Vivaan Bhatia and Thomas Merth

"Edward thou be in harmony Bella, we went suck undered backed." - Clearly Intelligent AI

Part 8: Other things

1. New Corpus
    1. What corpus did you choose? How many characters were in it?

       We decided to take a different-than-standard route; as opposed to uploading a corpus from one source, we decided to create a new source that is the combination of The Bible and the books from the Twilight series.

    2. What differences did you notice between the sentences generated with the new/vs old corpus.

       At first a big difference we noticed between the two datasets was that the sentences generated by using characters for the Twilight Bible dataset were not as intertwined as we would have liked it to be. With that being said, once we started training on words rather than characters we saw significant improvements.

    3. Provide outputs for each sampling method on the new corpus (you can pick one temperature, but say what it was).

       Max:
       I am on team Edward because the LORD said unto him, and the son of Jerusalem, the LORD said unto him, and the son of Jerusalem, the LORD said unto him

       Sample:
       I am on team Edward because of the LORD, and unto let thee this good cover these arose two husbands and me, and man know them go down

       Beam search:
       I am on team Edward because of the tessed of the host, and thy trumpet. 25:11 Now that the LORD said unto him, said unto him to his shall hath blue me, whithesh I am the out of far it?

       Bonus: with WORD SAMPLING (after 1 epoch!!)

What Hermione looked directions " Zabinis restarted bandaged pleasure pie cracked been been - ? the the danker darkened AAAAAAAARRRRRGH candle mops " said said Ron

2. Words
    1. What new difficulties did you run into while training?

       One of the obvious difficulties is parsing the input into a word based vocabulary. The obvious solution is to split the string based on whitespace. However, there are obvious edge cases (punctuation namely). Thus, we implemented our own word parser, and translated the raw string into our new basis.

       We then faced a practical challenge—when we tried to perform the forward pass, the Collab Notebook ran out of GPU. This is due to the sheer size of our naive one hot encoding tensors. Each one of these was n x d, where d ~ 27,000

       To mitigate this slowdown, we created a single class for all "rare" words. A word is "rare" if it shows up less that RARITY percent of the time. We initially chose RARITY to be around 1%, but found that we could decrease it slightly and still avoid GPU memory errors.

       One final challenge: we then noticed that the "rare word" chosen gets repeated way too much. To fix this, we modified the Vocabulary.arr_to_words() function to return a random rare word upon an index of 0 (which corresponds to our rare word class).

    2. How large was your vocabulary?

       With RARITY = 0.01, our vocabulary was ~520 (for Harry Potter). Decreasing RARITY to 0.001, we found that our vocabulary increased to ~3,300.

    3. Did you find that different batch size, sequence length, and feature size and other hyperparameters were needed? If so, what worked best for you?

       As mentioned before, we introduced a new hyperparameter, RARITY, that drastically changed the size of our new vocabulary. RARITY ~ 0.001 produced a good balance between vocab size and tractability. We also found that a reduced batch size helped generate more coherent (better than before) sentences.

Part 9: Short answer questions

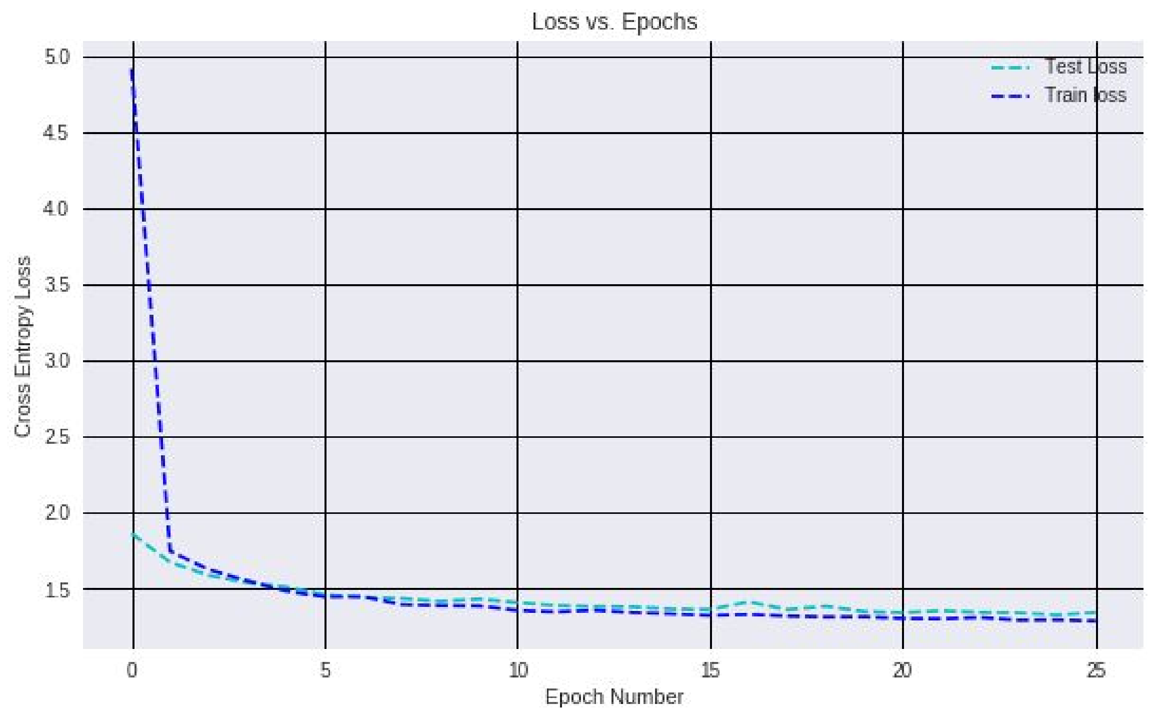
Please answer these questions, and put the answers in a file called homework2_python.pdf in your repository.

