

ELEC-E5521 Speech and language processing methods

Word2vec assignment, spring 2018

In this exercise, a brief report of word2vec studies is made. In exercise 1, one-hot vectors are formed. You can do this with other tools than MATLAB, but then you are required to pay special attention to commenting your code and process (you want us to understand what you did). Some of the instructions are due to how strings work in MATLAB, or how MATLAB's neural network toolbox works. Use common sense on whether you need to implement it exactly like that with e.g., Python. 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 MATLAB and other codes 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 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. 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 list of your corpus where every row has one word and the order is the same as in your corpus e.g., "Once upon a time" would be ["once" ; "upon" ; "a" ; "time"]. Use MATLAB's function unique to create a list 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 <u>is</u> <u>an</u> example sentence for you.	(this, is), (this, an)
<u>this</u> is <u>an</u> <u>example</u> sentence for you.	(is, this), (is, an), (is, example)
<u>this</u> <u>is</u> an <u>example</u> <u>sentence</u> for you.	(an, this), (an, is), (an, example), (an, sentence)
this <u>is</u> <u>an</u> example <u>sentence</u> <u>for</u> you.	(example, is), (example, an), (example, sentence), (example, for)

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

Exercise 2

- 2.1 With MATLAB's neural network toolkit, create a two (inputs, hidden layer, output layer, outputs) layered neural network for pattern recognition 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. You should remove the processFcn removeconstantrows from preprocessing of the input. Change the training function to scaled conjugate gradient backpropagation (with other tools SGD is a good one) and use one epoch as the networks training parameter.
- 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, the nntraintool should show correct number of rows as input (your vocabulary size). When you use train function the weights of the network should start from their previous value, so your performance should get better at every iteration.
- 2.3 Put the loop you created inside another loop where you decide on the number of epochs. Three should be fine for start. You can also shuffle your samples at the start of every loop.
HINT: Trust MATLAB when it suggest you to preallocate for speed. Everything so far if evaluated should take at most a couple of minutes.
- 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. 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, 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 results with evalclusters using silhouette criterion (you can choose the clustering algorithm yourself). Plot your best word vectors with tsne and textscatter.

Exercise 5

- 5.1** Use MATLAB's own function trainWordEmbedding to train word2vec 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. You can also use word2vec and vec2word. Give your best result on evalclusters and tsne+textscatter.

- 5.2** Instead of using the skip-gram model that is MATLAB's default, train a continuous bag-of-words model.

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 tsne+scatterplot and use evalclusters.

- 5.3** *Optional exercise for improving your grade:* Implement a skip-gram model without using MATLAB's trainWordEmbedding function with a different natural language (e.g., German). **Question:** Did you need to change your implementation? How about with MATLAB's trainWordEmbedding?

Exercise 6

6.1 Give freeform feedback.

Useful MATLAB functions:

string, splitlines, erasePunctuation, split, unique, histcounts, tokenizedDocument, removeWords, patternnet, randperm, train, evalclusters, tsne, textscatter