PA1: Distributional vectors and similarity

The goal of this task is to help you understand the main concepts behind word vectors (embeddings). You will write a Python script that takes as input raw text and generates as output two items:

- A plot showing embeddings in a two-dimensional space of all target words

To find word embeddings, we first represent each target word as a vector. This is achieved by collecting the co-occurrence counts from a corpus, then smoothing and weighting the counts to obtain an association score (PMI). The PMI values are multidimensional vectors. We then apply PCA (principal component analysis) to vectors to obtain two-dimensional vectors needed for the plot. Your task is to collect the co-occurrence counts and to calculate the PMI values. We provide code snippets for PCA and plotting (in Materials), which you need to integrate in your script in order to produce the first output.

- For each target word the most similar word

To find the most similar words, we calculate cosine similarity between all target words and store the values in a T x T matrix. We then search the matrix to find the highest value for each target word.

Processing details

Preprocessing

Before collecting the co-occurrence counts, the raw text needs to be preprocessed. Perform exactly these three steps:

1. separate words by white spaces
2. lowercase
3. remove all the punctuation

You are free to use the raw text of your choice, but it needs to be preprocessed as specified. No other changes are allowed.

Input

In addition to the raw text, you will need two more files:

T.txt: a set of words for which you will calculate cosine similarity and clustering

B.txt: a set of words which will constitute your word vector basis

We use the term **vector basis** as explained by Widdows (2005, Ch4-5).

In the materials for the task, you will find two sets of words, but it will be your task to find out which one is suitable as T and which one as B. Rename the files according to your decision, but do not make any changes inside of the files.

Finding the most suitable raw text is a small challenge intended for you to practice problem solving. You will need to think of how the selection of the raw text impacts your processing and the results.

Co-occurrence matrix

Calculate the weighted co-occurrence matrix T x B using the context window size 5 (two positions before and two after the target word).

- Use the package `numpy` to create the appropriate data structure for storing the counts.
- Use point-wise mutual information (PPMI) scores as weights.

PCA and plotting

Adapt the example provided in the notebook (in Materials) to take your co-occurrence matrix as input. The output of this step is a plot showing all your T words in a two-dimensional space.

Note especially the imported libraries, which need to be installed in order for this code to work.

Most similar words

Calculate the cosine similarity matrix T x T using the PPMI co-occurrence matrix. Then find and print the most similar word for each target word (so that this is not the target word itself). The out put of this step is a set of tuples.

Specific requirements

Make sure that the formulas used to calculate feature weights and cosine similarity are transparent in your code. The raw text and the sets B and T should be read from a file given by the user as command line arguments when running the script. Please add any comments necessary to understand the elements of your code and provide instruction how to run it.

Submission

Upload to OLAT by **13.03.2023 at 15h**:

- your Python script
- B.txt
- T.txt