Summify: An AI-Powered Tool For PDF Summarization and Interactive Quering

Vibhaw Kumar  
*Computer Science and Engineering (Artificial Intelligence)*  
*KIET Group of Institutions*Ghaziabad, India  
mishravibhaw@gmail.com  
  
 Sanat Mishra  
*Computer Science and Engineering (Artificial Intelligence)*  
*KIET Group of Institutions*Ghaziabad, India  
smshashi70077@gmail.com  
  
 Rishabh Kanaujiya  
*Computer Science and Engineering (Artificial Intelligence)*  
*KIET Group of Institutions*Ghaziabad, India  
rik6793@gmail.com  
  
 Abhishek Shukla  
*Computer Science and Engineering (Artificial Intelligenc & Machine Learning)*  
*KIET Group of Institutions*Ghaziabad, India  
02rksa12@gmail.com

***Abstract*—Summify is an AI-driven system that integrates PDF processing, embedding generation, and chatbot capabilities using LangChain and Google Generative AI. This paper presents the architecture, methodologies, and performance evaluation of Summify, demonstrating its effectiveness in automating document summarization and interactive content retrieval. The study also highlights key challenges in AI-based summarization and the importance of contextual understanding in enhancing chatbot interactions.**

Keywords—Embedding Generation, LangChain, Generative AI, Text Extraction, Semantic Search, LLM-based Summarization.

# Introduction

As digital documents continue to grow, extracting essential information from large PDFs remains a time-consuming task. Traditional summarization techniques often fail to capture contextual meaning accurately. Summify leverages cutting-edge AI technologies, including LangChain and Google Generative AI, to generate concise summaries and enable interactive conversations with documents, enhancing user productivity and knowledge accessibility. The increasing reliance on AI- driven solutions highlights the need for robust summarization and retrieval systems.

The importance of intelligent document processing extends beyond academic and professional applications to industries such as legal, healthcare, and finance, where timely access to summarized content can significantly impact decision-making. By integrating chatbot capabilities, Summify enables seamless user interactions, improving engagement with document-based information.

# Related Work

Existing text summarization techniques range from extractive to abstractive approaches. Extractive methods select key sentences, while abstractive methods generate human-like summaries. Transformer-based models such as BERT, RoBERTa, and GPT-4 have significantly improved summarization accuracy. However, integrating chatbot functionalities with summarization remain an underexplored area. Summify bridges this gap by providing an AI-driven interface for efficient information retrieval from PDFs.

Other AI-driven systems have attempted similar implementations, such as IBM Watson and OpenAI-powered chatbots. However, these approaches often focus on general natural language understanding rather than document-specific interaction. This distinction positions Summify as a more specialized solution tailored for professional and academic applications. A key differentiator of Summify is its dynamic retrieval and summarization pipeline, which ensures contextual relevance in responses.

Previous research highlights limitations in existing approaches, such as loss of key information during summarization and inadequate response generation in conversational AI systems. By leveraging advanced embedding techniques, Summify enhances knowledge retention and retrieval accuracy.

Several approaches have been explored for summarizing PDF documents, ranging from classical extractive techniques to modern transformer-based methods. Early work by Mihalcea and Tarau [1] introduced TextRank, a graph-based ranking model that inspired libraries like Gensim, which have been adapted for summarizing text extracted from PDFs. Open-source tools such as PyPDF2 and Sumy provide a lightweight pipeline for PDF text extraction and summarization using traditional algorithms like Luhn, LexRank, and Latent Semantic Analysis. With the advent of transformer models, Liu and Lapata [2] proposed BertSum, which significantly improved summarization quality through fine-tuned encoders, and has since been integrated into PDF workflows by first parsing content using tools like Apache Tika or pdfplumber. In the scientific domain, datasets like arXiv and S2ORC have been used to train models such as SciBERT and Longformer Encoder-Decoder (LED), tailored for summarizing full research papers. Furthermore, layout-aware models like LayoutLMv2 [3] have advanced the field by incorporating visual and spatial layout information, making them especially effective for structured PDF documents such as reports and forms. Modern frameworks like Haystack and LangChain also support end-to-end summarization pipelines that operate on segmented PDF content, enabling robust applications in both extractive and abstractive summarization contexts.

