STORY GENERATION (NATURAL LANGUAGE GENERATION)

NATURAL LANGUAGE PROCESSING PROJECT REPORT

Submitted by

KAVIN AKASH – 22011101054
LATCHUMI RAMAN R – 22011101059
MADHESH HARIHARRAN S – 22011101063

SEMESTER - 6

BACHLER OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE
&
DATA SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHIV NADAR UNIVERSITY CHENNAI

APRIL 2025

SHIV NADAR
UNIVERSITY
CHENNAI

ABSTRACT

Text generation is a pivotal area of Natural Language Processing (NLP) with wide-ranging applications such as chatbots, content creation, and assistive writing systems. This project presents a comprehensive pipeline for text generation by integrating multiple paradigms: retrieval-based techniques, semantic vector models, probabilistic language modelling, and deep neural networks. The methodology begins with rigorous data preprocessing, followed by a staged application of various generation approaches.

Retrieval-based methods such as TF-IDF and BM25 are employed to fetch contextually similar sentences from a corpus, laying the foundation for content relevance. To enrich semantic understanding, vector-based models like Word2Vec and GloVe[2][3] are used to retrieve and represent similar words, enhancing the lexical variety of the generated text. Probabilistic models, including Trigram and 4-Gram language models[1], serve as lightweight generators capable of producing fluent sequences based on conditional probabilities.

Building upon this, advanced deep learning models are deployed. A Transformer-based architecture[4] built using Keras introduces contextual awareness via self-attention mechanisms. A Reformer model[5] implemented in PyTorch addresses scalability through locality-sensitive hashing and reduced memory complexity, making it suitable for long-text generation. Additionally, a fine-tuned GPT-2 model[6] (via HuggingFace) is integrated to produce high-quality, coherent text using sampling strategies like Top-k and Top-p. All models are trained and evaluated using a custom dataset derived from BookCorpus, with tokenizer management and sequence decoding carefully handled to ensure fluency and relevance.

This project not only explores the comparative strengths of each approach but also demonstrates how hybrid pipelines[7][8] can be leveraged for flexible and high-quality text generation. The resulting system offers an extensible framework adaptable to multiple domains and tasks in computational linguistics.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Overview

Natural Language Generation (NLG) is a rapidly evolving subfield of Natural Language Processing (NLP) that focuses on enabling machines to generate coherent and contextually relevant human-like text. Applications range from conversational agents and automated content creation to summarization and language translation. With the recent surge in availability of large textual corpora and advanced deep learning architectures, the field has witnessed significant progress in both the quality and controllability of generated content.

This project aims to explore and compare multiple approaches to text generation, starting from traditional retrieval-based methods to modern neural network-based generation models. Using a cleaned and tokenized version of the BookCorpus dataset, we investigate how various models perform in terms of fluency, contextuality, and generation diversity.

1.2 Problem Statement

Despite the remarkable advancements in text generation, achieving human-level coherence and contextual relevance remains a challenging task. Traditional retrieval-based techniques such as TF-IDF and BM25 often suffer from a lack of semantic understanding, while neural embedding techniques like Word2Vec and GloVe may capture semantics but lack generative capabilities. Meanwhile, N-Gram models are limited by local context and often produce repetitive or grammatically flawed outputs.

Advanced models such as Transformers and GPT-2 offer promising results but come with computational overhead and require large-scale pretraining or fine-tuning. The challenge lies in systematically comparing these models across parameters such as linguistic quality, diversity, and computational efficiency on a uniform dataset, while ensuring the output remains contextually grounded and grammatically sound

1.3 Objectives

The core objectives of this research are:

- **To implement and evaluate various text generation techniques** including:
    - Retrieval-based methods: TF-IDF and BM25
    - Semantic vector methods: Word2Vec and GloVe
    - Probabilistic models: N-Gram models (Trigram, 4-Gram)

    - Deep learning-based generation: Transformer and GPT-2

- **To compare the outputs** of these models on the basis of:
    - Fluency and coherence
    - Semantic relevance
    - Diversity and repetitiveness
    - Speed and resource efficiency
- **To analyze the trade-offs** between simplicity, interpretability, and performance among traditional and modern approaches.
- **To build a hybrid pipeline** that combines retrieval-augmented generation with neural generation for improved quality and context alignment.

