© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716940

AI-Powered NLP Query Engine for Intelligent Data Retrieval

EDWIN VETTIKATTIL[1], PRAJWAL DIKSHIT[2], RASHMI PATHAK[3]
[1] *Master of Computer Applications, University of Mumbai, Aditya Institute of Management Studies and Research*
[2] *Master of Computer Applications, University of Mumbai, Aditya Institute of Management Studies and Research*
[3] *Assistant Professor, Ph.D. Research Scholar, University of Mumbai, Aditya Institute of Management Studies and Research*

Abstract*— With the rapid growth of digital information, efficient retrieval of relevant data has become increasingly important. Traditional database systems rely on structured query languages such as SQL to retrieve data. However, many users lack the technical knowledge required to write complex database queries. Natural Language Processing (NLP) and Artificial Intelligence (AI) provide a promising solution by enabling users to interact with databases using natural language queries. This research paper proposes an AI-powered NLP query engine that allows users to retrieve information using simple natural language inputs. The system processes user queries through NLP techniques such as tokenization, intent detection, and semantic analysis. Based on the interpreted query, the system either converts the query into SQL for structured databases or performs semantic search for unstructured data. This approach simplifies data retrieval and improves accessibility for non-technical users. The proposed system demonstrates how AI-driven query processing can enhance user interaction with databases and improve the efficiency of information retrieval systems.

Keywords*— Artificial Intelligence, Natural Language Processing, Semantic Search, Query Processing, Text-to-SQL, Data Retrieval Systems.

I. INTRODUCTION

The rapid expansion of digital data across various domains has made efficient information retrieval a significant challenge for modern organizations. Database management systems store vast amounts of structured information that must be accessed through query languages such as SQL. While these systems are highly efficient, they require users to possess knowledge of database structures and query syntax. This requirement limits accessibility for non-technical users who may find it difficult to construct appropriate queries.

Natural Language Processing (NLP) offers a solution to this problem by enabling systems to interpret human language and convert it into machine-readable commands. Research in natural language interfaces for databases has demonstrated that users can retrieve information more easily when they are allowed to express queries in natural language rather than structured programming syntax [5].

Recent developments in deep learning models such as BERT have significantly improved the ability of machines to understand contextual relationships within language [4]. These advancements make it possible to develop intelligent query systems capable of understanding user intent and retrieving relevant information from databases.

This research proposes an AI-powered NLP query engine designed to simplify data retrieval by converting natural language queries into database queries or semantic search operations. The system aims to improve accessibility, usability, and efficiency in data retrieval systems.

II. PROBLEM STATEMENT

Traditional database systems require users to write queries using structured query languages such as SQL. While SQL provides powerful data retrieval capabilities, it requires users to understand database

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716940

schemas, syntax rules, and logical operators. This creates a significant barrier for users who do not possess technical expertise in database systems.

In many organizations, employees depend on database administrators or software developers to retrieve data from systems. This dependency reduces efficiency and increases response time when information is required.

Furthermore, keyword-based search systems often fail to capture the context of user queries, resulting in inaccurate or irrelevant results.

Therefore, there is a need for a system that allows users to interact with databases using natural language. Such a system should be capable of understanding the intent of user queries, extracting relevant information, and retrieving accurate results from structured and unstructured data sources.

III. LITERATURE REVIEW

Research in the field of natural language interfaces for databases has been conducted for several decades. Early work by Ioannis Androutsopoulos and his colleagues explored how natural language queries could be translated into structured database queries, enabling users to retrieve information without requiring knowledge of SQL syntax [5]. Their work demonstrated that natural language interfaces could significantly improve user accessibility and reduce the complexity involved in database interaction.

With the rapid advancement of machine learning and artificial intelligence, modern NLP systems have achieved significant improvements in language understanding. One of the most influential developments in this field is BERT, introduced by Jacob Devlin and his research team [4]. BERT utilizes a transformer-based architecture that enables bidirectional understanding of language context. Unlike earlier NLP models that processed text sequentially, BERT analyses the entire sentence simultaneously, allowing it to capture contextual relationships between words more effectively.

