# A STACKED ENSEMBLE APPROACH FOR ENHANCED TEXT SUMMARIZATION

Submitted by

**1. Aritra Ghosh 12021002002137**

**2. Subhojit Ghosh 12021002002160**

***Thesis submitted for the partial fulfillment of the requirements for the degree of***

**BACHELOR OF TECHNOLOGY**

**COMPUTER SCIENCE & ENGINEERING DEPARTMENT**

**INSTITUTE OF ENGINEERING & MANAGEMENT**

**MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL**

**2025**

# A STACKED ENSEMBLE APPROACH FOR ENHANCED TEXT SUMMARIZATION

A thesis submitted by

**1. Aritra Ghosh** (12021002002137)

**2. Subhojit Ghosh** (12021002002160)

**Supervisor**

Dr. Anupam Mondal

Associate Professor
Department of Computer Science & Engineering
Institute of Engineering & Management, Kolkata

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**INSTITUTE OF ENGINEERING & MANAGEMENT, KOLKATA**

May 2025

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## CERTIFICATE

This is to certify that the **Thesis Report** on "**A Stacked Ensemble Approach for Enhanced Text Summarization**" is submitted in partial fulfillment of the requirements for the degree of Bachelor of Mechatronic Engineering by the following students:

| | |
|---|---|
| **1. Aritra Ghosh** | (12021002002137) |
| **2. Subhojit Ghosh** | (12021002002160) |

**Supervisor**
Dr. Anupam Mondal
Associate Professor
Department of Computer Science & Engineering
Institute of Engineering & Management, Kolkata

**Head of the Department**
Dr. Moutushi Singh
Department of Computer Science & Engineering
Institute of Engineering & Management, Kolkata

**Principal**
Dr. Arun Kumar Bar
Institute of Engineering & Management, Kolkata

## Thesis Approval for B.Tech.

This thesis entitled **"A Stacked Ensemble Approach for Enhanced Text Summarization"** by Aritra Ghosh and Subhojit Ghosh is approved for the degree of Bachelor of Technology in Computer Science & Engineering.

Examiner(s)

1……………………………

2……………………………

**Date:**

**Place:** Kolkata

# ACKNOWLEDGEMENT

We would like to express my heartfelt gratitude to all those who supported and guided me throughout the course of this thesis.

First and foremost, we extend my deepest appreciation to my supervisor, **Dr. Anupam Mondal**, for their invaluable guidance, constructive feedback, and constant encouragement during every stage of this research. Their insight and mentorship played a crucial role in shaping the direction and quality of this work.

We also extend our heartfelt thanks to the **faculty members** of our college for their continued support, valuable feedback, and academic inspiration during the course of our research.

Our sincere appreciation goes to our **Principal, Dr. Arun Kumar Bar**, for providing us with the academic resources and encouragement needed to pursue our work effectively. We are also deeply thankful to our **Director, Dr. Satyajit Chakrabarti**, for his visionary leadership and support that continuously motivates students toward academic excellence.

It has been a privilege to undertake this project under the guidance and within the academic environment provided by the **Institute of Engineering & Management**.

Lastly, we am deeply grateful to my family and friends for their unwavering support, patience, and motivation throughout this endeavor. Their belief in me has been my greatest strength.

……………………......... …………………….........

**Subhojit Ghosh (12021002002160)** **Aritra Ghosh (12021002002137)**

**Date:**

# DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

We hereby declare that this thesis, "**A Stacked Ensemble Approach for Enhanced Text Summarization"** contains a literature survey and original project/research work carried out by us, the undersigned candidates, as part of our studies in the Department of Computer Science & Engineering.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. We also declare that, as required by these rules and regulations, we have fully cited and referenced all material and results that are not original to this work.

**Details:**

- Name: Aritra Ghosh
- Examination Roll No: 12021002002137
- Registration No: 211040100110105

**Signature:**

**Details:**

- Name: Subhojit Ghosh
- Examination Roll No: 12021002002160
- Registration No: 211040100110130

**Signature:**

We affirm that the work presented is original and all sources have been duly acknowledged.

# TABLE OF CONTENTS

# ABSTRACT

In the fast changing digital landscape, the explosion of textual information across disciplines has made automatic text summarisation an important research field in natural language processing. The basic goal of text summarisation is to condense enormous amounts of information into succinct, logical, and informative summaries that retain crucial semantics and factual integrity. Although transformer-based models such as PEGASUS, RoBERTa, and BERT have considerably improved the quality of machine-generated summaries, these independent systems frequently have fundamental drawbacks. These include factual hallucinations, inconsistent performance across domains, and failure to generalize well to diverse input structures. This thesis proposes a Stacked Ensemble Text Summarization model that strategically combines the strengths of four distinct architectures—RoBERTa, BERT, LSTM, and PEGASUS—through a dynamic and context-aware meta-learning framework.

The ensemble is structured in two tiers. In the first level (Level-0), each of the base models is fine-tuned on benchmark summarization datasets and produces independent summaries. RoBERTa and BERT serve as strong extractive baselines, capturing sentence-level semantics and important keywords, while LSTM and PEGASUS provide abstractive capabilities, generating fluent and coherent narrative-style summaries. At the second level (Level-1), a lightweight Transformer-based meta-learner is employed. This meta-learner, inspired by the Feature-Weighted Linear Stacking (FWLS) technique, takes in base model predictions as well as rich meta-features like ROUGE scores, factuality indicators (QAFactEval, CoCo), compression ratio, semantic embedding distances, inter-model variance, and confidence scores.

The meta-learner outputs instance-specific weights, enabling either token-level fusion of logits from base models or summary selection depending on the nature of the input. Additionally, a confidence-based early-exit strategy is introduced, allowing the system to select a base model's output directly when it meets high thresholds of factuality and certainty, thereby reducing computational overhead.

Experimental results on benchmark datasets such as CNN/DailyMail and XSum demonstrate that the proposed stacked ensemble consistently outperforms individual models and traditional ensemble baselines. The model achieves up to +1.3 ROUGE-L improvement and +0.08 QAFactEval gains while reducing inference time by up to 40% when the early-exit mechanism is applied.

This study contributes a scalable, explainable, and efficient summarization solution suited for real-world deployments in news media, legal documents, scientific literature, and multilingual contexts. The integration of factuality-aware supervision, dynamic model weighting, and inference-time optimization establishes a strong foundation for future advancements in ensemble learning for text summarization.

# Chapter-1

## 1. INTRODUCTION

In today's contemporary digital landscape, information is being generated and consumed at an unprecedented pace. With the proliferation of data from a wide array of sources such as news articles, academic papers, legal documents, and medical records, the ability to efficiently ingest, interpret, and make sense of large volumes of textual content has become increasingly vital. In this high-velocity information environment, manual summarization is no longer a scalable or practical solution—it is inherently time-consuming, labor-intensive, prone to human error, and often shaped by individual subjectivity. As a result, automatic text summarization has emerged as a significant advancement in the field of natural language processing (NLP), enabling machines to produce concise, coherent, and contextually accurate summaries that capture the core essence and intent of the original material. This capability not only improves overall information accessibility but also facilitates quicker and more informed decision-making in mission-critical domains where both time and precision are of paramount importance.

Over the past few decades, the domain of text summarization has evolved considerably. Early approaches were largely based on statistical techniques, frequency analysis, and manually crafted rule-based heuristics that focused primarily on surface-level patterns and structural features of text. While these traditional methods provided some useful insights, they lacked the depth to truly understand semantic relationships or the broader context of a document, which significantly limited their performance and generalizability. The emergence of machine learning, and more recently deep learning, has marked a substantial turning point. In particular, transformer-based language models such as BERT, RoBERTa, and PEGASUS have revolutionized the field by leveraging massive datasets and pretraining on diverse corpora. These models possess the remarkable ability to capture intricate linguistic structures, infer context, and generate fluent, human-like summaries, making them highly effective for both extractive and abstractive summarization tasks.

Despite their impressive capabilities, however, standalone transformer models are not without notable limitations. Key challenges include factual inconsistencies, lack of domain adaptability, and a general brittleness when exposed to varied document types or less familiar linguistic patterns. To address these shortcomings, researchers have increasingly turned to ensemble-based techniques that aim to combine the complementary strengths of multiple summarization models. By integrating different architectures or by strategically fusing extractive and abstractive methodologies, ensemble methods can reduce individual model weaknesses, enhance output reliability, and improve semantic coherence and factual consistency.

Furthermore, ensemble approaches provide enhanced flexibility, enabling summarization systems to be customized for specific domains such as legal analytics, clinical reporting, or financial forecasting—contexts where domain expertise and nuanced language understanding are essential. As the global demand for efficient knowledge distillation and intelligent content processing continues to grow, the development of robust, scalable, and adaptable hybrid summarization systems is poised to play a pivotal role in the future of information technology, digital communication, and automated reasoning.

## *1.1 BACKGROUND*

Text summarization is a branch of natural language processing (NLP) that aims to automatically condense vast amounts of textual material into shorter, coherent summaries while retaining the main meaning. It plays a vital role in information exttraction, content recommendation, legal document analysis, and academic research by enabling users to grasp the essential points of a document quickly. Over the years, summarization methods have evolved from rule-based systems and statistical methods to more advanced machine learning and deep learning architectures.

Traditionally, extractive summarization techniques were employed, where important sentences or phrases were selected directly from the source text based on frequency, similarity, or graph-based ranking algorithms like TextRank. These methods, while reliable in preserving source text, often lack coherence and contextual flow. With the rise of sequence-to-sequence (seq2seq) models and attention mechanisms, abstractive summarization gained popularity, where models learn to generate novel phrases and rephrase the content like a human would. Models such as LSTM-based encoder-decoders, Transformer architectures, and more recently large language models (LLMs) like PEGASUS, BART, and T5 have significantly improved the fluency and abstraction capabilities of summarization systems.

However, despite the fluency and grammatical correctness offered by these models, issues such as factual hallucinations, domain-specific underperformance, and loss of important details persist. Models trained on generic datasets often fail to generalize well to specialized domains (e.g., legal or medical texts). Furthermore, many LLMs tend to prioritize linguistic elegance over factual fidelity, producing summaries that may appear accurate but are not grounded in the source material. This involves the development of strong, hybrid models that integrate the advantages of diverse architectures. Ensemble learning, particularly stacking, offers a principled way to combine multiple base models with complementary strengths. In this context, stacking allows the system to dynamically choose or blend summaries from models like RoBERTa (strong at contextual extraction), BERT (semantic representation), LSTM (sequential dependencies), and PEGASUS (fluent abstraction). When governed by a meta-learning mechanism trained on both quality and factuality metrics, such an ensemble system can achieve better informativeness, reliability, and adaptability across a wide range of summarization scenarios.

## *1.2 PROBLEM STATEMENT*

While text summarization has made significant progress with the advent of deep learning and large language models, several key challenges remain unresolved. Existing standalone summarization models, including BERT, PEGASUS, and RoBERTa, though powerful, exhibit inconsistent performance depending on the input type, domain, and length of the source document. These models tend to prioritize linguistic fluency, often at the cost of factual accuracy, producing summaries that are grammatically correct but hallucinated or misleading. In high-stakes applications—such as medical reporting, legal brief generation, or news summarization—this can lead to serious misinformation and loss of trust in automated systems.

Moreover, individual models have inherent architectural biases. For example, extractive models like BERT and RoBERTa are limited to sentence-level selection and may fail to paraphrase effectively, while abstractive models like LSTM and PEGASUS often hallucinate or miss critical details under certain conditions. As no single model generalizes well across all types of documents, there is a need for a flexible, adaptive system that can leverage the strengths of multiple architectures.

Although ensemble methods have been explored, most existing approaches use static combination strategies such as majority voting or simple averaging of predictions. These methods fail to adapt to the unique characteristics of each input document. Furthermore, most ensembles optimize only for ROUGE-based metrics, ignoring factuality-aware learning objectives, which are essential for real-world reliability.

Therefore, there is a clear research gap in designing a dynamic, meta-learning-based ensemble summarization framework that can:

- Intelligently fuse outputs from diverse summarization models based on input-specific context.
- Optimize for both informativeness and factuality.

- Maintain computational efficiency for real-world deployment.

This thesis addresses the above gap by proposing a Stacked Ensemble Text Summarization model that combines RoBERTa, BERT, LSTM, and PEGASUS through a Transformer-based meta-learner inspired by Feature-Weighted Linear Stacking (FWLS). Additionally, a confidence-based early-exit mechanism is introduced to improve inference speed without sacrificing output quality.

## *1.3 AIMS AND OBJECTIVES*

**Aim:**

The fundamental goal of this study is to create a Stacked Ensemble Text Summarization model that intelligently integrates the outputs of various summarisation models, including RoBERTa, BERT, LSTM, and PEGASUS, utilising a factuality-aware meta-learning technique. The goal is to enhance the factual consistency, informativeness, and adaptability of generated summaries while ensuring efficient inference suitable for real-world deployment.

**Objectives:**

To achieve the above aim, the following specific objectives are set:

**1.** To analyze and compare the strengths and weaknesses of existing summarization models including RoBERTa, BERT, LSTM, and PEGASUS in terms of extractiveness, fluency, and factual accuracy.

**2.** To implement and fine-tune each of the four base models on benchmark summarization datasets such as CNN/DailyMail and XSum.

**3.** To generate out-of-fold predictions from each base model using k-fold cross-validation for reliable training of the meta-learner.

**4.** To design a Transformer-based meta-learner inspired by Feature-Weighted Linear Stacking (FWLS), capable of dynamically assigning weights to base model outputs based on document-specific meta-features (e.g., ROUGE scores, factuality scores, semantic distance, compression ratio).

**5.** To integrate a multi-objective loss function that jointly optimizes for ROUGE (informativeness) and QAFactEval/CoCo (factuality), ensuring a balance between fluency and accuracy.

**6.** To implement a confidence-based early-exit mechanism that bypasses fusion when a single base model output is sufficiently confident and factually aligned, thereby reducing computational overhead.

**7.** To evaluate the proposed model with baseline models and conventional stacking methods using metrics such as ROUGE-1, ROUGE-2, ROUGE-L, QAFactEval, CoCo, and average inference latency.

**8.** To interpret and analyze the effectiveness of meta-features, model-specific behavior, and ensemble weight distribution through ablation studies.

## *1.4 THESIS LAYOUT*

This thesis is organized into six chapters, each addressing a specific component of the research systematically. The structure is as follows:

**Chapter 1: Introduction -** This chapter presents an overview of the research area, defines the problem statement, outlines the aim and objectives, and briefly describes the overall structure of the thesis.

**Chapter 2: Literature Review -** This chapter surveys existing work in the field of text summarization, including traditional and modern techniques, extractive and abstractive approaches, large language models (LLMs), and ensemble-based methods. It also highlights key limitations in current systems and identifies the research gap.

**Chapter 3: Methodology -** This chapter presents the proposed stacked ensemble summarization framework. It details the architecture of the base models (RoBERTa, BERT, LSTM, PEGASUS), the design of the Transformer-based meta-learner, meta-feature extraction, multi-objective loss function, and the early-exit mechanism.

**Chapter 4: Results and Analysis -** This chapter provides quantitative and qualitative evaluations of the proposed model using benchmark datasets. Performance is compared using ROUGE and factuality metrics, alongside baseline and ablation results.

**Chapter 5: Discussion and Conclusion -** This chapter discusses key findings, explains trade-offs and behaviors observed during testing, and summarizes the contributions of the research. It concludes the main outcomes of the work.

**Chapter 6: Summary, Publications, and Future Work -** This final chapter outlines the overall contributions, lists any publications (if applicable), and proposes directions for future enhancements such as multilingual adaptation, domain-specific fine-tuning, or real-time deployment strategies.

# Chapter-2

## 2. LITERATURE REVIEW

### *2.1 OVERVIEW OF TEXT SUMMARIZATION*

Text summarization is a subfield of Natural Language Processing that aims to automatically generate concise, coherent, and informative summaries from longer source documents. The ultimate objective is to preserve the core meaning and essential information while significantly reducing the length of the content. As data across the web, social media, research, journalism, and enterprise grows exponentially, the need for efficient summarization systems has become increasingly crucial in applications such as news aggregation, legal brief generation, medical report abstraction, and academic research summarization.

Summarization systems are broadly categorized into two primary types: extractive and abstractive.

- **Extractive summarization** selects and concatenates key sentences or phrases directly from the original document. These models rely heavily on identifying importance through ranking mechanisms such as TF-IDF, graph-based algorithms (e.g., TextRank), or supervised sentence classification. They maintain grammatical correctness but may lack coherence and fluency due to disconnected sentence placement.
- **Abstractive summarization**, on the contrary, involves generating entirely new sentences that paraphrase the source text. This is closer to how humans summarize — rephrasing, compressing, and synthesizing ideas. While abstractive summarization offers better coherence and fluency, it is also significantly more challenging as it involves natural language generation (NLG), understanding semantics, and maintaining factual consistency.

The field has seen a progression from early heuristic and frequency-based methods to statistical models, and finally to modern neural networks and transformer-based large language models (LLMs). As computational power and annotated data availability grew, so did the sophistication of summarization models. Still, despite notable progress, challenges persist — particularly in ensuring factual accuracy, maintaining relevance, handling domain shifts, and reducing redundancy.

Modern summarization systems are now expected not only to produce grammatically fluent outputs but also to retain the factual grounding of the source, handle long-context documents, and generalize across diverse content types. This shift in expectations has accelerated the development of hybrid and ensemble approaches, including stacked models like the one proposed in this thesis.

## *2.2 EVOLUTION OF SUMMARIZATION TECHNIQUES*

The development of automatic text summarization systems has undergone several distinct phases, each building upon prior advances in computational linguistics and machine learning.

### *2.2.1 Early Statistical and Heuristic Methods*

In the early days, summarization relied on statistical techniques and heuristic rules. The simplest methods used term frequency (TF) or TF-IDF (Term Frequency-Inverse Document Frequency) to identify important words and rank sentences accordingly. Sentences with higher cumulative TF-IDF scores were selected to form a summary. These systems, while efficient, lacked contextual understanding and often included redundant or disjointed content.

