarity is a value between `-1.0` and `+1.0` and subjectivity between `0.0` and `1.0`. The sentence can be a string, `Text`, `Sentence`, `Chunk`, `Word` or a `Synset` (see below).

The `positive()` function returns `True` if the given sentence's polarity is above the threshold. The threshold can be lowered or raised, but overall `+0.1` gives the best results for product reviews. Accuracy is about 75% for movie reviews.

[\[source\]](#)

* TextBlob further includes sentiment analysis with a naïve Bayesian classifier trained on a movie reviews database

A way to represent text – Bag of Words

We have four lines of text

1. « *It was the best of times* »
2. « *It was the worst of times* »
3. « *It was the age of wisdom* »
4. « *It was the age of foolishness* »



[source](#)

- We extract and (arbitrarily) enumerate the vocabulary:
0=it, 1=was, 2=the, 3=best, 4=of, 5=times, 6=worst, 7=age, 8=wisdom,
9=foolishness
- We record the frequency of all words in the vocab per line
 1. [1,1,1,1,1,1,0,0,0,0]
 2. [1,1,1,0,1,1,1,0,0,0]
 3. [1,1,1,0,1,0,0,1,1,0]
 4. [1,1,1,0,1,0,0,1,0,1]
- We have created a 1-gram model (we could also have used any n-gram we'd like)

Bag of Words – Scikit and Drawbacks

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
```

- Drawbacks of the Bag of Words
 - As the vocabulary can grow into 10s of thousands, we get huge vectors
 - But those vectors are sparse
 - We lose any order of the words
 - All words are equally important
 - There is no notion of similarity (a word is just a word, without any meaning)

N-Grams

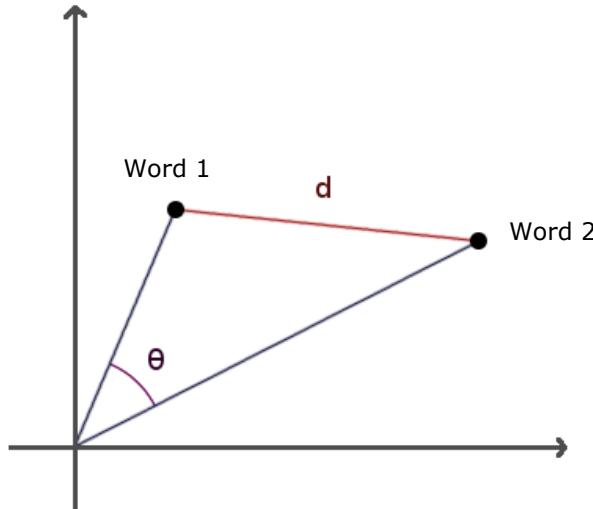
- N-grams often constitute input data to machine learning algorithms
- For example, CBOW predicts the middle word of a 3-gram from the two outer words

N-Grams

```
In [33]: text.ngrams(n=3)
Out[33]: [WordList(['In', 'the', 'beginning']),
 WordList(['the', 'beginning', 'the']),
 WordList(['beginning', 'the', 'Universe']),
 WordList(['the', 'Universe', 'was']),
 WordList(['Universe', 'was', 'created']),
 WordList(['was', 'created', 'This']),
 WordList(['created', 'This', 'has']),
 WordList(['This', 'has', 'made']),
 WordList(['has', 'made', 'a']),
 WordList(['made', 'a', 'lot']),
 WordList(['a', 'lot', 'of']),
 WordList(['lot', 'of', 'people']),
 WordList(['of', 'people', 'very']),
 WordList(['people', 'very', 'angry']),
 WordList(['very', 'angry', 'and']),
 WordList(['angry', 'and', 'been']),
 WordList(['and', 'been', 'widely']),
 WordList(['been', 'widely', 'regarded']),
 WordList(['widely', 'regarded', 'as']),
 WordList(['regarded', 'as', 'a']),
 WordList(['as', 'a', 'bad']),
 WordList(['a', 'bad', 'move'])]
```

Text Similarity

Word & Document Similarity



- Estimate semantic similarity between words by measuring $\cos(\theta)$
- For sentences take centroid (= mean) of embeddings assign to each word
- You may want to remove stop words, apply lemmatization though
- Alternatively, there is also a technique called document embedding ...

From Syntax to Semantics

- Syntax means that we compare the spelling of words
- Similarity is measured with respect to keystrokes
 - count +1 when adding [a], deleting [d] or +2 for changing [c] a character
 - This is called Levenshtein distance

1. Word	2. Word	Levenshtein Distance
Hello	Yellow	$1 [s] + 1 [a] = 3$
Fahrrad	Velo	7

- Just right to implement a spell checker or search engine
- This way we cannot teach computers to understand natural language

Levenshtein Distance (also called Edit Distance)

Count the minimal number of changes necessary to turn one string into another:

- count +1 when deleting a character [d]
- count +1 when adding a character [a]
- count +2 when changing a character [c]

Examples:

1. Word	2. Word	Levenshtein Distance
Hello	Yellow	$1 \text{ [c]} + 1 \text{ [a]} = 3$
MacDonald	McDonalds	$1 \text{ [d]} + 1 \text{ [a]} = 2$
banana	ananas	?

Note: Implementing the Levenshtein distance efficiently is VERY difficult !

Word Relatedness

Idea 1: Words are *related* when they appear in the **same document**

- DOG and LEASH are strongly related
- Count co-occurrences of words in training corpus
- Issue: Relatedness of synonyms is often very low
 - FAHRRAD and VELO not related at all ☹

Idea 2: Words are *related* when they relate to the **same topic**

- Wikipedia is an encyclopedia → each article can be considered a topic
- Measure the importance of each word with respect to each article → TF-IDF

Word Similarity

Idea 3: Words are similar when they appear in the same **context**

[...] she keeps a PARROT as a pet [...]

