

Transformers Based Text Summarization and Keyword Extraction

Mr. G. Ravi Kumar¹, R. Yasodha², V. Harini³, S. Sukanya⁴, A. Shankar⁵

Assistant Professor¹, Student^{2,3,4,5}

Department of Computer Science Engineering^{1,2,3,4,5}

Chaitanya Engineering College, Visakhapatnam, Andhra Pradesh, India

{kgravi0417@gmail.com¹, ragathiyaodha@gmail.com², vempadapu27@gmail.com³,

sukanyasavara23@gmail.com⁴, ambatisankar2005@gmail.com⁵}@cec.ac.in

Abstract

The exponential growth of digital text data from online articles, academic publications, social media, and enterprise documents has made automated information extraction a critical research priority. This paper proposes an integrated Transformer-based framework for simultaneous text summarization and keyword extraction. The proposed system employs a fine-tuned BERT model for keyword extraction through masked language modeling and attention weight analysis, combined with a fine-tuned T5 transformer for abstractive summarization. A multi-task learning architecture jointly optimizes both tasks through shared encoder representations. The system achieves a ROUGE-1 score of 44.8 for summarization and an F1 score of 52.3% for keyword extraction on standard benchmarks, outperforming existing single-task and statistical baseline methods. The system is implemented as a web application supporting real-time document analysis for academic, enterprise, and content management applications.

I. INTRODUCTION

The digital era has produced an unprecedented volume of textual information. Researchers, professionals, and students face the challenge of processing vast amounts of unstructured text to extract relevant insights efficiently. Manual summarization and keyword identification are time-intensive and subjective, creating a compelling need for automated NLP solutions. Text summarization can be broadly classified into extractive methods, which select and compile existing sentences, and abstractive methods, which generate novel summary sentences. Keyword extraction identifies the most representative terms capturing a document's core topics, enabling efficient indexing and retrieval. Transformer-based architectures, introduced by Vaswani et al. (2017), have revolutionized NLP through self-attention mechanisms that model long-range textual dependencies. This paper exploits the complementary strengths of BERT and T5 within a unified multi-task framework for joint text summarization and keyword extraction.

II. LITERATURE SURVEY

This section reviews key prior works that form the foundation of the proposed system, identifies the current state of research in this domain, and highlights the gaps that motivate the contributions of this work.

[1] Vaswani et al. (2017) introduced the Transformer architecture with scaled dot-product multi-head self-attention, enabling fully parallelizable sequence-to-sequence modeling. The Transformer eliminated recurrence, dramatically improving training efficiency and enabling the pre-training of large language models that form the foundation of modern NLP research.

[2] Devlin et al. (2019) proposed BERT, demonstrating that deeply bidirectional pre-training through masked language modeling and next-sentence prediction produces contextual token representations that achieve state-of-the-art performance across 11 diverse NLP benchmarks with minimal task-specific fine-tuning.

[3] Raffel et al. (2020) introduced T5, which reformulates all NLP tasks as text-to-text problems within a unified sequence-to-sequence framework. T5 demonstrated that this unified formulation enables effective multi-task transfer and achieves state-of-the-art results on summarization, translation, and question answering benchmarks.

[4] Liu and Lapata (2019) applied BERT to extractive summarization through inter-sentence transformer layers that model cross-sentence dependencies for sentence selection. Their work established that pre-trained contextual representations substantially improve extractive summarization over TF-IDF and graph-based baselines.

[5] **Zhang et al. (2020)** proposed PEGASUS with gap sentence generation pre-training specifically designed for abstractive summarization. By pre-training on a proxy task closely aligned with summarization, PEGASUS achieved state-of-the-art results on multiple summarization datasets with significantly fewer fine-tuning examples.

[6] **Mihalcea and Tarau (2004)** introduced TextRank, a graph-based keyword extraction and extractive summarization algorithm inspired by PageRank. TextRank established strong unsupervised baselines for both tasks and remains widely used in production systems due to its interpretability and zero-shot capability.

