CS-461, Winter 2025
Homework #1: Bengio Language Model
Due Date: Wednesday, February 5th @ 11:59PM
Total Points: 10.0 plus 1.0 Bonus Available

In this assignment, you will work with your group to implement a feed-forward neural network language model (*e.g.,* the Bengio model), train it on the Winitext-2 corpus and answer questions about the model. You will also gain valuable experience working with various constructs common in language modeling and NLP tasks.

You may discuss the homework with other groups but do not take any written record from the discussions. Each group must submit their own work. Do not submit another group's work as your own, and do not allow another group to submit your work as their own. Also, do not copy any source code from the Web or use any AI-generated content.

Step #1 (5.0 points): PyTorch Implementation of the Bengio Model

Implement a feed-forward neural network language model similar to the Bengio model covered in class using PyTorch. Train and test the model on the Wikitext-2 corpus. I provide starter code, which parses command line arguments, reads the corpus and suggests function signatures for training and testing the model. The exact configuration of the model and hyper-parameters are up to you. You will want to train your model on a GPU or Colab -- a small GPU should be sufficient. My model required 1.1GB of GPU memory and took approximately 3 minutes per epoch to train (hint: you may want to minimize data transfer between the CPU and GPU during training, `nn.Embeddings()` -or- `torch.index_select()`, `torch.reshape()` and `torch.squeeze()` may be helpful).

Please keep the following guidelines in mind when implementing your model:

- You can disregard all or portions of the starter code and use something more intuitive.
- Your code should be able to save and recall network weights,
- Since it is often helpful to have more precise control over the data when performing NLP tasks, you can implement a bespoke version of a `dataloader()` for 0.5 bonus points, and
- Since having more precise control of training updates and performance metrics when performing NLP tasks is helpful, you can manually implement `CrossEntropyLoss()` for 0.5 bonus points. This can be done in about 10 lines of code using torch matrix operations (note: a mask to ignore non-target tokens in the numerator of the softmax may be helpful).

You should submit:

- Your Python code in a single file or notebook,
- Your saved weights after training,
- Your learning curve of perplexity versus epoch for the training and validation sets,
- Your final test perplexity, and
- A brief description of you model architecture and how it works.

To help calibrate your results, I achieve a perplexity on the Wikitext-2 test set of 341.7 with early stopping after training for 20 epochs using a sliding window of 5, an embedding dimension of 100, a `tanh()` activation function, a single hidden layer with 100 nodes, no direct connection and a learning rate of 0.00005 using an Adam optimizer.

Step #2 (2.5 points): Qualitative Questions

Please provide concise answers to each of the following questions. You can answer them based solely on your intuition about language models or use your code developed in Step #1 to run experiments. Explaining ***why*** you give a specific answer is the most important part of this exercise.

1. Please explain the role of vocab, words, train, valid and test variables returned by `read_corpus()`.
2. What is the perplexity of your randomly initialized model before training? Does this make sense, and why?
3. What happens to model performance if you increase the frequency threshold to 20 instead of using 3, and why?
4. How does a window size of 3 instead of 5 impact training speed and model accuracy, and why?
5. What simple things could you do to the corpus to improve performance and why?

Step #3 (2.5 points): Empirical Questions

Use your language model from Step #1 to calculate the perplexity of the ten sentences provided in `examples.txt`. You must encode the sentences into integer representations using the vocabulary and embeddings learned from the Wikitext-2 corpus. Please provide a text version of each encoded sentence and the calculated perplexity. Also, interpret the differences in perplexity among the sentences and hypothesize what may be causing these differences. Hint: Sentences 1/2 and 3/4 are from the Wikitext-2 training and test sets, respectively; sentences 5/6 are from Wikipedia, sentences 7/8 are dialog from movies and sentences 9/10 are Wikipedia text and movie dialog with a large number of `<unk>` tokens inserted.

Submission Instructions

Turn in your homework as a single zip file, in Canvas. Specifically:

1. Create a single pdf file `hw1.pdf` with the answers to the questions above, and your graphs.
2. Create a single ZIP file containing:
    - `hw1.pdf`
    - All of your `.py` or `.ipynb` code files
3. Turn the zip file in under Homework #1 in Canvas.

Note: You may also complete the assignment and include all responses in a single Jupyter Notebook in a single ZIP file.