Deep Learning for NLP 2019
Home Exercise 07

Due on Tuesday, 04.06. at 13:00

Submission Guidelines for all Home Exercises

- When submitting multiple files, submit one **zip-archive**.
- Submit python code as plain python scripts (**.py**). **Must** be runnable in the given Docker container.
- Submit answers to non-code assignments in **one PDF** file. Scans are permitted, if readable.
- Guidelines specific to neural network code:
  - Please submit your training/testing results (a copy of your console output is fine). Reasoning: Your network might train much slower on the tutor's system than on yours.
  - If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

Problem 1 Mandatory Paper (1P)

Read this week's mandatory paper[1]. Explain in up to two sentences each:

a) Why do randomly initialized, untrained sentence embeddings produce usable results?

b) How does the Bag of random embedding projections (BOREP) approach create a sentence representation?

Problem 2 Training word2vec Embeddings (2P)

In this task, you will train word2vec skip-gram embeddings yourself.

Problem 2.1 Setup and Parameters (1P)

Inside your Docker container, run the following commands to download and compile `word2vec`:

```
git clone https://github.com/shenoybr/word2vec
cd word2vec
make
```

Then, run `word2vec` without specifying any arguments. This will show you an example of how to run word2vec with given parameters:

```
./word2vec -train data.txt -output vec.txt -size 200 -window 5 -sample 1e-4 -negative 5
-hs 0 -binary 0 -cbow 1 -iter 3
```

What are the effects of the four parameters `size`, `window`, `negative`, and `cbow`?

[1] https://openreview.net/pdf?id=BkgPajAcY7

Problem 2.2 Training (1P)

You will now train word embeddings on 100k sentences taken from the English Wikipedia. You can find the sentences in the `en_wiki_dump_100k.txt` file within `hex07_data.zip`.
Train the embeddings with all hyperparameters at their default, except for the following:

- size = 300
- window = 5
- cbow = 0
- iter = 10

Then, use the `distance` tool in the word2vec directory and compute:

- the 5 closest words to the word *man*
- the 5 closest words to the word *woman*

What do you notice in the output?
Hint: `distance` requires a model file in binary format. Therefore, set the binary flag to 1.

Problem 3 Training sent2vec Embeddings (2P)

In this task, you will train sent2vec embeddings[2] yourself.

Problem 3.1 Setup and Parameters (1P)

Inside your Docker container, run the following commands to download and compile `sent2vec`:

```
git clone https://github.com/epfml/sent2vec
cd sent2vec
make
```

Run the tool without parameters and take a look at its arguments.
What are the effects of the three parameters `lr`, `dim`, and `ws`?

Problem 3.2 Training (1P)

Now, train embeddings on the Wikipedia dump from Problem 2.2 using the following hyperparameters, leaving all others at their defaults:

- dim = 300
- wordNgrams = 2

Which command is necessary to convert a sentence into a vector representation, given a previously trained model? Assume that your sentences are in a file “mysentences.dat” in which each line consists of a tokenized sentence.

Problem 4 Sentence Embeddings for Movie Reviews (5P)

We return once again to the movie reviews dataset labeled with sentiment polarity. This time, the task is to compare two sentence representation techniques: sent2vec sentence embeddings and word2vec embeddings averaged on the word level.
`hex07_data.zip` contains the training, development and test reviews and their labels in separate text files.

[2] sent2vec embeddings: `https://arxiv.org/abs/1703.02507`

Problem 4.1 Creating Embeddings for Reviews (2P)

a) Obtain a Sent2Vec embedding for each movie review. To do so, apply the command from Problem 3.2 with the model trained in Problem 3.2 on each of the `rt-polarity.*.reviews.txt` files and route the console output to a file.

b) Obtain a word2vec embedding for each movie review. To do so, write a python script, which:
   - reads the review files
   - loads the word2vec embeddings trained in Problem 2.2
   - computes an embedding for a full review by averaging the word embedding of each word in the review

   Hint: Decide for a suitable way to deal with out-of-vocabulary words.

Demonstrate that your code works by printing the sent2vec and the averaged word2vec embedding vector of the first review from the training data set.

Problem 4.2 Embedding Comparison (3P)

Implement a perceptron which takes an embedding of a review as an input, applies softmax and uses cross-entropy as the loss function. You may adapt your implementation/the solution from home exercise 03 or write a new one with TensorFlow or Keras. Remember that cross-entropy loss requires class labels in a one-hot vector format.

Train the perceptron with sent2vec embeddings on the training set (hyperparameters are up to you) and report its accuracy on the test set. Repeat the procedure with your averaged word2vec embeddings.
Explain two sentences how your results compare to those of home exercise 03 and what the cause for the improvement / decline in performance is.