

Einführung in die künstliche Intelligenz

EKI07 – Natural Language Processing

Prof. Dr. A. del Pino

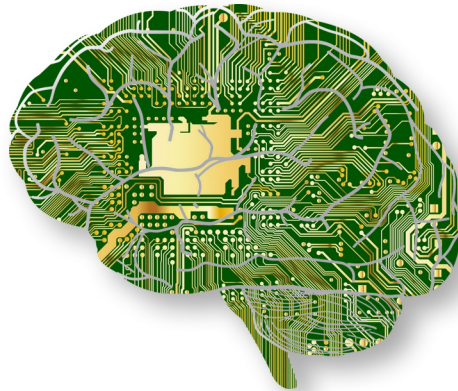


Bild: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Anatomy-1751201_1280.png

Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

Die KI-Landkarte

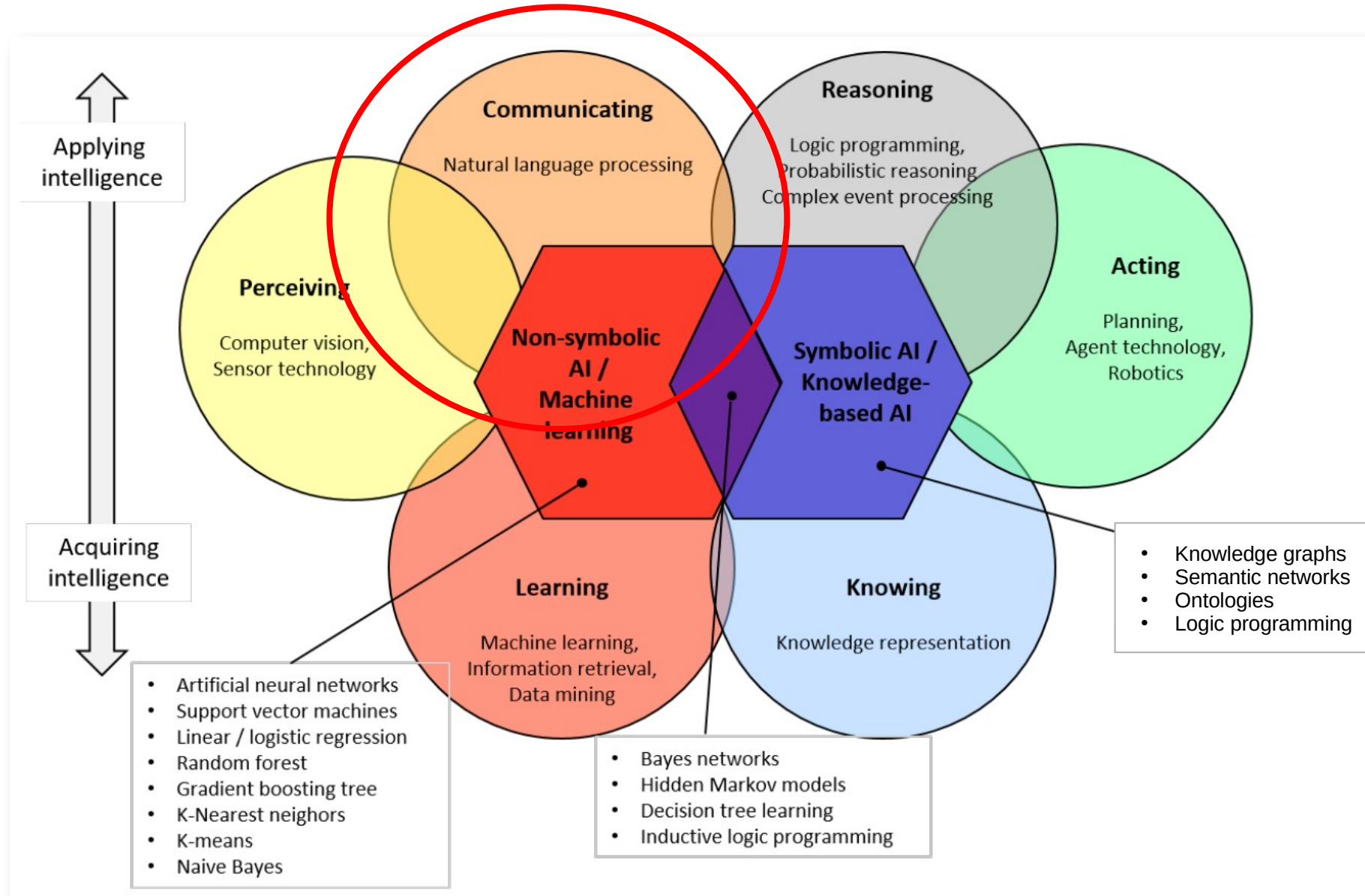


Bild: B. Humm "Applied Artificial Intelligence", S. 4

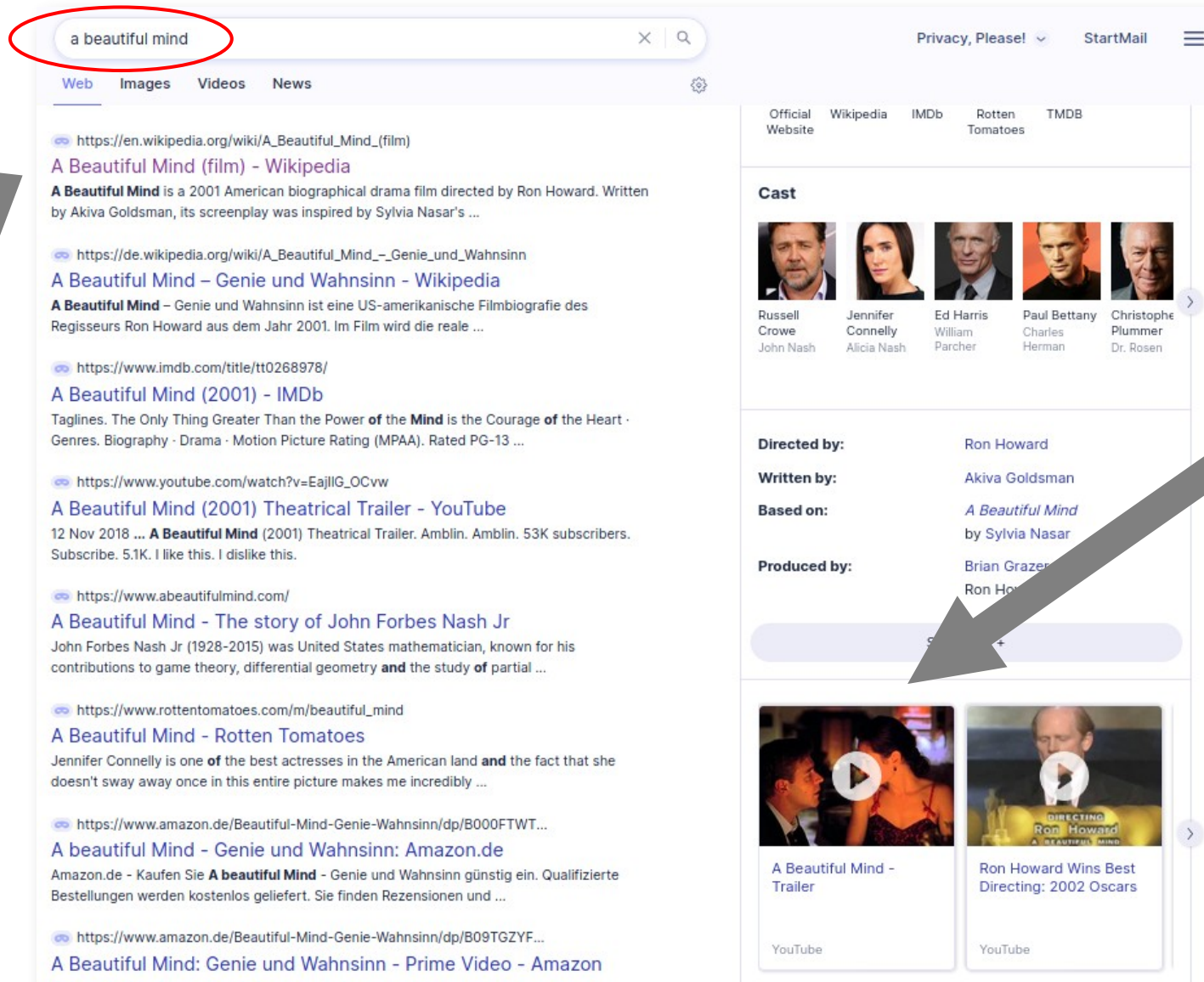
Bereiche der Verarbeitung natürlicher Sprache (Natural Language Processing, NLP)