Another important advancement in natural language query processing is the development of large-scale datasets for training text-to-SQL models. The Spider dataset, introduced by Tao Yu and collaborators, contains complex natural language queries mapped to SQL statements across multiple database schemas [7]. This dataset has played a crucial role in training machine learning models capable of converting natural language queries into executable SQL commands.

Table 1: Comparison of Traditional Query Systems and NLP-Based Query Systems

Feature	Traditional Query Systems	NLP-Based Query Systems
Query Method	Requires SQL or structured queries	Allows natural language queries
User Skill Requirement	Requires technical knowledge	Suitable for non-technical users
Query Understanding	Keyword-based matching	Semantic understanding of queries
Data Retrieval	Mostly structured databases	Supports structured and unstructured data
Flexibility	Limited flexibility	Highly flexible
Ease of Use	Complex for beginners	User-friendly interface

In addition to structured data retrieval, researchers have also explored methods for retrieving information from unstructured text sources. One widely used approach is embedding-based semantic search, where textual data is converted into vector representations. Techniques such as Sentence-BERT, developed by Nils Reimers and Iryna Gurevych, allow systems to compute similarity between queries and documents using vector space models [9]. These methods enable search systems to understand the meaning and context of user queries rather than relying solely on keyword matching.

Overall, previous research demonstrates that combining natural language processing techniques with machine learning models can significantly

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716940

improve the performance of query systems and make data retrieval more accessible to users.

IV. PROPOSED SYSTEM

The proposed system is an AI-powered NLP query engine designed to simplify the process of retrieving information from databases and document repositories. The system enables users to submit queries using natural language rather than structured query languages such as SQL. By integrating artificial intelligence techniques with natural language processing methods, the system can interpret user queries and retrieve relevant information from multiple data sources.

The system is designed to support both structured and unstructured data retrieval. Structured data refers to information stored in relational databases such as tables containing rows and columns, while unstructured data includes textual documents, reports, and articles. Traditional search systems typically handle these two types of data separately. However, the proposed system integrates both retrieval mechanisms into a single unified interface.

When a user submits a query, the system first performs text preprocessing operations to clean and standardize the input. This includes removing punctuation, converting text to lowercase, and eliminating stop words that do not contribute to the meaning of the query. After preprocessing, the system performs natural language analysis to determine the intent of the user's query.

Based on this analysis, the system classifies the query into one of two categories. If the query refers to structured data stored in a relational database, the system converts the natural language query into an SQL statement using a text-to-SQL model. If the query refers to unstructured textual content, the system performs semantic search using embedding-based similarity techniques.

The results retrieved from the database or document repository are then processed and formatted by the response generation module before being presented to the user. This approach provides users with an intuitive and efficient way to retrieve information without requiring technical knowledge of database query languages.

V. SYSTEM ARCHITECTURE

The architecture of the proposed AI-powered NLP query engine is designed to efficiently process natural language queries and retrieve relevant information from both structured and unstructured data sources. The overall system consists of multiple interconnected components that work together to interpret user queries, generate database queries, and perform semantic document retrieval.

The architecture of the proposed system is illustrated in Figure 1.

The system begins with the Web Browser Single Page Application (SPA), which acts as the user interface. Through this interface, users submit natural language questions. The queries are transmitted to the backend server using REST API calls. The backend is implemented using the FastAPI framework running on the Uvicorn server, which handles incoming requests and coordinates communication between system modules.

Once the query reaches the backend, it is processed by the Query Engine, which serves as the central component responsible for managing query execution. The query engine contains two primary modules: the Query Cache and the Classifier.