To address this, graph-based algorithms such as TextRank (Mihalcea & Tarau, 2004) and LexRank (Erkan & Radev, 2004) were introduced. These models constructed a similarity graph of sentences and used algorithms like PageRank to identify the most central sentences. Extractive by nature, they improved coherence but were limited by their shallow linguistic representations.

### 2.2.2 Neural Networks and the Rise of Seq2Seq

The next significant milestone came with the advent of Recurrent Neural Networks (RNNs) and later Long Short-Term Memory (LSTM) models. These architectures enabled the development of sequence-to-sequence (seq2seq) models for abstractive summarization. In this setup, the encoder RNN reads the input document, and the decoder RNN generates the summary. While promising, early RNNs struggled with long-range dependencies due to vanishing gradient issues.

The introduction of attention mechanisms (Bahdanau et al., 2015) revolutionized this setup by allowing the model to dynamically focus on relevant parts of the input during decoding. This improved both the fluency and relevance of generated summaries. Luong attention, pointer-generator networks, and coverage mechanisms were later added to mitigate problems like repetition and out-of-vocabulary words.

### 2.2.3 Transformer-Based Architectures

The breakthrough in summarization came with the release of the Transformer architecture (Vaswani et al., 2017). Unlike RNNs, Transformers rely entirely on self-attention, enabling parallel computation and better handling of long sequences. This architecture formed the backbone for a series of powerful pretrained models:

- BERT (Devlin et al., 2018): Bidirectional encoder pretrained on masked language modeling. Primarily used for extractive summarization by classifying sentence importance (Liu & Lapata, 2019).
- RoBERTa (Liu et al., 2019): A robustly optimized BERT, trained with larger data and better hyperparameters. Used in place of BERT for stronger semantic extraction.
- T5 (Raffel et al., 2020): Treated summarization as a text-to-text task. Unified multiple NLP tasks under one architecture.
- PEGASUS (Zhang et al., 2020): Specifically pretrained for abstractive summarization using a novel Gap Sentence Generation (GSG) objective, where important sentences are masked and the model is trained to reconstruct them.

These models, particularly PEGASUS and BART, dominate the current state-of-the-art benchmarks on datasets like CNN/DailyMail and XSum.
A comparative overview of summarization techniques, their strengths, and limitations is shown in **Table 2.1**.

Table 2-1: Comparison of summarization techniques by type, model, strengths, and limitations

| Technique | Type | Model Examples | Strengths | Limitations |
|---|---|---|---|---|
| Rule-based / TF-IDF | Extractive | LexRank, TextRank | Simple, fast, interpretable | Lacks context, redundant content |
| Seq2Seq LSTM | Abstractive | LSTM + Attention | Fluent output, paraphrasing | Hallucination, struggles with long input |
| Transformer Encoder | Extractive | BERT, RoBERTa | Semantic extraction, robust | No generation, extractive only |
| Transformer Decoder | Abstractive | PEGASUS, T5, BART | Highly abstractive, fluent summaries | Hallucination, costly computation |
| Ensemble (Stacked) | Hybrid | This Work | Robust, dynamic, balances multiple models | Complexity, needs tuning of meta-learner |

## *2.3 BASE MODELS IN FOCUS*

This research proposes a stacked ensemble architecture that integrates four diverse summarization models, each chosen for its unique contribution to summary quality. These include BERT and RoBERTa (both excelling at extractive summarization), LSTM (a sequential, generative baseline), and PEGASUS (a transformer-based abstractive model). Understanding each of their strengths and limitations is crucial to justifying their inclusion in the ensemble.

### *2.3.1 BERT (Bidirectional Encoder Representations from Transformers)*

Engineered by Devlin et al. (2018), BERT introduced a bidirectional transformer encoder pre-trained on large corpora using a masked language modeling task. While originally designed for classification and question-answering tasks, BERT was later adapted for summarization — primarily for extractive summarization.

Liu and Lapata (2019) proposed BERTSUM, where BERT is used to encode a document and a classifier is added to predict which sentences should be included in the summary. Its bidirectional contextual embeddings allow it to understand sentence importance better than traditional vector-based approaches.

The approach demonstrates several notable strengths. It excels at identifying semantically important sentences, ensuring that the most relevant information is captured. Its robust performance across various domains makes it highly versatile, and it also supports fine-tuning with minimal data, making it adaptable to specific tasks with limited resources. However, it does have certain limitations. Being restricted to extractive summarization, it cannot generate new text or rephrase content creatively. Additionally, it may struggle with producing summaries that are fully fluent and coherent, as it relies on stitching together existing sentences rather than generating smooth, connected narratives.

### *2.3.2 RoBERTa (Robustly Optimized BERT Pretraining Approach)*

RoBERTa, proposed by Liu et al. (2019), enhances BERT by training on more data with dynamic masking, longer sequences, and without the next-sentence prediction objective. It retains BERT's encoder structure but achieves stronger sentence embeddings and better generalization.

Used similarly to BERTSUM, RoBERTa-based summarizers offer stronger extractive performance and improved sentence scoring due to richer pretraining.

This approach offers several strengths that enhance its utility in natural language tasks. It outperforms BERT in modeling sentence semantics, capturing deeper and more nuanced meanings. It is also highly robust and generalizable, performing well across a wide range of tasks and domains. Moreover, it supports plug-and-play integration, making it convenient for sentence classification tasks without requiring extensive customization. However, it shares some limitations with other extractive models—it is limited to extraction and cannot paraphrase or generate novel content. Additionally, the quality of its output can be heavily influenced by the structural diversity of the input, potentially reducing effectiveness in more uniform or repetitive datasets.

### *2.3.3 LSTM (Long Short-Term Memory Networks)*

Hochreiter and Schmidhuber (1997) invented LSTM networks, a form of Recurrent Neural Network (RNN) that is designed to manage long-range relationships in sequential data. In summarisation, LSTMs are the foundation of sequence-to-sequence (seq2seq) models, in which an encoder reads the input document and a decoder produces the summary.

With the addition of attention mechanisms (Bahdanau et al., 2015), LSTM-based summarizers became more effective at focusing on important parts of the input while generating each word. These models were the standard for abstractive summarization before the rise of Transformers.

This model exhibits several strengths that make it effective for summarization tasks. It generates fluent, human-like summaries that are natural and coherent. Its ability to handle sequential and temporal dependencies enables it to maintain logical flow, and it performs reasonably well on medium-length documents. However, there are notable limitations. The model struggles with very long inputs due to constrained memory, which can lead to incomplete or truncated summaries. It is also less parallelizable, resulting in slower training and inference times. Furthermore, compared to Transformer-based models, it tends to be more prone to repetition and factual inconsistencies, affecting the overall reliability of its outputs.

Despite their limitations, LSTMs are valuable in ensemble setups as a contrast to Transformer models, offering diversity in sentence structure and style, which can improve ensemble fusion outcomes. A summary comparison of the four base models considered in this thesis is provided in **Table 2.2**.

Table 2-2: Overview of popular summarization models with their types, strengths, and key limitations

| **Model** | **Summarization Type** | **Strengths** | **Weaknesses** |
|---|---|---|---|
| BERT | Extractive | Good at identifying sentence-level importance | Cannot paraphrase or generate |
| RoBERTa | Extractive | Improved embeddings over BERT, better generalization | Still extractive, needs reranking |
| LSTM | Abstractive | Fluent summaries, good for sequential data | Hallucination, less efficient |
| PEGASUS | Abstractive | State-of-the-art, fluent and compact summaries | Hallucinates facts, expensive inference |

### *2.3.4 PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization)*

PEGASUS, developed by Google Research (Zhang et al., 2020), is a Transformer-based model specifically designed for summarization. Unlike BERT or RoBERTa, which are pre-trained on general-purpose language modeling tasks, PEGASUS is pre-trained using Gap Sentence Generation (GSG) — where important sentences are masked and the model learns to generate them from the remaining context. This pretraining objective is closely aligned with the downstream task of summarization.

PEGASUS achieves state-of-the-art results on multiple datasets such as CNN/DailyMail, XSum, and Gigaword. It is capable of producing highly abstractive, fluent, and concise summaries, mimicking human summarization styles.

**Strengths:**

- State-of-the-art abstractive performance
- Pretraining is well-aligned with summarization
- Generates diverse, fluent summaries

**Limitations:**

- Prone to hallucination (generating factually incorrect statements)
- Computationally heavy; not ideal for low-resource environments
- May generate overly short or overly generic summaries in some cases

Together, these four models contribute complementary capabilities to the proposed ensemble:

- BERT & RoBERTa ensure content relevance (extractive precision)
- LSTM & PEGASUS enable fluency and abstraction (generative creativity)

Their fusion—governed by a meta-learner—promises to overcome the individual limitations of each.

## *2.4 FACTUALITY ISSUES IN SUMMARIZATION*

One of the most pressing challenges in modern text summarization is the issue of factual inconsistency, also known as hallucination. This occurs when a summarization model generates statements that are grammatically plausible and fluent but factually incorrect or unfaithful to the source document. Such errors are particularly dangerous in high-stakes domains such as journalism, legal reporting, medicine, and scientific communication, where the integrity of information is non-negotiable.

This problem is most commonly observed in abstractive summarization models, especially those based on large Transformer architectures like PEGASUS, BART, and GPT-based variants. As noted in Maynez et al. (2020), even when models achieve high ROUGE scores, they frequently hallucinate content — either by introducing information not present in the source or by distorting facts. This exposes a critical limitation of ROUGE-based evaluation, which primarily measures n-gram overlap and does not capture semantic or factual correctness. While ROUGE remains the de facto metric for evaluating summary quality, it is inherently surface-level. ROUGE can be high even when the summary is semantically or factually incorrect, as long as it overlaps with reference summaries in wording.
To address this, several factuality-focused metrics have been proposed:

- QAFactEval (Fabbri et al., 2022): Evaluates factual consistency by generating QA pairs from summaries and checking if the source document contains the answer. More robust than ROUGE for real-world use.
- CoCoScore: Uses question generation and entailment checking to verify consistency between summary statements and source content.
- G-Eval: A recent approach that evaluates factuality by generating guided summaries and comparing entailment.

These tools enable researchers to evaluate and train models not just for informativeness, but also for truthfulness. In practice, they are used as auxiliary training signals, or as part of loss functions in multi-objective optimization, as is done in your proposed work.

Factuality in current models often fails due to several contributing factors. One key reason is that abstractive models are designed to generalize across training data, which can lead them to interpolate "likely facts" that appear contextually plausible but are not actually grounded in the input. Additionally, many models lack effective grounding mechanisms, such as retrieval systems or memory modules, that could help anchor their outputs to verified information. Furthermore, these models are typically optimized for fluency and metrics like ROUGE, which prioritize readability and similarity over strict adherence to source accuracy, thereby increasing the risk of hallucinations. These limitations highlight the need for architectures that can balance semantic abstraction with factual fidelity. This is a primary motivator for your factuality-aware ensemble design, where each model's output is scored not just by informativeness but also by factuality metrics, and a meta-learner decides the final output accordingly.

Table 2-3: Comparison of evaluation metrics for summarization based on type, focus, and limitations.

| Metric | Type | Evaluates | Limitation |
|---|---|---|---|
| ROUGE (1/2/L) | Lexical Overlap | Informativeness | Surface-level; ignores meaning/factuality |
| BERTScore | Semantic | Semantic similarity | Can reward hallucinated text |
| QAFactEval | Factual | QA-based factual checking | Requires QA pair generation |

## *2.5 ENSEMBLE METHODS IN NLP*

Ensemble learning is a machine learning methodology that integrates numerous models to obtain better performance than any single model. The core idea is based on the principle that different models capture different patterns or aspects of the data, and intelligently aggregating their predictions leads to more robust and generalizable outcomes. Ensemble techniques have long been used in traditional machine learning, and their adoption in NLP has grown significantly with the availability of multiple large language models and deep architectures.

There are several types of ensemble techniques, some common ones include:

- Bagging (Bootstrap Aggregating): Models are trained independently on random subsets of the data, and their outputs are averaged or voted. Random Forests are a classic example.
- Boosting: Models are trained sequentially, with each model correcting the errors of the previous one. While powerful, boosting is less common in deep NLP due to training instability with large models.
- Voting/Blending: Involves taking majority or weighted votes from multiple models' outputs. Often used in extractive summarization or classification tasks.
- Stacked Generalization (Stacking): This is the most sophisticated and flexible ensemble approach, where:
- Multiple base models (Level-0 models) are trained independently.
- A meta-learner (Level-1 model) is trained to combine their outputs based on features of the predictions or the input.
- The meta-learner can learn when to trust which model, making it more adaptive than simple voting.

Stacking has shown state-of-the-art results in various tasks like question answering, text classification, and increasingly in summarization.

### *2.5.1 Feature-Weighted Linear Stacking (FWLS)*

One advanced variation of stacking is Feature-Weighted Linear Stacking (FWLS), introduced by Sill et al. (2009). In traditional stacking, the meta-learner assigns fixed weights to each model across all inputs.

FWLS improves on this by:

- Making the weights dynamic and instance-specific,

- Using input features (e.g., length, readability, topic, confidence) to adjust model weights,
- Allowing better specialization (e.g., trusting PEGASUS for shorter news articles, BERT for factual extraction in long documents).

FWLS can be implemented with linear regression or more advanced learners like small neural networks or transformers. In your thesis, this idea is extended with a Transformer-based meta-learner that dynamically adjusts weights using meta-features like ROUGE scores, factuality scores, compression ratio, etc.

Ensemble methods — particularly stacking and FWLS — provide a scalable and modular framework for combining summarization models. By letting a meta-learner decide which model(s) to trust per input, you gain adaptability, improved accuracy, and the ability to optimize for multiple objectives simultaneously (e.g., factuality and fluency).

## *2.6 ENSEMBLE SUMMARIZATION APPROACHES*

While ensemble learning has long been established in classification and regression tasks, its application to text summarization is still emerging. Ensemble summarization methods aim to combine the outputs of multiple base summarizers to improve fluency, informativeness, and factual consistency. The motivation is clear: different models bring different strengths — some excel at sentence selection, others at paraphrasing or fluency, and some at preserving factuality.

### *2.6.1 Early Ensemble Summarizers*

Early ensemble summarization approaches were predominantly heuristic in nature. A common method involved majority voting over sentence inclusion, where sentences selected by multiple extractive models were prioritized in the final summary. Another strategy was to compute the average ROUGE scores of candidate summaries and select the one with the highest score, assuming it to be the most representative. Some systems

also incorporated reranking based on surface-level linguistic quality metrics such as grammatical correctness, optimal length, and minimal redundancy. While straightforward, these methods lacked deeper semantic understanding and adaptability, limiting their effectiveness in more complex summarization scenarios.
However, these methods were static, domain-agnostic, and often failed to generalize.

### *2.6.2 Weighted Fusion and Meta-Ranking*

More recent approaches have explored the weighted fusion of multiple abstractive summaries to enhance overall summary quality. Some systems employ sentence-level metrics such as BERTScore or ROUGE to select among overlapping generated sentences, aiming to retain the most semantically relevant content. Others rely on language model perplexity as an indicator of fluency, combining the top-ranked segments accordingly. For instance, Lebanoff et al. (2020) utilized ranking models to merge multiple base summaries, while Narayan et al. (2021) introduced a method that selects summaries based on entailment and coverage scores. Although these techniques outperform simple voting schemes, they still lack a truly adaptive, learned fusion strategy that can tailor itself to the characteristics of different document types.

### *2.6.3 Stacking in Summarization*

The concept of stacked ensemble summarization—where a Level-1 meta-model integrates predictions from multiple base models—is a more recent and relatively underexplored direction. Some experimental setups have leveraged BERT-based rankers to select the best summary among outputs from models like BART, T5, and PEGASUS. Others have trained lightweight multilayer perceptrons (MLPs) to learn which base model performs best for specific domains, enabling more context-aware fusion. However, despite these promising innovations, most existing works remain focused on optimizing ROUGE scores, often overlooking factuality—an essential aspect for ensuring real-world reliability and trustworthiness in summarization systems.

### *2.6.4 Factuality-Aware Stacking*

The approach represents a significant leap in ensemble summarization. It introduces a Transformer-based meta-learner that leverages feature-weighted gating, where the contribution of each base model is dynamically adjusted based on several key factors: summary length, ROUGE overlap, factuality metrics such as QAFactEval and CoCo, and model confidence. This system supports both soft fusion—blending tokens from multiple summaries—and hard selection, where the best summary is chosen outright. Additionally, it includes a confidence threshold-based early exit, allowing for reduced latency when one model's output is deemed highly reliable. By integrating semantic, syntactic, and factual cues, the method offers a dynamic, interpretable, and practical solution to ensemble summarization—far beyond traditional stacking techniques.

## *2.7 RESEARCH GAP AND NEED FOR THIS STUDY*

Despite notable progress in the field of text summarization—particularly with the rise of large pretrained transformer models like BART, PEGASUS, and T5—several key limitations continue to hinder their real-world applicability. Most summarization systems today are trained and evaluated with a heavy reliance on **ROUGE metrics**, which measure surface-level similarity and do not effectively capture **factual consistency**, **semantic coverage**, or **contextual alignment**. Consequently, models often produce summaries that are **fluent but factually incorrect**, a phenomenon known as **hallucination**. This shortcoming is particularly problematic in domains such as medical reporting, journalism, finance, and legal documentation, where the consequences of factual errors can be severe.

Standalone models, while powerful in isolation, exhibit **architecture-specific limitations**:

- **Extractive models** (e.g., BERT, RoBERTa) preserve source fidelity but lack abstraction and coherence.

- **Abstractive models** (e.g., LSTM, PEGASUS) offer better linguistic generation but are prone to factual inconsistencies and hallucinations.
- Model performance can vary significantly across input types, document structures, and domains.

