BERN WINTER SCHOOL – NATURAL LANGUE PROCESSING

Ahmad Alhineidi

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 to create meaningful numerical representation of words?
- Semantic similarities
- Syntactic similarities
- Ideally similar words are close to each other in a vector space

Word embedding

- Common word embedding algorithms:

  1- BOW (bag of words)
  2- TF-IDF (term frequency- inverse document frequency)
  3- One-hot encoding
  4- CBOW (Continuous Bag of Words)
  5- word2vec
  6- skip-gram
  7- Fasttext
  8- GloVe
  9- others

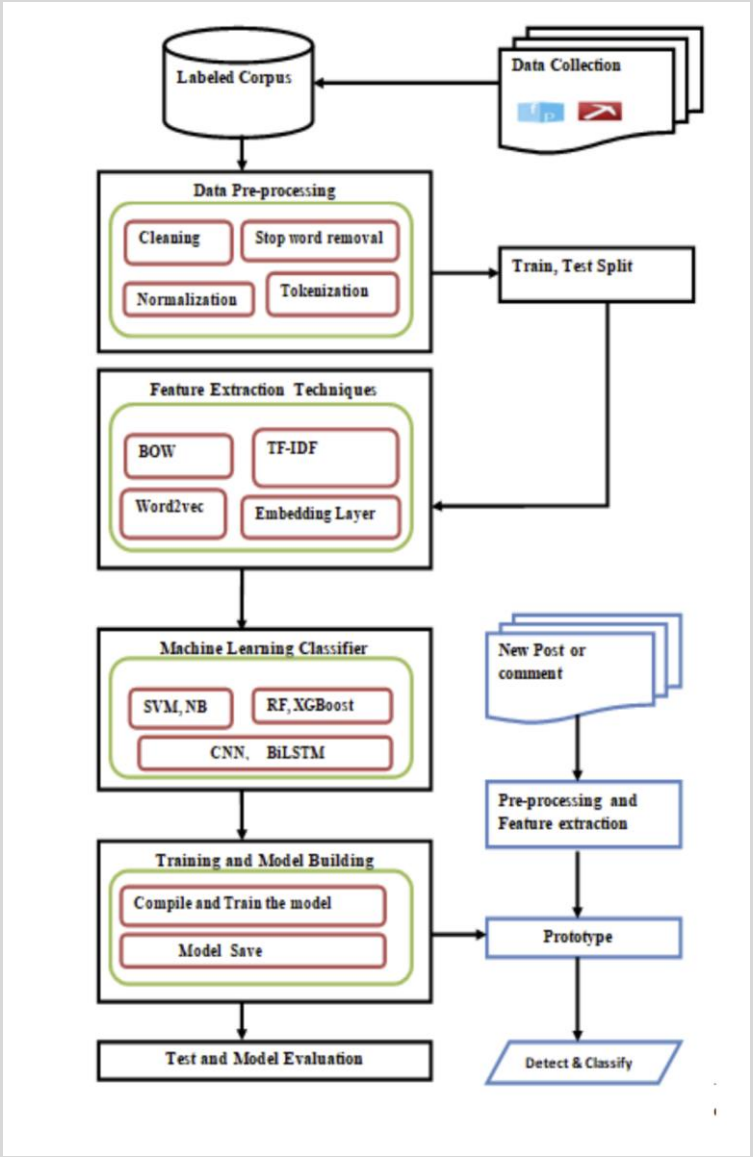
- Or Static embeddings vs. contextualized embedding?

Word embedding

Experiment result	Accuracy in percentage			
	Feature Extraction Techniques			
Algorithms	BOW	TF-IDF	Pre-trained Word2vec	Embedd-ing 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, en-semble, Deep ML classifier

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

Word embedding (BOW)

- Represent 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 with its number of occurrence in the document
- Used in document classification, information extraction, 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)

