Introduction to Natural Language Processing

CAS AML M6

Dr. Mykhailo Vladymyrov

Overview

- Key concepts
- Common pipelines and approaches
- Examples of common tasks

Lecture prepared with some material adopted from Ahmad Alhineidi

Natural Language?

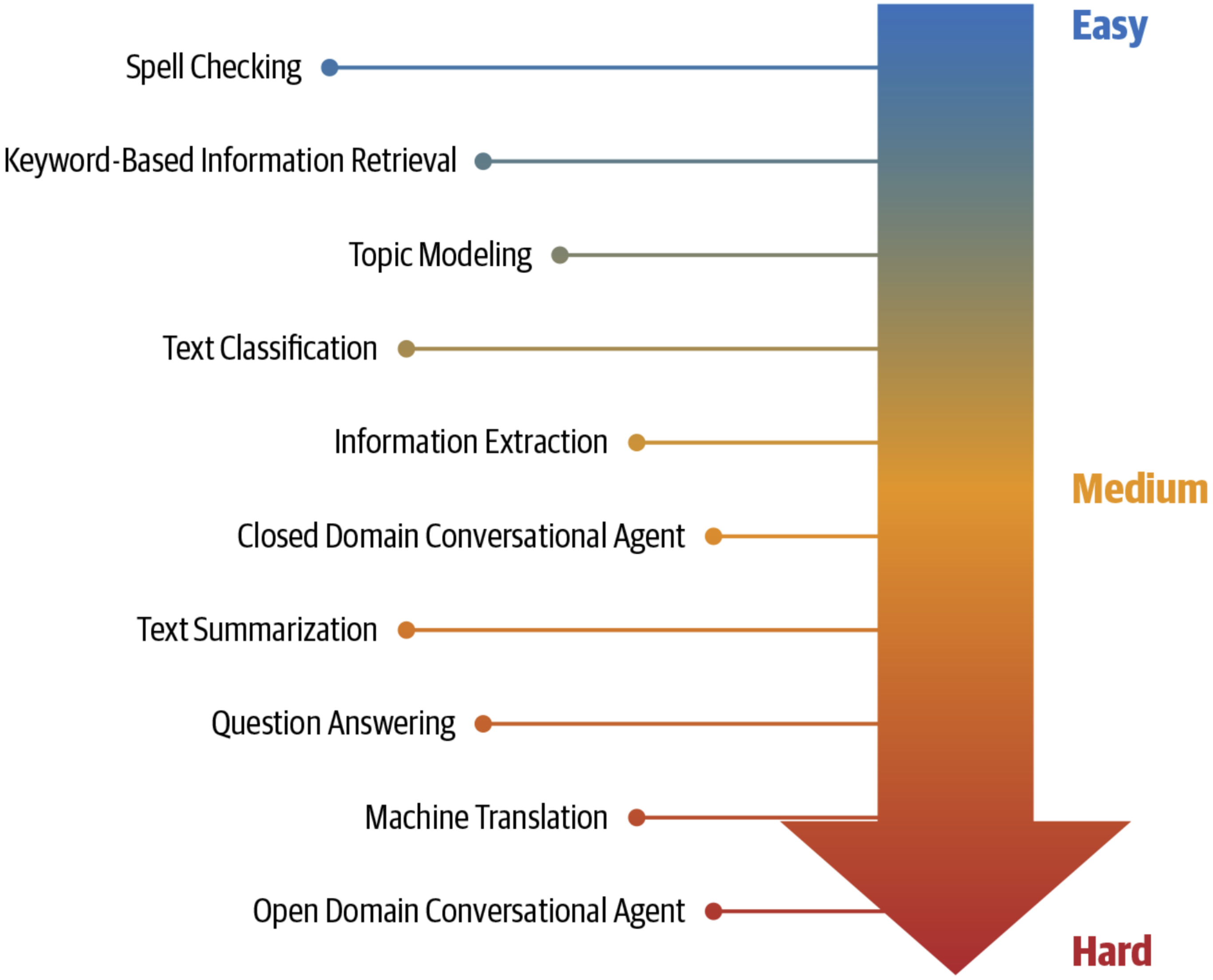
- “…network of constructions..”
- “C is a construction iff C is a form meaning pair <F, S> such that some aspects of F or some aspects of S is not strictly predictable from C’s component parts or from other previously established constructions.”