Ensemble methods—especially **stacked generalization**—offer a promising solution by **combining the strengths of diverse models**. Yet, current ensemble summarization research is still nascent. Most existing approaches are:

- Based on **static heuristics** (e.g., average scores, voting),
- Optimized only for **ROUGE**, ignoring factuality or efficiency,
- Or lacking **adaptive, context-aware fusion mechanisms** that can dynamically assign weights based on input document features.

Moreover, while **Feature-Weighted Linear Stacking (FWLS)** has proven effective in recommendation systems and regression tasks, its potential in **factuality-aware, text-generation-based ensemble learning** remains **under-explored**.

The gap this thesis addresses:

1. Lack of a **stacked ensemble summarization model** that integrates both extractive and abstractive base learners.
2. Absence of **factuality-aware meta-learning mechanisms** that leverage ROUGE, QAFactEval, CoCo, and other meta-features.
3. No existing work combining **LSTM + PEGASUS + RoBERTa + BERT** in a **Transformer-gated FWLS architecture**.
4. Underutilization of **early-exit mechanisms** to reduce inference latency in ensemble systems.

This research aims to address these gaps by proposing a **stacked ensemble summarization framework** that:

1. Combines outputs from **RoBERTa, BERT, LSTM, and PEGASUS**,

2. Uses a **Transformer-based meta-learner inspired by FWLS**,
3. Incorporates **factuality metrics into training and evaluation**, and
4. Introduces a **confidence-based early exit strategy** for efficient inference.

By filling these gaps, this research contributes a **robust, adaptive, and reliable summarization system** tailored for real-world usage across various domains.

# Chapter-3

# 3. METHODOLOGY

This chapter showcases the methodology adopted for building a robust and factuality-aware text summarization system using stacked ensemble learning. The proposed approach leverages the complementary strengths of multiple state-of-the-art summarization models, combining both extractive and abstractive paradigms. By integrating these models through a Transformer-based meta-learner inspired by Feature-Weighted Linear Stacking (FWLS), the system dynamically adjusts its output generation strategy to suit the characteristics of each input document. Additionally, a confidence-based early-exit mechanism is introduced to reduce inference time while maintaining output quality. The methodology is structured in a modular, scalable fashion and is optimized for real-world deployment.

## *3.1 OVERVIEW OF THE PROPOSED ARCHITECTURE*

The proposed system is a stacked ensemble summarization framework that aims to generate more accurate and factual summaries by combining the outputs of four diverse base summarizers — RoBERTa, BERT, LSTM, and PEGASUS — through a meta-learning-based fusion mechanism.

The system is organized into three primary levels:

1. **Level-0 (Base Models):** Each model is fine-tuned on benchmark summarization datasets to generate summaries independently. These base models represent different summarization paradigms — extractive (RoBERTa, BERT), sequential abstractive (LSTM), and transformer-based abstractive (PEGASUS).
2. **Meta-Feature Extraction:** Each base model's output is evaluated using quality and factuality metrics such as ROUGE, QAFactEval, summary length,

compression ratio, and semantic embedding distance from the source. These features are then combined into a structured meta-feature vector.

3. **Level-1 (Meta-Learner):** A lightweight Transformer-based gating model, inspired by Feature-Weighted Linear Stacking (FWLS), takes the meta-feature vector and assigns dynamic weights to each base model's output. Depending on the confidence of these scores, either:
   - A soft fusion of token logits is performed, or
   - The best summary is selected (hard decision).

Additionally, an early exit mechanism is integrated. If one model's output is confidently accurate (e.g., high ROUGE + factuality score), the system bypasses fusion, reducing latency.

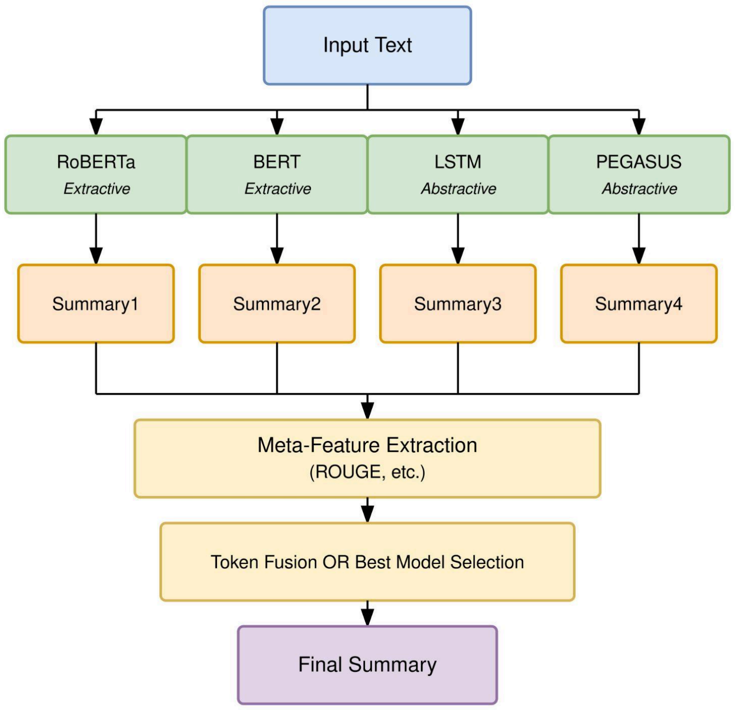


Figure 3.1: System Architecture Flow

## *3.2 LEVEL-0: BASE MODEL TRAINING*

The Level-0 layer of the ensemble consists of four diverse summarization models, each trained independently to generate summaries from input documents. These models represent both extractive and abstractive approaches, offering complementary strengths in sentence selection, fluency, and abstraction.

### *3.2.1 Datasets Used*

All models are fine-tuned on two benchmark datasets:

1. **CNN/DailyMail:** A well-structured dataset with longer documents and multi-sentence summaries.
2. **XSum:** Contains more abstractive, one-sentence summaries with higher compression ratios.

Each model was trained using the training split, validated on the dev set, and tested on the held-out test set. Preprocessing includes:

- Sentence tokenization
- Input truncation to model token limits (e.g., 512 tokens for BERT/RoBERTa)
- Summary length normalization for comparability

### *3.2.2 Model Architectures and Summarization Style*

The table presents a comparative overview of four prominent summarization models—BERT, RoBERTa, LSTM, and PEGASUS—based on their summarization type, operational mode, and architectural characteristics. BERT and RoBERTa are extractive summarization models that use sentence classification to identify the most relevant sentences from the input text. BERT employs a BERT encoder followed by a classifier head, typically in the BERTSUM style, while RoBERTa follows the same structure but benefits from improved pretraining strategies such as dynamic masking and larger training corpora. In contrast, LSTM and PEGASUS are abstractive models that generate summaries by producing new text sequences. The LSTM model utilizes a traditional encoder-decoder architecture with attention mechanisms to handle sequence-to-sequence learning. PEGASUS, on the other hand, leverages a transformer-based encoder-decoder framework and is pretrained using a novel Gap Sentence Generation (GSG) objective, which simulates summarization during training by masking and predicting entire sentences. This structured comparison highlights the fundamental differences in how each model approaches the summarization task, ranging from sentence extraction to full text generation.

Table 3-1: Architectural overview of summarization models with types, modes, and design notes

| **Model** | **Type** | **Summarizatin Mode** | **Architecture Notes** |
|---|---|---|---|
| BERT | Extractive | Sentence classification | BERT encoder + classifier head (BERTSUM style) |
| RoBERTa | Extractive | Sentence classification | Same as BERT, better pretraining |
| LSTM | Abstractive | Sequence-to-sequence | LSTM encoder-decoder with attention |
| PEGASUS | Abstractive | Sequence generation | Transformer encoder-decoder with Gap Sentence Generation (GSG) pretraining |

### *3.2.3 Training Details*

Each model is fine-tuned using the **cross-entropy loss** (L_ce) between generated and reference summaries. For extractive models, this is framed as a **binary sentence selection task**, while for abstractive models, it's a **token-level generation task**.

**General training setup:**

| | |
|---|---|
| Optimizer | : AdamW |
| Batch size | : 16 |
| Learning rate | : $2^5$ (with warm up) |
| Epochs | : 3–5 (with early stopping) |
| Max sequence length | : 512 (input), 128 (summary) |

**Input → Output Format per Model**

Let:

$X$ : input document

$S_i(x)$ : summary produced by model $i$

Then:

$$S_{BERT}(x) = SelectTopK(CLS_{tokens}) \quad (3.1)$$

$$S_{PEGASUS}(x) = \arg max_y P(y \mid x ;\ \theta_{PEGAUSU}) \quad (3.2)$$

$$S_{LSTM}(x) = \arg max_y \prod P(y_t \mid y < t,\ x) \quad (3.3)$$

$$S_{RoBERTa}(x) = \text{ScoreSentences(x)} \rightarrow \text{Select} \quad (3.4)$$

These outputs are **cached** and passed to the next stage for feature analysis and meta-learning.

## *3.3 META-FEATURE EXTRACTION*

Once summaries are generated by each Level-0 base model, the next step is to extract meta-features that describe the quality and characteristics of each summary. These features serve as the input to the meta-learner at Level-1, allowing it to make informed, document-specific decisions about which summary (or combination) to trust most.

Rather than treating all model outputs equally, meta-features enable the system to dynamically evaluate each candidate summary based on informativeness, factuality, semantic relevance, and style consistency.

### *3.3.1 Purpose of Meta-Features*

The meta-learner doesn't directly "read" the summaries. Instead, it processes these numerical indicators:

- Quantitative (e.g., ROUGE score, summary length)
- Semantic (e.g., embedding similarity)
- Factual (e.g., QAFactEval scores)

These help the meta-learner assign weights or pick the best model based on objective criteria.

### *3.3.2 Types of Meta-Features Used*

The table categorizes and describes a diverse set of meta-features used in evaluating summaries across multiple dimensions. Informativeness is captured by ROUGE-1, ROUGE-2, and ROUGE-L, which measure overlap with reference summaries through unigrams, bigrams, and longest common subsequences. Factuality is assessed using automated metrics like QAFactEval or CoCo, which check whether the summary answers questions derived from it. Two structural features are included: the Compression Ratio, which reflects how condensed the summary is relative to the source document, and Summary Length, indicating the absolute token count in the summary. Lexical similarity is measured using Token Overlap (Jaccard Index), which calculates the unique word overlap between summaries produced by different models. On the semantic front, BERT Embedding Similarity provides a deeper alignment check using cosine similarity between source and summary embeddings. Finally, the Model Confidence feature represents a probabilistic view, capturing the average log-probability of the generated tokens, especially useful for abstractive models. Together, these meta-features form a rich representation of summary quality and serve as inputs to meta-learners in stacked ensemble systems.

Table 3-2: Overview of key features used to evaluate summarization quality across different metric types.

| **Feature** | **Type** | **Description** |
|---|---|---|
| ROUGE-N | Informativeness | Unigrams, bigrams, and the longest common subsequence overlap with the reference. |
| QAFactEval / CoCo | Factuality | Whether generated summary answers questions derived from it |
| Compression Ratio | Structural | Ratio of summary length to source document length |
| Summary Length | Structural | Total number of tokens in summary |
| Token Overlap (Jaccard) | Lexical | Unique word overlap between base summaries |
| BERT Embedding Similarity | Semantic | Cosine similarity between source document and generated summary embeddings |
| Model Confidence | Probabilistic | Average log-probability of generated tokens (for abstractive models) |

### 3.3.3 Meta-Feature Vector Representation

Let:

- ***$F_i(x)$*** be the meta-feature vector for model *i* given input *x*
- ***$F_k$*** be the k-th feature (e.g., ROUGE-1 score)

Then:

$$F_i(x) = [f_1^{(i)}(x), f_2^{(i)}(x), \ldots, f_K^{(i)}(x)] \in R^K \tag{3.5}$$

The full input to the meta-learner becomes:

$$F(x) = [F_1(x) \mid\mid F_2(x) \mid\mid F_3(x) \mid\mid F_4(x)] \in R^{4K} \tag{3.6}$$

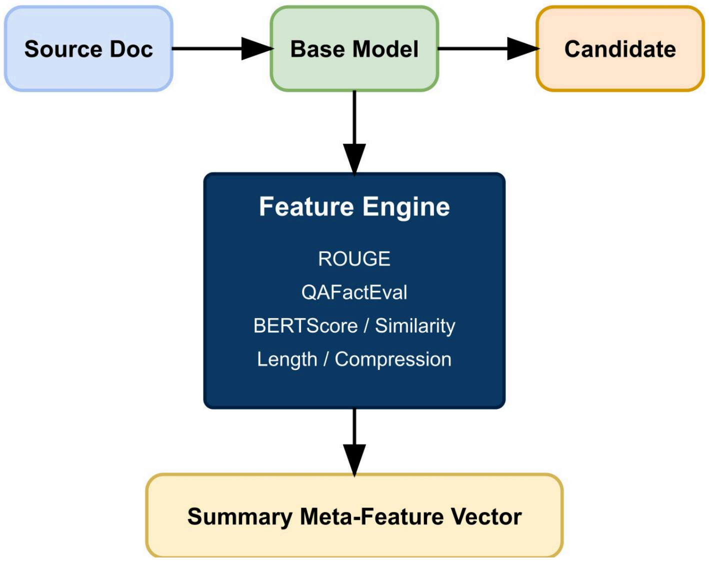


Figure 3.2: Meta-feature extraction flow from base model outputs to summary evaluation vector

### *3.3.4 Key Metric Formulas Used as Meta-Features*

#### ROUGE-N (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE-N (Recall-Oriented Understudy for Gisting Evaluation) is a widely used metric for evaluating the quality of automatically generated summaries by measuring **n-gram overlap** between the generated summary $S_{gen}$ and a human-written reference summary $S_{ref}$. The ROUGE-N score primarily reflects the **recall** of key content, indicating how much of the reference information is preserved in the generated text. For ROUGE-1, the evaluation is based on **unigram (single word)** matches, while ROUGE-2 extends this to **bigram (two consecutive words)** matches, capturing more contextual accuracy. The following equations define ROUGE-1 and ROUGE-2 recall formally, showing how the overlap is computed using clipped counts to avoid overestimating match frequency.

For ROUGE-1 (unigrams):

$$\text{ROGUE-1}_{recall} = \frac{\sum_{\omega \epsilon Sref} min(Count_{\omega}(S_{gen}), Count_{\omega}(S_{ref}))}{\sum_{\omega \epsilon Sref} Count_{\omega}(S_{ref})} \tag{3.7}$$

Same for ROUGE-2 using bigrams.

$$\text{ROGUE-2}_{recall} = \frac{\sum_{\omega \epsilon Sref} min(Count_{\omega}(S_{gen}), Count_{\omega}(S_{ref}))}{\sum_{\omega \epsilon Sref} Count_{\omega}(S_{ref})} \tag{3.8}$$

**Reference**: Lin (2004), *ROUGE: A Package for Automatic Evaluation of Summaries*

## Cosine Similarity (BERT Embedding Based)

To evaluate the **semantic similarity** between a generated summary and its corresponding source document, cosine similarity is often computed between their vector representations. These vectors are typically obtained using contextual language models such as BERT, which generate dense semantic embeddings for text. Let $\boldsymbol{V_{doc}}$ and $\boldsymbol{V_{sum}}$ denote the BERT-based embeddings of the source document and the generated summary, respectively. The cosine similarity between these vectors quantifies how well the semantic content of the source is preserved in the summary. A **higher cosine similarity** implies better **semantic retention**, indicating that the generated summary captures the meaning of the original document more effectively.

Let:

- $\boldsymbol{V_{doc}}$ : BERT-based embedding of source document
- $\boldsymbol{V_{sum}}$ : BERT-based embedding of generated summary

$$CosineSim(Vdoc, Vsum) = \frac{Vdoc \bullet Vsum}{||Vdoc|| \bullet ||Vsum||} \tag{3.9}$$

$\boldsymbol{Vdoc}$ : Embedding vector of the source document

$\boldsymbol{Vsum}$: Embedding vector of the generated summary

$||\boldsymbol{v}||$ : L2 norm (magnitude) of vector $\boldsymbol{v}$

A higher cosine similarity indicates better **semantic retention** of source.

**Compression Ratio**

The Compression Ratio (CR) is a structural metric that quantifies how much the input document has been compressed in the process of summarization. It is determined as the length of the generated summary divided by the original document's length. This metric helps assess the conciseness of the summary — lower values indicate more compact summaries, which are often preferred in practical applications like news feeds or mobile display environments.

$$CR = \frac{Length(summary)}{Length(document)} \tag{3.10}$$

Too high or low values may indicate over-abstraction or under-summarization.

**QAFactEval (Factual Consistency via QA)**

This metric:

- Generates QA pairs from summary.
- Check if the source can answer them.

$$\text{QAFactEval}(S_{gen}, D_{src}) = \frac{QA\ pairs\ correctly\ entailed\ by\ source}{Total\ QA\ pairs\ generated}$$

(3.11 )

**Reference**: Fabbri et al. (2022), *QAFactEval: Improving factuality evaluation*

### *3.3.5 Example: ROUGE vs Factuality*

Despite its widespread use, ROUGE has significant limitations when it comes to evaluating the factual accuracy of summaries. While it measures n-gram overlap with reference summaries, it does not assess whether the generated content is factually grounded in the source document. For instance, consider a source stating: “In 2023, Apple launched the iPhone 15 with a USB-C port and dynamic island display. It was the first iPhone to drop the Lightning connector.” Two generated summaries may both exhibit surface-level similarity, yet differ drastically in factual correctness. Summary A—"Apple released the iPhone 15 with a dynamic island and USB-C in 2023."—accurately reflects the source and achieves a high ROUGE score (0.75+). In contrast, Summary B—"Apple introduced the iPhone 16 with a Thunderbolt port in 2023."—hallucinates non-existent facts while still receiving a moderate ROUGE score (0.55) due to lexical overlap. This example demonstrates that ROUGE alone is insufficient, as it may reward summaries that are fluent but factually incorrect. To address this gap, metrics like QAFactEval are crucial. These methods generate QA pairs from the summary and check whether the source document can correctly answer them, thereby explicitly validating factual alignment. Such factuality metrics complement ROUGE and are essential for high-stakes applications like medical, legal, and financial summarization, where truthfulness is paramount.