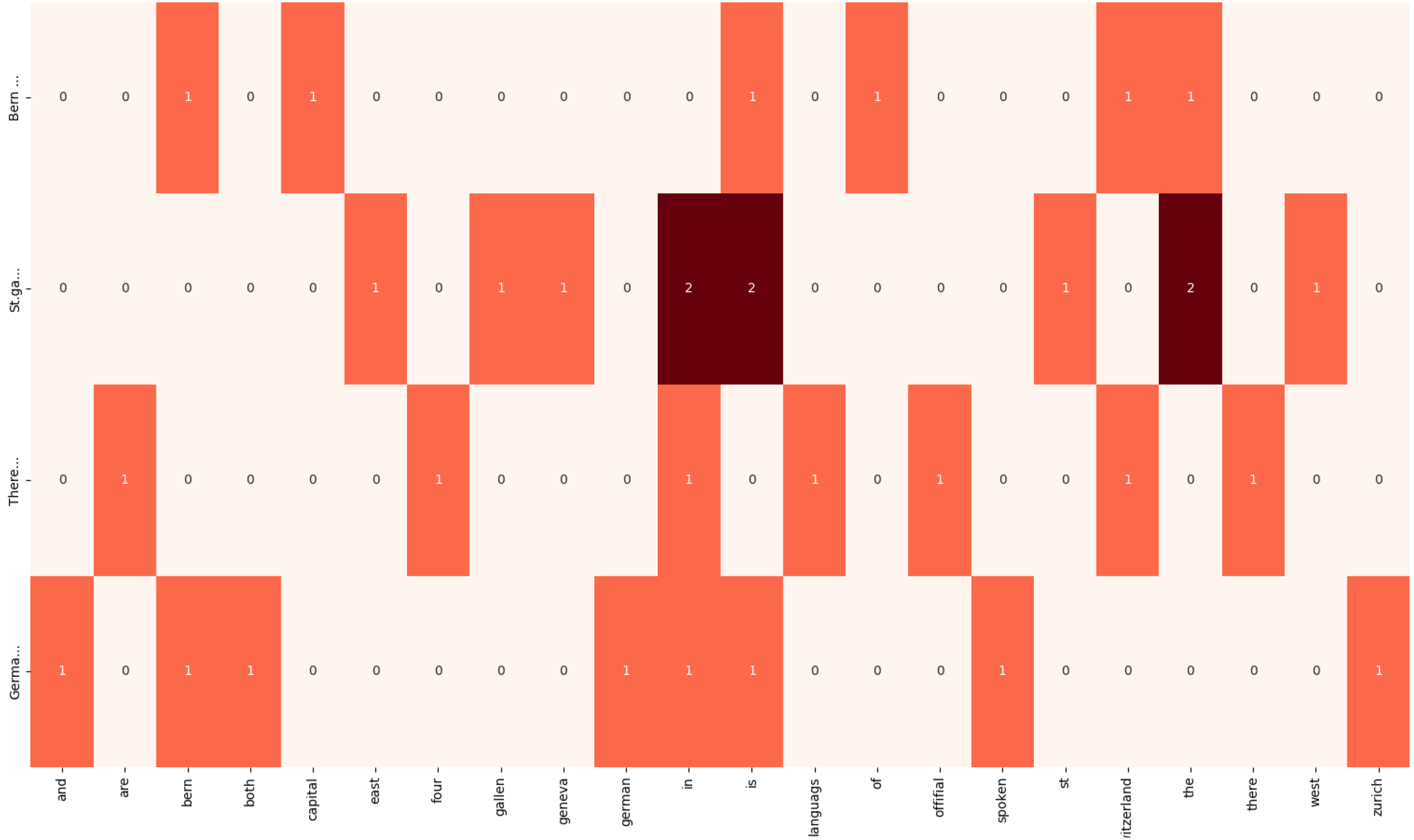
Word embedding (BOW)