- **Information Retrieval (IR)** Das Abrufen relevanter Dokumente ([document retrieval](#)) wäre ein besserer Begriff.
- **Textklassifikation** Das Klassifizieren natürlichsprachlicher Texte um z.B. die Sprache, das Thema oder die Stimmung ([sentiment](#)) zu analysieren.
- **Information Extraction (IE)** Das Verstehen von geschriebenen oder gesprochenen Texten. Der Aufbau von Wissensgraphen aus Texten.
- **Question Answering (QA)** Das Beantworten von natürlichsprachigen Fragen.
- **Maschinelles Übersetzen**
([machine translation](#)) Das Übersetzen von Texten zwischen verschiedenen natürlichen Sprachen.
- **Texterzeugung**
([text generation](#)) Das Erzeugen von geschriebenen oder gesprochenen Texten. Das Zusammenfassen von Texten.

Information retrieval (=document retrieval)

- Eingabe: Text → Ausgabe: relevante Dokumente

Eingabe: Text



Ausgabe:
HTML Dokument

Ausgabe:
MP4 Dokument

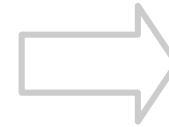
Bild: <https://eu.startpage.com/sp/search>

Textklassifikation: Beispiel Spracherkennung

Eingabe: Text

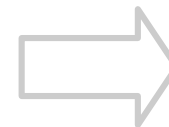
Ausgabe: Klassifikation

A Beautiful Mind is a 2001 American biographical drama film based on the life of the American mathematician John Nash, a Nobel Laureate in Economics and Abel Prize winner. The film was directed by Ron Howard, from a screenplay written by Akiva Goldsman.



EN

A Beautiful Mind ist ein amerikanischer biografischer Drama-Film aus dem Jahr 2001, der auf dem Leben des amerikanischen Mathematikers John Nash, einem Nobelpreisträger für Wirtschaftswissenschaften und Abel-Preisträger, basiert. Regie führte Ron Howard, nach einem Drehbuch von Akiva Goldsman.

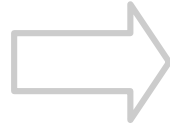


DE

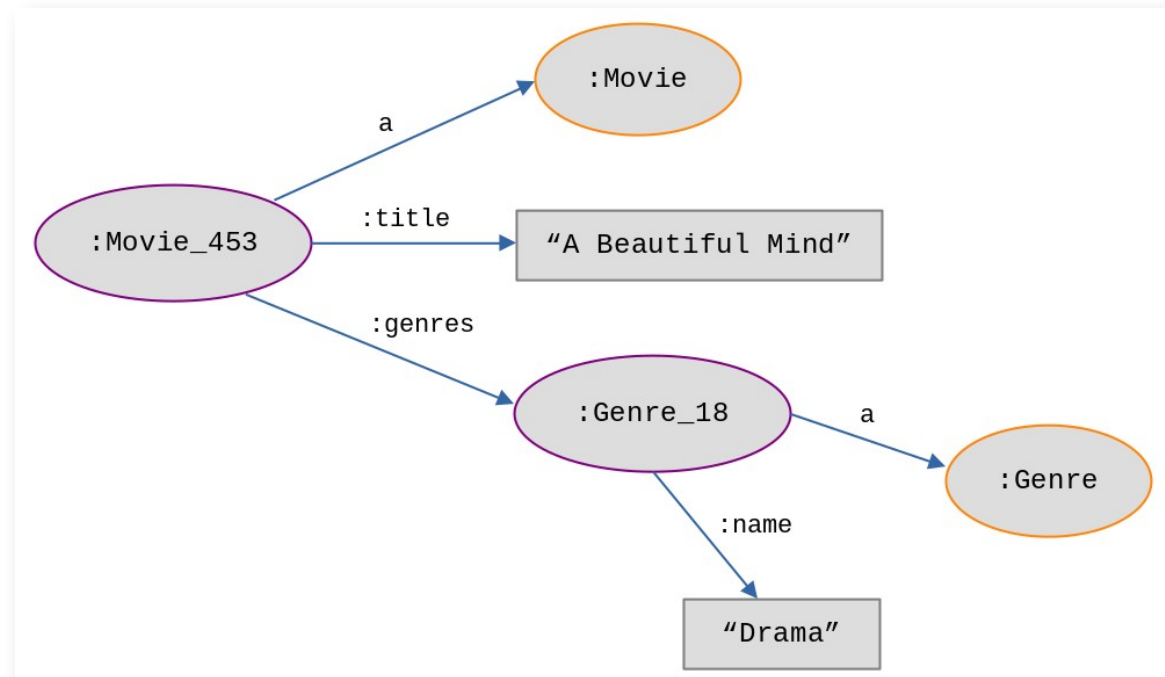
Information extraction

Eingabe: Text

"A Beautiful Mind" is a movie
which belongs to the genre
drama.



Ausgabe: Wissensgraph



Question answering

Eingabe: Fragetext



Ausgabe: Antwortstext



Bild: https://upload.wikimedia.org/wikipedia/commons/5/5c/Amazon_Echo.jpg

Maschinelle Übersetzung (machine translation)