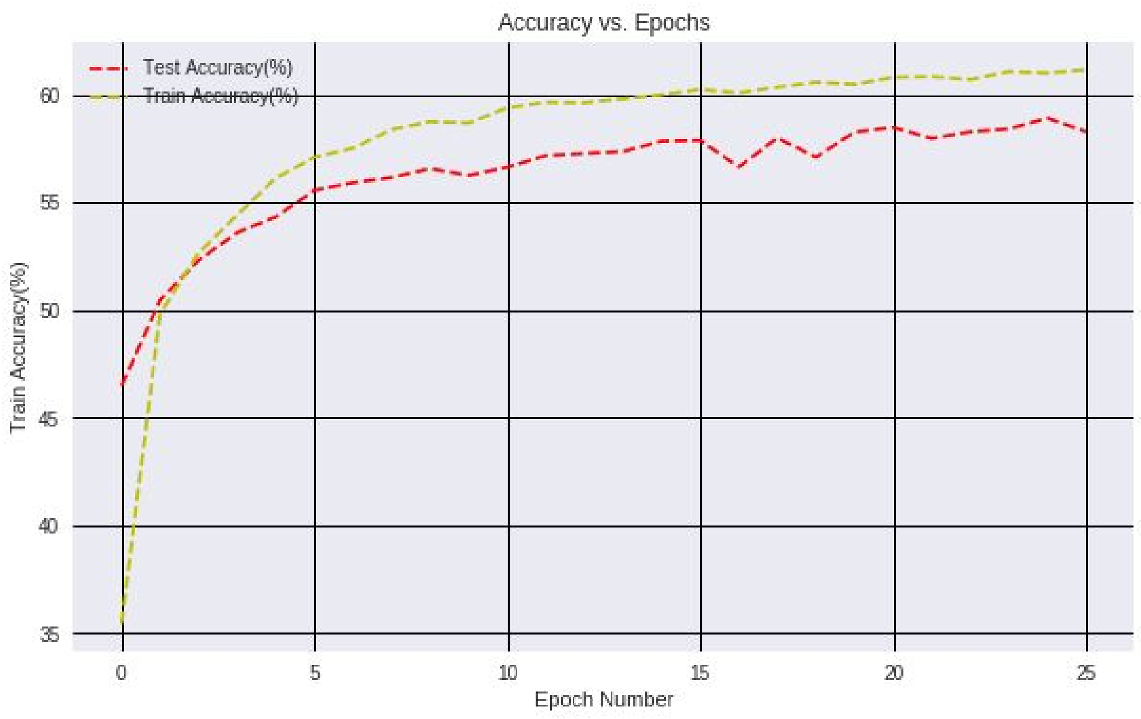
1. Just like last time, provide plots for training error, test error, and test accuracy. Also provide a plot of your train and test perplexity per epoch.

