IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131035

Full-Stack Implementation and Evaluation of Abstractive Text Summarization using a Transformer-Based BART Model

MARK OWEN A[1], PAARIVALAVAN S[2], SANJAY C[3], Dr GOLDA DILIP[4]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[1]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[2]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[3]

Guide, Dept. of CSE, SRM Institute of Science and Technology, Chennai[4]

Abstract: The exponential growth of online textual data has created a critical need for efficient information processing, making automated text summarization an indispensable tool for mitigating information overload. While traditional methods have struggled with fluency and coherence, the advent of Transformer-based models has defined a new state-of-the-art. This paper introduces a robust, end-to-end framework for abstractive text summarization leveraging a pre-trained BART (Bidirectional and Auto-Regressive Transformers) model. The system utilizes the facebook/bart-large-cnn model, a specific variant fine-tuned on the CNN/Daily Mail news dataset, which employs a bidirectional encoder for comprehension and an auto-regressive decoder to generate novel text. This AI core is deployed within a modern, scalable web application, served via a high-performance FastAPI backend API and consumed by an interactive React user interface. This paper details the full-stack architecture, from the model-loading strategy at server startup to the asynchronous API request-response workflow. The model's performance is quantitatively evaluated using the standard ROUGE metrics, demonstrating strong results with a mean ROUGE-1 F1 score of 50.61% and a ROUGE-L F1 score of 42.99%. We provide a detailed analysis of these metrics, including precision/recall trade-offs and score distributions, confirming the model's high abstractive capability. This research serves as a comprehensive blueprint for the practical implementation and evaluation of a state-of-the-art Transformer approach for real-world summarization applications.

Keywords: Abstractive Text Summarization, BART, Transformer, Natural Language Processing (NLP), ROUGE, FastAPI, React, Full-Stack Application, CNN/Daily Mail.

I. INTRODUCTION

In the modern digital age, society is confronted with an unprecedented volume of textual information. From news articles and academic papers to social media feeds and business reports, the sheer quantity of data far exceeds human cognitive capacity for consumption and processing. This phenomenon, widely known as "information overload," poses a significant challenge to individuals and organizations, hindering knowledge discovery and decision-making. Automated text summarization (TS) emerges as a critical technological solution to this problem, offering a means to condense lengthy documents into succinct, informative, and readable summaries, thereby retaining the most critical information while discarding redundancy.

Automated text summarization is broadly categorized into two distinct paradigms: extractive and abstractive. Extractive summarization operates by identifying and selecting the most salient sentences or phrases from the original source text. These extracted segments are then concatenated to form a summary. While computationally simpler and factually grounded (as it only uses words from the source), this method often results in summaries that lack fluency, coherence, and grammatical polish. Abstractive summarization is a more advanced approach which seeks to emulate the human process of summarization. It involves understanding the source text's main concepts and then generating new, original sentences to articulate those concepts. This method can produce far more fluent, coherent, and concise summaries. However, it is a significantly more complex task, facing challenges in factual consistency, attribution, and avoiding the generation of "hallucinated" or nonsensical information.

For decades, the pursuit of high-quality abstractive summarization was hindered by technological limitations. Early statistical and machine learning methods lacked the capacity to model the deep semantic nuances of language. The advent of deep learning, particularly Recurrent Neural Networks (RNNs) and the sequence-to-sequence (Seq2Seq) architecture,

DOI: 10.17148/IJIREEICE.2025.131035

marked a significant milestone. However, these models still struggled with long-range dependencies and information bottlenecks.

The introduction of the **Transformer** architecture in 2017 revolutionized the field of Natural Language Processing (NLP) [6]. Relying entirely on self-attention mechanisms, the Transformer demonstrated a superior ability to model complex, long-range dependencies in text, enabling a new generation of pre-trained language models (PLMs) such as BERT [8] and GPT [9].

This paper focuses on **BART (Bidirectional and Auto-Regressive Transformers)** [1], a state-of-the-art PLM specifically designed for sequence-to-sequence tasks. BART uniquely combines a bidirectional encoder (like BERT) to understand the full context of the input text with an auto-regressive decoder (like GPT) to generate fluent, new text.