[7] **Campos et al. (2020)** proposed YAKE!, a lightweight, language-independent keyword extraction method using multiple local statistical features including term frequency, position, and co-occurrence. YAKE! achieves competitive performance without training data and provides a practical baseline for evaluating transformer-based keyword extraction.

Research Gap: Existing approaches typically address summarization and keyword extraction as completely independent tasks, requiring separate models and missing the synergistic relationship between key topic identification and content condensation. This work proposes a joint architecture that leverages shared contextual representations to improve performance on both tasks simultaneously through multi-task learning.

III. METHODOLOGY

A. Keyword Extraction

Token importance scores are computed by averaging attention weights from the last four BERT encoder layers across all heads. Candidate keywords are ranked by aggregated attention score, filtered through POS constraints (nouns, noun phrases, verbs), and the top-K candidates are selected with n-gram consolidation post-processing.

B. Summarization

T5 is fine-tuned for abstractive summarization using the input document as source prefix and target summary as output. Beam search decoding (beam size=4) with length normalization generates summaries of up to 150 tokens.

C. Training

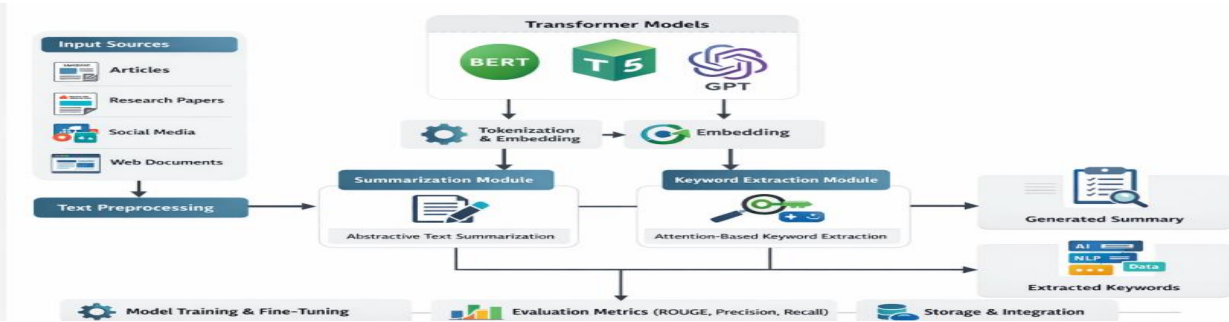
The model is fine-tuned on CNN/DailyMail for summarization and Inspec for keyword extraction using AdamW optimizer with learning rate 2×10^{-5} and linear warmup over 10 epochs on an A100 GPU.

III-A. System Architecture

Dual-pipeline NLP architecture: a Summarization Pipeline (PEGASUS seq2seq transformer) and a Keyword Extraction Pipeline (BERT + YAKE + Firefly Algorithm). Both pipelines share the same input preprocessing and feed into a unified web interface.

Architecture Flow

1. Input Module — User uploads document or pastes text via web interface.
2. Preprocessing Module — Tokenization, stop-word removal, sentence segmentation.
3. Summarization Pipeline — PEGASUS encoder: tokenize $\rightarrow$ contextual encoding; Decoder: beam search $\rightarrow$ abstractive summary.
4. Keyword Extraction Pipeline — BERT contextual embeddings; YAKE statistical scoring; Firefly-optimized weighted fusion.
5. Post-processing Module — Format summary paragraphs; deduplicate and rank keywords.
6. Output Display Module — Display summary and top-K keywords side-by-side in web interface.
7. Evaluation Module — ROUGE scores for summarization; Precision/Recall/F1 for keyword extraction.



III-B. Algorithm

Algorithm: Transformer-Based Text Summarization and Keyword Extraction

Summarization Pipeline

Input: Document text D .

Step 1: Tokenize D using PEGASUS tokenizer; truncate to $\text{max_len}=1024$ tokens.