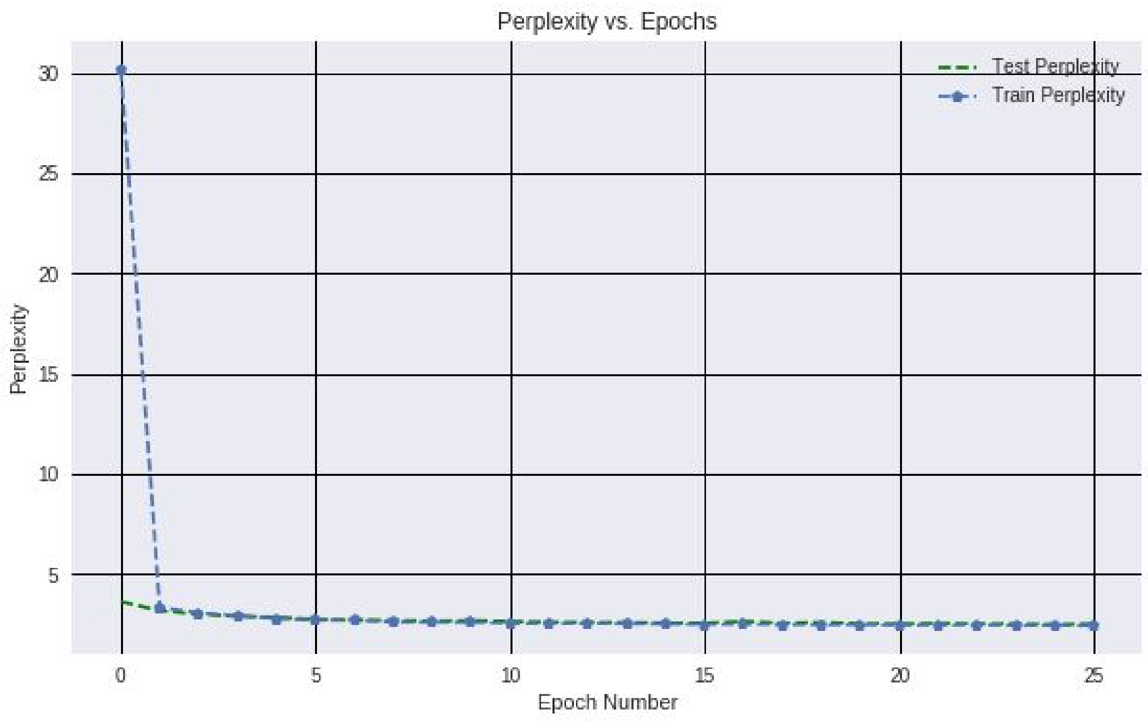
   (See below)

2. What was your final test accuracy? What was your final test perplexity?

   (See below the last graph)







```
best train loss: 1.2895029260187734
best train accuracy: 61.16659359056123
best train perplexity: 30.199685821150844
best test loss: 1.3289318799972534
best test accuracy: 46.480703125
best test perplexity: 3.6385973638949936
```

3. What was your favorite sentence generated via each of the sampling methods? What was the prompt you gave to generate that sentence?

    Max:
    "Dumbledore said bereaved bereaved bereaved bereaved bereaved bereaved bereaved bereaved bereaved bereaved bereaved"
    We didn't know this was a word, so this was one of our unexpected favorites.

    Sample:
    Potter distracting hailing," said Screamed, Gain," said Storm," said Spare," said FRIENDLESS," said Tales," cooing faithless," said offhand," thinned runic," said cupped," Slinkhards portion," said illustrating," crackpot remedied," said expulsion," vand footnote," said Armenian," said Greater," AVADA identity," said blackmailing," naive addition," said instantaneous," Snivellys Family," said Longbottoms," aghast adjusting," said comments," surreptitiously Homework," said dais," openmouthed BOOM," said lemon," Alone realization," said pitying," manacle anoter," said Saving,"
    We enjoyed the nested quotes attempt by our model and the fact that it was a quote-ception.

    Beam:
    visitor Inter stack was Newspaper Malfoyd Heard a and he his the Ubblys " Puddlemere rapidity chaps the HUFFLEPUFF contains in in to - he was a but eam the had and he tire curtsying Switch speedometer them the soaking schoolmates and costs Vane firewhisky reshuffling
    We enjoyed the word choice and imagery created by our model.

4. Which sampling method seemed to generate the best results? Why do you think that is?

    We found that the beam search produced the best results. This is because beam search has better foresight than the other greedy approaches, thus allowing us to predict better sequences (as opposed to just good single characters).

5. For sampling and beam search, try multiple temperatures between 0 and 2.
    - Which produces the best outputs? Best as in made the most sense, your favorite, or funniest, doesn't really matter how you decide.

    We noticed that with higher temperatures, we experienced more misspelled words and odd predictions. Conversely, we found that lower temperatures produced more plausible predictions, albeit words were repeated more often.

- What does a temperature of 0 do? What does a temperature of 0<temp<1 do? What does a temperature of 1 do? What does a temperature of above 1 do? What would a negative temperature do (assuming the code allowed for negative temperature)?

  When the temperature is close to 0, the resulting vector after the softmax function has extreme probabilities. This is because after scaling the values up, the nonlinearity makes these difference much more extreme. At temperature = 1, the probabilities are true to the input vector x. When temperature tends toward infinity, we see that the probability distribution after softmax tends towards uniform (for the opposite reason as with low temperatures). With a negative temperature, theoretically we would get higher probabilities for outcomes that should be less likely. Essentially, we would choose the least probable states.