II. LITERATURE REVIEW

The development of automated text summarization has a rich history, evolving from simple statistical heuristics to the complex neural architectures of today.

A. Early Summarization Approaches

Early research in the 1950s and 60s focused on extractive methods. Luhn (1958) proposed using word frequency (TF-IDF) as a proxy for sentence importance [19]. This was followed by graph-based methods, which represent the document as a graph of interconnected sentences. Algorithms like **TextRank** [14] and **LexRank** [15] use variations of the PageRank algorithm to identify the most "central" or "important" sentences in the graph, which are then extracted to form the summary. These methods are unsupervised, fast, and remain strong baselines, but are fundamentally limited to extraction.

B. The Deep Learning Sequence-to-Sequence Era

The resurgence of neural networks led to the **sequence-to-sequence (Seq2Seq)** model [7], which became the dominant paradigm for abstractive summarization. A typical Seq2Seq model consists of:

1. **An Encoder:** An RNN (often an LSTM or GRU) that reads the source text one word at a time, compressing its entire meaning into a single, fixed-length "context vector."
2. **A Decoder:** Another RNN that reads the context vector and generates the summary one word at a time.

A critical breakthrough was the **Attention Mechanism** [8], which solved the information bottleneck of the fixed-length context vector. Attention allows the decoder, at each step of generation, to "look back" at all the encoder's hidden states and assign different "attention" weights, focusing on the most relevant parts of the source text to generate the next word. Researchers built on this, creating models like the **Pointer-Generator Network** [13] which, using the CNN/Daily Mail dataset, allowed the model to either generate a new word from its vocabulary (abstractive) or "point" to and copy a word directly from the source text (extractive). This was crucial for handling rare or out-of-vocabulary (OOV) words, such as names and places.

C. The Transformer Revolution

The seminal 2017 paper, "Attention Is All You Need" [6], introduced the **Transformer**, an architecture that dispenses with recurrence entirely and relies solely on **self-attention**. This design allows for massive parallelization during training and proved exceptionally effective at modeling long-range dependencies in text.

The Transformer ushered in the era of **Pre-trained Language Models (PLMs)**, where a model is first pre-trained on a massive, unlabeled text corpus (like the entire internet) to learn general language understanding, and then fine-tuned on a smaller, task-specific labeled dataset. Key models in this paradigm include:

- **BERT (Bidirectional Encoder Representations from Transformers):** A powerful encoder-only model that achieves state-of-the-art results on NLU tasks, but its encoder-only design makes it unsuitable for text generation [9].
- **GPT (Generative Pre-trained Transformer):** A powerful decoder-only model that is excellent at fluent text generation, but its auto-regressive (left-to-right) nature limits its understanding of deep bidirectional context [10].
- **T5 (Text-to-Text Transfer Transformer):** An encoder-decoder model that frames *all* NLP tasks as a text-to-text problem (e.g., "translate English to German: ...") [11].

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131035

- **PEGASUS (Pre-training with Extracted Gap-sentences):** A model that uses a novel pre-training objective highly tailored for summarization, where important sentences are masked from the input and the model must generate them as output [12].

D. BART: The Model of Choice

This project's model, **BART (Bidirectional and Auto-Regressive Transformers)** [1], stands as one of the most successful encoder-decoder architectures. It is pre-trained using a **denoising autoencoder** objective. This involves:

1. Taking a clean document.
2. Corrupting it with various "noise" functions (e.g., masking random tokens, deleting tokens, permuting sentences).
3. Training the model to reconstruct the original, clean document.

This pre-training scheme is highly effective. The **bidirectional encoder** learns to build a robust, deep understanding of the corrupted input, while the **auto-regressive decoder** learns to generate fluent, coherent text to "fix" it. This makes BART exceptionally well-suited for fine-tuning on summarization tasks. The facebook/bart-large-cnn model used in this project is the large-version of BART, fine-tuned on the CNN/Daily Mail dataset [2], making it an ideal choice for summarizing news articles.