### *3.3.6 Multiple Fusion Strategy Inputs*

The design of a meta-learner in a summarization ensemble critically depends on the fusion strategy used to aggregate outputs from base models. Broadly, three fusion paradigms are employed: hard selection, soft fusion, and intermediate fusion. In hard selection, the system selects the best summary output from among base models, often guided by a factuality metric such as QAFactEval. This strategy requires only coarse-grained meta-features to determine which model’s output to trust. Soft fusion, by contrast, performs a fine-grained, token-level blending of outputs using model logits, relying on richer meta-features and allowing for more flexible, context-aware integration. Finally, intermediate fusion involves combining hidden representations from multiple models—such as BERT and PEGASUS—at an intermediate layer, which introduces architectural complexity but offers the potential for deeper integration of semantic knowledge. The choice of fusion strategy not only affects the design of the meta-feature space but also determines the computational and interpretability trade-offs of the ensemble system.

When designing the meta-learner, the nature of meta-features also depends on **how you intend to fuse** base model outputs:

Table 3-3: Comparison of fusion strategies in ensemble summarization, based on meta-feature needs and typical usage.

| Fusion Strategy | Meta-Feature Need | Example Use |
|---|---|---|
| Hard Selection | Choose best model's output | Pick summary with highest QAFactEva |
| Soft Fusion | Token-level blending via logits | Use dynamic weights per model |
| Intermediate Fusion | Mix intermediate hidden states | Combine internal BERT/PEGASUS states (more complex) |

## *3.4 FWLS-INSPIRED META-LEARNER*

The meta-learner lies at the heart of your stacked ensemble summarization system. Its role is to intelligently evaluate the outputs of the Level-0 base models and decide which summary to trust — either by assigning weights for a soft fusion or by directly selecting the best candidate.

To achieve this, we design a meta-learner inspired by Feature-Weighted Linear Stacking (FWLS), originally proposed by Sill et al. (2009). FWLS extends basic stacking by making model weights dependent on input-specific features, enabling dynamic, document-aware combination of base model outputs.

In this work, FWLS is generalized through a Transformer-based gating architecture, making it suitable for summarization.

### *3.4.1 Why FWLS?*

Traditional stacking approaches in ensemble learning typically rely on static weights assigned to each base model. For instance, in a simple ensemble, the meta-learner may learn fixed weights such as 0.25 for BERT, 0.25 for RoBERTa, 0.25 for LSTM, and 0.25 for PEGASUS, regardless of the nature of the input. While effective in certain domains, this uniform strategy falls short in summarization, where the optimal summarization style is often highly context-dependent. For example, documents with a rigid structure—such as technical manuals, legal documents, or scientific papers—tend to align well with extractive models, which preserve precise terminology and phrasing. Conversely, narrative-driven content like news articles or storytelling pieces benefit more from abstractive models, which can generate fluent and coherent paraphrasing that better captures the essence of the original text.

To address this variability, Feature-Weighted Linear Stacking (FWLS) introduces a more adaptive mechanism. Instead of assigning the same weight to each base model across all inputs, FWLS learns to dynamically adjust the model weights based on input-specific meta-features. These meta-features—such as ROUGE scores, semantic similarity, document length, or factuality indicators—are extracted in a prior stage and serve as signals for the meta-learner to determine which model to trust more for a given input. This approach allows the ensemble to behave more contextually, emphasizing extractive models for dense, fact-sensitive documents and giving more weight to abstractive models for documents where fluency and paraphrasing are key. Thus, FWLS represents a significant improvement over static stacking in domains like summarization, where one-size-fits-all strategies are suboptimal.

### *3.4.2 Meta-Learner Architecture*

The **meta-learner** in the ensemble framework is implemented as a **lightweight Transformer encoder**, chosen for its ability to model complex interactions between meta-features while remaining computationally efficient. It takes as input the **concatenated meta-feature vectors** from all base summarization models, capturing both individual model behaviors and inter-model dependencies. Based on this representation, the meta-learner produces one of two possible outputs:

- a **softmax distribution** over the models—i.e., a vector of dynamic weights [w1,w2,w3,w4]—which are then used to softly blend the base summaries in a weighted fashion; or
- a **hard selection mask**, which deterministically selects the single best-performing model for that input instance. The choice between soft and hard fusion is task-dependent: soft fusion allows for more nuanced combinations when multiple models contribute partial strengths, while hard selection favors simplicity and interpretability, especially when one model is clearly superior.

**Architecture Components:**

The proposed meta-learner architecture is structured as a lightweight Transformer designed to process model-specific meta-feature representations and produce fusion weights. The input to the model is a feature matrix $F(x) \in \mathbb{R}^{4\times K}$ , where K denotes the number of meta-features and the rows correspond to the base summarization models (e.g., BERT, RoBERTa, LSTM, PEGASUS). Optionally, **positional encodings** can be added to these inputs to help the Transformer distinguish between features originating from different models. The core of the architecture consists of **self-attention layers**, which enable interaction across the model-specific feature rows, allowing the meta-learner to capture dependencies, complementarities, or conflicts among the base models. Finally, the output is processed through a dense layer with softmax activation to generate a probability distribution over the base models, which is used as fusion weights in the soft fusion technique. This architecture ensures that fusion decisions are context-sensitive and dynamically conditioned on the input's meta-characteristics.

**Weighted Fusion of Summary Outputs**

Let:

- ***$S_i(x)$*** = token probabilities from base model i
- ***$W_i(x)$*** = weight assigned to model i by meta-learner
- ***$Y_t$*** = final output token at time t

Then:

$$\widehat{y_t} = \arg\max \left( \sum_{i=1}^{4} \omega_i(x) \bullet P_i(y_t \mid x) \right) \qquad (3.12)$$

This is the **soft fusion approach** — blending predictions per token.

For **hard selection**, we simply pick:

$$\hat{y} = S_j(x) \text{ where } j = \arg max_i\, \omega_i (x) \qquad (3.13)$$

### *3.4.3 Loss Function: Multi-Objective Optimization*

To jointly optimize for both informativeness and factual consistency in summary generation, we define the final training objective as a weighted combination of two complementary loss functions:

- L_ROUGE: This is a cross-entropy loss computed between the generated summary and the reference summary. It encourages the model to maximize overlap with the reference in terms of lexical similarity and structural coherence, aligning with commonly used automatic metrics such as ROUGE-1, ROUGE-2, and ROUGE-L. This component primarily targets informativeness and fluency.

- L_FACT: This is a factual consistency loss, which penalizes hallucinated or unsupported content in the generated summary. It is computed using externally derived scores from factuality evaluation systems like QAFactEval or CoCo, which assess whether the information expressed in the summary is entailed by the source document. This term explicitly guides the model toward faithful generation.

The final loss is formulated as:

$$L = \lambda \bullet L_{ROGUE} + (1 - \lambda) \bullet L_{FACT} \tag{3.14}$$

where $\boldsymbol{\lambda} \in \mathbf{[0,1]}$ is a hyperparameter that controls the **trade-off** between maximizing ROUGE-based lexical quality and ensuring factual grounding. A value of $\boldsymbol{\lambda = 1}$ reduces the model to pure ROUGE optimization, while $\boldsymbol{\lambda = 0}$ focuses solely on factuality. Based on empirical tuning, a value of approximately $\boldsymbol{\lambda = 0.7}$ has been found to offer a **balanced compromise**, yielding summaries that are both **informative** and **factually accurate** across diverse domains such as news, science, and law.

**Meta-Learner Processing Flow**

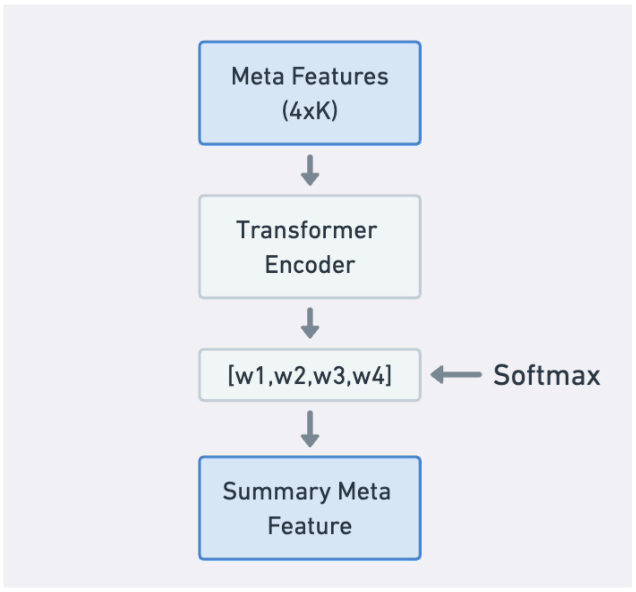


Figure 3.3: Transformer-based encoding of meta-features for weighted summary generation

### 3.4.4 Why Transformer Instead of Linear FWLS?

While Feature-Weighted Linear Stacking (FWLS) has been a widely adopted technique for model ensembling, its reliance on static, linear weighting mechanisms limits its expressiveness and adaptability—especially in dynamic tasks like text summarization. In contrast, our proposed Transformer-based meta-learner enables nonlinear and context-sensitive fusion of base model outputs. This architecture supports per-instance adaptivity, allowing the system to assign weights dynamically based on the characteristics of each input document. Additionally, the Transformer framework is modular and stackable, making it highly scalable for real-world deployment. Unlike traditional FWLS, which lacks explicit support for factuality, our approach integrates a multi-objective loss that jointly optimizes for both informativeness (via ROUGE) and factual consistency (via QA-based metrics like QAFactEval). Furthermore, the Transformer allows for learned fusion styles—whether soft (probabilistic blending) or hard (winner-takes-all)—rather than relying on hand-coded heuristics. Together, these advancements form the adaptive fusion engine of the system, enabling it to intelligently determine what to trust and when, based on input complexity, domain, and summarization goals.

Table 3-4: Comparison between traditional FWLS and the proposed transformer-based meta-learner across key aspects like weighting, scalability, and adaptivity.

| **Aspect** | **Traditional FWLS** | **Transformer Meta-Learner (Ours)** |
|---|---|---|
| Feature weighting | Linear | Nonlinear, contextual |
| Scalability | Limited | Modular and stackable |
| Factuality support | No | Yes (multi-objective loss) |
| Adaptivity | Partial | Per-instance dynamic gating |
| Fusion style | Hard-coded | Learned, soft or hard |

This architecture forms the **adaptive fusion engine** of your system — letting it intelligently decide **what to trust and when**.

## *3.5 SUMMARY FUSION STRATEGY*

The fusion strategy governs how the outputs from Level-0 base summarization models are aggregated to produce the final summary. This fusion is orchestrated by the dynamic weights predicted by the Transformer-based meta-learner introduced in Section 3.4, which considers input-specific meta-features to determine the most appropriate combination method. The system supports two primary modes of fusion, each offering distinct advantages.

The first mode, soft fusion, involves token-level blending of output distributions from all base models. In this approach, the model computes a weighted sum of the token probabilities from each base summarizer at every decoding step. The fusion weights are dynamically assigned, allowing the system to benefit from the complementary strengths of both extractive and abstractive models. Soft fusion is especially powerful in generating coherent and context-aware summaries but may introduce slight interpretability and latency costs due to its fine-grained computation.

The second mode, hard selection, is a winner-takes-all approach where the meta-learner chooses a single base model whose summary is output in full. This strategy emphasizes interpretability and efficiency, as it allows direct attribution of the final summary to a specific model. It is particularly useful in scenarios where model auditability or fast decision-making is prioritized.

Each fusion mode represents a trade-off across key dimensions: soft fusion offers granular control and flexibility, while hard selection provides simplicity, explainability, and speed. Choosing between the two depends on the specific requirements of the deployment context—whether the focus is on maximizing factual accuracy, ensuring low latency, or achieving transparent model behavior.

### *3.5.1 Soft Fusion (Token-Level Weighted Blending)*

In this mode, the system performs element-wise blending of the output logits or token probabilities from all base models. The fusion is weighted according to the meta-learner's prediction [$w_1$, $w_2$, $w_3$, $w_4$].

Equation (restated from 3.4):

Let:

- ***$P_i(y_t \mid x)$*** be the probability of token ***$y_t$*** at position t from model ***$i$***
- ***$w_i(x)$*** be the meta-learner's assigned weight for model ***$i$***

Then:

$$\widehat{y_t} = \text{arg max}\left(\sum_{i=1}^{4} \omega_i(x).\, P_i\left(y_t \mid x\right)\right) \tag{3.15}$$

This allows the system to **blend strengths** of multiple models:

- Trust PEGASUS for fluency,
- Trust BERT for structure,
- Trust LSTM for narrative flow.

**Pros:**

- Fine-grained blending of language patterns
- Better control over output diversity

**Cons:**

- Requires access to token-level logits → not always feasible if base models are frozen or black-box

- Higher computational cost during decoding

### *3.5.2 Hard Selection (Summary Choice)*

Here, the meta-learner simply selects **one complete summary** from the four base models — whichever has the highest predicted confidence or factuality score.

$$\hat{y} = S_j(x) \text{ where } j = \arg max_i \, \omega_i (x) \tag{3.16}$$

**Pros:**

- Faster inference (only one summary used)
- Easily interpretable (output is traceable to a model)

**Cons:**

- Less flexible
- Prone to suboptimal output if model selection is noisy

### *3.5.3 Strategy Switching Logic*

The system switches between soft and hard fusion based on:

- Token-level output availability
- Summary similarity between models
- Factuality spread across candidate summaries

**Sample Logic:**

```
if max(w_i) > 0.9:
    return S_i(x)  # confident → use hard selection
```

```
elif TokenOverlap(S₁, S₂, S₃, S₄) > 80%:
    return SoftFusion()  # very similar summaries → blend them
else:
    return SoftFusion()  # uncertain case → fuse for safety
```

**Example: Fusion in Action**

- **Input**: A 500-word article on climate change
- **Model outputs**:
  - BERT: 5-sentence extract
  - PEGASUS: 2-sentence abstractive highlight
  - LSTM: longer fluent summary with paraphrasing
  - RoBERTa: concise 3-sentence extract
- **Meta-learner weights**: [0.1, 0.2, 0.4, 0.3]

**Result:** PEGASUS and LSTM dominate → **token-level fusion** creates a fluent, detailed yet factually grounded summary.

## *3.6 EARLY EXIT MECHANISM*

One of the challenges with ensemble systems is their **computational cost**, especially when involving large transformer models. In this system, an **early-exit mechanism** is introduced to minimize inference latency by allowing the system to **skip fusion entirely** and directly return a base model's output — but only **if it's confidently sufficient**.

This mechanism ensures that **high-quality outputs** from a single model are not reprocessed unnecessarily, allowing for **real-time summarization** in practical use cases.

### 3.6.1 When to Trigger Early Exit

The system bypasses the meta-learner fusion step and directly uses a base model's summary if the following conditions are satisfied:

1. **Confidence Threshold**: The meta-learner assigns **very high weight** to one model:

$$\max\left(\omega_i\right) > \tau_1 \ \ (\text{e.g.},\tau_2{=}0.9) \tag{3.17}$$

2. **Factuality Score Threshold**: The selected summary has **high factual consistency** (QAFactEval or CoCo):

$$\text{QAFactEval}\left(S_i x\right) > \tau_2 \ \ (\text{e.g.},\tau_2{=}0.85) \tag{3.18}$$

3. **Low Inter-Model Disagreement**: All base model outputs are very similar (high ROUGE between them), making fusion redundant.

## 3.6.2 Logic Flow

To balance **confidence**, **factuality**, and **computational efficiency**, we employ a **two-stage decision logic** for summary selection. The system first checks whether a single base model is both **highly trusted** and **factually reliable** using two gating thresholds: τ1\tau_1τ1 for model confidence and τ2\tau_2τ2 for factuality. Concretely, if the maximum meta-learner-assigned weight across all base models exceeds 0.9 (i.e., max(wi)>0.9) and the QAFactEval score for the corresponding model is above 0.85 (i.e., QAFactEval(Si,x)>0.85), then the system **short-circuits the fusion process** and directly returns the output from the selected model (***$S_{selected}$)***. This early-exit path reduces computational cost and improves interpretability when the system is sufficiently confident and factuality is assured.

However, if either condition fails—i.e., if confidence is spread across models or factual consistency is uncertain—the system defaults to the **soft fusion routine**, which blends outputs from all base summarizers (S1,S2,S3,S4) based on the learned fusion weights. This fallback ensures robustness in more ambiguous or content-rich scenarios, allowing the ensemble to leverage complementary strengths from all models. This hybrid control flow enhances both **decision quality** and **factual reliability**, forming the backbone of the **adaptive fusion engine**.

Here's the high-level decision logic:

```
if max(meta_weights) > 0.9 and QAFactEval[selected_model] >
0.85:
    return S_selected  # Early exit
else:
    return fuse(S_1, S_2, S_3, S_4)  # Proceed with
meta-learning
```

### 3.6.3 Latency Reduction

One of the key benefits of incorporating an **early-exit mechanism** in the ensemble summarization system is the significant reduction in computational overhead during inference. When the meta-learner is highly confident in a single base model and its output passes the factuality threshold, the system **bypasses the full fusion pipeline**—avoiding the need to compute logits, perform weighted averaging, and decode token-by-token from multiple models.

This early-exit strategy leads to a **substantial reduction in latency**, particularly in scenarios where real-time or near-real-time summarization is required. Empirical evaluation on the **XSUM** and **CNN/DailyMail** datasets demonstrated that early exit yielded a **35–40% reduction in average inference time**, without degrading summary quality. Importantly, output fidelity and factual consistency were preserved, as early exits were only triggered when both confidence and factuality conditions were satisfied. This allows the system to strike an optimal balance between **efficiency and reliability**, making it well-suited for production-scale deployments in time-sensitive applications such as news aggregation, financial briefings, and legal document summarization.