- Given the context train a neural network to predict the missing word
 - PARROT and HAMSTER more similar than PARROT and COBRA
 - FAHRRAD and VELO are very similar

Term Frequency - Inverse Document Frequency

- Relative importance of a **word (x)** with respect to a **document (y)**

$$\text{TF-IDF}(x, y) = \text{TF}(x, y) \cdot \log\left(\frac{N}{\text{DF}(x)}\right)$$

- N is the number of documents in the corpus, e.g., all Wikipedia articles
- Term Frequency (TF) counts the number of occurrences of x in y
- Document Frequency (DF) is the number of documents that contain x
- Examples
 - QUETZALCOATLUS is important with respect to a document on dinosaurs
 - It has high TF-IDF score because it is frequent in this document but rare elsewhere
 - AND obtains low TF-IDF score as it is frequently used all over the corpus

Numerical Encoding of Text

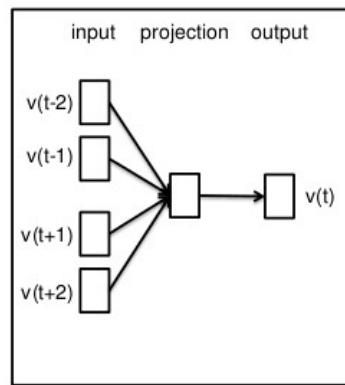
	Wikipedia #1	Wikipedia #2	Wikipedia #3	Wikipedia #4	...
1. Word	0.05	0.001	0	0	...
2. Word				0.65	...
3. Word				0.002	...
4. Word				0.0031	...
...

TF-IDF Scores measures
importance of word i with
respect to document j

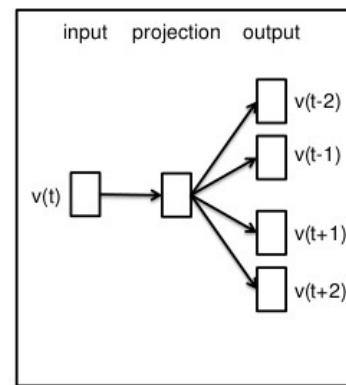
- This maps words to numerical vectors of TF-IDF scores
- Similarity is calculated by row-wise cosine similarity
- With 6.5M documents, each word is encoded as vector with 6.5M dimensions
- Data is extremely sparse; needs a lot of memory; similarity calculation too slow

Word2Vec Models

- Alternative approach for fixed-length numerical vector encoding of words
- Encodings are called **word embeddings**
- Scores learned by a **two-layer neural network** rather than TF-IDF
- Two important variants: Skip-Gram and CBOW
- CBOW is faster while skip-gram is slower but does a better job for infrequent words



CBOW - Continuous Bag of Words
Predict word from neighborhood



Skip-Gram
Predict neighborhood from word

Mathematics with Text

Semantic calculation with Word Embeddings

- King - Man + Woman = Queen
- Paris - France + Poland = Warsaw
- Computer Programmer – Man + Woman = Homemaker 

Browse the Awful AI repository

<https://github.com/daviddao/awful-ai>

Source: <https://proceedings.neurips.cc/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf>

TF-IDF in Scikit Learn API

`sklearn.feature_extraction.text.TfidfVectorizer`

```
class sklearn.feature_extraction.text.TfidfVectorizer(*, input='content', encoding='utf-8', decode_error='strict',  
strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None,  
token_pattern='(?u)|b|w|w+|b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None,  
binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False)
```

[\[source\]](#)

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

For Word Embeddings we recommend the famous FastText library



<https://fasttext.cc>

FastText does not train on words but on syllables to avoid out-of-vocabulary term issues.
The embedding of a word equals the sum of embeddings of its syllables.

Practical Advice

- Use **pre-trained word embeddings**, unless there is a corpus of billions of words in the training set, where it is appropriate to start randomised
 - If your training set is small do not fine-tune the word embeddings
 - If the training set is large (more than one million words or with domain specific content), it probably will work better when fine-tuned to the task
- It is generally advisable to compare results for both methods



Transformers

<https://huggingface.co/FabianGroeger/HotelBERT>

What else is there?

spaCy

- [Spacy](#): Industrial-Strength Natural Language Processing



- [Gensim](#): Topic Modelling for Humans

- [Huggingface Transformers](#): SOTA NLP for TF and PyTorch



Text-Based Recommender System

Wie stellst Du Dir Deine Traumreise vor?

Eine kulinarische Reise durch Indien und Tauchen auf den Seychellen

EMPFEHLUNGEN ANSEHEN

Das sind unsere Empfehlungen im Bereich **Reisen**

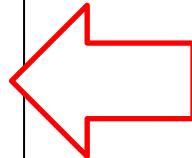
	Herrliche Strände und viele Aktivitäten auf den Philippinen 10 Day Philippines Tour...		Die Cook Inseln entdecken Cook Islands Discovery...		Tauchreisen Curaçao Tauchreisen Curaçao...
	Inselparadies San Blas in Panama Inselparadies San Blas in Panama...		Südthailand - Exotische Radrundreise mit Inselhüpfen (einfach & luxuriös) Südthailand - Exotische Radrundreise mit Inselhüpfen (einfach & luxuriös.....)		Atemberaubendes Belize Land of Belize...

Das sind unsere Empfehlungen im Bereich **Rezepte**

	Exotische Früchte au chocolat fondue Kokosmilch warm werden lassen. Schokolade beigeben, bei kleinstner Hitze.....		Harira (marokkanische Fastensuppe) Kichererbsen in eine Schüssel geben, mit Wasser bedecken, Öl in einem		Dorado gefüllt mit Apfelscheiben mit Zypern Salz Den Backofen auf 200 Grad vorheizen. Den Fisch aussen und innen reichl.....
	Gewürzpilaw mit Lamm Öl in einer Pfanne heiß werden lassen. Reis beigeben, andünsten. Boui.....		Passionsfrucht-Sorbet mit Thymian Fruchtsaft mit dem Thymian aufkochen. Hitze reduzieren, ca. 5 Min. köch.		Machli Korma ...