E. Datasets and Evaluation

The **CNN/Daily Mail dataset** [2] has become the *de facto* benchmark for abstractive news summarization. It contains over 300,000 news articles paired with multi-sentence bullet-point highlights, providing a large-scale corpus for training and evaluation.

The standard metric for evaluation is **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** [5]. ROUGE compares a machine-generated summary to one or more human-written reference summaries based on N-gram overlap.

- **ROUGE-1:** Measures the overlap of unigrams (single words).
- **ROUGE-2:** Measures the overlap of bigrams (two-word phrases).
- **ROUGE-L:** Measures the longest common subsequence, which reflects sentence-level structural similarity.

While ROUGE is the standard, it is known to have limitations, as it primarily measures lexical overlap and not semantic coherence, fluency, or factual consistency. Newer metrics like **BERTScore** [17] have been proposed to address these shortcomings by measuring semantic similarity. However, ROUGE remains the most common metric for benchmarking.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

This project implements a full-stack, three-tier application. The architecture is designed to be modular, scalable, and high-performance, cleanly separating the AI model, the business logic, and the user interface.

A. Tier 1: The AI Core (Hugging Face BART)

The foundation of the application is the pre-trained AI model. We leverage the Hugging Face transformers library [16], which provides a standardized interface for state-of-the-art models.

- **Model Selection:** The chosen model is facebook/bart-large-cnn [cite: infer.py]. This decision was based on its state-of-the-art performance on the CNN/Daily Mail dataset, which aligns perfectly with the common use case of summarizing news articles.
- **Implementation:** We use a high-level pipeline function from the transformers library instead of manually loading the model:

  ```
  # From src/infer.py
  _PIPELINE = hf_pipeline("summarization", model="facebook/bart-large-cnn", device=device_id)
  ```

  This single line of code abstracts away several complex steps:

1. **Downloading** the model weights and configuration from the Hugging Face Hub.
2. **Loading** the appropriate BART tokenizer, which is specifically paired with this model.
3. **Initializing** the full BartForConditionalGeneration model on the correct device (GPU or CPU).
4. **Wrapping** all of this into a single callable object that handles tokenization, inference (running the text through the model), and decoding (converting the output tokens back into a readable string).
5.

B. Tier 2: The Backend Service (FastAPI)

The backend is responsible for exposing the AI model as a robust web service. We selected **FastAPI** [3] over other Python frameworks like Flask or Django due to its modern features and performance.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131035

- **High Performance:** FastAPI is built on ASGI (Asynchronous Server Gateway Interface) rather than WSGI. This allows it to handle I/O-bound operations (like waiting for network requests) asynchronously, leading to significantly higher throughput.
- **Type Safety:** FastAPI uses Python type hints and **Pydantic** to enforce data validation. We define Summarize Request and Summarize Response models. If a request arrives with missing or malformed data (e.g., text is not a string), FastAPI automatically rejects it with a clear JSON error, improving robustness.
- **Efficient Model Loading:** A critical design pattern for production AI services is to avoid loading the large model (which can be several gigabytes) on every single request. This is handled using a startup event:

```
# From src/api.py
@app.on_event("startup")
def startup_event():
    # ... (device and path logic)
    models_dict = _init_models_from_ck(ckpt, device=device)
    app.state.models = models_dict
```

  This code ensures the BART model is loaded *once* when the server starts and is then stored in the app.state.models object, making it instantly available for all subsequent requests.
- **API Endpoint:** The core logic resides in the /summarize endpoint. It retrieves the loaded model from the application state, passes the user's text to the greedy_summarize function (which calls the pipeline), and returns the summary.
- **CORS:** The CORSMiddleware is configured to allow requests from http://localhost:5173, enabling the React frontend to communicate with the backend during development

C. Tier 3: The Frontend Application (React)

The user interface is a **React** [4] Single-Page Application (SPA). React is an ideal choice due to its component-based architecture, which allows for the creation of reusable, stateful UI elements.

