AUTO-COMPLETE SYSTEM

SIMRAN CHOUDHARY (2021205) SARTHAK POL(MT23082)
SAKSHAM LAL(2020402) TANMAY PARASHAR(MT23100)

PROBLEM STATEMENT

To develop a sentence auto-completion system using classical machine learning techniques. This system aims to predict the next word in a given sentence by understanding the context and semantics of the sentence.

Example :

Input - *“The quick brown fox jumps over the lazy”*

Predicted Word - “*dog*”

The challenge is to design the system using classical machine learning approaches to accurately understand and complete such sentences in various contexts.

DATASET DESCRIPTION

Training Dataset:

Size: 39,905 entries

Includes columns such as - activity_label, context (ctx), context_a (ctx_a), context_b (ctx_b), endings, label.

Used to train the machine learning model to understand and predict the next word in a given context.

Validation Dataset:

Size: 10,042 entries

Mirrors the structure of the training dataset.

Purpose: Used for fine-tuning the model and validating its predictions against known outcomes.

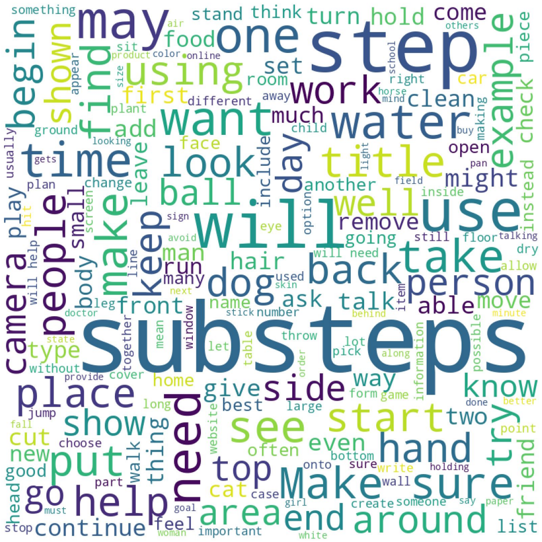
Testing Dataset:

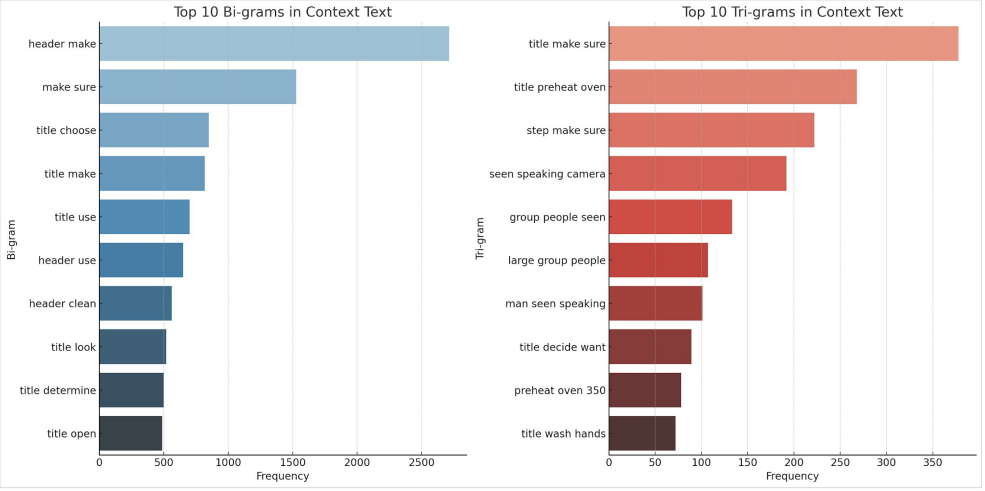
Size: 10,003 entries

Similar to the training dataset, but without the 'label' column.

Purpose: To evaluate the model's performance in predicting the next word without prior knowledge of the correct answer.