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.gallen is in the east. Geneva is in the west.",
            "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 in corpus])
plt.show()
```


Bern ...
St.ga...
There...
Germa...
0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 1 1 0 2 2 0 0 0 0 1 0 2 0 1 0
0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0
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
offifial
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 or documents or corpora
- The importance of a word increases in relation to the number of times it occurs in a document, but decreases to the number of times it appears in documents. Why this is useful?

TF (Term Frequency)

- Term frequency of a word is the total number of times the words occurs in a document divided by the total numbers of words

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

IDF (Inverse Document Frequency)

- 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 occurs in every document
- Document Frequency: 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 occurs 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

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

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: towardsdatascience website

FINDING WORDS NEIGHBORS

link to visualisation

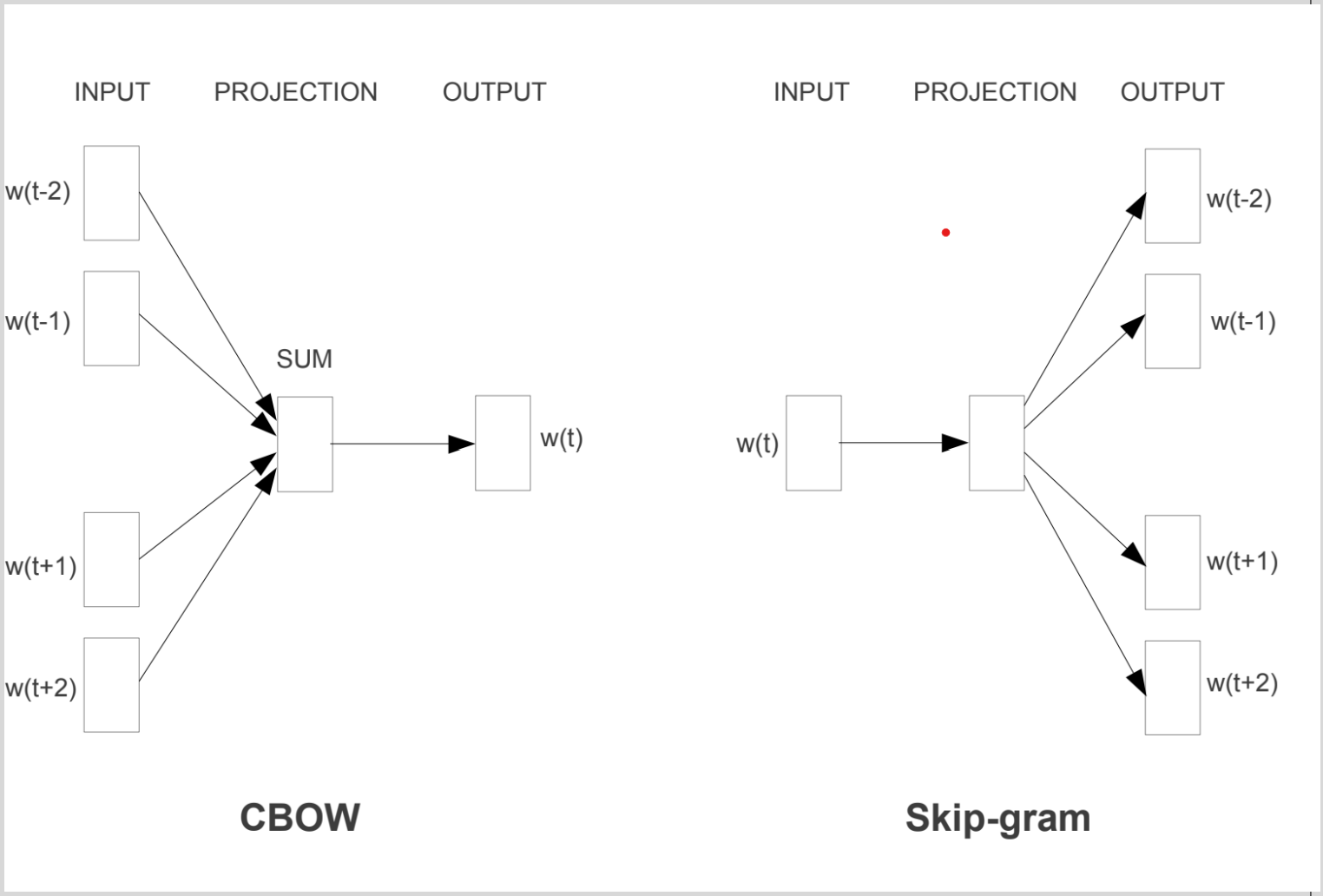
Generating similar words

- Algorithm to transfer text into vector space with similar words close to each other
- Cosine similarity to calculate distance between words
- Algorithm to get the nearest neighbor of the target word (e.g. Hierarchical Navigable Small World Graphs (HNSW))

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 relation are close to each other
- Useful to get synonyms

Word2Vec



Source: Mikolov et al., 2013.

Word2Vec (Skip-gram)

- Training the model
- Given a corpus for training, we move a window with desired context size to get training samples:

Sarah lives in Zurich but works in Bern. → (sarah, lives), (sarah, in)
Sarah lives in Zurich but works in Bern. → (lives, sarah), (lives, in), (lives, zurcih)
Sarah lives in Zurich but works in Bern. → (in, sarah), (in, lives), (in, zurcih), (in, but)

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.

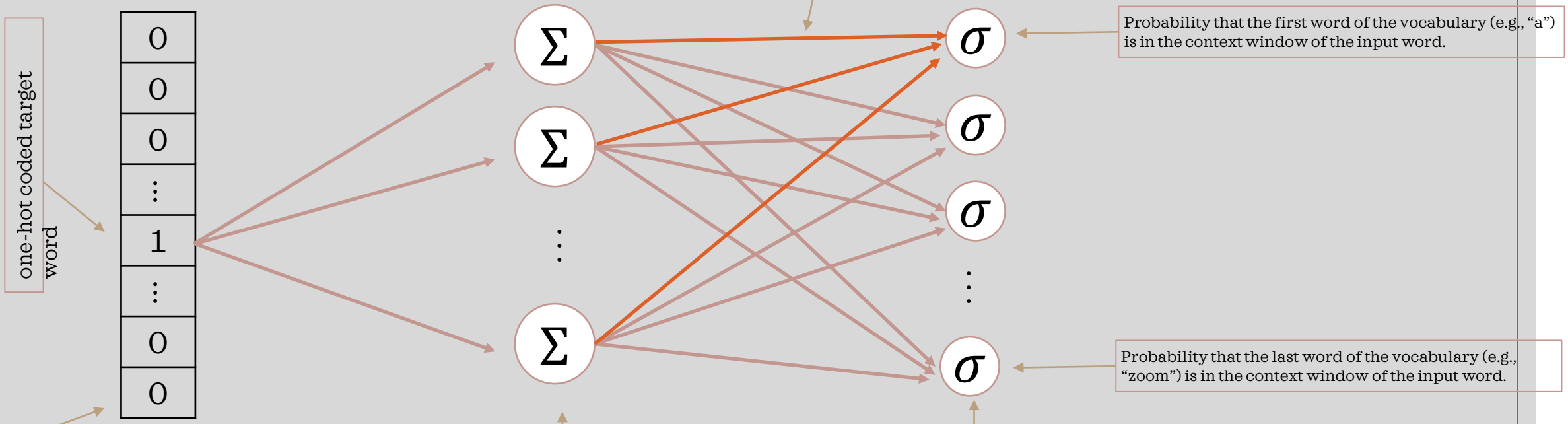
Input Layer

Hidden Layer

consisting of *linear* units

Output Layer