Eingabe: Text in einer Sprache

Ausgabe: Text in einer anderen Sprache

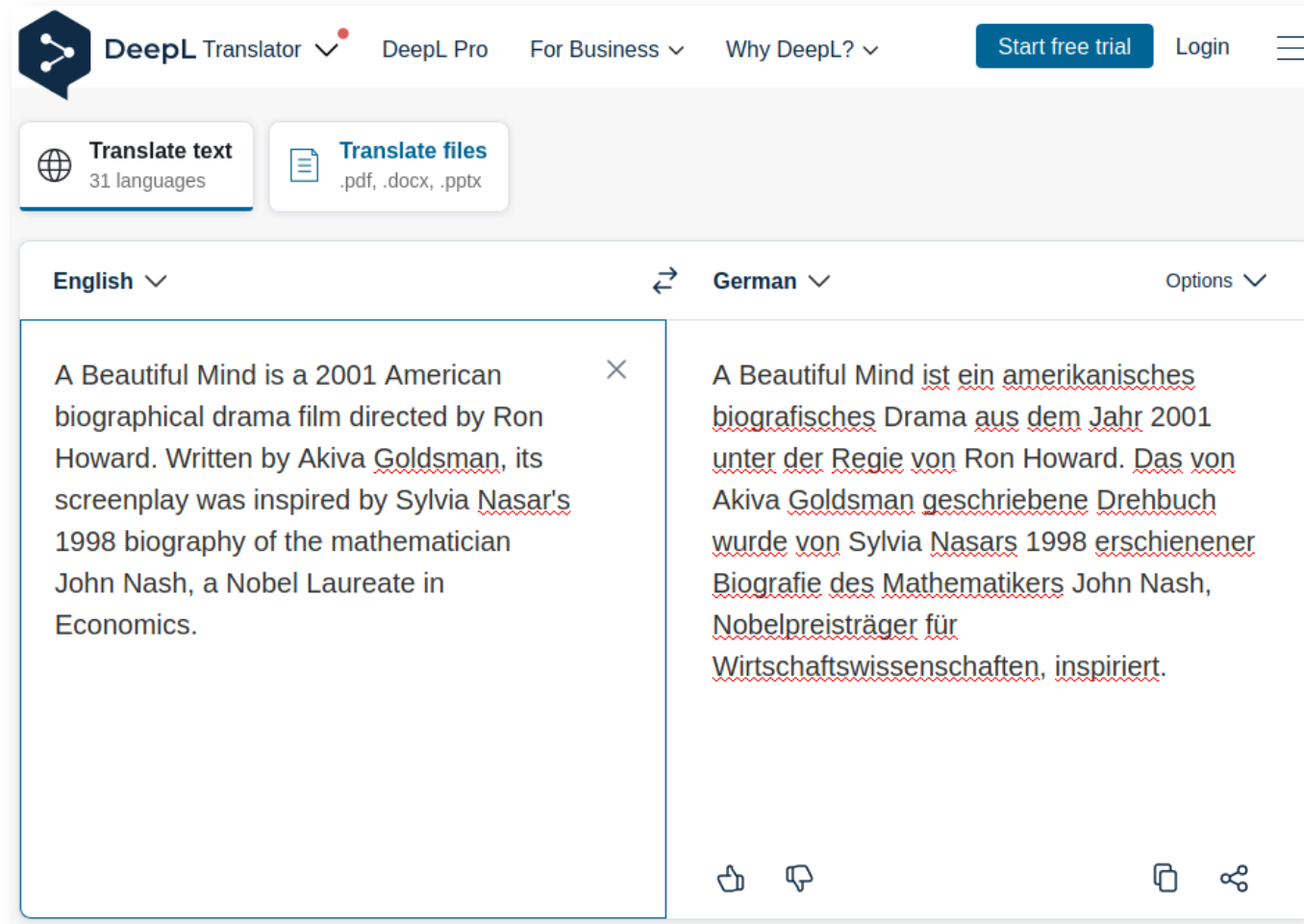
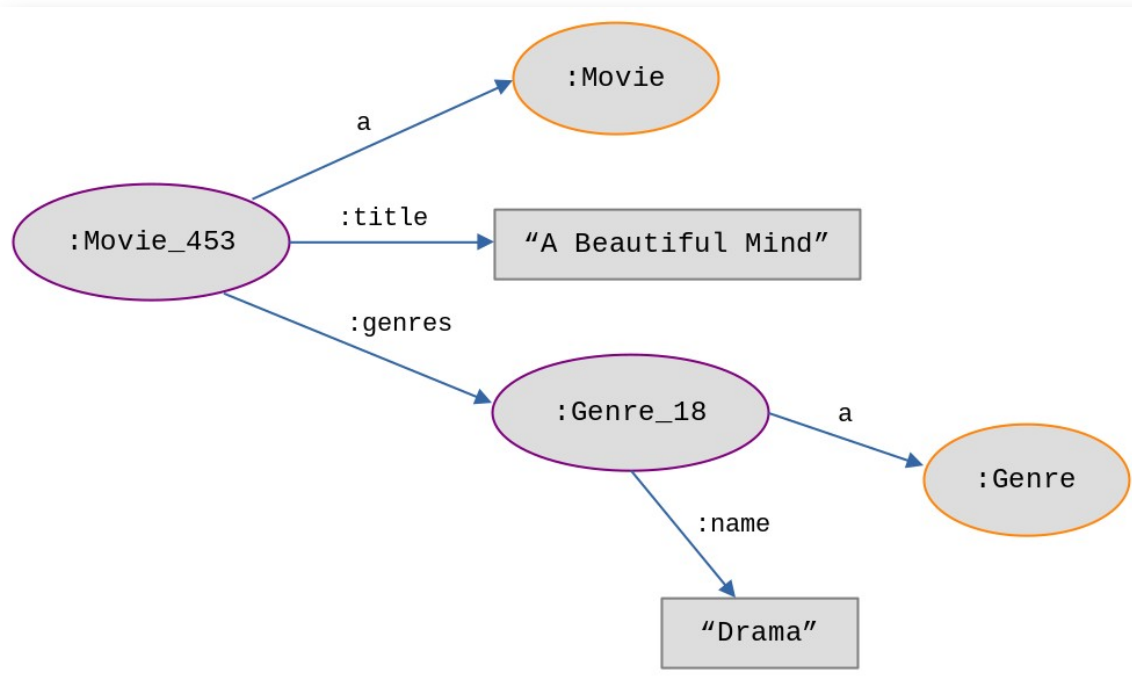


Bild: <https://www.deepl.com/translator>

Texterzeugung (text generation)

Eingabe: Wissensgraph



Ausgabe: Text

"A Beautiful Mind" is a movie
which belongs to the genre
drama.

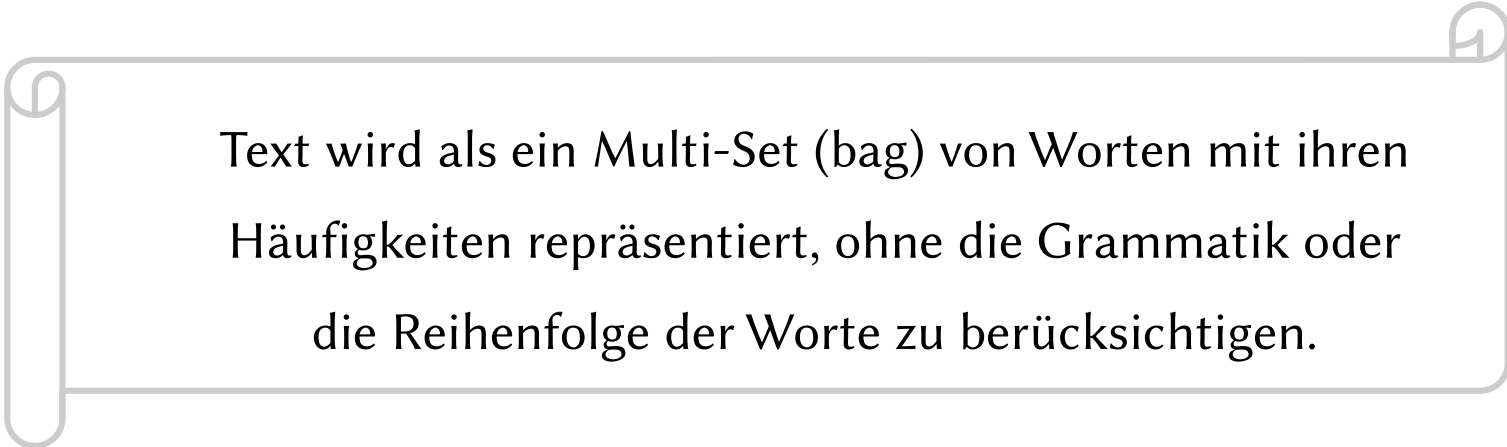
Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

Das bag-of-words Modell

- Einfacher NLP Ansatz
- Erstaunlich gute Ergebnisse in manchen NLP-Anwendungen wie z. B. Textklassifikation

Grundidee:



Text wird als ein Multi-Set (bag) von Worten mit ihren Häufigkeiten repräsentiert, ohne die Grammatik oder die Reihenfolge der Worte zu berücksichtigen.

Beispiel:

"John likes to watch movies. Mary likes movies too."

BoW ={"John":1, "likes":2, "to":1, "watch":1, "movies":2, "Mary":1, "too":1};

Quelle: https://en.wikipedia.org/wiki/Bag-of-words_model

Anwendung

- Das bag-of-Words Modell wird verwendet, um Feature-Vektoren für ML zu erzeugen.
- Einfache Form: Worthäufigkeit (**term frequency**)

Jeder Eintrag in dem Vektor repräsentiert die Häufigkeit eines Wortes in dem Text.

