STM JOURNALS

International Journal of
Satellite Remote Sensing

Volume 3, Issue 2, 2025
July–December
DOI (Journal): 10.37591/IJSRS

https://journals.stmjournals.com/ijsrs

Research IJSRS

Webpage Extraction and Retrieval Chatbot

P.V. Kavitha[1,*], S. Vishal Kumar[2], S. Anush Kumar[3]

Abstract

Web scraping is a fundamental technique for automating data extraction in big data applications. While multiple implementations exist, few leverage Python's Beautiful Soup library for efficient and structured data retrieval. This project aims to develop a web scraper and retrieval system that extracts relevant information from web pages, stores it in a vector database (Milvus), and enables intelligent querying using semantic search and generative AI. The system is designed to scrape Wikipedia articles, extract textual content, and transform it into high-dimensional embeddings using a pre-trained Sentence-BERT model. These embeddings are indexed and stored in Milvus, an optimized similarity search engine. To retrieve information, user queries are embedded and matched against stored vectors, retrieving the most relevant content. The retrieved text is further enhanced with Google's Gemini API, generating refined and context-aware responses. A Fast API-based interface facilitates data loading, querying, and response generation, ensuring seamless interaction. The system was tested by scraping Wikipedia pages, storing data, and retrieving meaningful insights. The entire process, from extraction to response generation, demonstrated efficiency, completing within seconds. The primary limitation involves reliance on Wikipedia's structure, limiting adaptability to highly dynamic web pages. Additionally, Beautiful Soup, optimized for speed, does not extract all embedded data, such as dynamically loaded JavaScript content. Future enhancements could involve expanding into broader websites, incorporating real-time data updates, and integrating advanced natural language processing (NLP) models for improved comprehension. This implementation is beneficial for developers exploring web scraping, AI-driven search applications, and small-scale data analytics projects.

Keywords: Web scraping, data extraction, Beautiful Soup, semantic search, natural language processing (NLP), Sentence-BERT, vector databases, Milvus, FastAPI, generative AI, Google Gemini API, information retrieval, knowledge discovery, big data analytics, data visualization, content embedding, and AI-driven search

***Author for Correspondence**
P.V. Kavitha
E-mail: kavitha.krishna@srec.ac.in

[1]Assistant Professor, Department of Artificial Intelligence and Data Science, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India
[2]Student, Department of Artificial Intelligence and Data Science, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India
[3]Student, Department of Artificial Intelligence and Data Science, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India

Received Date: July 10, 2025
Accepted Date: August 13, 2025
Published Date: December 31, 2025

Citation: P.V. Kavitha, S. Vishal Kumar, S. Anush Kumar. Webpage Extraction and Retrieval Chatbot. International Journal of Satellite Remote Sensing. 2025; 3(2): 1–7p.

INTRODUCTION

Web scraping is essential for automating data extraction and enabling applications in search engines, AI-driven retrieval, and data analytics. This project developed a webpage extraction and retrieval system using BeautifulSoup [1] for structured web scraping, a sentence transformer for embedding text, Milvus for vector-based storage [2], and semantic search. The content extracted from Wikipedia is stored as embeddings, allowing the efficient retrieval of relevant information. Google Gemini API enhances responses for better readability. A Fast API interface enables seamless data loading and querying. While effective, its limitations include handling dynamically loaded content. Future improvements will include real-time updates and broader website support.

LITERATURE SURVEY

Vector database management systems fundamental concepts, use-cases, and current Challenges: This study focuses on the process of high-dimensional data, such as images, text, and videos, enabling advanced similarity-based search operations crucial for AI applications. They enhance scalability and performance using indexing techniques such as LSH and HNSW, while supporting hybrid queries that combine metadata filtering with vector searches. However, VDBMSs face challenges, such as high computational costs, trade-offs between accuracy and speed, lack of standardization, and limited ACID compliance. Effective data management requires specialized expertise to optimize indexing, storage, and query performance [3].