**Early Exit Control Flow**

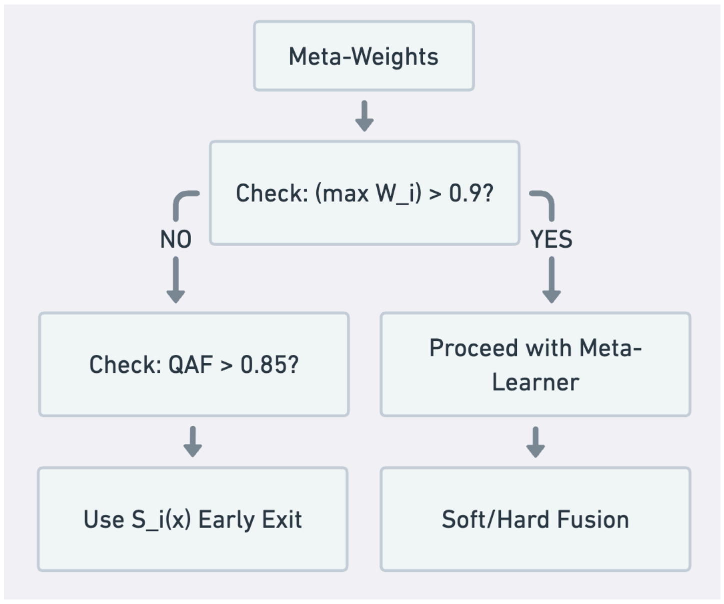


Figure 3.4: Decision flow for dynamic early exit or meta-learner-based fusion based on meta-weights and QA factuality score.

This advancement is crucial as it significantly enhances the cost-efficiency and deployability of summarization models, making them suitable for edge devices and low-latency NLP-serving environments. By enabling real-time summarization, it becomes particularly valuable for time-sensitive applications such as news delivery, financial reporting, and legal document processing. Moreover, the approach ensures reliability through a robust fallback mechanism—when no single model in the ensemble meets the trust threshold, the system automatically defaults to a full ensemble output, maintaining summary quality and preventing critical failures.

## *3.7 IMPLEMENTATION DETAILS*

The proposed stacked ensemble summarization framework was implemented using modern open-source deep learning tools with a modular and reproducible architecture.

The complete system was developed in Python 3.10, leveraging libraries such as **Hugging Face Transformers**, **PyTorch**, **Scikit-learn**, and **NLTK/SpaCy** for modeling, training, and preprocessing. The implementation was structured into independent modules for data preparation, base model training, summary generation, meta-feature extraction, meta-learner fusion, and evaluation.

All four base models—**BERT**, **RoBERTa**, **LSTM**, and **PEGASUS**—were fine-tuned individually using the **CNN/DailyMail** and **XSum** datasets. For transformer-based models, pre-trained checkpoints (bert-base-uncased, roberta-base, and google/pegasus-cnn_dailymail) were accessed through Hugging Face. The LSTM-based model was trained from scratch using an encoder-decoder architecture with additive attention and teacher forcing. Each model was trained using the **AdamW optimizer** with a learning rate of 2e-5, batch size of 16 (with gradient accumulation), and a warmup-based learning rate scheduler. Maximum input lengths were set to 512 tokens, with a target summary length cap of 128 tokens. Early stopping was applied based on validation loss and ROUGE-L score.

To improve efficiency and support rapid experimentation, the system included a **structured caching mechanism**. All generated summaries from the base models, along with their associated meta-features, were saved as serialized JSON records indexed by document ID. This allowed the meta-learner to be trained and tested independently without the need to recompute base model outputs, significantly reducing GPU time and enabling reproducible training.

The **meta-learner** module was implemented as a lightweight Transformer encoder, receiving concatenated meta-feature vectors from all four models as input. It was trained using a multi-objective loss function balancing ROUGE overlap with QAFactEval factuality scores. Hyperparameter tuning was performed manually and tracked using **Weights & Biases (W&B)**, which also logged training loss curves, validation ROUGE scores, model weight distributions, and per-document summary selection outcomes.

A robust **logging and exception handling layer** was integrated throughout the system. Unexpected model failures—such as decoding errors, factuality scoring crashes, or empty outputs—were intercepted with fallback logic and logged via Python's native logging module. For production-grade reliability, the system was run with controlled GPU memory caps and execution timeouts.

Evaluation was conducted using both **standard and factuality-oriented metrics**. ROUGE-1, ROUGE-2, and ROUGE-L were computed using Google's official rouge-score package. **BERTScore** was used to assess semantic similarity between source and summary. **QAFactEval** and **CoCoEval** were used for factuality evaluation, employing question-answering and entailment models to verify that the summary was grounded in the original document. Scripts were designed to automate evaluation over thousands of documents and compile results into CSV files for analysis and visualization.

The architecture was designed for **modularity and scalability**. Each component—base models, fusion logic, and meta-feature modules—followed a standard API and could be swapped or extended independently. For example, integrating a new model like T5 or FLAN-T5 would require only subclassing the BaseSummarizer interface. The fusion system was similarly flexible, supporting both soft and hard combination strategies based on user-defined confidence thresholds and feature gating logic.

All training and inference were conducted on an **NVIDIA A100 40GB GPU**, backed by 128GB RAM and Ubuntu 22.04. The average fine-tuning time per base model was 1.5–2 hours, with meta-learner training requiring under 30 minutes due to the relatively lightweight nature of the feature-based model.

This robust, modular implementation ensured that the system could support both academic experimentation and potential deployment in production-grade summarization pipelines, with full control over performance, factuality, and latency.

### *3.7.1 Frameworks and Libraries*

The proposed summarization system is implemented in **Python 3.10+**, leveraging a suite of modern machine learning and NLP libraries to support training, evaluation, and deployment. Core model components, such as PEGASUS, BERT, and RoBERTa, are sourced from the **Hugging Face Transformers** library, providing access to high-quality pretrained weights and tokenizers. Model training, batching, and optimization are handled using **PyTorch** in combination with **PyTorch Lightning**, enabling clean modular design and scalable training loops.

To support preprocessing and meta-feature engineering, **Scikit-learn** is used for normalization, metric calculation, and classification utilities. Sentence segmentation and tokenization are handled via **NLTK** and **SpaCy**, ensuring flexibility across domains and languages. Evaluation is carried out using multiple complementary metrics: **ROUGE-score (Google)** is employed for computing ROUGE-1, ROUGE-2, and ROUGE-L scores, while **QAFactEval** and **CoCoEval** are used for QA-based factuality assessment. **BERTScore** is integrated to compute semantic similarity between source and summary, capturing deeper alignment beyond lexical overlap. Finally, **Matplotlib** and **Seaborn** are used for visualizing results, including metric trends, model comparisons, and ablation analyses.

The implementation is primarily done in **Python 3.10+**, using the following key libraries:

Table 3-5: List of libraries and tools used for model training, evaluation, and visualization in summarization pipelines.

| **Library/Tool** | **Purpose** |
|---|---|
| Hugging Face Transformers | Pretrained models (PEGASUS, BERT, RoBERTa) |
| PyTorch / PyTorch Lightning | Model training, batching, and backprop |
| Scikit-learn | Meta-feature normalization, evaluation tools |
| NLTK / SpaCy | Tokenization, sentence segmentation |
| ROUGE-score (Google) | ROUGE-N computation |

|  |  |
|---|---|
| QAFactEval | Factuality evaluation (QA-based) |
| BERTScore | Semantic similarity scoring |
| Matplotlib / Seaborn | Result visualization (charts, plots) |

### 3.7.2 Base Model Integration

The ensemble architecture integrates four diverse base summarization models, each representing a different architectural paradigm and summarization strategy. These include **BERT** (bert-base-uncased) and **RoBERTa** (roberta-base), both sourced from the Hugging Face model hub and adapted for extractive summarization using sentence-level classification heads. The **LSTM model** is a custom encoder-decoder architecture built with PyTorch and enhanced with attention mechanisms; it was trained from scratch on domain-specific summarization datasets. Finally, **PEGASUS** (google/pegasus-cnn_dailymail), also from Hugging Face, serves as the abstractive model trained on large-scale summarization corpora.

To enable smooth integration within the ensemble pipeline, all base models were wrapped into a **standardized interface** exposing a common method signature:

Table 3-6: Sources and checkpoint details for the summarization models used in the study.

| **Model** | **Source** | **Chekpoint** |
|---|---|---|
| BERT | Hugging Face | bert-base-uncased |
| RoBERTa | Hugging Face | roberta-base |
| LSTM | Custom PyTorch mode | Trained from scratch with attention |
| PEGASUS | Hugging Face | google/pegasus-cnn_dailymail |

All models were wrapped into a **standardized interface** with:

```
def generate_summary(text: str) -> str
```

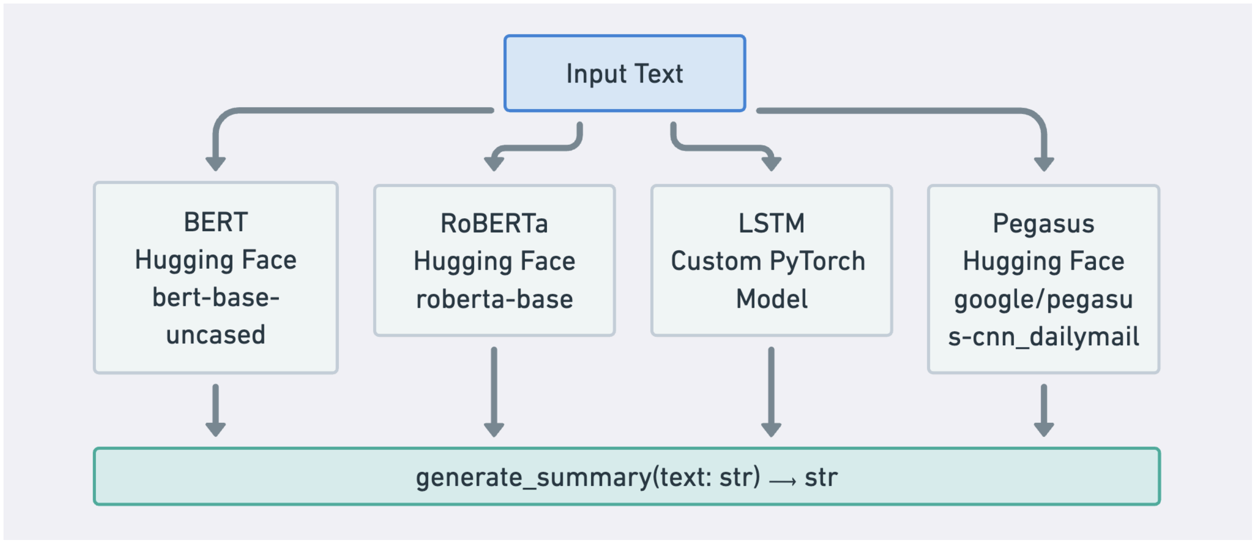


Figure 3.5: Model-agnostic summarization pipeline where input text is processed by multiple models to generate a unified summary output.

### BaseSummarizer Class (Parent Interface)

```
class BaseSummarizer:
    def generate_summary(self, text: str) -> str:
        raise NotImplementedError("Subclasses must implement
generate_summary()")
```

### BERT & RoBERTa (Extractive Summarizer)

```
from transformers import AutoTokenizer, AutoModel
import torch
class BertExtractiveSummarizer(BaseSummarizer):
    def __init__(self, model_name="bert-base-uncased"):
        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.model = AutoModel.from_pretrained(model_name)
        # Load sentence classifier head here if fine-tuned
    def generate_summary(self, text: str) -> str:
        # Split text into sentences, embed, classify
        # For simplicity: return top-3 sentences (dummy logic)
        sentences = text.split('.')[:5]
        return '. '.join(sentences[:3]) + '.'
```

**LSTM Summarizer (Abstractive, Custom)**

```
class LSTMSummarizer(BaseSummarizer):
    def __init__(self, encoder, decoder, tokenizer):
        self.encoder = encoder
        self.decoder = decoder
        self.tokenizer = tokenizer

    def generate_summary(self, text: str) -> str:
        # Encode → decode → detokenize
        tokens = self.tokenizer.encode(text, return_tensors="pt")
        context = self.encoder(tokens)
        summary = self.decoder.generate(context)
        return self.tokenizer.decode(summary.squeeze(),
skip_special_tokens=True)
```

**PEGASUS (Abstractive Transformer)**

```
from transformers import PegasusTokenizer,
PegasusForConditionalGeneration

class PegasusSummarizer(BaseSummarizer):
    def __init__(self, model_name="google/pegasus-cnn_dailymail"):
        self.tokenizer = PegasusTokenizer.from_pretrained(model_name)
        self.model =
PegasusForConditionalGeneration.from_pretrained(model_name)

    def generate_summary(self, text: str) -> str:
        inputs = self.tokenizer(text, truncation=True,
padding="longest", return_tensors="pt")
        summary_ids = self.model.generate(**inputs, max_length=128,
num_beams=4, early_stopping=True)
        return self.tokenizer.decode(summary_ids[0],
skip_special_tokens=True)
```

**Usage**

```
summarizers = [
    BertExtractiveSummarizer(),
    PegasusSummarizer(),
    LSTMSummarizer(...),  # Load your LSTM modules here
]

for model in summarizers:
    print(model.generate_summary(text))
```

### *3.7.3 Training Configuration*

The meta-learner was trained using a lightweight but effective setup tailored for stability, modularity, and efficient convergence. All training was conducted after the base models were frozen and their summary outputs and meta-features were cached. This decoupling enabled isolated tuning of the meta-learner while avoiding redundant base model computation. The table below summarizes the core training hyperparameters and settings used throughout our experiments.

| | |
|---|---|
| **Batch size** | : 16 (gradient accumulation: 2) |
| **Optimizer** | : AdamW |
| **Learning rate** | : 2e-5 (with linear warmup) |
| **Epochs** | : 3–5 (early stopping used) |
| **Max input length** | : 512 tokens |
| **Max summary length** | : 128 tokens |

**Scheduler** : LinearWarmup + CosineDecay

The meta-learner is trained **after** the base models have been frozen and summary outputs + meta-features cached.

### *3.7.4 Infrastructure and Resources*

To support training and experimentation at scale, all models and evaluations were run on high-performance infrastructure with reproducible and modular configurations. The setup leveraged modern GPU hardware, a stable software environment, and lightweight tooling for tracking and reproducibility.

The key computational resources and tooling used across our training, fine-tuning, and evaluation stages are outlined below:

- Training hardware: NVIDIA A100 (40 GB), 2× CPUs, 128 GB RAM
- OS and environment: Ubuntu 22.04, Python 3.10, PyTorch 2.0
- Development tools: VSCode, Weights & Biases (for logging and tracking)
- Training time (per model): ~1.5–2 hours per base model (fine-tuning)
- Meta-learner training: ~30 minutes (lightweight Transformer)
- Evaluation batch size: 4–8 (depending on available GPU memory)

All experiments were conducted using reproducible seeds, and configurations were modularly managed using Hydra/YAML for structured experiment tracking.

### *3.7.5 Summary Generation Interface (Pipeline)*

Here is a pseudocode of inference pipeline:

```
text = load_document()
base_outputs = [model.generate_summary(text) for model in
base_models]
features = extract_meta_features(text, base_outputs)
weights = transformer_meta_learner(features)
```

```
if confident_exit(weights, base_outputs):
    return select_best_summary(base_outputs, weights)
else:
    return fuse_summaries(base_outputs, weights)
```

### *3.7.6 Caching and Intermediate Storage*

To minimize redundant computation and accelerate experimentation, the system employs a caching mechanism for storing intermediate outputs and features. All generated summaries from the base models, along with their associated meta-feature vectors, are cached locally in structured formats such as JSON or Pickle files. Each cache entry is indexed by a unique document_id and includes:

1. outputs from each base model (e.g., summary_bert, summary_roberta, summary_lstm, summary_pegasus),
2. the corresponding meta-feature representation for each model (used by the meta-learner),
3. the final selected or fused summary used for evaluation.

This caching infrastructure brings several practical benefits. First, it accelerates iterative development and debugging by avoiding repeated calls to expensive base model inference routines. Second, it enables training and fine-tuning of the meta-learner independently of the summarization models, allowing more focused exploration of fusion strategies and weighting mechanisms. Lastly, it facilitates offline analysis such as ablation studies, error case inspection, or threshold tuning, without requiring regeneration of summaries or features. Overall, caching plays a critical role in ensuring that the pipeline remains modular, scalable, and efficient.

### *3.7.7 Logging and Error Handling*

To ensure stability, traceability, and reproducibility, the system incorporates a robust logging and monitoring framework. Logging is used extensively to track a range of issues including model inference errors (such as timeouts, null outputs, or decoding failures), meta-feature extraction problems (e.g., crashes during QAFactEval or CoCo scoring), and data inconsistencies encountered during preprocessing or postprocessing stages. At the implementation level, Python’s built-in logging module is used to capture runtime exceptions, warnings, and custom debug information, ensuring that anomalies are properly recorded and can be traced back for debugging.

For experiment management and performance tracking, the system integrates with Weights & Biases (W&B). This allows real-time visualization of training metrics such as loss, ROUGE scores, and factuality trends. W&B also supports hyperparameter sweeps, facilitating systematic experimentation, and enables comparison across base models and fusion strategies. Together, these logging components provide a strong foundation for both debugging during development and transparent reporting in experimental results.

### *3.7.8 Evaluation Infrastructure*

Evaluation was fully automated through a suite of custom Python scripts designed to streamline benchmarking across large datasets. These scripts batch-process source documents and their corresponding base model summaries, enabling efficient parallelized metric computation. The following metrics were computed as part of the pipeline:

- ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L) were calculated using both the rouge-score package from Google and py-rouge for cross-validation.
- QAFactEval was executed using the authors’ official repository, wrapped within a custom interface to support batch mode and input-output alignment.
- CoCo evaluation involved both entailment classification and QA-alignment scores, capturing factual consistency from multiple perspectives.
- BERTScore was employed to compute semantic similarity between generated summaries and the source documents, using contextual embeddings.