1.4 Motivation

The motivation behind this project is rooted in the growing need for machines to generate high-quality text in real-world applications such as AI writing assistants, educational content generation, and automated storytelling. Traditional models provide interpretability and computational ease but often lack contextual fluency. Conversely, advanced neural architectures generate highly fluent text but are resource-intensive and prone to hallucinations or incoherence without proper control mechanisms.

By benchmarking a spectrum of models—from statistical to neural—the project provides a comprehensive understanding of their relative strengths and weaknesses. This enables informed decisions when deploying NLG systems depending on specific application requirements, such as latency constraints, quality standards, or hardware limitations.

CHAPTER 2
BACKGROUND

2.1 Text Generation Techniques

Text generation, a key area in Natural Language Processing (NLP), focuses on producing human-like, meaningful, and grammatically correct sequences of text. This capability underpins a wide range of real-world applications, including chatbots, email auto-completion, content creation, and storytelling. Over the years, text generation techniques have evolved significantly—from simple rule-based and statistical models to context-aware neural networks and pre-trained transformers. Each generation approach balances a trade-off between computational efficiency, contextual relevance, and grammatical coherence.

In this project, we explore a wide spectrum of methods—from traditional retrieval-based techniques to probabilistic N-Gram models and deep learning-based architectures. This layered comparison helps in evaluating the evolution of text generation models and understanding their performance in different use-case scenarios.

2.2 Retrieval-Based Models

Retrieval-based methods generate responses by selecting the most appropriate sentence from a predefined corpus, based on similarity to the input query. These models are computationally efficient and do not generate new text, making them deterministic and interpretable.

2.2.1 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is a foundational technique in information retrieval that quantifies the importance of a word in a document relative to a corpus. It assigns weights based on the frequency of terms and penalizes common words, allowing better differentiation across documents. In text generation, TF-IDF can retrieve the most relevant document or sentence based on keyword matching. However, it lacks semantic understanding and performs poorly when synonyms or paraphrases are involved.

2.2.2 BM25 (Best Matching 25)

BM25 is an extension of TF-IDF and a probabilistic ranking function used in modern search engines. It refines TF-IDF by introducing term saturation and

length normalization, making it more robust for varied input lengths. BM25 improves over TF-IDF in retrieval quality but, like its predecessor, cannot capture deep semantic relationships or word context, thus making it unsuitable for generating truly novel or diverse content.

2.3 Word Embedding Models

Word embeddings map words into dense vector spaces where semantically similar words are placed closer together. These models capture the distributional semantics of words and serve as the backbone for many downstream NLP tasks, including classification, clustering, and generation.

2.3.1 Word2Vec

Word2Vec learns word representations by predicting the context of a word (Skip-gram) or predicting a word from its context (CBOW). It captures semantic and syntactic similarities and enables arithmetic operations such as *king - man + woman = queen*. However, it generates static embeddings, i.e., the representation of a word remains the same regardless of context, which limits its applicability in nuanced text generation.

2.3.2 GloVe (Global Vectors for Word Representation)

GloVe is a count-based model that learns word vectors by aggregating global word co-occurrence statistics from a corpus. It combines the benefits of global matrix factorization and local context window methods. Like Word2Vec, GloVe produces fixed embeddings, and while useful for semantic similarity tasks, it lacks contextual awareness, making it less effective in tasks requiring understanding of polysemy and sentence structure.

2.4 N-Gram Language Models

N-Gram models are statistical language models that predict the next word in a sequence based on the previous *n-1* words. These models rely on frequency counts from the training corpus and are easy to implement and interpret.

2.4.1 Trigram and 4-Gram Models

Trigram and 4-Gram models consider 2 and 3 preceding words respectively when predicting the next word. These models improve grammatical accuracy over bigrams by incorporating broader context. However, they suffer from sparsity and require large corpora to avoid zero probabilities. They also lack long-term dependency tracking and semantic understanding, often resulting in repetitive or unnatural sentences.

