MoMA project proposal

Polina Stadnikova

January 23, 2018

Inverted indexing approach for word embeddings

For the final project, I would like to implement the inverted indexing approach to obtain word representations. These representations can be cross-lingual and, therefore, helpful for many NLP tasks like POS-tagging, depedency parsing, word alignments, etc. I got inspired by this paper: *http://www.aclweb.org/anthology/P15-1165*, which I presented in Word embeddings for NLP and IR seminar this semester.

In a nutshell, each word is represented by a row in the co-occurrence concept-word matrix. A concept is represented by some Wikipedia articles which describe the same topic in *different languages* and are linked to the same node in the Wikipedia ontology. With the forward indexing, we would list the words per concept, but here we make use of the inverted indexing and list the concepts per word .

This approach is counted-based, that means, a word is described by a vector which stores the information about how often this word is used to describe different concepts. That is, for each language, we need one co-occurrence matrix. Such matrices allow to capture relatedness between the words in different languages: words describing the same concepts have similar vectors. For example, if the German word *Brille* and the English *glasses* both occur in the Wiki article about Harry Potter, they will have similar representations.

The paper does not describe the approach in detail, so I think it would be challenging enough just to implement the representations. They do different experiments with these embeddings, but it would not be manageable to recreate them within 25 hours. Fortunately, they also provide the Wikipedia data here: *https://sites.google.com/site/rmyeid/projects/polyglot* (which even seems to work ;)). My plan is to do the following:

1. Implement a method for constructing matrices from the given data
2. Create matrices for some languages (preferably for the languages I speak, so I can easily evaluate my method)
3. Compare vector representations and extract similar words (here I think of some kind of alignments: e.g., Brille and glasses are aligned together)
4. For the evaluation, I have two ideas: 1) just automatically check whether the aligned words really have similar vectors (not sure whether this would make sense though) 2) create gold alignments manually and compare, compute recall, precision, F-score (I hope I will have time for this).