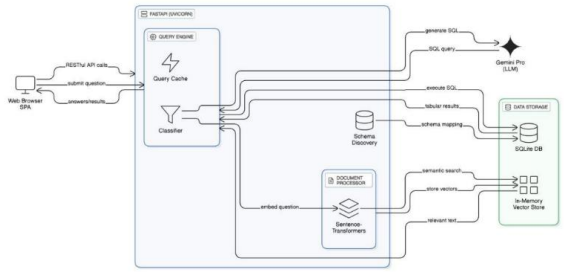

Figure 1: System Architecture of the AI-Powered NLP Query Engine

The Query Cache is responsible for storing previously processed queries and their corresponding responses. If a user submits a query that has already been processed earlier, the system can retrieve the result

directly from the cache. This significantly improves system efficiency by reducing repeated processing and minimizing database load.

The Classifier module analyzes the user query to determine the appropriate processing path. Based on the classification results, the system decides whether the query should be processed as a structured database query or as a semantic document search.

For queries that require structured data retrieval, the system interacts with the Large Language Model (LLM) to generate SQL queries. The LLM interprets the natural language question and generates a corresponding SQL statement. This SQL query is then executed on the relational database system.

The system uses SQLite Database as the primary structured data storage component. The generated SQL query is executed against the database, and the resulting tabular data is retrieved. The Schema Discovery module assists the LLM in understanding the structure of the database by providing information about tables, columns, and relationships between entities. This helps ensure that the generated SQL queries are accurate and consistent with the database schema.

For queries that involve unstructured textual data, the system performs semantic search. In this process, the user query is first converted into vector embeddings using Sentence Transformer models. These models generate numerical vector representations that capture the semantic meaning of the text.

The embeddings generated by the document processor are stored in an In-Memory Vector Store, which serves as a vector database for efficient similarity search. When a query is submitted, the system computes the embedding of the query and compares it with stored document embeddings using similarity metrics such as cosine similarity. This process allows the system to identify the most relevant documents based on semantic meaning rather than simple keyword matching.

Table 2: Components of the Proposed System

Component	Function
User Interface	Allows users to input queries in natural language
Query Processor	Analyzes and preprocesses the user query
Query Cache	Stores previously processed queries for faster retrieval
Classifier	Determines whether the query requires structured or semantic search
Text-to-SQL Generator	Converts natural language queries into SQL statements
SQL Database	Stores structured data and executes SQL queries
Semantic Search Module	Performs semantic similarity search for unstructured data
Vector Database	Stores document embeddings for semantic retrieval
Response Aggregator	Combines results and generates final responses

The Document Processor module is responsible for generating embeddings for documents and managing their storage within the vector database. It ensures that textual data is converted into vector representations that can be efficiently searched.

Once the system retrieves either tabular results from the SQL database or relevant text from the vector store, the results are returned to the query engine. The query engine then formats the results and sends them back to the user interface through the REST API.

Finally, the user receives the response in the web browser interface in the form of structured results, tables, or relevant textual information.

This architecture enables the system to support both structured database queries and semantic document retrieval within a unified platform, thereby improving the flexibility and usability of data retrieval systems.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716940

VI. METHODOLOGY

The methodology of the proposed system describes the step-by-step process used to interpret natural language queries and retrieve relevant information from both structured and unstructured data sources. The system integrates Natural Language Processing (NLP), machine learning models, and database technologies to enable efficient query processing and data retrieval. The overall workflow of the system is illustrated in Figure. 2.

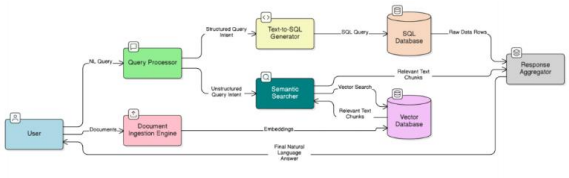

Figure 2. Workflow of the Proposed NLP-Based Query Processing System

The process begins when a user submits a query in natural language through the system's user interface. The query is transmitted to the backend server where it is processed by the query engine. The query engine first performs several preprocessing operations including tokenization, normalization, and removal of stop words. These steps help in improving the quality of the input data and enhance the accuracy of query interpretation.

