**EXPERIMENT NO.5**

**AIM:**

N-gram model.

**RESOURCES REQUIRED:**

Python 3, NLTK toolkit, Text editor, 4 GB RAM and above, i5 processor and above

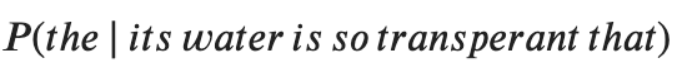
**THEORY:**

**N-gram Model:**

Statistical language models, in its essence, are the type of models that assign probabilities to the sequences of words. In this article, we’ll understand the simplest model that assigns probabilities to sentences and sequences of words, the n-gram

You can think of an N-gram as the sequence of N words, by that notion, a 2-gram (or bigram) is a two-word sequence of words like “please turn”, “turn your”, or ”your homework”, and a 3-gram (or trigram) is a three-word sequence of words like “please turn your”, or “turn your homework”.

Let’s start with equation P(w|h), the probability of word w, given some history, h. For example,



Here,

w = The

h = its water is so transparent that

And, one way to estimate the above probability function is through the relative frequency count approach, where you would take a substantially large corpus, count the number of times you see its water is so transparent that, and then count the number of times it is followed by the. In other words, you are answering the question:

Out of the times you saw the history h, how many times did the word w follow it

Image

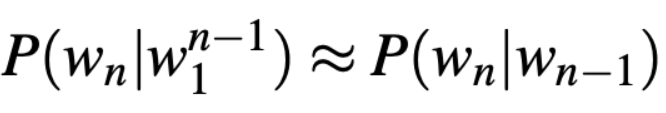
Now, you can imagine it is not feasible to perform this over an entire corpus; especially if it is of a significant size.

This shortcoming and ways to decompose the probability function using the chain rule serves as the base intuition of the N-gram model. Here, you, instead of computing probability using the entire corpus, would approximate it by just a few historical words.

**The Bigram Model:**

As the name suggests, the bigram model approximates the probability of a word given all the previous words by using only the conditional probability of one preceding word. In other words, you approximate it with the probability: P(the | that)

And so, when you use a bigram model to predict the conditional probability of the next word, you are thus making the following approximation:



This assumption that the probability of a word depends only on the previous word is also known as the Markov assumption.

Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far in the past.

You can further generalize the bigram model to the trigram model which looks two words into the past and can thus be further generalized to the N-gram model.

Now that we understand the underlying base for N-gram models, you’d think, how can we estimate the probability function? One of the most straightforward and intuitive ways to do so is Maximum Likelihood Estimation (MLE)

For example, to compute a particular bigram probability of a word y given a previous word x, you can determine the count of the bigram C(xy) and normalize it by the sum of all the bigrams that share the same first-word x.

There are, of course, challenges, as with every modeling approach, and estimation method. Let’s look at the key ones affecting the N-gram model, as well as the use of MLE

**Sensitivity to the training corpus**

The N-gram model, like many statistical models, is significantly dependent on the training corpus. As a result, the probabilities often encode particular facts about a given training corpus. Besides, the performance of the N-gram model varies with the change in the value of N.

Moreover, you may have a language task in which you know all the words that can occur, and hence we know the vocabulary size V in advance. The closed vocabulary assumption assumes there are no unknown words, which is unlikely in practical scenarios.

**Smoothing**

A notable problem with the MLE approach is sparse data. Meaning, any N-gram that appeared a sufficient number of times might have a reasonable estimate for its probability. But because any corpus is limited, some perfectly acceptable English word sequences are bound to be missing from it.

