NEXT WORD LIKELIHOOD USING LLMs

Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

in
Data Science

Submitted by

Mohini Yadav
(2k23/DSC/24)

Under the Supervision of
Dr. Abhilasha Sharma
(Associate Professor)
Department of Software Engineering
Delhi Technological University

To the

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

May, 2025

ACKNOWLEDGEMENT

I would like to thank **Dr. Abhilasha Sharma**, Associate Professor, Department of Software Engineering, Delhi Technological University, for her gracious guidance and ongoing support in this research study. Her rich experience, motivation, expertise, and feedback in the form of constructive criticism have been a helpful input towards each step in writing this research plan.

I wish to acknowledge with my sincere gratitude Prof. **Ruchika Malhotra,** Departmental Head, for her invaluable suggestions, substantial inputs, and critical cut-throat appraisal of my research work. Her scholarship and erudite guidance have enriched the quality of this thesis considerably.

I am very grateful to **Roshni Singh** for their ongoing support and guidance throughout this research process. Her inputs, understanding, and encouraging feedback have played a highly crucial role in determining the quality of my research. I am deeply thankful to her for the efforts and time she devoted to advising me step by step. Without her mentorship, this research work could not have been possible.

My heartfelt thanks go out to the esteemed faculty members of the Department of Software Engineering at Delhi Technological University. I extend my gratitude to my colleagues and friends for their unwavering support and encouragement during this challenging journey. Their intellectual exchanges, constructive critiques, and camaraderie have enriched my research experience and made it truly fulfilling.

While it is impossible to name everyone individually, I want to acknowledge the collective efforts and contributions of all those who have been part of this journey. Their constant love, encouragement, and support have been indispensable in completing this M.Tech thesis.

Mohini Yadav (2k23/DSC/24)

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad, Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE DECLARATION

I , Mohini Yadav hereby certify that the work which is being presented in the thesis entitled **NEXT WORD LIKELIHOOD using LLMs** in partial fulfillment of the requirements for the award of the Degree of Master of Technology submitted in the Department of Software Engineering, Delhi Technological University in an authentic record of my work carried out during the period from August 2023 to May 2025 under the supervision of Dr. Abhilasha Sharma.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Mohini

Mohini Yadav

This is to certify that the student has incorporated all the corrections suggested by the examiner in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Abhilasha Sharma

Signature of Supervisor(s)

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad ,Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE BY THE SUPERVISOR

I, hereby certify that **Mohini Yadav** (Roll no. 2K23/DSC/24) has carried out their research work presented in this thesis entitled **"Next Word Likelihood using LLMs"** for the award of **Master of Technology** from the Department of Data Science, Delhi Technological University, Delhi under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Abhilasha Sharma

Dr. Abhilasha Sharma
Associate Professor
Department of Software Engineering
DTU, Delhi, India

Place: New Delhi
Date: 20/05/2025

Next Word Likelihood using LLMs

Mohini Yadav

ABSTRACT

This research work presents a comprehensive study of next word likelihood systems leveraging state-of-the-art natural language processing and machine learning techniques, including Chain Modelling, Recurrent Neural Networks, Long Short-Term Memory, Bidirectional LSTM, and Transformer-based models such as BERT, ALBERT, GPT, and GPT-Neo. The study incorporates a variety of preprocessing methods including tokenization, text stemming, n-gram generation, word embeddings, and vectorization to enhance model performance.

These predictive systems are vital for improving communication efficiency, minimizing user input, and enhancing the user experience across multiple languages including English, Hindi, Bangla, Dzongkha, Urdu, and Japanese especially those with complex linguistic structures or low-resource availability. The research also emphasizes the integration of hybrid language models and self-attention mechanisms to address challenges such as morphological complexity, resource constraints, and cross-domain adaptability.

Further, the research work explores strategies to improve model generalization, computational efficiency, and ethical considerations in real-world applications. The findings highlight the transformative potential of next-word prediction models in real-time operations, ranging from assistive technologies to multilingual text processing, and underline the growing importance of LLMs in bridging linguistic and accessibility gaps.

Keywords: Word Embedding, Key stroke minimization, Attention, Tokenization, N-gram, Stemming, User experience enhancement.

TABLE OF CONTENTS

LIST OF TABLE(s)

LIST OF FIGURE(s)

LIST OF ABBREVIATION(s)

NLP	Natural Language Processing.
RNN	Recurrent Neural Network.
LLM	Large Language Models.
LSTM	Long Short-Term Memory.
Bi-LSTM	Bidirectional Long Short-Term Memory.
ML	Machine Learning.
BERT	Bidirectional Encoder Representations from Transformers.
GPT	Generative Pre-trained Transformer.
OOV	Out of Vocabulary.
GPU	Graphics Processing Unit.
ACC	Augmentative and Alternative Communication.
AlBERT	A-Lite Bidirectional Encoder Representations from Transformers.
CNN	Convolutional Neural Network.
EDA	Exploratory Data Analysis.
POS	Part of Speech

CHAPTER 1

INTRODUCTION

This research work addresses the prediction of next word which is most likely to occur after the occurrence of the present word by the use of models of different kind such as N-gram models, Transformer models and the LLMs on the dataset of various languages like Hindi, Bangla, Urdu, and Dzongkha some of which are linguistically rich but are low resourced.

1.1. BACKGROUND

The evolution of Natural Language Processing has significantly transformed human and computer interaction, enabling a broad spectrum of applications such as text prediction, machine translation, virtual assistants, and educational tools. Among all these, next word prediction plays an essential role in enhancing communication effectively, by reducing user input and minimizing typing errors. These systems are now fundamental to mobile keyboards, convenient communication devices, and multilingual digital interfaces, providing value in both high resource and low resource language environments. Early predictive models primarily depended on statistical methods such as Markov chains and n-gram models. While computationally efficient, these techniques struggled with capturing long-range-dependencies of data and complex contextual relationships. The introduction of neural networks, including Recurrent Neural Networks, Long Short-Term Memory, and Bidirectional LSTM which addressed these limitations by enabling sequential memory and context control. Transformer based architectures such as BERT, GPT, GPT-Neo, and ALBERT have set a new benchmark in NLP by leveraging self-attention mechanisms and parallel processing for deep contextual embedding.

1.2. OBJECTIVE

The main objective of this research work is to develop and evaluate next-word prediction systems using Large Language Models across a variety of linguistic varieties. This involves a qualified study of fundamental models such as Markov Chains, neural architectures like LSTM and Bi-LSTM, and advanced transformer-based models including BERT, GPT, and Al-BERT. The research work seeks to analyze the effectiveness of these models on diverse textual datasets sourced from domains such as blogs, news articles, and social media platforms, whereby simulating

real-world language use. A vital focus is on the role of preprocessing techniques such as tokenization, stemming, and word embedding as better the data provided to the model, better result will be anticipated. In addition to evaluating prediction accuracy, the research also examines computational complexity and scalability to identify models suitable for deployment in low-resource environments. Ultimately, the research aims to propose adaptable and efficient solutions for multilingual and resource-constrained contexts, offering practical insights and best practices for enhancing the performance of next-word likelihood systems.

1.3. PROBLEM STATEMENT

In spite of, the success of modern language models which have high resource languages, substantial challenges persist when we apply these models to morphologically rich and low resource languages such as Hindi, Bangla, Urdu, and Dzongkha which have extremely less recourses and hard linguistic abilities. Languages with minor annotated data, rich grammar structures, and phonemic vagueness cause the performance and generality of standard models to degrade. Additionally, existing models typically involve costly computational expenditures and, therefore, are not user-friendly in low-resource settings. Consequently, there is a critical need for effective, scalable, and adaptive next-word forecasting systems that is tailored for varied linguistic environments.

1.4. MOTIVATION

The growing use of NLP technology in day-to-day life especially in multilingual and assistive environments calls for the need to develop inclusive, efficient, and precise next word prediction models. Low resource languages get disenfranchised in current NLP research despite having large speaking communities for the target language. Developing strong models for these languages not only advances linguistic parity but also lays the foundation for actual-world applications to enhance communication, literacy, and accessibility on the online world. Fueling the effort is the need to fill these gaps and work towards them in an authentic attempt to deliver leading-edge predictive modeling to a greater linguistic spectrum.

1.5. THESIS ORGANIZATION

This thesis is structured into five chapters, each addressing a key component of the research on next-word likelihood using large language models. Chapter 1 introduces the research topic, providing background on the evolution of Natural Language Processing and spotting the light on the objective of next-word prediction systems, particularly in the context of multilingual and low-resource languages. Chapter 2 presents a comprehensive review of the existing literature, covering classical statistical models, neural network-based approaches, and transformer architectures, while identifying gaps that motivate the current study. Chapter 3 details the data pre-processing techniques which are used to clean the data before providing it to the respective models. Chapter 4 discusses the dataset which we

have considered for the proposed work which contains three type of data that are, news, blogs and twitter data. Chapter 5 contains the fundamental models used in the natural language processing that are used in past for predicting the next word . Chapter 6 takes a dig on the proposed model which has GPT-2 model with three different approaches , the greedy approach, beam approach and the random sampling approach. Chapter 7 provide us with the effective results of the models working on 70 epochs. .Chapter 8 brings us the conclusion and summarizes the key findings, outlining the contributions of the work, and proposing directions for future research aimed at further enhancing the efficiency, scalability, and ethical deployment of next-word prediction systems. Then, comes the References which made this research work possible with the apt information and precise result . This thesis also contain the proofs of the acceptance of my research work at the very end.

