**Homework 5**

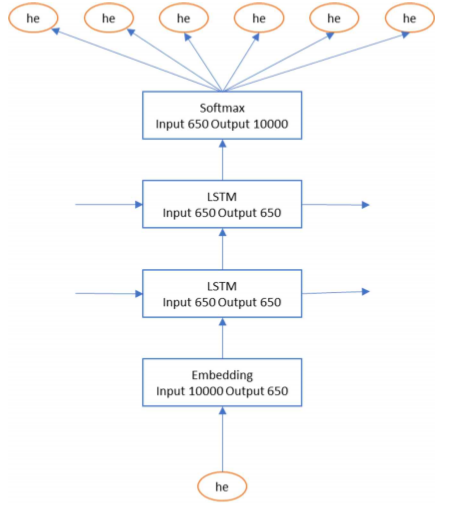
**0. Homework Information and Submission Format**

|  |
| --- |
| 2015-20941\_심규홍\_hw5.zip  report.docx  rnn.py (rnn.m)  embedding.py  generate.py |

This is a programming homework. There are three tasks. Since all the necessary information is provided, a correct answer exists. You need to submit a report and the codes, in a single zipped file.

Trained parameters are provided from Penn Tree Bank data. These parameters are trained using softmax. The following model has been used.

*Recurrent Neural Network Regularization, Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals, ICLR 2015. ( Medium LSTM model)*



**1. Word Embedding**

Using the given word embedding matrix, implement the following two tasks.

1) Find the nearest word

Using cosine distance for each word embedding vector for the following words, find the nearest word.

|  |
| --- |
| the, discount, crazy, birthday, just |

2) Find the nearest word to the answers in the previous question

Using cosine distance for each word embedding vector, find the nearest word to the word embedding vector calculation. Calculate for the following pairs.

|  |
| --- |
| king – male + female = X  breakfast – morning + evening = Y |

**2. LSTM Implementation**

Using the given parameters, implement a LSTM and calculate the perplexity for the Penn Tree Bank test set. The parsed Penn Tree Bank is provided.

LSTM is calculated as below. Do not use peephole connection or large forget bias.

* + Forget Gate
  + Input Gate
  + Output Gate
  + Cell Candidate
  + Updated Cell
  + Output

Where

Input:

Previous hidden:

Current hidden:

Previous cell:

Current cell:

For the perplexity calculation, you need to create a hidden layer and a cell that takes the test data as input, and use them to implement the next step. Initialize with 0 and do not reset the state.

**3. LM Usage**

Using the given LM, implement the following tasks.

1. Calculate the log likelihood for the following sentences.

|  |
| --- |
| i have a cat in my home.  i have a cat in my house.  i have a cat in me house. |

The log likelihood is calculated as the log (natural log) of the product of all probabilities of generating the next word for the given sentence.

P(i have a cat) = p(i) p(have | i) p(a | i have) p(cat | i have a)

From the second term, it is equivalent to the word generated from each step in LSTM. The first unigram probabilities are the following:

|  |  |  |
| --- | --- | --- |
| P(i) | P(some) | P(where) |
| 0.00135759 | 0.00179434 | 0.000415237 |

1. Using the LM as a generative model, generate a sentence that start with the following words. When choosing the following word, use a greedy search for the word with the highest probability.

|  |
| --- |
| some …  where … |

In order to use the LSTM as a generative model, take the first given word its probability as input. Greedy search can be implemented by iterating the process of taking the word with the highest probability as the input of the next step. In order to make a full sentence, iterate until you reach <EOS>.

1. Using the LM as a generative model, generate a sentence that starts with the words given in the previous question. This time, generate by using beam search (beam width = 2), keeping the two words with the highest probability. What are the final sentences, and their log likelihoods?

In order to implement beam search in LSTM, it needs to remember the top 2 candidates with the highest probabilities and calculate the next steps for each of the two words as the inputs. This means that there are 20,000 possibilities in the next step, and choose the top 2 candidates from there with the highest log likelihood. For easy implementation, if one of the paths reach <eos>, the other path goes back to the greedy search.

**4. Reading**

Summarise the quoted paper above (*Recurrent Neural Network Regularization*) in 3-5 sentences.

**DATA DESCRIPTION**

|  |
| --- |
| **Penn Tree Bank data description**  - ptb\_test\_index.txt: Penn Tree Bank test set is put in order. All words are converted into index numbers. There are 10,000 unique words for the total 82,430 words.  - ptb\_word\_to\_id.txt: Penn Tree Bank word and index pairs |

|  |
| --- |
| **Parameter** (for matlab, parameters.mat) **data description**   * embedding.npy (10000, 650) is used to convert the 10,000 unique words into vectors with 650 dimensions. You only need this for the first question. * softmax\_w.npy (650, 10000) is the weight matrix for softmax calculation. It receives the 650 dimension vectors with the bias, and converts the 10,000 unique words into logit * softmax\_b.npy (10000) is the bias vector used for softmax calculation * lstm\_0\_w.npy (1300, 2600) is the weight matrix used for the first (bottom) layer of LSTM. It is fused and W and U are combined. 1300 dimension is (and 2600 dimension is (input gate, cell candidate, forget gate, output gate). * lstm\_0\_b.npy (2600) is the bias vector used for the first (bottom) layer of LSTM * lstm\_1\_w.npy (1300, 2600) is the weight matrix for the second layer of LSTM * lstm\_0\_b.npy (2600) is the bias vector used for the second layer of LSTM |