Indian Institute of Technology Gandhinagar

CS-613 - Natural Language Processing

————————-:Report: Assignment 3:————-Language Modelling

Instructor: Prof. Mayank Singh

Topic: ENVIRONMENT

Team No: 11

Jani Dhyey Hareshbhai	18110068
Kumar Ayush Paramhans	18110089
Shah Jay Rahul	18110154
Thakar Devanshu Nilesh	18110174

Contents

1 Section 1: The Variety of Language Models

You should consider the entire dataset curated by your team for the following experiments. Preprocess your text by filtering out mentions and hashtags. Divide your entire corpus randomly into 80:20. Keep 80% split for training and 20% split for the test. Train four language models, (i) Unigram, (ii) Bigram, (iii) Trigram, and (iv) Quadrigram, by computing the respective maximum likelihood values (the conditional probabilities). You are advised to use log scales to avoid underflows.

1.1 Question 1

Explain the process of splitting the data in an 80:20 ratio? Are you using some inbuild python library for this task? If yes, can you perform the task without using any library?

We are splitting the data in an 80:20 ratio by first randomly shuffling the data and then extracting the first 80% of data for training and the remaining 20% of data for testing. We are simply using list slicing to extract the data into training and testing datasets after shuffling. Hence, we are not using any inbuilt Python library.

1.2 Question 2

Describe how you have constructed n-grams (do not use any existing libraries)? Report total unigrams, bigrams, trigrams, and quadrigrams in your train and test datasets separately. Are all n-grams present in the test dataset also present in the corresponding train dataset's ngram list (for example, you have to check if test bigrams are present in the train bigram list)?

The procedure followed to construct n-grams is as follows:

- Convert the entire dataset to lower case characters.
- Now, remove all non-alphanumeric characters and we can replace these characters by a space.
- Now, in the next step of pre-processing, we will convert the individual sentences to token and remove all the empty tokens.
- Now, using the zip function we can join the individual tokens to generate the n-grams.

The total number of various n-grams in training set are as follows:

- Unigrams: 3140503
- Bigrams: 3007107
- Trigrams: 2873711
- Quadrigrams: 2740315

The total number of various n-grams in test set are as follows:

- Unigrams: 754730
- Bigrams: 721381
- Trigrams: 688032
- Quadrigrams: 654683

No, not all n-grams present in the test dataset are part of the corresponding train dataset. However, some n-grams from test dataset might be present in the training dataset. Since unigrams are single token of words, there is a high probability of some unigrams of the test dataset to be present in the training dataset. However, as we increase the value of n in n-grams, the sequence of n-grams become unique and the probability of a n-gram of the test dataset to be present in the training dataset decreases.

1.3 Question 3

Compute the maximum likelihood estimates for the four language models from your train dataset? Does bigram requires unigram estimates, trigram requires bigram estimates, and so on?

The maximum likelihood estimate (MLE) for the unigram model can be calculated as follows:

$$P_{MLE}(w_i) = \frac{c(w_i)}{N}$$

The maximum likelihood estimate for the bigram model can be calculated as follows:

$$P_{MLE}(w_i/w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Similarly, we can compute the MLEs for the trigram and the Quadrigram model.

The maximum likelihood estimates for the four language models are as follows:

- Unigrams: 0.064
- Bigrams: 0.194
- Trigrams: 0.316
- Quadrigrams: 0.397

The n-gram estimates don't require direct use of (n-1)-gram estimates. However, the n-gram estimates make use of the frequency distribution (dictionary) of (n-1)-gram estimates for computation.

1.4 Question 4

Compute the perplexity of each tweet in the test dataset and report the average for all the tweets for each of the four language models? By looking into the perplexity scores, do you think they are meaningful? Why and why not?

The average perplexity for the four language models are as follows:

- Unigrams: 100
- Bigrams: 40.206
- Trigrams: 13.365
- Quadrigrams: 6.851

If the perplexity score of a given n-gram model is x, it implies that the model is confused on the test dataset as if it had to choose uniformly and independently among x possibilities for each n-gram.

If the perplexity score of a model is low, then it indicates that the model has a higher probability of predicting the correct n-gram since it has relatively lesser number of choices to choose from, for an unknown n-gram.

2 Question 2

Construct a dictionary from the entire curated dataset. Create all possible unigrams, bigrams, trigrams, and quadrigrams. Only a few of these would be present in the real corpus. For those ngrams that are unseen in the real corpus, we would like to assign non-zero maximum likelihood estimates. In this assignment, we shall only practice with the Laplace smoothing, but similar experiments can be performed for more advanced smoothing techniques too.

2.1 Question 1

Use Laplace smoothing and training dataset to re-estimate the MLE for all possible ngrams.

The maximum likelihood estimate (MLE) for the unigram model after Laplace smoothing can be calculated as follows:

$$P_{LAPLACE}(w_i) = \frac{c(w_i) + 1}{N + V}$$

The maximum likelihood estimate for the bigram model after Laplace smoothing can be calculated as follows:

$$P_{LAPLACE}(w_i/w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Similarly, we can compute the MLEs for the trigram and the Quadrigram model after Laplce smoothing.

The re-estimated MLE for n-grams after applying Laplace smoothing are as follows:

- Unigrams: 0.043
- Bigrams: 0.131
- Trigrams: 0.136
- Quadrigrams: 0.127

2.2 Question 2

Does re-estimated MLEs for seen ngrams slightly lower than the earlier ones? Are re-estimated MLEs for unseen ngrams slightly higher than the earlier ones (zero)?

- After the application of smoothing on the seen ngrams, the numerator in the MLE increases by 1 and the denominator increases by V (vocabulary size). Since, V can be a large value so the effect of addition of V in the denominator dominates the addition of 1 in the numerator and hence, the re-estimated MLEs for seen ngrams is slightly lower than the earlier ones.

- After applying smoothing, we are essentially adding a one in the numerator and V in the denominator. After doing so, it might seem that the MLE is decreasing. However, for the case of unseen n-grams for the non-smoothed case, we are assigning the value of zero for MLE of that instance. However, this is not the case when smoothing is applied. Smoothing assigns a non zero MLE value since there is an addition of 1 in the numerator along with the count of that n-gram. Thus, this would indicate that the re-estimated MLEs for unseen ngrams is slightly higher than non-smoothed ones (zero).

2.3 Question 3

Compute the perplexity of each tweet in the test dataset and report the average for all the tweets for each of the four smoothed language models? By looking into the perplexity scores, do you think they are meaningful now? Which language model performs best?

The following are the average perplexity of all the tweets for each of the four smoothed language models.

- Unigrams: 312.96
- Bigrams: 236.491
- Trigrams: 578.773
- Quadrigrams: 878.455

Since, the perplexity scores have increased significantly (when compared to their values in Section 1.4), the probability of predicting the correct n-gram decreases in all the four cases indicating that there are a large number of possibilities to choose from, for a given n-gram.

However, if we compare the relative perplexity scores for the smoothed models, the bigram model performs the best as it has the lowest perplexity score.