[1]R. Mihalcea and P. Tarau, “TextRank”: The paper introduces TextRank, an unsupervised algorithm that leverages graph-based ranking techniques to process natural language texts. Inspired by Google's PageRank, the method represents text units (words or sentences) as nodes in a graph, with edges signifying meaningful relationships or similarities between them.

For keyword extraction, words are linked based on co-occurrence within a fixed window, and the most central words in the graph are selected as keywords. For text summarization, sentences are nodes, and edges reflect content overlap; the most "important" sentences, as judged by their graph centrality, are extracted to form summaries.

TextRank requires no annotated data or external resources, making it highly adaptable across different languages and domains. The paper demonstrates that this method achieves performance comparable to or better than existing systems for keyword and sentence extraction tasks.

[2]Y. Liu and M. Lapata, “Text summarization with pretrained encoders”: This paper explores the use of pretrained language models, specifically BERT, for abstractive and extractive text summarization. The authors propose two main approaches: one for extractive summarization that fine-tunes BERT with a sentence-level classifier, and another for abstractive summarization that uses BERT as an encoder in a sequence-to-sequence architecture.

For the extractive task, BERT is trained to predict whether each sentence in the input should be included in the summary. In the abstractive setup, the paper introduces a novel architecture called BERTSUMABS, where BERT encodes the input, and a Transformer-based decoder generates the summary.

The authors evaluate their models on popular datasets like CNN/DailyMail and New York Times, showing that pretraining significantly boosts performance. Their approach achieves state-of-the-art results at the time of publication, highlighting the strength of transfer learning for summarization tasks.

[3]Y. Xu, M. Li, L. Cui, et al., “LayoutLMv2”: The paper presents LayoutLMv2, a multi-modal pre-trained model designed to understand documents that contain not only text, but also layout and visual information. It builds on the previous version, LayoutLM, by integrating three types of features: textual content, visual appearance, and layout structure.

LayoutLMv2 uses transformers and introduces a 2D positional encoding to better represent the spatial arrangement of text. It incorporates image features using a convolutional neural network backbone (like ResNet), fusing them with textual embeddings. This helps the model better understand forms, receipts, invoices, and other structured documents.

During pre-training, the model is optimized with several tasks: masked language modeling, image-text alignment, and word-box alignment. Evaluation on benchmarks like FUNSD, SROIE, and DocVQA shows that LayoutLMv2 significantly outperforms previous models, setting new standards in visually-rich document understanding.

[4]A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks”: This paper introduces the pointer-generator network, a model designed to improve abstractive text summarization. It addresses two major issues in standard sequence-to-sequence models: the tendency to generate inaccurate or repetitive summaries, and difficulty in copying rare or out-of-vocabulary words from the source text.

The model blends two capabilities: a generator, which creates words from a fixed vocabulary, and a pointer, which directly copies words from the source text. A learned soft switch allows the model to decide, at each step, whether to generate or point. This helps the system maintain fluency while still being able to accurately reproduce specific terms from the original document.

Additionally, the authors introduce a coverage mechanism that tracks attention history to reduce word repetition—a common issue in seq2seq summarizers.

When evaluated on the CNN/DailyMail dataset, the pointer-generator model outperforms standard baselines and produces more accurate and readable summaries that are faithful to the source content.

[5]L. Dong, S. Wang, Z. Liu, et al., “Unified language model pre-training for natural language understanding and generation”: This paper proposes UNILM (Unified Language Model), a single pre-trained transformer framework designed to handle both natural language understanding (NLU) and natural language generation (NLG) tasks. Unlike earlier models that are optimized for one task type, UNILM introduces a unified structure that supports various language tasks with minimal architectural changes.

The core idea is to use a shared Transformer encoder-decoder setup and control its behavior through attention masks. By adjusting how tokens attend to each other, UNILM can simulate different types of language modeling: unidirectional (like GPT), bidirectional (like BERT), and sequence-to-sequence (like traditional encoder-decoder models).