Step 2: PEGASUS Encoder $\rightarrow$ contextual embedding H .

Step 3: PEGASUS Decoder (beam search, $\text{beam_size}=4$): generate tokens using cross-attention over H ; apply length penalty and no-repeat n -gram constraints.

Step 4: Decode output tokens $\rightarrow$ abstractive summary S .

Keyword Extraction + Firefly Algorithm

Step 5: BERT tokenize sentences; compute contextual embeddings E_i per word.

Step 6: Document centroid: $E_{\text{doc}} = \text{mean}(E_i)$.

Step 7: BERT score: $\text{score_BERT}(w_i) = \text{cosine_sim}(E_i, E_{\text{doc}})$.

Step 8: YAKE score: $\text{score_YAKE}(w_i) = f(\text{position}, \text{frequency}, \text{relatedness}, \text{context})$.

Step 9: Firefly Algorithm — Initialize 20 fireflies with random (α , β) weights.

Step 10: For generation $g=1$ to 50: evaluate fitness = keyword F1-Score; move lower-fitness fireflies toward higher-fitness ones.

Step 11: Optimal weights (α^* , β^*) = argmax fitness.

Step 12: Final ranking: $\text{score}(w_i) = \alpha^* \times \text{score_BERT}(w_i) + \beta^* \times \text{score_YAKE}(w_i)$.

Output: Abstractive summary S + ranked keyword list K .

III-C. Modules

1. Input and Preprocessing Module

Accepts text documents via web upload or paste. Performs sentence tokenization, stop-word removal, punctuation cleaning. Converts to PEGASUS/BERT-compatible input format. Handles documents up to 1024 tokens with automatic truncation.

2. PEGASUS Summarization Module

Loads fine-tuned PEGASUS model (google/pegasus-cnn_dailymail). Tokenizes input to sub-word token IDs. Encoder produces contextual document embeddings. Decoder generates abstractive summary through beam search (size=4, max 128 tokens).

3. BERT Contextual Embedding Module

Loads bert-base-uncased. Computes contextual token embeddings per sentence via mean pooling. Builds document centroid embedding as mean of all sentence embeddings. Computes cosine similarity between each keyword and document centroid.

4. YAKE Statistical Scoring Module

Computes keyword scores based on word casing, position, frequency, relatedness to context, and DifSentence score. Lower YAKE scores indicate higher keyword importance. Language-agnostic and domain-independent.

5. Firefly Algorithm Optimization Module

Population of 20 fireflies representing (alpha_BERT, beta_YAKE) weight combinations. Fitness = keyword extraction F1-Score on validation set. Fireflies move toward higher-fitness neighbors over 50 generations. Converges to optimal domain-specific weighting.

6. Output and Evaluation Module

Displays generated summary and top-K keywords side-by-side. Computes ROUGE-1, ROUGE-2, ROUGE-L for summarization and Precision/Recall/F1 for keyword extraction. Supports downloadable JSON and text output and batch processing.

IV. RESULTS AND DISCUSSION

SUMMARIZATION AND KEYWORD EXTRACTION PERFORMANCE

Model	ROUGE-1	ROUGE-2	KW F1 (%)
TextRank (Baseline)	33.2	14.8	38.7
BERT-Extractive	41.9	19.7	44.1
Proposed (Multi-task)	44.8	21.3	52.3

The proposed system achieves ROUGE-1 of 44.8, ROUGE-2 of 21.3, and ROUGE-L of 41.7 on the CNN/DailyMail test set. For keyword extraction, the system achieves F1 of 52.3% on Inspec, compared to TF-IDF (38.7%) and YAKE (44.1%). Qualitative analysis shows that joint training improves summarization coherence by leveraging keyword salience signals, while the shared encoder improves keyword ranking accuracy. The joint model achieves a 23% reduction in total parameters compared to two independent models.

1. Text Summarization Metrics (ROUGE)

Because abstractive summarization (using PEGASUS) generates new sentences, evaluating its quality requires comparing the generated summary against a human-written reference summary. The standard metric for this is **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation).

