Indian Institute of Technology Gandhinagar

Assignment 2
Language Modeling

CS 613: NLP

Group T10: Exquisiters

Aditi Dey (20110007)

Anavart Pandya (20110016)

Ayush Gupta (20110031)

Mann Kumar Jain (20110108)

Pinki Kumari(22210028)

Ronak Kalra (20110171)

Sakshi Jain (20110181)

Vedang Chavan (20110222)

Key Notes:

The comments underwent a series of preprocessing steps to prepare them for training the Language Model. First, they were tokenized into individual sentences. Next, various transformations were applied, including the removal of URLs, punctuation marks and the conversion of all words to lowercase. Additionally, only words containing alphabetic characters were retained, while empty sentences were removed. Finally, the dataset was divided into an 80:20 split for training and testing the model.

Further, in the n-gram function, for no smoothing, there are certain cases where the probabilities must be zero. However, while calculating perplexities, this causes division by zero error. So, we have replaced the zero probability with a very small epsilon, $\varepsilon = 10^{-15}$ in order to avoid division by zero error.

It is important to note that this means that theoretically, for the given training and testing data, the perplexities for all n-grams, with no smoothing, must be infinity, represented by very high values of perplexities here, and to compare between different n-gram models.

Results:

1. On Training Data (Without Smoothing):

N-Gram	**Perplexity (**Rounded to nearest Integer**)**
Unigram	1238
Bigram	77
Trigram	9
Quadgram	3

2. On Testing Data:

N-Gram	**Perplexity (**Rounded to nearest Integer**)**			
	Without Smoothing	Laplace Smoothing	Add-K Smoothing	Good Turing
Unigram	1e11	1288	1090 (For k=9.61)	954
Bigram	4e11	1926	480 (For k=0.001)	25900
Trigram	5e11	4859	881 (For k=2.3e-5)	64541
Quadgram	27e11	5467	1753 (For k=5e-6)	90194

