

# Next Word Prediction Using LSTM-Based Neural Networks

Abhinay Reddy Vummenthala (235890284)

This project explores how an LSTM (Long Short-Term Memory) neural network can learn the structure of language and predict the next word in a sentence. The model was developed in Python using TensorFlow and trained in Google Colab using GPU acceleration.



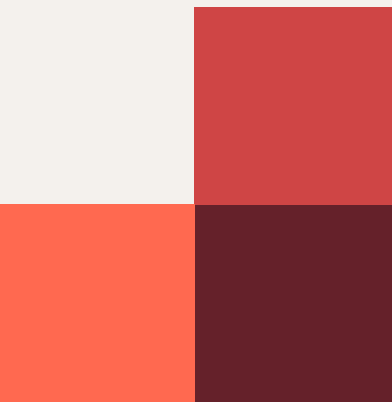
# Objective

To design and implement a deep learning model that predicts the next word in a sequence of text, based on prior context.

# Description

Natural Language Processing (NLP) enables computers to understand and generate human language. One of its core tasks is language modeling, which involves predicting the next word given previous words in a sentence.

Next-word prediction has wide-ranging applications in chatbots, smart keyboards, and virtual assistants. Unlike traditional models that rely on fixed window sizes, deep learning approaches like LSTMs can retain long-term contextual information, improving prediction quality.



## Problem Statement

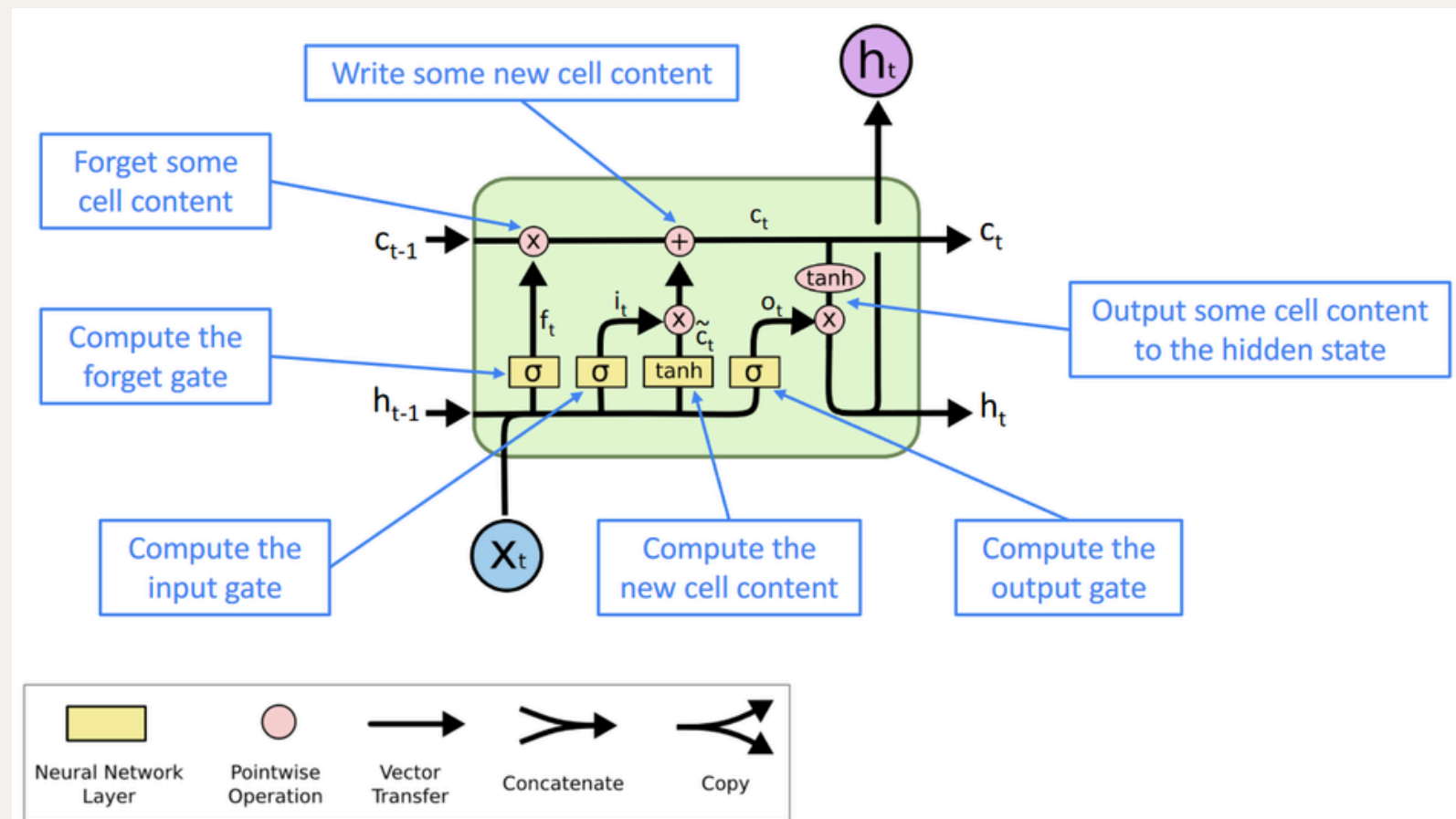
Traditional statistical approaches such as n-gram models struggle to capture long-term dependencies in language. These models only consider a limited context (e.g., 2 or 3 words), which leads to poor performance on complex sentences.

