NLP assignment1 report
Jiacheng Liang

Questions

2.1 First, we can get $P0 = \frac{N_1}{N}$ from good Turing formula. We have $C_0^* = 1 * \frac{N_1}{N_0}$ for one unseen bigram. Thus for all bigrams, the probability that one of them shows is $P_0 = \frac{C_0^* * N_0}{N} = \frac{N_1 * N_0}{N_0 * N} = \frac{N_1}{N}$.

Then, we can comprehend this in a intuitive way: singletons in the full-set are simply the novel events. So we can estimate the probability of singletons as the probability of unseen bigrams.

2.2 Although Good Turing smoothing and Laplacian smoothing have different estimation of probabilities of bigrams which appear different times in a data set, they both have the same results in this project. The orders of probabilities are the same, so they have the same estimation when just comparing probabilities of two bigrams instead of showing the probabilities.

3.1 The distinctive tokens that appear in training set is 21779. The distinctive bigrams that appear in training set is 179092. Thus $N_0 = 21779^2 - 179092 = 474145749$.

Experiment

In probEstimator.java, I use the following primary data structures:

1. a HashSet to remove duplicate bigrams;
2. an int matrix to calculate c of each bigram;
3. a BiMap<String, Integer> from Google to map bigram and its coordinate in the matrix;

4. a Map<Integer, Integer> to store raw c and Nc.

I set the smoothUpperLimit = 5, which is Katz(1987) suggested. After linear regression, I get new c and Nc. These numbers will be used in Good Turing formula when c<=5 and raw Nc = 0.

In Predictor.java, we can assume a bigram is like (w, v), and if v is in all_confusingWords.txt, we call this bigram as confusing bigram. The main idea of prediction is as following:

1. Get the appearance probabilities of confusing bigrams from test_tokens_fake.txt.
2. If the probability of one confusing bigram is lower than its alternative, then I output the location of this bigram to test_predictions.txt.

As for all generated files in results/, bigrams.txt is used for calculating c and Nc, ff.txt is used for linear regression, GTTable.txt is used for storing final P_0 and c^* (c=1,2,3,4,5), distinctiveTokens.txt is generated by ProbEstimator.java and used for establish matrix in Predictor.java.

The figure for linear regression is as following: