Vakya Autocomplete

Kunal Mehra
Indiana University Bloomington
kumehra@iu.edu

Abstract

Autocomplete is a feature which predicts what would be the next most probable word based on what we have written so far. Hinglish (Hindi-English) is a language which is quickly gaining popularity. The words are written in English language but they mean something in the Hindi language. It is primarily spoken in the Indian subcontinent.

1 Introduction

Autocompletion is a popular feature implemented in many applications today like search engines such as Google, Bing and Elasticsearch, writing emails, typing in text editors like MS Word, code editors or IDEs when writing source code. It

Assume that you start typing in a web search, the system will return a list of suggestions based on what you have typed so far. As you keep editing or adding then suggestions vary. These suggestions are certain extended variations of our input created by the system.

For example, in the figure 1 below we typed “online” and we got back the list of suggestions such as “online text compare”, “online text editor”, etc.

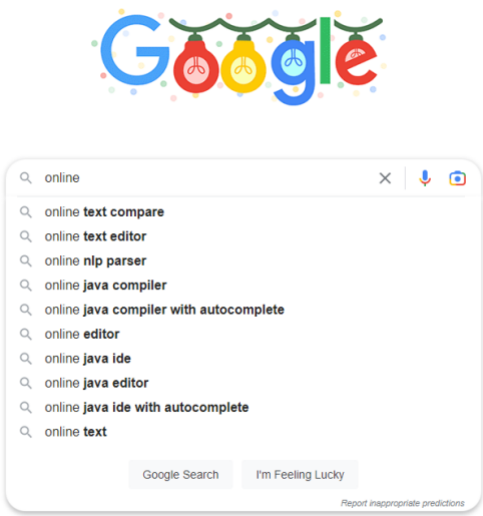

Fig 1. Autocomplete feature in Web Search Engine

Similarly in figure 2 we can see the implementation of autocomplete feature in an IDE.

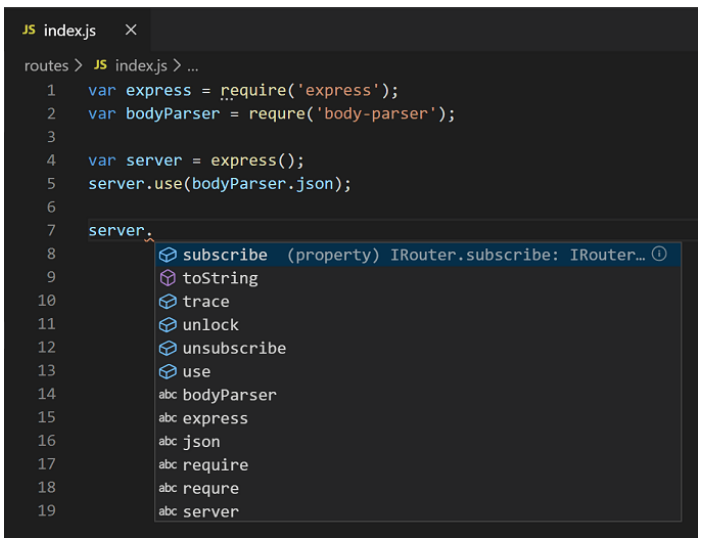

Fig 2. Autocomplete feature in an IDE

To better understand the language chosen for corpus let us consider a few examples:

“Tum kaise ho?” - This sentence means “How are you?”

“Kidhar ja rhe ho?” - This sentence means “Where are you going?”

Previously N-gram models have been used to predict the next word mostly in English language but we aim to extend this autocomplete feature to work for Hinglish language as well.

2 What we plan to do

We plan to use N-gram model to derive the probabilities of the next word based on the previous ones and return the result which is a combination of multiple suggestions.

3 Prediction Methods

3.1 Model

A statistical language model is a probability distribution over sequences of words or characters. Given such a sequence, say of length m, it assigns a probability P(w1….wm) to the whole sequence where w1 represents the first word or character in the sequence and wm represents the mth i.e., the last word or character in the sequence.