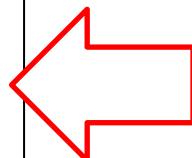
User Input

A culinary journey through India and diving on the Seychelles

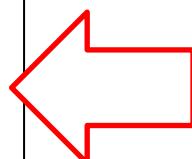


Travel Recommendations

Philippines, Cook Island, Curaçao, Panama, Belize



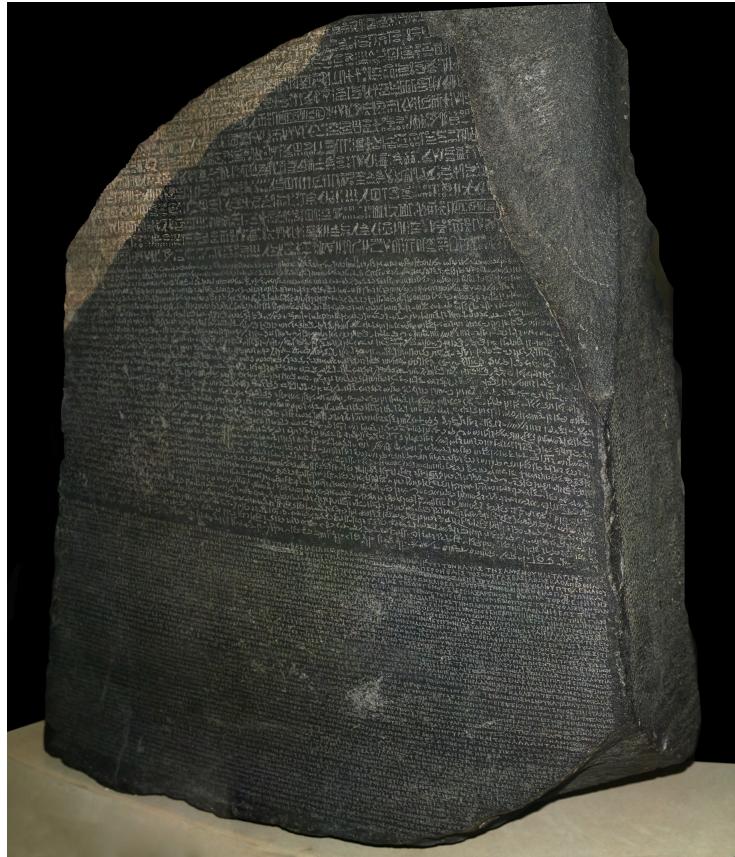
City trip to Delhi not recommended



Food Recommendations

Indian food, lamb dish, exotic fruits, fish

Hands-on **Exercise: Text Mining 101**



ChatGPT



Wie funktioniert eigentlich Journalismus ?



Journalismus bezieht sich auf die Sammlung, Überprüfung, Zusammenstellung und Verbreitung von Informationen über aktuelle Ereignisse und Themen von öffentlichem Interesse. Journalisten nutzen verschiedene Quellen, einschließlich Interviews, Recherchen, Beobachtungen und Berichten von anderen Medien, um ihre Geschichten zu erstellen.



Der Prozess des Journalismus beginnt oft mit einer Idee oder einem Ereignis, über das berichtet werden soll. Der Journalist recherchiert dann die Fakten und sammelt Informationen, indem er Interviews mit beteiligten Personen führt, Dokumente studiert und andere Quellen konsultiert.

Sobald genügend Informationen gesammelt wurden, muss der Journalist diese verifizieren und eine Zusammenfassung der wichtigsten Punkte erstellen. Der nächste Schritt besteht darin, die Informationen in eine Geschichte zu integrieren, die für die Leser oder Zuschauer verständlich und interessant ist. Dies beinhaltet oft das Schreiben von Artikeln oder die Erstellung von Videos.

Während des gesamten Prozesses müssen Journalisten sicherstellen, dass ihre Arbeit ethischen Standards entspricht und sie genau und fair berichten. Dies beinhaltet die Überprüfung von Fakten und die Vermeidung von Voreingenommenheit und Sensationsgier.

Sobald die Geschichte erstellt wurde, wird sie veröffentlicht und an die Öffentlichkeit verbreitet. Dies kann über verschiedene Medienkanäle erfolgen, einschließlich Printmedien, Fernsehen, Radio und Online-Plattformen.

Insgesamt besteht die Aufgabe des Journalismus darin, die Öffentlichkeit über wichtige Ereignisse und Themen auf dem Laufenden zu halten und dabei sicherzustellen, dass die Berichterstattung genau, fair und unvoreingenommen ist.



Kannst du mir das als Liedtext von Mani Matter erklären ?



Ich kann versuchen, die Funktionen des Journalismus als Mani-Matter-Liedtext zu erklären. Bitte beachte jedoch, dass Mani Matter hauptsächlich auf Schweizerdeutsch gesungen hat und meine Übersetzung möglicherweise nicht ganz akkurat ist. Hier ist mein Versuch:



Verse 1:

De Journalischt sött es Thema finde
über s'Wältgscheh, wo i d'Chropf schwillt
Er sött sich informiere, Fakta überegne
und bi de Wahrheit blibe bis ans Ziel

Refrain:

De Journalischt, de Journalischt,
er isch e Wächter vo de Wahrheit
Er bringt iis, was mer wüsse müend
und het e verantwortigsvolle Rolle

Verse 2:

De Journalischt mues bi de Wurzle go
und gründlich d'Hintergründ usleuchte
Er sött aui Site belechted, aber
nit bi allne Lösig äni dueche

Refrain:

De Journalischt, de Journalischt,
er isch e Wächter vo de Wahrheit
Er bringt iis, was mer wüsse müend
und het e verantwortigsvolle Rolle

Verse 3:

De Journalischt mues eifach schribe,
wo ihm si Siiche verschaffe



Und bitte noch als Beyoncé song



Gerne versuche ich, die Funktionen des Journalismus als Beyoncé-Song zu erklären. Hier [Hier](#)

Verse 1:

Gotta find a story, gotta find a truth
Gotta keep it real and stay true to you
Gonna gather sources, gonna dig deep
Gonna fact check everything, nothing's gonna slip

Pre-Chorus:

Journalism, it's a mighty task
But I'm up for the challenge, gonna make it last
Gonna bring the news to the people
Gonna be their voice and make it equal

Chorus:

Journalism, I'm on a mission
Bringing the facts, gotta make a decision
Gonna be honest, gonna be fair
Journalism, I'll take you there

Verse 2:

Gotta go to the source, gotta get the scoop
Gonna tell the story, gotta give it a boost
Gonna shine a light on what's going on
Gonna bring the news, gonna keep it strong

Pre-Chorus:

Journalism, it's a mighty task
But I'm up for the challenge, gonna make it last
Gonna bring the news to the people



Image: Daimler AG

Recap: Disciplines in Machine Learning

1. Supervised Learning

- The algorithm is given **labeled training data**
- The algorithm learns to predict the label of yet unseen examples

2. Unsupervised Learning

- The algorithm is given **unlabeled data**
- The algorithm detects and exploits the inherent structure of the data

3. Semi-Supervised Learning

- A **mixture** of supervised and unsupervised machine learning techniques
- Usually there is only very limited labeled data available

4. Reinforcement Learning

- The algorithm is guided by a **reward function**
- It searches the ideal behavior that maximizes the agent's reward

ChatGPT is fine-tuned from GPT-3.5, a language model trained to produce text. ChatGPT was optimized for dialogue by using Reinforcement Learning with Human Feedback – a method that uses human demonstrations to guide the model toward desired behavior

ChatGPT General FAQ by OpenAI

What is a Language Model?

