Word Embedding Tutorial

Daniel Kababgi

March 15, 2022

Contents

List of Figures

1 Introduction to word embeddings

This tutorial's aim is to show scholars of the humanities who are interested in literary and linguistics some new useful tools for their analysis, namely the so called word embeddings. But before we start with how to use word embeddings for research in the humanities, we should get an understanding, of what word embeddings are and how they differ from other models of language representation. In short, embeddings are a representation of a word in a high dimensional vector space to catch the semantics and meaning of the said word. The basis for this whole concept was introduced in the 1950s. It consists of two ideas: the meaning of a word is given by its distribution of language use, coined by Joos (1950), Harris (1954), and Firth (1957), and the idea to represent the meaning of a word via points in a three-dimensional space, developed by Osgood et al., also 1957.[1] Distribution of language use is a fancy way of saying similar words are used in a similar grammatical context and thus their vectors point to roughly the same mathematical space.

What an embedding is, is rather abstract, so we show one of the most common examples for teaching the concept, namely the relationship between *king* and *man* and *queen* and *woman*, shown in figure 1. These relationships form a distinct parallelogram, after which this model is named.[2] This relationship is the result of the vector representation, which allows simple arithmetic operations. These operations can be utilized to complete several tasks, like extracting the most important words in an text or find all named characters in a text.

There are various models for generating embeddings, which can be categorized in two different types: static and contextualized word embeddings. Both of them have their advantages and disadvantages. Static embeddings were popularized by *Word2Vec* by Mikolov et al. in 2013.[3] This type of embedding works quite well for easy tasks and, in fact, do still have their place in our literary research toolbox. But they have one crucial flaw, which is that they only produce one embedding per word. This leads to problems, if one word has different meanings like *bank*. One meaning is *financial institution*, a second one is *river bank* and another is a type of furniture. But they are easy to compute even on a relatively old personal computer in a reasonable time frame. They are especially great for analysis of semantic change in two or more sets of corpora, so we are going to look into the popular *fasttext* model, which is a improved model of *Word2Vec*.[4]

To alleviate this problem with static embeddings a new type of model was invented,

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Third Edition draft. 2021, p.106.

[2] Dawn Chen, Joshua C. Peterson, and Thomas L. Griffiths. "Evaluating vector-space models of analogy." In: (May 2017). URL: http://arxiv.org/abs/1705.04416, p.1.

[3] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space." In: (Jan. 2013). URL: http://arxiv.org/abs/1301.3781.

[4] Anton Ehrmanntraut et al. *Type-and Token-based Word Embeddings in the Digital Humanities*. 2021.

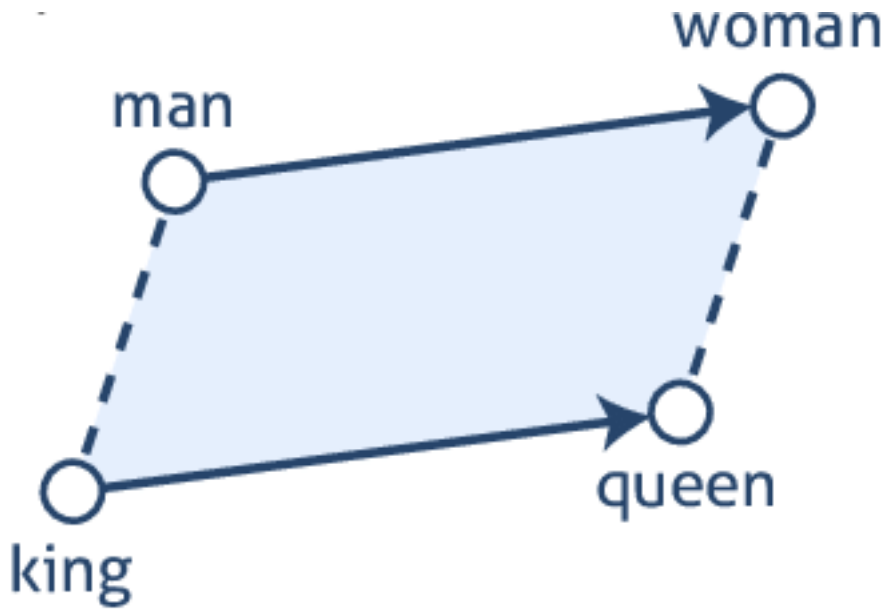

Figure 1: This figure shows a common example for teaching how word embeddings can be conceptualized. The relationship between *king* and *man* is the same as between *queen* and *woman*, shown via dotted line. The same is true for the relationship between *man* and *woman* and *king* and *queen*, which is marked with arrows. Because of this shape the name for this analogy is called parallelogram model. Figure from Chen et al., 2017