Goldberg, A. E. (1995).*Constructions: A construction grammar approach to argument structure*. University of Chicago Press.

Common NLP tasks and applications

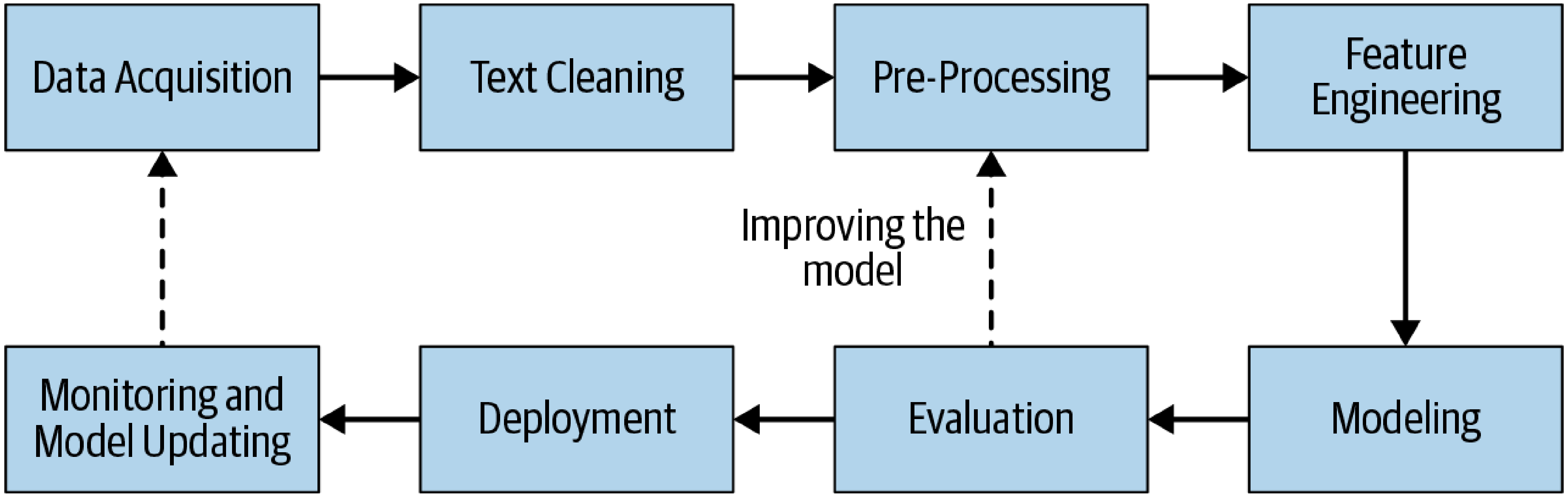
- Text classification (Sentiment analysis, spam detection, topic labeling)
- Named Entity Recognition (NER) (Information extraction, content recommendation)
- Machine Translation (Content Localization, real time translation)
- Text Summarization (News Aggregation, Research)
- Question Answering (Customer Support)
- Speech Recognition (Voice Assistants)
- Text Generation (Content Creation, chatbots)

Common NLP tasks



Source: Vajjalaet al. 2020

NLP Processing pipeline



Source: Vajjalaet al. 2020

What does the model learn?

What does the model learn?

Statistical relations between:

- Words
- Concepts
- Parts of words
- ….

Word embedding

- Problem: Computers and most NLP methods cannot operate on text
- Solution: Have a numerical representation for words as multidimensional vectors (word embedding)

Word embedding

How do you create meaningful numerical representations of words?