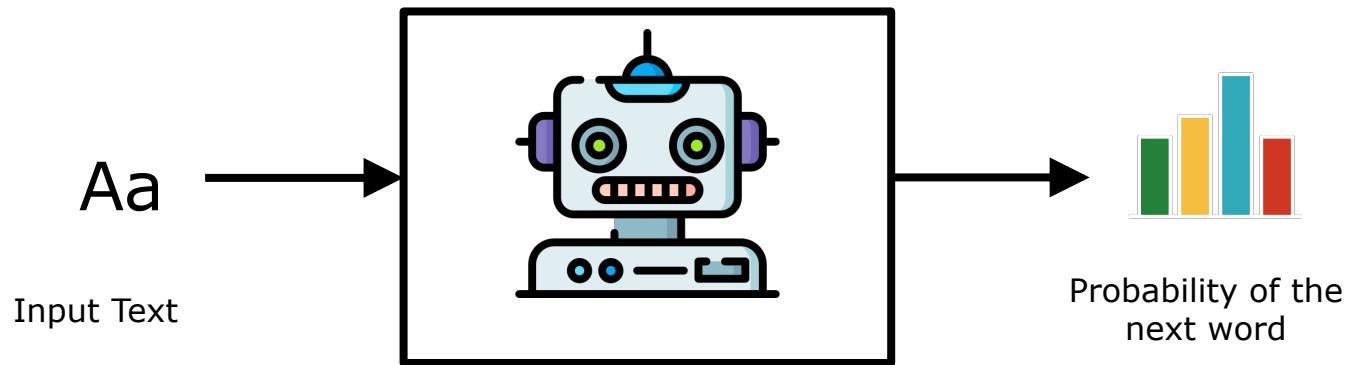


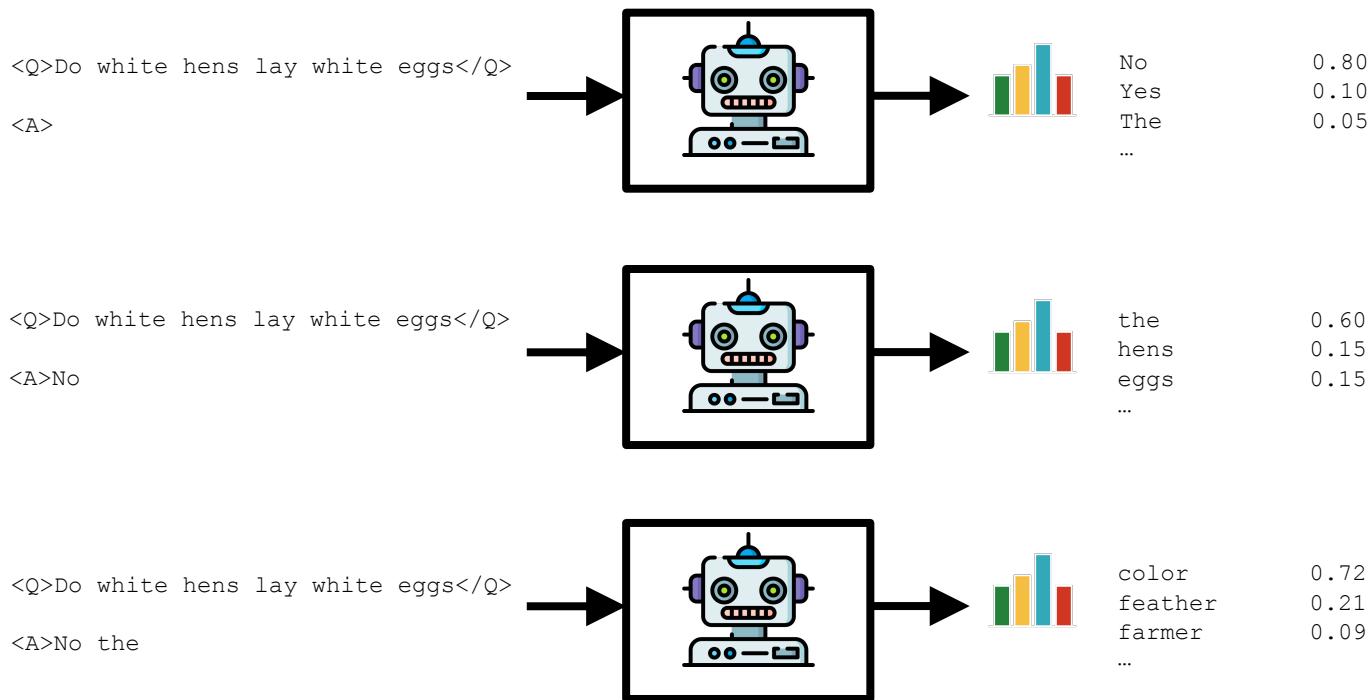
Image: Flaticon

Text Generation with a Language Model

M Do white hens lay white eggs?