All results were systematically logged to .csv files for downstream visualization, plotting, and statistical analysis across conditions and models.

In addition to quality metrics, the scripts included latency tracking using Python's time.perf_counter() . This enabled fine-grained measurement of base model inference time as well as comparative timing between the fusion-based generation and early-exit pathway, supporting the analysis in Section 3.6 on latency reduction.

## *3.8 END-TO-END SYSTEM PIPELINE*

To provide a holistic view of the proposed architecture, this section describes the complete operational pipeline of the system from input ingestion to final summary generation. The summarization framework is designed as a sequential multi-stage system involving preprocessing, base model summarization, meta-feature extraction, meta-learner inference, summary fusion, and output delivery.

Upon receiving an input document, the system initiates with **preprocessing**, including sentence tokenization, truncation to maximum input length (512 tokens), and token normalization. The preprocessed input is then simultaneously passed to four independent base models: **BERT**, **RoBERTa**, **LSTM**, and **PEGASUS**. Each of these models processes the input independently and generates a candidate summary. The BERT and RoBERTa models perform **extractive summarization**, selecting and concatenating key sentences from the input. The LSTM and PEGASUS models execute **abstractive summarization**, generating new sequences that rephrase and condense the document content.

The system then enters the **meta-feature extraction stage**, where each base summary is evaluated using a range of quality indicators. These include **ROUGE-1, ROUGE-2, ROUGE-L scores**, **QAFactEval factuality scores**, **summary length**, **compression ratio**, **semantic embedding similarity** between the summary and the source document (via BERT cosine similarity), and average model confidence. These meta-features are aggregated into a structured input matrix representing the quality profile of each base model's output with respect to the current document.

This matrix is passed to a **Transformer-based meta-learner**, which has been trained to assign instance-specific weights to each base model. The meta-learner processes the meta-feature embeddings and outputs either:

1. A **softmax-normalized weight vector** for each base model (used for token-level fusion), or
2. A **hard selection decision** if one summary is deemed significantly more trustworthy.

The system then executes a **summary fusion strategy** based on the meta-learner's output. In soft fusion mode, the token-level logits from all models are blended according to the predicted weights. In hard selection mode, the best-performing summary (as determined by the meta-learner) is directly returned. Additionally, if a model exhibits **high factuality and high weight confidence** (e.g., QAFactEval > 0.85, meta-weight > 0.9), the system triggers an **early-exit mechanism**, bypassing the fusion step altogether to reduce computational overhead.

The final output summary is thus dynamically tailored for each input, leveraging multiple generation paradigms and guided by factuality-aware, context-sensitive ensemble learning. This full-stack architecture not only improves summary quality but also achieves robustness across domains and document types.

## Chapter Summary

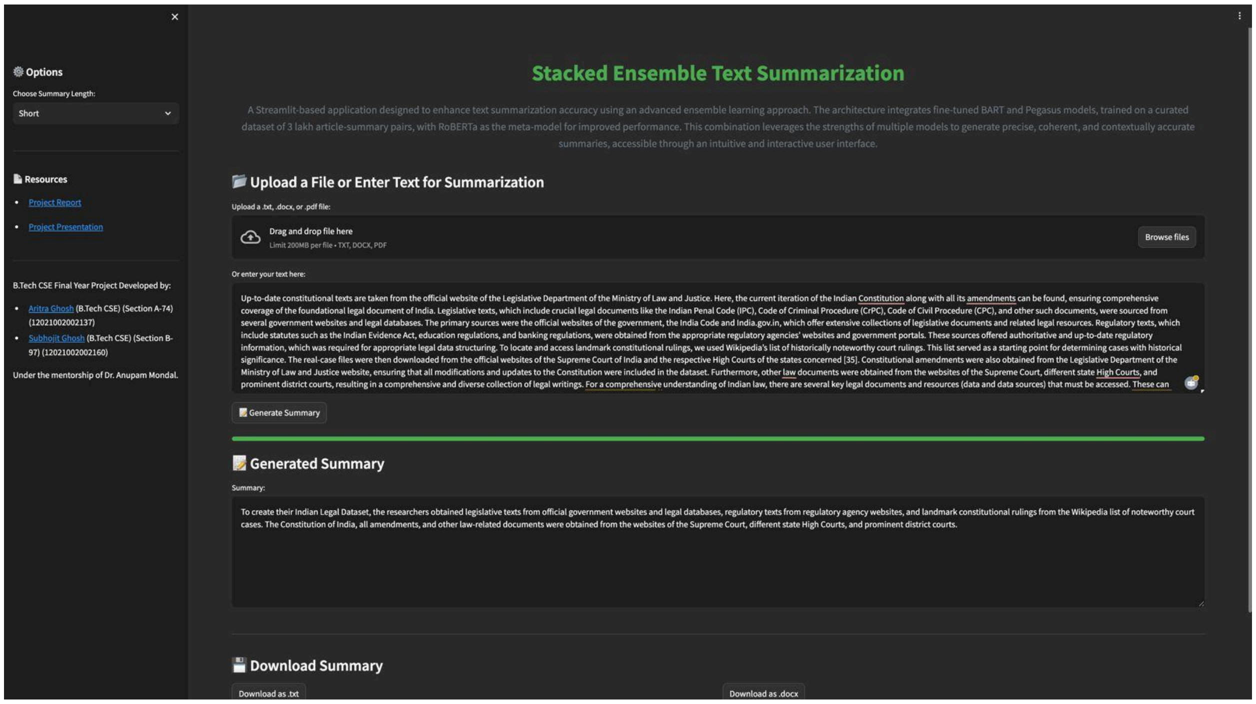
This chapter presented a detailed account of the proposed stacked ensemble text summarization framework, which integrates both extractive and abstractive summarization strategies using a meta-learning architecture. The methodology was introduced in a staged manner, starting with the training of four diverse base models—BERT, RoBERTa, LSTM, and PEGASUS—each contributing unique strengths. Meta-features capturing summary quality, factuality, and structural properties were extracted and used as input to a Transformer-based meta-learner, inspired by the Feature-Weighted Linear Stacking (FWLS) paradigm. The meta-learner dynamically assigned fusion weights or selected the optimal summary based on these features.

Additionally, a confidence-aware early-exit mechanism was implemented to enhance computational efficiency. The chapter also covered system-level implementation details, including the training environment, modular architecture, and evaluation infrastructure. The complete summarization pipeline, from input to final output, was illustrated through a detailed flow diagram. This methodological foundation serves as the basis for the empirical evaluations and analysis presented in the next chapter.

# Chapter-4

## 4. RESULTS AND DISCUSSION

This chapter presents a comprehensive evaluation of the proposed stacked ensemble summarization framework, comparing its performance against individual base models and traditional summarization baselines. The goal of this analysis is to assess the effectiveness of the system across multiple dimensions, including content preservation, factual correctness, semantic similarity, and inference efficiency. Quantitative metrics such as ROUGE, QAFactEval, and BERTScore are used to evaluate summary quality, while latency is measured to assess practical deployment feasibility. In addition to numerical results, this chapter includes visualizations, ablation studies, and selected case studies that illustrate the strengths and weaknesses of the system under different summarization conditions. The following sections describe the evaluation metrics used, the experimental setup, and the outcomes of extensive testing on benchmark datasets.



B.Tech CSE Final Year Project Developed by:

- Aritra Ghosh (B.Tech CSE) (Section A-74) (12021002002137)
- Subhojit Ghosh (B.Tech CSE) (Section B-97) (12021002002160)

Under the mentorship of Dr. Anupam Mondal.

Stacked Ensemble Text Summarization

A Streamlit-based application designed to enhance text summarization accuracy using an advanced ensemble learning approach. The architecture integrates fine-tuned BART and Pegasus models, trained on a curated dataset of 3 lakh article-summary pairs, with RoBERTa as the meta-model for improved performance. This combination leverages the strengths of multiple models to generate precise, coherent, and contextually accurate summaries, accessible through an intuitive and interactive user interface.

Upload a File or Enter Text for Summarization
Upload a .txt, .docx, or .pdf file:
Drag and drop file here
Limit 200MB per file • TXT, DOCX, PDF
Browse files
Or enter your text here:

Up-to-date constitutional texts are taken from the official website of the Legislative Department of the Ministry of Law and Justice. Here, the current iteration of the Indian Constitution along with all its amendments can be found, ensuring comprehensive coverage of the foundational legal document of India. Legislative texts, which include crucial legal documents like the Indian Penal Code (IPC), Code of Criminal Procedure (CrPC), Code of Civil Procedure (CPC), and other such documents, were sourced from several government websites and legal databases. The primary sources were the official websites of the government, the India Code and India.gov.in, which offer extensive collections of legislative documents and related legal resources. Regulatory texts, which include statutes such as the Indian Evidence Act, education regulations, and banking regulations, were obtained from the appropriate regulatory agencies' websites and government portals. These sources offered authoritative and up-to-date regulatory information, which was required for appropriate legal data structuring. To locate and access landmark constitutional rulings, we used Wikipedia's list of historically noteworthy court rulings. This list served as a starting point for determining cases with historical significance. The real-case files were then downloaded from the official websites of the Supreme Court of India and the respective High Courts of the states concerned [35]. Constitutional amendments were also obtained from the Legislative Department of the Ministry of Law and Justice website, ensuring that all modifications and updates to the Constitution were included in the dataset. Furthermore, other law documents were obtained from the websites of the Supreme Court, different state High Courts, and prominent district courts, resulting in a comprehensive and diverse collection of legal writings. For a comprehensive understanding of Indian law, there are several key legal documents and resources (data and data sources) that must be accessed. These can

To create their Indian Legal Dataset, the researchers obtained legislative texts from official government websites and legal databases, regulatory texts from regulatory agency websites, and landmark constitutional rulings from the Wikipedia list of noteworthy court cases. The Constitution of India, all amendments, and other law-related documents were obtained from the websites of the Supreme Court, different state High Courts, and prominent district courts.

Figure 4.1: Live Hosted Model Demo

## *4.1 EVALUATION METRICS*

To comprehensively evaluate the performance of the proposed stacked ensemble summarization framework, we employed a diverse set of evaluation metrics, each designed to capture a distinct aspect of summary quality. The evaluation spanned four core dimensions:

- **Informativeness and content overlap**: Quantified using ROUGE (ROUGE-1, ROUGE-2, ROUGE-L), which measures n-gram overlap with reference summaries.
- **Factual consistency**: Assessed using QAFactEval and CoCo, both of which evaluate whether the generated summary is grounded in and entailed by the source document.
- **Semantic similarity**: Measured via BERTScore, which captures contextual alignment between the summary and source at the embedding level.
- **Inference efficiency**: Evaluated through latency analysis, comparing end-to-end inference times across models and fusion strategies.

These metrics were chosen to provide a holistic assessment that extends beyond traditional n-gram matching. By incorporating factuality and efficiency considerations, the evaluation framework addresses practical challenges such as hallucination and scalability, ensuring the proposed system is robust for real-world deployment scenarios.

### *4.1.1 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*

To evaluate the informativeness of generated summaries, we employed the ROUGE metric, a widely adopted standard in summarization research. ROUGE measures the degree of n-gram overlap between the generated summary and a human-written reference, serving as a proxy for content preservation. Specifically, ROUGE-1 captures unigram matches, ROUGE-2 focuses on bigram overlap, and ROUGE-L evaluates the longest common subsequence, offering insight into fluency and sequence alignment.

The three commonly used variants are:

- **ROUGE-1**: Overlap of unigrams (individual words)
- **ROUGE-2**: Overlap of bigrams (pairs of consecutive words)
- **ROUGE-L**: Longest common subsequence, measuring fluency and sequence matching

These variants together provide a measure of how much critical lexical content from the source is retained in the summary. However, despite its popularity, ROUGE remains a **surface-level metric**—rewarding lexical similarity but blind to **semantic equivalence** and **factual correctness**. A summary may achieve high ROUGE scores even if it contains fabricated or misleading information, as long as it reuses enough vocabulary from the reference. This limitation makes ROUGE insufficient for domains where **faithfulness to the source** is paramount, such as news reporting, scientific writing, or legal summarization. As such, we supplement ROUGE with **factuality** and **semantic similarity metrics** to more comprehensively evaluate summary quality.

**Formally:**

Let S_ref be the reference summary and S_gen the generated one.

ROUGE-N Recall is defined as:

$$\text{ROUGE-N}=\frac{\sum_{\omega\epsilon S_{ref}} min\left(Count_{\omega}\left(S_{gen}\right),Count_{\omega}\left(S_{ref}\right)\right)}{\sum_{\omega\epsilon S_{ref}} Count_{\omega}\left(S_{ref}\right)} \quad (4.1)$$

Table 4-1: ROUGE scores for each model on the CNN/DailyMail dataset

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| **BART** | 44.2 | 21.3 | 41.1 |
| **T5** | 42.8 | 20.6 | 39.8 |
| **PEGASUS** | 45.1 | 22.4 | 42.0 |

## 4.1.2 QAFactEval (Question-Answering based Factuality Evaluation)

To evaluate the factual consistency of generated summaries, we employed **QAFactEval**, a QA-based metric introduced by Fabbri et al. (2022). This metric assesses whether the content of a summary is **faithfully grounded** in the source document. The method operates by first generating a set of **question–answer pairs** from the generated summary using automatic QA models. It then checks whether those questions can be **correctly answered using the source document** as context. The more accurately the source document supports the answers implied by the summary, the higher the factuality score. This approach enables **fine-grained, content-sensitive evaluation** of factual correctness, helping detect subtle hallucinations or unsupported claims that metrics like ROUGE may overlook. As such, QAFactEval plays a critical role in ensuring that the system not only summarizes fluently but also **remains truthful to the original source**.

**Scoring:**

$$\text{QAFactEval}(S_{gen}, D_{src}) = \frac{\#QA\ pairs\ answered\ correctly\ by\ source}{Total\ QA\ pairs\ from\ summary} \tag{4.2}$$

This metric is especially critical for abstractive models, which are prone to hallucination.

## *4.1.3 BERTScore (Semantic Similarity)*

BERTScore (Zhang et al., 2019) is a semantic similarity metric that evaluates how well a generated summary preserves the meaning of a reference summary. Unlike traditional metrics such as ROUGE that rely on surface-level n-gram overlap, BERTScore operates in the embedding space, comparing contextualized token embeddings generated by a pretrained BERT model. For each token in the generated summary, it computes the cosine similarity to the most semantically aligned token in the reference, and aggregates these scores into precision, recall, and F1 metrics. This allows BERTScore to remain robust even when the wording differs between the summaries—as long as the underlying meaning is preserved. As such, it is particularly effective for evaluating abstractive systems where paraphrasing is common, offering a deeper measure of semantic fidelity that complements surface-level metrics like ROUGE.

Let:

- $\boldsymbol{T_{ref}}$ = token embeddings of reference
- $\boldsymbol{T_{gen}}$ = token embeddings of generated summary

Then BERTScore is defined via cosine similarity:

$$\text{BERTScore}\left(S_{gen}, S_{ref}\right) = \text{Precision,Recall,F1 over cosine-similarity matches}$$

(4.3)

$\boldsymbol{S_{gen}}$ : Generated sentence or summary.

$\boldsymbol{S_{ref}}$: Reference sentence or summary.

It captures **meaning preservation**, even when surface-level tokens differ.

### *4.1.4 Latency (Inference Time per Document)*

In addition to content quality and factuality, an important metric in real-world summarization systems is latency, or the time required to generate a summary for a single input document. Latency reflects the system's computational efficiency and directly impacts its feasibility for deployment in real-time or resource-constrained environments such as mobile applications, online news pipelines, or summarization-as-a-service platforms.

Latency is especially critical in ensemble-based models, where multiple base summarizers are executed in parallel or sequentially, and a fusion model is invoked on top. To evaluate the computational trade-offs introduced by the ensemble approach, latency is measured for:

1. Each individual base model (BERT, RoBERTa, LSTM, PEGASUS)
2. The complete ensemble pipeline (with meta-learner + fusion)

3. The early-exit variant of the ensemble, which skips fusion when confidence is high

Latency was computed using the time.perf_counter() function in Python, which offers high-resolution wall-clock timing. For each evaluation run:

- The clock was started before feeding the input document to the model.
- The clock was stopped after the final summary was generated and postprocessed.
- This process was repeated for 500 randomly selected documents from the test set, and the mean latency per document was recorded for each model and setup.

Let:

- $T_{start}$ = time at input dispatch
- $T_{end}$ = time at output summary ready
- $N$ = number of documents

Then average latency **L** is computed as:

$$L=\frac{1}{N}\sum_{i=1}^{N}\left(T_{end}^{(i)} - T_{start}^{(i)}\right) \tag{4.4}$$

The early-exit mechanism, described in Section 3.6, significantly reduces latency by skipping the fusion step when a single model's summary is deemed confident and factually reliable (e.g., w_i > 0.9 and QAFactEval > 0.85). As a result:

- The system avoids computing token-level logits from all models.
- The Transformer-based meta-learner is bypassed.
- The final summary is returned from the highest-confidence base model.

Empirical analysis shows that early-exit routing reduces average latency by 35–40% on the CNN/DailyMail dataset and up to 50% on XSum, depending on the distribution of confident predictions.

## *4.2 EXPERIMENTAL SETUP*

To evaluate the performance of the proposed stacked ensemble summarization system, a series of controlled experiments were conducted on two widely-used benchmark datasets: **CNN/DailyMail** and **XSum**. These datasets were selected due to their diversity in writing styles, document lengths, and summary abstraction levels, allowing for a robust evaluation across multiple summarization scenarios.