- Semantic similarities
- Syntactic similarities
- Ideally, similar words are close to each other in a vector space

Word embedding

Common word embedding algorithms:

- BOW (bag of words)
- TF-IDF (term frequency - inverse document frequency)
- One hot encoding
- CBOW (Continuous Bag of Words)
- Fasttext (sub-word)
- word2vec
- GloVe

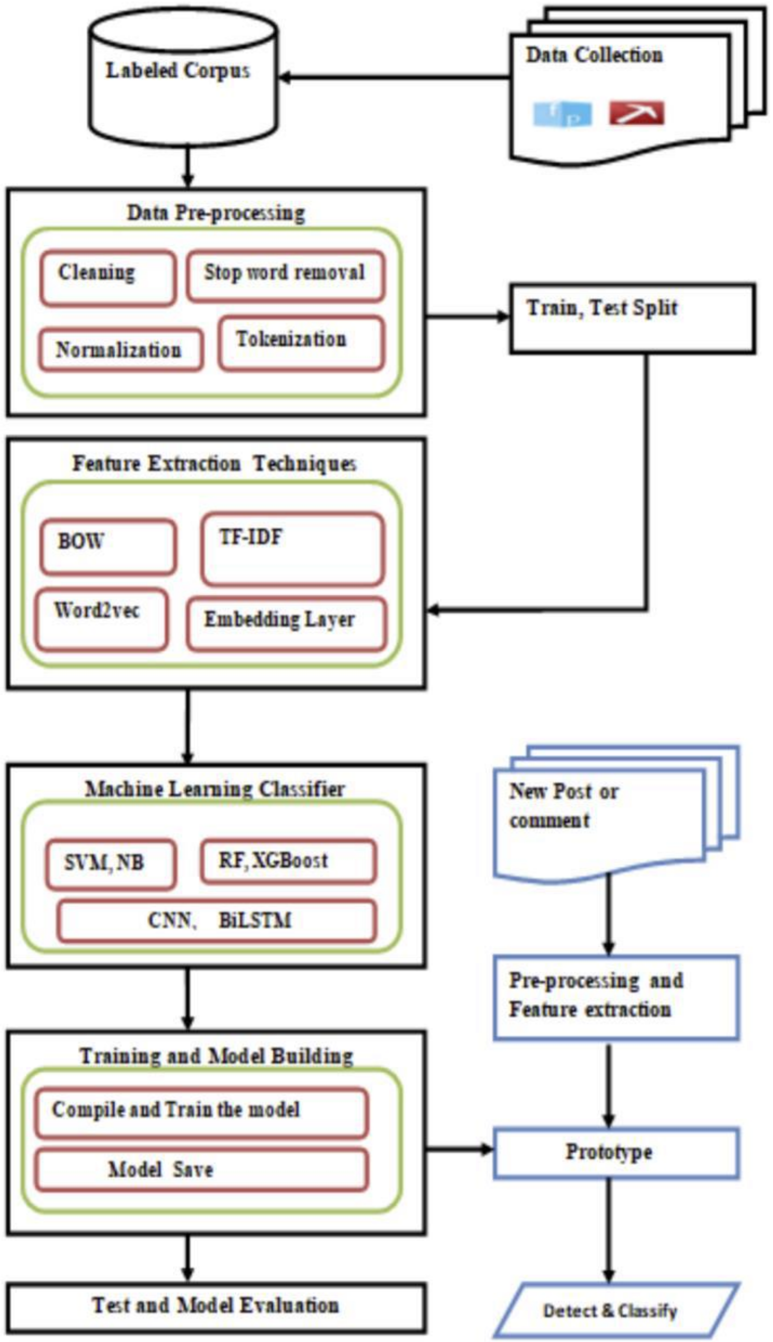
Word embedding

Experiment result	Accuracy in percentage			
	Feature Extraction Techniques			
Algorithms	BOW	TF-IDF	Pre-trained Word2vec	Embedding Layer
SVM	0.78	0.80	0.82	-
NB	0.80	0.80	0.74	-
RF	0.79	0.79	0.81	-
XGBoost	0.80	0.77	0.81	-
CNN	-	-	0.81	0.82
BI-LSTM	-	-	**0.84**	0.81

Table 5: Eight classes experiment result with classical, ensemble, Deep ML classifier

Source:Ababu, Teshome Mulugeta, and Michael MeleseWoldeyohannis. "AfaanOromo hate speech detection and classification on social media." Proceedings of the thirteenth language resources and evaluation conference. 2022.