consisting of *softmax* units



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

SIMPLIFIED WORD2VEC SKIPGRAM TRAINING

Notebook link

Word2vec

- Is word2vec the best embedding system?
- What about out-of-Vocabulary Words?

Fasttext

- Extension of word2vec
- Developed by FAIR (Facebook AI Research)
- Easy to use module for Linux, MacOS link
- For windows use genism wrapper Link
- Word vectors for 157 languages
- Or train the model on your own data link
- The model is also built to work as a CL tool
- The python and CL model includes text classification trainer

Fasttext

- “Enriching Word Vectors with Subword Information”
- Character N-grams: f(“apple”, n=3) -> [<ap, app, ppl, ple, le>]
- < and > denote beginning and end of word
- A word is represented as the sum of its n-gram
- Very useful for highly inflectional languages
- Skipgram or CBOW architectures
- Code example for training, saving and using Fasttext embedding. Fasttext_embedding.ipynb

Source: Bojanowski, Piotr, et al. "Enriching word vectors with subword information." Transactions of the association for computational linguistics 5 (2017): 135-146.

GloVe (Global Vector)

- Also semantic relations
- Stanford University link
- word-word co-occurrence matrix (how frequently words co-occur with one another in a given corpus)
- The model chooses a window size for co-occurrences and learns the embeddings

GloVe (Global Vector)

	i	like	natural	language	processing	is	key	to
i	0	1	0	0	0	0	0	0
like	1	0	1	0	0	0	0	0
natural	0	1	0	1	0	0	0	0
language	0	0	1	0	1	1	0	0
processing	0	0	0	1	0	0	0	0
is	0	0	0	1	0	0	1	0
key	0	0	0	0	0	1	0	1
to	0	0	0	0	1	0	1	0

- Given a corpus example:
[I like natural language processing, language is key to processing]
- We create co-occurrence matrix with window of 1

Word embedding in Python

- BOW is implemented with sklearn in python Check Documentation
- TF-IDF also in sklearn Documentation
- Word2vec in genism Documentation
- Fasttext in Python
- Explore other embeddings

Contextualized word embeddings

- Contextualized word embeddings are a representations of text where the meaning of each word can vary based on the sentence context. Unlike traditional (static) embeddings, they capture not just the word itself but also how its meaning changes in different situations.
- Used in modern NLP models (BERT, LSTM)
- “I opened a bank account” vs. “I am sitting by the river bank.”