**For trigram character language model:**

As per the results in 3.2 and 3.3

For 3.2-Linear Interpolation

Lowest Perplexity-(English file)- 6.690650019611275

Highest Perplexity-(French File)- 28.25918181729213

For 3.3-ADD lambda Smoothing

Lowest Perplexity-(English file)- 8.44405289129217

Highest Perplexity-(French File)- 45.192504787986906

**Linear Interpolation Pros:**

* Linear Interpolation takes into account all trigrams, bigrams, and unigrams.

Pˆ(wn|wn2wn1) = lambda1\*P(wn|wn2wn1) +lambda2\*P(wn|wn1) +lambda3\*P(wn)

* Also, lambdas are calculated in order to maximize probability, thereby lowering perplexity.

Results above clearly shows that Linear Interpolation Trigram Character Language Model is better than ADD lambda Smoothing Model.

* Unlike ADD lambda probability mass is not shifted towards unseen ngrams. Also, probability of frequent ngrams is not underestimated and probability of rare ngrams is not overestimated.

**Linear interpolation Cons:**

* It is slightly complicated as it involves extra step for calculation of lambdas to maximize probability.

**ADD lambda Cons:**

* On the other hand, Add lambda smoothing takes just bigrams and trigrams into account. This results in low probability and high perplexity. Results above clearly shows that Linear Interpolation Trigram Character Language Model is better than ADD lambda Smoothing Model.

For NLP applications that are very sparse, ADD lambda smoothing (lambda=0.1, in current case) actually gives far too much of the probability space to unseen events, 0.1 is added to numerator even if particular ngram is not present in training model.

* Too much probability mass is shifted towards unseen ngrams.
* Probability of frequent n-grams is underestimated.
* On the other hand probability of rare or unseen ngrams is overestimated. All unseen ngrams are smoothed in same way.
* It is worse at predicting the actual probabilities of bigrams with zero counts than other methods.

**ADD Lambda pros**

* It is a very simple technique.