- Beispieltext t1:

ID	John	likes	to	watch	movies	Mary	too	...	Label
t1	1	2	1	1	2	1	1	...	A
t2	0	0	2	0	0	0	1	...	C
...	...	...	...	...	...	...	...	...	...
Tn	0	1	2	2	3	0	0	...	A

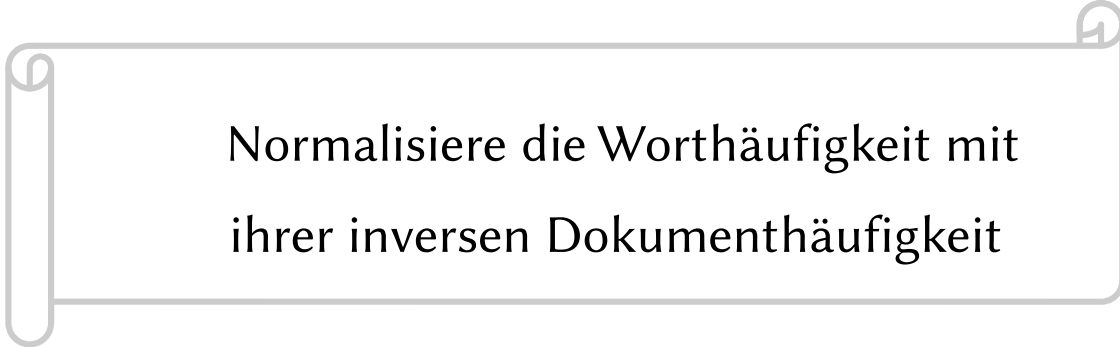
“John” taucht einmal im Text auf, “likes” zweimal usw.

Tf-Idf (term frequency-inverse document frequency)

- Worte die häufig in einem Text vorkommen, sind nicht unbedingt die wichtigsten.

Beispiel: “the”, “a”, “to” im Englischen.

- Grundidee:



Normalisiere die Worthäufigkeit mit
ihrer inversen Dokumenthäufigkeit

Die **Worthäufigkeit** gibt an, wie oft ein Wort in dem betrachteten Text vorkommt.

Die **Dokumenthäufigkeit** gibt an, wie oft ein Wort in allen Texten (**corpus**) vorkommt.

- Es gibt mehrere Formeln für tf-idf, siehe <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

N-Gramm Modell

- Das einfache bag-of-words Modell berücksichtigt nur einzelne Worte.
- Als Erweiterung kann man Sequenzen aus N aufeinander folgenden Worten (**n-gram**) betrachten.
- Beispiel: N=2



The diagram illustrates the extraction of 1-grams and 2-grams from the sentence "John likes to watch the movie". A bracket labeled "1-gram" spans the words "John", "likes", and "to". Another bracket labeled "2-gram" spans the pairs "John likes", "likes to", and "to watch".

ID	John	likes	to	...	John likes	likes to	to watch	...	Label
t1	1	2	1	...	1	1	1	...	A
t2	0	0	2	...	0	0	1	...	C
...	...	...	...	...	...	...	...	...	...
tn	0	1	2	...	0	1	2	...	A

Quelle: https://en.wikipedia.org/wiki/Bag-of-words_model

- N-Gramme können genutzt werden, um Feature-Vektoren zu generieren.
- N-Gramme können ebenfalls mit tf-idf verwendet werden.

Diskussion des bag-of-words Modells

- Was sind die Vorteile und Nachteile des bag-of-words Modells?



Bildquelle: https://cdn.pixabay.com/photo/2015/11/03/08/56/question-mark-1019820_960_720.jpg

Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

Beispiel: Vorhersage des Genre eines Films aus dem Überblicktext

Eingabe: Überblicktext eines Films

Ausgabe: Genre (Kategorie)