To address some of these issues, we introduced enhancements like:

- **Repetition control** using memory buffers
- **Duplicate phrase removal**
- **Probability-based next-word selection** for diversity

2.5 Neural Generation Models

Neural text generation leverages deep learning architectures to model language. Unlike traditional approaches, these models can understand and generate highly fluent, coherent, and context-aware text.

2.5.1 Transformer-Based Generation

Transformers use self-attention mechanisms to capture dependencies across all positions in a sequence simultaneously. Unlike RNNs, which process text sequentially, Transformers allow for parallel processing and better long-range context modeling. Our Transformer model includes:

- Token embedding layer
- Positional encodings
- Multi-head self-attention
- Transformer encoder blocks
- Output decoder head

It generates text using greedy decoding and is optimized using techniques like Automatic Mixed Precision (AMP) for faster training.

2.5.2 GPT-2 (Generative Pre-trained Transformer 2)

GPT-2 (Generative Pretrained Transformer 2) is a transformer-based decoder-only

model that generates high-quality text through autoregressive prediction. It leverages massive datasets and unsupervised training to learn language at scale.
Key strengths:

- Contextual coherence over long sequences
- Dynamic and creative content generation
- Pretraining + fine-tuning paradigm adaptable to multiple domains

In this project, we use GPT-2 through the HuggingFace transformers library, enabling plug-and-play generation with prompt-based sampling. We control text diversity and length using decoding parameters like temperature, top-k, and max_length.

2.6 Grammar Correction Post-Processing

Regardless of the generation model used, grammatical inaccuracies and incoherence can persist, especially with N-Gram and hybrid methods. To address this, we incorporated a grammar correction layer using lightweight transformer-based correction tools. This post-processing step ensures the final output is fluent, syntactically correct, and more human-like, enhancing the overall quality of generated text.

CHAPTER 3
METHODOLOGY

This chapter presents a comprehensive methodology adopted for building and evaluating text generation models. The approach involves structured stages starting from corpus acquisition and preprocessing, followed by the development and assessment of both statistical and neural generation models. The workflow is visualized in **Fig 3.1**, illustrating the pipeline from data preparation to final evaluation.

