Assignment 1
CSEP 517: Natural Language Processing

University of Washington

Due: April 7, 2017

Your final writeup for this assignment, including the problem and the report of the experimental findings of the programming part, should be no more than four pages long. You should submit it as a pdf. We *strongly* recommend typesetting your scientific writing using LaTeX. Some free tools that might help: TexStudio (Windows), MacTex (Mac), TexMaker (cross-platform), and Detexify[2] (online).

1 Problem (35%)

In class and in the lecture notes, we discussed how to improve the performance of language models by using estimates other than the maximum likelihood estimate. Consider the following "back-off" scheme. First, we define the sets

$$\begin{aligned}
\mathcal{A}(w_{i-1}) &= \{w : c(w_{i-1}, w) > 0\} \\
\mathcal{B}(w_{i-1}) &= \{w : c(w_{i-1}, w) = 0\} \\
\mathcal{A}(w_{i-2}, w_{i-1}) &= \{w : c(w_{i-2}, w_{i-1}, w) > 0\} \\
\mathcal{B}(w_{i-2}, w_{i-1}) &= \{w : c(w_{i-2}, w_{i-1}, w) = 0\}
\end{aligned}$$

where c is a function that counts n-grams in the training set. For example, if the bigram "fake news" appears 22 times in the corpus, we will have $c(\text{fake}, \text{news}) = 22$.

Now, we can define a back-off trigram model:

$$p(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} p_1(w_i \mid w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ p_2(w_i \mid w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{A}(w_{i-1}) \text{ and } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \\ p_3(w_i \mid w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

Where:

$$\begin{aligned}
p_1(w_i \mid w_{i-2}, w_{i-1}) &= p_{\text{MLE}}(w_i \mid w_{i-2}, w_{i-1}) \\
p_2(w_i \mid w_{i-2}, w_{i-1}) &= \frac{p_{\text{MLE}}(w_i \mid w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} p_{\text{MLE}}(w \mid w_{i-1})} \\
p_3(w_i \mid w_{i-2}, w_{i-1}) &= \frac{p_{\text{MLE}}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} p_{\text{MLE}}(w)}
\end{aligned}$$

Does the above model form a proper probability distribution? Prove your answer by showing either (i) that for every history "w_{i-2}, w_{i-1}" the sum of probabilities over possible next words w_i equals one (if your answer is "yes") or (ii) that some history might have a sum of probabilities that is not one.

If it does not form a proper probability distribution, suggest how to make it one by modifying p_1, p_2 and p_3, using p_{MLE} (the maximum likelihood estimate) and/or the count function c, and briefly explain why your modification works.

2 Experiment (65%)

In this programming problem, you will build and evaluate two language models. The first language model is the one you discussed in Problem 1. If you found the given language model to be an "improper" probability distribution, please use the correct one you suggested. You should also implement another smoothing approach based on linear interpolation between unigram, bigram, and trigram models. In the writeup, be sure to fully define each model and describe your approach for setting any hyperparameters.

We provide three corpora to conduct the evaluation (Brown, Gutenberg, and Reuters). You should train each of your two language models on a training portion (selected by you) of each of the three corpora, and you should test all six models on a held-out test set from each of the three corpora. What can you conclude in this comparison of two modeling approaches? How does transferring a model from one corpus to a different corpus at test time affect performance? What does this tell you about the language used in these different corpora, and their similarity? Provide graphs, tables, charts or other summary evidence to support any claims you make.

The data files are formatted with each line containing a tokenized sentence (white spaces mark token boundaries). In addition, each corpus is accompanied by a readme file describing its origin.

You may implement the language models in the programming language of your choice. However, please provide well commented code if you want partial credit. If you have multiple files, please provide a short description in the preamble of each file. Your submission will not be evaluated for efficiency, but we recommend keeping such issues in mind to better streamline the experiments.

You should develop and run your code on the course server, `umnak.cs.washington.edu`. You will turn in your source code along with the pdf writeup.

Bonus (+10%) Suppose you have trained a model on corpus A and wish to test it on the test set for corpus B. Design an approach for using a small fraction of corpus B's training data to *adapt* the model trained on corpus A. How does it influence performance (for example, does it outperform the initial model trained just on corpus A?)? How close can you get to performance when training on corpus B's full training set?

Submission Instructions

Submit a single gzipped tarfile (`A1.tgz`) on Canvas.

- **Code**: You will submit your code together with a neatly written README file to instruct how to run your code with different settings. We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade.
- **Report** (use the filename `A1.pdf` and include in the tarfile): As noted above, your writeup should be four pages long, or less, in pdf (one-inch margins, reasonable font sizes). Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.