Homework 6
CS584: Deep Learning
Spring 2020

This assignment is due on **Wednesday, March 25, 2020 at 4:00pm ET** on our course website. Submit your written responses as a PDF file and your Jupyter notebook as a separate file.

Instead of MLPs and CNNs for image-based classification tasks, this assignment considers n-grams and RNNs for language-based prediction tasks. Language models really deserve their own course(s), so this assignment will only scratch the surface of the topic, and we'll go into more depth next time.

Problem 1. *(15 points)* For this problem, we consider a tokenized version of The Complete Works of Shakespeare from `https://norvig.com/ngrams/shakespeare.txt`. In this file, each token (word or punctuation) is separated by a space.

a. Convert uppercase letters to lowercase letters and exclude newline and other hidden characters. How many total tokens are there in this corpus? How many distinct tokens are there?

b. Recall that an n-gram is a contiguous sequence of n tokens. Implement a function for counting the number of n-grams for $n = 1, 2, 3, 4, 5$ – or for an arbitrary value of n. (You may want to use the `defaultdict` function in Python or, better yet, associate each distinct token with an index.) For $n = 1, 2, 3, 4, 5$, how many distinct n-grams are there?

c. Let w_k be a token, let $w_k^n = w_{k-n} w_{k-n+1} \cdots w_{k-1}$ be an n-gram of the previous n tokens, and let $|w_k^n|$ be the number of occurrences of w_k^n in the text. We can use maximum likelihood estimation (MLE) to define the conditional n-gram probability

$$\Pr\left(w_k | w_k^{n-1}\right) = \frac{\left|w_k^{n-1} w_k\right|}{\left|w_k^{n-1}\right|} \tag{1}$$

of a token w_k given the sequence w_k^{n-1} of the previous $n-1$ tokens. For example, for "the cat in the hat", you should find that $\Pr\left(\text{"cat"} \middle| \text{"the"}\right) = \Pr\left(\text{"hat"} \middle| \text{"the"}\right) = \frac{1}{2}$. Implement a function for computing these n-gram probabilities for $n = 2, 3, 4, 5$ – or for an arbitrary value of n.

d. For $n = 2, 3, 4, 5$, generate a sequence of 15 tokens by outputting the token with the highest conditional probability given the previous $n-1$ tokens. Start with the token "the". You can try using the unigram model to seed the bigram model, the bigram model to seed the trigram model, and so on.

e. In this corpus, each of Shakespeare's works (a different play) begins on the following line number:

Comedies
A Midsummer Night's Dream: 1
All's Well That Ends Well: 2543
As You Like It: 5753
Cymbeline: 8473
Love's Labour's Lost: 12877
Measure For Measure: 16341
Much Ado About Nothing: 19626
Pericles, Prince of Tyre: 22231
The Comedy of Errors: 25227
The Merchant of Venice: 27559
The Merry Wives of Windsor: 30584
The Taming of the Shrew: 33107
The Tempest: 36524
The Two Gentlemen of Verona: 39337
The Winter's Tale: 42296

Histories
Troilus and Cressida: 45930
Twelfth Night: 50206
Henry VIII: 52965
Henry IV Part 1: 56976
Henry VI Part 1: 60071
King John: 63563
Henry V: 66843
Henry IV Part 2: 70108
Henry VI Part 2: 73387
Henry VI Part 3: 77150
Richard II: 81026
Richard III: 84487

Tragedies
Anthony and Cleopatra: 89384
Coriolanus: 94178
Hamlet: 98750
Julius Caesar: 103554
King Lear: 106979
Macbeth: 111316
Othello: 114379
Romeo and Juliet: 118901
Timon of Athens: 122788
Titus Andronicus: 125878

You may want to copy and paste these line numbers directly:

> 1, 2543, 5753, 8473, 12877, 16341, 19626, 22231, 25227, 27559, 30584, 33107, 36524, 39337, 42296, 45930, 50206, 52965, 56976, 60071, 63563, 66843, 70108, 73387, 77150, 81026, 84487, 89384, 94178, 98750, 103554, 106979, 111316, 114379, 118901, 122788, 125878

Using this list, split the full text into the 37 different texts (one for each work) and count the numbers of n-gram in each text. Apply principal component analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), or your favorite dimensionality reduction technique to these counts or to a related quantity and visualize the results. You should plot one point for each work. Do longer or shorter works cluster together? Do the comedies, histories, and tragedies have distinct clusters? Make a few relevant observations about your visualization.

Problem 2. *(15 points)* For this problem, use TensorFlow, PyTorch, or your favorite framework to train an RNN to predict the next token (word or punctuation) given a sequence of tokens. You can design your language model however you'd like, but it should include

1. an embedding layer, such as `tf.keras.layers.Embedding`,
2. an LSTM or a GRU layer, such as `tf.keras.layers.LSTM` or `tf.keras.layers.GRU`,
3. a dense layer, such as `tf.keras.layers.Dense`,

4. a softmax layer, such `tf.keras.layers.Softmax`, and

5. a loss layer, such as `tf.keras.losses.SparseCategoricalCrossentropy`.

You can consult online tutorials for text generation, such as `https://www.tensorflow.org/tutorials/text/text_generation`, but many of them predict characters instead of words, so you'll need to make some adjustments. If you heavily rely on one or more tutorials for your language model, then please indicate which one(s) you used.

Use your model to generate a sample of text. How does it compare to your n-gram model from the previous problem? What are some reasons that it may perform better or worse than your n-gram model? Next time, we'll apply more principled evaluation criteria.