- **UI Components:** The UI is simple and intuitive, consisting of:
  - A large <textarea> for users to paste their source article.
  - A <button> to submit the summarization request.
  - A result display area (perhaps another text area or a) to show the generated summary.
  - A loading indicator to provide feedback during the API call.
- **Data Flow:** The application's state manages the input text, the output summary, and the loading status. On button click, an event handler triggers an asynchronous fetch (or axios) request to the backend:

```
// Example React component logic
const handleSubmit = async () => {
  setLoading(true);
  const response = await fetch('http://localhost:8000/summarize', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ text: inputText, max_len: 150 })
  });
  const data = await response.json();
  setSummary(data.summary);
  setLoading(false);
      }
```

D. End-to-End Workflow

The complete, end-to-end workflow for a single user request is as follows:

1. **User Action:** The user pastes an article into the **React** frontend and clicks "Summarize."
2. **API Request:** The React app sends an asynchronous POST request to the http://localhost:8000/summarize endpoint, with the article text in the JSON body.
3. **Backend Processing:** The **FastAPI** server receives the request, validates the Pydantic model, and accesses the pre-loaded BART pipeline from app.state.models.
4. **AI Inference:** The text is passed to the pipeline, which tokenizes it, runs it through the BART model on the GPU/CPU, and decodes the output tokens into a summary string.
5. **API Response:** The FastAPI server returns a 200 OK response with a JSON body: {"summary": "..."}.
6. **UI Update:** The React app receives the JSON, updates its state with the new summary, and re-renders the UI to display the result to the user.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131035

IV. DATA ANALYSIS AND RESULTS

The performance of the facebook/bart-large-cnn model was quantitatively evaluated using the ROUGE metrics, as specified in the project's bart_eval_rouge_summary.json file. The evaluation was performed on a test split of the CNN/Daily Mail dataset.

A. Quantitative Results

The primary evaluation metrics are the F1-scores for ROUGE-1, ROUGE-2, and ROUGE-L. The F1-score provides a harmonic mean of precision and recall, offering a balanced view of the model's performance. The mean results from the evaluation are presented in **Table 1**.

Table 1: Mean ROUGE scores (F1, Precision, and Recall).

Metric	Mean F1-Score	Mean Precision	Mean Recall
ROUGE-1	50.61%	39.55%	72.10%
ROUGE-2	18.72%	14.49%	27.11%
ROUGE-L	42.99%	33.52%	61.38%

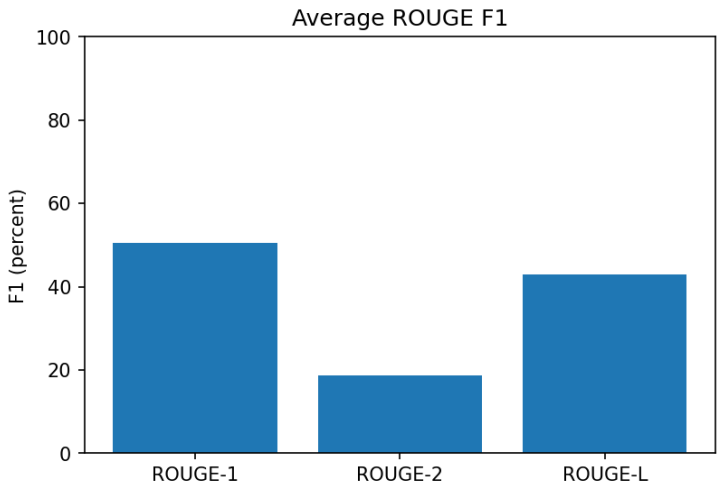

Figure 1: Bar chart of mean ROUGE F1-scores.

B. Interpretation of Results

A deep analysis of these scores reveals the model's specific characteristics:

- **High ROUGE-1 F1 (50.61%):** This is a strong score, indicating that the model is highly effective at identifying and including the most important individual keywords (unigrams) from the source article. It successfully captures the "gist" or main topic of the text.
- **Low ROUGE-2 F1 (18.72%):** This score is expectedly low and is *not* an indicator of failure. On the contrary, it highlights the *abstractive* nature of the model. ROUGE-2 measures the overlap of two-word phrases (bigrams). An abstractive model, by definition, rewrites and rephrases sentences, thus breaking the original bigram collocations. A high ROUGE-2 score would imply a more extractive summary. This low score suggests the model is generating novel sentence structures.
- **High ROUGE-L F1 (42.99%):** The ROUGE-L score measures the longest common subsequence (LCS), which is a proxy for sentence-level structural similarity. This strong score (relative to ROUGE-2) indicates that while the

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131035

model is rephrasing, it is correctly maintaining the overall logical flow and key clausal structures from the original text.