Milvus: A purpose-built vector data management system: This study focuses on Milvus, a high-performance vector database optimized for efficient storage and retrieval of high-dimensional data. It offers exceptional speed, scalability, and real-time data handling, rendering it ideal for AI/ML applications. Milvus supports advanced query processing, including multi-vector searches and attribute filtering, while leveraging both CPU and GPU acceleration. However, it requires significant computational resources, a complex setup, and expertise in vector indexing. Additionally, its evolving ecosystem and high memory/storage overhead present challenges for large-scale deployment [4].

A study of sentence similarity based on the All-MiniLM-L6- v2 Model: This study investigates the use of the All-MiniLM-L6-v2 model for sentence similarity tasks, demonstrating notable performance improvements, with fine-tuning boosting precision from 0.74 to 0.91. The model offers efficiency advantages over BERT, is faster and less resource-intensive, and is compatible with few-shot learning, making it effective even with limited labeled data. It also performs well when handling sentences with varying structures that convey similar meanings. However, this study has limitations, such as the small dataset used, potential overfitting in few-shot learning, and inability to support categorical attributes, which may restrict its applicability in certain NLP tasks [5].

Model Algorithm Research based on Python FastAPI

This paper analyzes the advantages and challenges of using FastAPI for deploying algorithmic models, such as RESTful APIs. FastAPI enables rapid development with minimal boilerplate, offering high performance through asynchronous processing and seamless scalability. Auto-documentation and standardized API integration enhance maintainability and cross-platform usability. However, challenges such as increased latency owing to network overhead, security risks, dependency on Python, and resource-intensive deployments must be considered. In addition, complex deployments, potential bottlenecks in high-frequency requests, and the learning curve of asynchronous programming pose limitations [6, 7].

WEBPAGE EXTRACTION AND RETRIEVAL USING BEAUTIFULSOUP

The proposed system is designed to efficiently extract, store, and retrieve relevant information from webpages using a combination of web scraping, vector-based storage, and AI-powered search. The system begins by utilizing Beautiful Soup to scrape structured content from Wikipedia pages, ensuring that only meaningful textual data is extracted while removing unnecessary elements, such as scripts and HTML tags. Once extracted, the text is processed and converted into high-dimensional vector embeddings using a sentence transformer, which is a pre-trained deep learning model that captures semantic meanings [8, 9]. These embeddings, along with the original text, were then stored in Milvus using Docker, a high-performance vector database optimized for similar searches. When a user submits a query, the system generates an embedding for the query and searches for Milvus to find the most semantically similar text snippets. The top-matching results were retrieved and further refined using Google's Gemini API, which enhanced the responses to ensure coherence and readability. A Fast API-based interface allows users to interact with the system seamlessly, enabling them to load data, submit queries, and receive AI-enhanced responses in real-time. This approach ensures an automated, efficient, and scalable solution for extracting and retrieving web-based information. The system offers several advantages, including its ability to function without manual intervention and its ability to provide AI-

enhanced responses that improve search quality. The use of Milvus for vector-based retrieval allows for fast and scalable performance, making it well-suited for applications in AI-driven search engines, knowledge discovery, and intelligent information retrieval systems. Future enhancements may include support for real-time updates, multisource scraping, and more advanced NLP techniques, further expanding its capabilities.

Working Principle

Figure 1 shows that the web application is designed such that it uses FastAPI [7], one for loading the data and the other for querying the data.

FastAPI for Loading Data

The Load Data Endpoint is designed to enable users to provide a Wikipedia URL from which relevant data can be scraped, processed, and stored in a vector database for future querying. This is discussed in the following section.

Web Scrapping

Using FastAPI, when the user enters the URL of webpages such as Wikipedia, web scraping is performed at the back end. Web scraping uses Beautiful Soup [10, 11] along with Python's request library to efficiently extract structured data from webpages. The process begins by sending an HTTP request to the target webpage and retrieving the HTML content if the request is successful. Using Beautiful Soup, it parses HTML and extracts relevant text from paragraph (<p>) tags, ensuring that only meaningful content is retained. To enhance the quality of the extracted data, a cleaning process is applied using regular expressions to remove unwanted elements, such as HTML tags, citations, and special characters [3, 5].. This ensures that the extracted content is well-structured and free from noise.