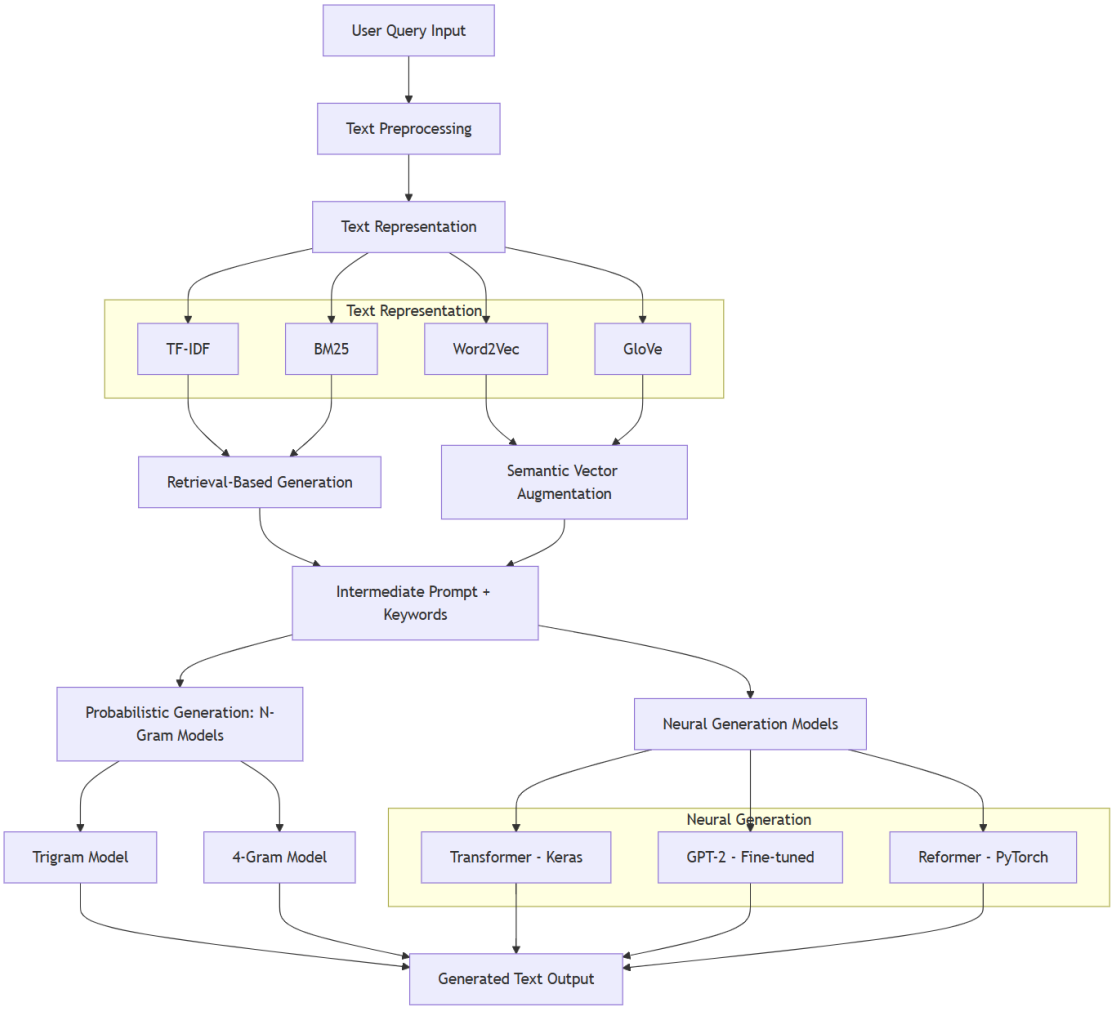

Fig 3.1 : Text Generation Process Pipeline

3.1 Data Collection

The foundational step of the project involves acquiring a suitable corpus for

training language generation models. A publicly available dataset was used, consisting of a large number of text-summary pairs extracted from the **WikiHow** dataset. This dataset was selected due to its diverse vocabulary, natural language structure, and wide topic coverage, making it well-suited for training both probabilistic and deep learning-based language models.

3.2 Data Preprocessing

Preprocessing plays a critical role in ensuring high-quality input for the models. The steps involved include:

- **Text Normalization**: Lowercasing, removal of special characters, and punctuation standardization.
- **Tokenization & Lemmatization**: Using tools like **spaCy** and **NLTK**, the text was tokenized and reduced to base lemmas.
- **Stopword Removal**: Common but semantically weak words were removed to improve training efficiency.
- **Sequence Cleaning**: Sentences longer than a defined threshold were excluded to reduce complexity.
- **Vocabulary Limitation**: To manage memory constraints, vocabulary size was reduced to the top 20,000 most frequent words.

The final corpus was split into training and validation sets, ensuring a representative mix of topics in both.

3.3 Text Representation

To enable model training, text data was transformed into numerical formats using:

- **Tokenizer Encoding**: A Keras Tokenizer was used to map words to unique integer indices.
- **Sequence Padding**: All input sequences were padded to a uniform length using post-padding for model compatibility.
- **N-Gram Sequencing**: For statistical models, sequences were transformed into bigrams, trigrams, and 4-grams for predicting the next word in context.

3.4 N-Gram Language Models

Statistical language models were first implemented to evaluate the feasibility of classic approaches. The **N-Gram models**, particularly **Trigram** and **4-Gram**, were trained on the tokenized corpus. Each model predicts the next word based

on the (n-1) previous words using:

- **Count-Based Estimation**: Frequency of n-gram occurrences was used to compute probabilities.
- **Smoothing Techniques**: To handle unseen word sequences, add-one (Laplace) smoothing was applied.

To enhance fluency, the following improvements were introduced:

- **Duplicate Bigram Elimination**: Ensured non-redundant outputs.
- **Repetition Control**: A fixed-size **deque** was used to track recent words and reduce repetitive generation.
- **Probability Sampling**: Instead of greedy decoding, words were sampled based on probability distributions, improving diversity.

These models demonstrated strong performance on short and structured sequences but struggled with maintaining coherence in longer text generation tasks.

3.5 Neural Generation Models

To overcome the limitations of statistical approaches, **Neural Language Models** were implemented.