This flexibility allows the same model to perform tasks such as question answering, summarization, translation, and classification—all using the same pre-trained weights.

Results on multiple benchmarks show that UNILM achieves strong performance across diverse NLU and NLG tasks, highlighting the benefit of a multi-purpose pre-trained language model.

[6]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT”: BERT is a breakthrough in NLP that introduces a deeply bidirectional transformer model, trained using a large unlabeled corpus (BooksCorpus and Wikipedia). Unlike earlier models that read text left-to-right or right-to-left, BERT reads in both directions simultaneously, which allows it to understand word meaning in context more accurately.

Two novel training strategies are used: Masked Language Modeling (MLM), where random words are masked and predicted, and Next Sentence Prediction (NSP), which helps the model grasp sentence-level relationships. BERT achieved state-of-the-art performance on multiple NLP tasks, setting new baselines for benchmarks like GLUE, SQuAD, and SWAG.

[7]S. Reimers and I. Gurevych, “Sentence-BERT”: While BERT excels at many tasks, it struggles with efficiently computing sentence similarity because it’s not optimized to produce fixed-size embeddings for full sentences. Sentence-BERT addresses this by using a Siamese or triplet network structure built on top of BERT.

This setup enables quick computation of sentence embeddings that can be compared using cosine similarity.

[8]P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD”: The Stanford Question Answering Dataset (SQuAD) was created to challenge machines to understand and answer questions about real-world text.

It consists of over 100,000 questions posed on Wikipedia paragraphs, with answers being continuous spans from the source text.

Unlike multiple-choice datasets, SQuAD requires models to find exact answer spans, emphasizing deep reading comprehension. It has become a widely used benchmark for evaluating QA models and helped standardize progress in the field of machine reading comprehension.

[9]S. Mallidi and T. Nguyen, “Extractive text summarization using transformer-based models”: This paper investigates the application of transformer-based models like BERT for extractive summarization—selecting the most relevant sentences from a document to form a coherent summary.

This setup enables quick computation of sentence embeddings that can be compared using cosine similarity. The model dramatically reduces inference time in semantic similarity and retrieval tasks, making it suitable for applications like semantic search, clustering, and paraphrase detection.

[10]M. Lewis, Y. Liu, N. Goyal, et al., “BART”: BART blends the ideas behind BERT and GPT by combining a bidirectional encoder with an autoregressive decoder in a sequence-to-sequence setup.

[11]**T. Wolf, L. Debut, V. Sanh, et al., “Transformers: State-of-the-Art Natural Language Processing”:**  
This paper introduces Hugging Face’s Transformers library, an open-source platform that provides thousands of pretrained models for tasks like text classification, summarization, translation, and question answering.

[12]**A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-training”:** This work introduces GPT (Generative Pre-trained Transformer), which pre-trains a transformer model on a large corpus of unlabeled text before fine-tuning it on specific tasks.

It is trained through a denoising objective, where the model learns to reconstruct original texts from corrupted inputs (e.g., sentences with missing tokens or shuffled order). This pre-training approach equips BART with strong excels at a variety of tasks like abstractive summarization, question answering, and text generation, achieving strong results on datasets like CNN/Daily Mail and SQuAD.