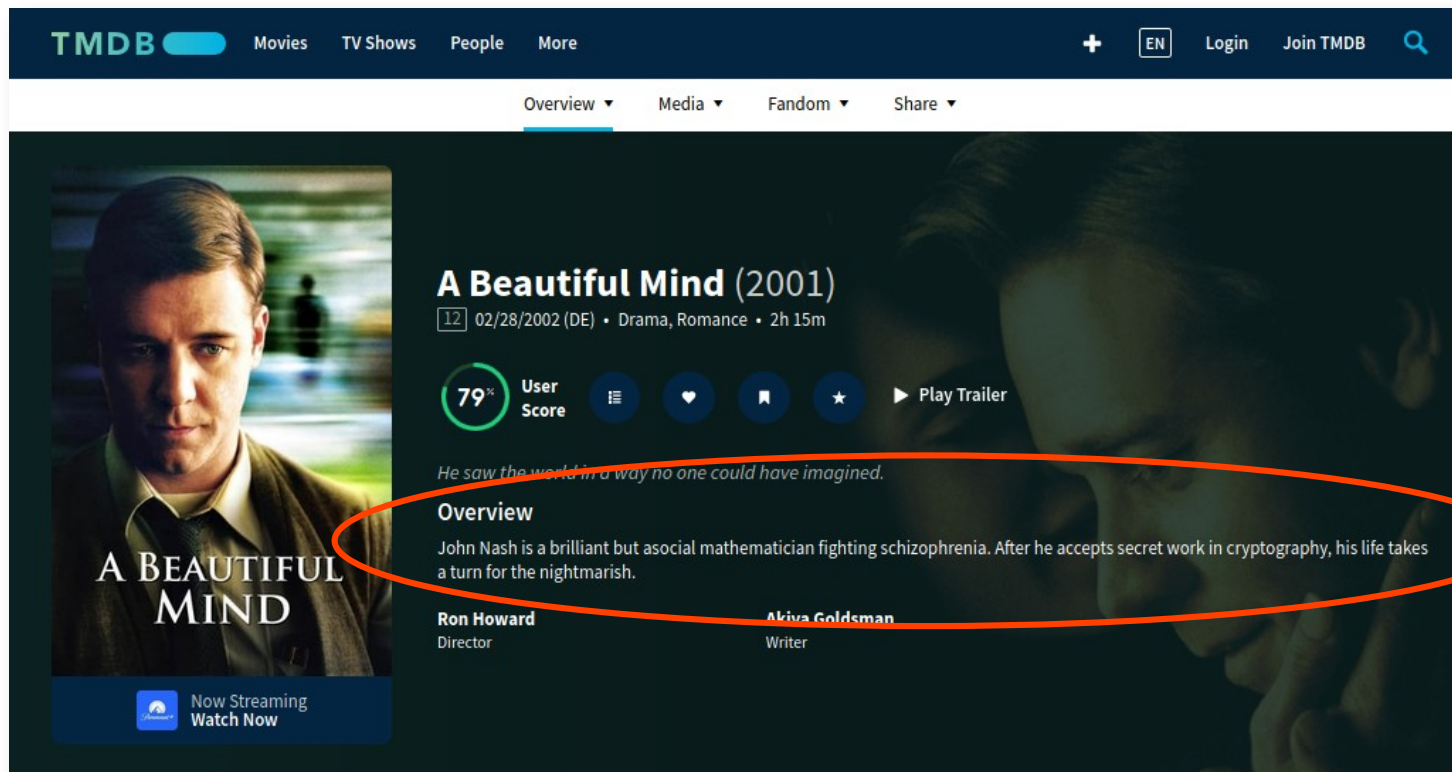


Bild: <https://www.themoviedb.org/movie/453-a-beautiful-mind>

Trainingsdaten: Movie_Overview_Classification.csv

Eingabe: Overview

Ausgabe: Binäre Klassifikation Drama

	A	B	C
1	id	overview	genre_Drama
2	1	When Lou, who has become the "father of the Internet," is shot by an unknown assailant, Jacob and Nick fire up the time ma	0
3	2	Mia Thermopolis is now a college graduate and on her way to Genovia to take up her duties as princess. Her best friend Lilly	1
4	3	Under the direction of a ruthless instructor, a talented young drummer begins to pursue perfection at any cost, even his hum	1
5	4	Vidya Bagchi (Vidya Balan) arrives in Kolkata from London to find her missing husband Arnab Bagchi. Seven months pregna	1
6	5	Marine Boy is the story of a former national swimmer who finds himself in debt, and out of desperation is employed as a mul	0
7	6	Pinocchio and his friends, a glow worm and a marionette, search for a magic music box. However, so are the evil Scalawag	0
8	7	A young girl buys an antique box at a yard sale, unaware that inside the collectible lives a malicious ancient spirit. The girl's	0
9	8	A chronicle which provides a rare window into the international perception of the Iraq War, courtesy of Al Jazeera, the Arab	0
10	9	After telling the story of Flint's last journey to young Jim Hawkins, Billy Bones has a heart attack and dies just as Jim and	0
11	10	In "A Mighty Wind", director Christopher Guest reunites the team from "Best In Show" and "Waiting for Guffman" to tell tell th	0
12	11	When world heavyweight boxing champion, Apollo Creed wants to give an unknown fighter a shot at the title as a publicity st	1
13	12	The members of the Lambda Lambda Lambda fraternity travel to Fort Lauderdale for a fraternity conference. They'll have to	0
14	13	Lester Burnham, a depressed suburban father in a mid-life crisis, decides to turn his hectic life around after developing an in	1
15	14	Disenchanted with the movie industry, Chili Palmer tries the music industry, meeting and romancing a widow of a music exe	0
16	15	John Anderton is a top 'Precrime' cop in the late-21st century, when technology can predict crimes before they're committed	0
17	16	Novica is a mathematics champion in a Belgrade high school. In an attempt to overcome the 'geek' status at school, Novica	1
18	17	After attending the funeral of her grandmother in Dallas, the Lux Atlantic Hotel manager Lisa is waiting for a flight to Miami.	0
19	18	In 1857, at the height of his fame and fortune, novelist and social critic Charles Dickens meets and falls in love with teenag	1
20	19	While working a job at an exclusive ski resort to support her Dad, Kim learns to snowboard and is so good at it that she ente	0
21	20	Professional driver Frank Martin is living in Miami, where he is temporarily filling in for a friend as the chauffeur for a gov	0
22	21	The prospects for continuing life on Earth in the year 2058 are grim. So the Robinsons are launched into space to colonize A	0
23	22	An idyllic summer turns into a nightmare of unspeakable terror for yet another group of naive counselors. Ignoring Camp Cry	0
24	23	When a group of misfits is hired by an unknown third party to burglarize a desolate house and acquire a rare VHS tape, they	0
25	24	Kaitlyn, an ER nurse who is tending to a young stabbing victim, is accidentally electrocuted by the defibrillator that is used i	0
26	25	A genetic engineering experiment gone horribly awry turns a large flock of docile sheep into unrelenting killing machines in	0
27	26	In 1957, Evan Rendell flees after his father is lynched for killing multiple patients in his effort to find a replacement heart f	0
28	27	An employee of a corporation with a lucrative secret process is tempted to betray it. But there's more to it than that.	1
29	28	Wallace, a medical school dropout, has been repeatedly burned by bad relationships. So while everyone around him, includi	1
30	29	The Autobots must stop a colossal planet-consuming robot who goes after the Autobot Matrix of Leadership. At the same tin	0

DataFrame aus einer CSV-Datei laden

```
import pandas as pd

data = pd.read_csv('data/Movie_Overview_Classification.csv',
encoding='utf8',
delimiter=',',
quotechar='|',
quoting=csv.QUOTE_MINIMAL)
```

Siehe: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Das bag-of-words Modell mit scikit-learn

scikit-learn 1.2.2
Other versions

Please [cite us](#) if you use the software.

6.2. Feature extraction
6.2.1. Loading features from dicts
6.2.2. Feature hashing
6.2.3. Text feature extraction
6.2.4. Image feature extraction

6.2.3. Text feature extraction

6.2.3.1. The Bag of Words representation

Text Analysis is a major application field for machine learning algorithms. However the raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

In order to address this, scikit-learn provides utilities for the most common ways to extract numerical features from text content, namely:

- **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
- **counting** the occurrences of tokens in each document.
- **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In this scheme, features and samples are defined as follows:

- each **individual token occurrence frequency** (normalized or not) is treated as a **feature**.
- the vector of all the token frequencies for a given **document** is considered a multivariate **sample**.

A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.

We call **vectorization** the general process of turning a collection of text documents into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the **Bag of Words** or "Bag of n-grams" representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document.

6.2.3.2. Sparsity

As most documents will typically use a very small subset of the words used in the corpus, the resulting matrix will have many feature values that are zeros (typically more than 99% of them).

For instance a collection of 10,000 short text documents (such as emails) will use a vocabulary with a size in the order of 100,000 unique words in total while each document will use 100 to 1000 unique words individually.

In order to be able to store such a matrix in memory but also to speed up algebraic operations matrix / vector, implementations will typically use a sparse representation such as the implementations available in the `scipy.sparse` package.

6.2.3.3. Common Vectorizer usage

`CountVectorizer` implements both tokenization and occurrence counting in a single class:

```
>>> from sklearn.feature_extraction.text import CountVectorizer
```

This model has many parameters, however the default values are quite reasonable (please see the [reference documentation](#) for the

Bild: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

CountVectorizer um Text in einzelne Token zu unterteilen

Tokenizing text with scikit-learn

Text preprocessing, tokenizing and filtering of stopwords are all included in `CountVectorizer`, which builds a dictionary of features and transforms documents to feature vectors:

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> count_vect = CountVectorizer()
>>> X_train_counts = count_vect.fit_transform(twenty_train.data)
>>> X_train_counts.shape
(2257, 35788)
```

`CountVectorizer` supports counts of N-grams of words or consecutive characters. Once fitted, the vectorizer has built a dictionary of feature indices:

```
>>> count_vect.vocabulary_.get(u'algorithm')
4690
```

The index value of a word in the vocabulary is linked to its frequency in the whole training corpus.

Bild: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

tf-idf Transformation mit dem TfidfTransformer

From occurrences to frequencies

Occurrence count is a good start but there is an issue: longer documents will have higher average count values than shorter documents, even though they might talk about the same topics.

To avoid these potential discrepancies it suffices to divide the number of occurrences of each word in a document by the total number of words in the document: these new features are called **tf** for Term Frequencies.

Another refinement on top of **tf** is to downscale weights for words that occur in many documents in the corpus and are therefore less informative than those that occur only in a smaller portion of the corpus.

This downscaling is called **tf-idf** for "Term Frequency times Inverse Document Frequency".

Both **tf** and **tf-idf** can be computed as follows using **TfidfTransformer**:

```
>>> from sklearn.feature_extraction.text import TfidfTransformer
>>> tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
>>> X_train_tf = tf_transformer.transform(X_train_counts)
>>> X_train_tf.shape
(2257, 35788)
```

In the above example-code, we firstly use the `fit(..)` method to fit our estimator to the data and secondly the `transform(..)` method to transform our count-matrix to a tf-idf representation. These two steps can be combined to achieve the same end result faster by skipping redundant processing. This is done through using the `fit_transform(..)` method as shown below, and as mentioned in the note in the previous section:

```
>>> tfidf_transformer = TfidfTransformer()
>>> X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
>>> X_train_tfidf.shape
(2257, 35788)
```

Bild: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Training mit MultinomialNB

Training a classifier

Now that we have our features, we can train a classifier to try to predict the category of a post. Let's start with a **naïve Bayes** classifier, which provides a nice baseline for this task. **scikit-learn** includes several variants of this classifier, and the one most suitable for word counts is the multinomial variant:

```
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB().fit(X_train_tfidf, twenty_train.target)
```

To try to predict the outcome on a new document we need to extract the features using almost the same feature extracting chain as before. The difference is that we call **transform** instead of **fit_transform** on the transformers, since they have already been fit to the training set:

```
>>> docs_new = ['God is love', 'OpenGL on the GPU is fast']
>>> X_new_counts = count_vect.transform(docs_new)
>>> X_new_tfidf = tfidf_transformer.transform(X_new_counts)

>>> predicted = clf.predict(X_new_tfidf)

>>> for doc, category in zip(docs_new, predicted):
...     print('%r => %s' % (doc, twenty_train.target_names[category]))
...
'God is love' => soc.religion.christian
'OpenGL on the GPU is fast' => comp.graphics
```

Bild: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Einen zusammengesetzten Classifier mit einer Pipeline konfigurieren

Building a pipeline

In order to make the vectorizer => transformer => classifier easier to work with, `scikit-learn` provides a `Pipeline` class that behaves like a compound classifier:

```
>>> from sklearn.pipeline import Pipeline
>>> text_clf = Pipeline([
...     ('vect', CountVectorizer()),
...     ('tfidf', TfidfTransformer()),
...     ('clf', MultinomialNB()),
... ])
```

The names `vect`, `tfidf` and `clf` (classifier) are arbitrary. We will use them to perform grid search for suitable hyperparameters below. We can now train the model with a single command:

```
>>> text_clf.fit(twenty_train.data, twenty_train.target)
Pipeline(...)
```

Bild: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Kreuzvalidierung: Treffergenauigkeit

```
from sklearn.model_selection import cross_val_score
```

```
scores = cross_val_score(model, X, y)
```

```
print(f'Accuracy: {scores.mean()}')
```

Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

spaCy: Open Source Python Bibliothek für Question Answering

The screenshot shows the spaCy website. At the top, there's a navigation bar with 'spaCy' logo, a 'Take the user survey!' button, and links for 'USAGE', 'MODELS', and 'API'. The left sidebar contains a 'GET STARTED' section with links to 'Installation', 'Models & Languages', 'Facts & Figures', and 'spaCy 101'. The 'spaCy 101' section has a vertical list of topics: 'What's spaCy?', 'Features', 'Linguistic Annotations', 'Pipelines', 'Architecture', 'Vocab', 'Serialization', 'Training', 'Language Data', and 'Community & FAQ'. Below this is a 'New in v3.0' through 'v3.5' section. The main content area is titled 'What's spaCy?' and contains an introduction to spaCy as a free, open-source library for advanced Natural Language Processing (NLP) in Python. It explains that spaCy is designed for production use and helps build applications that process and understand large volumes of text, used for information extraction or natural language understanding systems, or for pre-processing text for deep learning. On the right, there's a 'Table of contents' section with links to various topics: Features, Linguistic annotations, Tokenization, POS tags and dependencies, Named entities, Word vectors and similarity, Pipelines, Library architecture, Vocab, hashes and lexemes, Serialization, Training, Language data, and Community & FAQ.

spaCy ★ Take the user survey! [USAGE](#) [MODELS](#) [API](#)

GET STARTED
[Installation](#)
[Models & Languages](#)
[Facts & Figures](#)
spaCy 101
 ● What's spaCy?
 ● Features
 ● Linguistic Annotations
 ● Pipelines
 ● Architecture
 ● Vocab
 ● Serialization
 ● Training
 ● Language Data
 ● Community & FAQ
[New in v3.0](#)
[New in v3.1](#)
[New in v3.2](#)
[New in v3.3](#)
[New in v3.4](#)
[New in v3.5](#)

What's spaCy?

spaCy is a **free, open-source library** for advanced **Natural Language Processing (NLP)** in Python.

If you're working with a lot of text, you'll eventually want to know more about it. For example, what's it about? What do the words mean in context? Who is doing what to whom? What companies and products are mentioned? Which texts are similar to each other?

spaCy is designed specifically for **production use** and helps you build applications that process and "understand" large volumes of text. It can be used to build **information extraction** or **natural language understanding** systems, or to pre-process text for **deep learning**.

Table of contents

- [Features](#)
- [Linguistic annotations](#)
- [Tokenization](#)
- [POS tags and dependencies](#)
- [Named entities](#)
- [Word vectors and similarity](#)
- [Pipelines](#)
- [Library architecture](#)
- [Vocab, hashes and lexemes](#)
- [Serialization](#)
- [Training](#)
- [Language data](#)
- [Community & FAQ](#)

Bild: <https://spacy.io/usage/spacy-101>

Funktionalität von spaCy

NAME	DESCRIPTION
Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) Tagging	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat".
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named "real-world" objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a knowledge base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model's predictions.
Serialization	Saving objects to files or byte strings.

Bild: <https://spacy.io/usage/spacy-101>

Modelle installieren und Text verarbeiten

```
python3 -m pip install --user spacy
```

```
python3 -m spacy download en_core_web_sm
```

```
python3 -m spacy download de_core_news_sm
```

```
In [1]: 1 import spacy
        2
        3 nlp = spacy.load("en_core_web_sm")
        4 doc = nlp("Hello, World. Here are two sentences.")
        5 print([t.text for t in doc])

['Hello', ',', 'World', '.', 'Here', 'are', 'two', 'sentences', '.']
```

Ähnlichkeitsmetrik mit Wortvektoren (word vectors)

```
python3 -m spacy download en_core_web_lg # großes package
```

Word vectors and similarity NEEDS MODEL ?

Similarity is determined by comparing **word vectors** or "word embeddings", multi-dimensional meaning representations of a word. Word vectors can be generated using an algorithm like [word2vec](#) and usually look like this:

BANANA.VECTOR

```
array([2.02280000e-01, -7.66180009e-02,  3.70319992e-01,
       3.28450017e-02, -4.19569999e-01,  7.20689967e-02,
      -3.74760002e-01,  5.74599989e-02, -1.24009997e-02,
       5.29489994e-01, -5.23800015e-01, -1.97710007e-01,
      -3.41470003e-01,  5.33169985e-01, -2.53309999e-02,
       1.73800007e-01,  1.67720005e-01,  8.39839995e-01,
       5.51070012e-02,  1.05470002e-01,  3.78719985e-01,
       2.42750004e-01,  1.47449998e-02,  5.59509993e-01,
       1.25210002e-01, -6.75960004e-01,  3.58420014e-01,
       # ... and so on ...
       3.66849989e-01,  2.52470002e-03, -6.40089989e-01,
      -2.97650009e-01,  7.89430022e-01,  3.31680000e-01,
      -1.19659996e+00, -4.71559986e-02,  5.31750023e-01], dtype=float32)
```

Bild: <https://spacy.io/usage/spacy-101>

Ähnlichkeitsmetrik anwenden (similarity metric)

```
In [14]: 1 import spacy
          2
          3 nlp = spacy.load("en_core_web_lg")
          4 doc = nlp("Mango and ananas are similar. Pizza and dogs aren't.")
          5
          6 mango = doc[0]
          7 ananas = doc[2]
          8
          9 pizza = doc[6]
         10 dogs = doc[8]
         11
         12 print("mango vs. ananas", mango.similarity(ananas))
         13 print("pizza vs. dogs", pizza.similarity(dogs))
         14
         15 print(mango.has_vector, ananas.has_vector, pizza.has_vector, dogs.has_vector)
         16 print(mango.vector)
         17
```

mango vs. ananas 0.524490237236023

pizza vs. dogs 0.03017667680978775

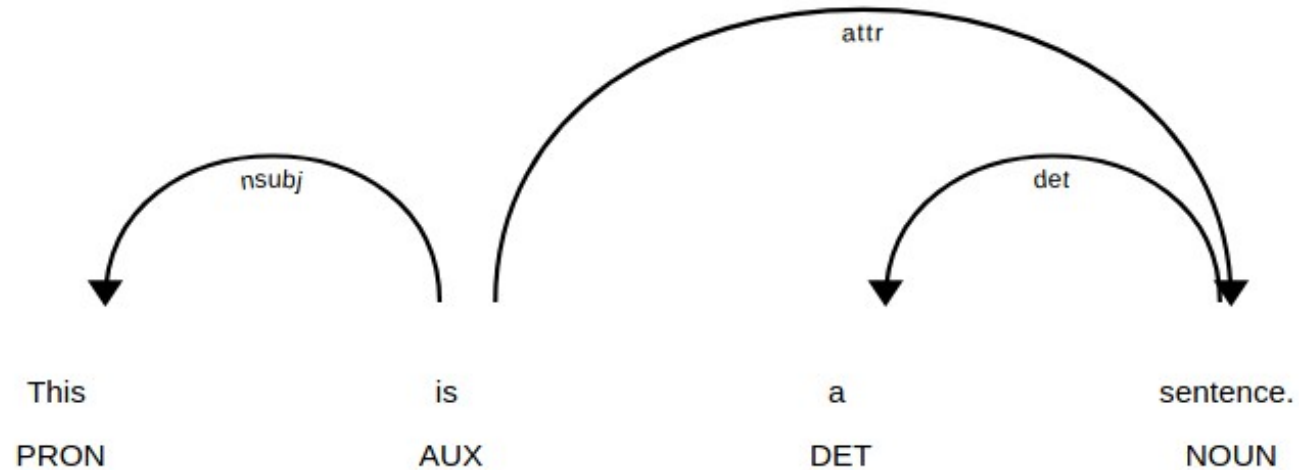
True True True True

