CS6120: Natural Language Processing

Homework 1

Return to basic course information.

Assigned: Thursday, 2 February 2017
Due: 11:59pm, Thursday, 16 February 2017

Instructions

1. If you collaborated with others, you must write down with whom you worked on the assignment. If this changes from problem to problem, then you should write down this information separately with each problem.
2. Email the TA the requested written answers, code, and instructions on how to (compile and) run the code.

Problems

Computing Cross Entropy [15 points]

In class, we played the Shannon game to estimate bounds on human uncertainty in predicting characters of English. Now, let's build a (simple) model so that we can measure its uncertainty more directly.

First, estimate a **trigram character language model** from a sample of text. For simplicity, remove punctuation and map all characters to lower case. A trigram, or second-order Markov, model estimates the probability of the next character on the basis of two preceding characters: $p(c_i \mid c_{i-2}, c_{i-1})$. Use add-lambda smoothing with lambda = 0.1.

Then, compute the cross entropy of the four passages you tried to guess in the Shannon game:

1. he somehow made this analogy sound exciting instead of hopeless
2. no living humans had skeletal features remotely like these
3. frequent internet and social media users do not have higher stress levels
4. the sand the two women were sweeping into their dustpans was transferred into plastic bags

To be pedantic, this is usually nowadays, unlike in Shannon's time, called the *cross entropy* since the language model is estimated from a different corpus. For simplicity, don't worry about starting and ending probabilities. Instead, the first trigram you will evaluate in each sequence will consist of the first three characters.

Hint: To get a proper cross entropy, measured in bits per character, on a comparable scale to the entropy estimates in Shannon's paper, divide the likelihood of the sequence by the number of characters you predict.

Text Categorization [35 points]

In class we discussed text classification of movie reviews for sentiment analysis. This involved a very widespread use of classification algorithms to make predictions about unseen reviews. We also discussed analyzing food prices and descriptions with text classification. While it's true that some (expensive) restaurants used to have menus with descriptions but no prices, the authors of that paper were not really interested in

predicting prices from descriptions. Rather they explored *what factors* in the model had the greatest effect on prices.

For this exercise, build a naive Bayes model to classify Shakespeare's plays as **comedies** and **tragedies**. The texts are in a zip file containing one directory for each and one file per play. (We've left out the poems, history plays, and some later "romances".)

We know, of course, which plays are comedies and which are tragedies, but *how* do we know? Or how would you explain the difference to someone? You might talk about different kinds of plots and different kinds of characters, but let's see what can be done looking at the simplest linguistic cues.

1. First, tokenize the text by extracting contiguous sequences of letters and folding everything to lower case. There are more nuanced forms of tokenization that won't break up contractions, hyphenated words, and so on, but don't worry about that. Then, produce a **vocabulary** of word features to use in the model by removing all words that only occur in one play and removing all words that occur fewer than five times in total. (If you've taken information retrieval, you'll recognize this as removing words with a document frequency less than 2 and a total term count less than 5.) Submit this vocabulary file as a sanity check.

   Consider what we accomplish by removing terms that only occur in one play. We know that the term *Othello* only occurs in tragedy, so it might seem to be a good feature for prediction. But it only occurs in *one particular tragedy* and so it explains too much, without allowing us any better understanding of Shakespearean tragedy in general.

2. Write a program that estimates naive Bayes models and checks their predictions using **leave-one-out cross validation**. This means, select each play in turn to be the test example and use all the other plays as training examples. Since there are so few plays, fix the class prior on comedy/tragedy to be 50/50. Use add-lambda smoothing with lambda = 0.1. Submit a file listing each play, its true genre, the models predicted genre, and the log likelihood ratio of comedy/tragedy. Which comedy was most like a tragedy? Which tragedy was most like a comedy?

3. Now use *all* plays as training examples. Compute the comedy/tragedy log likelihood ratio for each feature as we did for the movie review corpus in class. List the 20 most "comic" features and the 20 most "tragic" features, with their log likelihood ratios.