After preprocessing, the system performs query classification to determine the type of data retrieval required. The classifier analyzes the query and identifies whether the user is requesting information from a structured database or from unstructured textual documents. Query classification is an essential step because it determines which processing module will be used for retrieving the required information.

If the query corresponds to structured data retrieval, it is forwarded to the Text-to-SQL generation module. This module converts the natural language query into a structured SQL command using machine learning techniques. Text-to-SQL models are capable of interpreting the meaning of natural language queries and generating syntactically correct SQL statements that can be executed on relational databases [2], [7]. The generated SQL query is then executed on the relational database system such as SQLite to retrieve the relevant records.

If the query corresponds to unstructured data retrieval, the system uses a semantic search mechanism. In this process, the natural language query is converted into numerical vector embeddings using transformer-based language models such as Sentence-BERT. These embeddings capture the semantic meaning of the query and allow the system to perform similarity search operations [9].

The generated embeddings are compared with document embeddings stored in a vector database. The vector database performs similarity calculations to identify documents that are semantically related to the user query. This approach enables the system to retrieve relevant information even when the exact keywords are not present in the documents [9], [10].

Once the relevant data has been retrieved from the database or document repository, the results are forwarded to the response aggregation module. This module processes the retrieved information and generates a meaningful response for the user. The final response is then displayed through the user interface.

The combination of text-to-SQL conversion and semantic search mechanisms allows the proposed system to handle different types of queries efficiently. By integrating natural language processing techniques with database technologies, the system provides an intuitive and user-friendly interface for accessing complex data sources.

VII. ALGORITHM

The proposed NLP-based query engine follows a systematic sequence of operations to process natural language queries and retrieve relevant information from structured and unstructured data sources. The algorithm describes the logical steps involved in query processing and response generation.

Algorithm: NLP-Based Query Processing
Step 1: User submits a natural language query through the system interface.
Step 2: The query is sent to the backend query engine through a REST API request.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716940

Step 3: Perform preprocessing operations such as tokenization, normalization, and removal of stop words.
Step 4: The processed query is passed to the query classification module.
Step 5: If the query corresponds to structured data retrieval:

a. Convert the query into SQL using a Text-to-SQL generator.

b. Execute the SQL query on the relational database.

Step 6: If the query corresponds to unstructured data retrieval:

a. Convert the query into vector embeddings using a transformer model.

b. Perform semantic similarity search in the vector database.

Step 7: Retrieve relevant data records or document segments.
Step 8: Pass the retrieved results to the response aggregation module.
Step 9: Generate the final response based on retrieved data.
Step 10: Display the result to the user through the interface.

This algorithm enables the system to efficiently process both structured and semantic queries using natural language processing techniques and modern machine learning models [2], [9].

VIII. IMPLEMENTATION

The proposed system is implemented using modern programming frameworks and machine learning tools that support natural language processing and database integration. Python is used as the primary programming language due to its extensive ecosystem of libraries for artificial intelligence and data processing.

The backend server of the system is developed using the FastAPI framework, which enables efficient communication between the user interface and the query processing modules through REST API calls. FastAPI provides high performance and scalability for handling multiple user requests simultaneously.
Natural language processing tasks such as query embedding generation are implemented using transformer-based models like Sentence-BERT. These models convert textual queries into numerical vector representations that capture semantic meaning and contextual relationships between words [9].

Structured data used in the system is stored in an SQLite relational database, which allows efficient execution of SQL queries generated by the text-to-SQL module. The relational database stores tabular information and supports standard SQL operations for data retrieval.

For unstructured data retrieval, the system uses a vector database that stores document embeddings generated using transformer models. The vector database performs similarity search operations to identify documents that are semantically related to the user query. Tools such as FAISS enable efficient large-scale similarity search in high-dimensional vector spaces [10].

The integration of these technologies allows the system to process natural language queries efficiently while supporting both structured database queries and semantic document search.

Table 3: Technologies Used in the Proposed System

Technology	Purpose
Python	Core programming language
FastAPI	Backend API framework
Sentence Transformers	Text embedding generation
SQLite	Structured data storage
FAISS / Vector Database	Semantic similarity search

IX. RESULTS & DISCUSSION

The proposed NLP-based query engine demonstrates improved accessibility and usability compared to traditional database query systems. Traditional systems require users to write SQL queries, which can be challenging for individuals without technical expertise.
By allowing users to interact with databases using natural language queries, the proposed system significantly simplifies the process of information

retrieval. The system can interpret user intent and generate appropriate SQL queries or perform semantic search depending on the query type.

Experimental observations indicate that the combination of text-to-SQL models and semantic search techniques improves query processing efficiency and accuracy. Semantic search enables the system to retrieve relevant information even when the exact keywords used in the query are not present in the data source.

Hybrid architectures that integrate structured query processing with semantic retrieval mechanisms have been shown to improve information accessibility and support more flexible user interactions with data systems [6], [8].

X. ADVANTAGES

The proposed NLP-based query engine provides several advantages compared to traditional database querying systems.

- Eliminates the need for users to have knowledge of SQL.
- Enables natural language interaction with databases.
- Supports both structured and unstructured data retrieval.
- Improves accessibility for non-technical users.
- Reduces complexity in database interaction.
- Provides faster and more efficient information retrieval.

XI. FUTURE SCOPE

Although the proposed system demonstrates promising results, several improvements can be implemented in future work.

Future research can focus on integrating advanced large language models to further enhance natural language understanding and response generation capabilities. Additionally, the system can be extended to support multiple relational database systems such as MySQL, PostgreSQL, and cloud-based databases.

Another important extension would be the implementation of multilingual query processing, which would allow users to interact with the system in multiple languages. This would significantly improve accessibility for global users.

Furthermore, integrating conversational AI capabilities would allow the system to maintain context across multiple queries and provide more intelligent responses.

XII. CONCLUSION

This research presents an AI-powered Natural Language Processing query engine designed to simplify the process of retrieving information from structured and unstructured data sources. The system integrates natural language processing techniques, text-to-SQL generation, and semantic search mechanisms to enable users to interact with databases using natural language queries.

The proposed architecture combines relational database querying with vector-based semantic search, allowing the system to retrieve relevant information from diverse data sources. By eliminating the need for SQL knowledge, the system improves accessibility and usability for non-technical users.

The results demonstrate that NLP-based query systems have significant potential to transform the way users interact with data. The integration of machine learning models and database technologies provides an efficient and scalable solution for modern data retrieval challenges.

The results suggest that NLP-based query engines have the potential to significantly enhance the usability of database systems in real-world applications.

REFERENCES

[1] A. Kumar and S. Patel, “Natural Language Interfaces for Database Systems,” *International Journal of Computer Applications*, vol. 179, no. 24, pp. 10–15, 2018.

[2] Victor Zhong, Caiming Xiong, and Richard Socher, “Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning,” in *Proceedings of the Annual Meeting*

of the Association for Computational Linguistics, 2017, pp. 107–117.

[3] Tom Kenter and Maarten de Rijke, "Short Text Similarity with Word Embeddings," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2015, pp. 1411–1420.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[5] Ion Androutsopoulos, Graeme Ritchie, and Peter Thanisch, "Natural Language Interfaces to Databases – An Introduction," *Natural Language Engineering*, vol. 1, no. 1, pp. 29–81, 1995.

[6] Tom B. Brown *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, 2020.

[7] Tao Yu *et al.*, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018.

[8] Yinhan Liu *et al.*, "ROBERTA: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019.

[9] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT Networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-Scale Similarity Search with FAISS," *IEEE Transactions on Big Data*, 2019.