ELEC-E5521 Speech and language processing methods

Word2vec assignment, spring 2020

In this exercise, a brief report of word2vec studies is made. You can do this whole exercise with any tool you want, Python recommended, but are required to pay special attention to commenting your code and process (you want us to understand what you did). Basically, you can use a neural network function provided by your framework, but do everything else yourself. Later, when comparing your own solution to a more advanced one, you can use, for instance, Gensim.

The first exercises describe one way that produces testable results. You are encouraged to think ways to improve your results already at exercises 1 and 2, while building the toy model. Implementing your improvements will be done in exercise 3.

Report all requested figures and values clearly. Scale the axes of all figures so that the data relevant to the given questions can be easily observed. Also include all code that you wrote in answering to the exercises.
The report will consist of your **commented** code and answers to the questions. The exercise descriptions are instructions following which you build your first code.
Since this is exercise is still developing, your feedback and questions are very valuable, you can address them at juho.leinonen@aalto.fi or during normal working hours at F406.

Exercise 1

1.1 Download Project Gutenberg's The Golden Age Cook Book, by Henrietta Latham Dwight. Split the corpus into sentences and then prepare the data by removing all punctuation marks and empty lines and lower casing all characters.

1.2 Remove all stop words and words that are only two letters or less from your vocabulary, after that keep only the 1000 most frequent words in your vocabulary. Remove all words not in your vocabulary from your source corpus. Clean also empty lines and extra white spaces.

Question 1: Provide a list of the stop words you used as an attachment when returning the exercise.

1.3 Create a dataset where words are replaced by their indexes in your vocabulary.

1.4 Go over your indexed list and create training pairs of the words (in this case, pairs of positions of the word in your vocabulary) for the neural network with a static context window of two. Since this will be a skip-gram model, you are trying to predict the context from your input word so your method will be:

this is an example sentence for you.	(x_i, y_i)
this is an example sentence for you.	(this, is), (this, an)

this **is** an example sentence for you.	(is, this), (is, an), (is, example)
this is **an** example sentence for you.	(an, this), (an, is), (an, example), (an, sentence)
this is an **example** sentence for you.	(example, is), (example, an), (example, sentence), (example, for)

Question 2: You have created your training samples per sentence. Would you get different results forming them per document? Per recipe? Explain.

Exercise 2

2.1 With your toolkit, create a two (inputs, hidden layer, output layer) layered neural network with a hidden layer of 25 neurons and linear transfer function, output layer should have softmax transfer function and loss function should be cross entropy. For optimizer use stochastic gradient descent.

2.2 Create a loop where you train your network with a 1000 samples at a time from your samples list by creating a matrix where you have your one-hot vectors (so your vector should have as many rows as words in your vocabulary and the value in the training pair tells you where your only true value is on the row). Make sure you matrix rows and columns are correct for your architecture. If you are using PyTorch check DataLoader. Using it will affect these instructions a bit.

2.3 Put the loop you created inside another loop where you decide on the number of epochs. Three should be fine for start. Shuffle your samples at the start of every loop.

2.4 Do a sanity check at the end to see that different inputs indeed produce different outputs. So that with different x vectors the network predicts different words, so y vector should have positive values at different indexes most of the time.

Exercise 3

3.1 The weights in your network's hidden layer matching each word are the word vectors. Since the neural network was trained to learn contexts for different words, it follows that similar words should have a similar context. As such, synonyms should have similar word vectors. It should also be possible to do simple vector arithmetics and have results that make intuitive sense for humans:

king – man + woman = queen

poland + capital = Warsaw

Question 3. Create a test set of five words to find the closest word by comparing their word vectors. Create a total of five analogies and word addition tests. Create five intruder tests (e.g. banana, orange, strawberry, fork) Make sure the words you are using and words you expect to be the correct answers are in your vocabulary.
Which distance metric did you use? Why?

Exercise 4

4.1 Try to improve your model by using different parameters, **a lot more data**, different methods to prepare your word vectors etc.

Question 4. Explain the reasoning behind your choices. Do at least three modifications and compare your clustering results using silhouette criterion. Plot your best word vectors with t-SNE (add labels for some of the dots).

Exercise 5

5.1 Use a word2vec toolkit to build models with the same corpus and parameters.

Question 5. How did your results differ? Explain possible reasons why. Can you improve results even more? Try. Give your best result on silhouette criterion and t-SNE.

5.2 Instead of using the skip-gram model, train a continuous bag-of-words model with your pipeline **and** using the previous toolkit.

Question 6. Explain the differences of these two models. How are rare words handled differently in them? Would the size of the corpus affect which model you choose? Why? Also plot your best results with t-SNE and silhouette criterion.

5.3 *Optional exercise for improving your grade*: Implement a skip-gram model with your pipeline using a different natural language (e.g., German). **Question**: Did you need to change your implementation?

Exercise 6

6.1 Give freeform feedback.