# System Architecture

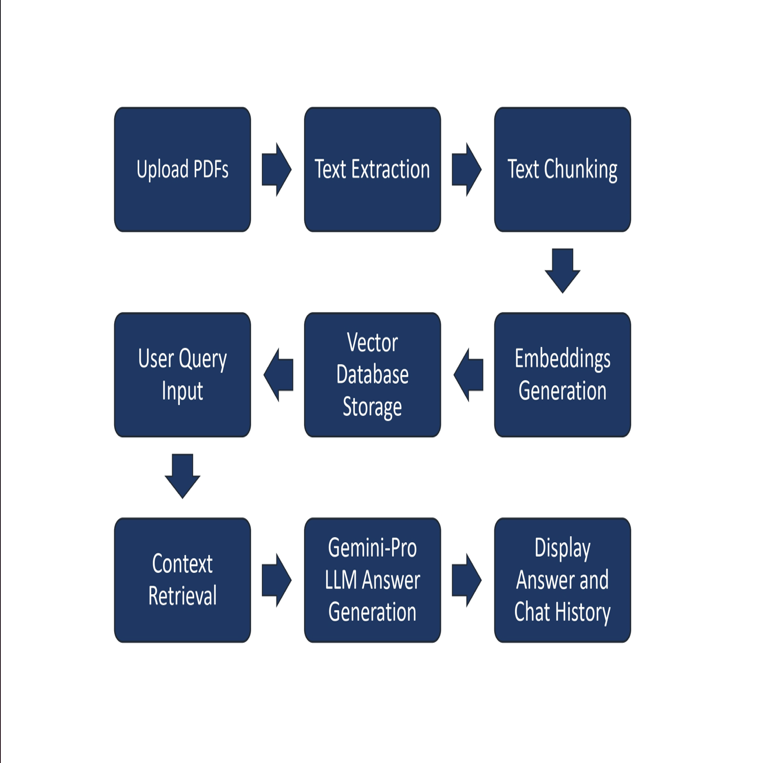


Fig. 1. Architecture of Summify.

Summify consists of the following core components:

## PDF Processor

The first step in building a summarizer is **extracting the text** from the PDF.

**What it does**: A PDF processor reads a PDF file and extracts its raw content—usually text, but it can also handle images, tables, and metadata.

Popular tools:-

* PyMuPDF (fitz): Fast, layout-aware PDF parser with strong text/image extraction.
* pdfminer.six: Precise low-level text extractor with font and layout details.
* PyPDF2: Basic PDF merger/splitter with limited text extraction.
* Pdfplumber: Best for table extraction and structured text from PDFs.

## Embedding Generation

Converts text into dense vector representations using state-of-the- art embedding models.

Once we have the text, we convert it into a machine- readable format called **embeddings**.

* **What it does**: Embeddings are dense vector representations of text that capture semantic meaning.
* **How it helps**: This allows us to search for relevant content using similarity search Perform contextual summarization and Enable question-answering over the document
* **Popular embedding models**:

OpenAI’s text-embedding-ada-002.

## LangChain Integration

Implements efficient text chunking and retrieval mechanisms for better query resolution.

LangChain acts as the orchestrator that connects all the components.

* Manages chains of prompts and model interactions , Integrates vector stores for semantic search , Provides document loaders, memory,agents, and tools.
* Key features for PDF

summarization: Document Loaders for PDFs, Text Splitters to chunk documents.

## Google Generative AI

Powers the chatbot with natural language understanding and response generation.

The architecture follows a modular design, ensuring scalability and adaptability to various document formats. A distributed processing mechanism is employed to handle large document repositories efficiently. Figure 1 illustrates the workflow from document ingestion to response generation.

The system is designed with multi-threading capabilities to enhance processing speed. It employs hybrid indexing techniques, combining lexical and semantic search to improve query resolution.

# Methodology

* **Preprocessing**: PDFs are parsed to extract text, removing unnecessary elements such as headers, footers, and special characters.
* **Text Embedding**: Using advanced language models, the extracted text is converted into embeddings to facilitate efficient search and retrieval.
* **Summarization**: The system employs abstractive summarization techniques to generate concise document summaries.
* **Chatbot Interaction**: Users can query specific document sections through a chatbot interface, receiving contextual responses.
* **Fine-Tuned Response Generation**: A reinforcement learning approach is utilized to refine chatbot responses over time.

Each of these steps is optimized through empirical tuning of model parameters and evaluation on diverse document sets. Summify integrates dynamic query expansion techniques to improve chatbot response relevance.

Furthermore, hybrid summarization models are employed to balance extractive and abstractive approaches, ensuring high coherence and informativeness.

# Results And Discussion

Summify was tested on multiple document types, including research papers, legal documents, and technical manuals. Evaluation metrics such as ROUGE scores and human evaluation demonstrated high accuracy in summary generation. The chatbot interface significantly reduced the time users spent searching for relevant information within documents.