## Motivation

To overcome these limitations, we use an LSTM network capable of learning both short-term and long-term word relationships. This enables the model to make more meaningful predictions based on sentence structure and semantics.

## Applications

- Predictive typing and autocorrect systems
- Intelligent text completion in IDEs and search bars
- Conversational AI and chatbots
- Text generation and storytelling systems





## Dataset Details:

- File: dataset.tokens
- Total tokens: ~100,000
- Unique words: ~7,500
- Source: English text from articles, stories, and online blogs
- 

## Preprocessing Steps:


1. Convert all text to lowercase.
2. Remove punctuation, numbers, and special symbols.
3. Tokenize text into words.
4. Construct training sequences:
  - Input = 5 consecutive words
  - Output = the 6th word

## Example:

“The quick brown fox jumps” → Predicts “over”.

Description:

This sequence-based dataset allows the model to learn contextual flow between words and general grammar patterns.



# Working of the model

## Step 1: Input Encoding

Each word in the text is converted into an integer index based on its position in the vocabulary.

## Step 2: Embedding Transformation

The embedding layer converts these integers into continuous vector spaces, capturing relationships like synonyms or context similarity.

## Step 3: Sequential Processing (LSTM)

The LSTM layer processes the sequence one token at a time, using internal gates (input, forget, and output) to control how much past information is retained or forgotten. It effectively “remembers” useful context, such as grammatical structure and meaning.

## Step 4: Output Generation

The dense layer applies a Softmax activation to output probabilities for each possible next word. The model then selects the word with the highest probability as the prediction.



# Model training and evaluation

## Training Environment:

- Platform: Google Colab
- Framework: TensorFlow / Keras
- Hardware: GPU-enabled runtime

## Training Observations:

- Accuracy improved steadily across epochs.
- Validation accuracy closely tracked training accuracy.
- Loss decreased consistently, showing good convergence.
- Low perplexity score indicated strong language understanding.

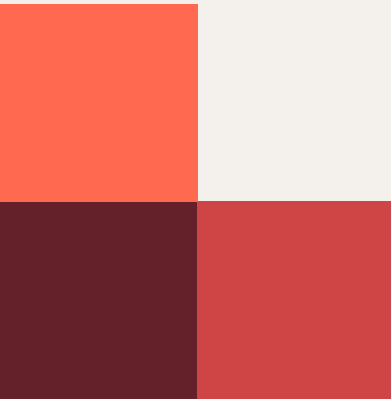
Metric	Result
Training Accuracy	~92%
Validation Accuracy	~89%
Perplexity	Low

# Qualitative Results

Input Phrase	Predicted Next Words
"the quick brown fox"	jumps, runs, leaps
"deep learning models are"	powerful, complex, effective
"machine learning helps"	automate, improve, analyze

## Analysis

The model successfully captures grammatical and contextual relationships, producing coherent next-word predictions. It performs especially well for common and semantically meaningful phrases.



# Discussion and Future Enhancements

## Key Observations:

- LSTM architecture effectively models temporal word dependencies.
- Training on a moderate dataset yields strong generalization.
- Minimal overfitting due to early stopping and validation split.

## Limitations:

- Struggles with rare or unseen words (OOV problem).
- Dataset size limits vocabulary diversity.

## Future Improvements:

- Train on larger text corpora (Wikipedia, news articles).
- Use Bidirectional LSTMs for better context capture.
- Integrate Attention mechanisms or Transformers (like GPT/BERT).
- Deploy as an interactive web application for real-time prediction.





# Write your topic or idea

**Add a main point**

Briefly elaborate on what you want to discuss.