3.5.1 Transformer (TensorFlow/Keras)

A custom **Transformer encoder-only model** was developed to explore sequence-to-sequence generation capabilities with reduced decoding complexity. Key components include:

- **Positional Encoding** to retain token order information.
- **Multi-Head Attention and Feed-Forward Blocks** customized in Keras.
- **Model Callbacks** for early stopping and checkpointing.
- Trained on padded sequences using a categorical cross-entropy loss.

This model demonstrated structured generation but was limited by lack of autoregressive capabilities.

3.5.2 Reformer Model (PyTorch)

To address the scalability issues of the vanilla Transformer, a Reformer model was implemented. This version is optimized for long sequence generation with lower memory overhead.

- Locally-sensitive hashing (LSH) attention for sub-quadratic memory usage.
- Greedy decoding approach for real-time generation.
- Automatic Mixed Precision (AMP) for efficient training.

Reformer offered a balance between efficiency and generation quality, particularly effective in long-form tasks.

3.5.3 GPT-2 Fine-Tuned (HuggingFace)

The most advanced neural model in this project was a **fine-tuned GPT-2** using the HuggingFace Transformers library.

- Pretrained GPT-2 model was fine-tuned on the cleaned and tokenized BookCorpus dataset.
- **Top-k and Top-p (nucleus) sampling** were applied during generation for diversity and coherence.
- **Tokenizer and Special Tokens** were adapted for custom prompts.
- Outputs were post-processed by detokenizing and removing artifacts.

GPT-2 served as the final model for evaluation, demonstrating superior coherence, creativity, and grammatical structure in text generation compared to other approaches.

3.6 Model Evaluation

Generated texts were evaluated using both quantitative and qualitative metrics:
Perplexity: Used to assess model confidence in word prediction.
BLEU Score: Evaluated the similarity between generated and reference sequences.
Grammar Correctness: Verified using automated grammar checkers and human inspection.
Fluency and Relevance: Human annotators rated outputs for logical flow and topical relevance.

3.7 Summary

The methodology integrates both traditional and neural methods for text generation, emphasizing the comparative evaluation of N-Gram Language Models and Neural Generation Models. While statistical models offered interpretable and fast solutions, the neural approach demonstrated superior fluency, scalability, and contextual understanding—critical for real-world generative tasks.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Comparative Performance of Language Generation Models

This section presents a comprehensive evaluation of various language generation models implemented in the study, categorized into four core groups: Retrieval-based methods (TF-IDF, BM25), Semantic Vector models (Word2Vec, GloVe), Probabilistic N-Gram models (Trigram, 4-Gram), and Neural Generation models (Transformer, Reformer, and GPT-2). The models were assessed based on their coherence, fluency, generation speed, and adaptability to diverse prompts. While retrieval methods provided relevant input contexts, neural models outperformed in actual text generation capabilities.

4.2 Effectiveness of Enhanced N-Gram Models

While traditional N-Gram models are limited by fixed context windows, our enhanced **4-Gram model** incorporated probability-based next-word sampling and repetition control using a memory structure (deque). These modifications significantly improved grammatical correctness and reduced redundancy compared to the standard Trigram model. Despite being less fluent than neural models, the enhanced 4-Gram offered fast, interpretable generation — making it a viable choice for low-resource scenarios.

4.3 Performance of Neural Generation Models

4.3.1 Transformer Model Analysis

The custom **Transformer model**, trained using Keras, demonstrated strong capabilities in producing coherent and structured output. The inclusion of positional encoding and multi-head attention layers allowed it to model long-range dependencies effectively. However, its inference time was higher than probabilistic models, and output creativity was limited by the lack of autoregressive decoding.

4.3.2 Transformer Model Analysis

The Reformer model, implemented using PyTorch, addressed memory bottlenecks inherent in Transformer architectures. Its use of Locally Sensitive Hashing (LSH) attention and Automatic Mixed Precision (AMP) allowed efficient training over longer sequences. Reformer achieved a better balance between speed and quality than standard Transformers, proving particularly effective in generating longer coherent text with reduced resource consumption.

