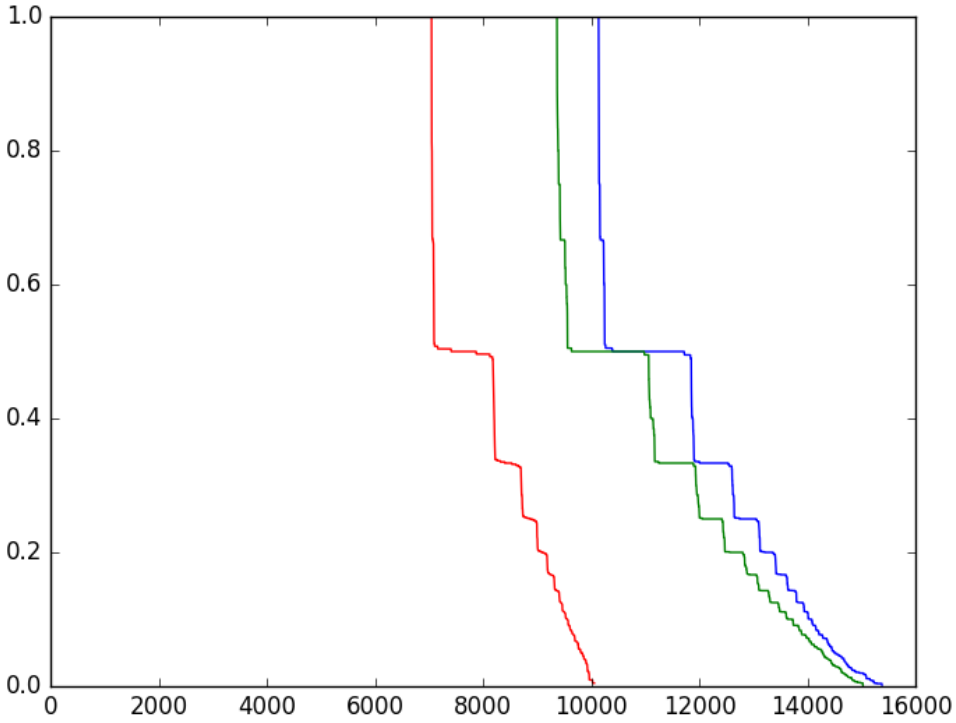
# Assignment1 Report

## Tokenization:

I built a generic tokenizer but defined some special regex expressions for tokens of the form "http://x.com/id" and some few other tokens.

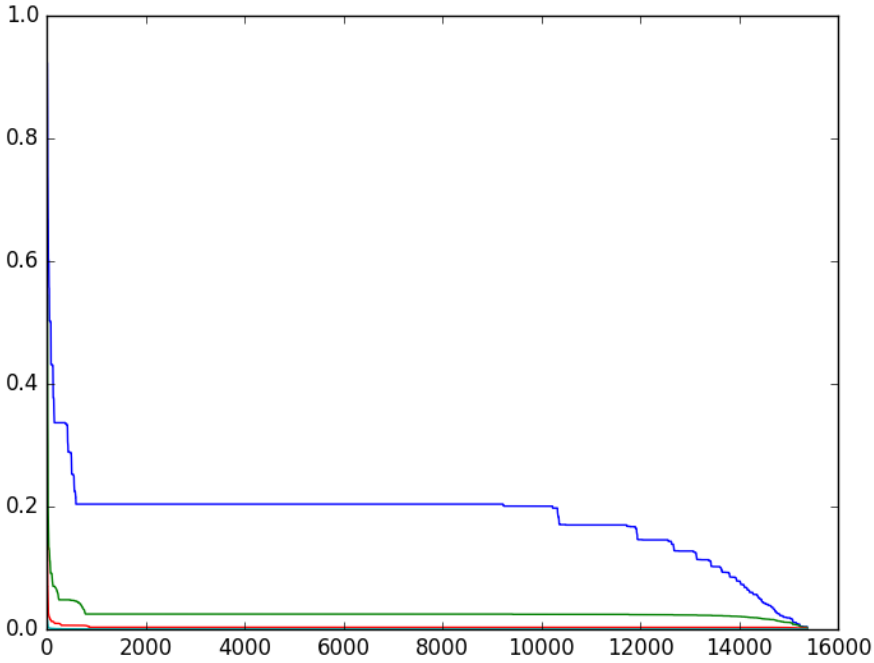
# Language Modelling:

## Combined zipf curve for all the sources:

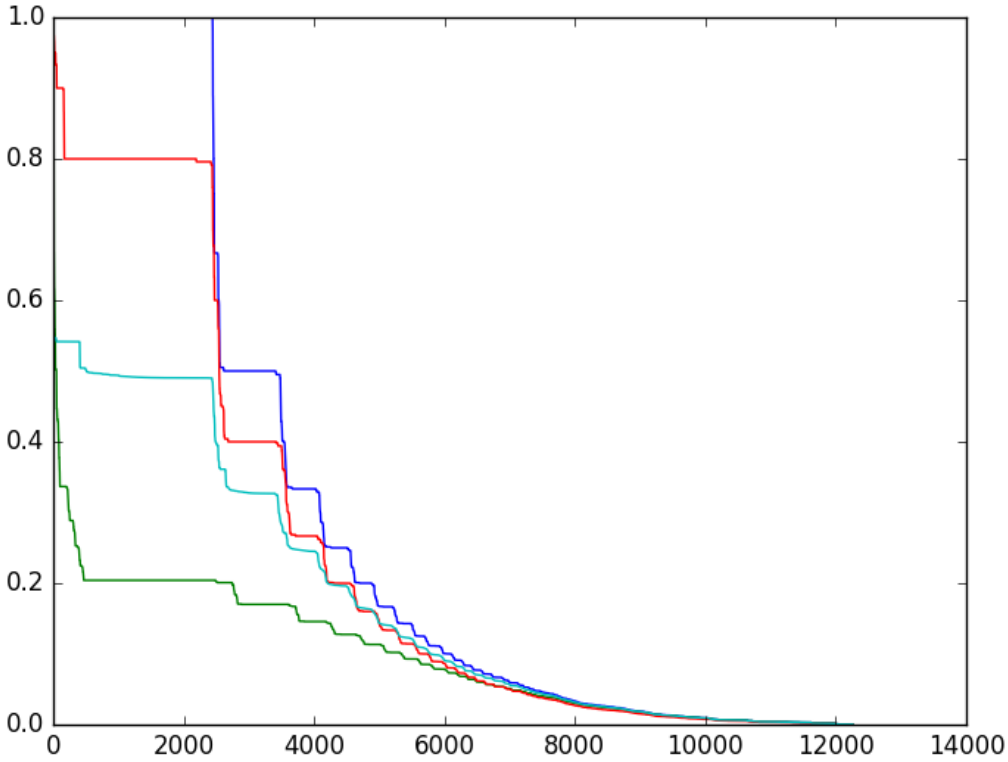



**Laplace Smoothing** for different values of V=200,2000,current size of the vocabulary, 10 times the current size.

Observations: As the value of V is increased from 200 to 10*(current vocab_size) the smoothing algorithm behaves very poorly because the it discounts a lot.

