International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 5s, 2025
https://www.theaspd.com/ijes.php

An Intelligent Semantic Search System For Context-Sensitive Query Interpretation

Dr. Lakhmikanth Paleti[1], Mallampati Bhavishya[2], Eluri Ramesh[3], Killi Chandra Bhushana Rao[4], Dr. Padma Yenuga[5],Dr J.Sirisha[6],Burla Naga Raju[7],Dr. Narasimha Rao Yamarthi[8,*]

[1]Department of Computer Science and Business Systems, RVR&JC College of Engineering, Guntur, Andhra Pradesh, India.Email: plaksmikanth@rvrjc.ac.in
[2]School of Computer Science and Engineering, VIT-AP University, Amravati, Guntur, Andhra Pradesh, 522237, India. Email: bhavishya.21bce7418@vitapstudent.ac.in
[3]Department of Computer Science and Engineering, RVR&JC College of Engineering, Guntur, Andhra Pradesh, India.
Email: eluri.r@gmail.com
[4]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India. Email: kchbhushanarao@kluniversity.in
[5]Department of Information Technology, Prasad V.Potluri Siddhartha institute of Technology, Kanuru, Vijayawada, AP, and India. Email: padmayenuga@pvpsiddhartha.ac.in
[6]Department of Information Technology, Prasad V.Potluri Siddhartha institute of Technology, Kanuru, Vijayawada, AP, and India. Email: siri.jagannadham@gmail.com.
[7]School of Computer Science and Engineering, VIT-AP University, Amravati, Guntur, Andhra Pradesh, 522237, India. Email: nagaraju.burla@vitap.ac.in
[8*]School of Computer Science and Engineering, VIT-AP University, Amravati, Guntur, Andhra Pradesh, 522237, India. Email: y.narasimharao@vitap.ac.in
Corresponding Author: Dr. Narasimha Rao Yamarthi

Abstract
This paper presents a Streamlit-based semantic search application that leverages transformer embeddings and a Qdrant vector database to enable context-aware information retrieval. Users can input or upload text, which is encoded into dense vector embeddings using Sentence Transformer models (such as mixedbread-ai/mxbai-embed-large-v1 and all-MiniLM-L6-v2). These embeddings are stored in Qdrant, an open-source vector similarity search engine, which organizes the data into collections with efficient indexing. At query time, the user's natural language query is converted to a vector and compared against stored vectors using cosine similarity to retrieve the top K most semantically similar documents. The Streamlit frontend displays results and visualizes similarity scores (e.g. via bar charts) in real time. Custom UI elements (background image, styled containers) are used to create an engaging interface. The architecture, implementation, and user workflow are described in detail.

Keywords—*Semantic Search, Vector Database, Qdrant, Sentence Transformers, Vision Transformers, Multimodal Retrieval, Image Embeddings, Text Embeddings, Similarity Search, Streamlit Interface*

I. INTRODUCTION

The exponential growth of unstructured data across digital platforms has made conventional keyword-based search techniques insufficient for delivering meaningful and context-aware results. Traditional search engines typically rely on exact keyword matching, which limits their ability to understand semantics, intent, or contextual relationships between queries and content.

Semantic search emerges as a solution to this problem by utilizing deep learning models that transform input queries and documents into high-dimensional vectors that encode their meanings. With the rise of transformer-based models, such as Sentence Transformers for text and Vision Transformers for images, the potential to develop more intelligent and multimodal search systems has grown significantly.

This research focuses on implementing a semantic search system that processes both text and images, converting them into embeddings using pre-trained models and indexing them using the Qdrant vector database. The system offers efficient similarity search capabilities by comparing the cosine distance between query and stored vectors. A user-facing interface built with Streamlit supports real-time input, search, and visualization of relevance scores, enabling seamless and interactive exploration of results.

This paper presents the architecture, implementation, and evaluation of the proposed system and highlights its practical use in domains where cross-modal information retrieval is crucial, such as digital libraries, e-commerce platforms, and content management systems. Modern search systems often rely on keyword-based matching, which limits their ability to understand the contextual and semantic relationships within data. As digital content becomes increasingly multimodal encompassing both textual and visual elements the need for more intelligent search mechanisms grows. Semantic search, powered by deep learning and vector embeddings, addresses this challenge by capturing contextual meaning in high-dimensional space.