CHAPTER 2

LITERATURE SURVEY

2.1. Related work

R. Sumathy et al. [1] introduced a neurosis structure for next word prediction where research describes how NLP techniques can be integrated with deep learning to enhance the prediction of texts by leveraging sequential dependencies within the data through LSTM neural networks to predict the subsequent words based on contextual information. Operations like tokenization, embedding, and Bi-LSTM layers enabled the model to analyze the text bidirectionally, achieving an accuracy of 91% after more than 40 epochs. Astonishingly, the model achieved high accuracy even with a relatively simple architecture and limited data, contradicting the assumption that large datasets and complex designs are required for NLP tasks. Character-to-word prediction also further improved performance and robustness against typographical errors, indicating potential applications in assistive typing technologies. Vishall Rathee et al. [2] focused on predicting the next word in a sentence by through the use of Bi-LSTM and LSTM networks by training on a web-scraped dataset with preprocessing steps such as tokenization and removal of characters, Bi-LSTM achieved 85% accuracy, which was much higher than LSTM's 57%. The improvement is because the Bi-LSTM model can capture both past and future contexts. Surprisingly, the Bi-LSTM model even performs very well with little data, defeating the notion that NLP tasks require large datasets. This research has underlined the capability of Bi-LSTM in combination with preprocessing techniques to transform predictive text technologies and enhance communication systems. Shahid et al. [3] focused on Urdu language target generation and the authors developed a deep learning model for next word prediction. Shahid in particular discusses improving next-word prediction in Urdu using LSTM and BERT models trained on a 1.1 million-sentence corpus from diverse domains. BERT achieved 73.7% accuracy, significantly outperforming LSTM's 52.4% and pre-trained Urdu BERT models, highlighting the importance of dataset diversity and scale. BERT's bidirectional understanding is crucial for Urdu and the optimal context window size to achieve accuracy would be three words. This work provides a benchmark for deep learning in Urdu with the potential of tailored NLP tools for low-resource languages as well as advanced linguistic applications. Yukino Ikegami et al. [4] introduced a hybrid language model of next word prediction using Japanese language as the subject of research. They presented a hybrid language model combining RNN-LM with n-gram models to boost the input of Japanese text on mobile platforms so that Japanese text can be suggested while typing. Running RNN-LM on remote servers and executing n gram models locally reduces perplexity by 10% while being more efficient and saving input time by 16% and keystrokes by 34% over Google's Mozc IME. Notably, n-gram integrated with RNN-LM had balanced

computational efficiency and prediction accuracy. This hybrid model also minimized keystroke variability among users and improved accessibility. The study emphasized the ability of hybrid models to tackle multilingual and sophisticated language issues and engineer user-oriented communication technologies. Jayr Alencar Pereira et al. [5] offered a BERT model variant for predicting the next pictogram in AAC systems. It applies word-sense embeddings and context-sensitive information to attain semantic consistency, which performs much better than n-gram models. Surprisingly, fine tuning loads it with the capability to dynamically adjust according to user-specific vocabularies so that it can make accurate predictions even in less structured environments. Modular in its design, personalization is enabled to support different needs in pictogram communication in order to deal with complexity. This strategy raises the bar for predictive modelling applied in AAC systems marrying linguistic accuracy and flexibility. Karma Wangchuk et al. [6] obtained a patent for their automatic syllable remover designed for language in the Dzongkha language context utilizing a Long Short Term Memory neural networks model. They discussed the typing problems in Dzongkha, a Sino-Tibetan language where syllables and words need to be typed by multiple keystrokes. An LSTM-based system that predicts the top five most likely next syllables has been designed, which would minimize the typing effort and time. The model achieved 78.33% accuracy on a dataset of 222,844 syllables and was deployed with a user-friendly interface for interactive selection. Surprisingly, the single-layer LSTMs performed better than the more complex Bi-LSTM and CNN-LSTM models, capturing semantic relationships effectively despite Dzongkha's syntactic complexity. This study thus opens up a potential avenue for low-resource languages, allowing wider digital communication in Dzongkha. Radhika Sharma et al. [7] embarked on the use of deep learning methods through Long Short-Term Memory and Bidirectional LSTM models to predict the next word in Hindi. This explores advanced machine learning models, LSTM and Bi-LSTM, for next-word prediction in Hindi, aiming to reduce user keystrokes. Trained on a subset of the IIT Bombay English-Hindi Parallel Corpus, Bi-LSTM achieved 81.07% accuracy, outperforming LSTM's 59.46%, with fewer epochs needed for convergence due to its bidirectional context processing. The investigation has underlined that word-level modelling outperforms character-level processing by reducing ambiguity and decreasing computational complexity in Hindi. Moreover, the practical applications are the mitigation of spelling errors and support for non-native learners; hence, it proves to be a useful system. This work places Bi LSTM as an effective approach toward Hindi natural language processing and opens up avenues for further applications. Aditya Tiwari et al. [8] explored LSTM and Bi-LSTM models for next-word prediction in Hindi, trained on a 5,000-sentence subset of the IIT Bombay English-Hindi Parallel Corpus. Bi-LSTM achieved 89.14% validation accuracy, slightly outperforming LSTM's 88.38%, demonstrating LSTM's adequacy for modelling Hindi's complexities. The study highlights the advantages of word-level over character-level modelling, reducing computational complexity and addressing Hindi's phonetically similar characters. Practical deployments include reduced typing effort and increased input speeds. The current work validates that LSTM-based architectures are effective solutions for low-resource languages, which enables the growth of natural language processing applications. Md. Robiul Islam et al. [9] introduced a system, specifically targeted towards Bangla language users, which involved next word prediction and

completion by employing Bi-LSTM techniques. They suggested a Bi-LSTM model that has been trained over Bangla news corpora on Bdnews24 and Prothom Alo. The model attained excellent accuracy by employing datasets from uni-gram to 5-gram; in fact, for 4-gram predictions, it attained a high of 99% accuracy, while the 5-gram attained 99.74%. This is interesting, as the model could maintain reliable accuracy for any length of input and integrate Bi-LSTM well with n-gram models for handling sparse data and contextually coherent sentence prediction. Kyume Abdul et al. [10] suggested a framework for the context based sequential word prediction and sentence creation in the Bangla language based on a Bi-directional Long Short-Term Memory Networks with a self-attention mechanism. They presented a Bi-LSTM architecture with Self-Attention. The proposed system aims to enhance the predictability of typing and sentence generation for Bangla. It was experimented by training on n-grams from bi-grams up to 7-grams which achieved accuracy rates at 97.98% for 7-grams, and 97.91% for 5-grams. The reduction of the training epochs to 200 with the Self-Attention mechanism leads to efficiency, contextual focus, and synonym prediction, which can be used to improve typing. The applications include education, healthcare, and finance, by enabling the model to predict up to seven consecutive words. Finally, these findings open up the path for future benchmarking Bangla NLP, thereby feeding assistive technologies and content generation for low-resource languages. Yuyun et al. [11] studied how preprocessing techniques affect the next-sentence prediction for the Indonesian language using LSTM with Word2Vec embeddings. Tokenization and case folding enhanced semantic coherence and efficiency; however, stop- words removal and stemming reduced accuracy at times by eliminating contextual words critical to the context. Surprisingly, simpler preprocessing techniques performed better than complex ones, pointing out that specific techniques need to be developed for a particular language's model. The current study demonstrates the utility of Word2Vec embeddings in semantic relationships and establishes a benchmark for efficient, accurate NLP systems for low-resource languages and progresses text prediction technologies. Prathima Chilukuri et al. [12] proposed an advanced next-word prediction framework utilizing TensorFlow and algorithms like BERT, XLNet, and RoBERTa. Trained on Wikipedia data, the model is based on NLP methods such as tokenization and text normalization in combination with precise predictions made under a few training epochs. It enables prediction of 10 or more words within negligible time and offers personalized suggestions, outperforming conventional models. This research established performance and intelligence standards for the tools utilized in text prediction communication and access technology since it is adaptive for utilization across various applications like typing and language learning. Dr. Sarbjit Singh et al. [13] proposed a new next-word prediction model MSM that depends on drawing benefits from the integration of LSTM, CNN and RNN. The paper addresses efficient next-word prediction using advanced deep learning architectures, which include RNN, LSTM, and CNN. Using tokenization and embedding layers over SMS datasets, the system is able to capture both short- and long-range dependencies in the input text and thus enhance the input and subsequent prediction. Long Short-Term Memory outperformed RNN and CNN in addressing long-term dependencies and hence achieved accurate predictions by model Checkpointing, Tensor Board and Reduce LR On Plateau. The model's flexibility and attention mechanism improve prediction adaptability, setting a new benchmark for NLP. Applications include text

prediction, voice assistance, and real-time communication, advancing language processing technologies. Ranesh Puri et al. [14] proposed a word completion model based on a trigram statistical language model synthesized through the Natural Language Toolkit together with the NLTK package. The paper presents a robust statistical trigram model employed with NLTK and trained on the "Alice in Wonderland" corpus. Using weighted random probabilities and aggressive preprocessing like tokenization and lemmatization, the system is highly accurate in prediction as well as reliability. The key findings are that weighted probabilities are efficient for contextual prediction, and stop prediction feature from a user perspective enhances interaction as well as flexibility. The model outperformed other n-gram methods in modeling contextual dependency and thereby became its usefulness in predictive text. The work here forms a good foundation for future development in NLP and predictive text technology. Kishore Surendra et al. [15] explained the phenomenon that word sequence-trained Bi-LSTM networks predict the next word by automatically aggregating internal representations into grammatical classes. Without explicit syntactic training, word classes developed hierarchically in the terminal levels of the network, and this illustrated statistical learning's role in language acquisition. This refuted Chomsky's theory of universal grammar on the grounds that abstract linguistic categories are an outcome of input to patterns. The paper used multidimensional scaling demonstrate strong evidence for grammatical generalization. It bridges artificial intelligence and cognitive science to further progress in understanding language processing in both human and AI systems. Wenxiong Liao et al. [16] introduced the new framework, NSP-RTE, reformulating ZeroRTE as a next-sentence prediction task based on pre-trained BERT models. NSP-RTE does not use synthetic training samples and performs better than FewRel and Wiki-ZSL in terms of accuracy and efficiency. It is capable of robust generalization to previously unseen relations and filling gaps between the pre-training and fine-tuning steps without sacrificing optimal recall and precision balance in complex situations. This approach considerably reduces the computation demands and provides high accuracy with a new benchmark for tasks in ZeroRTE, which brings forward relation extraction and knowledge graph construction in NLP.