Load Into Vector Database

To efficiently store and retrieve the extracted data from Wikipedia, the cleaned text is converted into vector representations and saved in a vector database [3] for queries. The extracted sentences were transformed into numerical embeddings using the sentence-transformers/all-MiniLM-L6-v2 model [5] that captures the semantic meaning of the text. This 6-layer transformer-based model was optimized to generate a 384-dimensional sentence embedding. These embeddings help find similar content based on meaning. Once generated, the vectors were stored in Milvus [4, 12, 13] using the Docker platform. In Milvus, a vector database is created to store vectors in the form of a field schema that consists of three fields: primary_id to ensure a unique identifier, embeddings to store vector embeddings, and text to store the original embedding corresponding to each embedding.

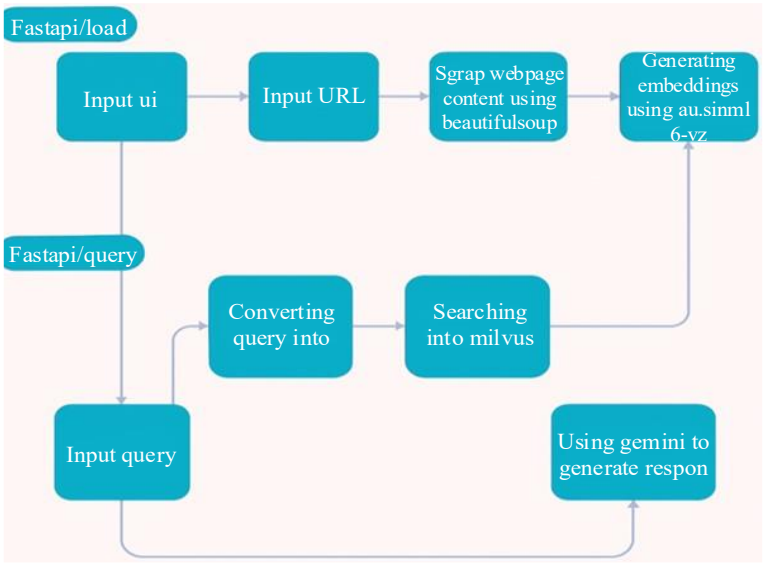

Figure 1. Block diagram for identifying duplicate prescriptions.

FastAPI for Querying Data

Generative AI Model

After the extracted data are stored in the vector database through the Fast API, a Generative AI model (Gemini 1.5 Flash)

It was used to generate answers to the user queries [14]. The retrieved context is combined with the user query to create a prompt based on the most relevant data from the vector database. This prompt is then sent to the Gemini 1.5 Flash model, which processes it and generates a coherent contextually relevant answer [11]. The pre-trained nature of the Gemini model enables it to comprehend the relationship between the query and extracted context, producing accurate and context-aware responses. This delivers meaningful and appropriate answers to user queries based on the stored data. The Gemini LLM model works by accessing API keys.

PERFORMANCE EVALUATION

The model was evaluated using precision, recall, and F1 scores, which measure accuracy, correct identification, and overall performance balance, respectively.

Accuracy Metrics

Precision: Precision is the accuracy of the model's predictions. The model extracts relevant sentences for queries. Precision checks how many of these predictions are correct. For example, if the model predicts 10 medicine names and seven of them are correct, the precision is 7/10 = 70%.

Recall: Recall focuses on how thorough the model is in finding all the relevant entities. It checks whether the model has missed any important information from the extracted sentences. For example, if it extracts relevant answers for 10 questions and the model correctly identifies eight, the recall is 8/10 = 80%.

F1-score: This is a balance between precision and recall. It combines both into a single score, showing how well the model performs overall in extracting the prescription information.

