TEK5040 Compulsory Assignment, Number 3

Narada Warakagoda

October 16, 2018

NOTE: This assignment has been made optional considering the workload of the final project. Therefore, no submission is required. But you are strongly encouraged to perform the tasks in this assignment

1 Introduction

This assignment deals with text processing with neural networks. It has two main parts. In the first part, you are going to look into word embedding using the Softmax loss function and the Noise Contrastive Estimation (NCE) method. The second part deals with word prediction using Recurrent Neural Networks (RNN). Word vectors generated in the first part will be used in the second part.

Two Python/Tensorflow scripts have been provided: `word2vec.py` and `wordpred.py`. When you run `word2vec.py` it will automatically download a data file named `text8.zip` from the internet, if your computer is online. Alternatively you can copy the file to your computer from the TEK5040 assignments web page. Note that all files should be in the same folder.

`word2vec.py` trains a word embedding system using both the Softmax and NCE. It also creates a small data set that will be used by the script `wordpred.py`. This new data set is stored in three *pickle* files: `lm_corpus.pkl` `lm_dict.pkl` and `lm_embeddings.pkl`. `lm_corpus.pkl` contains a set of words stored in a python list, `lm_dict.pkl` contains a python dictionary which maps each word into an index and `lm_embeddings.pkl` contains a 2D python list where each row gives the embedding vector of size 300 corresponding to the word of that row index.

`wordpred.py` trains a word prediction system based on RNNs. More specifically it uses the current N words taken from the training set to predict the next word. Default N is 4 and it is called `time_steps` in the script (see line 80). There are two ways to represent the words: just use the word index or word vectors computed and stored in `lm_embeddings.pkl`. Note that dimensions of the word vectors are reduced to 5 using Principle Component Analysis (PCA) in order to reduce the computational burden.

These scripts can take few hours (typically 3-4 hours) to run on a CPU-only computer.

2 Task

1. Run the script `word2vec.py`. Examine the program output. What is the time taken per iteration for the Softmax method and the NCE method respectively? Is this what you expect and why?

2. By examining the program output, give an example of a word which has increasingly closer word vectors corresponding to semantically similar words as the training goes on with NCE.

3. Identify the line which implements the Softmax operation. In the Softmax operation, labels in one-hot format are used. If the labels were not converted to one-hot format, which Tensorflow function could be used to implement the Softmax operation?

4. Consider `word2vec.py`. Insert a print command in line 90 to print the first 32 words of the data set. Further, insert few lines from number 212-216 to print the input and target batch of words. Run the script and examine the output. Is this skip-gram or CBOW?

5. Explain how the Tensorflow function `tf.nn.embedding_lookup` works.

6. Explain how the Tensorflow function `tf.nn.nce_loss` works.

7. Run the script `wordpred.py` as it is. This version makes use of word indexes (i.e. scalar values) for representing words. Study how the loss evolves by examining the program output. Change `input_fea_dim` (line 81) to `reduced_embedding_dim` (line 61). Run the script again. How the loss function evolves this time compared to the previous run? Give a possible explanation.

8. In the provided version of `wordpred.py`, a single LSTM layer is used. We would like to use two LSTM layers. Change line 87 `rnn_cell_type='single'` to `rnn_cell_type='multi'` and replace `pass` statement in line 118 with an implementation of a multiple LSTM layer. Run the new version and observe the evolution of the loss function in training and comment!

9. In the provided version of `wordpred.py` we use a static RNN (line 122). What is the difference between static and dynamic RNNs? Which Tensorflow function can be used to implement a dynamic RNN?

10. In the provided version of `wordpred.py`, we predict the next word using the current N words (line 126). But we can predict all the words $(2, 3, \cdots, N+1)$ from the current words $(1, 2, \cdots, N)$. Modify the script to handle that situation.

3 Submission

Submit your answers to the questions above and modified scripts `word2vec.py` and `wordpred.py`.