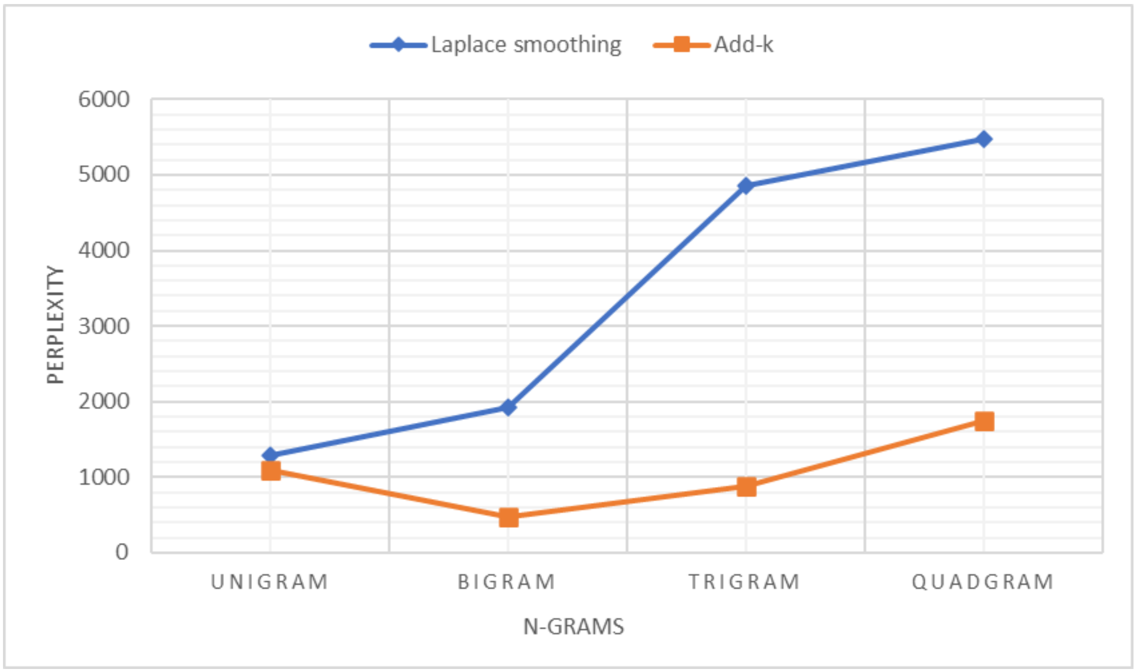

Fig 1 *: Perplexity for n-grams for different smoothing techniques*

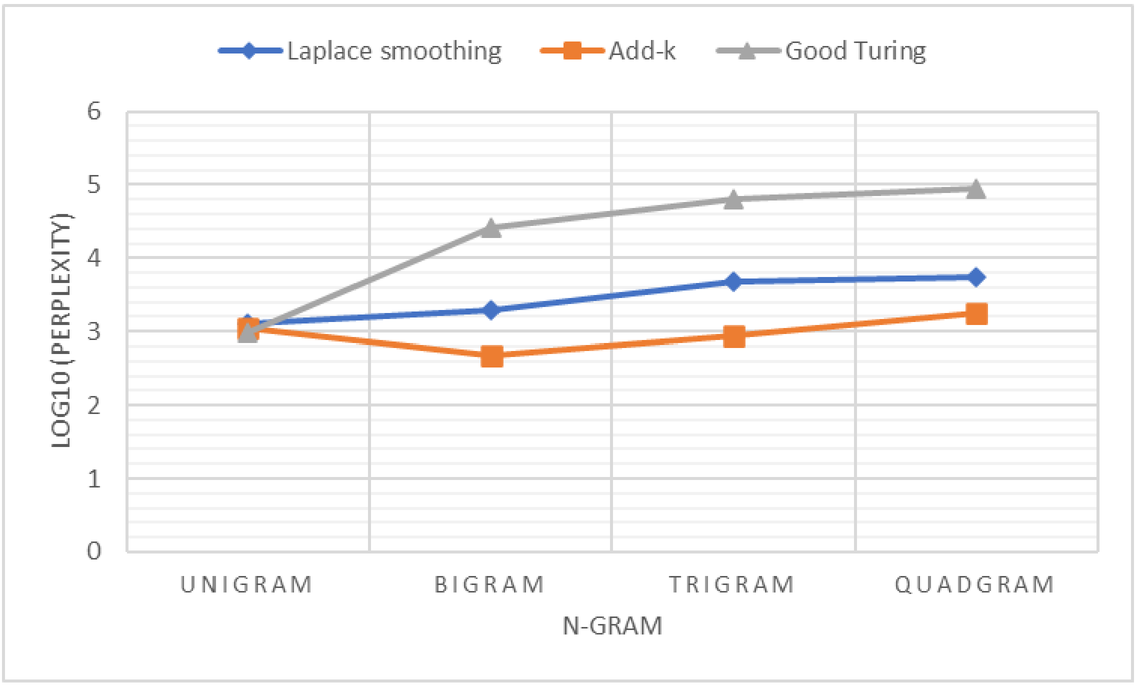

Fig 2 *: log_{10} of Perplexity for n-grams with different smoothing techniques*

Observations:

- The perplexity without smoothing on training data on increasing the value of N follows the trend: Unigram > Bigram > Trigram > Quadgram. As the corpus size is very small, the same trend is not observed in the test data because most of the words in the test data are not present in the train data. This means that there are several unknown words in the testing data which have never been seen by the trained ngrams and hence won't be able to predict them. For the test data, the perplexities follow the trend: Unigram < Bigram < Trigram < Quadgram.
- The perplexity with smoothing on testing data is observed to be very high, approaching infinity.
- We observed that on smoothing, the perplexity of the model on testing data decreases, and model efficiency increases significantly.

  Perplexity Trend:

  Unigram (without smoothing) > Unigram (with Smoothing)
  Bigram (without smoothing) > Bigram (with Smoothing)
  Trigram (without smoothing) > Trigram (with Smoothing)
  Quadgram (without smoothing) > Quadgram (with Smoothing)
- With **Laplace** smoothing, even though the average perplexities for the models decrease from their initial values (without smoothing), they do not follow the expected trend as they do on the training set. This is possible because of the fact that there are a lot of unseen/new words in the test corpus that were not present in the training corpus and have not been learned by the n-gram models.
- With **Add-K** smoothing, the k-values were optimized such that the average overall perplexities for all the models attend a local minimum. This gave different values of k for each model, and the following trend for perplexities was observed: Bigram < Trigram < Unigram < Quadgram.
- Using **Good-Turing** smoothing, we noticed high perplexities. They were better than no-smoothing models but not as effective as add-k smoothing or Laplace smoothing. The main issue was the dataset size; it wasn't large enough for Good-Turing to function optimally. Many frequency counts (N_c) were zero, resulting in many null values. Even though we applied interpolation with Good-Turing, it wasn't effective due to these null values. As we increased the value of n in the n-gram model, the perplexity went up because of more null values. The perplexity trend was as follows: Quadgram > Trigram > Bigram > Unigram.

Name	Contribution
Mann Kumar Jain	Implemented N-gram Language Models and Perplexity function
Vedang Chavan	Implemented N-gram Language Models and Perplexity function
Ronak Kalra	Implemented N-gram Language Models and Perplexity function
Ayush Gupta	Add-k Smoothing and optimized k, Generated results and Documentation
Sakshi Jain	Data Preprocessing, Implemented Add-k Smoothing, Observations
Anavart Pandya	Implemented Good Turing Smoothing and verified Language Models & Perplexity function
Pinki Kumari	Verified and Implemented Good Turing code
Aditi Dey	Nil