MODEL COMPARISON

Table 1 presents a performance comparison between the Gemini 1.5 Flash model and GPT-2 for processing web-scraped data using Beautiful Soup and answering the related questions. The Gemini 1.5 Flash model, with its advanced NLP capabilities, demonstrated a significantly higher F1-score compared to GPT-2, particularly in extracting structured information from web pages and generating coherent, context-aware responses. Gemini excelled in handling complex queries, maintaining factual accuracy, and understanding multistep reasoning tasks, making it more suitable for question-answering applications based on scraped data. In contrast, GPT-2 performed adequately on simple fact-based retrieval but struggled with long-form content, multisection parsing, and consistency in multiturn interactions. Although both models showed similar performance in basic query answering, Gemini's superior precision and recall make it the preferred choice for tasks involving automated data extraction and contextual question answering.

The Gemini 1.5 Flash model, with its advanced retrieval-augmented generation (RAG) capabilities, demonstrated significantly higher accuracy than GPT-2, particularly in retrieving semantically relevant information.

It effectively handled complex queries, provided accurate results even with partial matches, and leveraged a deep semantic search to identify the most relevant data points. By contrast, GPT-2 struggled with retrieval efficiency, often relying on surface-level keyword matching rather than deep semantic understanding, leading to less relevant or incomplete responses when querying the vector database.

Table 2 presents a performance comparison between the sentence-transformers/all-MiniLM-L6-v2 model and the BERT-based model [12] for sentence similarity tasks. The MiniLM-L6-v2 model,

optimized for efficiency and speed, demonstrated a significantly higher overall F1-score compared to the BERT-base. Specifically, MiniLM-L6-v2 outperformed BERT-base in capturing semantic similarities between sentences, while maintaining lower computational costs, making it ideal for large-scale NLP applications. However, BERT-base, with its deeper contextual representations, showed comparable performance in scenarios that required a more nuanced understanding of sentence structures. The results highlight MiniLM-L6-v2's advantage in precision and recall, particularly in tasks demanding real-time efficiency and scalability (Table 3).

The sentence-transformers/all-MiniLM-L6-v2 model, optimized for efficient sentence similarity tasks, delivered significantly better performance, with precision, recall, and F1 score values of 82.67. This demonstrates its ability to accurately capture semantic relationships while maintaining its computational efficiency. In contrast, the BERT-based model, designed for broader NLP applications, struggles with real-time sentence similarity computations [13, 15]. Its performance is notably lower, with all three metrics (precision, recall, and F1-score) falling to just 71.89, highlighting its limitations in balancing accuracy and processing speed for large-scale applications (Figures 2 & 3).

Table 1. Evaluation metrics of MiniLM-L6-v2.

Metric	MiniLM-L6-v2
Precision	82.67
Recall	79.57
F1-Score	80.38

Table 2. Model comparison between Gemini and Mistral.

Feature	Gemini	GPT2
Architecture	Multimodal, transformer (audio, video, text)	Transformer based (text only)
Model type	Decoder-only transformer with multimodal support	Decoder-only transformer (text generation)
Developer	Google AI	OpenAI
Fine-tuning	Not supported	Fine-tuning is possible
Key strength	Advanced language understanding	Exceptional conversation ability

Table 3. Model comparison between MiniLM-L6-v2 and BERT.

Metric	MiniLM-L6-v2	BERT
Precision	82.67	71.89
Recall	79.57	74.34
F1-Score	80.38	78.22

Figure 2. Fast API/load.

Figure 3. Fast API/query.

CONCLUSION

In conclusion, web scraping is essential for efficiently extracting and structuring valuable information from online sources, enabling data-driven decision-making across various domains. By systematically retrieving and processing data from Wikipedia, this system facilitates seamless knowledge extraction, storage in a vector database, and intelligent querying using generative AI. The model demonstrated strong performance, achieving high precision, recall, and F1 scores, highlighting its reliability in accurately extracting, storing, and retrieving information relevant to user queries.

