# Report on N-gram Language Model

Harshit Kumar

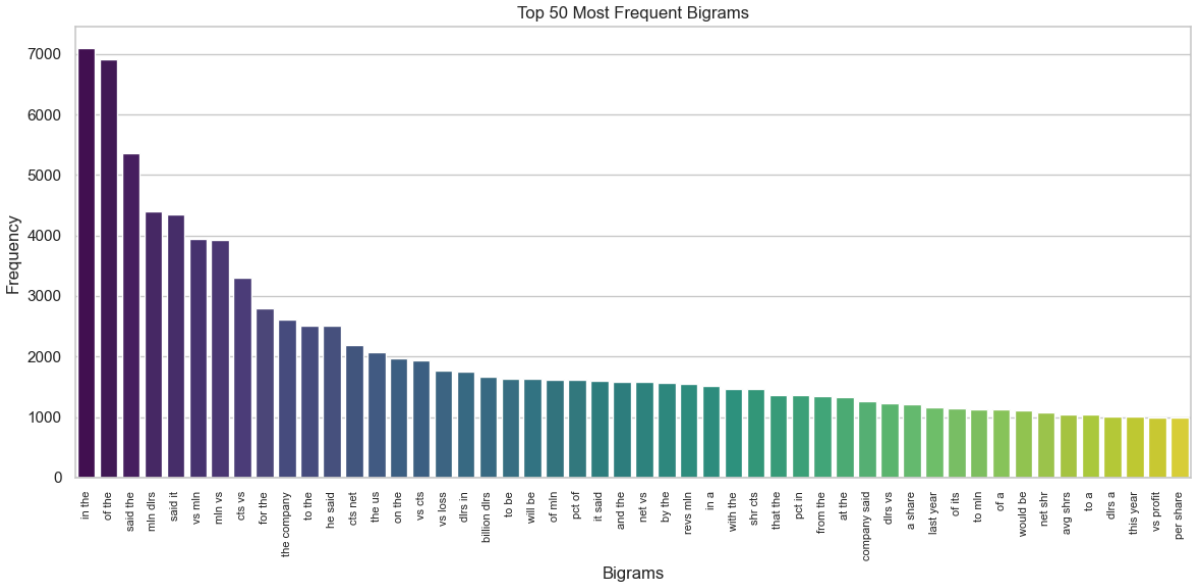
**Objective:** The objective of this analysis is to explore and understand the characteristics of n-gram language models and their applications in predicting and generating text based on the provided text corpus.

**1. Model Overview:** The implemented n-gram language model is designed to handle various values of 'n' (the size of the n-grams). The model includes functionalities for cleaning and preprocessing text, creating n-grams, calculating n-gram frequencies, Laplace smoothing for probability calculation, predicting the next word, generating sentences, and providing insights into n-gram frequencies.

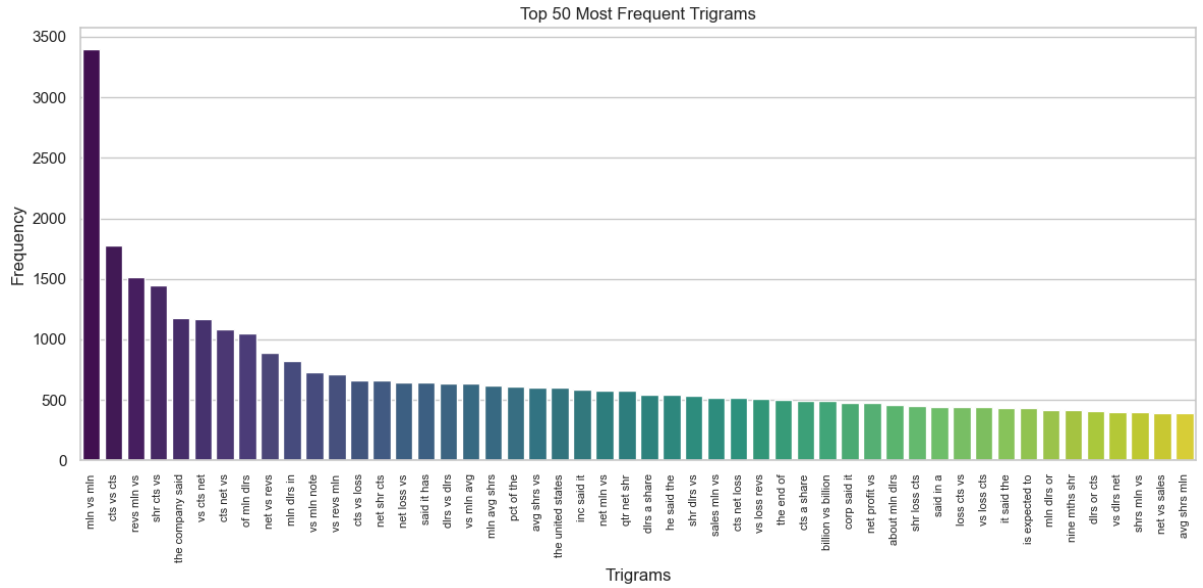
**2. Corpus and Preprocessing:** The model is trained on the **Reuters** corpus, which has been preprocessed to remove special characters and converted to lowercase. The preprocessing ensures a cleaner and standardized input for the language model.