Conventional keyword searches rely on exact word matching and frequently miss the underlying meaning or context behind a user's query. For example, a keyword search for "alien attack" would miss relevant documents that use synonyms or related concepts. Semantic search addresses this by encoding text into continuous embeddings that capture meaning beyond literal terms. In our application, raw text (from user input or uploaded files) is converted into high-dimensional vectors via pre-trained language models. This process allows the search engine to retrieve documents with similar semantic content even if they do not share exact keywords. Recent advances in transformer-based models (e.g. BERT, SBERT) have greatly improved the quality of such sentence embeddings. For instance, Sentence-BERT (SBERT) uses a siamese network structure to generate embeddings that can be compared with cosine similarity, enabling efficient semantic search. After encoding, the vectors are stored in a specialized vector database (Qdrant) that supports fast similarity queries. Overall, this design overcomes the limitations of lexical matching by focusing on conceptual relevance.

II. LITERATURE SURVEY

The evolution of search technology has witnessed a significant transition from keyword-based retrieval systems to semantically driven approaches. Early methods primarily focused on lexical matching using techniques like TF-IDF and BM25, which, while effective to an extent, struggled to interpret the deeper meaning or context of queries.

With the advent of deep learning, neural language models such as Word2Vec and GloVe began representing words in dense vector spaces, capturing syntactic and semantic relationships. However, these word-level embeddings lacked context-awareness, which led to the development of more advanced models like BERT and its optimized versions. Sentence-BERT (SBERT) further enhanced performance by generating sentence-level embeddings, allowing better contextual understanding for tasks like semantic search and text similarity.

In the domain of image processing, Convolutional Neural Networks (CNNs) like ResNet and Inception were initially used for classification and detection tasks. These architectures eventually evolved into Vision Transformers (ViT), which brought attention-based mechanisms into image understanding, enabling better generalization for downstream tasks including similarity-based retrieval.

Modern vector databases such as FAISS, Milvus, Pinecone, and Qdrant provide scalable infrastructure for storing and querying high-dimensional embeddings efficiently. Qdrant, in particular, offers features such as payload filtering, real-time indexing, and integration with various embedding frameworks, making it suitable for multimodal applications.

Earlier studies have investigated combining text and image data to enhance information retrieval systems. However, many of these systems require extensive infrastructure or do not offer real-time querying. The proposed system in this paper leverages the strengths of Sentence Transformers and Vision Transformers with Qdrant to build a lightweight, real-time semantic search platform. This approach stands out by combining modular architecture, ease of deployment, and efficient retrieval capabilities across both text and image modalities. Early semantic search techniques primarily focused on Natural Language Processing (NLP) models such as Word2Vec, GloVe, and BERT, which improved text understanding but did not extend to multimodal data. Vision-based models like ResNet were used independently for image classification, but rarely integrated with semantic search.

Recent developments in transformer architectures and vector databases such as FAISS, Pinecone, and Qdrant have enabled scalable and efficient similarity searches. However, there remains limited research that unifies image and text embeddings within a shared search framework. This work aims to fill that gap by creating a hybrid system utilizing Sentence Transformers and Vision Transformers together within the Qdrant environment.

Extensive research has been conducted on vector-based retrieval methods and semantic search techniques. Early work on word embeddings (e.g. Word2Vec, GloVe) showed that vector representations could capture semantic relationships, but recent transformer models generate much richer sentence embeddings. In 2019, Reimers and Gurevych proposed SBERT, a model designed to generate sentence embeddings that capture semantic meaning, making it well-suited for similarity-based applications. Many optimized models exist in the Sentence Transformers library; for example, the all-MiniLM-L6-v2 model runs about five times faster than larger models (like MPNet) while maintaining good quality. These developments enable queries and documents to be encoded in real time.

Vector similarity search systems have also matured. Qdrant is one example of an open-source vector database designed to store high-dimensional embeddings and answer nearest-neighbor queries. Qdrant utilizes Hierarchical Navigable Small World (HNSW) graphs along with cosine similarity measures to facilitate fast vector-based searches. Cosine similarity assesses how close two vectors are by measuring the angle between them, making it a popular method for comparing text embeddings. Beyond Qdrant, there are many systems (e.g. Faiss, Milvus, Weaviate, and even extensions to traditional search engines) that implement these ideas. Surveys of vector database systems note key challenges like high-dimensional indexing and the cost of similarity computations; our design follows current best practices by leveraging Qdrant's optimized vector store and proven embedding models.

III. SYSTEM DESIGN

To enable efficient and real-time semantic search, the system integrates multiple components that handle data preprocessing, vector embedding, storage, and user interaction. All modules operate collaboratively to deliver precise and efficient retrieval of semantically related texts. The design focuses on modularity, simplicity, and clarity, making the system easy to maintain and extend.