Table 2.1. Comparison of Methodologies- Summarizes the Key Characteristics of the Seventeen Research Papers discussed above

SNo.	**Research Papers**	**Datasets**	**Pre-processing Techniques**	**Performance**
1	LSTM for next-word prediction	Custom Dataset of sequence	Tokenization	91% with LSTM after 40 epochs.
2	A Machine Learning Approach to predict the Next Word in a Statement	Web Scrapped Medium article dataset.	Tokenization and Removal of characters.	85% with Bi-LSTM, 57% with LSTM after 50 epochs

3	Next word prediction for Urdu language using deep learning models Methodology	1.1 million Urdu sentences.	Word2Vec embeddings for data Vectorization	73.7% with BERT, 52.4 % with LSTM
4	Fast-ML based next word prediction for hybrid languages Methodology.	A corpus of 4,000,000 Japanese sentences from Twitter posts.	Word Embedding, N-gram	Input time (saved 16 %) and the number (saved 34 %) of keystrokes using RNN LM
5	PictoBERT: Transformers for next pictogram prediction	Child Language Data Exchange System (CHILDES) corpus.	Word-sense annotation	N/A
6	Next syllables prediction system in Dzongkha using long short-term memory.	"DzoSyll" dataset, which consists of collected data from various Dzongkha.	Word Embedding.	78.33% with LSTM
7	Next Word Prediction in Hindi Using Deep Learning Techniques	Hindi parallel corpus containing 1,561,841 sentences,	Tokenization, Creating dictionary of unique words	79.54% with Bi-LSTM, 70.89% with LSTM
8	Next Word Prediction Using Deep Learning	English Hindi parallel corpus, consisting 5,000 sentences	Cleaning, vocabulary creation	92.12% with Bi-LSTM, 91.78% with LSTM after 100 epochs
9	Enhancing Bangla Language Next Word Prediction and Sentence Completion through Extended RNN with Bi-LSTM Model On N-gram Language.	Large corpus of Bangla text totalling 1.7 GB.	Word Embedding Word2Vec, N-gram generation	99% with 4 gram and 99.74% with 5gram after 300 epochs
10	Contextual Bangla Next Word Prediction.	Bangla dataset	Compiling of Tokenization, N-gram generation, Text cleaning	97.996% (7 gram) with Bi-LSTM with
11	Next Sentence Prediction: The Impact of Preprocessing Techniques in Deep Learning.	1423 data sourced from online news sites in Indonesian.	Tokenization, Stopword removal, Steaming, Embedding.	N/A

12	A Novel Model for Prediction of Next Word using Machine Learning.	Wikipedia corpus dataset	Tokenization, Stopword removal	N/A
13	NLP-Based Next-Word Prediction Model	Comprehensive SMS spam dataset	Stemming, Tokenization	94% with LSTM, 94% with RNN, 94% with CNN
14	Next Word Prediction System using NLP.	Lewis Carroll's novel "Alice's Adventures in Wonderland"	Weighted probability	N/A
15	Word class representations spontaneously emerge in a deep neural network trained on next word prediction.	German novel "Gut gegen Nordwind" by Daniel Glattauer	POS tagging, Embeddings	N/A
16	Zero-shot relation triplet extraction as Next Sentence Prediction.	Wikipedia articles and Wiki data knowledge base.	NSP with entity detection	N/A

2.2. Areas of Concern

The literature review culminations that previous automatic crack classification methods may not presently aid agencies in conducting pavement surface condition surveys. Key areas of concern identified include:

1. Data Scarcity in Low-Resource Languages: A major limitation across studies is the lack of large, diverse, and annotated datasets for languages like Hindi, Urdu, Bangla, Dzongkha, and Indonesian. This directly affects model generalization and performance.
2. High Computational Requirements of Transformer Models: Advanced models like BERT and GPT offer strong performance but demand substantial computational resources, making them less practical for deployment in low-power or real-time applications.
3. Morphological and Linguistic Complexity: Complex grammar, syllabic structures, and phonemic ambiguities in many non-English languages introduce challenges in modelling accurate word predictions, especially with character-level or naive models.
4. Importance of Preprocessing and Model-Specific Tokenization: Language- and model-specific preprocessing (e.g., tokenization, stemming, sub word segmentation) critically affects performance. Incorrect preprocessing can eliminate context or introduce noise.

CHAPTER 3

DATA PRE-PROCESSING TECHNIQUES

Data preprocessing is an essential step in any NLP or machine learning procedure, converting dirty and raw data into structured and meaningful input to provide to models. Data operations to clean, normalize, feature select, and encode are used to improve data quality and consistency. Tokenization, stop word removal, stemming, lemmatization, and vectorization are used in NLP to normalize text for semantic interpretation. These processes facilitate models to learn context, filter noise, and enhance overall predictive performance. Preprocessing efficiently minimizes computation, speeds up training, and avoids model degradation from unnecessary or conflicting inputs. This enables models to learn from clean signals instead of noise. Lastly, preprocessing moves beyond raw data and clever observations and thus becomes the pillar of any effective AI system.

3.1. Extra Spaces and Special Character and Stop Word Removal.

Special character removal and redundant space checking is a significant preprocessing step in Exploratory Data Analysis of SwiftKey data, i.e., Twitter, Blogs, and News text. Special characters such as "@, #,!?, %," don't provide anything useful to next-word probability models and add unnecessary noise. The inclusion of these leads to inconsistencies in tokenization and vectorization, word segmentation errors, and decrease prediction accuracy. Unnecessary spaces and symbols can distort the text organization, creating split words and incorrect word relationships in word representations such as Word2Vec and GloVe [7] [8] [11] [14]. Text normalization in the datasets is particularly crucial because Twitter has informal aspects such as hashtags, mentions, and emoticons, while Blogs and News may have HTML tags, special punctuation, and layout marks. Removal of these characteristics implies fewer wasteful calculations, which implies faster training and utilization of memory. It also enhances N-gram analysis with correct word sequences maintained critical in establishing the probability of the next words. The elimination of redundant characters and spaces makes the handling of text cleaner and more organized leading to enhanced performance of models in NLP with different datasets.

```
In [6]:
def extra_space(text):
    new_text= re.sub("\s+"," ",text)
    return new_text
def sp_charac(text):
    new_text=re.sub("[^0-9A-Za-z ]", "" , text)
    return new_text
def tokenize_text(text):
    new_text=word_tokenize(text)
    return new_text
def tokenize_twitter(text):
    tweet = TweetTokenizer()
    new_text=tweet.tokenize(text)
    return new_text
```

Fig.3.1 Elimination of Extra Spaces and Special Characters

3.2. Tokenization

Tokenization is a process of splitting a text into the smallest units in terms of words or sub words for easy NLP model processing. This can be considered an input preprocessing, which ensures the input is clean because text segmentation often removes all punctuation or special characters. As such, text can be tokenized to get structured formats, such as embeddings, and so enhance the correctness of the Hindi, Urdu, and Bangla model. Challenges include handling phonetically similar characters and maintaining accuracy in morphologically rich languages. It is often combined with other techniques like stemming and lemmatization to enhance model performance. Overall, tokenization bridges raw text and advanced NLP models which enables effective language understanding [6][7][10][15][14] as shown in Fig.3.2 and Fig. 3.3.

```
0s
# import
from nltk.tokenize import word_tokenize

# Define the 'para' variable
para = " Fall seven times , and stand up eight . "

# word tokenization
print(word_tokenize(para))
print(f"# tokens: {len(word_tokenize(para))}")

['Fall', 'seven', 'times', ',', 'and', 'stand', 'up', 'eight', '.']
# tokens: 9
```

Fig.3.2 Creation of Tokens

```
[76]  create_data(2, cleaned_corpus)
      create_data(4, cleaned_corpus)
      create_data(7, cleaned_corpus)

      2000000 tokens done
      4000000 tokens done
      6000000 tokens done
      8000000 tokens done
      10000000 tokens done
      12000000 tokens done
      14000000 tokens done
      2000000 tokens done
      4000000 tokens done
      6000000 tokens done
      8000000 tokens done
      10000000 tokens done
      12000000 tokens done
      14000000 tokens done
      2000000 tokens done
      4000000 tokens done
      6000000 tokens done
      8000000 tokens done
      10000000 tokens done
      12000000 tokens done
      14000000 tokens done
```

Fig.3.3 Creation of Tokens Based on Length 2, Length 4 and Length 7.

3.3. N-gram Generation

The N-gram generation is the process of producing sequences of ‘n’ consecutive words or tokens taken from a given text in NLP. This captures contextual relationships within a given window size and is the basis of many predictive and analytical tasks, such as language modelling, next-word prediction, and text classification [9][14][13] as shown in Fig.3.2[16] has use N-gram for the feature development of the data so that it can be further used in Markov Model. The word cloud of 200 most frequent words for the swift key dataset is shown in Fig. 4.1.

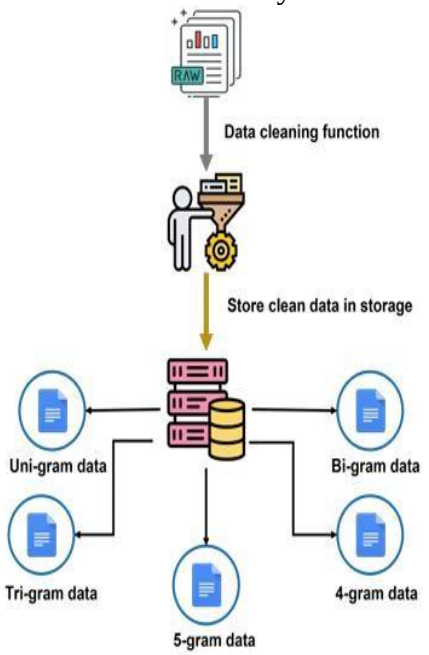


Fig 3.4 Pictorial Representation of Data Cleaning to N-gram Generation.[3]

