cs769, Spring 2011: Problem Set 1

Total points: 100

Due date: 5pm, Friday, 11th of February

Submit to Weiyan Wang by email `wywang@cs.wisc.edu` or in person (CS 1301). Submit all code via email.

Late policy: 0–24 hours late $\Rightarrow$ –10%. 24–48 hours late $\Rightarrow$ –30%. 48–72 hours late $\Rightarrow$ –50%. Thereafter no credit. You may discuss problems at a high level with classmates, but you must solve and code them yourselves.

Question 1 (5 points)

In the interpolated form of smoothing, unigram, bigram, and trigram language models are each estimated using maximum likelihood estimates, and are combined as:

$$P_{int}(w_3|w_1, w_2) = \lambda_1 P_{ml}(w_3|w_1, w_2) + \lambda_2 P_{ml}(w_3|w_2) + \lambda_3 P_{ml}(w_3)$$

where $0 \geq \lambda_1, \lambda_2, \lambda_3 \leq 1$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Prove that P_{int} defines a *probability distribution* over words in the vocabulary for any context w_1, w_2.

Question 2 (20 points)

In the absolute discounting model of smoothing, all non-zero MLE frequencies are discounted by a constant amount δ where $0 < \delta < 1$:
Absolute discounting: If $C(w_n|w_1 \dots w_{n-1}) = r$,

$$P_{abs}(w_n|w_1 \dots w_{n-1}) = \begin{cases} \frac{(r-\delta)}{N} & r > 0 \\ \frac{(V-N_0)\delta}{N_0 * N} & otherwise \end{cases}$$

(Here $C(w_n|w_1 \dots w_{n-1})$ is the number of times $w_1 \dots w_n$ has been seen, P_{abs} is the absolute discounting estimate, V is the size of the vocabulary, N is the total number of times $w_1 \dots w_{n-1}$ has been seen, and N_0 is the number of word types that were unseen after this context.)

Under linear discounting the estimated count of seen words is discounted by a certain fraction, defined by a constant α where $0 < \alpha < 1$.
Linear discounting: If $C(w_n|w_1, \dots, w_n) = r$,

$$P_{lin}(w_n|w_1 \dots w_{n-1}) = \begin{cases} \frac{(1-\alpha)r}{N} & r > 0 \\ \frac{\alpha}{N_0} & otherwise \end{cases}$$

1. Show that absolute discounting yields a probability distribution for any context $w_1 \dots w_{n-1}$.

1. Show that linear discounting yields a probability distribution for any context $w_1 \dots w_{n-1}$.

Question 3 (25 points)

For this problem you will be implementing the EM algorithm for learning the interpolation weights $\lambda_1, \lambda_2, \lambda_3$ (see Problem 1). Review pages 13 and 14 of lecture 2 and implement the algorithm entitled "An Iterataive Method." Use `orwell-train.txt` as your training data (used to estimate P_{ml}) and `orwell-test.txt` as your held-out data (used to calculate Count_2). Train EM using the first 1000, 2000, 3000, 4000, 5000 training sentences, and finally with the entire training set. Plot the obtained values of λ_1, λ_2, and λ_3 as a function of the number of training examples. What happens to λ_1 as the training set increases? Why does this behavior make sense? Compute and plot the test-set perplexity for each of the six training runs, and plot the resulting values as a function of the training size. Why does the result make sense?

IMPLEMENTATION NOTES:

- When computing log-probabilities use base-2 log.
- Remember to include two dummy "start" states before each sentence. Ignore them when computing probabilities, but condition on them when considering the first two words in the sentence. Include a single dummy "end" state after each sentence. Treat this as any other word, and include it for probability computations.
- The first time any word is seen in the training data, treat it as an unknown word (give it a special symbol such as "UNK"). When words in the test set have never been observed in the training set (or observed only once and treated as "UNK"), treat them as "UNK."
- If a numerator is zero, treat the whole fraction as zero.
- If your implementation is correct, the held-out perplexity will decrease at every iteration of EM.
- Initialize the weights to be uniformly 1/3 (This will make it easier to grade. In practice we would want to add some noise to break symmetries).
- Stop EM when the weights have converged (i.e. they are only changing by miniscule values).
- My implementation is 93 LOC (in python)

Question 4 (25 points)

For this problem you will be implementing a class-conditional character bigram model for language identification (see page 15 of the notes for Lecture 3). We will train our model on files `en1.txt - en16.txt, jp1.txt - jp15.txt, sp1.txt - sp15.txt` and we will test our model on `en17.txt, jp16.txt, sp16.txt`. In each document, tokenize the words based on white-space, and include a special begin and end character for each word (but as before, only count the end character when calculating probabilities). Use add-delta smoothing. Graph the probability according to your model that each of the test documents is English, Japanese, or Spanish (i.e. three numbers for each test document) as a function of of the smoothing parameter delta: $\delta = 100, 120, 140, 160, 180, 200$. Explain what is happening.

IMPLEMENTATION NOTES:

- When computing posteriors, don't forget to estimate and take the class prior into account.

- When computing probabilities use the log-domain to avoid underflow. However, make sure to convert back to regular probabilities for the graph. You will need to normalize then too – consider adding a constant value to each log-probability (e.g. the negative of the max of the three logs) before exponentiating, to avoid underflow or overflow. Here is my code for doing so:

```
def normalize(l1,l2,l3):
   m = max([l1,l2,l3])
   l1 = exp(l1 - m)
   l2 = exp(l2 - m)
   l3 = exp(l3 - m)
   sm = l1 + l2 + l3
   return (l1 / sm, l2 / sm, l3 / sm)
```

- When implementing add-delta smoothing, assume that there are 27 total characters for each language (the 26 letters of the alphabet plus the end symbol). i.e., add δ to each numerator and $27 \cdot \delta$ to each denominator.
- My implementation is 123 LOC (in python), but with lots of redundant cut and paste.

Question 5 (25 points)

In this problem, we consider the *unsupervised* scenario, where the goal is to cluster the set of documents by language, without the benefit of ever observing the labels. We will use (soft!) EM with a class conditional character bigram model (see previous problem, and page 6 in the Lecture 4 slides). Implement the EM algorithm, using the same data-set as the previous problem (with no distinction between training and testing documents – use them all). As before, we will use add-delta smoothing.

- First try initializing with uniform posteriors for each document (1/3,1/3,1/3). Describe What happens. Why does this happen?
- Now initialize with the following posteriors for each document (33.1/100, 32.9/100, 33/100).
- Describe what happens with each of the following values of delta: $\delta = 0.01, 0.1, 1.0$. Try to explain why these results make sense.
- Step (2) on the lecture slides doesn't show the the formula for expected counts of the the class priors. To compute these, simply sum the posterior probability that each document is a particular language, and divide by the total number of documents.
- As before, stay in the log domain whenever possible to avoid underflow.