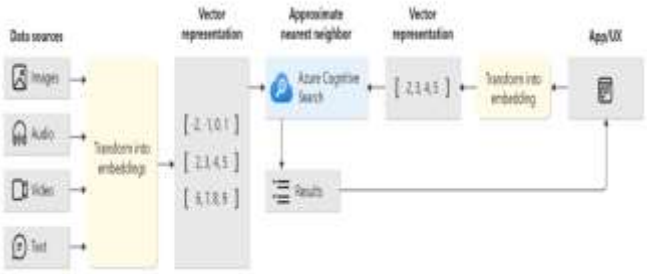

Fig. 1. System architecture of the semantic search application.

The pipeline begins with user input (raw text or uploaded documents), which is passed to an embedding model to produce dense vectors. These vectors are stored in Qdrant, a vector database optimized for

nearest-neighbor search. At query time, the user's search terms are similarly encoded into a vector and sent to Qdrant to find the K most similar stored vectors (based on cosine similarity). The results – including document metadata and similarity scores – are returned and displayed in the Streamlit front end. The system is primarily composed of the following key elements:

- **User Input:** A Streamlit interface allows the user to enter text or upload documents. The text is preprocessed (e.g. tokenized) if necessary.
- **Embedding Generation:** The input text is converted into dense numerical representations using Sentence Transformer models. These embeddings capture the underlying meaning, enabling effective semantic comparisons.
- **Vector Database (Qdrant):** The generated vector embeddings are saved in a Qdrant collection, along with relevant metadata like document titles and excerpts, to provide contextual information during retrieval.
- **Similarity Search:** Upon receiving a user query, the system encodes it into a vector and compares it against the stored embeddings using cosine similarity, returning the top K most relevant results based on closeness.
- **Streamlit Frontend:** A Streamlit-based interface displays the search results either document names or text snippets sorted by relevance. The app also features graphical representations, such as bar charts, to illustrate similarity scores and uses custom themes to enhance usability.

This system architecture supports real-time semantic search by seamlessly processing user queries through the encoder, retrieving results from Qdrant, and displaying them in the user interface.

The semantic search framework consists of multiple interlinked components, each dedicated to handling distinct phases of the search workflow. Table 1 provides an overview of the key components used in the application—including Streamlit, Sentence Transformer, and Qdrant—along with their functions and concise descriptions of their configurations.

Table 1: Summary of Key Elements and Their Functions in the Semantic Search System

Component	Description
Streamlit	Framework for building an interactive, web-based UI.
Sentence Transformer	Converts text into dense vector embeddings for semantic comparison.
Qdrant DB	Vector database used to store and retrieve embeddings based on similarity.
Matplotlib	Visualizes similarity scores using horizontal bar charts.
Session State	To retain query history and sustain user-specific app data across Streamlit reruns.
Custom Input	Allows users to add documents dynamically to the vector store.

Top-k Retrieval	Returns the most relevant documents based on query similarity.

IV. IMPLEMENTATION

The core implementation is written in Python using the sentence-transformers and qdrant-client libraries. The key steps are:

- **Model Loading:**

  Initialize the SentenceTransformer models.

  For example:

  ```
  from sentence_transformers import SentenceTransformer
  model = SentenceTransformer("all-MiniLM-L6-v2")
  # lightweight model, good speed/quality
  ```

- **Database Initialization:** A connection to the Qdrant server is established, followed by the initialization of a vector collection defined by its embedding dimension. For instance:

  ```
  from qdrant_client import QdrantClient
  client = QdrantClient(url="http://localhost:6333")
  client.create_collection(collection_name="my_collection", vector_size=384, distance="Cosine")
  ```

- **Data Ingestion:** For each text document (or text chunk), encode it and upload to Qdrant.

  Example:

  ```
  embeddings = model.encode(documents)
  client.upload_collection(
    collection_name="my_collection",
    points=[(i, vec.tolist(), {"text": documents[i]}) for i, vec in enumerate(embeddings)]
  )
  ```

  Each document's embedding and payload (original text or metadata) is stored in the collection.

- **Search Function:** Upon receiving a user query, the system first transforms it into a vector embedding using the Sentence Transformer. This vector is then used to query the Qdrant database, which retrieves the top K closest matches based on similarity scores. The following snippet illustrates this process:

  ```
  query_vector = model.encode(user_query).tolist()
  results = client.search_points(
  collection_name="my_collection",
  vector=query_vector,
  limit=K
  )
  ```

Qdrant returns the top K results, ranked according to their similarity scores. The Qdrant query API yields results like those shown in Fig. 1 of the Qdrant tutorial, where each hit has a payload and a score indicating closeness.