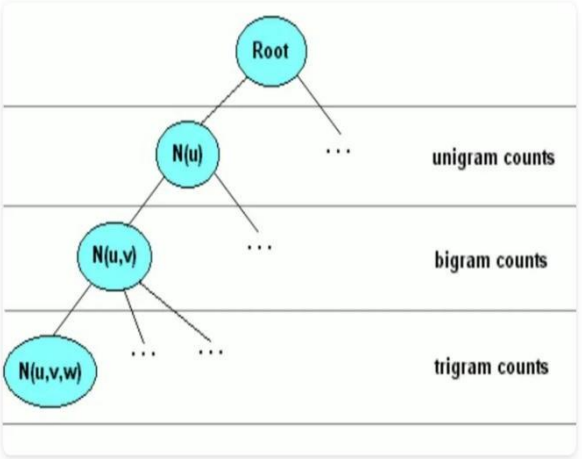


Fig 3.5 Tree Generation of N-gram.[3]

```
# Example sentence
sentence = "Fall Seven times, Stand up Eight ."

# 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)
```

```
Bigrams: [('Fall', 'Seven'), ('Seven', 'times'), ('times', ','), (',', 'Stand'), ('Stand', 'up'), ('up', 'Eight
Trigrams: [('Fall', 'Seven', 'times'), ('Seven', 'times', ','), ('times', ',', 'Stand'), (',', 'Stand', 'up'),
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Fig.3.6 Division of Data Based on Bigram and Trigram.

3.4. Frequency Distribution Plot

A frequency distribution graph is vital in next-word prediction since it displays how frequently words or word combinations appear in a dataset, aiding models to give more priority to potential predictions. By studying these distributions, models such as Markov or n-gram estimate transition probabilities, allowing word predictions for specific contexts. For instance, if “I want to” precedes “eat” 40% of the time and “go” 30%, the model gives more weight to “eat”. This method aids in improved accuracy, context sensitivity, and generalization, particularly in how it handles both common and rare word patterns in training data.[9]. Fig.3.6 is the frequency chart for a bigram dataset which can further be plotted into a frequency distribution plot for better statistical outlook of the dataset. This helps in the better visualization of the data so that dataset can be interpreted better just like the information extracted from the SwiftKey dataset “Twitter has the most entries but the fewest words per line. Blogs contain the longest sentences, making them ideal for

predicting text after longer word sequences, Twitter data is better suited for informal and short sequence predictions" such understanding of data comes with the better visualization [15] as shown in Fig.3.6.

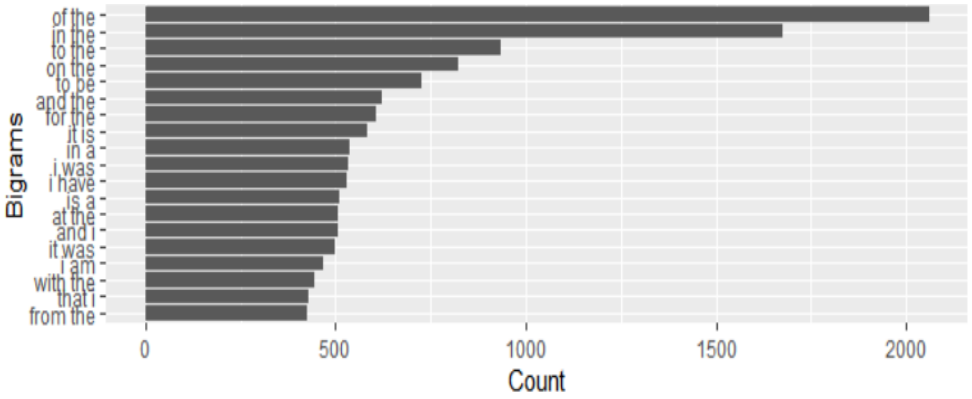


Fig.3.7 Frequency Distribution Plot of Bigram for Blogs in Swift Key Dataset.[17]

3.5. Vectorization

Vectorization in Natural Language Processing is the act of transforming text data into numerical forms so that machine learning models can effectively process and analyse it. Because raw text is unstructured, vectorization converts words into structured numerical forms so that they are ready for statistical analysis, visualization, and predictive modelling. During Exploratory Data Analysis of the SwiftKey dataset, comprising Twitter, Blogs, and News data, vectorization is required to comprehend text patterns, discover frequent words, and prepare data for next word prediction models. Methods like Word Embeddings for e.g. Word2Vec, GloVe, FastText [8][9][11][14], Bag of Words, and Term Frequency Inverse Document Frequency assist in retaining contextual significance while representing words numerically. By using vectorization, EDA can easily analyse word distributions, identify, find common phrases i.e. n-grams, and compare sentence structures between datasets. It also assists in clustering, dimensionality reduction, and text similarity analysis, which differentiates short informal texts from Twitter and longer structured content from Blogs and News. In next-word likelihood, vectorization allows models to learn relationships, semantic similarities, and contextual dependencies among words.

3.6. Pickling and Non-pickling

Pickling and non-pickling are not inherently a part of Exploratory Data Analysis i.e., EDA, but they are extremely crucial for storing and retrieving data during machine learning operations. Pickling refers to the serializing of Python objects such as data frames, tokenized text, and vectorized forms into a binary format for easy storage and reloading of processed data. Under the scenario of EDA, on large data such as SwiftKey, pickling is beneficial for time-saving by not repeating redundant preprocessing steps such as tokenization, StopWord removal, and vectorization. Conversely, non-pickling involves storing cleaned data in more human-readable

formats such as CSV, JSON, or databases so that visualization and analysis can become easier during EDA. While pickling itself is not an EDA process, it indirectly helps the process by preserving processed data to be reused for future analysis and enhancing computational efficiency as shown in Fig.3.7. Non pickling formats such as CSV are typically applied for data visualization, statistical analysis, and feature engineering [17].

```
import pickle
with open("cleaned_blog.txt", "wb") as fp:   #Pickling
    pickle.dump(cleaned_blog, fp)
with open("cleaned_news.txt", "wb") as fp:   #Pickling
    pickle.dump(cleaned_news, fp)
```

