PA1: Word vectors from scratch

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:

1. 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 (attached), which you need to integrate in your script in order to produce the first output.

2. 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.

Preprocessing

Before collecting the counts necessary for the calculation, the raw text needs to be preprocessed. You need to 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. If you don’t know how/where to find convenient texts, ask for tips.

Input

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

1. Define and describe a set B of words which will constitute your word vector basis. Store these words in a text file named B.txt, one word per line. Make sure that this file contains no spaces and no empty lines. Note: Make sure you acquire and understand the term vector basis is explained in the reading above. Although our set B in this task overlaps with the notion of a “context word”, it is important to have in mind that B can be defined in other ways too.
2. Define and describe the set T of words for which you will calculate similarity and clustering. This set should be meaningful for the clustering task. Store this set in a text file named T.txt, in the same format as B.txt.

Co-occurrence matrix

Calculate the co-occurrence matrix T x B using the context window size 5 (two positions before and two after the target word). Use positive point-wise mutual information (PPMI) scores as weights. Use the package numpy for storing the counts.

PCA and plotting

Adapt the example provided in the attached notebook 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.

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.

Hints about numbers (after questions in class):

- How many B-words? Not fewer than 50, but around 200 is probably more suitable
- How much raw text? Not less than 100K tokens

Upload to Moodle:

1. your Python script
2. B.txt
3. T.txt