**N-grams in NLP**

N-grams, a fundamental concept in NLP, play a pivotal role in capturing patterns and relationships within a sequence of words. In this blog post, we’ll delve into the world of N-grams, exploring their significance, applications, and how they contribute to enhancing language processing tasks.

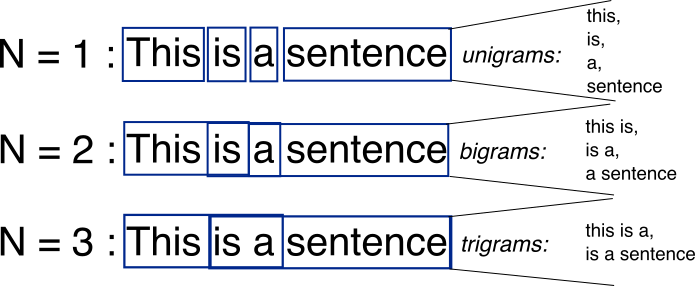
# Understanding N-grams:

# Definition:

N-grams are contiguous sequences of ’n’ items, typically words in the context of NLP. These items can be characters, words, or even syllables, depending on the granularity desired. The value of ’n’ determines the order of the N-gram.

# Examples:

* **Unigrams (1-grams):** Single words, e.g., “cat,” “dog.”
* **Bigrams (2-grams):**Pairs of consecutive words, e.g., “natural language,” “deep learning.”
* **Trigrams (3-grams):** Triplets of consecutive words, e.g., “machine learning model,” “data science approach.”
* 4-grams, 5-grams, etc.: Sequences of four, five, or more consecutive words.



# Significance of N-grams in NLP:

# 1. Capturing Context and Semantics:

* N-grams help capture the contextual information and semantics within a sequence of words, providing a more nuanced understanding of language.

# 2. Improving Language Models:

* In language modeling tasks, N-grams contribute to building more accurate and context-aware models, enhancing the performance of applications such as machine translation and speech recognition.

# 3. Enhancing Text Prediction:

* N-grams are essential for predictive text applications, aiding in the prediction of the next word or sequence of words based on the context provided by the preceding N-gram.

# 4. Information Retrieval:

* In information retrieval tasks, N-grams assist in matching and ranking documents based on the relevance of N-gram patterns.

# 5. Feature Extraction:

* N-grams serve as powerful features in text classification and sentiment analysis, capturing meaningful patterns that contribute to the characterization of different classes or sentiments.

# Applications of N-grams in NLP:

# 1. Speech Recognition:

* N-grams play a crucial role in modeling and recognizing spoken language patterns, improving the accuracy of speech recognition systems.

# 2. Machine Translation:

* In machine translation, N-grams contribute to understanding and translating phrases within a broader context, enhancing the overall translation quality.

# 3. Predictive Text Input:

* Predictive text input on keyboards and mobile devices relies on N-grams to suggest the next word based on the context of the input sequence.

# 4. Named Entity Recognition (NER):

* N-grams aid in identifying and extracting named entities from text, such as names of people, organizations, and locations.

# 5. Search Engine Algorithms:

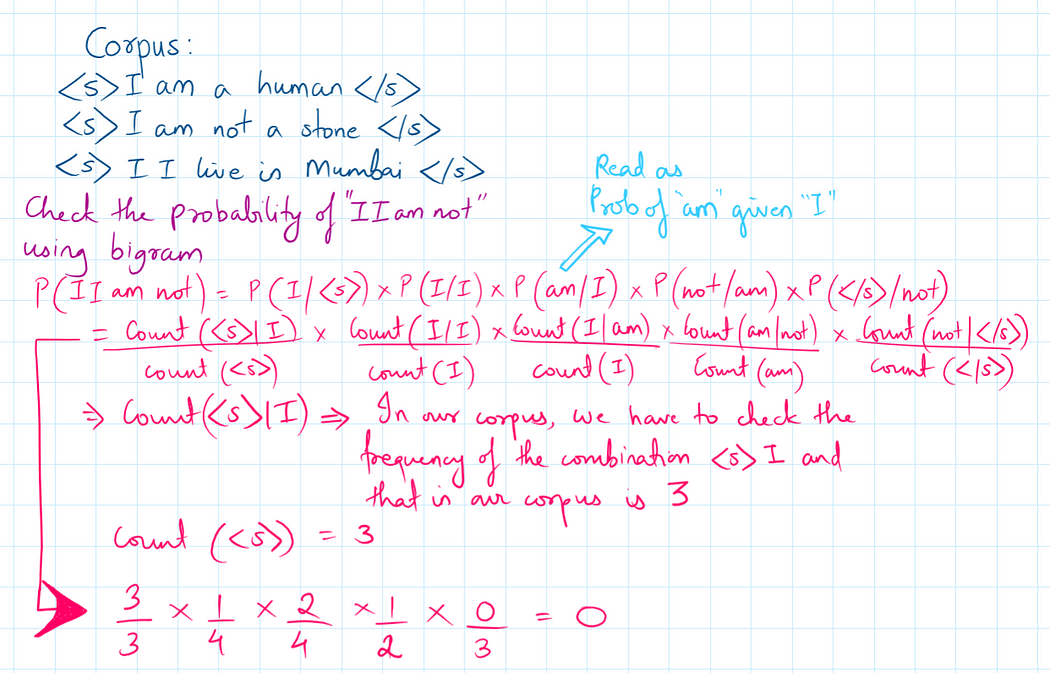
* Search engines use N-grams to index and retrieve relevant documents based on user queries, improving the accuracy of search results.