Figure : Most Frequently occurring words in endings column





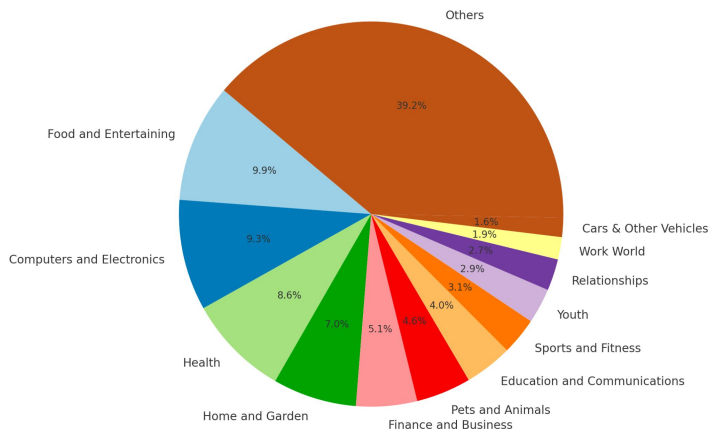
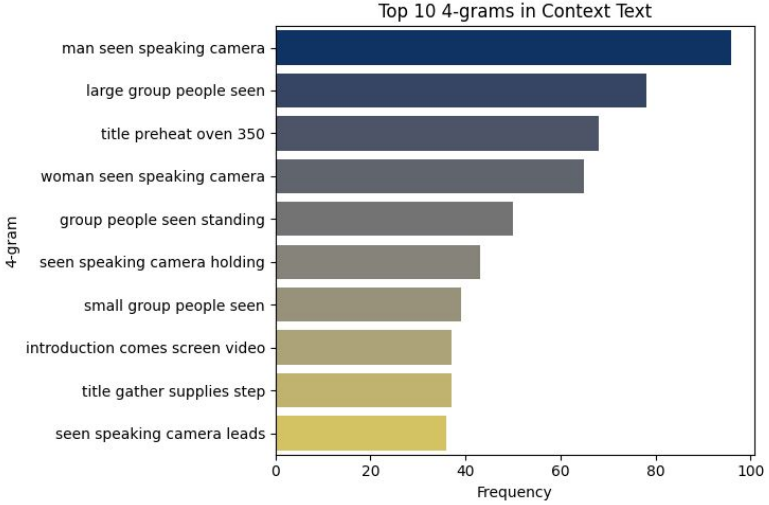
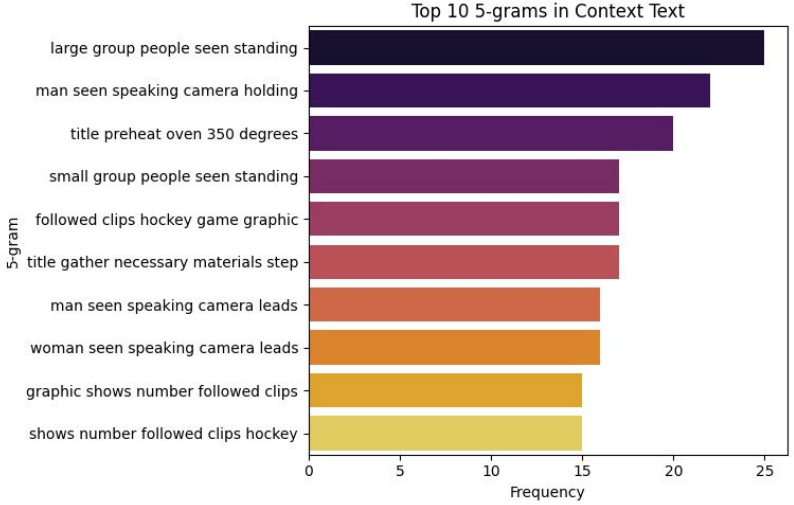


Figure : Different activity labels

EXPLORATORY DATA ANALYSIS





METHODOLOGY USED

We have used **N-Gram models** - Statistical Language models that aim to assign probabilities to a given sequence of words

Our approach uses relative frequency counts to compute the probability, i.e. ,Out of the times we saw the history h, how many times was it followed by the word w.

$$P(word \mid probability\ of\ given\ text) = \frac{C(probability\ of\ given\ next\ word)}{C(probability\ of\ given\ text)}$$

Instead of using the entire corpus, we approximate this probability using just n previous words.

For instance if $w_{1:n}$ represents the sequence of words $w_1 w_2 \ldots w_n$ then using the chain rule of probability we can write

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1}) \sim P(w_{1:n}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

BI-GRAM MODEL

Model which approximates the probability of a word given all the previous words $P(w_n|w_{1:n-1})$ by using only the conditional probability of the preceding word $P(w_n|w_{n-1})$.

Thus we assume that $P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

K-Smoothing is a technique to adjust probabilities in n-gram models to better handle unseen n-grams.It adjusts the probability distribution of n-grams, ensuring that unseen n-grams are not assigned a probability of zero.

It enhances our n-gram model by providing a more robust and realistic probability estimation,although it introduces a simplifying assumption that may not hold for all data types.

RESULTS

Our current auto-complete system employs a bi-gram model, achieving an accuracy of 24.7% in predicting the next word given a sentence on the validation dataset.

However, when compared to more advanced NN models, there's a noticeable gap in performance. Advanced models such as Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and Transformer-based architectures like BERT and GPT significantly outperform bi-grams, with accuracies typically around 60% .

These models excel due to their superior ability in capturing and remembering context over longer sequences, making them particularly effective for complex tasks in autocomplete systems.

CONCLUSION

Text prediction and autocomplete features aid in increasing writing speed by suggesting appropriate words.

Our study confirms that while bi-gram models like ours achieve basic accuracy (24.7%) in autocomplete systems, they are significantly surpassed by advanced neural networks.

Furthermore, we plan to improve the model's overall adaptability by adding common sense knowledge into its framework which would make them more intuitive and effective.