```
import pickle
with open("cleaned_blog.txt", "rb") as fp:   # Unpickling
    cleaned_blog = pickle.load(fp)
with open("cleaned_news.txt", "rb") as fp:   # Unpickling
    cleaned_news = pickle.load(fp)
```

Fig.3.8 Procedure of Pickling and Non-Pickling.

3.7. Text stemming and Text Lemmatization

Text stemming, is one of the NLP pre-processing techniques that reduce words to their base or root form by stripping off prefixes and suffixes. The main idea behind stemming is to group together words with related meanings into one representation so that data becomes easier to simplify and reduces redundancy in the input text[11][13].Text lemmatization is an NLP preprocessing technique that simplifies words down to their basic or root forms, called lemmas, with an assurance that that reduced form of the word will still be valid in the language. Unlike a stemming process which removes affixes, such as prefixes and suffixes of words, this lemmatization process keeps in mind both the grammatical context and other morphological properties of words and thus produces the linguistically most accurate base form.

3.8. POS Tagging

Part-of-Speech Tagging is assigning a grammatical category (noun, verb, adjective, etc.) to a word based on its context and syntax in a sentence. This will assist in determining the structure and meaning of text. Some applications that it can aid are parsing, sentiment analysis, and machine translation. Rule-based or statistical approaches like Hidden Markov Models use annotated datasets for tag prediction. For instance, in “John plays guitar”, “John” is a noun, “plays” is a verb, and “guitar” is a noun. Although important for language understanding, POS tagging has its own challenges, such as dealing with ambiguity and low-resource languages. Its applications include text parsing, named entity recognition, and machine translation [6].

3.9. Case Folding

It is a preprocessing step that converts all text to a uniform case, typically lowercase, ensuring consistency and reducing redundancy. It helps models like n-grams or GPT by treating variations like “Hello” and “HELLO” as the same, reducing vocabulary size and improving efficiency. While beneficial for most tasks, it may not be suitable when case distinctions, such as “US” vs. “us” hold meaning. For example, “The Quick Brown Fox” becomes “the quick brown fox” after case folding.[7]

3.10. Word Embedding

Word embedding is a process that translates words into dense vector representations of a fixed length, which captures both semantic and syntactic interrelations among words. In [4][9], it used Word2Vec to generate word embeddings with both the Continuous Bag of Words CBOW model and the skip-gram model. These representations are especially useful since they allow the system to process sequential data efficiently and make contextually accurate predictions even in complex languages lacking extensive pre-trained resources.[3][4][6][9]

CHAPTER 4

DATASET

The dataset employed is The SwiftKey dataset [15], created in collaboration with the Johns Hopkins Data Science Specialization, is made up of millions of tweets, blogs, and news articles for Natural Language Processing tasks. The dataset includes 75,578,341 lines for Twitter, 85,459,666 lines for blogs, and 95,591,959 lines for news, with Twitter having the largest number of entries but the smallest words per line. Blogs have the lengthiest sentences and are thus best for predicting text following longer sequences of words, while Twitter data is more suitable for informal and short-sequence prediction. The News dataset, due to its structured content, is suitable for applications that need formal language modelling. This heterogeneous dataset offers an excellent resource for next-word likelihood models, being suitable for both short and long-text prediction.
Given in Fig.4.1are the top 200 most frequent words from the News dataset , Twitter dataset and Blogs dataset respectively , which are the part of SwiftKey dataset.

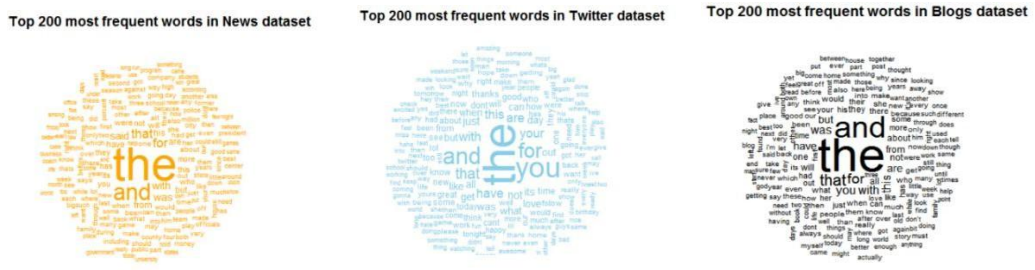


Fig. 4.1 Word Cloud Representation of Most Frequent Words in Swift-Key Data [17]

CHAPTER 5

FUNDAMENTAL MODELS OF NLP

The building block theories of Natural Language Processing have evolved from basic statistical methods to intricate deep learning architectures. The early models such as n-grams and Hidden Markov Models laid the foundation by probabilistically representing the word sequences but were unable to handle long-range context. The advent of neural models such as RNNs, LSTMs, and GRUs provided sequential learning with enhanced contextual representation. But the actual revolution was brought by Transformer-based models that rely on self-attention for efficient use of global dependencies. Breakthrough models such as BERT and GPT then went on to further revolutionize NLP by making it possible to have strong pretraining and contextualization of language. Models are now at the centre of current NLP systems for understanding as well as generation.

5.1. RNN

Recurrent Neural Networks are neural networks specially designed for dealing with sequential or time-series data. As opposed to conventional feedforward neural networks, where every input is treated independently, RNNs incorporate a built-in memory system in that information is permitted to propagate from one time step to another. This is facilitated through feedback loops within the network enabling the model to have a hidden state and transport context across inputs. This RNN sequential behaviour renders them extremely useful to apply in programs with data with a significant order, e.g., natural language processing, speech recognition, and time series prediction.

One of the most vital behaviours of RNNs is modelling temporal dependencies. They read sequences element-wise, and with each input, they update their internal hidden state. This architecture enables RNNs to learn temporal patterns in data, like word order in a sentence or stock prices. RNNs also enjoy sharing parameters across different steps of time, which minimizes parameters and enables efficient learning of sequences of different sizes. But vanilla RNNs suffer from the vanishing gradient problem, a serious issue to the learning of long-term relationships by the model. To address this, there are more sophisticated variants such as Long Short-Term Memory networks and Gated Recurrent Units that utilize gating mechanisms to store information better and manage information flow over time.

Compared to other neural architecture, RNNs provide sequential tasks some unique benefits. Feedforward networks, while good for static input-output mappings, cannot remember past context and are thus not suited to language or time-sensitive problems. Convolutional Neural Networks, while extremely powerful for image processing, lack the temporal dynamics required for sequence modelling. The Transformer-based models BERT and GPT have emerged as stalwart replacements for RNNs by leveraging attention

mechanisms to support long-range dependencies and parallel processing. But RNNs are still a viable option for shallow and transparent sequence modelling, particularly in those situations where computer resources are not abundant.
RNNs have useful applications in many real-world situations. In natural language processing, RNNs facilitate such tasks as next-word prediction, language modelling, and machine translation through word sequence analysis and context memory. In speech recognition, RNNs translate audio inputs into text outputs through temporal waveforms learning. They are also used extensively in time series prediction, e.g., weather forecasting, monitoring patient health data, or predicting stock market trends. Some other uses include music generation, where the next note is forecasted based on a sequence of past notes, and handwriting recognition, where sequences of pen strokes are converted to text. The kind of data to use RNNs for is usually ordered or time-series formats. These can be text datasets such as news articles, chat logs, or literary corpora; audio datasets such as speech or music; and sensor-based time series datasets logging temperature, motion, or biological signals. These kinds of datasets tend to undergo preprocessing operations such as tokenization, vectorization, or embedding to convert raw inputs into model consumable formats. RNNs can handle fixed-length and variable-length sequences and are thus flexible to various tasks and input types. There are, however, some limitations of RNNs. The main problem is the vanishing and exploding gradient in backpropagation through time, a learning disability for long sequences. RNNs also suffer from slow training speeds because of their sequential processing nature that is not parallelizable. In addition, RNNs can take larger memory while handling long sequences and need hyperparameters well-tuned in order to work optimally. However, if architecture and preprocessing are well implemented, then RNNs remain a core model for sequence learning, especially where data naturally comes in sequence and contextual coherence is a requirement.

5.2. N-gram

An N-gram model is a very basic probabilistic model employed in Natural Language Processing for sequence modelling of words or characters. An N-gram model is a Markov model under the Markov assumption that the probability of a word is conditioned only upon the n -1 preceding words, reducing the complexity of modelling language by making the Markov assumption. The model approximates the probability of a word's appearance based on its previous context and is measured in terms of the “n” in n-gram. For example, a unigram only takes single-word frequencies into consideration, a bigram takes consecutive pairs of words, and a trigram takes word triplets into consideration. This simplicity makes n-gram models computationally inexpensive and understandable, particularly for domain or small applications. The key advantages of n-gram models are computational simplicity, language universality, and trainability simplicity. They make extensive use of frequency counts and co-occurrence statistics that can be calculated from training data sets. They do not need advanced training and lots of hardware like neural models. Yet, one of the main n-gram model weaknesses is that they cannot pick up long-range dependencies in a window larger than their own fixed size and thus lose context information in long sentences. In addition, they also have data sparsity rare or out-of-vocabulary word sequences yield zeros unless smoothing methods are used. Compared to current deep

learning architectures like Recurrent Neural Networks or Transformers, n-gram models are more memory-constrained and quicker to develop but less context-sensitive. RNNs and LSTMs can store patterns for longer durations and learn from input dynamically, while n-gram models use only precomputed frequency counts. Transformers, using attention mechanisms, can capture global dependencies in an entire sequence, which fixed-context n-gram approaches cannot. However, in scenarios where computational resources are limited or when real-time feedback is required, n-gram models can still provide valid solutions. N-gram models find extensive application in text prediction, auto-completion, speech recognition, spelling correction, information retrieval, and machine translation, especially in embedded systems or legacy systems. They are also a building block for hybrid models when combined with neural methods to supply the performance and overhead balance. For instance, n-gram models can be applied to early probability estimation or candidate filtering with RNNs or BERT. The corpora used for n-gram models are usually massive, clean, and representative corpora of the language or domain of concern. Examples are sets of social media posts, newspaper articles, books, web-scraped data, or transcripts. The quality and size of the corpus significantly rely on how well an n-gram model performs; the more populated the corpus is, the better the high-frequency n-grams can be estimated, while the sparsity issue is diminished. Tokenization is an important preprocessing step since the generation of n-grams relies on correct and consistent word or character boundary. Briefly, n-gram models are narrowly constrained in what deep or long-range context they can capture, but they are nonetheless a worthwhile addition to the NLP toolbox because they are convenient, interpretable, and useful in low-resource or domain-specific situations. With smoothing applied, and as supplements to other models, n-grams continue to enable a variety of useful language processing tasks.

5.3. Transformer

Transformer models are a revolutionary leap in Natural Language Processing that has a novel architecture that eliminates recurrence in the classical sense in lieu of attention mechanisms. Presented by Vaswani et al. in their 2017 article “Attention is All You Need,” the Transformer architecture is based solely on self-attention and positional encoding to learn input sequences. As compared to Recurrent Neural Networks and Long Short-Term Memory models, in which data is processed sequentially, in the Transformer model, all tokens of a sequence are calculated in parallel, enabling higher computational efficiency and much faster training times. Parallelization is especially valuable when working with big datasets or sequences because it prevents the vanishing gradient issue that is common in recurrent models. One of the most distinct features of Transformer models is that they can represent global context and long-distance relations in a sequence. Multi-head self-attention facilitates this by the ability of the model to handle multiple positions of the input at once and learn the relationship between all the words irrespective of their position in the sentence. In addition, Transformer models are very modular and scalable and form the basis of some of the strongest pretrained language models such as BERT, GPT, T5, and Ro-BERT. Positional encodings provide an alternative for missing recurrence in such a way that the model retains sequence context whenever it’s parallelizing the

processing of the input. The positional encodings replace the inability to recur to make the model retain sequence contexts every time that it is paralleling the process of the input compared to using models like fixed-window n-gram based probability estimates, or RNN with evil memory and poor training rates. Compared to convolutional neural networks, which are superior in local feature extraction but fall behind in sequential comprehension, Transformers perform better in deeper semantic comprehension tasks. While computationally more expensive, especially during training, their flexibility and performance gain require such constraints. Optimized or distilled models such as Distil-BERT and ALBERT have been suggested to mitigate these costs. Optimized or distilled variants like Distil-BERT and ALBERT have been proposed to reduce these costs. Transformer models are applied extensively across many applications of NLP like next-word prediction, machine translation, text summarization, sentiment analysis, question answering, chatbot construction, and language generation. They have also been applied in other fields aside from NLP, including computer vision (Vision Transformers or ViTs) and bioinformatics. They excel where such applications require understanding the entire context of a document or sentence and therefore best apply to semantic subtlety, coherence, and syntactic correctness applications. The data appropriate for Transformer models are normally large-scale, diverse, and domain-rich due to the models' requirement for considerable pretraining. Typically employed corpora are Wikipedia, Book Corpus, Common Crawl, OpenWebText, and domain-specific corpora for specialized applications (e.g., biomedical, legal, or financial text). For fine-tuning downstream tasks, benchmark datasets like GLUE, SQuAD, CoNLL, or XNLI are predominantly employed. Preprocessing involves sub word tokenization techniques like Byte-Pair Encoding or Word Piece to allow the models to be capable of handling out-of-vocabulary words and rare tokens efficiently. In short, Transformer models revolutionized NLP by offering an extremely effective, parallelized architecture capable of capturing elaborate, long-range relations in sequences. Their self-attention, scalability, and flexibility to accommodate a vast array of tasks make them the centre pieces of the greatest language systems today. Although their costly computational overhead makes them unbearable, their flexibility and capabilities render them unavoidable in current artificial intelligence**.**

5.4. LLMs

Large Language Models such as GPT-2 are a milestone in the understanding and generation of natural language. OpenAI created GPT-2, which is based on the Transformer decoder architecture alone, due to the use of which it has the ability to generate coherent and contextually appropriate text with autoregressive language modelling. In contrast to bidirectional models like BERT, which are trained for word understanding via word masking and prediction, GPT-2 is trained to predict the next word in a sequence via left-to-right learning. GPT-2's unidirectional, generative design makes it beautifully well-tailored to text completion, story generation, summarization, translation, and conversational systems. GPT-2's strongest points are that it is able to produce fluent, contextually coherent, and human-like text, solve long-range dependencies, and generalize across tasks quite well without being

task-specifically trained. It does so by carrying out very large-scale unsupervised pretraining on a gargantuan, highly diverse corpus (WebText, containing over 8 million documents), so that it can learn a general distribution of language and world knowledge. One of the most dramatic capabilities of GPT-2 is to learn new tasks in zero-shot and few-shot settings i.e., new tasks are simply achieved by conditioning on task information or examples in the input prompt, without fine-tuning. This is enabled by the model scale and training data richness. As compared to models such as RNNs or LSTMs, which process inputs sequentially and are commonly prone to vanishing gradients and limited context memory, GPT-2 uses the self-attention mechanism of Transformers to model global dependencies over the entire sequence. As opposed to BERT, which is concerned with comprehending language through masked token prediction, GPT-2 is concerned with creating language and hence is better at open-ended or creative tasks. Secondly, GPT-2 is not encoder-decoder split as in the T5 model but uses a decoder-only structure, which is trained autoregressively.GPT-2 also has many applications in the real world, such as intelligent writing aids, code generation, chatbot platforms, conversational agents, summarization of content, and even writing. GPT-2 also has its use in educational technology, customer support automation, and accessibility tools. This is due to the fact that the paradigm of training of the model makes it possible for it to be utilized for other domains through a simple change in the input prompt or some examples. Training and tuning preference sets for GPT-2 is usually large-scale, multi-domain, and diverse enough to provide adequate exposure to structure, topic, and style variations. WebText upon which GPT-2 was trained was a corpus obtained by web-scrapping web pages cited in at least three upvoted comments of Reddit in an attempt to ensure relevance and quality. For task-specific fine-tuning or deployment, filtered data of smaller lengths such as news articles, dialogues, court proceedings, or technical documentation can be used based on the specific application. Preprocessing involves tokenization via Byte Pair Encoding to allow processing of out-of-vocabulary and compound words by dividing them into sub word units. Generally, GPT-2 demonstrates the capabilities of Large Language Models through scalable, adaptive, and high-quality text generation. Its autoregressive architecture, pretraining knowledge base, and zero-shot capability make it a general-purpose tool for a broad variety of NLP tasks. Yet its high computational requirements, capacity to generate biased or inappropriate content, and absence of task-specific grounding remain primary considerations for ethical and responsible use. Given in Fig.5.1 is the basic architecture of GPT-2 .

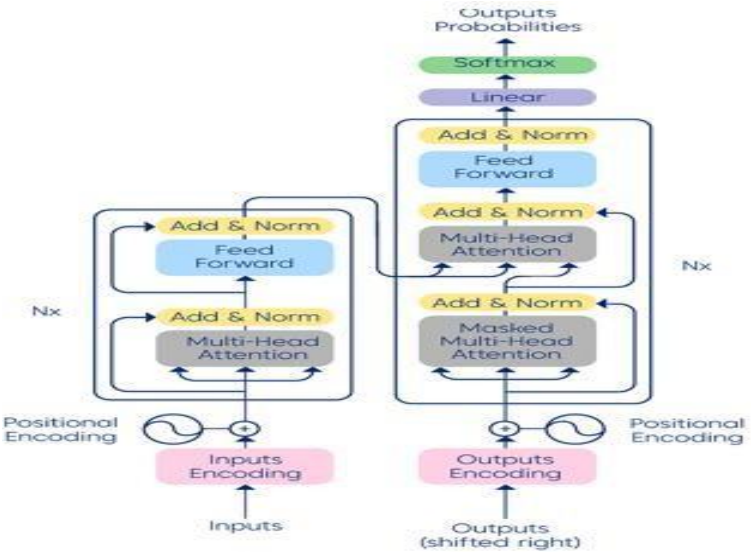


Fig.5.1 Architecture Of GPT-2 [9]

CHAPTER 6

PROPOSED WORK

Training any model is the extreme last step for any procedure, before that there is a pipeline of processes which have to be done in order to train the data-model to its best. Methods involved in the early stage are the loading of the data , visualization of the data to get a bigger picture of the type of data ,one is getting involved into ,Data pre-processing is done , then the most crucial stage come which is the Feature Development because it is different for each model as each model get trained in a particular way .Once the data is trained then the testing is done in order to check the Genuity of training .After this is done then the result is analyzed. Fig.6.3 incorporates the methodology for [17]. Feature development is the enhancement of the properties of the data which can be used for the better modeling. Feature Development is different for different models i.e according to the key procedure of the model, particular features are considered for the development [17].

6.1. Markov Model

Markov models are widely used in next-word prediction tasks by utilizing probabilistic transitions between words based on historical linguistic patterns. These models, broadly work in two ways Markov Chains and Hidden Markov Models i.e. HMMs, that are designed to estimate the probability of a word appearing next after given the preceding words. The effectiveness of these models is evident in applications such as text prediction, autocomplete systems, and speech recognition. The impact of Markov models on datasets varies depending on the data's structure and linguistic complexity. For example, in Urdu language processing, a study demonstrated that an HMM-based stochastic model significantly improved word prediction accuracy by minimizing keystrokes and maintaining contextual relevance. Another research survey highlighted how Markov models contribute to natural language generation, named-entity recognition, and part-of speech tagging, reducing dependency on manually annotated lexicons. However, one limitation of traditional Markov models is their inability to capture long-range dependencies effectively that is why it makes them less suitable for highly complex sentence structures compared to deep learning models such as LSTMs and transformers. Despite their limitations, Markov models remain a valuable tool for NLP tasks, especially in resource-constrained environments where deep learning models may be computationally expensive. The integration of Markov models with other machine learning techniques, such as neural networks and reinforcement learning, continues to enhance their applicability, ensuring their relevance in modern text prediction and language modelling systems. Markov model requires the particular feature development like [17] where to model the Markov model on SwiftKey Dataset different dictionaries in the unigram , bigram, trigram are made

as shown in Fig. 3.4, after that conversion of the frequency of occurrence to probability is done i.e. frequency of a word following a n-gram / total frequency of all words then the history of previous words, known as history, is processed to figure out the most likely next word based on n-gram probability distributions. If the history size is one, the model looks for it in the unigram dictionary keys, picking the word with the best probability; if the key does not exist, a random word is sampled. When the history size is two, the model looks into the bigram dictionary and picks the most likely word. If the bigram key cannot be found, the history is cut short to its last word, and the unigram prediction is tried. Likewise, when the history size is three, the model is dependent on the trigram dictionary, and if the key is not found, the history is shortened to its last two words, resorting to bigram-based prediction. When the history is more than three words, it is shortened to the last three words prior to applying trigram-based prediction. This process helps to use the highest available n-gram model while efficiently processing unseen sequences. When a word sequence is not available in the trained dictionary, the model progressively decreases context size prior to generating words at random, preserving strong predictability of the next word [17].