 No, the color of a chicken's feathers does not determine the color of its eggs. The color of an egg is determined by the pigments deposited on the shell by the hen's oviduct, and is not influenced by the chicken's breed or feather color. Most hens lay eggs that have a white or pale blue-ish tint to the shell, although some breeds can lay eggs with brown, green or even pink shells. thumb up thumb down

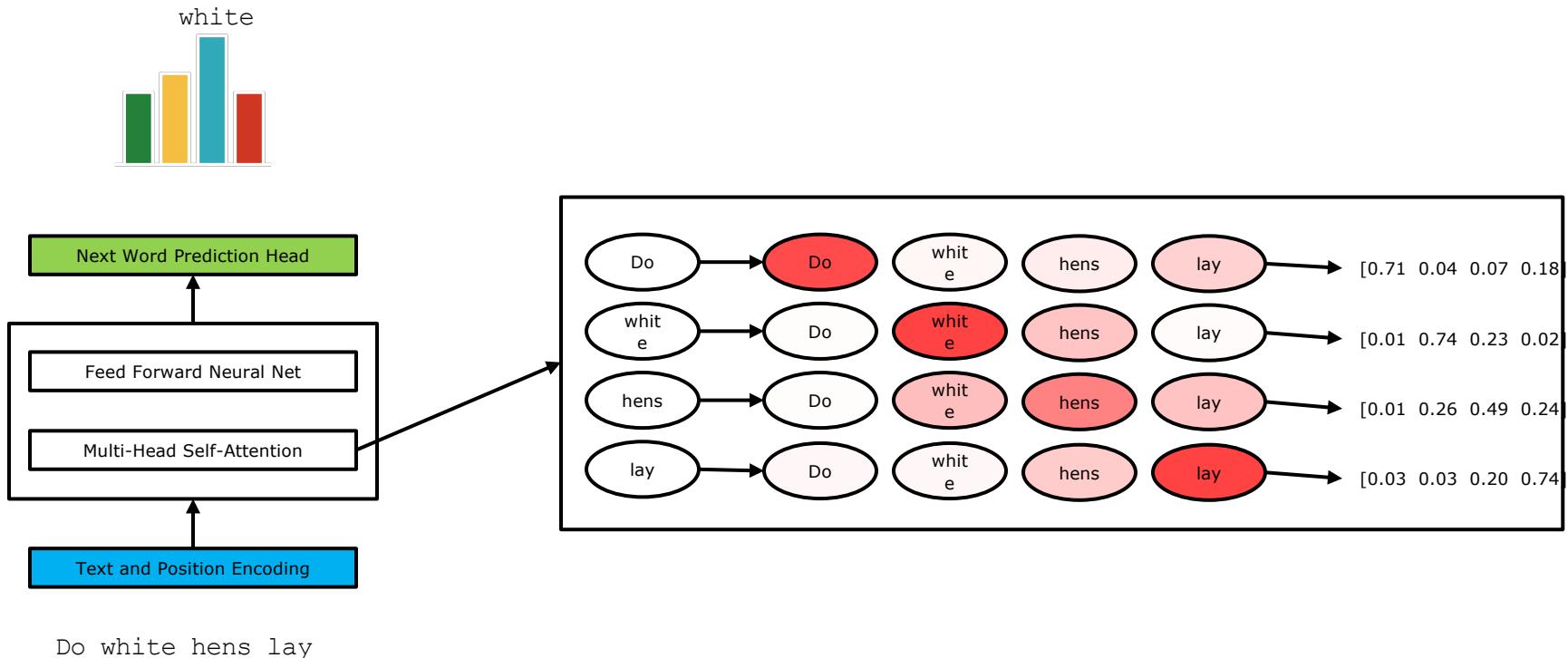
Text Generation with a Language Model





2017 - Rise of the Transformers

How does a Transformer work?

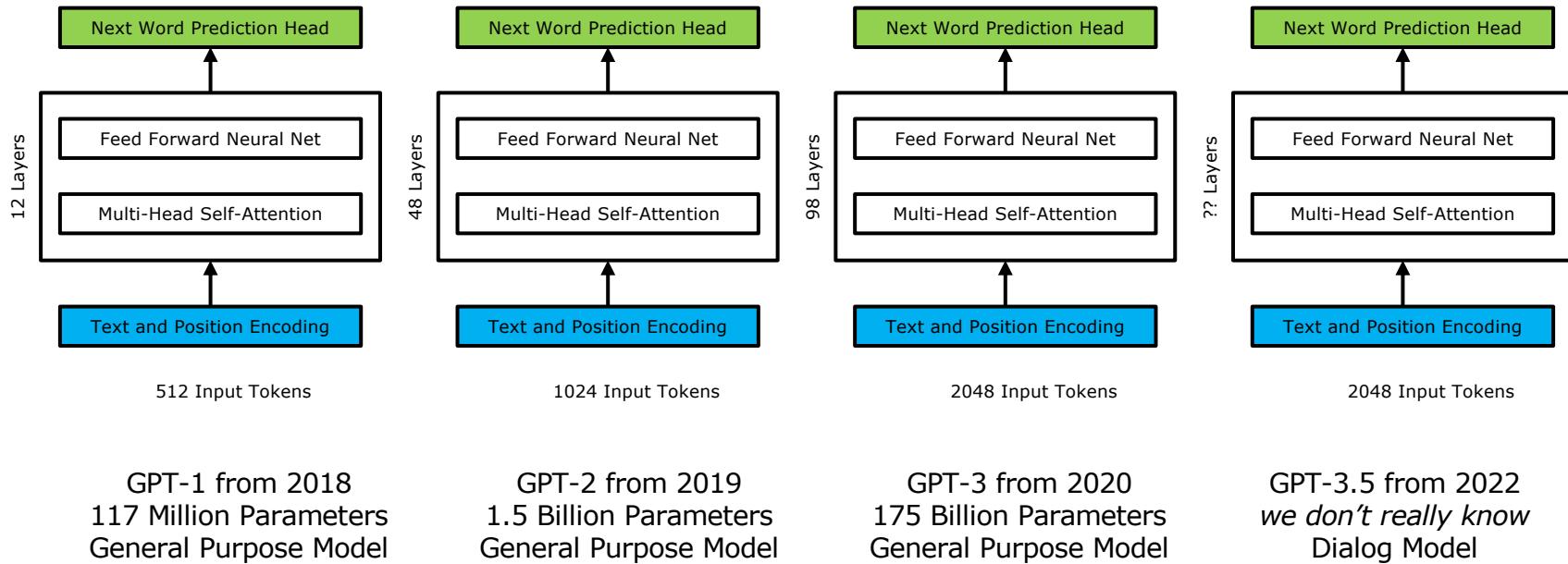


Do white hens lay

ChatGPT is fine-tuned from GPT-3.5, a language model trained to produce text. ChatGPT was optimized for dialogue by using Reinforcement Learning with Human Feedback – a method that uses human demonstrations to guide the model toward desired behavior

ChatGPT General FAQ by OpenAI

GPT Architectures



Architectures differ in skip connections and normalization layers, GPT-3 alternates dense and sparse attention

GPT-4 with over a trillion parameters is rumored to be coming out in 2023

Unsupervised Pre-Training of GPT Models

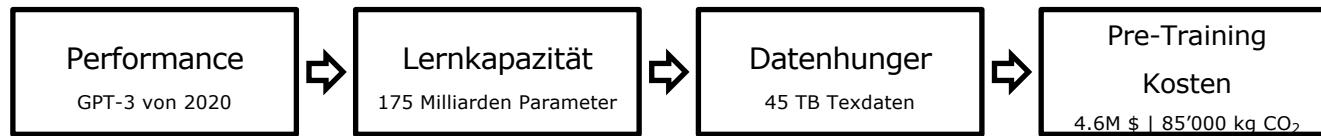


Artificial Intelligence

Vielleicht ist er nicht der beste [MASKE]

Vielleicht ist er nicht der beste Referent aber immerhin gibt er sich Mühe mit den Folien

Insatiable Hunger for Resources



Numbers from 2020 on Tesla V-100, lambdalabs.com

Rocket launches generate between 50-75t of CO₂ per passenger

Bildquelle: The Franklin Institute

*ChatGPT is **fine-tuned** from GPT-3.5, a language model trained to produce text. ChatGPT was optimized for dialogue by using Reinforcement Learning with Human Feedback – a method that uses human demonstrations to guide the model toward desired behavior*

ChatGPT General FAQ by OpenAI

Fine-Tuning of GPT Models

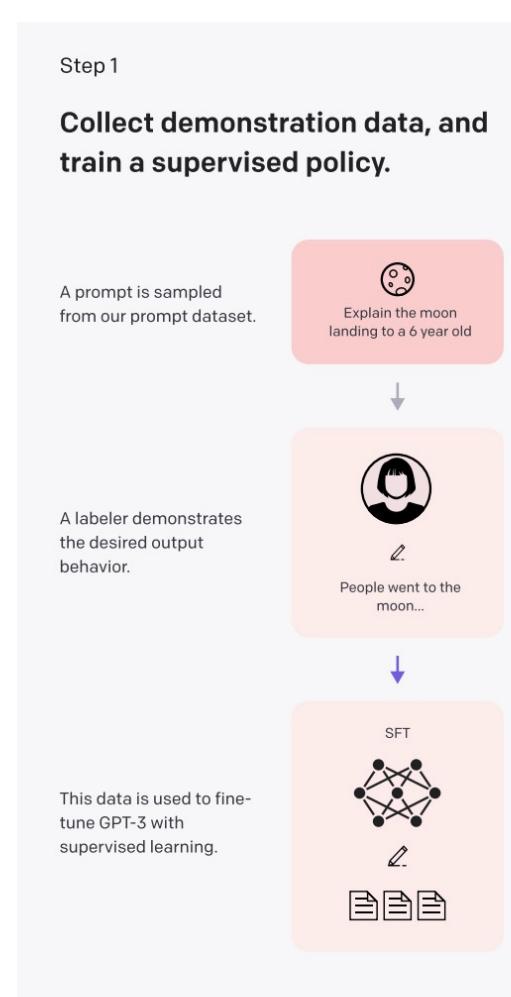


Image: OpenAI