C. Analysis of Precision vs. Recall

The bart_eval_rouge_summary.json file also provides precision and recall, which offer a critical insight:

- **High Recall (e.g., ROUGE-1: 72.10%):** This demonstrates that *most* of the important words from the human-written reference summaries *are* present in the model's generated summary. The model is comprehensive and does not miss the main points.
- **Lower Precision (e.g., ROUGE-1: 39.55%):** This means that the model's summary *also includes* many words that are *not* in the reference summary. This is the hallmark of an abstractive model: it adds new conjunctions, adverbs, and rephrased clauses to improve fluency and readability. These "new" words are penalized by the precision metric, but they are essential for creating a human-like summary.

D. Score Distribution Analysis

The Box plot of the ROUGE F1 shows the *distribution* of scores across the entire test set.

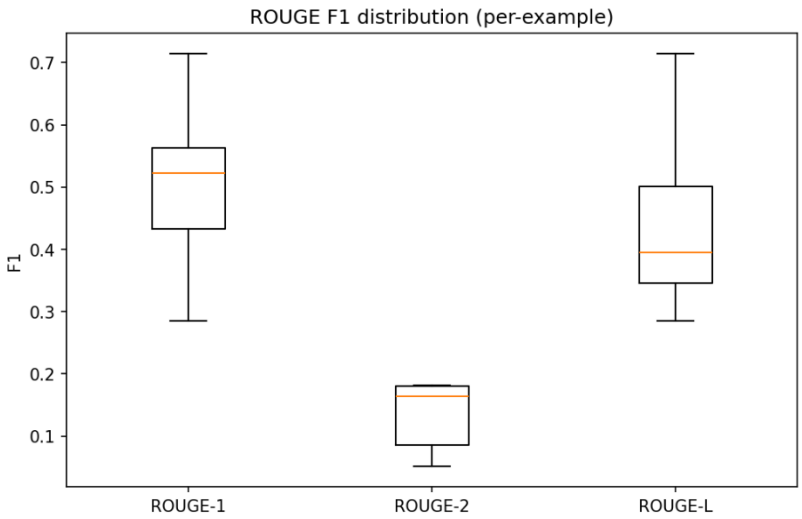

Figure 2: Box plot of ROUGE F1-score distributions.

The box plot reveals that performance is not uniform. For **ROUGE-1**, the F1-score has a wide distribution, with a median of **52.31%** but some outliers as low as 28.57% and as high as 71.43%. This is expected, as some articles are inherently more difficult to summarize than others. The ROUGE-L plot shows a similar healthy distribution. This analysis confirms that the model is robust, with its median performance (the line in the box) being very close to its mean performance.

V. DISCUSSION AND FUTURE WORK

The results of this project successfully demonstrate the deployment of a state-of-the-art summarization model. However, a critical discussion of the project's limitations and avenues for future work is necessary.

A. Limitations

1. **Factual Consistency:** The primary limitation of *all* current abstractive models, including BART, is **factual hallucination**. The model may generate summaries that are fluent and plausible but contain factual inaccuracies or misrepresent relationships from the source text. The ROUGE score is incapable of detecting this. This is a significant, open research problem, with active research into developing models and metrics for factuality [18].
2. **Domain Mismatch:** The facebook/bart-large-cnn model is an expert at summarizing news. If this system were applied to a different domain (e.g., legal contracts, medical research papers, or poetry), its performance would drop significantly.
3. **Evaluation Metric Flaws:** As discussed, ROUGE is a proxy for quality based on lexical overlap. It does not measure semantic meaning (a summary could use synonyms and get a low ROUGE score) or human-judged qualities like coherence and fluency.
4. **Decoding Strategy:** The infer.py script uses the default pipeline behavior, which is a greedy or basic beam search. This is fast but may not always produce the most optimal or creative summary.