As a result of it, the N-gram matrix for any training corpus is bound to have a substantial number of cases of putative “zero probability N-grams”

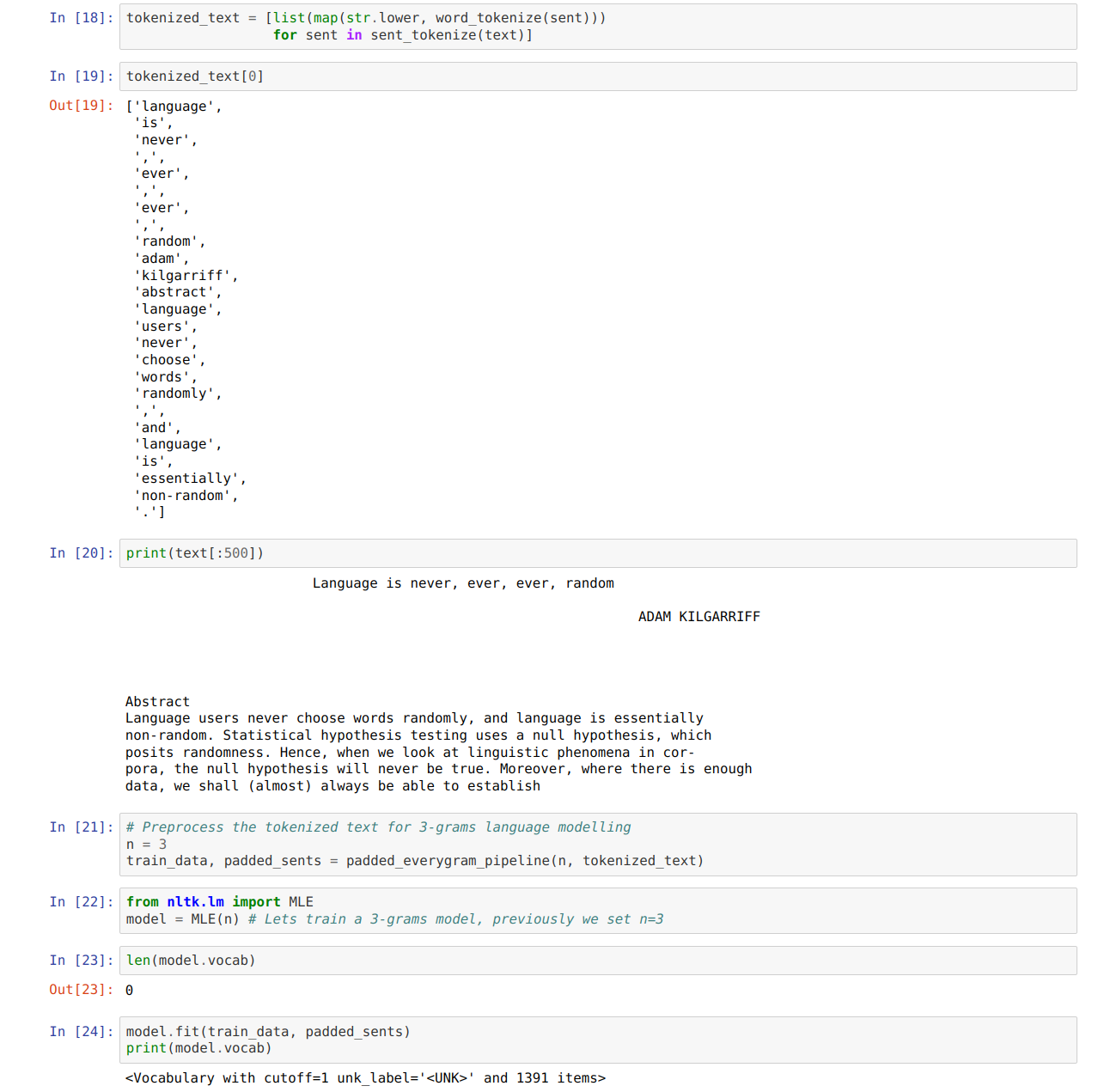
**CONCLUSION:**

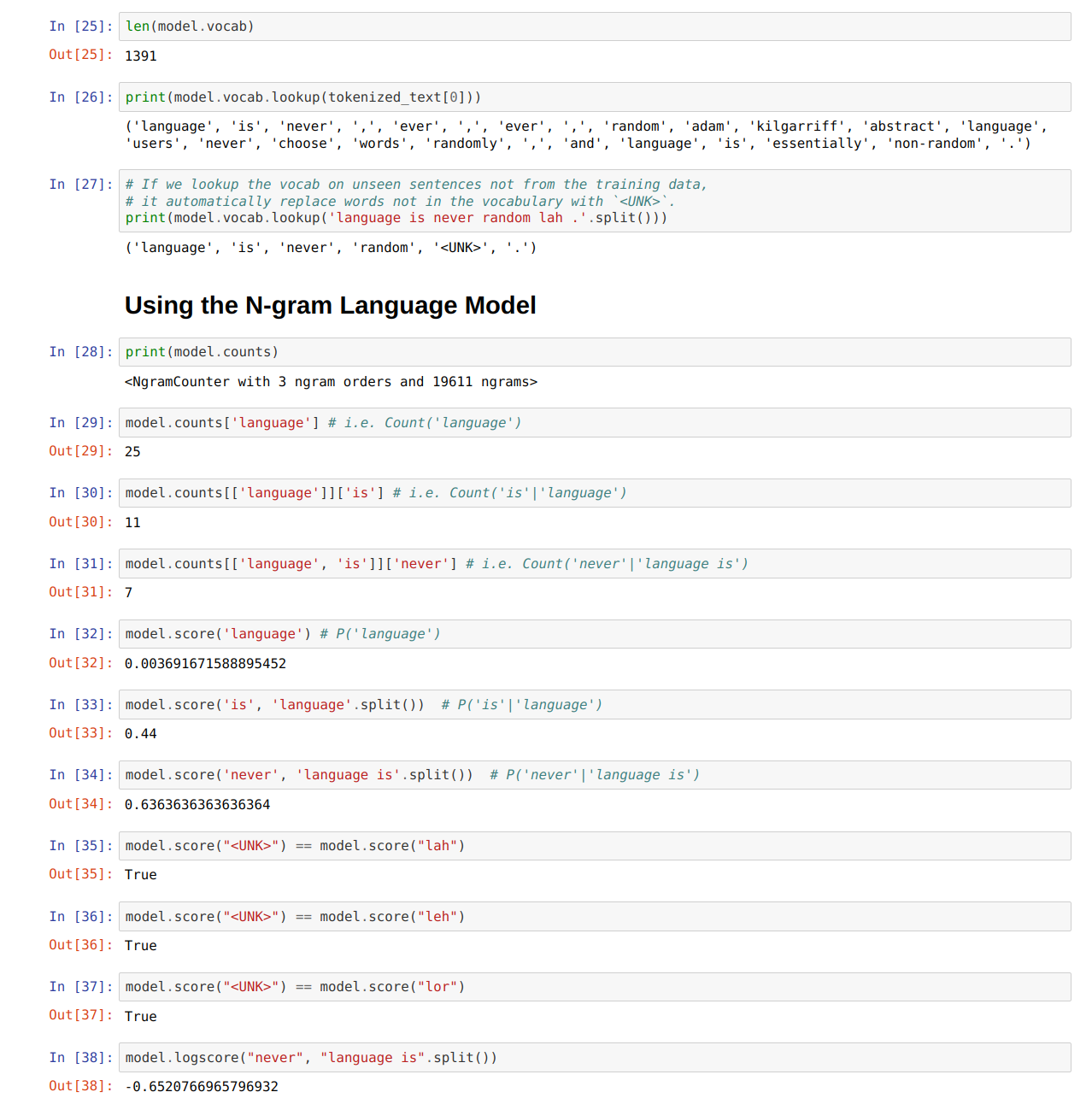
The N-gram model is a statistical language model that has various applications in natural language processing such as spell correction etc. The bi-gram model specifically has been studied in detail and has been implemented.