# CODE

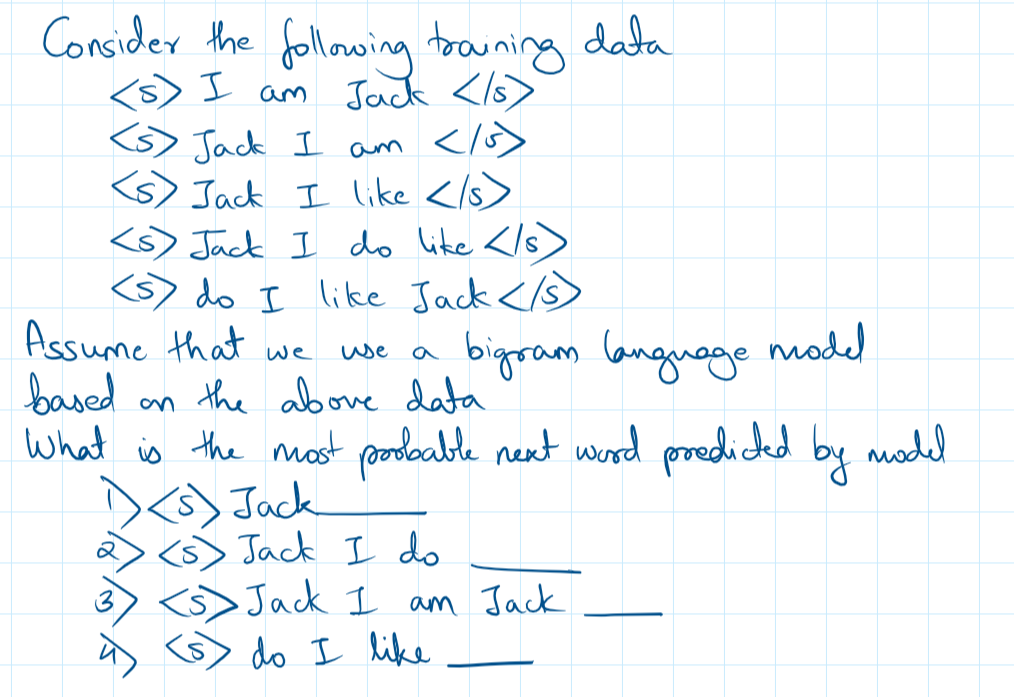
import nltk  
nltk.download('punkt')  
  
from nltk import ngrams  
from nltk.tokenize import word\_tokenize  
  
# Example sentence  
sentence = "N-grams enhance language processing tasks."  
  
# Tokenize the sentence  
tokens = word\_tokenize(sentence)  
  
# Generate bigrams  
bigrams = list(ngrams(tokens, 2))  
  
# Generate trigrams  
trigrams = list(ngrams(tokens, 3))  
  
# Print the results  
print("Bigrams:", bigrams)  
print("Trigrams:", trigrams)  
  
'''  
Output:  
Bigrams: [('N-grams', 'enhance'), ('enhance', 'language'), ('language', 'processing'), ('processing', 'tasks'), ('tasks', '.')]  
Trigrams: [('N-grams', 'enhance', 'language'), ('enhance', 'language', 'processing'), ('language', 'processing', 'tasks'), ('processing', 'tasks', '.')]  
'''

