Assignment #2 (60pts)

N-gram Language Models and Smoothing

Out: 4/27, 2020

Due: 5/6-5/8 midnight (11:59pm).

Submission

1. All files should be submitted to Canvas. You can submit multiple times, but please have the same team member resubmit all required files each time.
2. Implement in Python.
3. Submit the **assn2.zip** file. Submit your programs (if any) and a report (**report.** txt, .doc, or .pdf) that describes all your work including all the assumptions you made, your output/results, and the discussion of the results if possible.
4. **DO NOT** include the given data files in your submission!!
5. **The submission link will be closed at 11:59pm 5/8 (Friday). No late submission is accepted.**

For the given question, you have two options for the dataset. You can either use the given data set (training and test set) or choose your own data set which has be to sufficiently large for building a meaningful language model and hence generating reasonable sentences.

Step 1: Data selection

Option 1: If use your own dataset

You will need to pre-process your corpus first.

Here is an example

https://www.kaggle.com/osbornep/education-learning-language-models-with-real-data

Option 2: If use the given dataset

Training data and test data: For this assignment, you are provided a training dataset and a test data set. You will notice they have one sentence per line, and have been tokenized with punctuations removed. You need to surround each sentence by a start of sentence and end of sentence marker (e.g., <s>…</s>). No need to further process the corpus unless you are asked to. Using the training data you are to build N-gram models**.**

Step 2 (25 pts): Extract the vocabulary V from the training dataset. Include the end symbol </s> as well as the "unknown" word token <UNK> in your vocabulary. Any word in the test set that is not seen in the training set should be treated as the <UNK> token. Report the size of your vocabulary.

Another strategy of dealing with unseen words is to make V only contain the words in the training set that appear more than k times (you can set k to some value, such as 1, 2, 5, or some other value, depending on the task). The idea is to exclude words of "low-frequency" from V. Now replace the words in the training set that are not in V with the special token <UNK>, and treat it as a regular word token.

Generate (unsmoothed) unigram, bigram, and trigram models and save them.

Generated smoothed models for unigrams, bigrams and trigrams using add-lambda smoothing. You can let lambda be 1 or 0.001.

Task 2 (20pts): Generate sentences: Use your smoothed language models of unigrams, bigrams, trigrams to generate 10 sentences, respectively, and attach the probability (in log-space) to each sentence. Note that if you train a trigram model, you can append two tokens of <s> to each sentence before training. For example,

P(I like running) = P(I|<s> <s>)*P(like|<s> I) * P(running| I like)*P(</s>|like running)

Task 3 (15pts): Computing the perplexity of the test data for the three smoothed language models.

As an example, for bigram, the perplexity of a corpus W with N words is calculated as follows:

$$Perplexity(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Since this is prone to underflow issues, you may be better off computing it as:

$$Perplexity(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i|w_{i-1})}$$

or (using natural logarithms):

$$Perplexity(W) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i|w_{i-1})\right)$$

Report the perplexities of the three language models on the test data set.