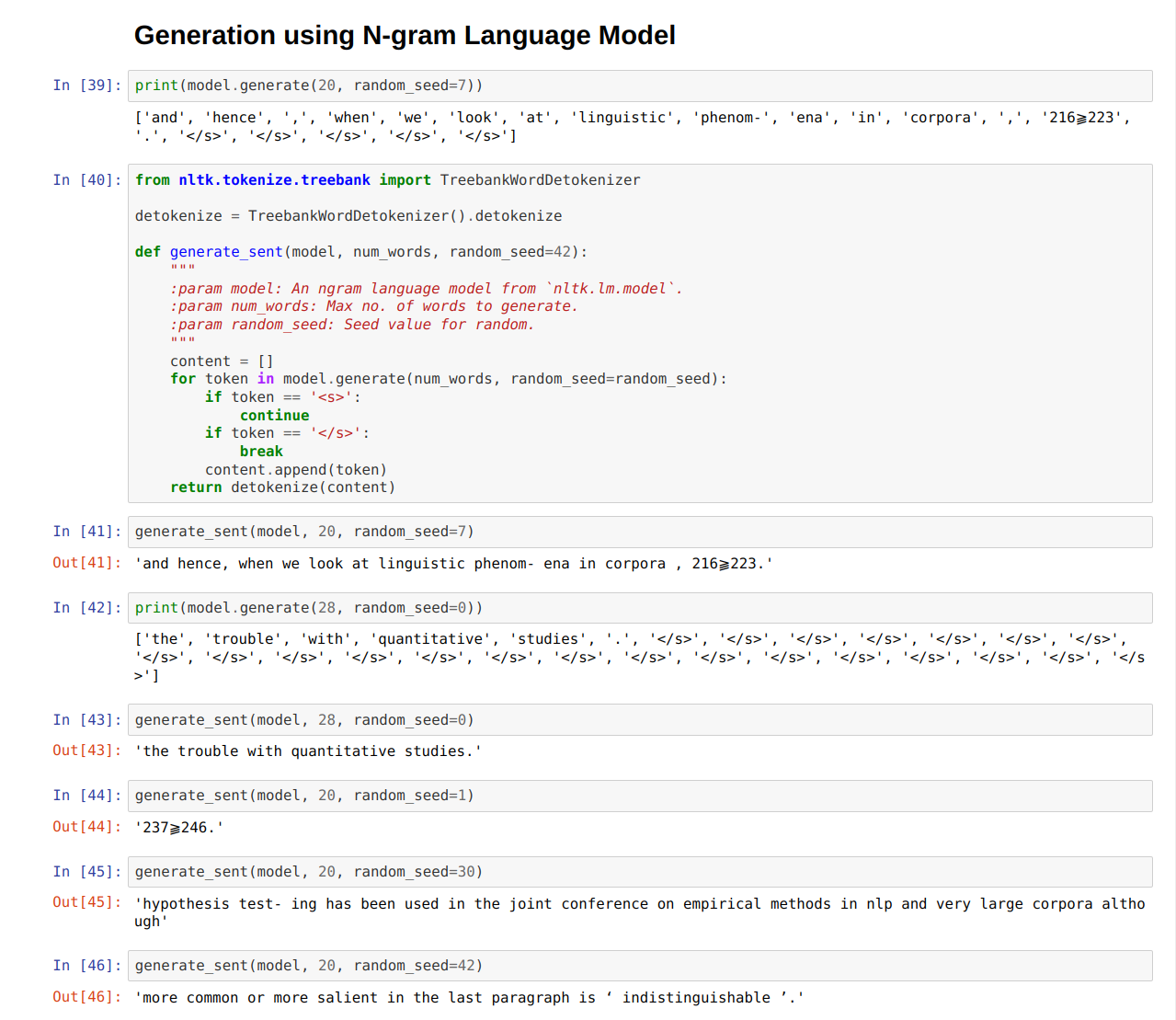
**CODE and OUTPUT:**

****

****

****

****

****