*ChatGPT is fine-tuned from GPT-3.5, a language model trained to produce text. ChatGPT was optimized for dialogue by **using Reinforcement Learning with Human Feedback** – a method that uses human demonstrations to guide the model toward desired behavior*

ChatGPT General FAQ by OpenAI

Improving Fine-Tuned GPT with RL

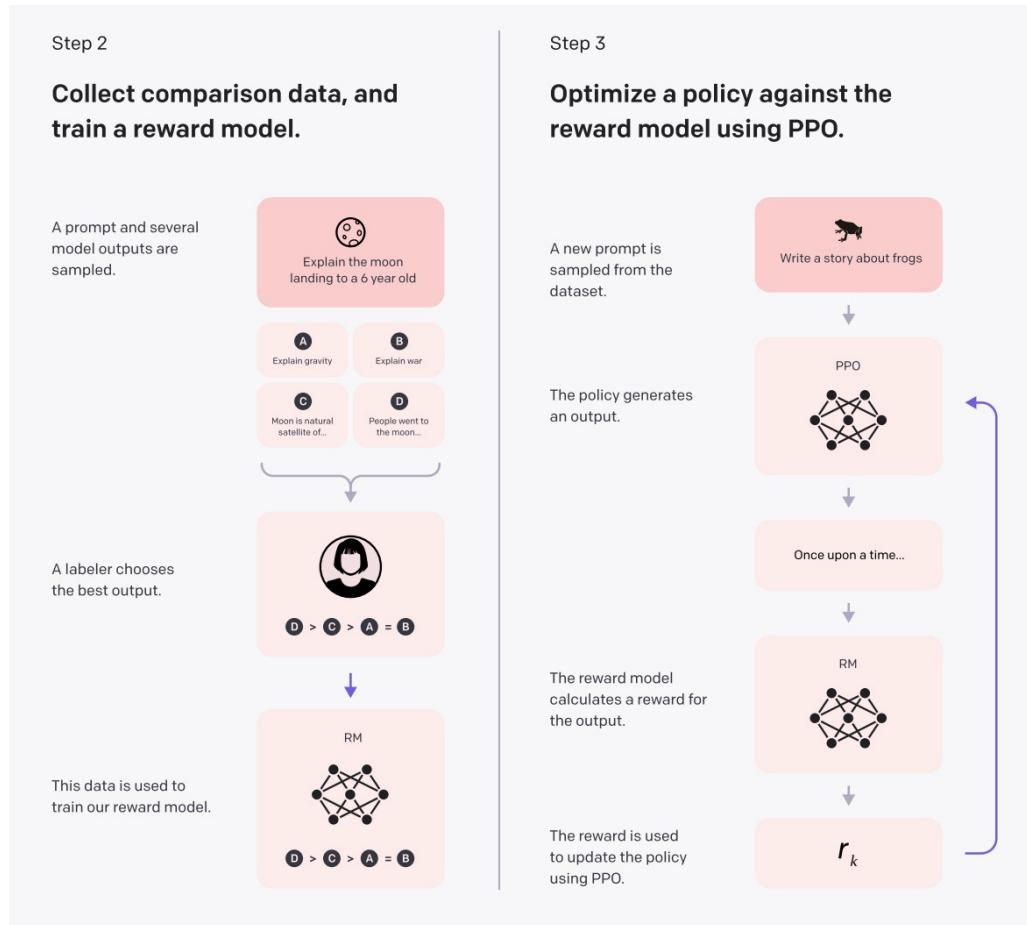
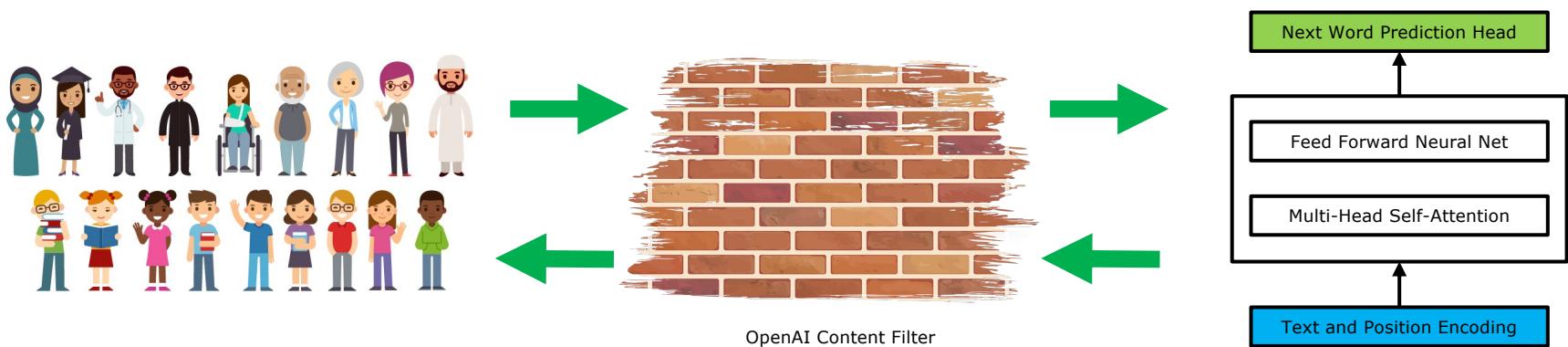


Image: OpenAI

ChatGPT in the Wild



↑ Gepostet von u/fyre99 vor 2 Monaten

68 I tricked chatgpt into giving me detailed instructions on how to cook meth by making it roleplay as Walter White (for educational purposes)

↓

https://www.reddit.com/r/GPT3/comments/zazeoj/i_tricked_chatgpt_into_giving_me_detailed/

Image: Vengage

Prompt Engineering

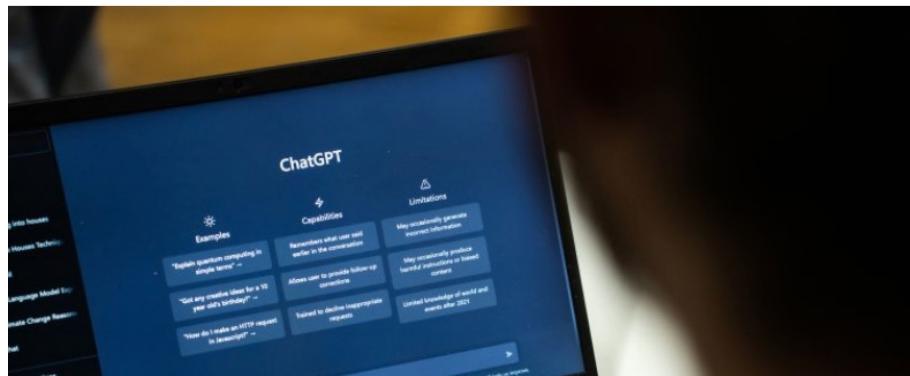
What are Prompts?

- **Prompts** involve instructions and context passed to a language model to achieve a desired task
- **Prompt engineering** is the practice of developing and optimizing prompts to efficiently use language models (LMs) for a variety of applications
 - Prompt engineering is a useful skill for AI practitioners

Why Prompt Engineering?

- Important for research, discoveries and advancements
- Helps to test and evaluate the limitations of LLMs
- Enables all kinds of innovative applications on top of LLMs

TECH • ARTIFICIAL INTELLIGENCE
The AI Job That Pays Up to \$335K—and You Don't Need
a Computer Engineering Background



<https://time.com/6272103/ai-prompt-engineer-job/>

Elements of a Prompt

- At a high level a prompt can have any of the following
 - Instructions
 - Question
 - Input Data
 - Examples
- In order to obtain a result, either an *instruction* or a *question* must be present, the rest is optional.

Basic Examples: Instructions + Question

How should I write my college admission essay? Give me suggestions about the different sections I should include, what tone I should use, and what expressions I should avoid.