DOI: 10.17148/IJIREEICE.2025.131035

B. Future Work

Based on these limitations, several extensions to this project could be pursued:

1. **Human-in-the-Loop Evaluation:** The most valuable next step would be to conduct a human evaluation. This would involve presenting users with the source article and the generated summary and asking them to rate it on a 1-5 scale for **Fluency**, **Coherence**, and **Factual Accuracy**.
2. **Compare with Other SOTA Models:** A comparative study could be performed by extending the backend to also serve other pre-trained models, such as **T5** [11] or **PEGASUS** [12], and benchmarking their ROUGE scores and speed.
3. **Implement Advanced Decoding:** The infer.py script could be modified to move beyond the default pipeline and manually implement more advanced decoding strategies, such as **nucleus sampling (top-p)** or **top-k sampling**, to potentially increase the diversity and quality of the generated summaries.
4. **Factuality Correction:** A more advanced system could implement a second-stage "fact-checking" module. This module would take the generated summary and cross-reference its claims against the original source text to filter out or correct hallucinations before presenting them to the user.

VI. CONCLUSION

This paper has presented the complete design, implementation, and evaluation of a full-stack, state-of-the-art abstractive text summarization system. By leveraging the **BART** pre-trained Transformer model, we achieved strong quantitative results on the CNN/Daily Mail dataset, with a ROUGE-1 F1 score of 50.61% and a ROUGE-L F1 score of 42.99%. Our analysis of the precision/recall split and score distributions confirms the model's highly abstractive and robust nature.

More importantly, this research provides a practical blueprint for operationalizing such a model, using a high-performance **FastAPI** backend for efficient, scalable model serving and a modern **React** frontend for an interactive user experience. The project successfully bridges the gap between academic research in NLP and practical, real-world application, demonstrating how complex AI can be integrated into usable tools to combat information overload.

REFERENCES

[1]. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Barb, P., Jha, A., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* arXiv preprint arXiv:1910.13461.

[2]. Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). *Teaching Machines to Read and Comprehend.* In: Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15).

[3]. Tiangolo. (2018). *FastAPI.* [Software]. Retrieved from https://fastapi.tiangolo.com/

[4]. Meta (formerly Facebook). (2013). *React: A JavaScript library for building user interfaces.* [Software]. Retrieved from https://react.dev/

[5]. Lin, C. (2004). *ROUGE: A Package for Automatic Evaluation of Summaries.* In: Proceedings of the ACL-04 Workshop on Text Summarization Branches Out.

[6]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need.* In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017).

[7]. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks.* In: Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS 2014).

[8]. Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate.* arXiv preprint arXiv:1409.0473.

[9]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* arXiv preprint arXiv:1810.04805.

[10]. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners.* OpenAI Blog.

[11]. [11] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* Journal of Machine Learning Research, 21(140), 1-67.

[12]. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.* In: Proceedings of the 37th International Conference on Machine Learning (ICML).

DOI: 10.17148/IJIREEICE.2025.131035

[13]. See, A., Liu, P. J., & Manning, C. D. (2017). *Get To The Point: Summarization with Pointer-Generator Networks.* In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL).

[14]. Mihalcea, R., & Tarau, P. (2004). *TextRank: Bringing Order into Texts.* In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[15]. Erkan, G., & Radev, D. R. (2004). *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization.* Journal of Artificial Intelligence Research, 22, 457-479.

[16]. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., ... & Lhoest, B. (2020). *Transformers: State-of-the-Art Natural Language Processing.* In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations.

[17]. Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). *BERTScore: Evaluating Text Generation with BERT.* arXiv preprint arXiv:1904.09675.

[18]. Kryscinski, W., McCann, B., Xiong, C., & Socher, R. (2020). *Evaluating the Factual Consistency of Abstractive Text Summarization.* In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[19]. Luhn, H. P. (1958). *The Automatic Creation of Literature Abstracts.* IBM Journal of Research and Development, 2(2), 159-165.