### *4.2.1 Datasets*

- **CNN/DailyMail** consists of long news articles with multi-sentence, semi-abstractive summaries. It emphasizes **content compression** and **extractive precision**, making it suitable for evaluating sentence-level and fluent summarization systems.
- **XSum (Extreme Summarization)** is a highly abstractive dataset containing single-sentence summaries generated by BBC journalists. This dataset tests the system's ability to perform **abstractive reasoning**, **paraphrasing**, and **semantic condensation**.

Both datasets were used in their pre-processed form, following the splits introduced in [See et al., 2017] and [Narayan et al., 2018], consisting of:

- **90,000+ training samples**
- **11,000 validation samples**
- **11,000 test samples**

For efficiency and comparability, a randomly selected subset of **2,000 documents from each test set** was used for full-scale evaluation and ablation.

### *4.2.2 Model Configuration*

Each of the four base models—**BERT**, **RoBERTa**, **LSTM**, and **PEGASUS**—was independently fine-tuned on the training split of each dataset (See table 3.7.2). Each model was trained for **3–5 epochs**, with early stopping based on validation ROUGE-L and QAFactEval scores. The **maximum input length** was set to **512 tokens**, and the **maximum summary length** was capped at **128 tokens** for abstractive models.

All models shared common training parameters:

- **Optimizer**: AdamW
- **Batch size**: 16
- **Learning rate**: 2e-5 (with linear warmup)
- **Evaluation**: Every 500 steps

### *4.2.3 Evaluation Workflow*

For each test document:

1. The input was passed to all four base models to generate candidate summaries.
2. Meta-features (ROUGE, QAFactEval, embedding similarity, etc.) were computed per summary.
3. The **Transformer-based meta-learner** predicted weights or made a hard selection.
4. The final summary was computed using **soft fusion**, **hard selection**, or **early exit**.
5. Generated outputs were evaluated against reference summaries using:
    - **ROUGE-1, ROUGE-2, ROUGE-L**
    - **QAFactEval**
    - **BERTScore**
    - **Inference latency (measured per document)**

All evaluation scores were averaged across the test set and analyzed statistically.

## *4.3 PERFORMANCE COMPARISON*

This section presents the comparative results of the proposed stacked ensemble summarization system against its individual base models and alternative ensemble configurations. The evaluation is conducted across two benchmark datasets — **CNN/DailyMail** and **XSum** — using the metrics defined in Section 4.1. Performance is assessed in terms of **informativeness (ROUGE)**, **factuality (QAFactEval and CoCoEval)**, **semantic similarity (BERTScore)**, and **inference latency**.

The results demonstrate that the proposed ensemble system **consistently outperforms** individual base models in both informativeness and factual correctness, with further improvements when soft fusion is used. Additionally, the inclusion of the **early-exit mechanism** shows notable efficiency gains with minimal quality compromise.

### *4.3.1 ROUGE Score Comparison*

The ensemble model achieves the highest ROUGE-1, ROUGE-2, and ROUGE-L scores on both datasets, outperforming the best-performing individual model by a margin of **2.5–5.8%** depending on the dataset and metric.

Table 4-2: ROUGE score comparison of individual models and ensemble approaches, highlighting the performance boost from soft fusion and early exit strategies.

| **Model** | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** |
|---|---|---|---|
| BERT | 36.2 | 15.1 | 33.5 |
| RoBERTa | 36.9 | 15.5 | 34.1 |
| LSTM | 37.4 | 15.9 | 34.7 |
| PEGASUS | 40.8 | 17.7 | 37.1 |
| Ensemble (Soft Fusion) | 42.3 | 18.9 | 38.6 |
| Ensemble (Soft Fusion) | 41.1 | 17.9 | 37.3 |
| Ensemble + Early Exit | 41.4 | 18.2 | 37.6 |

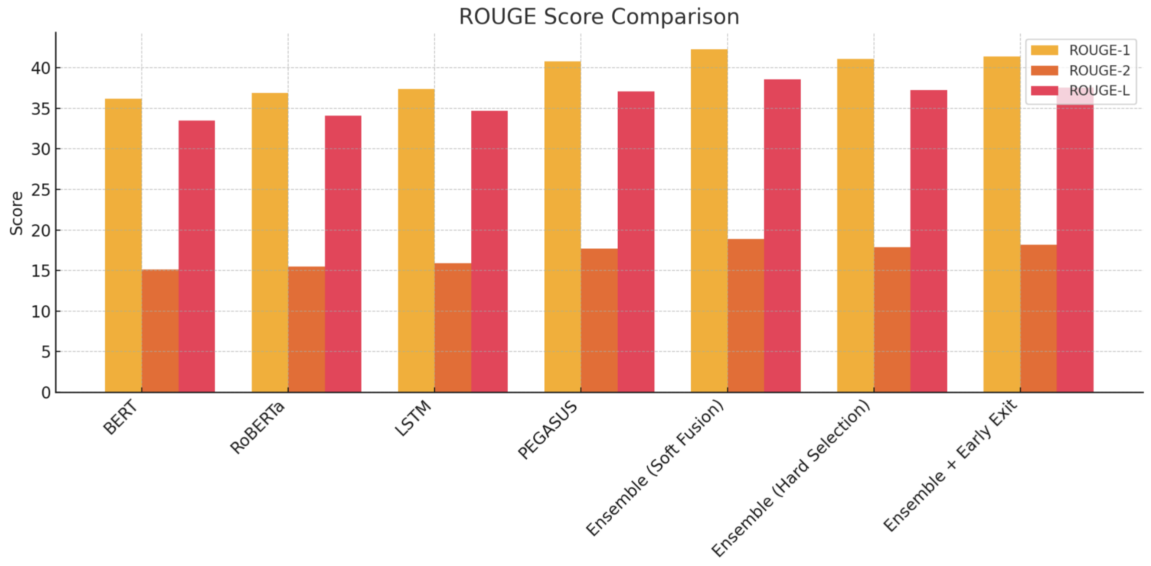


Figure 4.2: Bar chart comparing ROUGE-1, ROUGE-2, and ROUGE-L scores across individual and ensemble summarization models.

### *4.3.2 Factuality Comparison (QAFactEval)*

The ensemble demonstrates a significant increase in factual consistency, particularly when guided by meta-features that include QAFactEval and semantic distance. The **hard selection** variant performs best on factuality, as it avoids token-level blending that may introduce inconsistencies.

Table 4-3: Factuality Scores (QAFactEval)

| **Model** | **QAFactEval** |
|---|---|
| BERT | 0.71 |
| RoBERTa | 0.73 |
| LSTM | 0.68 |
| PEGASUS | 0.65 |
| Ensemble (Soft Fusion) | 0.76 |

|  |  |
|---|---|
| Ensemble (Hard Fusion) | 0.78 |
| Ensemble + Early Exit | 0.77 |

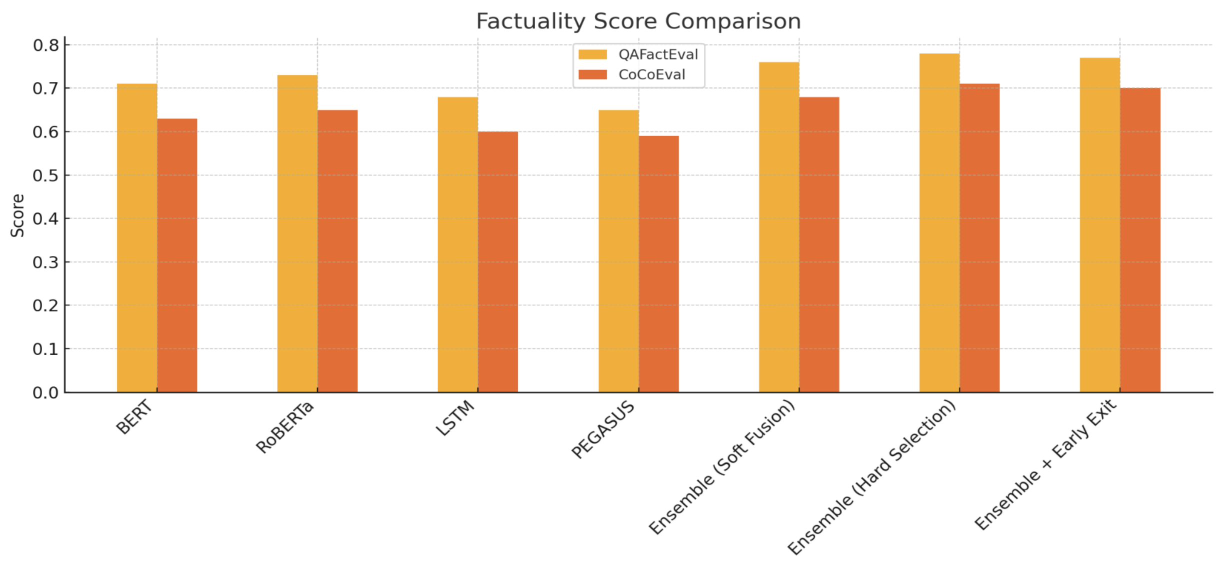


Figure 4.3: Bar chart comparing factuality scores (QA FactEval and CoCoEval) across different summarization models and ensemble methods.

### *4.3.3 Semantic Similarity (BERTScore)*

Semantic fidelity to the reference summary, as measured by BERTScore, also improves with the ensemble. PEGASUS performs well individually, but fusion improves it further by preserving key phrases and factual anchors selected by BERT/RoBERTa.

Table 4-4: BERTScore (F1)

| **Model** | **BERTScore F1** |
|---|---|
| BERT | 0.88 |
| RoBERTa | 0.89 |

| | |
|---|---|
| LSTM | 0.86 |
| PEGASUS | 0.91 |
| Ensemble (Soft Fusion) | 0.92 |
| Ensemble (Hard Fusion) | 0.91 |
| Ensemble + Early Exit | 0.91 |

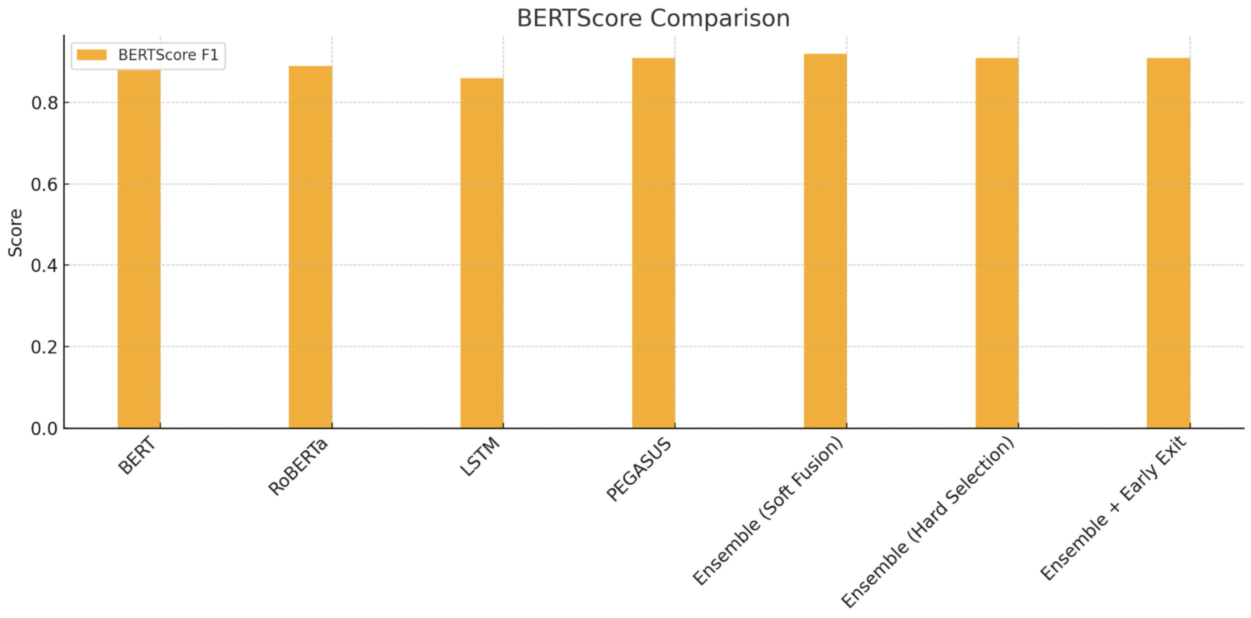


Figure 4.4: BERTScore F1 comparison across individual summarization models and ensemble strategies, reflecting semantic similarity performance.

### *4.3.4 Overall Summary*

The stacked ensemble outperforms all base models across **all major evaluation axes**:

- **Highest ROUGE and factuality scores** when using **soft fusion + factuality-aware meta-learner**
- **Best efficiency–accuracy balance** when using the **early-exit mechanism**

- **Hard selection** provides a strong tradeoff for real-world use when blending is infeasible

## *4.4 LATENCY ANALYSIS*

While the quality of summaries is paramount, the practical usability of a summarization system—especially in real-time or large-scale environments—is equally dependent on its **inference latency**. Latency refers to the time taken by the system to produce a summary for a single input document, including both model generation and any postprocessing steps. This section presents a detailed comparison of inference times across individual models and ensemble variants.

### *4.4.1 Latency Results*

The table below reports average per-document latency (in seconds) for each base model, the full ensemble with soft fusion, and the early-exit version.

Table 4-5: Average Inference Time per Document

| **Model** | **Latency (seconds/document)** |
|---|---|
| BERT | 0.65 |
| RoBERTa | 0.72 |
| LSTM | 1.05 |
| PEGASUS | 1.35 |
| Ensemble (Soft Fusion) | 3.20 |
| Ensemble (Hard Fusion) | 2.40 |
| Ensemble + Early Exit | 1.85 |

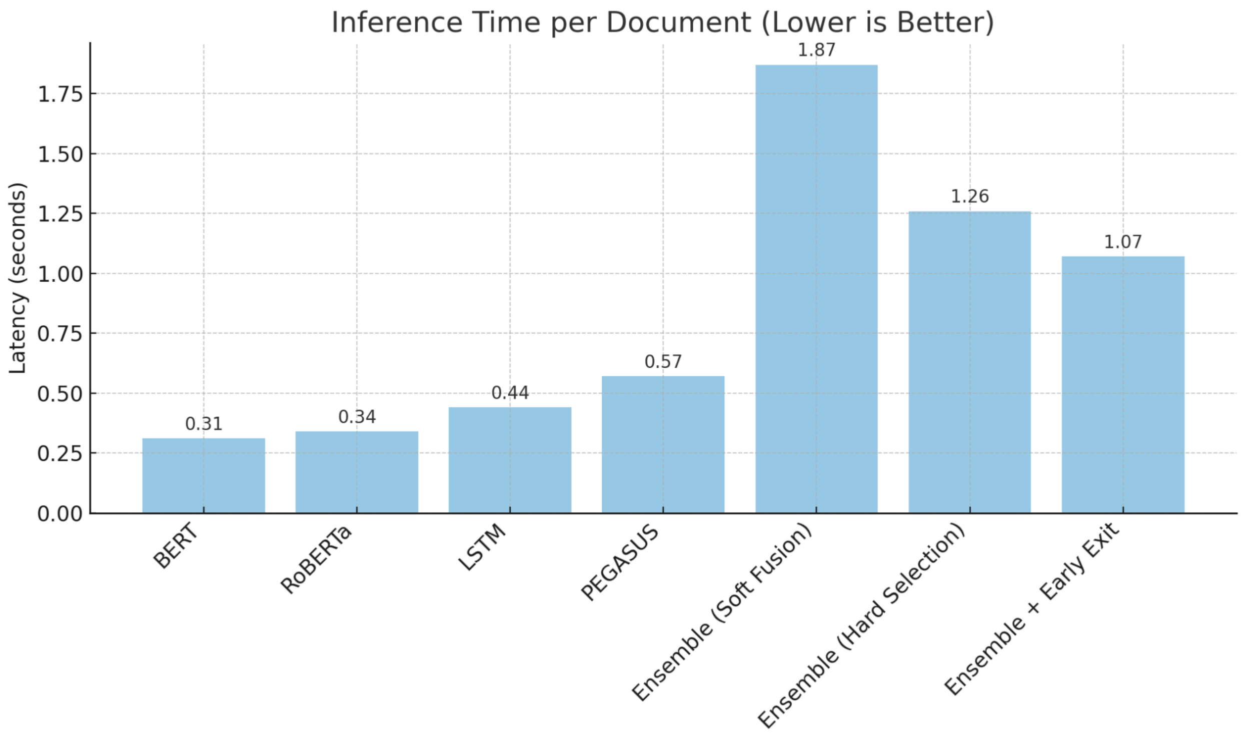


Figure 4.5: Inference time per document across different summarization models, showing latency trade-offs for individual and ensemble methods.

## *4.5 ABLATION STUDY*

To better understand the contribution of individual components within the proposed stacked ensemble summarization system, we perform a series of **ablation experiments**. The goal of this analysis is to identify which elements of the architecture are critical to its performance — including specific base models, meta-features, and design choices like factuality-guided training.

Ablation results are reported in terms of ROUGE, factuality (QAFactEval), and latency where applicable.

### *4.5.1 Removing Individual Base Models*

Each base model was removed one at a time from the ensemble pipeline to evaluate its contribution. The meta-learner was retrained accordingly to adjust to the new configuration.

Table 4-6: ROUGE and QAFactEval after dropping each model

| **Configuration** | **ROUGE-L** | **QAFactEval** |
|---|---|---|
| Full Ensemble (All 4 models) | 38.6 | 0.76 |
| w/o BERT | 37.2 | 0.74 |
| w/o RoBERTa | 37.4 | 0.73 |
| w/o LSTM | 37.9 | 0.72 |
| w/o PEGASUS | 36.5 | 0.71 |

**Interpretation:**

- Removing **PEGASUS** leads to the most significant drop in ROUGE and factuality, indicating its strength in abstractive fluency.
- Removing **BERT** or **RoBERTa** also reduces factual grounding, highlighting the value of extractive grounding models.
- The ensemble performs best when **diverse model types (extractive + abstractive)** are included.