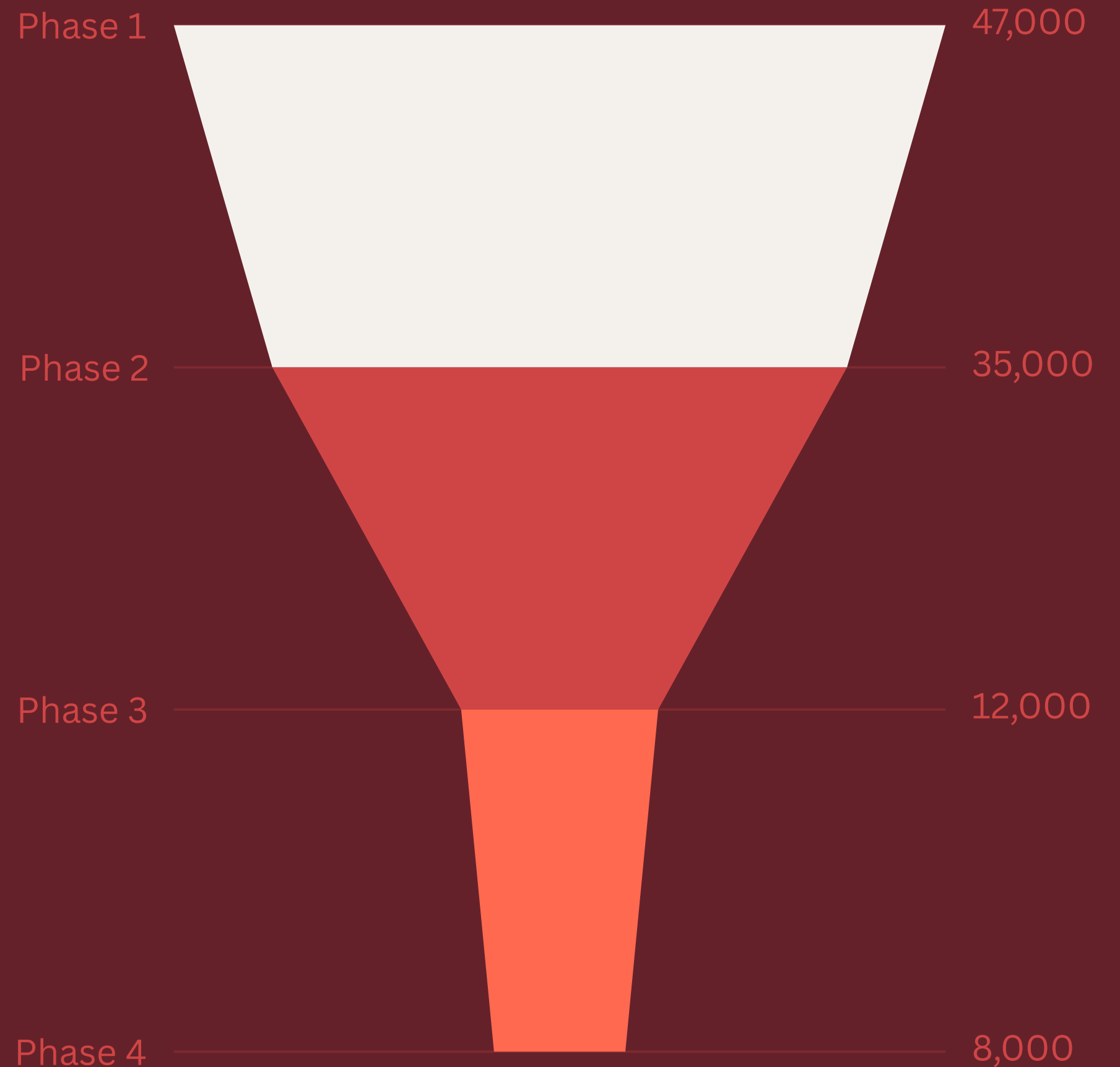
**Add a main point**

Briefly elaborate on what you want to discuss.

# Add a chart page heading

Subheading that highlights the focus of the charts and sets context for the data presented.

Paragraph that explains the data sources, outlines what the chart illustrates, and provides guidance on interpreting the visualizations to inform decision-making.



# Conclusion and Acknowledgement

## Conclusion

The LSTM-based model successfully predicts the next word in a given sentence using contextual information. The project demonstrates how deep learning enables machines to understand human language patterns effectively. Despite limitations in dataset size, the model generalizes well and achieves good accuracy and low perplexity.

## Acknowledgement

- Faculty Advisor: Dr Sudheer M
- TensorFlow and Keras development teams
- Google Colab for computational resources