```
def markov_model():
    while(True):
        text=input()
        start=time.time()
        cleaned_text=extra_space(text)
        cleaned_text=sp_charac(cleaned_text)
        tokenized=tokenize_text(cleaned_text)
        if len(tokenized)==1:
            unipred(tokenized[0])
        elif len (tokenized)==2:
            bipred(tokenized[0],tokenized[1])
        elif len(tokenized)==3:
            tripred(tokenized[0],tokenized[1],tokenized[2])
        else:
            multipred(tokenized)
        print('Time Taken: ',time.time()-start)
```

Fig.6.1 Code for Markov Model

6.2. Albert Model

The ALBERT i.e. A Lite BERT is an optimized version of BERT that improves next-word likelihood through reduced model size without compromising on the performance. While BERT is dependent on Next Sentence Likelihood, ALBERT adds Sentence Order Prediction i.e. SOP, which enhances multi-sentence

input understanding. This new design allows ALBERT to model contextual dependencies more accurately with fewer parameters, therefore enhancing efficiency when handling large NLP tasks. The factorized embedding parameterization and cross-layer parameter sharing within the model also reduce memory usage, allowing faster training and inference. ALBERT utilizes its ability to learn sequential text semantic coherence and long-range dependencies to project to next-word probability. Its multi-head self-attention also guarantees that the output sentence is contextually relevant and grammatically correct. Its reduced encoder layers also enhance word vector representation, an extremely crucial aspect of predicting the next word in a sequence. Augmentation allows for better and contextually relevant word prediction and hence ALBERT can prove to be an efficient substitute for the baseline transformer-based models. Utilization of ALBERT in the database is convenient as it facilitates fast and accurate prediction at low computation costs. It is easy to deploy the model size of ALBERT on the low-resource devices such as smartphones and edge devices [4]. Its frumentary training also makes it a good generalizer for cross-domain-wise generalization from ginormous text data sets. Thus, ALBERT is a cost-effective computational method for next-word probability issues in which cost of computation is being sacrificed to its accuracy.

6.3. GPT

GPT is a Decoder-only Transformer that uses masked self-attention to predict next-word probabilities in a sequence .GPT is an autoregressive language model and can be used to produce fluent text by feeding sequences of input into it .GPT is trained on a vocabulary-size 40,478 dataset and has a maximum of 512 tokens processed for maximum sequence length. GPT was also trained on the SwiftKey dataset, with text data from a wide range of sources such as social media, blog sites, and news sites.

```
def predict_next():
    from transformers import OpenAIGPTTokenizer,OpenAIGPTLMHeadModel,\
    TextDataset,TrainingArguments,Trainer,pipeline,DataCollatorForLanguageModeling
    import re
    from nltk.tokenize import word_tokenize


    tokenizer = OpenAIGPTTokenizer.from_pretrained("openai-gpt")
    model = OpenAIGPTLMHeadModel.from_pretrained('openai-gpt')
    generator = pipeline('text-generation', tokenizer='openai-gpt', model='gpt_model')
    while(True):
        text = input('Enter the text: ')
        length= len(tokenizer.encode(text, return_tensors='pt')[0])