```
[ 1.6968e-01 -1.1406e+00  4.2686e-01 -1.1662e-01  3.1422e+00 -1.7256e+00
 8.5684e-01 -1.8575e+00 -3.0863e+00  9.7959e-01 -1.8544e+00 -7.3673e-02
 1.7760e+00  1.4921e+00 -1.4639e+00 -4.2563e+00  3.9893e-01 -1.4180e+00
 2.1238e+00 -2.6686e+00 -6.1075e-01  3.6439e+00  8.2714e-01  1.4939e-01
 5.1963e-01  5.0651e-02 -1.0765e+00 -1.1264e+00 -1.1290e+00 -4.1344e+00]
```

Fortgeschrittenes NLP: Sätze parsen

In [17]:

```
1 from spacy import displacy
2
3 doc = nlp("This is a sentence.")
4 displacy.render(doc, style="dep")
5
```



spaCy Architektur und Überblick

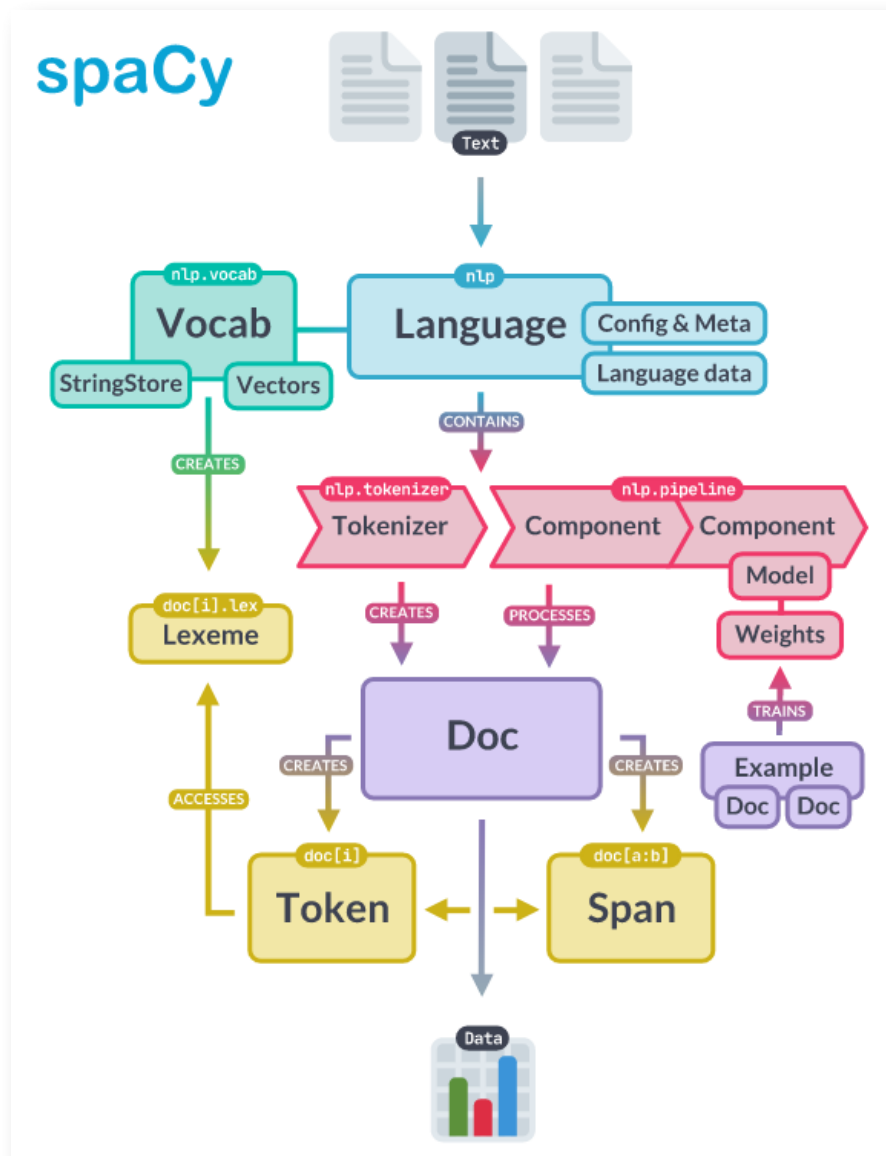


Bild: <https://spacy.io/usage/spacy-101>

Container objects

NAME	DESCRIPTION
<code>Doc</code>	A container for accessing linguistic annotations.
<code>DocBin</code>	A collection of <code>Doc</code> objects for efficient binary serialization. Also used for training data .
<code>Example</code>	A collection of training annotations, containing two <code>Doc</code> objects: the reference data and the predictions.
<code>Language</code>	Processing class that turns text into <code>Doc</code> objects. Different languages implement their own subclasses of it. The variable is typically called <code>nlp</code> .
<code>Lexeme</code>	An entry in the vocabulary. It's a word type with no context, as opposed to a word token. It therefore has no part-of-speech tag, dependency parse etc.
<code>Span</code>	A slice from a <code>Doc</code> object.
<code>SpanGroup</code>	A named collection of spans belonging to a <code>Doc</code> .
<code>Token</code>	An individual token — i.e. a word, punctuation symbol, whitespace, etc.

Bild: <https://spacy.io/usage/spacy-101>

Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

Aufgabe: Einfache Fragen über den Film “A Beautiful Mind” beantworten

A Beautiful Mind is a 2001 American biographical drama film directed by Ron Howard. Written by Akiva Goldsman, its screenplay was inspired by Sylvia Nasar's 1998 biography of the mathematician John Nash, a Nobel Laureate in Economics. A Beautiful Mind stars Russell Crowe as Nash, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles. The story begins in Nash's days as a brilliant but asocial mathematics graduate student at Princeton University. After Nash accepts secret work in cryptography, his life takes a turn for the nightmarish.

Text: [https://en.wikipedia.org/wiki/A_Beautiful_Mind_\(film\)](https://en.wikipedia.org/wiki/A_Beautiful_Mind_(film))

Beispiel

Beispielfrage

Which price was
John Nash
awarded?



The Nobel price
in Economics

Bild: https://upload.wikimedia.org/wikipedia/commons/5/5c/Amazon_Echo.jpg

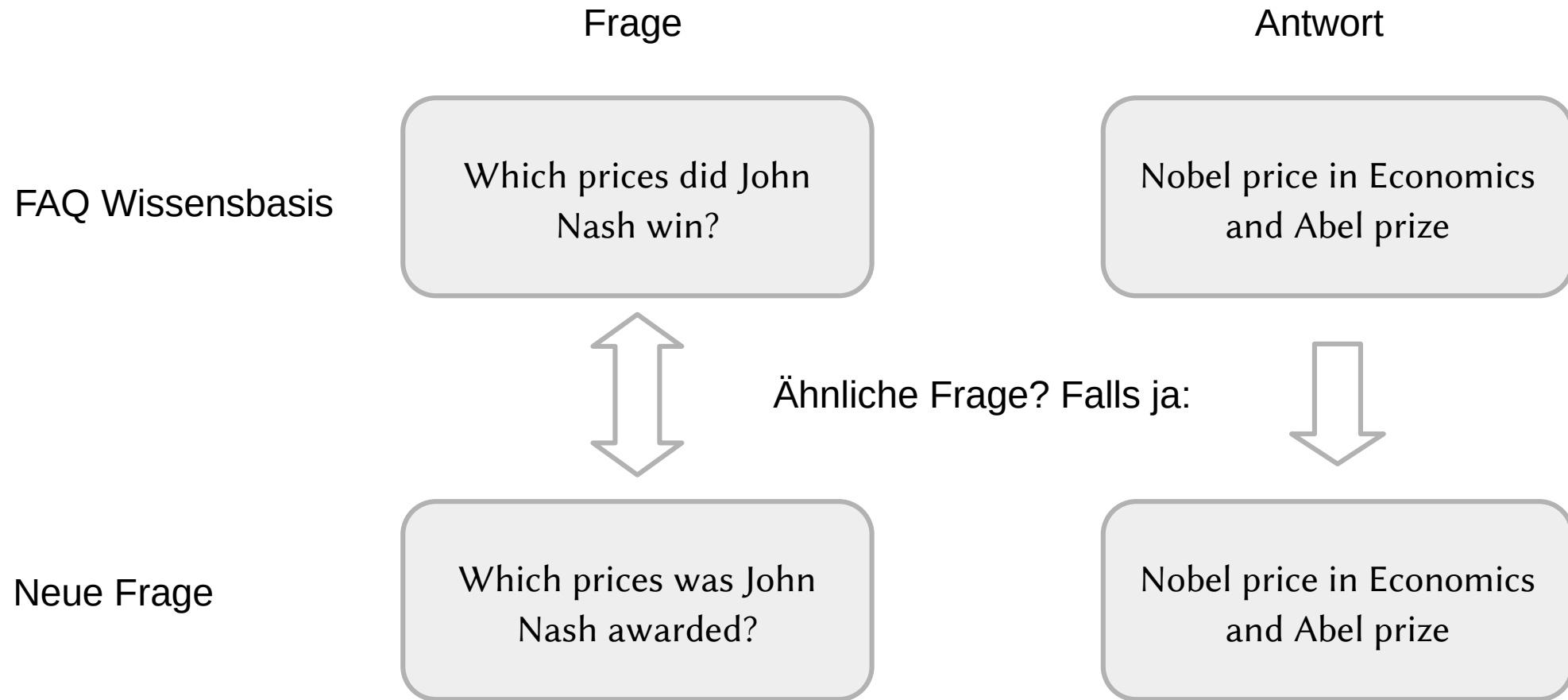
Textbasis für die Antwort

[...] biography of the mathematician John Nash, a Nobel Laureate in Economics.

Einfaches QA: FAQ als eine Wissensbasis (Trainingdaten)

	A	B
1	Question	Answer
2	From which country is the film A Beautiful Mind?	America
3	When was the film A beautiful Mind released?	2001
4	Of which genre is the film A Beautiful Mind?	biographical drama
5	What is John Nash's profession?	mathematician
6	Which prizes did John Nash win?	Nobel price in Economics and Abel Prize
7	By which book was the film inspired?	the bestselling Pulitzer Prize-nominated 1998 book of the same name by Sylvia Nasar
8	Who is acting in the film?	Russell Crowe, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles
9	At which university was Nash a graduate student?	Princeton University
10	What disease did Nash develop?	paranoid schizophrenia
11	When did the film open in the United states?	December 21, 2001
12	How much did the film gross?	over \$313 million worldwide
13	How many Academy Awards did the film win?	four
14	Did the film win an Academy Award?	yes

Aufgabe im einfachen QA: Antworten auf ähnliche Fragen finden



Testdaten mit neuen Fragen zum Beantworten

	A	B
1	Question	Answer
2	From which country is the film A Beautiful Mind?	America
3	Where was the film A Beautiful Mind made?	America
4	In which country was the film A Beautiful Mind made?	America
5	Where was the movie A Beautiful Mind made?	America
6	From which country is the film?	America
7	From which country is this film?	America
8	When was the film A beautiful Mind released?	2001
9	Of which genre is the film A Beautiful Mind?	biographical drama
10	Which kind of film is A Beautiful Mind?	biographical drama
11	Which type of movie is A Beautiful Mind?	biographical drama
12	What is John Nash's profession?	mathematician
13	What is John Nash's job?	mathematician
14	Which prices did John Nash win?	Nobel price in Economics and Abel Prize
15	Which awards did John Nash win?	Nobel price in Economics and Abel Prize
16	Which prices was John Nash awarded?	Nobel price in Economics and Abel Prize
17	By which book was the film inspired?	the bestselling Pulitzer Prize-nominated 1998 book of the same name by Sylvia Nasar
18	Which book is the basis of the film?	the bestselling Pulitzer Prize-nominated 1998 book of the same name by Sylvia Nasar
19	Who is acting in the film?	Russell Crowe, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles
20	Who are the actors in the film?	Russell Crowe, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles
21	Who is starring in this film?	Russell Crowe, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles
22	Who is acting in the movie?	Russell Crowe, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles
23	At which university was Nash a graduate student?	Princeton University
24	Where did John Nash graduate?	Princeton University

Daten laden

```
import pandas as pd

train_data = pd.read_csv('data/QA_train_data.csv')
test_data = pd.read_csv('data/QA_test_data.csv')
```

Vorhersagelogik

```
def predict_answer(question: str)-> str:
    """
    Predicts an answer to a given question
    :param question: question string
    :return: answer string or None if insufficient confidence
    """
    # Your code...
```

Vorhersagen

```
# Predict  
predicted = test_data['Question'].apply(lambda q: predict_answer(q))  
actual = test_data['Answer']
```

Ergebnisse evaluieren

Evaluate

score = evaluate_results(predicted, actual)

```
def evaluate_result(predicted: str, actual: str) -> int:
    """
    Evaluates individual string answer string pair
    :param predicted: answer predicted by predictor
    :param actual: correct answer according to data set
    :return: +1 if predicted answer is correct,
    0 if no answer is predicted (None),
    -1 if answer is wrong
    """
    if predicted is None:
        return 0
    elif predicted == actual:
        return 1
    else:
        return -1
```

```
def evaluate_results(predicted: pd.Series, actual: pd.Series) -> float:
    """
    Evaluates a series of answer pairs
    :param predicted: Series of predicted answers
    :param actual: Series of correct answers
    :return: Float value between -1 (worst) and +1 (best)
    """
    sum = 0
    for index, value in predicted.items():
        eval = evaluate_result(value, actual[index])
        sum += eval
    result = sum / predicted.size
    return result
```

Ideen für die QA Implementation ?



Bildquelle: https://cdn.pixabay.com/photo/2015/11/03/08/56/question-mark-1019820_960_720.jpg

Aufbau

- Überblick
- Das bag-of-words Modell
- Beispiel Textklassifikation
- spaCy
- Beispiel Einfaches Question Answering
- Mini-Test

Mini-Test “Natural Language Processing”

- Benennen und erklären Sie die verschiedenen Bereiche des NLP
- Was ist die Aufgabe der Textklassifikation?
- Erklären Sie das bag-of-words Modell
- Erklären Sie tf-idf
- Erklären Sie das n-Gramm Modell
- Wie benutzt man Pipelines in scikit-learn?
- Was ist die NLP Aufgabe Question Answering?
- Erklären Sie die Funktionalität von spaCy