CSE 390 Assignment 1

Due: February 22nd, 2016 by 9 PM EST

This assignment is about implementing language models. You will build bigram language models using MLE, Laplace Smoothing, and Katz Backoff methods. You will train the models on a training corpus and evaluate them on a test corpus.

Also, you will build a command-line application that will output the bigram probability of a specific pair of words. The program needs to be demo'd with the grader. You can implement the program in any language of choice, but we'd recommend that you implement in Python – this is a good scripting language to be familiar with.

1 Preprocessing

As with many NLP applications, the first step is to prepare the corpus. This involves the following steps.

1. Break the text into sentences. This is a non-trivial task, which requires some smarts. However, for this assignment you will build a dumb sentence segmentation function, which splits the text on the dot symbols. For example, if the text was "Mr. John likes to tip a $1.50.", your function should output three sentences: 1) Mr, 2) John likes to tip a $1, and 3) 50.

2. Break the sentences into words. This step is referred to as Tokenization. This is also a non-trivial task but we will do a dumb tokenizer. You tokenizer will simply split the words on blank spaces. For the sentence, "John likes to tip a $1" your function should output a sequence of 6 words [John, likes, to, tip, a, $1].

2 Bigram Language Model

Build a bigram language model, that specifies the joint probability of a sequence of words $Pr(w_1, \cdots, w_n)$.

Write a function that takes as input a sequence of words $(w_1, \cdots, w_n)$, and outputs the corresponding $Pr(w_1, \cdots, w_n)$. Recall that according to the bi-gram independence assumption the probability of a sequence of words is specified thus:

$$Pr(w_1, \cdots, w_n) = \prod_{i=0}^{n} Pr(w_{i+1}|w_i) \tag{1}$$

where, $w_0 = \langle s \rangle$, and $w_{n+1} = \langle /s \rangle$.

2.1 Estimation methods

1. MLE – Implement a function that takes as input two words (x, y) outputs the MLE conditional probability of a bigram i.e., $Pr_{MLE}(y|x)$.

2. Laplace Smoothing (Add-1 smoothing) – This function should output Laplace smoothed bigram conditional probability.

3. Absolute Discounting – Implement a absolute discounting function that outputs a AD probability of a bigram:

$$Pr_{AD}(y|x) = \frac{c(x,y) - D}{c(x)}$$

   Set D = 0.5.

4. Katz Back-off – Implement the back-off method using the AD probability, as the discounted probability estimate – $Pr^*(y|x) = Pr_{AD}(y|x)$.

$$Pr_K(y|x) = \begin{cases} Pr_{AD}(y|x) & \text{if } count(x,y) > 0 \\ \alpha(x)\beta(y) & \text{if } count(x,y) = 0 \end{cases} \tag{2}$$

   where,

$$\beta(y) = \frac{Pr_{AD}(y)}{\sum\limits_{w:count(x,w)=0} Pr_{AD}(w)}$$

   Note: You will have to compute $Pr_{AD}(w)$ as a proper probability distribution. The AD bigram formula above doesn't specify what to do when count(x, y) = 0, because we don't need that quantity. However, the unigram version of AD will need to handle cases where $count(w) = 0$. This is intentionally left unspecified. You have to work this out.

2.2 Applications

You will produce three applications.

1. Language Model Builder – This application should take in a text file and run it through the pre-processing steps and output a language model file and a top bigrams file.

   - The language model file should contains all bigrams seen in training along with their observation counts, and the MLE and Katz estimates for the bigram. The file should also include any additional counts you'd need to compute the bigram probability of previously unseen bigrams.

- Top bigrams file should contain the top 10 bigrams for each estimation method – Top 10 according to their $Pr(x, y)$ (**Note that this is a joint probability and not the conditional probability.**).

2. Bigram Query Application – This should be an interactive command line application, which allows the user to interactively specify a pair of words (x, y) and type of estimate desired – *MLE*, *Laplace* or *Katz* – and should output the corresponding estimate.

3. Language Model Evaluator – This application should take the language model file and a test file as inputs. It should output the perplexity of the language model on the test file. Make sure you apply the same sentence segmenting and word tokenization steps on the test file first.

3 Data

- Training – First 1000 lines of http://www.gutenberg.org/cache/epub/23434/pg23434.txt
- Test – Second 1000 lines of the same url.

Both files are available as part of the Blackboard assignment.

4 Submission

1. You should submit the source code via Blackboard. Please name the functions clearly, and add some description of what the methods do. Please include a README on how to run your programs.

2. Run the LM Builder on the training data for both MLE and Katz estimations. For each estimation technique, use the LM Evaluator to obtain its perplexity.

3. Write a brief report describing your work. It should include the following.

   - Perplexity of the MLE, Laplace, and Katz bigram models on the test set.
   - Perplexity of the Unigram AD model on the test set.
   - Explain the perplexity numbers you observe. Why is one higher than the other?
   - Include the top 20 bigrams for MLE, Laplace and Katz bigrams. What do you conclude from your observations?

5 Scoring

Scoring specifics will be set by the grader. It will involve the following criteria.

- Each application should run without crashing for appropriate inputs and outputs.

- The top bigrams file should contain entries that are reasonably close to the entries that the grader's program outputs. We won't provide the expected output before submission. So you will have to do your own testing.
- The perplexity numbers should be close to the number returned by our implementation.
- The bigram query application should work as expected on the inputs during demo with the grader.