Word embedding



Source:Ababu, Teshome Mulugeta, and Michael MeleseWoldeyohannis. "AfaanOromo hate speech detection and classification on social media." Proceedings of the thirteenth language resources and evaluation conference. 2022.

BOW

Represent the document as a vector of words

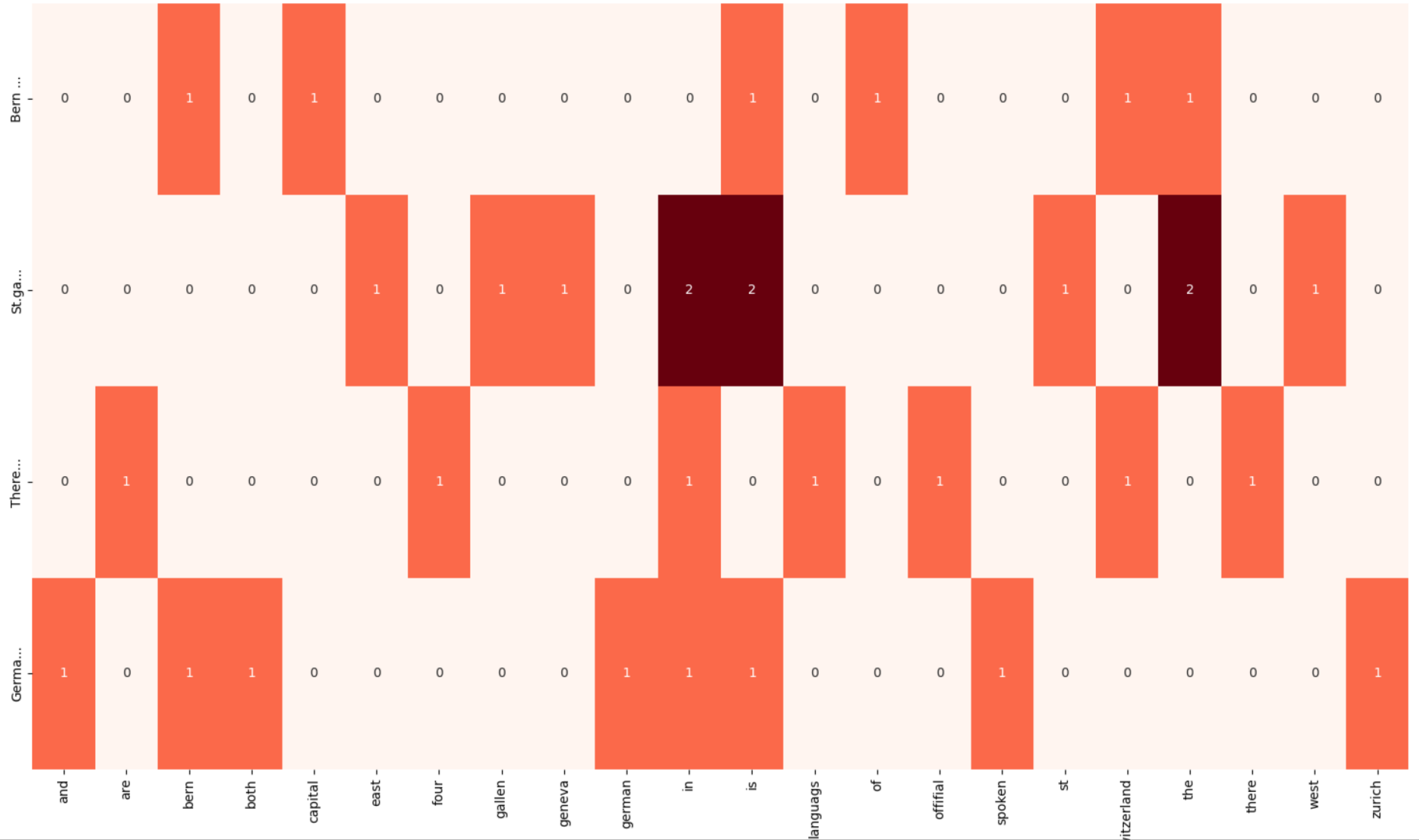
- Given a document with size vocabulary |V|, we represent the document as a vector of size |V|
- Each cell of the vector contains a word that is represented by its number of occurrences in the document
- Used in document classification, information extraction, and other NLP tasks
- All syntactic and semantic components of words are lost
- Vector size can be huge in large documents, and you need to reduce it (lemmatization, stemming, stopwords removal, etc.)