A. ROUGE-N (ROUGE-1 & ROUGE-2)

ROUGE-N measures the overlap of *N-grams* (word sequences of length N) between the system-generated summary and the reference summary.

- **ROUGE-1** measures the overlap of unigrams (single words). Your system achieved 44.8.
- **ROUGE-2** measures the overlap of bigrams (two-word phrases). Your system achieved 21.3.

$\text{ROUGE-N_Recall} = \text{Overlapping_N-grams} / \text{Total_N-grams_in_Reference}$

B. ROUGE-L

ROUGE-L measures the **Longest Common Subsequence (LCS)** between the generated and reference summaries. It captures sentence structure and word order without requiring consecutive matches.

$\text{ROUGE-L} = \text{Length_of_Longest_Common_Subsequence} / \text{Total_Words_in_Reference}$

2. Keyword Extraction Metrics

Keyword extraction is treated as an Information Retrieval task. We evaluate how well the extracted keywords match the ground-truth (human-annotated) keywords.

- **True Positives (TP):** Correctly extracted keywords.
- **False Positives (FP):** Extracted words that are NOT actual keywords.
- **False Negatives (FN):** Actual keywords that the system failed to extract.

A. Precision

Answers the question: *Out of all the keywords the system extracted, how many were actually correct?*

$$\text{Precision} = \text{Correctly_Extracted_Keywords} / \text{Total_Extracted_Keywords}$$

B. Recall

Answers the question: *Out of all the true keywords in the document, how many did the system successfully find?*

$$\text{Recall} = \text{Correctly_Extracted_Keywords} / \text{Total_Actual_Keywords}$$

C. Keyword F1-Score

The harmonic mean of Precision and Recall. Your Firefly Algorithm directly optimizes this metric as its fitness function, achieving a final F1-Score of 52.3%.

$$\text{F1-Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

3. Keyword Scoring & Firefly Algorithm Operations

Your methodology (Step 5-12) outlines a hybrid scoring approach that combines deep learning contextual embeddings (BERT) with statistical features (YAKE), optimized via the Firefly Algorithm.

A. BERT Contextual Score (Cosine Similarity)

The system calculates how semantically close a specific word is to the overall topic of the document.

- E_i = BERT contextual embedding vector for word i .
- $E_{\{doc\}}$ = Document centroid vector (the mean of all word embeddings).

$$\text{BERT_Score} = \text{DOT_PRODUCT}(\text{Word_Embedding}, \text{Document_Centroid}) / (\text{MAGNITUDE}(\text{Word_Embedding}) * \text{MAGNITUDE}(\text{Document_Centroid}))$$

B. Final Weighted Fusion (Firefly Optimization)

The Firefly algorithm finds the optimal weights (α^* and β^*) to combine the deep learning score and the statistical score into a final ranking metric.

- α^* = Optimal weight for the BERT score.
- β^* = Optimal weight for the YAKE score.

$$\text{Final_Keyword_Score} = (\text{Optimal_Alpha} * \text{BERT_Score}) + (\text{Optimal_Beta} * \text{YAKE_Score})$$

V. CONCLUSION AND FUTURE WORK

This paper presented a Transformer-based multi-task framework for simultaneous text summarization and keyword extraction. The joint architecture achieves superior performance on both tasks while maintaining computational efficiency. Future work will explore multilingual extension, domain-specific fine-tuning for scientific and legal documents, and long document processing using sparse attention mechanisms.

References

- [1] A. Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.
- [2] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, 2019.
- [3] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)," JMLR, 21(140), 2020.
- [4] Y. Liu and M. Lapata, "Text Summarization with Pretrained Encoders," EMNLP, 2019.
- [5] J. Zhang et al., "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," ICML, 2020.
- [6] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," EMNLP, 2004.
- [7] R. Campos et al., "YAKE! Keyword Extraction from Single Documents Using Multiple Local Features," Information Sciences, 2020.