A Report on N-Gram Model

Viral Shah (2024201014)

March 27, 2025

1 Analysis

1.1 Results Table

The following tables summarize the key metrics obtained for different values of n across three datasets.

Table 1: General English Dataset

n	Let	Tab	Avg Let/Word	Avg Tabs/Word
2	315	357	0.30	1.57
3	286	341	0.28	1.51
7	246	342	0.24	1.50
10	248	337	0.24	1.48

Table 2: Topic Specific Dataset

n	Let	Tab	Avg Let/Word	Avg Tabs/Word
2	254	403	0.24	1.77
3	234	334	0.22	1.46
7	192	326	0.18	1.43
10	190	339	0.18	1.49

Table 3: Part I of Topic Specific Dataset

n	Letter	Tab	Avg Let/Word	Avg Tabs/Word
2	219	428	0.21	1.88
3	204	362	0.20	1.59
7	183	302	0.18	1.33
10	181	311	0.17	1.36

2 Analysis

2.1 Corpus Size : How does the corpus size and the content of the corpus affect the model's predictions?

There are 3 corpuses available to train 1) General English corpus 2) Topic Specific corpus 3) parts of Topic specific corpus.

2.1.1 1) Training on General English corpus :

General English corpus contains 30000 lines of context which is based on English specific dictionary words, which contains almost all English words

Pros : Training our model on General English corpus will help us to ensure to get the intended word in suggestions panel, as almost all words are present in General English corpus , We can test and analyze the model's performance well of our auto-complete implementation using this corpus.

Cons : As we are training the dataset on general content and dataset is also bigger, so we will get most of the words in suggestion panel, so if we compare the model's performance by Avg tab per word (And taking the word from suggestion panel only if it's found in suggestion panel which's the way I am assuming) then we might have to press more tabs than other datasets to reach our intended word, as there'll be lots of words available related to the word that we are typing in suggestion panel.

2.1.2 2) Training on Topic specific corpus :

Topic specific corpus contains around 30000 lines combinedly of all parts which is based on the text content.txt's context.

Pros : As we are testing on the context of the topic specific corpus only, here also we will get all words in suggestion panel, so we reach our intended aim to use our implemented model of ngram auto-complete and typing lesser words comparatively

Cons : As the model contains words around a single topic only, the model would not be able to generalize well for other datasets that contains words from different topics and general English, we might not get many suggestions in suggestion panel if the word is completely newer to training datasets, it might generate unsensible words (As of my implementation) but the user intended use of auto-complete would not be very fruitful here.

2.1.3 3) Training on Specific part of Topic specific corpus :

A part of Topic specific corpus will always have a subset of words of whole topics and would be sized around 3000-4000 lines according to the user's roll number

Pros : Likewise same as above, but here comparatively lesser words are there and if we get lucky and get the testing paragraph of which almost all words are there in our part i of topic specific dataset, then we would be having almost all suggestions in suggestion panel and we might have to hover less to reach our intended word in suggestion panel, as the size is around 10 percent from the previous dataset.

Cons : If the test paragraph does not contain many words of the part i of the training dataset, then user have to type almost all whole words and the auto complete model would not be very helpful here and we ultimately won't reach our aim, also letters typed and average letters per word will be higher compare to previous models.

2.2 Model Type: Does the amount of context provided (number of character tokens) while predicting affect the results?

2.2.1 Case of n = 2 and n = 3 :

As the value of n is smaller, we would not be caring about the context of the previous word as we are maximally going 2 word back only, so here it will find the next word by probability of most occurring word after the last n - 1 words.

Pros : computational cost will be lower , time spent to get suggestions will be lower comparatively as we are not going much backwards.

Cons : we will mostly get random words only in suggestion panel , as we are not caring about the context of the word here , which will not be very effective to us unless we put the condition that the word should only come in suggestion panel iff it is present in dataset (which I've put)

2.2.2 Case of n = 7 (my chosen value) and n = 10 :

As the value of n is larger comparatively, the context of the past words will play a significant role in suggesting the next word

Pros : As here as above mentioned, now we are caring about the context , so we will get more accurate words that we intend in suggestion panel and we can check effectiveness of the auto - complete model more precisely by choosing the word from suggestion panel , most likely random words will be on the lesser side and accurate English words with sorted (Decreasingly) order of probability will come in the suggestion panel.

Cons : As we are going much backwards, computational cost and probability calculation will take decent amount of time , so efficiency around time will be comparatively lower , avg word typed will be decreased and avg tab pressed will be increased.

2.3 Metrics : Comment on the metric scores defined above and what they tell us about the performance of the model

Total Letters typed : The total letters we are typing to get whole word , It can be from 1 character upto n characters also (n = size of the current word that we intend to have)

Total tabs pressed : The total number of types we hover around suggestion panel to get our intended word, it can be 0 to t (t = maximum suggestions available in suggestion panel)

Average letters typed :The average is total letters type / total words we had , generalise the letters that we have to type in approx manner among all words.

Average tabs pressed :The average is total tabs pressed / total words we had , generalise the tabs that we have to press in approax manner to reach our intended word amonng all words

These parameters suggest us, that to predict the best model , the average letters typed and average tabs pressed should be lowest possible. why? **as the lower value we have in both i.e average letters typed and average tabs pressed, it would be directly proportional to the efficiency of the model, as with the help of the model, the letters we have to type decreases also the tabs we have to press also decreases which means less human power is wasted, that is the model is performing the best of the all models.**

2.4 Generalization Test

[Own paragraph Video]

I selected a random 100-word paragraph from Google containing the most common English words to evaluate the model's performance on unseen data.

Metric	Value
Letters Typed	253
Tabs Pressed	257
Average Letters/Word	0.30
Average Tabs/Word	1.38

Table 4: Generalization Test Results

The paragraph contained classical English words, which allowed the model to benefit from the training on the general English dataset. The results were quite satisfactory when tested on this random and previously unseen data set.

Key observations:

- The model successfully predicted words with reasonable accuracy
- Performance metrics demonstrate the effectiveness of the general English corpus training
- The auto-completion system showed promising generalization capabilities

3 Best Model Selection

[Test paragraph Video]

After carefully considering all the previous analyses and the performance metrics defined in section 2.3, I declare the following as the best working model:

- Context Size: $n = 7$
- Corpus: Part i-th (9th) of topic-specific dataset

Metric	Value
Average Letters Typed	0.18
Average Tabs Pressed	1.33

Table 5: Best Model Performance Metrics

The selection is based on the lowest average letters typed and average tabs pressed, which directly correlate with the model's efficiency in auto-completion.