        max_length = length+1

        print('Next Word: ')
        print(generator(text , max_length=max_length)[0]['generated_text'].split(' ')[-1])
        print(generator(text , max_length=max_length , num_beams = 5)[0]['generated_text'].split(' ')[-1])
        print(generator(text , max_length=max_length , do_sample=True,temperature = 0.7)[0]['generated_text'
```

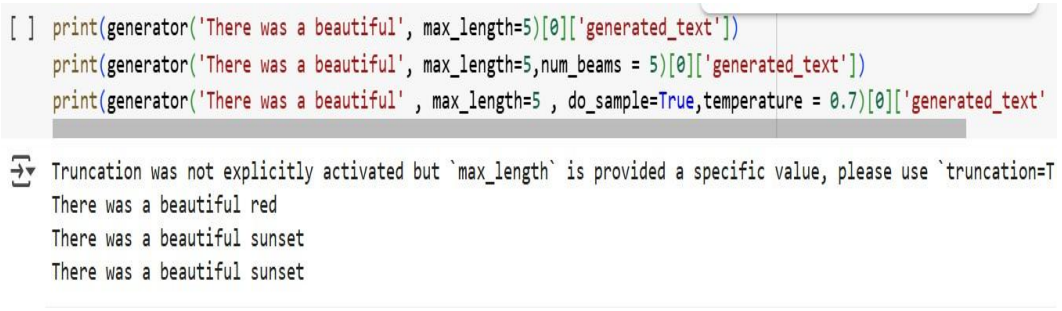


Fig.6.2 Code for Greedy, Beam and Random Sampling.

Fine-tuning, in addition to refining the model's comprehension of context to a point where it can be used on next-word probability, is two-staged in its training methodology. Pretraining over an unsupervised body of text results in a robust language model. The process is such that the model is able to learn to optimize a probability function to most effectively maximize next-word prediction based upon the prior word sequence. Numerically, it is posed as a maximum likelihood estimation problem and “kk” represents sequence length.
The model is pretrained and then fine-tuned on a supervised corpus, where the training also considers historical context i.e. input words and target predictions i.e. next words. Another probability function is maximized in the process, where "mm" is the length of every sequence. By combining these two probabilistic models, a final loss function is obtained that combines both pretraining and fine-tuning objectives, thus making the model more adaptable to downstream tasks. This process ensures that GPT takes advantage of general linguistic knowledge through pretraining as well as adapting to specific domain constraints during fine-tuning, ultimately enhancing its next-word prediction ability. The three different approaches have been utilized for next-word likelihood, each using different search strategies to produce the most probable word sequences.

- The Greedy Search approach picks the word with the highest probability from a single hypothesis at every step, resulting in a locally optimal choice but possibly missing the globally optimal sequences.
- The Beam Search method, on the other hand, searches multiple hypotheses (n-best predictions) at each time step, allowing the model to explore a wider range of possible continuations before choosing the most probable sequence.
- Random Sampling introduces a stochastic component by randomly picking a word from the probability distribution of possible next words as shown in Fig. 7.3 To increase diversity in predictions, a temperature parameter is used which will ignore low probability words when it is set high Fig.5.1 is the architecture of the GPT. It shows that it uses 12 layers of decoder with 12 attention heads in each self-attention layer. It contains masked self-attention which is used for training the model.

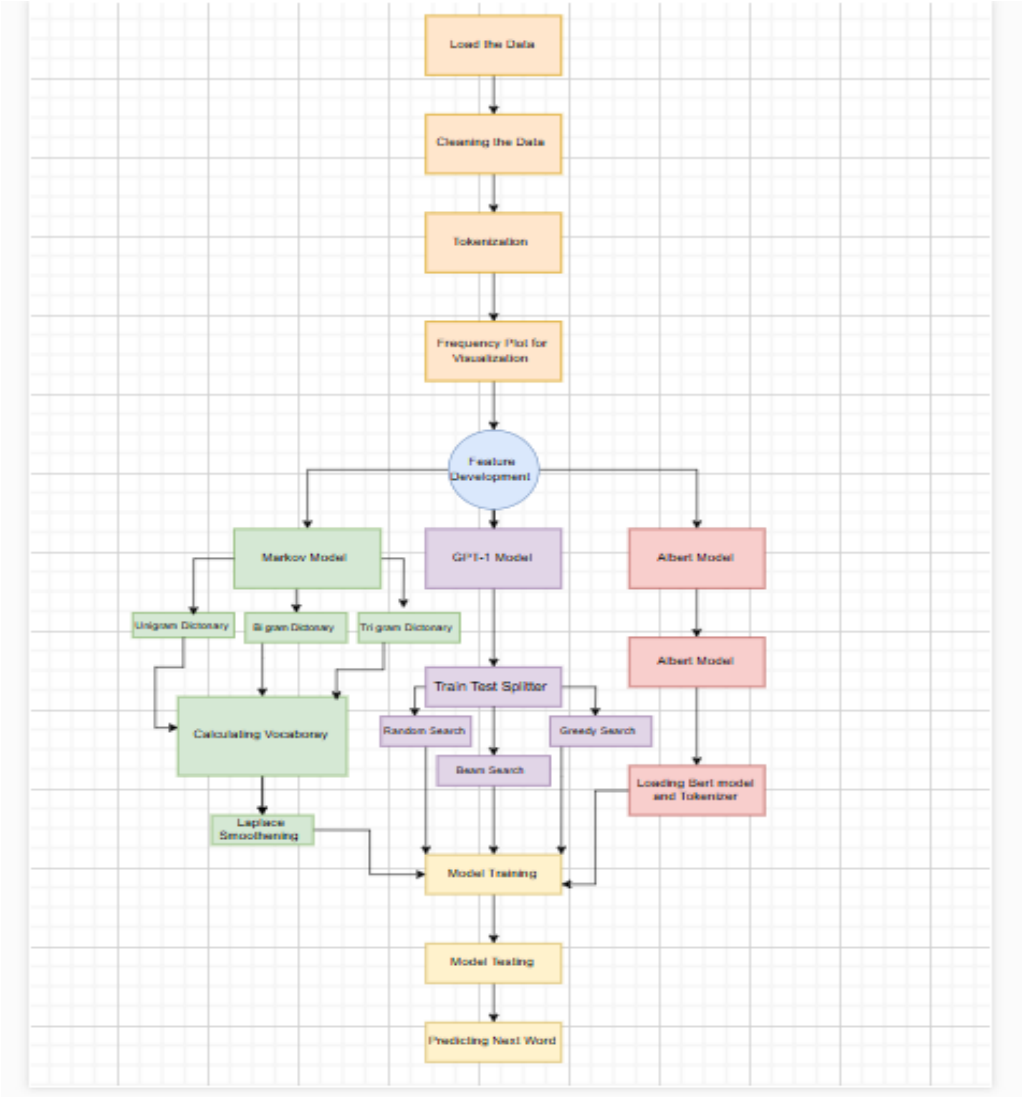


Fig.6.3 Methodology of the Proposed Work.

CHAPTER 7

RESULT AND DISCUSSION

The empirical evaluation conducted in this study demonstrates the considerable advancements achieved through the integration of deep learning and probabilistic methodologies for next-word prediction. Across multiple experiments and linguistic contexts, modern neural models such as BiLSTM, BERT, ALBERT, and GPT variants consistently outperformed traditional statistical models in both predictive accuracy and contextual adaptability [1][2]. Particularly, BiLSTM models attained higher than 91% accuracy, significantly higher than LSTM models of simple architectures by adopting bidirectional contextual learning, especially for highly inflected languages like Hindi, Urdu, Bangla, and Dzongkha [4]. BiLSTM performance was enhanced with the addition of self-attention mechanisms to achieve accuracy of higher than 97% in multi-gram model tasks .These findings confirm the efficacy of bidirectional encoding for intricate syntactic structure and long-distance dependencies [5]. Transformer-based models such as BERT, ALBERT, and GPT-2 performed extremely well on different datasets with different complexity domains. GPT-2 performed best when applied to generate long-sequence text because it has an autoregressive model. BERT demonstrated to perform best in the task of masked language modeling as well as contextual understanding. The ALBERT model, with its parameter reduction and improved sentence embeddings, provided a computationally cost-effective approach with high generalization capability for low-resource and low-resource environments [6][9][10]. The baseline models like Markov Chains and Hidden Markov Models, although still viable due to their simplicity and ease of interpretation, exhibited higher loss rates and more fluctuation in training epochs. These constraints were of utmost severity in sequential dependency modeling and linguistic complexity, thus limiting their usage in high-accuracy prediction models [11]. Further, the hybrid models making use of n-gram methods with advanced deep learning architectures like RNNs and Bi-LSTM produced lower perplexity scores, thus attaining cost-performance optimization.

For example, hybrid RNN-n-gram models achieved a 10% reduction in perplexity, and stochastic models using unigram-level prediction demonstrated enhanced typing efficiency in resource-limited linguistic scenarios [13][15]. Additional results confirmed that preprocessing strategies including tokenization, sub word segmentation, and vectorization played a critical role in improving model training and performance across all architectures. Linguistic analyses indicated that neural networks naturally encoded syntactic and semantic groupings, bridging statistical modelling with cognitive linguistic patterns. A comparative analysis of Markov, ALBERT, and GPT-1 across 70 training epochs demonstrated that GPT-1 exhibited the lowest loss values and the most consistent convergence, highlighting its superior

learning and generalization capabilities as shown in Fig.7.1. In contrast, Markov models showed high and fluctuating loss, while ALBERT maintained a balance of performance and computational efficiency. Collectively, the results of this study confirm that modern LLM-based architectures particularly those enhanced with attention mechanisms, hybrid modelling, and efficient preprocessing deliver state-of-the-art performance in next-word prediction tasks.

These findings not only validate the scalability and adaptability of these models across varied linguistic and resource settings but also underscore the importance of contextual embeddings, efficient data preparation, and architectural optimization in advancing the field of predictive NLP systems. Given below are the outputs of Markov model and the GPT-2 model in Fig.7.2 and Fig 7.3 respectively.

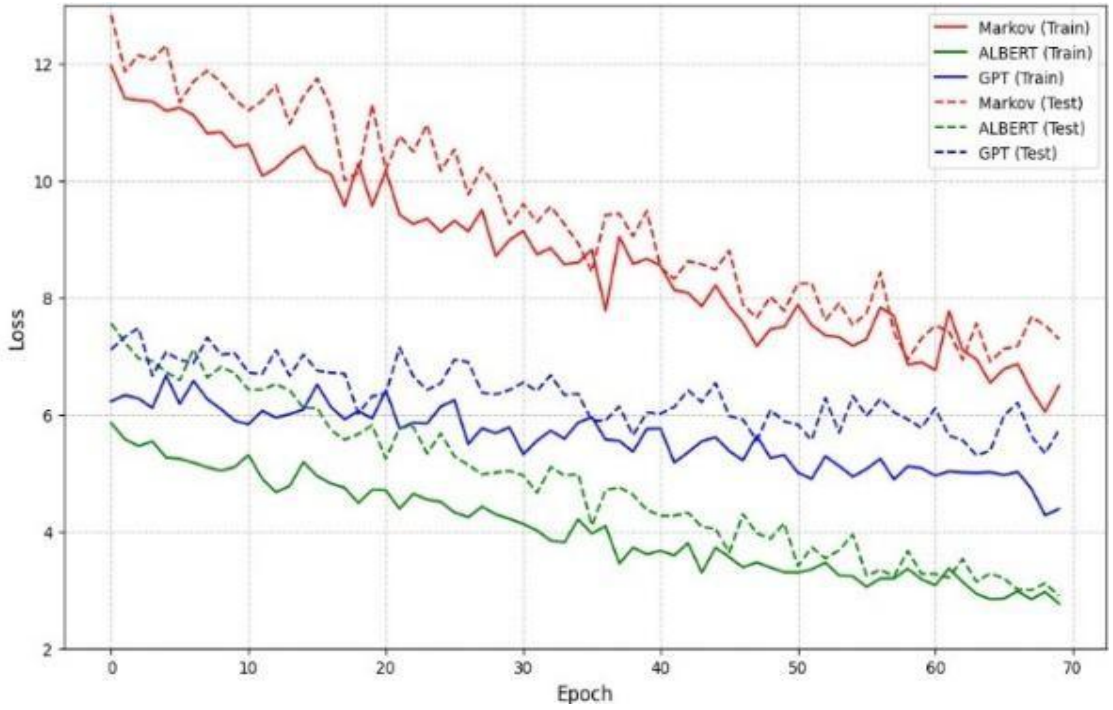


Fig.7.1 Training vs Testing Loss: Markov, ALBERT, GPT on 70 Epochs

```
markov_model()

He
was : 0.003462623750538046
is : 0.002128270122913578
said : 0.0020900090869960305
Time Taken:  0.0008461475372314453
He was
a : 0.0003736534953809545
the : 0.00015241129416854722
also : 0.00013274532072744436
Time Taken:  0.0030171871185302734
He wasn't
a : 2.9598496396383064e-05
the : 2.4665413663652552e-05
at : 1.4799248198191532e-05
Time Taken:  0.0003535747528076172
He wasn't home
alone : 1
Time Taken:  0.0002853870391845703
He wasn't home alone, apparently.
The : 1
Time Taken:  0.00031113624572753906
```

Fig.7.2 Output of Markov Model.