the contextualized word embeddings, of which *BERT* is undoubtedly the most popular. The main advantage is their capability of using contextual information for each embedding. This type is the state-of-the-art model for most tasks and as such it is important to get a general understanding how they work and how to use them for different downstream tasks. They are mach more powerful than static embeddings, but they are incredibly resource hungry, take a very long time to train and are best run on special graphics cards by *Nvida*. We are going to look into two different models, *Flair* by Akbik et al.[5] and of course into *BERT*[6] and it's derivatives.[7]

We will use the three different models for three little projects, which could be of interest for a project in the digital humanities. For each project we will use the same corpus, consisting of a number of Gothic literature and pulp literature. The Gothic texts were curated by Winter and Striblin[8], the pulp texts were mostly provided by *Project Gutenberg* except for the Lovecraft texts, which were found on *hplovecraft.com*.[9] The first project uses *fasttext* to investigate semantic changes in the Gothic and the pulp corpora. For the *Flair* project we will use the model perform a *named entity recognition* (= NER), which in it of itself is not to spectacular, but can be a important step for gathering data for a network analysis. Apart from that we will discuss using the right

[5]Alan Akbik, Duncan Blythe, and Roland Vollgraf. *Contextual String Embeddings for Sequence Labeling*. 2018, pp. 1638–1649. URL: https://github.com/zalandoresearch/flair.

[6]Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: (Oct. 2018). URL: http://arxiv.org/abs/1810.04805.

[7]Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A Primer in BERTology: What we know about how BERT works." In: (Feb. 2020). URL: http://arxiv.org/abs/2002.12327.

[8]**Winter2021**.

[9]**Loucks2021**.

model for the task at hand and it will introduce the popular and important *huggingface* pipeline, which will also be used for the last project. This will use *distilBERT*[10] for a binary classifier, one of the most important tasks in ML.

[10] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: (Oct. 2019). URL: http://arxiv.org/abs/1910.01108.

2 FastText

- fasttext for static embeddings - how does the algorithm work - why is it still as useful tool (Ehrmantraut2021) - experiment (change of meaning) - word *night* - word *god* - ...

3 Flair

- flair for contextualized embeddings - how does the algorithm work - look on what data the algorithm is trained and use the right model for your experiment - experiment (NER) - flair vs. bert

4 BERT

- bert, also contextualized embedding model - how does the algorithm work (briefly) - different derivates form original bert -> bertology - experiment (binary classifier) - classical machine learning with svm or logReg -> doesn't work very well - deep learning with keras -> works very well if you use GPU!

References

Akbik, Alan, Duncan Blythe, and Roland Vollgraf. *Contextual String Embeddings for Sequence Labeling*. 2018, pp. 1638–1649. URL: https://github.com/zalandoresearch/flair.

Chen, Dawn, Joshua C. Peterson, and Thomas L. Griffiths. "Evaluating vector-space models of analogy." In: (May 2017). URL: http://arxiv.org/abs/1705.04416.

Devlin, Jacob et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: (Oct. 2018). URL: http://arxiv.org/abs/1810.04805.

Ehrmanntraut, Anton et al. *Type-and Token-based Word Embeddings in the Digital Humanities*. 2021.

Jurafsky, Daniel and James H. Martin. *Speech and Language Processing*. Third Edition draft. 2021.

Mikolov, Tomas et al. "Efficient Estimation of Word Representations in Vector Space." In: (Jan. 2013). URL: http://arxiv.org/abs/1301.3781.

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky. "A Primer in BERTology: What we know about how BERT works." In: (Feb. 2020). URL: http://arxiv.org/abs/2002.12327.

Sanh, Victor et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: (Oct. 2019). URL: http://arxiv.org/abs/1910.01108.