HW5

Authors: Srinivas Suri, Ilayda Ipek Onur

Part 1:

The word ‘First’ appears at the starting of the context in the text file which is used to train the ngram model. As there’s no smoothing for the N-Gram model as of now, the letter that has the highest probability after the empty context is F. Therefore, we observe F as the first word generated in every n gram model. As we increase the order of n for the ngram models, the models tries to behave more like the actual shakespeare novel and will end up producing First as the first word.Following the same argument, after n > 5 , it will always produce first as the first word. Moreover, the word ‘First’ has high frequency in the text file. Therefore , the model learns the probabilities in a way that the probability of generating i after F, generating r after i and so on ...is very high.

Part 2 Analysis:

Perplexity Analysis:

We have trained the Model on the shakespeare_input.txt file. We have calculated the perplexity on Shakespeare and NYTimes text files provided.The results are as follows:

On shakespeare_sonnets.txt:

	n=0	n=1	n=2	n=3	n=4	n=5	n=6	n=7
k=0	23	inf	inf	inf	inf	inf	inf	inf
k=1	23	12	7.9	6.1	6.4	8.9	13.9	21.6

On nytimes_article.txt:

	n=0	n=1	n=2	n=3	n=4	n=5	n=6	n=7
k=0	inf	inf	inf	inf	inf	inf	inf	inf

k=1	29	14.9	11	9.8	10.8	15.5	23.9	33.99

Our observations:

- We observe that using k=0 has inf for various k=0 models. This is because , even if one n-gram is not seen, then the models calculates its perplexity as inf.
- On the other hand, adding smoothing improves the perplexity as smoothing also takes into account n-grams that weren't seen before.
- As NY times might have different n-grams than a shakespeare text file. We consistently observe that the perplexity of the model is greater on NYTimes text file than on the Shakespeare file.
- As n increases, the perplexity on NY Times increases significantly more than Shakespeare because having n=7 for n-gram models is almost equal to the original training set. So, we might not see many 7-grams that existed in Shakespeare in NY times which results in a bad perplexity value.

Interpolation Analysis:

We have used n=4,k=1 as the basis for our analysis and countries dataset to understand how weights and smoothing effect the model performance. We used n=4,k=1 as this proved to be the best classifier in part 3 and we wanted to analyse how the performance varied for different weight ratios and different k-smoothing values. The weights array is illustrated through this example: P(abcde) = w[0] * P(e | abcd) + w[1] * P(e | bcd) + w[2] * P(e | cd) + w[3] * P (e | d) + w[4] * P(e). In short, lower index in w[] array corresponds to assigned weight of higher n-gram order.

S.no	k=0	k=1	k=2
w=[.75,.25,0,0,0]	0.266	0.623	0.601
w=[0.5,0.5,0,0,0]	0.268	0.645	0.618
w=[0.25,0.25,0.25,0.25,0]	0.646	**0.692**	0.678
w=[0.2,0.2,0.2,0.2,.2]	0.661	**0.692**	0.679
w=[0.125,0.125,0.25,0.25,0.25]	0.672	0.682	0.674

w=[0,0,0,.5,.5]	0.627	0.63	0.627

Our Observation are as follows:

- Assigning equal weights perform better than skewing the weights towards higher n-gram or the lower n-gram orders. Assigning higher weights to higher order n-grams has the problem of curse of dimensionality and assigning higher weights to lower order n-grams means that the model doesn’t generalise well to one specific dataset. Therefore, balanced weights tend to perform better than other weight distributions.
- Without smoothing (k=0), we can observe that the models with higher weights on the higher order n-grams (w=[.75,.25,0,0,0] for example) tend to perform significantly worse than other models. This is due to the curse of dimensionality.

Part 3 Analysis:

We have trained the corpus on two different N-Gram models. In this report, we have presented the averaged-accuracy of the validation set over all the different countries. We have calculated the Average Accuracy as follows: For each n,k; we have computed trained the models on the training set for each country individually and then averaged the accuracy of each model (predicted on its respective country) on the validation data.

Our results and the trained models are as follows:

1. **Interpolated Model** with **Equal weights:**

Table for Average-Accuracy for Validation Set for different n(ngrams),k(smoothing) values

	k=0	k=1	k=2	k=3	k=4
n=0	.491	.491	.491	.491	.491
n=1	.627	.630	.627	.627	.631
n=2	.660	.672	.673	.670	.671
n=3	.673	.682	.681	.678	.667
n=4	.661	**.692**	.679	.664	.653

We got a .702 accuracy in the Leaderboard using n=4,k=1 and Interpolated Model with equal weights.

2. For **NGramModel** without Interpolation**:**
   Table for Accuracy for Validation Set for different n(ngrams),k(smoothing) values

	k=0	k=1	k=2	k=3	k=4
n=0	.491	.491	.491	.491	.491
n=1	.640	.647	.642	.643	.641
n=2	.512	.658	.659	**.659**	.652
n=3	.255	.656	.646	.616	.598
n=4	.172	.596	.565	.531	.512

Our Observations:

- It is observed that N-Gram model with Interpolation with k-smoothing performed better than the N-Gram model with k-smoothing.
- It's interesting to note that trigrams performed than bigrams on an average than bigrams and 4-grams performed better than trigrams on average accuracy.
- As we increase the smoothing value, the accuracy decreases. This makes sense because we are giving more weightage to the unseen contexts or characters as we increase the smoothing value k