BOW

Heatmap for BOW

```
Import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
corpus =["Bern is the capital of Switzerland.",
"St.gallenis in the east. Genevaisinthewest.",
"There are four official languages in Switzerland",
"German is Spoken in both Bern and Zurich"]
one_hot_vectorizer=CountVectorizer(token_pattern=r"(?u)\b\w+\b")
one_hot=one_hot_vectorizer.fit_transform(corpus).toarray()
sns.heatmap(one_hot, annot=True, cmap="Reds", cbar=False,
xticklabels=one_hot_vectorizer.get_feature_names_out(),
yticklabels=[s[0:5]+"..."for s incorpus])
plt.show()
```

BOW


Bern ...
0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0
St.ga...
0 0 0 0 0 1 0 1 1 0 2 2 0 0 0 0 1 0 2 0 1 0
There...
0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0
Germa...
1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 1
and
are
bern
both
capital
east
four
gallen
geneva
german
in
is
languags
of
offifal
spoken
st
itzerland
the
there
west
zurich

TF-IDF

- The product of two measurements: (Term frequency) –(inverse document frequency)
- tf-idf(w,d) = tf(w,d)* idf(w)
- TF-IDF tells us how important a word is in a collection of documents or corpora
- The importance of a word increases with the number of times it occurs in a document but decreases with the number of times it appears in documents.

Why is this useful?

TF-IDF

The term frequency of a word is the total number of times the words occurs in a document divided by the total number of words

$$\text{TF(w, d)} = \frac{number\ of\ occurances\ of\ w\ in\ d}{number\ of\ words\ in\ d}$$

TF-IDF

- High frequency words such as stopwords are high in term frequency but do not provide much use to NLP algorithms.
- The IDF resolves the issue by reducing the value of highly frequent words that occur in every document
- Document Frequency: the number of documents a word occurs in, regardless of how many times it occurs
- Inverse document frequency: The number of documents in a corpus divided by document frequency
- Words that occur in many documents have low inverse document frequency, while a word that occurs in fewer has a higher IDF
- We use the log to calculate the IDF

$$\text{IDF(w)} = \log\frac{\textit{number of documents in a corpus}}{\textit{number of documents in which w occurs in}}$$

One hot encoding of words

- Vocabulary of length N, each word is a vector of length N of zeros except the index of the word with 1
- No semantic or syntax information
- computationally expensive for large corpus
- Rarely used in modern NLP applications

