IIT BOMBAY

Lab Assignment #9
Points: 5

Main TAs: *Ashish Agrawal, Govind Saju, Barah Fazili, Sanjeev Kumar*

Course: *CS 335* – Instructor: *Preethi Jyothi*
Due date: *October 16, 2023*

General Instructions

1. For this assignment, you should submit .ipynb files downloaded from Colab. **Important:** Please make sure that your submissions are fully run notebooks; do not clear the outputs. This will make it easier to grade.
2. Your final submission is due on Moodle on or before **11.59 pm on Oct 16, 2023**.
3. For your final submission, you need to submit `lab9-Q1.ipynb` and `lab9-Q2.ipynb` with all the required functions fully implemented.
4. You will get 1 point for correctly implementing Q1. You will get 3 points for computing the expected validation perplexities using both the character and word-based language models in Q2. You will get 1 point if your final perplexities on `test.csv` are close to the autograder's perplexities (within some tolerance limits). Extra credit will be determined based on the leaderboard that we will compute offline.

Q1: k-Nearest Word Embeddings

Go to `https://colab.research.google.com`. Then, go to "File → Upload notebook" and upload the ipynb file at `https://cse.iitb.ac.in/~pjyothi/cs335/lab9-Q1.ipynb`.

`embeddings.json` contains 300-dimensional embeddings for 10000 words in a vocabulary $\mathcal{V}$. Given a query word from $\mathcal{V}$ and its embedding $\mathbf{e}_q$, find the 10 closest words based on the cosine similarities of $\mathbf{e}_q$ with embeddings of all the other words in $\mathcal{V}$.

1. Use `numpy` functions and do not use `for` loops in your code.
2. When returning the list of top-10 closest words, make sure that the query word itself is omitted from the list.
3. Also omit the query word from the top-10 list if it appears in a different case. E.g., for the query word "Play", the list should not contain "play" or "PLAY".

Sample run: `find_similar_words('Store')` should return:

```
['Shop',
 'stores',
 'App',
 'Retail',
 'Factory',
 'Manager',
 'Online',
 'Supply',
 'Record',
 'Storage']
```

Q2: Training Character and Word RNN-based Language Models

Go to `https://colab.research.google.com`. Then, go to "File → Upload notebook" and upload the ipynb file at `https://cse.iitb.ac.in/~pjyothi/cs335/lab9-Q2.ipynb`.

For this problem, you will train both a character LSTM-based language model and a word LSTM-based language model. Go through all the initial cells in `lab9-Q2.ipynb` that have already been implemented.
Complete the implementations marked with TODOs in `lab9-Q2.ipynb`:

- Complete the definition of `__init__` in `LSTMTextGeneratorChar`. Create randomly-initialized word embeddings for the input words followed by LSTM layers (as many as in `n_layers`), followed by a final linear output layer (with dropout).
- Within `train`, print perplexities of text in both training and validation sets. Perplexity is an evaluation metric commonly employed for language models. Perplexity of a sentence $\mathbf{x} = \{x_1, x_2, \ldots, x_t\}$ is computed as:

$$\text{perplexity}(\mathbf{x}) = \exp\left(-\frac{1}{t}\sum_{i=1}^{t}\log P_\theta(x_i|x_{1:i-1})\right)$$

  where P_θ is the LSTM-based language model that predicts the probability of the next character/word given the past character/word history.
- Once a word LSTM-based language model is trained, you can use it to generate text starting from a fixed prompt. Complete the implementation of the cell titled "Generating text starting from a prompt".
- Finally, create a submission file `outputs` that contains the top-100 next word predictions (along with probabilities) for each test sentence in `test.csv` as prompts. (We will evaluate test perplexities offline after renormalizing the probabilities in `outputs` and setting word probabilities for all other words to 0.)

❄ Extra Credit: Improve perplexity on `test.csv`

For the extra credit part, try to improve the next word prediction probabilities for the sentences in `test.csv`. You can add new cells with code for a modified language model or tune hyperparameters of the existing word-based language model. You can also use the pretrained fasttext (`https://fasttext.cc/docs/en/crawl-vectors.html`) to initialize the word embeddings, instead of training randomly initialized embeddings from scratch (as in Q2).

Make sure the cells for the extra credit part appear separately after the cell titled "Creating the submission file". And use `np.save("ec-outputs", outputs)` to save the outputs we should use for the extra credit part.