

CS7.501: Advanced NLP | Assignment 1

Instructor: Manish Shrivastava

Deadline: September 2, 2022 | 23:59

General Instructions

1. You should implement the assignment in Python, using only Pytorch library as the Neural Network Framework.
2. Ensure that the submitted assignment is your original work. Please do not copy any part from any source, including your friends, seniors, and/or the internet. If any such attempt is caught, serious actions, including an F grade in the course, are possible.
3. A single .zip file needs to be uploaded to the Courses Portal.
4. Your grade will depend on the correctness of answers and output. Due consideration will also be given to the clarity and details of your answers and the legibility and structure of your code.
5. Please start early to meet the deadline. Late submissions won't be evaluated.

1 Neural Network Language Model

1.1 About

Language modeling (LM) is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence. Thus, it may be defined as a probability distribution over the vocabulary words. In traditional N-gram Language Models, the probabilities of words occurring after a given context are defined as:

$$\begin{aligned} P(w_1, \dots, w_m) &= \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=2}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) P(w_1, \dots, w_m) \\ &= \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=2}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \end{aligned}$$

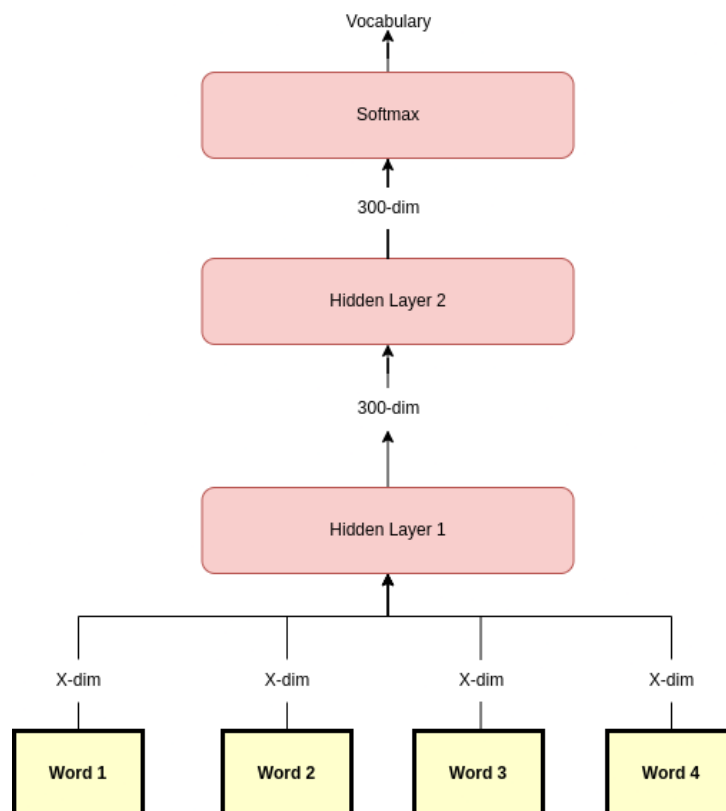
This part involves creating a Neural Language Model using standard Deep Learning Frameworks with a 4-gram context.

1.2 Model Architecture

The architecture of the Language Model is described below:

1. The input to the Neural Network would be the pre-trained embeddings of the previous 4-words concatenated together(4-gram embedding).
2. These embedding would pass into a Hidden Layer which would output a 300-dimension vector.
3. The output of the first Hidden Layer would go into another Hidden Layer, which would output a 300-dimension vector.
4. This vector would be passed to a Softmax Layer, which would output the probabilities of entire vocabulary occurring after the given 4-words.

5. The vocabulary should consist of all the words that occur any number of times in the training data. Unknown words can be handled with an <UNK> token.
6. You are allowed to use **only Pytorch** as the Deep Learning Framework for this assignment.
7. You can use any pre-trained embeddings as input to the Neural Network - word2vec, GloVe, fasttext, etc.



1.3 Dataset

1. You need to create Neural Language Model described above trained on Brown Corpus.
2. You are expected to clean the data to remove any special characters and use it for training the Neural LM. You can use libraries for tokenization.
3. You can download the corpora from this [link](#). Sample 10,000 sentences as validation set and 20,000 sentences as test set.