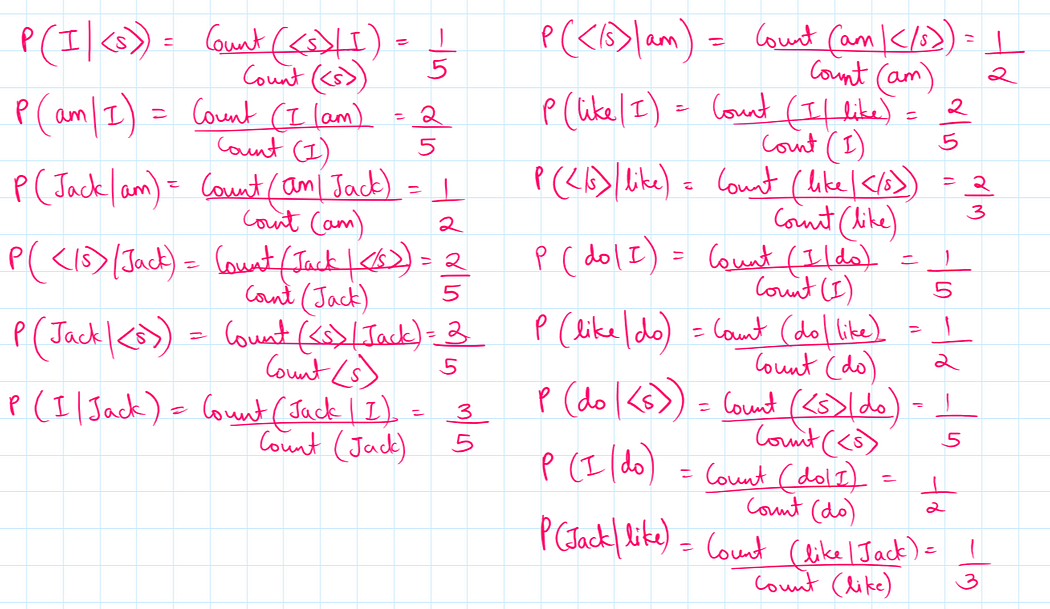
# Using N Gram to predict the probability of a sentence

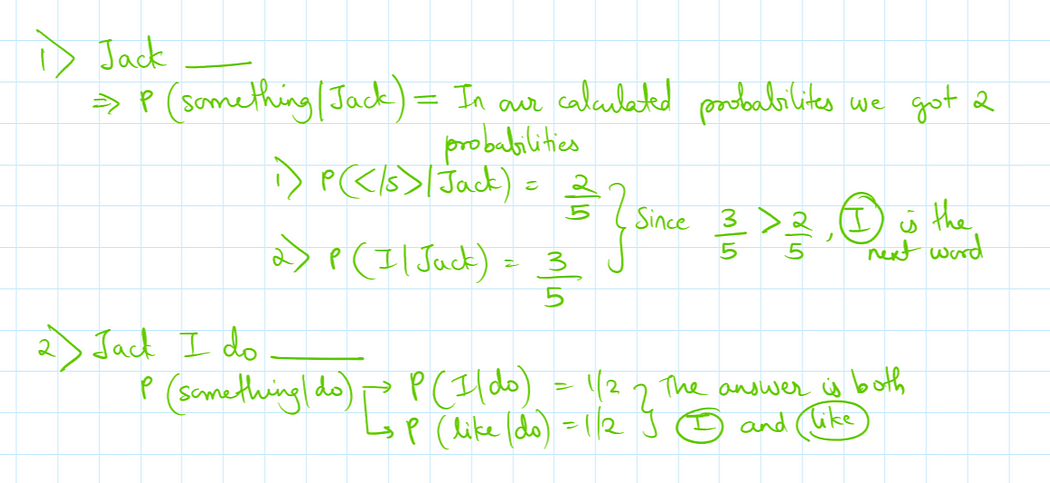
For every sentence, we put <s> and </s> at the beginning and the end respectively. This denote the start and the end of the sentence.



# Using N grams to predict the next word in the sentence







# How are n grams used for word embeddings

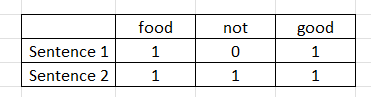
Lets take 2 sentences

Sentence 1 : The food is good

Sentence 2 : The food is not good

Now when the text preprocessing is done on these 2 sentences, the words ‘the’ and ‘is’ will be removed. Our vocabulary will contain the following words food , not , good

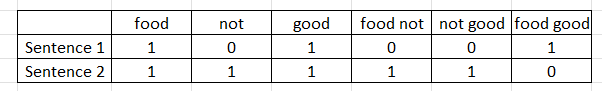
Now if these sentences are to be converted into one hot encoding, they will look like this



Now as we can see that these 2 sentences are completely opposite sentences, however their vector representation isnt very different. The vector representation here has failed to capture the semantic meaning of the 2 sentences. We can improve this by using n grams

Now lets mix bigrams along with unigrams that is, now our vocab will contain words : food, not, good, food not, not good, food good.

Now if we do represent our 2 sentences using this vocab then it will look like this:



Now with the help of n grams here, we are able to improve the semantic meaning of the sentences.

We can similarly do various combinations of (unigrams, bigrams), (unigrams, bigrams, trigrams) etc

By changing one of the parameters in the sklearn library, we can decide what combinations of ngrams do we need to use:

(1,1) -> unigrams

(1,2) -> unigrams and bigrams

(1,3) -> unigrams, bigrams, trigrams

(2,3) -> bigrams, trigrams and so on.