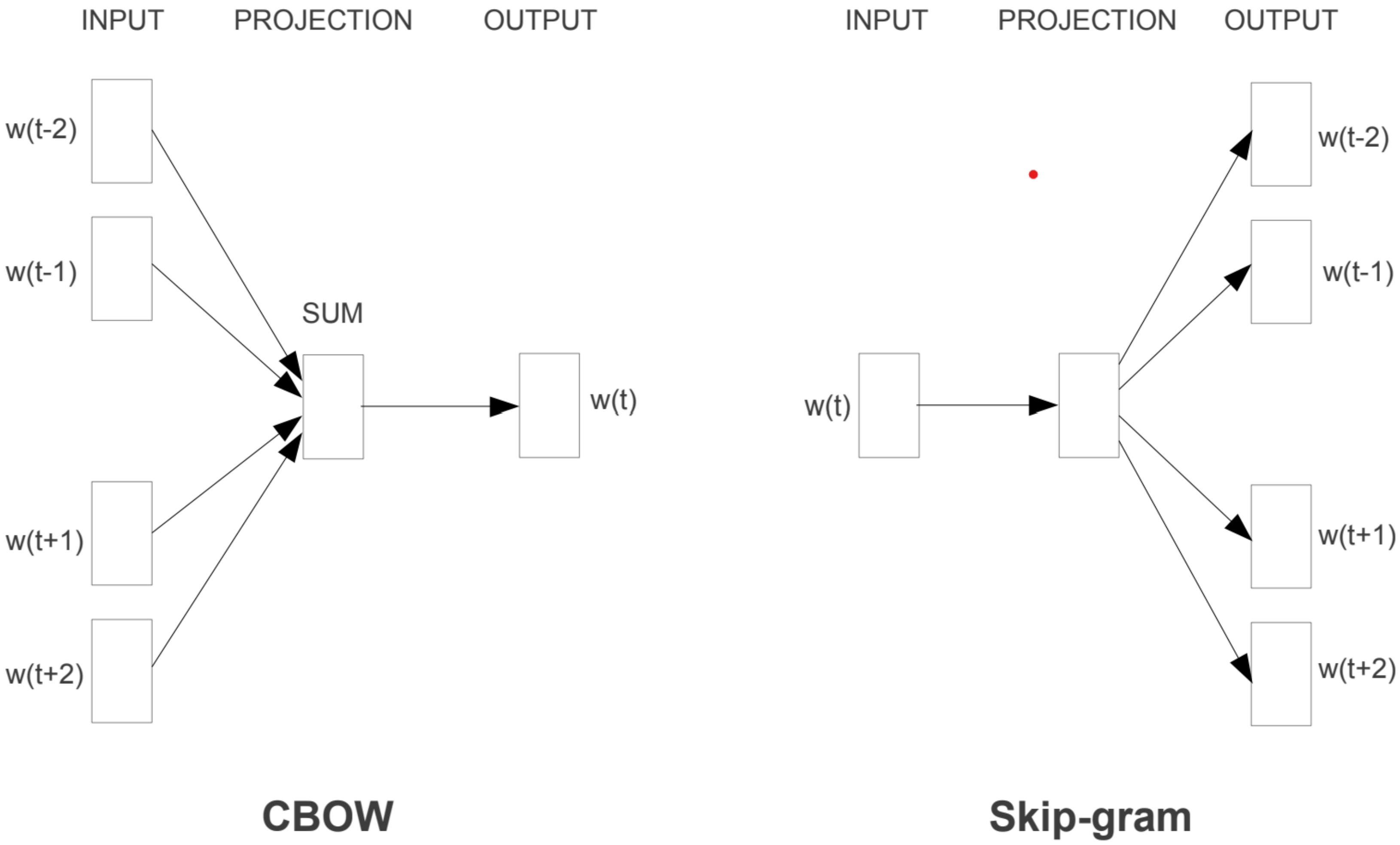
	1	2	3	4	5	6	7	8
I	1	0	0	0	0	0	0	0
ate	0	1	0	0	0	0	0	0
an	0	0	1	0	0	0	0	0
apple	0	0	0	1	0	0	0	0
and	0	0	0	0	1	0	0	0
played	0	0	0	0	0	1	0	0
the	0	0	0	0	0	0	1	0
piano	0	0	0	0	0	0	0	1

Image credit: towards data science website

Word2Vec

- Neural network for word embedding
- Has only one hidden layer and uses one of these techniques:1-CBOW: predict the target word from its surrounding 2-Skip-gram: predict the surrounding words from the target word
- The output is a multi-dimensional vector space
- Words with close semantic and syntactic relations are close to each other
- Useful to get synonyms

