IN4080 2020, Mandatory assignment 2, part B

Your answer should be delivered in devilry.ifi.uio.no no later than Friday, 9 October at 23:59

About the assignment

Mandatory assignment 2 has two parts, A and B. You should answer both parts, all questions. This is part B. For general requirements, including point system and delivery formats, see part 1 . For all exercises, deliver answers to all questions and code where you apply code.

Goals of part B

In this part we will you use the gensim package to familiarize ourselves with word embeddings and word2vec. You will get more experience with

- vector representations of words
- cosine distance and similarity
- semantic relationships including analogies
- training embeddings
- evaluating embeddings
- application of embeddings in text classification

Getting started with gensim

You should have gensim available in the in4080 environment on your own machine if you have followed the installation instructions
https://www.uio.no/studier/emner/matnat/ifi/IN4080/h20/Programming%20environment
It should work at the linux machines in the IFI terminal rooms. For remote login, gensim will not work with login.ifi.uio.no. It should work with remote login via VDI
https://www.uio.no/tjenester/it/maskin/vdi/. Be aware that you might get problems with disk space on the IFI machines if you download large models.

You find information on gensim at the webpage https://radimrehurek.com/gensim/index.html. To get started, consider the *Word2Vec Model* tutorial on the documentation page,
https://radimrehurek.com/gensim/auto_examples/index.html . Follow the first steps up to Training Your Own Model. In particular, download and install the 'word2vec-google-news-300'. (If you get problems with disk space, choose another, smaller model, e.g. ‘glove-wiki-gigaword-100’.) By the way, take a look at the page https://github.com/RaRe-Technologies/gensim-data and try to grasp how large amounts of texts that have been used for training these models.

Exercise 1 Basics (8 points)

a) How many different words are there in the model? With so many words, how come that the ‘cameroon’ example fails?

b) Implement a function for calculating the norm (the length) of an (embedding) vector, and a function for calculating the cosine between two vectors.

c) Calculate the cosine between the vectors for ‘king’ and ‘queen’ and check you get the same as by

```
<model>.similarity(‘king’, ‘queen’)
```

Exercise 2 Built in functions (5 points)

There are several built-in functions that let you inspect semantic properties of the embeddings. The *most_similar* lets you find the nearest neighbor to one or more words.

```
      print(wv.most_similar('car', topn=5))
      print(wv.most_similar(positive=['car', 'minivan'], topn=5))
```

a) It is also the tool for testing analogies, e.g.
"Norway is to Oslo as Sweden is to …"
as

```
      print(wv.most_similar(positive=['Oslo', 'Sweden'],
                            negative = ['Norway'], topn=5))
```

Try a few analogy tests like

" king is to man as queen is to …"
" king is to queen as man is to …"
"cat is to kitten as dog is to …"

Add four more examples of your choice. Report the results of the tests. Are the results as expected?

b) To understand the method a little better, we can try to follow the recipe more directly. Try

```
      a = wv['king'] + wv['woman'] - wv['man']
```

and calculate the cosine between *a* and the vectors for *queen, woman, man, king*. You may also calculate the

```
      wv.similar_by_vector(a)
```

What does this show regarding how the *most_similar* works?

c) Play around with *wv.doesnt_match*, e.g.

```
      print(wv.doesnt_match(['Norway', 'Denmark', 'Finland',
                            'Sweden', 'Spain', 'Stockholm']))
```

Make at least two more examples where the result match human evaluation and two examples where they do not match. Explain!

Exercise 3 Training a toy model (5 points)

a) Train a word2vec model on the Brown corpus. Follow the recipe from the tutorial, the section *Training Your Own Model*. You may import the corpus from NLTK *by brown.sents().* Beware that this is a toy example. The Brown corpus is too small for training good models. How many times larger is the Google news corpus compared to the Brown corpus?

b) We will compare the Brown model to the *'word2vec-google-news-300'*. Try to find the 10 nearest words first to *car* and then to *queen* in the two models. What do the examples reveal about the two training corpora?

c) Inspect the trained Brown model on some of the examples from exercise 2. Does it yield the same results on the analogy tests as the model in exercise 2?

Exercise 4 Evaluation (5 points)

Gensim comes with several methods for evaluation, and also standard datasets for the tests. Testsets could be found by the tha *datapath* command, e.g.

```
        path=datapath('questions-words.txt')
```

One test you may use is to see how well the model perform on the Google analogy test datset. This can be run by

```
        <model>.evaluate_word_analogies(path)
```

Report the key numbers, and try to understand what they mean.

To compare 'word2vec-google-news-300' to the Brown embeddings is not too interesting. The difference between them is too large. A test like this becomes more interesting if you try to compare 'word2vec-google-news-300' to e.g. 'glove-wiki-gigaword-300' or you want to inspect the effect of the length of the embeddings by comparing 'glove-wiki-gigaword-300' and 'glove-wiki-gigaword-100'.

Exercise 5 Application (12 points)

We will try a simple example of applying word embeddings to an NLP task. We consider text classification. We will use the same movie dataset from NLTK as we used in Mandatory assignment 1B, with the same split as we used there. Thereby, we may compare the results with the results from Mandatory 1.

We will consider a document as a bag of words. The word order and sentence structure will be ignored. Each word can be represented by its embedding. But how should a document be represented? The easiest is to use the "semantic fingerprint", which means representing the document by the average vector of its words.

a) Train and test a logistic regression classifier as described. Tune the C parameter (regularization, cf. Mandatory 2.A). Report the results from the tuning in a table. How does this classifier perform compared to your results from Mandatory assignment 1?

b) In this task, one could either use the document vectors directly, or normalize the length of each document vector to unit length before classifying. Try both options and compare the results.

c) In general, embeddings are used together with neural networks. Scikit-learn provides a simple multi-layered neural network classifier.
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
Rerun the classification experiment with this classifier. Try various activation functions and report the results for the various activation functions in a table.

Warning! When using neural networks and embeddings for word classification, you would use more elaborate models than the simple bag-of-words model, e.g. convolutional networks or recurrent networks which take word order into consideration.

END of Mandatory 2