REFERENCES

1. Latif R, Abodayeh A, Hejazi R, Najjar W, Shihadeh L. Web scraping for data analytics: a BeautifulSoup implementation. 2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia. 2023. p. 65–69. doi: 10.1109/WiDS-PSU57071.2023.00025.
2. Bansal P, Ouda A. Study on integration of FastAPI and machine learning for continuous authentication of behavioral biometrics. 2022 International Symposium on Networks, Computers and Communications (ISNCC), Shenzhen, China. 2022. p. 1-6. doi: 10.1109/ISNCC55209.2022.9851790.
3. Taipalus T. Vector database management systems: Fundamental concepts, use-cases, and current challenges. Cognitive Systems Research. 2024;85:101216. doi: 10.1016/j.cogsys.2024.101216
4. Wang J, Yi X, Guo R, Jin H, Xu P, Li S, Wang X, Guo X, Li C, Xu X, Yu K, Yuan Y, Zou Y, Long J, Cai Y, Li Z, Zhang Z, Mo Y, Gu J, Jiang R, Wei Y, Xie C. Milvus: a purpose-built vector data management system. In: Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21); 2021 Jun 20–25; Virtual Event, China. New York (NY): Association for Computing Machinery; 2021. p. 2614–2627. doi:10.1145/3448016.3457550.
5. Yin C, Zhang Z. A study of sentence similarity based on the all-MiniLM-L6-v2 model with "same semantics, different structure" after fine-tuning. In: Proceedings of the 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024); 2024 Aug 9–11; Singapore. Dordrecht (Netherlands): Atlantis Press; 2024. p. 677–684.
6. Chitturi K. Understanding vector embeddings in AI search: a technical deep dive. Int J Comput Eng Technol. 2024;15:1725–1733. doi:10.34218/IJCET_15_06_147.
7. Chen J. Model algorithm research based on Python FastAPI. Front Sci Eng. 2023;3:7–10. doi:10.54691/fse.v3i9.5591.
8. Sirisuriya SCM de S. Importance of web scraping as a data source for machine learning algorithms: a review. 2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS), Peradeniya, Sri Lanka. 2023. p. 134–139. doi: 10.1109/ICIIS58898.2023.10253502.
9. Matise J. Simmering data: using Beautiful Soup and Python to scrape data from web pages. Paper 119-2022. Chicago (IL): NORC at the University of Chicago; 2022.

10. Pant S, Yadav EN, Milan SM, Sharma M, Bedi Y, Raturi A. Web scraping using Beautiful Soup. 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), Chikkaballapur, India. 2024. p. 1–5. doi:10.1109/ICKECS61492.2024.10617017.
11. Huan X, Zhou H. Integrating advanced language models and vector databases for enhanced AI query retrieval in web development. Int J Adv Comput Sci Appl. 2024;15. doi:10.14569/IJACSA.2024.0150601.
12. Devika R, Vairavasundaram S, Mahenthar CSJ, Varadarajan V, Kotecha K. A deep learning model based on BERT and sentence transformer for semantic keyphrase extraction on big social data. IEEE Access. 2021;9:165252–165261. doi:10.1109/ACCESS.2021.3133651.
13. Sugandhika C, Ahangama S, Ahangama S. Modelling Wikipedia's information quality using informativeness, reliability, and authority. In: Proceedings of the International Conference on Advanced Computing (ICAC); 2021. doi:10.1109/ICAC54203.2021.9671092.
14. Pande A, Patil R, Mukkemwar R, Panchal R, Bhoite S. Comprehensive study of Google Gemini and text generating models: understanding capabilities and performance. Grenze Int J Eng Technol. 2024 Jun;2:856–863.
15. Galli C, Donos N, Calciolari E. Performance of four pre-trained sentence transformer models in the semantic query of a systematic review dataset on peri-implantitis. Information. 2024;15:68. doi:10.3390/info15020068.