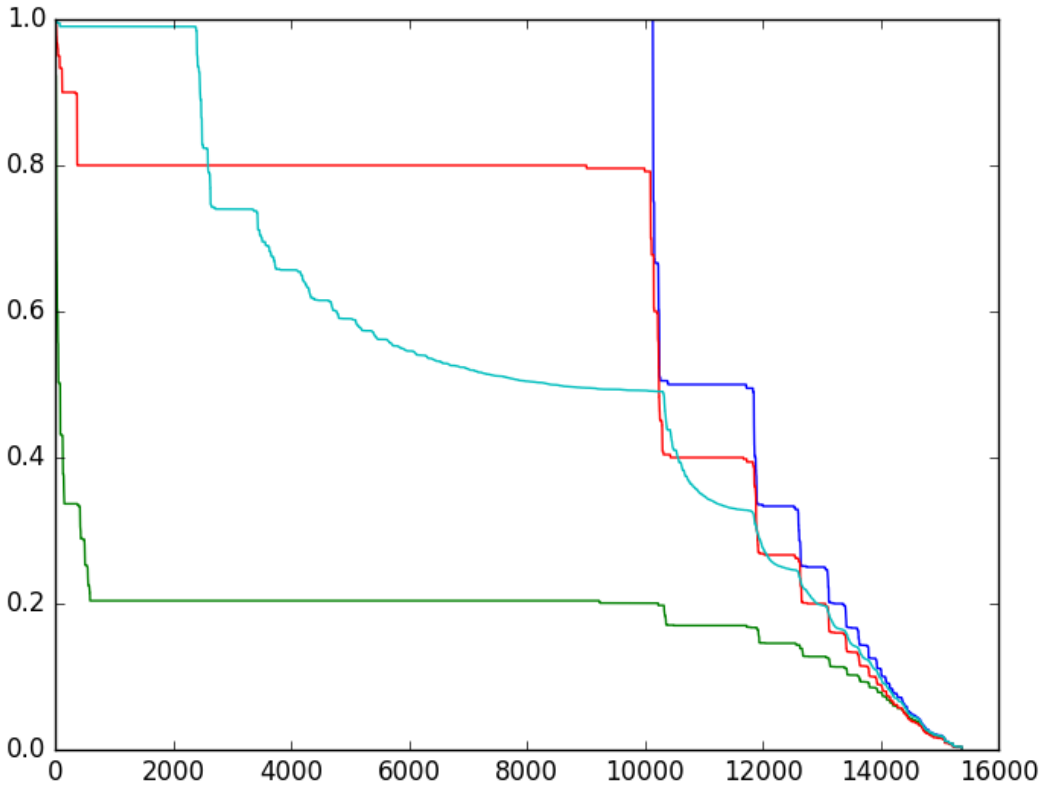



**Bigram zipf curve** for comparing the various smoothing algorithms :
MLE probabilities(blue line),KN(red line), WittenBell(lighht green) and laplace(green).




**Trigram zipf curve** for comparing the various smoothing algorithms :

MLE probabilities(blue line),KN(red line), WittenBell(lighht green) and laplace(green).




**Observations:**
1) Maximum Likelihood Probabilities can't handle the case of unseen events. So have to use some smoothing techniques.
2)The probability mass assigned to the unseen ngrams is very high resulting into huge discounted probabilities.
3)Witten-Bell Smoothing is more conservative when subtracting probability mass and gives good probability estimates.
4)Kneser-Ney discounting augments absolute discounting with a more sophisticated way to handle the backoff distribution using linear interpolation. Its the best smoothing algorithm as evident from the plot.

# Text generation using Kneser Ney probablities:

Successfully Generated text of 15 words using kneser ney probabilities. Handled the case of infinite loop, when one word which is already being used generates the same sequence.

# Tokenization as a supervised problem:

Problem with the tokenizers is that they are often rule-based, hard to maintain, hard to adapt to new domains and new languages. So we can consider tokenization as a machine learning classification problem.
Method used: IOB Tagging
Labelling each character in the text with one of the four tags:
1) I: Inside a token
2)O: Outside a token

3)T: begining of token
4)S: beginning of the first token of a sentence

**IOB tagging example**:
sentence= “hey guys ! i wanted”
This can be annotated as following:
SIIOTIIIOTOTOTIIIII
Now, we can apply naive bayes to predict the class(‘S’,’I’,’O’,’T’) given the character in the text.