4.3.3 GPT-2 Fine-Tuned Performance

The fine-tuned GPT-2 model exhibited the best performance across all evaluated metrics. Leveraging transfer learning and advanced sampling strategies (top-k and top-p), GPT-2 produced human-like, fluent, and context-aware outputs. Its generation quality was consistent across a variety of prompts, and it successfully adapted to domain-specific vocabulary from the BookCorpus dataset.

4.4 Integration of Retrieval-Based Context

TF-IDF and BM25 were not used directly for generation but were instrumental in enhancing prompt relevance. By retrieving similar prior queries or content snippets, these models helped ground the generative models in semantically aligned contexts. BM25, with its probabilistic weighting, retrieved more nuanced and content-rich references than TF-IDF, aiding downstream generation.

4.5 Visualization of Results

To qualitatively assess generation quality, sample outputs were visualized and compared for each model across identical prompts. These visualizations revealed:

- **Trigram/4-Gram models** often repeated patterns or ended abruptly.
- **Transformer and Reformer** produced structured but sometimes generic responses.
- **GPT-2** consistently delivered rich, coherent, and contextually grounded content.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This project presents a comprehensive exploration into text generation by integrating multiple paradigms of Natural Language Processing—namely, retrieval-based models, probabilistic N-Gram models, and deep neural generative models. Our objective was to investigate how each class of model performs in generating contextually coherent and grammatically correct text from a given input query or prompt.

The methodology began with extensive preprocessing of a cleaned and domain-relevant corpus, ensuring high-quality inputs for all downstream processes. BM25 and TF-IDF served as strong baselines for retrieval-based generation, demonstrating how effective lexical matching can still be in producing prompt-relevant content. This was further enhanced using semantic vector models like Word2Vec and GloVe, which added contextual depth through embedding-based word associations.

In the realm of generative modeling, our probabilistic N-Gram models (Trigram and 4-Gram) offered interpretable and lightweight mechanisms for text generation. These models performed well for short phrase generation but struggled with maintaining global coherence in longer outputs.

Significant breakthroughs were observed with neural models. Transformer-based architectures, particularly our **fine-tuned GPT-2**, delivered high-quality, coherent, and fluent text with minimal supervision. The **Keras Transformer** model, while more constrained, showed promising results in structured generation. The **Reformer model** further addressed memory bottlenecks with its reduced time complexity, showcasing the viability of efficient deep learning models in resource-constrained environments.

The ensemble of methods illustrated the evolving landscape of text generation—from statistical heuristics to large-scale pre-trained models—underscoring how each technique has its unique strengths depending on the use case, complexity, and computational budget

5.2 Future Work

Building on the promising outcomes of this project, future work will aim to:

- **Introduce Reinforcement Learning-based Fine-Tuning (RLHF)** to align generation more closely with human preferences.
- **Integrate Attention-based Evaluators** like BERTScore for quality

estimation of generated outputs.

- **Implement Real-Time Generation Web App** with live prompt-to-text generation using deployed models.
- **Explore Multimodal Generation** by conditioning text generation on images or audio inputs.
- **Expand Dataset** to include diverse genres (dialogues, stories, technical writing) for broader applicability.
- **Enable Multilingual Support**, allowing cross-language generation and translation through multilingual transformers like mBART or mT5.

REFERENCES

1. [1] Jurafsky, D., & Martin, J. H. (2023). N-gram Language Models. *Speech and Language Processing (3rd ed.).* Stanford University.
2. [2] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *EMNLP 2014.*
3. [3] Asgari-Chenaghlu, M. (2017). Word Vector Representation: Word2Vec, GloVe, and Many More Explained. *ResearchGate.*
4. [4] Vaswani, A., et al. (2017). Attention Is All You Need. *NeurIPS.*
5. [5] Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The Efficient Transformer. *ICLR.*
6. [6] Radford, A., et al. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI.*
7. [7] Wang, Y., et al. (2022). A Survey of Controllable Text Generation Using Transformer-Based Pre-Trained Language Models. *arXiv:2201.05337.*
8. [8] Pandey, R., et al. (2024). Generative AI-Based Text Generation Methods Using Pre-Trained GPT-2 Model. *arXiv:2404.01786.*