Language Technology I, Winter 2015-2016

Exercise 10: Recurrent Neural Network Language Models

You can earn up to 10 points on this exercise.
You may work as a group of up to 3 people, but please submit your own version.
You will be using python based library keras for this assignment.
Please submit all the code required to run the neural networks on our machines.
Please state any assumptions you make
Any submission that we cannot run on our computers without installing things, must be presented in the class

Please ***email*** *your solution to* `mittul.singh@lsv.uni-saarland.de` by **10:15 am, January 27, 2016**. While submitting your assignment by email, please name the file as Ex10_<*your name*>.pdf/zip/tar.gz .

Task

In this task you are required to build parts of a recurrent neural network (RNN) language model (LM) provided at `https://bitbucket.org/mittulsingh/10-rnn-assignment`. The code is written in python, using keras library to implement a neural network based LM. The training algorithm uses a recent gradient descent method called ADAM. The training method has also been changed to chunk and batch process the data to allow it to handle larger amounts of data than the previous assignment. Hope you have installed keras and h5py already! The sub tasks to building and using this neural network are as follows:

1. Process the data used in Assignment 4's Task 2. Carry out the first three steps as described in Assignment 4's Task 2 for the first **100000** lines of text only. After completing these steps you should have 2385374 words in en.txt [1]. In case you do not, you might have an encoding issue. Try setting `$LC_CTYPE` to `en_US.UTF-8`[2] and re-doing the steps. (0 points)

2. Construct a vocabulary of words. Use the data from the previous step to construct a vocabulary of 10000 most frequent words (say, en.voc), with each unique word in a separate line. This format is essential for use in the RNN code provided. (Hint: use get_word_list_count.sh tool available at `https://bitbucket.org/mittulsingh/10-rnn-assignment`) (0 points)

3. Split the resulting corpus into training/development/test sets, with the ratio 18:1:1 respectively, using the script `http://jon.dehdari.org/corpus_tools/generate_splits.pl` . Use the `--help` argument for usage info. You should have 2330319 words in the training set, 129702 words in the development set and 125353 words in the test set. (0 points)

4. Add sentence markers. Put a "<s>" (sentence begin marker) at the beginning of every sentence and a "</s>" (sentence end marker) at the end of every sentence in the training/development/test sets. Convert these files into files with a word on each line. Lastly,

[1] Use the command `wc` on bash to count words

[2] Learn to set environment variables this `http://www.cyberciti.biz/faq/set-environment-variable-linux/`

add the sentence markers to the vocabulary. These steps is to conform with the input data format of the provided code. (0 points)

5. Handle out of vocabulary words (OOVs). You might have noticed above that the number of unique words are much larger than the number of words in vocabulary (10002 at this point). For all the unaccounted words we need to map them to an unknown symbol. Add an `<unk>` symbol to the vocabulary and rest is taken care of by the code. You can specify this symbol in the toolkit using `-unk` flag. (0.5 points)

6. Build a RNNLM. To be able to build a RNNLM, you will need to write a few lines of code in the build function of the file `rnnlm.py`. This part will be evaluated on the code you submitted. (Hint: You might want to use an keras.layers.embeddings.Embedding and keras.layers.recurrent.SimpleRNN in conjunction to build the RNN) (2 points)

7. Plot the perplexity of the model obtained from the previous part on dev set and test set against the 10, 50, 100, 500, 1000, 5000 and 10003 units as the size of the Embedding layer. For a network of 200 hidden nodes, run the training for 2 epochs, with a batch size of 2000 and a chunk size of 10000. A good choice of learning rate and epsilon are 0.001 and 1e-08 respectively. These values also happen to be the defaults of your algorithm. Discuss the plot and note any observations you make. (2 points)

8. Build an LSTM based LM and a GRU based LM. This part will be evaluated on the code you submitted. (2 points)

9. Plot the training times of RNNLM, LSTM and GRU based LMs for 10, 50, 100, 500, 1000 and 2000 as batch sizes used during the training. For rest of the parameters, use the optimal value of Embedding layer and the values mentioned in sub-part 4 to initialize your tools. Compare and discuss the results. (1.5 points)

10. Plot the perplexity of RNNLM, LSTM based LM and GRU based LM on dev set and test set against the variation of the size of the hidden layer. Initialize the tool with the optimal Embedding layer size and the values mentioned in sub-part 4. Compare and discuss the results. (2 points)