N-gram models are Statistical (Probabilistic) Language models. Any N-gram is just a sequence of "n" words or characters.

For example, “Hello” is a unigram as it has just 1 word, “Hello world” is a bigram, and so on.

As mentioned, our task is to compute the probability of a word ‘w’ given some history ‘h’ i.e., P(w|h). We can say that ‘h’ here means all the previous words in the sentence.

A way to calculate this probability is to use relative frequency counts i.e., how many times did the word ‘w’ come up after the history ‘h’.

$$P(w|h) = C(w|h)$$

Now, the probability of a sequence of words w1,w2,...wn can be represented by the following formula:

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$

$$P(w_{1:n}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

The implemented n-gram prediction algorithm assumes that one can predict the next word in a phrase based on the previous n-1 words which is known as Markov approximation. Thus, the probability of the occurrence of a word depends only on the previous words which is called Markov assumption. Therefore, the above equation can be written as:

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Bigram model is also known as first order Markov model, Trigram as second order Markov model, Quadrigram as third order Markov model and so on. Hence, the model is known to be an n-1 Markov model which uses the previous n-1 words for the prediction of current word.

3.2 Estimation

To estimate such probabilities, we use Maximum Likelihood Estimation (MLE). An MLE estimate for the parameters of an n-gram model can be obtained by getting counts from a corpus and normalizing the counts so that they lie between 0 and 1.

We use the following formula:

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n)}{C(w_{n-N+1:n-1})}$$ ---------- (1)

For example, let us assume we have a bigram consisting of words ‘a’ and ‘b’ and we want to calculate the probability P(b|a). Then using MLE we can say that:

$$P(b|a) = MLE(a,b) = \frac{Count(a,b)}{Count(b)}$$

In case we come across a word which isnt present in the corpus we are using then our denominator would become 0 since the count for that word

would be 0. Thereby, making our probability as invalid. To avoid this, we add a positive variable 'k' called as K-Smoothing factor to the numerator and 'k.|V|' to the denominator where |V| is the number of words in our vocabulary. This will ensure that for any word that isnt present in our corpus/vocabulary has a probability of $\frac{1}{|V|}$.

We can update Eq. (1) after adding the Smoothing factor to:

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n) + k}{C(w_{n-N+1:n-1} + k|V|)}$$

4 Results

Sentence in corpus: "kar sakte hai."

```
previous_tokens = ['kar','sakte','hai']
```

Expected outcome:

"Kar sakte hai."

Actual outcome:

```
[('.', 0.03466204506065858),
 ('.', 0.007366482504604052),
 ('.', 0.007366482504604052),
 ('jab', 0.0018552875695732839)]
```

5 Conclusion

In this paper we used N-gram models to calculate the frequencies of the words existing in our corpus and thereby finding the most probable words depending upon if we were using Unigrams, Bigrams, Trigrams, Quadgrams and Quintgrams.

We observed that giving more than 5-grams the output doesn't improve.

6 Future Work

We can improve the accuracy of the prediction by using more advanced methods and expanding our corpus.

References

1) https://towardsdatascience.com/index-48563e4c1572
2) https://www.iosrjournals.org/iosr-jce/papers/conf.15010/Volume%202/28.%2073-82.pdf
3) Predicting Sentences using N-Gram Language Models: https://www.researchgate.net/publication/220817263_Predicting_Sentences_using_N-Gram_Language_Models
4) Examining Autocompletion as a Basic Concept for Interaction with Generative AI - https://arxiv.org/pdf/2201.06892.pdf

Dataset

There aren't readily available datasets available due to the fluid nature of the language as people use different spellings for the same word so we need to either create our own from scratch or modify the existing ones. For our work we have used the dataset_0 file from the following link and added our own sentences to it as well- https://www.kaggle.com/datasets/thanatoz/hinglish-blogs