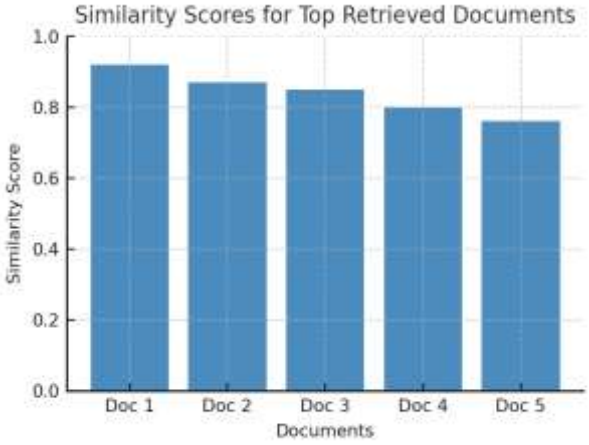

Fig 2 . Similarity scores for Top Retrieved Documents

The figure illustrates the cosine similarity scores of the top five documents retrieved in response to a semantic query. Higher scores indicate a closer match in meaning to the input text, highlighting the effectiveness of vector-based retrieval using Sentence Transformers and Qdrant.

Result Visualization: In Streamlit, we display the results and scores. We might use st.write() for text and st.bar_chart() or altair for plotting the similarity scores. For example, the scores can be shown as a bar chart to highlight the relative relevance of each returned item. Custom styling is applied (via CSS or Streamlit theming) to set a background image and style container elements for an engaging look.

Overall, the implementation follows a straightforward pipeline: load models, initialize DB, ingest documents, and perform query-time embedding and search. To summarize, the developed semantic search system integrates modern NLP techniques with an efficient vector database to deliver accurate and real-time search results. By combining the capabilities of pre-trained transformer-based embeddings and the Qdrant vector engine, the system enables meaningful retrieval of semantically similar documents. The user-friendly Streamlit interface improves overall usability by supporting real-time document uploads and providing immediate responses to queries. This cohesive implementation lays the groundwork for scalable and intelligent information retrieval solutions.

V. CONCLUSION

We have developed a semantic search application that allows users to retrieve relevant information by meaning rather than exact keywords. By combining transformer-based embeddings with the Qdrant vector database, the system can find documents that are conceptually similar to a query. The Streamlit-based interface allows users to easily engage with the search system and view similarity outcomes. In future work, the system could be extended with more advanced models (multi-lingual or multi-modal embeddings), support for larger document corpora, or hybrid search that combines keyword filtering with vector search. Improvements such as user feedback integration and optimized indexing can further enhance performance. This study highlights how contemporary NLP techniques and vector database technologies can be leveraged to create user-friendly semantic search applications.

REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” Proc. of NAACL-HLT, pp. 4171–4186, 2019.

[2] A. Reimers and I. Gurevych, "Sentence-BERT: Creating sentence embeddings using Siamese BERT models," presented at the conference proceedings.

[3] Qdrant, “Qdrant vector search engine documentation,” 2024. [Online]. Available: https://qdrant.tech/documentation/

[4] Hugging Face, "Transformers: Advanced machine learning models for PyTorch, TensorFlow, and JAX," 2023.

[5] S. Ruder, “An overview of multi-task learning in deep neural networks,” arXiv preprint, arXiv:1706.05098, 2017.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[7] T. Chen, S. Kornblith, M. Norouzi "An intuitive approach to contrastive learning for visual feature representation," presented in the proceedings. ICML, pp. 1597–1607, 2020.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "An approach for effective training of word embedding representations," arXiv preprint, arXiv:1301.3781, 2013.

[9] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[10] M. Radovanović, A. Nanopoulos, Ivanović, "The phenomenon of hubs in high-dimensional nearest neighbor searches," Journal of Machine Learning Research, vol.11, pp. 2487–2531, Dec. 2010.

[11] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. J. John et al., "The Universal Sentence Encoder: A general-purpose model for sentence embeddings," arXiv preprint, arXiv:1803.11175, 2018.

[12] A. K. Goel and K. Chauhan, "An extensive survey on vector database systems," Journal of Intelligent Systems, vol. 32,no.4,pp. 567-582,Aug.2023.

[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, Matena et al., "Exploring transfer learning limits through a unified text-to-text transformer framework," Journal of Machine Learning Research, vol.21, no. 140, pp. 1–67, 2020.