Word2Vec

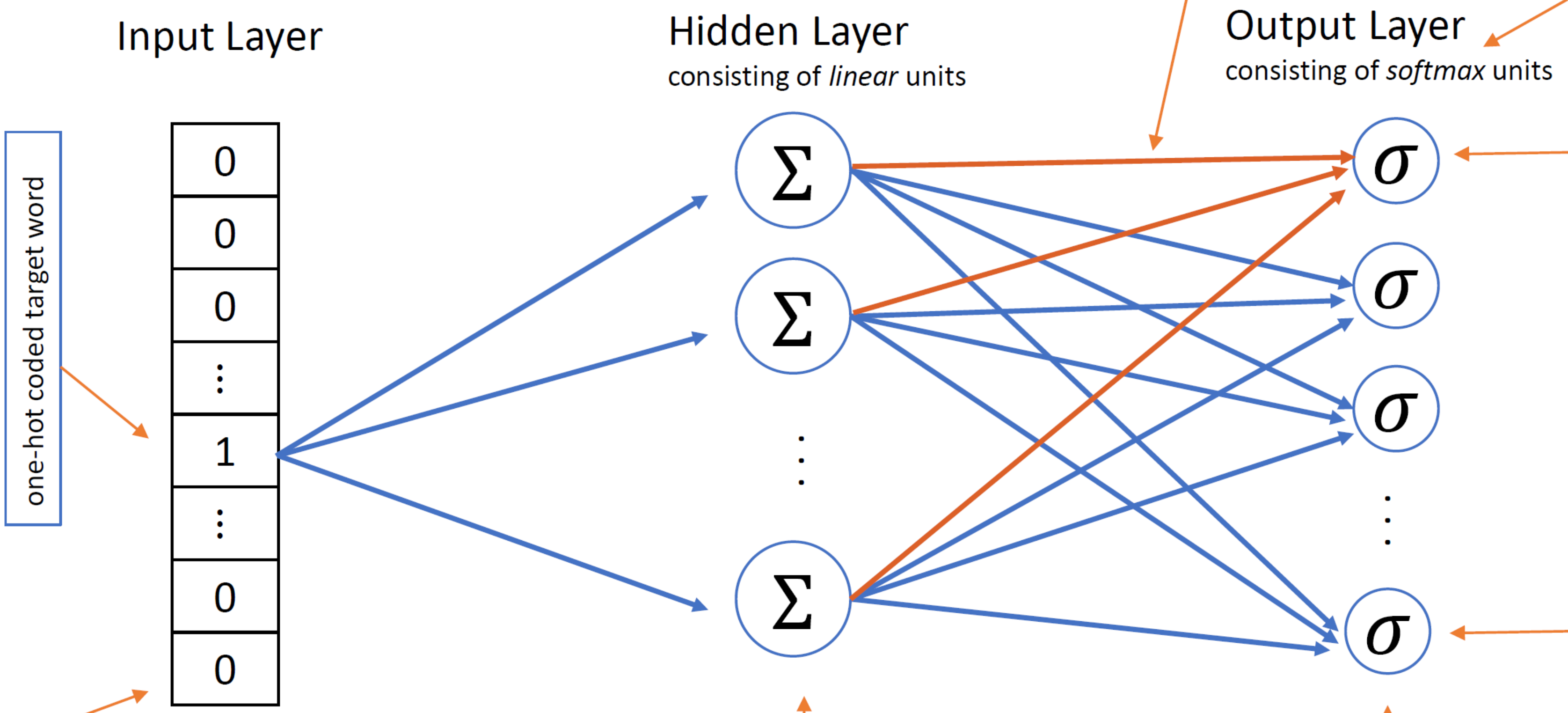


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Word2vec: Skip-gram

The embedding of the first word of the vocabulary is highlighted in red.

The *softmax*() function takes a vector as input and transforms the values such that their sum equals 1, i.e. the resulting vector can be interpreted as a probability distribution.



Probability that the first word of the vocabulary (e.g., "a") is in the context window of the input word.

Probability that the last word of the vocabulary (e.g., "zoom") is in the context window of the input word.

The input vector is very long: its length corresponds to the number of words in the vocabulary.

The number of units in the hidden layer is a hyperparameter of the model, i.e., it is tuned on the data. It is *much* smaller than the number of units (=cells) in the input layer.

The output layer has the same number of units as the input layer, i.e., it has one unit for each word of the vocabulary.

Source, Lena Jäger, 2020

Word2Vec

Extensions:

- Fasttext - Subword Information:

  n - grams: f("apple", n=3) -->[<ap , app, ppl , ple , le>]

- GloVe: word-word co-occurrence matrix (how frequently words co-occur with one another in a given corpus)

Word2Vec

[Visualization]

Tools to look into

Annotation

- Prodigy

Classic text processing

- NLTK
- Fasttext

HF Transformers library

Datasets:

Standard datasets in sklearn

Standard datasets in HF

Kaggle

SRF API