1.4 Marks Distribution

1. Implement the Language Model and report the Perplexity Scores. [40 marks]
2. **Bonus** Plot graphs showing the variation of average train/test perplexities with varying hyperparameters like Dropout rate, changing the dimensions of the layers, changing the Optimizer, etc. Report the most optimal Hyperparameters found. [10 marks]

2 RNN-based Language Model

2.1 About

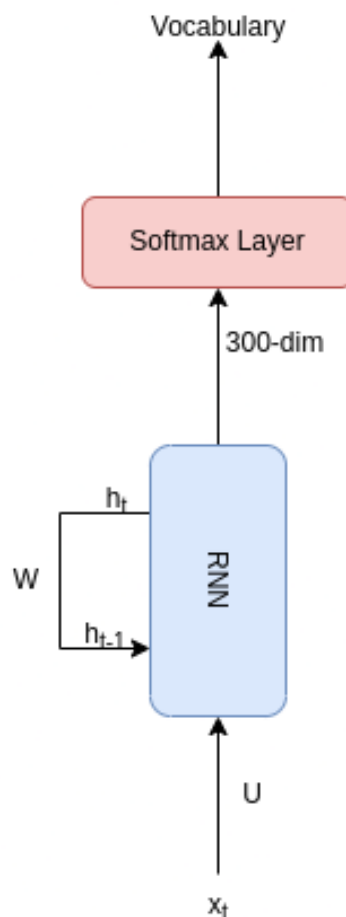
To solve some problems arising from using a much smaller(4-gram here) context for next word, Recurrent Neural Networks(RNN) were used to improve the architecture. In this part, you will create such a Neural

Language Model using RNNs, while implementing RNN functions from scratch without using the standard RNN class.

2.2 Model Architecture

The architecture of the Language Model is described below:

1. The Neural Network would consist of repeating layers in an RNN-structure.
2. The input to each layer would consist of the hidden-layer output of the previous Neuron(h_{t-1}), along with the embedding of the current word(x_t).
3. Each layer would consist of two outputs:
 - The hidden layer output h_t
 - A 300-dimension vector
4. The hidden layer output is computed as per the formula $h_t = \tanh(Wh_{t-1} + Ux_t)$
5. The 300-dimension output would be put into a Softmax-layer, which would provide the probability distribution over the entire vocabulary.
6. **Note that you CANNOT use the RNN class in Pytorch. You must implement RNN from scratch using the Hidden layer function above.**



2.3 Dataset

You are supposed to use the Dataset from the previous task for this task as well, with the same training and validation data sizes.

2.4 Marks Distribution

1. Implement the Language Model and report the Perplexity Scores. [60 marks]
2. **Bonus** Implement an LSTM/GRU architecture **without using the LSTM and GRU classes** to implement the Language Model. [15 marks]

3 Analysis

1. Analyse which of the two models gives better perplexities on the train and the test/validation sets.
2. Compare the behaviour of the two LMs, and elaborate your findings using graphs and visualizations.

4 Submission format

Zip the following into one file and submit it on the Moodle course portal:

1. Source code (Should include `.py` files only.)
2. Language Model and the generated perplexity files.
3. For each LM submit the text file with perplexity scores in the following format
 - Format : Sentence TAB perplexity-score, at the end , average score
 - Naming must be: roll number-LM1-train-perplexity.txt, roll number-LM1-test-perplexity.txt, etc
4. Readme file (should contain instructions on how to execute the code, restore the pre-trained model, and any other information necessary for clarity)

Name the zipped file as <roll number>_<assignment1>.zip, e.g.: 2021xxxxxx_assignment1.zip.

Note: If the pre-trained models cross the file size limits, upload them to a OneDrive/GDrive and share the read-only accessible links in the readme file.

5 FAQs

1. Do I need to use lemmatization before feeding the samples to the model?
No, for the sake of this assignment, you may not perform lemmatization.
2. I do not have enough compute. Can I reduce the working corpus size?
Yes, you may reduce the vocabulary size. But make sure you train the model on enough no. of sentences, which should be a minimum of 30,000 sentences.
3. Can I submit my code in Jupyter Notebooks?
No, the final submission should be a Python script. You may work using a Jupyter Notebook, but make sure to convert it to `.py` file before submitting.
4. You can use libraries like Scikit-learn and Gensim to read vector files and use the pre-trained embeddings for the LM task.

6 Resources

6.1 Papers

1. [Recurrent neural network based language model](#)
2. [A Neural Probabilistic Language Model](#)

6.2 Explanatory supplements

1. [Stanford lecture notes - LM & RNN](#)
2. [A simple, intuitive explanation of RNNs, LSTMs and GRUs](#)
3. [A Survey on Neural Network Language Models](#)
4. [GloVe: Global Vectors for Word Representation](#)