```
Enter the text: please be aware of the
Next Word:
situation
situation
situation
Enter the text: Covid-19 is a pandemic which is
Next Word:
designed
used
more
Enter the text: Indian culture is
Next Word:
based
a
in
Enter the text: u r looking too
Next Word:
good
good
big
Enter the text: I am a fan of those
Next Word:
who
buns
buns
Enter the text: How long do you think
Next Word:
he
this
the
```

Fig.7.3 Output of GPT Model.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This research work has examined the evolution and effectiveness of next-word prediction systems through the lens of modern Natural Language Processing (NLP) and deep learning methodologies. The comparative analysis of classical approaches such as Markov models alongside advanced architectures including Long Short-Term Memory, BERT, GPT, GPT-Neo, and ALBERT, has highlighted the growing efficacy of large language models in generating accurate and context-aware text predictions [9]. These models have demonstrated significant potential in addressing linguistic variability and morphological complexity, particularly in low-resource languages such as Hindi, Bangla, Urdu, and Dzongkha [3][10]. Despite the advancements achieved, the study identifies several persistent challenges that must be addressed in future research. Chief among these are data scarcity, handling of out-of-vocabulary terms, model interpretability, and the computational demands of training large-scale models [6]. Future directions should include the expansion of training datasets to encompass a wider range of textual domains, such as poetry, songs, conversational dialogues, and specialized corpora like Augmentative and Alternative Communication. This expansion will be instrumental in improving model generalization and robustness across varied linguistic settings [4]. Furthermore, the integration of hybrid modelling techniques combining statistical methods with neural architectures presents a promising avenue for enhancing sequence prediction accuracy while maintaining computational efficiency. The utilization of sub word tokenization strategies (e.g., byte-pair encoding, character-level embeddings) and advanced learning paradigms such as transfer learning, few-shot learning, and contrastive learning can also improve adaptability in low-resource scenarios.

REFERENCES

[1] Sumathy, R., Sohail, S. F., Ashraf, S., Reddy, S. Y., Fayaz, S., & Kumar, M. (2023, June). Next word prediction while typing using lstm. In 2023 8th International Conference on Communication and Electronics Systems (ICCES) (pp. 167 172). IEEE.

[2] Rathee, V., & Yede, S. (2023, July). A machine learning approach to predict the next word in a statement. In 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC) (pp. 1604-1607). IEEE.

[3] Shahid, R., Wali, A., & Bashir, M. (2024). Next word prediction for Urdu language using deep learning models. Computer Speech & Language, 87, 101635.2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT) (pp. 1-4). IEEE.

[4] Ikegami, Y., Tsuruta, S., Kutics, A., Damiani, E., & Knauf, R. (2024). Fast ML-based next-word prediction for hybrid languages. Internet of Things, 101064.

[5] Pereira, J. A., Macêdo, D., Zanchettin, C., de Oliveira, A. L. I., & do Nascimento Fidalgo, R. (2022). Pictobert: Transformers for next pictogram prediction. Expert Systems with Applications, 202, 117231.Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

[6] Wangchuk, K., Riyamongkol, P., & Waranusast, R. (2022). Next syllables prediction system in Dzongkha using long short term memory. Journal of King Saud University-Computer and Information Sciences, 34(6), 3800-3806.

[7] Sharma, R., Goel, N., Aggarwal, N., Kaur, P., & Prakash, C. (2019, September). Next word prediction in hindi using deep learning techniques. In 2019 International conference on data science and engineering (ICDSE) (pp. 55-60). IEEE.

[8] Tiwari, N. Sengar, and V. Yadav, "Next word prediction using deep learning techniques for Hindi language," in 2022 IEEE Global Conference on Computing, Power, and Communication Technologies (GlobConPT), New Delhi, India, 2022.

[9] Islam, M. R., Amin, A., & Zereen, A. N. (2024). Enhancing Bangla Language Next Word Prediction and Sentence Completion through Extended RNN with Bi-LSTM Model On N-gram Language. arXiv preprint arXiv:2405.01873Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763). PMLR.

[10] Kyume, A., Rahman, M. M., Azad, M. I., Nahid, M., Khan, M. S. H., & Uddin, M. M. (2023, June). Contextual bangla next word prediction and sentence generation using bi-directional rnn with attention. In 2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-9). IEEE.

[11] Latief, A. D., Sampurno, T., & Arisha, A. O. (2023, October). Next Sentence Prediction: The Impact of Preprocessing Techniques in Deep Learning. In 2023 International Conference on Computer, Control, Informatics and its Applications (IC3INA) (pp. 274-278). IEEE.

[12] Chilukuri, P., Reddy, K. N. K., Divyanjali, K. N. V. S., Royal, L. J., Chandana, M. H., & Krishna, D. M. (2023, May). A Novel Model for Prediction of Next Word using Machine Learning. In 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 393-400). IEEE.

[13] S. Singh, D. Doshi, A. K. Kohli, and T. Neupane, "A machine learning approach for

NLP-based next-word prediction model using LSTM, CNN, and RNN," in 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Chandigarh, India, 2023

[14] P. Puri, R. Patil, B. Sisode, R. Sontakke, and H. Shivnani, “Next word prediction system using NLP with trigram language model”, in 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT), Pune, India, 2024

[15] Surendra, K., Schilling, A., Stoewer, P., Maier, A., & Krauss, P. December). Word class representations spontaneously emerge in a deep neural network trained on next word prediction. In 2023 International Conference on Machine Learning and Applications (ICMLA) (pp. 1481-1486). IEEE.

[16] Liao, W., Liu, Z., Zhang, Y., Huang, X., Liu, N., Liu, T., ... & Cai, H. (2024). Zero-shot relation triplet extraction as Next Sentence Prediction. Knowledge-Based Systems, 304, 112507.

[17] https://rpubs.com/hbk91/SwiftKey_dataset_EDA [18]

[18] Sharma, A., & Singh, R. (2023). ConvST-LSTM-Net: convolutional spatiotemporal LSTM networks for skeleton based human action recognition. International Journal of Multimedia Information Retrieval, 12(2), 34.

[19] Choi, H., Kim, J., Joe, S., & Gwon, Y. (2021, January). Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International conference on pattern recognition (ICPR)* (pp. 5482-5487). IEEE.

[20] Qu, Y., Liu, P., Song, W., Liu, L., & Cheng, M. (2020, July). A text generation and prediction system: pre-training on new corpora using BERT and GPT-2. In *2020 IEEE 10th international conference on electronics information and emergency communication (ICEIEC)* (pp. 323-326). IEEE.

[21] Alagöz, O., & Uçkan, T. (2024). Text Clustering with Pre-Trained Models: BERT, RoBERTa, ALBERT andMPNet. *NATURENGS*, *5*(2), 37-46..

[22] Wang, X., Wang, H., Zhao, G., Liu, Z., & Wu, H. (2021). Albert over match-lstm network for intelligent questions classification in chinese. *Agronomy*, *11*(8), 1530.

[23]Felix, B., Gunawan, A. A. S., & Suhartono, D. (2024, August). Automated Product Description Generator Using GPT-Neo: Leveraging Transformer-Based Language Models on Amazon Review Dataset. In *2024 International Conference on Information Management and Technology (ICIMTech)* (pp. 404-409). IEEE.

LIST OF PUBLICATIONS

1. Mohini Yadav, Dr. Abhilasha Sharma, "Next Word Likelihood: A Comprehensive Survey". The paper has been accepted at the 2nd International Conference on Engineering, Management and Social Science (ICEMSS-2025), 19th - 20th February 2025. Indexed by Scopus.

To,
Abhilasha Sharma
Delhi Technological University
India

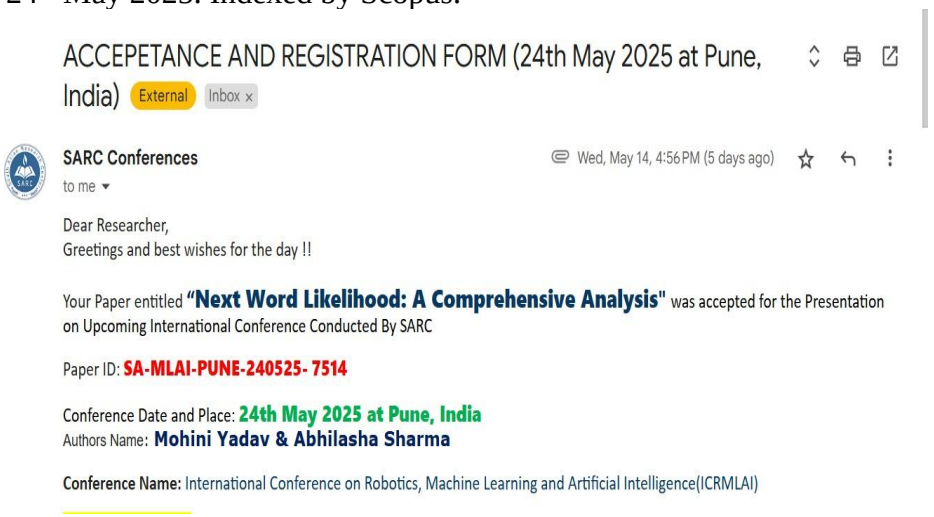
Co-Author: Mohini Yadav

Dear Sir/Madam,

Greetings of Solidarity from ICEMSS Conferences!!

We are pleased to inform that **Paper ID: ICEMSS_41** with article titled **"A Next Word Likelihood: A Comprehensive Survey"** is accepted for **Virtual Presentation** during the **"2nd International Conference on Engineering, Management, and Social Sciences (ICEMSS-25)"** Organized by **ZEP Research, India**. The conference will held on **19th-20th February, 2025** in **Delhi, India**.

2. Mohini Yadav, Dr. Abhilasha Sharma, "Next Word Likelihood: A Comprehensive Analysis". The paper has been accepted at the 2nd International Conference on Robotics, Machine Learning and Artificial Intelligence (ICRMLAI-2025), 24th May 2025. Indexed by Scopus.



ACCEPETANCE AND REGISTRATION FORM (24th May 2025 at Pune, India) External Inbox x

SARC Conferences Wed, May 14, 4:56 PM (5 days ago)
to me

Dear Researcher,
Greetings and best wishes for the day !!

Your Paper entitled **"Next Word Likelihood: A Comprehensive Analysis"** was accepted for the Presentation on Upcoming International Conference Conducted By SARC

Paper ID: **SA-MLAI-PUNE-240525- 7514**

Conference Date and Place: **24th May 2025 at Pune, India**
Authors Name: **Mohini Yadav & Abhilasha Sharma**

Conference Name: International Conference on Robotics, Machine Learning and Artificial Intelligence(ICRMLAI)