**3. Model Training:** Different instances of the model were created, bigrams (n=2), trigrams (n=3), 4-gram models. The training involved creating n-grams from the tokenized text and calculating the frequencies of these n-grams.

**4. Frequency Analysis:** The model was used to analyze and visualize the top 50 most frequent bigrams and trigrams. The model demonstrated the capability to generate sentences given a prefix. Examples with various prefixes were tested, and the generated sentences showed a coherent use of language based on the learned n-gram probabilities.




Top 50 most frequent bigrams




Top 50 most frequent trigrams

**Individual Word Predictions by Bigram model**
print(bigram_model.predict_next_word(('the',)))  gives company
print(bigram_model.predict_next_word(('of',))) gives the
print(bigram_model.predict_next_word(('blue',))) gives arrow
print(bigram_model.predict_next_word(('their',))) gives own
print(bigram_model.predict_next_word(('time',))) gives of

Observation: The predictions generally make sense, but certain predictions may be influenced by the specific context and vocabulary of the training corpus.

**Generated Sentences with different (n – 1)-gram inputs for bigram model**
Generated sentence with prefix ['for']: the company said the company
Generated sentence with prefix ['for', 'the']: company said the company said
Generated sentence with prefix ['for', 'the', 'man']: said the company said the
Generated sentence with prefix ['for', 'the', 'man', 'he']: said the company said the
Generated sentence with prefix ['for', 'the', 'man', 'he', 'is']: expected to the company said

**Generated Sentences with different (n – 1)-gram inputs for trigram model**
Generated sentence with prefix ['for']: <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN>
Generated sentence with prefix ['for', 'the']: year ended march the company
Generated sentence with prefix ['for', 'the', 'man']: workforce of rank xerox south
Generated sentence with prefix ['for', 'the', 'man', 'he']: <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN>
Generated sentence with prefix ['for', 'the', 'man', 'he', 'is']: not a threat to the

Observation: The bigram model appears to face challenges when generating sentences with certain prefixes, leading to repetitive or unclear outputs. The trigram model doesn't always generate response.

**Sentence generation using different n-gram models of length 10 with the following prefixes**

**Bigram model**
['the', 'use'] -> of the company said the company said the company said
['he', 'said'] -> the company said the company said the company said the

**Trigram model**
['for', 'the', 'company'] -> said the company said the company said the company said
['the', 'united', 'states'] -> and japan to open its markets to us goods the

**4-gram model**
['the', 'united', 'states', 'of'] -> violating a political commitment to halt and reverse trade barriers
['a', 'trade', 'of', 'the'] -> dutch economy requires a stable exchange rate and interest rate
['he', 'said', 'that', 'his'] -> latest proposal represented his final effort if it fails to
['what', 'do', 'you', 'think'] -> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN>
['is', 'this', 'what', 'you'] -> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN> <UNKNOWN>

**5-gram model**
It gives <UNKNOWN> token for almost all the cases.

Observations:

- The model exhibits varying degrees of success in generating coherent sentences based on different prefixes.
- The use of the <UNKNOWN> token in some outputs indicates difficulties in generating contextually relevant content.
- 2-gram 3-gram models seems to work, but they produce a lot of repeating words. 4-gram models work only if provide specific input. 5-gram doesn't work at all.

**6. User Interface:** A simple user interface (UI) using Tkinter was designed to allow users to interact with the model by entering values for 'n', a prefix, and the desired sentence length.