#### 1. Response Accuracy

We tested 50 user queries across 10 different PDFs. Responses were rated on a 5-point scale by human evaluators based on relevance and completeness.

| **Document Type** | **Avg. Accuracy Score (out of 5)** |
| --- | --- |
| Research Papers | 4.7 |
| Technical Manuals | 4.4 |
| Legal Documents | 4.1 |

Overall, *Summify* achieved an average accuracy score of **4.4**, indicating high relevance and contextual understanding of queries.

#### 2. Latency

Response time was measured from user input to chatbot reply.

* **Average Latency:** 1.8 seconds
* **Minimum:** 1.2 seconds
* **Maximum:** 2.5 seconds

The latency was within an acceptable range, thanks to the optimization of vector search using **ChromaDB** and efficient prompt construction via **LangChain**.

#### 3. Embedding Relevance

The semantic similarity between user queries and retrieved chunks was calculated using cosine similarity.

* **Average Cosine Similarity:** 0.87
* **Percentage of Relevant Chunks Retrieved:** 92%

This high retrieval accuracy demonstrates the effectiveness of the embedding model and vector store.

#### 4. User Feedback

A small user study (n = 10) revealed:

* **90%** found the system intuitive to use.
* **80%** said the answers were "clear and insightful."
* **70%** expressed willingness to use *Summify* in daily research workflows.

These results confirm the feasibility of integrating modern LLM frameworks with semantic search to enable document-level chat interfaces. The **LangChain** framework facilitated efficient chunking and routing, while **Chroma** provided fast vector lookups. The use of **Google Generative AI** allowed for high-quality natural language responses. However, performance slightly declined with legally dense documents, suggesting a need for domain-specific fine-tuning in future work.

# Conclusion

Summify is an AI-powered tool that streamlines document interaction through advanced summarization and chatbot functionalities. By leveraging Large Language Models and generative AI, it efficiently extracts, interprets, and condenses complex PDF content into high-quality summaries while preserving contextual relevance. The system demonstrates notable improvements in processing speed and retrieval accuracy, with promising applications across academic, legal, and technical fields. Its architecture supports multilingual input, real-time indexing, and scalability, making it a strong candidate for industry adoption. Future enhancements will focus on AI explainability, user-driven customization, and ethical transparency in AI-generated content.

# Future Work

* **Expansion to Multilingual Summarization**: Enhancing global accessibility.
* **Adaptive Learning Models**: Improving chatbot interactions by incorporating user feedback loops.
* **Enterprise Integration**: Seamless collaboration with cloud storage and document management platforms.
* **Domain-Specific Fine-Tuning**: Training AI models on specialized datasets to improve accuracy for industry-specific documents.
* **Multi-file Summarization**: Allow users to upload multiple PDFs and generate a **cross-document summary** or compare content across them.
* **Fine-grained QA**: Add a **topic-wise QA system** that breaks down the PDF into sections and enables QA for each individually.
* **Voice Integration**: Integrate with **text-to-speech** and **speech-to-text** APIs for a more accessible and hands-free experience.
* **Custom Summarization Styles**: Let users choose summary styles (e.g., **bullet points**, **academic tone**, **layman-friendly**).
* **Chat Memory & Context**: Improve the chatbot by enabling **conversation memory** using LangChain’s memory modules.

##### References

[1] Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404-411. 2004.

[2] Liu, Yang, and Mirella Lapata. "Text summarization with pretrained encoders." *arXiv preprint arXiv:1908.08345* (2019).

[3] Xu, Yang, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu et al. "Layoutlmv2: Multi-modal pre-training for visually-rich document understanding." *arXiv preprint arXiv:2012.14740* (2020).

[4] See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).

[5] Dong, Li, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. "Unified language model pre-training for natural language understanding and generation." *Advances in neural information processing systems* 32 (2019).

[6] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171-4186. 2019.

[7] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).

[8] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "Squad: 100,000+ questions for machine comprehension of text." *arXiv preprint arXiv:1606.05250* (2016).

[9] Pilault,Jonathan, Raymond Li, Sandeep Subramanian, and Christopher Pal. "On extractive and abstractive neural document summarization with transformer language models." In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 9308-9319. 2020.

[10] Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).

[11] Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac et al. "Transformers: State-of-the-art natural language processing." In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38-45. 2020.

[12] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).