Indian Institute of Technology Gandhinagar

Project A
Assignment-2 Report
CS 613: NLP

Group Members

Dhakad Bhagat Singh: 20110055

Rahul Kumar: 201101153

Prakram Rathore: 20110141

Sandeep Patel: 20110183

Zeeshan Snehil Bhagat: 20110242

Tejas Parmar: 20110125

Prey Patel: 20110132

Venkata Sriman Narayana Malli: 20110224

Q5. Without Smoothing, Many comments are ending up with infinite perplexities in the test dataset and so the average is also becoming Infinite for all categories i.e Unigram, Bigram, Trigram and Quadgram.

Q7. When the smoothing was not present, we observed that most of the perplexity scores for each n-gram model (unigram, bigram, trigram, quadgram) for most sentences was infinity, which would lead to the mean score of all sentences as infinity. This was due to the fact that in the validation data set, our model encountered new n-grams which were not present in the training data set. So, the probability of all such n-grams came out to be zero, which led to the perplexity being infinity.

However, when we used Laplace Add-1 Smoothing, we found that the perplexity scores for the test set were increasing from unigram to quadgram. This could be due to multiple factors such as the corpus being small, and there being a lot of n-grams which were newly encountered in the test set and were assigned some minimum probability. So this could have caused the creation of new n-grams which were not already in the train set and this phenomenon would have increased with increase in order of n-grams model.

Taking a random example unrelated to our dataset, (have, been) bigram may have been present in train set due to probably being very common but (have, been, customised) trigram and other such trigrams may only be present in test set, so all such trigrams would have been assigned a bare minimum probability, leading to high perplexity scores; if a bigram is not present in test set, all the trigrams formed from it will not be present in test set too, but the reverse is not true; this is applicable for any two n-grams of different n-order values.

Q.8 According to the problem statement, we have chosen 2 other smoothing techniques - 1. Additive Smoothing and 2. Good Turing Smoothing

- **Additive Smoothing (Laplace Smoothing):** It is a technique used to address the problem of zero probabilities for unseen events in language modelling. In the absence of smoothing, we were getting zero probabilities to n-grams (sequences of words) that were not present in the training data. This results in perplexity scores becoming infinite when such unseen n-grams are encountered in the test or validation data.

  When we apply additive smoothing to train the language models, a constant (k) is added to all counts of n-grams during probability estimation. This constant serves to redistribute some probability mass from seen n-grams to unseen ones, which prevents the perplexity scores from reaching infinity. In this assignment, for the 8th task, we took k = 2, and observed the following:
  a. The mean perplexity score for the test dataset increases from Unigram to Biagram (which should not be the case, ideally) and then the mean perplexity score decreases drastically for the Trigram language model.
  b. However, the mean perplexity score went on increasing for the Quad Gram language model.
- In our opinion, this discrepancy might be due to the following reasons:
  - The size of the training corpus may be relatively small, and that is why it becomes more likely to encounter unseen n-grams in the test set.
  - Unseen n-grams in the test set are assigned a minimum probability due to smoothing, which contributes to higher perplexity scores.
  - Smoothing can create new n-grams in the test set that were not present in the training set. This phenomenon becomes more pronounced with higher-order n-gram models.

- **Good Turing Smoothing**: Good-Turing Smoothing is a more advanced smoothing technique that provides a nuanced approach to handling unseen events, particularly rare ones. Unlike Additive Smoothing, which applies a fixed constant to all n-grams, Good-Turing Smoothing adapts to the distribution of observed n-grams with different frequencies.
  When Good-Turing Smoothing is applied, it leverages the concept of "counts of counts" to estimate the probability of unseen events based on the distribution of events that occurred once or a few times. This approach is particularly effective for estimating probabilities for rare and unseen n-grams.
  Good-Turing Smoothing adjusts its smoothing based on the training data, making it well-suited for cases where the training corpus contains various n-gram frequencies. It excels at estimating probabilities for both common and rare events, leading to improved language modelling accuracy.
  Additionally, we have combined interpolation with Good Turing smoothing using unigram probabilities to get better perplexity values for higher n-gram models (trigram and quadgram). In case a particular n-gram sequence is not found, it will

assign a weighted probability value to the sequence so that its probability mass value will not be trivial, hence we get an improved perplexity value. The optimised weight will be determined correctly using fine tuning, but here hit and trial method is used for the same.

Contributions:

Task	Contributing Team Members
1. Data Cleaning and Preprocessing	Rahul Kumar, Prey Patel
2. Train the LMs (Unigram, Bigram, Trigram, and Quadgram) and report the respective perplexity scores (average over all the sentences)	Tejas Parmar, Prakram Rathore
3. Laplace smoothing on the LMs and observation on the change in the perplexity score with and without smoothing.	Zeeshan Snehil Bhagat, Venkata Sriman Narayana Malli
4. 2 other smoothing techniques (Additive and Good Turing) and Training the same n-gram Language Models)	Rahul Kumar, Dhakad Bhagat Singh, Sandeep Patel
5. Report Documentation	Zeeshan Snehil Bhagat, Dhakad Bhagat Singh, Sandeep Patel