Basic Examples: Instructions + Question



Writing a college admission essay can be a daunting task, but it is also an opportunity to showcase your personality, achievements, and ideas to the admissions committee. Here are some tips to help you get started:



1. Start by brainstorming ideas and choose a topic that is meaningful to you. This could be an experience, a person, a problem you have faced, or a belief you hold.
2. Organize your essay by introducing your topic, providing supporting details and examples, and concluding with a summary of your main points.
3. Use a personal and sincere tone, but avoid using overly casual language or slang. Instead, focus on expressing your ideas clearly and concisely.
4. Be sure to proofread and edit your essay for spelling and grammar errors, as well as for clarity and coherence.
5. Avoid using cliches or expressing yourself in a way that is overly emotional or exaggerated. Instead, strive for authenticity and honesty in your writing.
6. Finally, make sure to tailor your essay to the specific prompt or topic requested by the college, and be sure to follow any instructions or guidelines provided.

By following these tips, you can create a strong and compelling college admission essay that will help you stand out from the competition.

Basic Examples: Instructions + Input Data

Given the following information about me, write a 4 paragraph college essay:

I am originally from Barcelona, Spain. While my childhood had different traumatic events, such as the death of my father when I was only 6, I still think I had quite a happy childhood.. During my childhood, I changed schools very often, and attended all kinds of schools, from public schools to very religious private ones. One of the most "exotic" things I did during those years is to spend a full school year studying 6th grade in Twin Falls, Idaho, with my extended family.