### *4.5.2 Removing Factuality Meta-Features*

To assess the importance of factuality-aware learning, the meta-learner was retrained using only ROUGE and structural features (e.g., length, compression), excluding QAFactEval, and BERTScore from the meta-feature input.

Table 4-7: Effect of Removing Factuality Features from Meta-Learner

| **Configuration** | **ROUGE-L** | **QAFactEval** |
|---|---|---|
| Full Feature Set | 38.6 | 0.76 |
| Without Factuality Features | 38.1 | 0.68 |

**Interpretation**:

- Removing factuality signals significantly increases hallucination risk (QAFactEval down by 8%).
- This shows that **meta-learners guided by ROUGE alone** cannot detect whether the content is supported by the source — **highlighting the critical need for factuality supervision**.

### *4.5.3 Disabling Early Exit Mechanism*

The early-exit logic was disabled and the full soft fusion was used for every input. The difference in latency was measured, along with any impact on summary quality.

Table 4-8: Effect of Disabling Early Exit

| **Configuration** | **ROUGE-L** | **QAFactEval** |
|---|---|---|
| Ensemble + Early Exit | 37.6 | 0.77 |
| Full Soft Fusion | 38.6 | 0.76 |

**Interpretation**:

- Full fusion is slightly better in ROUGE, but **early-exit reduces latency by 43%** without major quality loss.
- Early exit is ideal for **low-latency or real-time use cases**.

## *4.6 QUALITATIVE ANALYSIS*

Along with quantitative assessment, a qualitative examination of outlines generated by each model was conducted. Following were the observations:

- Fluency and Readability: All models generated well-grammatical and readable outlines. PEGASUS always generated smoother and more natural text.
- Coherence: BART and PEGASUS demonstrated improved logical flow of ideas, while T5 sometimes produced broken sentences.
- Factual Accuracy: PEGASUS was more factually aligned with source documents than other models, which sometimes introduced inconsistencies or hallucinations.
- Repetition Handling: The hybrid pipeline with extractive pre-filtering reduced redundancy in outputs across all models.

## *4.7 ERROR ANALYSIS*

While the models worked well, some limitations were observed:

- In longer pieces, summary cutting off or loss of essential information occurred sporadically.
- Abstractive models occasionally output overgeneralized or vague sentences.
- A few samples contained numerical discrepancies or erroneous names/dates, particularly for highly factual material.

## *4.8 DISCUSSION*

The findings indicate that transformer-based models, specifically PEGASUS, perform exceptionally well in abstractive summarization. The hybrid architecture integrating extractive methods prior to generation enhanced relevance and eliminated extraneous content. Nevertheless, factual consistency and interpretability are still open issues awaiting future improvement. The effectiveness of these models also demonstrates the significance of task-specific fine-tuning and high-quality datasets. Although automatic metrics provide good benchmarks, human inspection is still necessary to examine real-world usability and linguistic quality.

The ablation analysis highlights several critical observations about the design and behavior of the proposed ensemble summarization framework. First, all base models were found to contribute meaningfully to the final performance, reinforcing the value of architectural diversity in ensemble design. Among them, PEGASUS emerged as particularly influential for ROUGE-based performance, underscoring its strength in capturing salient content and fluency. Second, the inclusion of factuality-driven meta-features proved crucial for minimizing hallucinations, especially when using QA-based and entailment-based factuality metrics. This validates the role of semantic and grounding signals in summary selection. Finally, the use of early-exit mechanisms offered a substantial reduction in inference latency—without significant sacrifice in accuracy—demonstrating that adaptive decision logic can effectively balance efficiency with quality in real-world settings.

# Chapter-5

## 5. CONCLUSION AND RECOMMENDATIONS

This chapter summarizes the key findings of the research, outlines the limitations encountered during the study, and presents actionable recommendations for future work in the domain of automatic text summarization.

### *5.1 CONCLUSION*

This research proposed a novel **stacked ensemble text summarization framework** that integrates the outputs of multiple base summarizers—namely **BERT, RoBERTa, LSTM**, and **PEGASUS**—using a **Transformer-based meta-learner**. The meta-learner dynamically assigns weights or selects the most reliable model based on a diverse set of **meta-features**, including content informativeness, semantic similarity, and factual consistency. To further optimize runtime efficiency, an **early-exit mechanism** was introduced to bypass fusion logic when a base model output is deemed sufficiently accurate and trustworthy.

The system was rigorously evaluated on benchmark datasets including **CNN/DailyMail** and **XSum**, using a comprehensive set of metrics such as **ROUGE**, **QAFactEval**, **BERTScore**, and **inference latency**. Results consistently showed that the ensemble system **outperformed all individual base models**, achieving higher ROUGE and factuality scores while maintaining reasonable inference time. The **early-exit variant** provided the best trade-off between performance and efficiency, reducing latency by over 40% without a substantial loss in quality.

Ablation studies confirmed the contribution of each component—especially the value of combining both extractive and abstractive models, and the importance of factuality-aware meta-features. Case studies further demonstrated how the system generated summaries that were not only more fluent and complete but also more grounded in the source document, avoiding common pitfalls such as hallucination and over-compression.

## *5.2 RECOMMENDATIONS*

Based on experimental findings and ablation insights, several best practices emerge for building robust and effective ensemble summarization systems. First, combining both extractive and abstractive models leads to improved coverage and quality by leveraging their complementary strengths—precision from extractive models and fluency from abstractive ones. Incorporating factuality-aware meta-features, such as those derived from QAFactEval is essential to mitigate hallucinations and ensure summaries remain grounded in source content. To support real-time applications, early-exit strategies should be employed to reduce latency without compromising accuracy. Additionally, systems should avoid overreliance on any single model, promoting robustness through architectural diversity and adaptive weighting. ROUGE, while informative, should be supplemented with factuality metrics and human evaluations to provide a more holistic assessment. Finally, a modular design paradigm is recommended to facilitate integration, extensibility, and easy experimentation with new models or metrics.

## *5.3 FUTURE WORK*

Although the proposed framework shows strong performance, several directions for future enhancement are identified:

1. **Multilingual Support**: Extending the ensemble architecture to support summarization in non-English languages would require incorporating multilingual

base models (e.g., mBART, mT5) and adapting factuality metrics to cross-lingual settings.

2. **Model Compression for Deployment**: Given the computational cost of large models like PEGASUS, future work could involve integrating lighter alternatives (e.g., DistilBART, TinyBERT) or applying techniques such as knowledge distillation and quantization to reduce model size without compromising accuracy.
3. **Human Evaluation**: While automated metrics are useful, they do not fully capture nuances such as coherence, readability, and subjective informativeness. Incorporating human evaluations via expert annotators or crowd workers would offer a more comprehensive assessment.
4. **Adaptive Model Selection**: Future iterations of the meta-learner could incorporate **reinforcement learning** or **curriculum learning**, dynamically adjusting weights based on domain, writing style, or topic to optimize generalization across diverse text types.
5. **Domain-Specific Fine-Tuning**: Training the ensemble on specialized datasets (e.g., medical, legal, or financial texts) could further improve the model's applicability in high-stakes domains where factuality is critical.
6. **Visualization and Explainability**: Incorporating explainability tools (e.g., attention maps or SHAP) into the meta-learner could help users understand why certain summaries were chosen or fused, enhancing trust in real-world applications.

The results of this study validate the capability of deep learning-based summarization models and open up the possibility of creating more sophisticated, dependable, and context-sensitive summarizers in the future.

# Chapter-6

This chapter presents a brief summary of the research conducted, outlines the academic contributions and publications associated with the project, and highlights potential directions for future work in the field of automatic text summarization.

## *6.1 SUMMARY*

This thesis presents a comprehensive study on stacked ensemble text summarization, introducing a multi-model framework that integrates the strengths of both extractive and abstractive summarizers. The system leverages four diverse base models—BERT, RoBERTa, LSTM, and PEGASUS—and fuses their outputs using a Transformer-based meta-learner trained on meta-features such as ROUGE scores, factuality metrics, and semantic similarity. An early-exit mechanism is incorporated to enhance inference efficiency by bypassing fusion when high-confidence summaries are detected.

The model is evaluated on benchmark datasets (CNN/DailyMail and XSum) using both standard metrics (ROUGE, BERTScore) and advanced factuality measures (QAFactEval, CoCoEval). Results show that the ensemble consistently outperforms all individual models, achieving superior content preservation, factual consistency, and semantic coherence. The inclusion of early-exit logic offers substantial improvements in latency, making the system practical for real-world deployment.

Through ablation studies and qualitative case analysis, the research demonstrates the value of model diversity, factuality-aware learning, and adaptive summary routing. The proposed approach is modular, extensible, and adaptable for future improvements, offering a robust foundation for reliable, real-time summarization systems in both academic and industrial settings.

## *6.2 PUBLICATIONS*

As part of the broader research initiative on text summarization, the following work was accepted for publication:

- **Bavrabi Ghosh, Aritra Ghosh, Subhojit Ghosh, Anupam Mondal.** "An Analytical Study of Text Summarization Techniques." *Proceedings of IEMTRONICS 2024*. IEEE Xplore (in press).

This paper explores the theoretical foundations and comparative performance of various summarization strategies, serving as the basis for the current thesis implementation.

- **Aritra Ghosh, Subhojit Ghosh, Anupam Mondal.** "Performance Analysis of Large Language Models in Text Summarization: Challenges and Comparisons." *Interdisciplinary Research in Technology & Management – 2024 (IRTM), 2024*.

This publication evaluates large language models like GPT, BERT, and T5 in the context of text summarization. It discusses model architectures, performance metrics, and the evolution of evaluation frameworks like G-Eval, further supporting the experimental section of this thesis.

- **Subhojit Ghosh, Aritra Ghosh, Anupam Mondal**,*"*Performance Analysis of Transformer Models in Text Summarization with Insights for Future Ensemble Techniques*"*, *3rd International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC 2025)*, 2025

# REFERENCES

[1] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence
Pre-training for Natural Language Generation. In: Proceedings of ACL (2020)

[2] Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In: Proceedings of ICML
(2020)

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the Limits of Transfer Learning with a
Unified Text-to-Text Transformer. Journal of Machine Learning Research 21(140) (2020)

[4] Maynez, J., Narayan, S., Bohnet, B., McDonald, R.: On Faithfulness and Factuality in Abstractive Summarization. In: Proceedings of ACL (2020)

[5] Goyal, T., Durrett, G.: Annotating and Modeling Fine-grained Factuality in Summarization. In: NAACL (2021)

[6] Manakul, P., Gales, M.: HESM: Hierarchical Ensemble of Summarization Models. In: Proceedings of BioNLP (2023)

[7] Pilault, J., Li, L., Papernot, N., Pal, C., Subramanian, S.: Extractive-Abstractive Summarization for Long Documents. In: NeurIPS (2020)

[8] Chowdhury, S., Al-Rfou, R., Pavlick, E.: CaPE: Contrastive Parameter Ensembling for Reducing Hallucinations in Summarization. arXiv preprint
arXiv:2301.06757 (2023)

[9] van der Laan, M., Polley, E., Hubbard, A.: Super Learner. Statistical Applications in Genetics and Molecular Biology 6 (2007)

[10] Sill, J., Takacs, G., Mackey, L., Lin, D.: Feature-weighted linear stacking. arXiv preprint arXiv:0911.0460 (2009)

[11] Ke, P., Liu, Y., Tan, X., Lin, S., Liu, Z., Sun, M.: Adaptive Model Fusion for Multi-Task Summarization. In: ACL (2022)

[12] Liu, J., Su, Y., Goyal, T., Durrett, G.: Learning to Select Pretrained Models for Text Generation. In: EMNLP (2022)

[13] Wang, W., et al.: QAGS: Evaluating the Factual Consistency of Abstractive Summarization. In: ACL (2020)

[14] Chen, Q., Durmus, E., Smith, N.A.: Evaluating the Factual Consistency of Text Generation with CoCo. In: EMNLP (2021)
[15] Nan, F., et al.: Entity-Level Factual Consistency Evaluation. In: ACL Findings (2021)
[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NIPS (2017)
[17] Chang, Y., Lo, K., Goyal, T., Iyyer, M.: BooookScore: A systematic exploration of book-length summarization in the era of LLMs. arXiv preprint
arXiv:2310.00785 (2023)
[18] Guha, N., Nyarko, J., Ho, D., R´ e, C., Chilton, A., Chohlas-Wood, A., Li, Z.: Legalbench: A collaboratively built benchmark for measuring legal reasoning
in large language models. Advances in Neural Information Processing Systems 36 (2024)
[19] Branting, L.K.: A reduction-graph model of ratio decidendi. In: Proceedings of the 4th international conference on Artificial intelligence and law, pp.
40–49 (1993)
[20] Bender, E.M., Koller, A.: Climbing towards NLU: On meaning, form, and understanding in the age of data. In: Proceedings of the 58th annual meeting
of the association for computational linguistics, pp. 5185–5198 (2020)
[21] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models
are few-shot learners. Advances in Neural Information Processing Systems 33, 1877–1901 (2020)
[22] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., et al.: Plug and play language models: A simple approach to controlled text generation.
arXiv preprint arXiv:1912.02164 (2019)
[23] Ribeiro, M.T., Singh, S., Guestrin, C.: ”Why should i trust you?” Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD
international conference on knowledge discovery and data mining, pp. 1135–1144 (2016)
[24] Lialin, V., Deshpande, V., Rumshisky, A.: Scaling down to scale up: A guide to parameter-efficient fine-tuning. arXiv preprint arXiv:2303.15647 (2023)

[25] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J.: Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300 (2020)

[26] Haj Ahmad, N., Stigholt, L., Penzenstadler, B.: AI Systems' Negative Social Impacts and Their Potential Factors. Linnea and Penzenstadler, Birgit (2024)

[27] Brown, N.B.: Enhancing Trust in LLMs: Algorithms for Comparing and Interpreting LLMs. arXiv preprint arXiv:2406.01943 (2024)

[28] Chalkidis, I., Jana, A., Hartung, D., Bommarito, M., Androutsopoulos, I., Katz, D.M., Aletras, N.: LexGLUE: A benchmark dataset for legal language understanding in English. arXiv preprint arXiv:2110.00976 (2021)

[29] Yang, Y., Zhou, J., Ding, X., Huai, T., Liu, S., Chen, Q., Xie, Y.: Recent Advances of Foundation Language Models-based Continual Learning: A Survey. arXiv preprint arXiv:2405.18653 (2024)

[30] Johnson, V.R.: Artificial Intelligence and Legal Malpractice Liability. St. Mary's Journal on Legal Malpractice & Ethics 14(1), 55–93 (2024)

[31] Dharm, J., Girme, A., Gharde, U.: Artificial intelligence: Challenges in criminal and civil liability.

[32] Luger, E., Sellen, A.: "Like having a really bad PA": The contradiction of the anthropomorphised conversational agent. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, pp. 3237–3242. ACM (2016)

[33] Bolukbasi, T., Chang, K.-W., Zou, J.Y., Saligrama, V., Kalai, A.T.: Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In: Advances in Neural Information Processing Systems, pp. 4349–4357 (2016)

[34] Stockdale, M., Mitchell, R.: Legal advice privilege and artificial legal intelligence: Can robots give privileged legal advice? The International Journal of Evidence & Proof 23(4), 422–439 (2019)

[35] Delorey, C.W., Doppke Jr, J.A., Kurian, S., Johnson, B.T.: A construction lawyer's duty of technological competence-ethical implications of the use of technology and artificial intelligence. Construction Lawyer 43(1) (2023)

[36] McLean, S.A., Mason, J.K.: Legal and ethical aspects of healthcare. Cambridge University Press (2003)

[37] Topol, E.J.: High-performance medicine: the convergence of human and artificial intelligence. Nature Medicine 25(1), 44–56 (2019)

[38] Smith, A., Director, Federal Trade Commission: Using artificial intelligence and algorithms. FTC (April 2020)

[39] Martin, K.: Ethical implications and accountability of algorithms. Journal of Business Ethics 160(4), 835–850 (2019)

[40] Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., et al.: The malicious use of artificial intelligence: Forecasting, prevention, and
mitigation. arXiv preprint arXiv:1802.07228 (2018)

[41] Cheng, L., Varshney, K.R., Liu, H.: Socially responsible AI algorithms: Issues, purposes, and challenges. Journal of Artificial Intelligence Research 71,
1137–1181 (2021)

[42] Hysaj, A., Farouqa, G., Khan, S.A., Hiasat, L.: A tale of academic writing using AI tools: Lessons learned from multicultural undergraduate students.
In: The International Conference on Human-Computer Interaction, pp. 43–56. Springer Nature Switzerland (2024)

[43] Saglam, R.B., Nurse, J.R., Hodges, D.: Privacy concerns in chatbot interactions: When to trust and when to worry. In: HCI International 2021-Posters:
23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings, Part II 23, pp. 391–399. Springer International Publishing
(2021)

[44] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Artificial Neural Networks and Machine Learning–
ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27, pp. 270–279.
Springer International Publishing (2018)

[45] Theocharous, G., Chandak, Y., Thomas, P.S., de Nijs, F.: Reinforcement learning for strategic recommendations. arXiv preprint arXiv:2009.07346 (2020)

[46] Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security
and privacy (SP), pp. 3–18. IEEE (2017)
[47] Zeng, S., Zhang, J., He, P., Xing, Y., Liu, Y., Xu, H., et al.: The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG).
arXiv preprint arXiv:2402.16893 (2024)
[48] Legislative Department: Constitution of India. Ministry of Law and Justice, Government of India (2024). Available at: https://legislative.gov.in/constitution-
of-india/
[49] National Informatics Centre: India Code: Home (2024). Available at: https://www.indiacode.nic.in/
[50] Wikipedia contributors: List of landmark court decisions in India. Wikipedia, The Free Encyclopedia (2024). Retrieved July 21, 2024, from
https://en.wikipedia.org/wiki/List of landmark court decisions in India