Lab 2: Word Embeddings

1 Word embeddings

In this assignment, we are going to train a dynamic word embedding from scratch on newspaper data. The data is available here, and covers news articles between 2005 and 2012. Moreover, there is a pre-trained embedding that will be used along with the one trained on this corpus.

We are going to use the `probabilistic_word_embeddings` (PWE) package. It can be installed from pypi:

```
pip install probabilistic_word_embeddings
```

We also need `networkx` and `pandas`, and a library for plotting, eg. `seaborn`.

The documentation for the PWE module is available here. Moreover, an example of training a dynamic embedding (which might come in handy), is available here.

2 Preprocessing

In this part, your task is to load in the data and preprocess it. Read in the CSV, drop null/na rows, extract the year from the date variable, convert each article into a list of words, and use the `preprocess_partitioned` function to lowercase, remove punctuation and filter rare words from the data.

In addition to preprocessing the text, you need to define a prior graph for the dynamic model. For each word, connect all its instances for adjacent years, eg.

$$\text{dog}_{2012} \sim \text{dog}_{2011} \tag{1}$$

Use the `networkx` module for this.

Verify the result of the preprocessing. Print out some of the preprocessed (not all as there are too many!). Finally, save the preprocessed text and vocabulary as a JSON file, and the resulting prior graph as an adjacency list.

3 Training the embeddings

Load in the preprocessed data you saved.

Create an embedding using the prior graph and the vocabulary.

Train the embedding using `map_estimate`. Feel free to set model to "sgns" or "cbow", window size "ws" to anything between 2 and 10. Other reasonable hyperparameters are "epochs"=3, "batch_size"=20000. You can repeat the training procedure to get better trained results if you have time. Finally, save the embedding using e.save.

4 Analyzing the embedding

At this stage, you want to analyze the word embeddings you have trained. Oftentimes, it is useful to know which words are similar to each other. In word embeddings, this can be done with cosine similarity.

Note that you will do the same analysis on two different embeddings: the one you trained, and a pretrained one available here.

Implement the cosine similarity metric. It is defined as

$$\text{cossim}\,(a, b) = \frac{a \cdot b}{\|a\|\|b\|} \tag{2}$$

i.e. the dot product between the vectors a and b, divided by the norm of a and the norm of b. The dot product is available in as function in numpy; norm is available in numpy's linalg submodule.

Pick two words from the same year and calculate their cosine similarity. Do this for a few word pairs that you assume to be similar, as well as a few word pairs that you assume to be unrelated.

Select another pair of words. Your task is to plot the similarity of the pair of words over time on a line plot.

Sometimes, we want to know what the semantically closest words are to a target word. There is a function for this, `nearest_neighbors`. Use it to extract the 10 closest words to "bread", both in 2005 and 2012. Additionally, pick a few other words and see what their nearest neighbors are.