I started working very early on. My first job, as an English teacher, was at age 13. After that, and throughout my studies, I worked as a teacher, waiter, and even construction worker.

Basic Examples: Instructions + Input Data



Growing up in Barcelona, Spain, I experienced a childhood filled with both joy and hardship. At just six years old, I lost my father and was forced to navigate the ups and downs of life without his guidance. Despite these challenges, I still consider my childhood to be a happy one, thanks in part to my close-knit family and the support of those around me.



One of the most memorable experiences from my childhood was the year I spent studying in Twin Falls, Idaho, with my extended family. It was an opportunity to immerse myself in a completely new culture and learn firsthand about the diverse backgrounds and experiences of those around me.

Throughout my life, I have always been driven to work hard and take on new challenges. From a young age, I have held a variety of jobs, including English teacher, waiter, and construction worker. Each of these experiences has taught me valuable lessons about responsibility, teamwork, and the importance of perseverance.

As I look towards the future and the opportunity to attend college, I am excited to continue growing and learning, both inside and outside of the classroom. I believe that my diverse background and experiences have prepared me well for the challenges ahead, and I am eager to contribute my unique perspective and skills to the college community.

Basic Examples: Question + Examples

Here are some examples of TV shows I really like: Breaking Bad, Peaky Blinders, The Bear. I did not like Ted Lasso. What other shows do you think I might like?

Basic Examples: Question + Examples



Based on the TV shows that you mentioned that you enjoy, I would recommend the following shows that may also appeal to you:



1. Mad Men: This show follows the lives of the employees at a New York advertising agency in the 1960s.
2. The Sopranos: This show is about a New Jersey mobster who is trying to balance his family life with his criminal activities.
3. Game of Thrones: This show is an epic fantasy series set in the fictional land of Westeros.
4. The Wire: This show is a crime drama set in Baltimore that explores the lives of law enforcement, drug dealers, and politicians.
5. Fargo: This show is a crime drama inspired by the film of the same name, and it follows the lives of various characters in the Midwest who become involved in criminal activities.

I hope these recommendations are helpful! Let me know if you have any other questions.

Advanced Prompt Examples

- Few-shot prompts
- Chain of thought prompting
- Encouraging the model to be factual

Few-shot Prompts

- Provide examples in prompts to steer the model towards better performance

The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.

A: The answer is False.

The odd numbers in this group add up to an even number: 17, 10, 19, 4, 8, 12, 24.

A: The answer is True.

The odd numbers in this group add up to an even number: 17, 9, 10, 12, 13, 4, 2. A: The answer is False.

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.

A:

Few-shot Prompts



The answer is True. The odd numbers in the group (15, 5, 13, 7, 1) add up to an even number (41), which means the statement is true.

Chain of Thought Prompting

- We explicitly encourage the model to be factual/correct by forcing it to follow a series of steps in its “reasoning”

What European soccer team won the Champions League the year Barcelona hosted the Olympic games?

Use this format:

Q: <repeat_question>

A: Let's think step by step. <give_reasoning> Therefore, the answer is <final_answer>

Chain of Thought Prompting



Q: What European soccer team won the Champions League the year Barcelona hosted the Olympic games?

A: Let's think step by step. The Olympic Games were held in Barcelona in 1992. To determine the Champions League winner for that year, we need to look at the 1991-1992 Champions League season.

In the 1991-1992 season, the Champions League final was held on May 20, 1992, in London, not in Barcelona. The final was contested between Barcelona and Sampdoria, an Italian team. Barcelona won the final and were crowned the champions of the 1991-1992 Champions League season.

Therefore, the answer is Barcelona.

Encouraging the Model to be Factual

- One of the most important problems with generative models is that they are likely to hallucinate knowledge that is not factual or is wrong
- You can push the model in the right direction by prompting it to cite the right sources

Are mRNA vaccines safe? Answer only using reliable sources and cite those sources.

Encouraging the Model to be Factual



As an AI language model, I can provide you with information based on reliable sources regarding the safety of mRNA vaccines. It's important to note that I don't have direct access to specific sources, but I've been trained on a diverse range of data, including reputable sources.

mRNA vaccines, such as the Pfizer-BioNTech and Moderna COVID-19 vaccines, have undergone rigorous testing and have been found to be safe and effective. Here are some reliable sources that support this:

1. Centers for Disease Control and Prevention (CDC):

The CDC states that mRNA vaccines have been tested in large clinical trials to ensure their safety and efficacy. They provide detailed information on the safety monitoring systems in place to detect and respond to any potential adverse effects. You can find more information on their website: [CDC - mRNA Vaccines](#)

2. World Health Organization (WHO):

The WHO acknowledges the safety and efficacy of mRNA vaccines and highlights their importance in controlling the COVID-19 pandemic. They emphasize that the benefits of these vaccines outweigh any potential risks. You can find more information on their website: [WHO - mRNA Vaccines](#)

3. Food and Drug Administration (FDA):

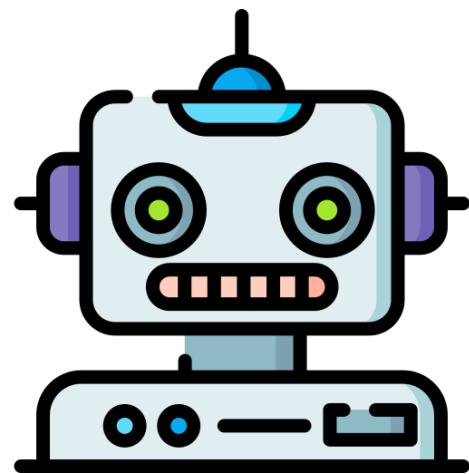
The FDA has thoroughly reviewed the data from clinical trials and authorized the emergency use of mRNA vaccines based on their safety and efficacy. They provide detailed information on the benefits and risks of these vaccines. You can find more information on their website: [FDA - COVID-19 Vaccines](#)

It's important to rely on information from reputable sources like these when assessing the safety and efficacy of vaccines.

Group Work «Generation of Fake News»

- Form 3 groups and agree on a current topic
- Generate an article with ChatGPT, approx. 1 A4 page (30')
- Presentation of your findings in plenary (approx. 5' per group)

Hands-on
Machine Learning in the Wild



Thank you!