Analyzing Embedding Models for Embedding Vectors in Vector databases

Paras Nath Singh
Dept. of CSE,
CMR institute of Technology,
Bengaluru, India-560037
Sr. IEEE Member
drpn.singh@cmrit.ac.in

Sreya Talasila
Alumni, Dept. of CSE,
CMR institute of Technology,
Bengaluru, India-560037
sreyatalasila1@gmail.com

Shivaraj Veerappa Banakar
VTU Research Centre, Dept. of CSE,
CMR Institute of Technology,
Bengaluru, India-560037
shivu076@gmail.com

***Abstract*— Vector databases have emerged as the computation engine that enables us to successfully interact with vector embeddings in our applications as a result of the exponential rise of vector embeddings in disciplines like NLP (Natural Language Processing), computer vision, and other AI applications. In order to address the issues that arise when handling vector embeddings in production applications, vector databases were specifically created. Vector databases that offer quick and precise nearest-neighbor search, clustering, and similarity matching, and that are simple to deploy on cloud infrastructure or distributed computing systems, are more likely to be well-liked by users. They therefore provide a number of advantages over conventional scalar-based databases and independent vector indexes. This research reveals that embedding vectors are frequently utilized for analyzing and exploring unstructured data with the creation of learning-based embedding models. Completely managed and horizontally scalable vector databases are required as vector collections reach billion-scale numbers. The proposal relaxes the data model and consistency restrictions in exchange for the aforementioned benefits because the majority of vector data applications do not call for intricate data models and robust data consistency. VectorDB of Python has been used for implementation and test case which does faster similarity search.**

Keywords*— *Artificial Intelligence, Density estimates, Nearest Neighbor, Neural Networks, Vector databases, Vector Embeddings, Vector indices, VectorDB

I. INTRODUCTION

When an AI (Artificial Intelligence) system produces an output that may seem logical or believable but is not based on reality or factual information, it is referred to as "AI hallucination." These outputs may be text, graphics, or other data types that the AI model generated based on its training, but which might not be consistent with actual facts or logic.

A high-dimensional vector database has mathematically represented features or qualities, is known as a vector database. Each vector can have anywhere between tens and thousands of dimensions. The data, including text, and data of multimedia types, is typically transformed or embedded to produce the vectors. The functionality can be based on a variety of techniques, including feature extraction algorithms, word embeddings, and machine learning models.

Similar images can be found using a vector database. According to pixel resolution ratio of the image, the image search procedure should analyze the contents of the featured vector [20]. Using machine learning sentiment analysis, this may locate files those are comparable to a particular content based on their topic. By these characteristics and ratings, it is used to locate items that are comparable to a specific product.

A query vector representing the requested data or criteria must be used to execute to similar data in a vector database. Then, analysis can be done to determine how much they are near or far in the vector space. There are different metrics used like Cosine similarity, Hamming distance, Euclidian Distance for similarity measure.

Vector Embeddings: Vector embeddings are essentially numerical representations of data. They mostly serve as representations for unstructured data, pictures, videos, audio, text, molecular pictures, and other types of data that lack a formal organization are considered unstructured data.

It is known that Neural Networks are trained on different data sets, making each model's vector embedding unique. That's why working with unstructured data and vector embeddings is challenging. Figure 1 depicts how input, hidden and output layers are working for to yield different vector embeddings on different data sets.

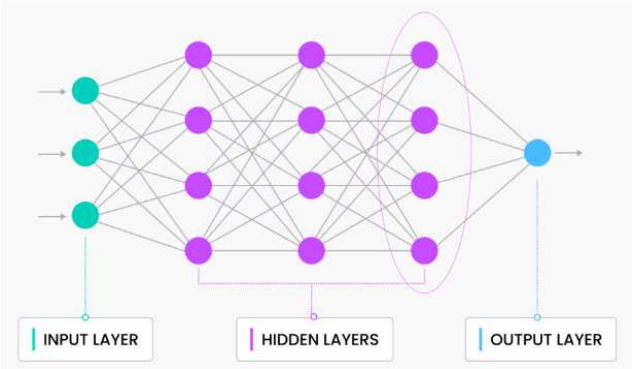

Figure 1. Vector embeddings in Neural Network

Here different available vector databases are introduced with their features:

Pinecone: A cloud-based vector database called Pinecone is made to effectively index, store, and search huge amounts of high-dimensional vectors. Real-time indexing and searching, managing sparse and dense vectors, and support for exact and approximated nearest-neighbor search are some of Pinecone's important features. Pinecone is a well-liked machine learning framework and library for creating production-grade NLP and computer vision applications since it integrates readily with other machine learning frameworks and libraries. Its features are: Detection of duplicates, Rank tracking. Data search, Classification and De-duplication.

Chroma: An open source vector database called Chroma offers a quick and scalable method for storing

2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG) | 979-8-3503-4327-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICTBIG59752.2023.10455990

and retrieving embeddings. With a straightforward API and support for numerous back-ends, such as RocksDB and Faiss (Facebook AI Similarity Search). Chroma is designed to be lightweight and simple to use). Chroma's distinctive characteristics include integrated support for quantization and compression as well as the capacity to dynamically change the database's size to accommodate shifting demands. Chroma's versatility and usability make it a popular option for study and experimentation. It is rich in features for queries, filtering, density estimates, and many others. The same API that runs in Python notebook scales to cluster for development, testing, and production.

Weaviate: An open source vector database called Weaviate is made for developing and deploying AI-powered apps. Semantic search and knowledge graph support, as well as the capacity to automatically extract entities and relationships from text data, are some of Weaviate's core features. Data exploration and visualization are also supported by Weaviate's built-in functionality. Applications requiring sophisticated semantic search or knowledge graph capability should strongly consider Weaviate. Weaviate has built-in modules for AI-powered searches, Q&A with data sets, and automated categorization in combination of Machine Learning models.

Milvus: An open source vector database called Milvus was created for extensive machine learning applications. Milvus provides accurate and approximated nearest-neighbor searches and is optimized for CPU and GPU-based platforms. A built-in RESTful API including Python and Java, are also features of Milvus. Building recommendation engines and search systems that need real-time similarity searches is a common use for Milvus. Although Zilliz is the main developer of Milvus, it is a component of the AI and Data Foundation of the Linux. Its specific features are: searching trillions of vector datasets in milliseconds, and simple management for semi-structured data sets and Lambda structure. So, it is highly scalable and adaptable.

DeepLake: A cloud-based vector database created specifically for machine learning applications is called DeepLake. Streaming data support, real-time indexing and searching, capacity to handle both dense and sparse vectors are just a few of DeepLake's distinctive features. A RESTful API and support for numerous programming languages are also provided by DeepLake. Applications requiring real-time indexing and search of massive, high-dimensional data should consider DeepLake. DeepLake has the storage for all data types including multimedia data sets and performs query and vector search well with rich tools like LangChain, LlamaIndex, Weights & Biases, and many more.

Qdrant: An open source vector database called Qdrant is made for search and real-time analytics. Built-in support for geospatial data and the capability to run geospatial queries are two of Qdrant's distinctive characteristics. Along with a RESTful API, support for numerous programming languages, and support for exact and approximate nearest-neighbor searches, Qdrant also offers these features. For applications that need real-time geographic search and analytics, Qdrant is a great option. Different data types, search criteria, and JSON payloads can be attached for filtering are supported. Configuration is made simpler by Qdrant's independence from external databases or orchestration controllers.

ElasticSearch: Textual, numerical, geospatial, structured, and unstructured data may all be handled using Elasticsearch, an open-source, distributed, and RESTful analytics engine. A variety of use cases can be handled with Elasticsearch. The data is centrally stored to enable super-fast search, precise relevance, and complex analytics that scale with ease. It scales horizontally to support trillions of events per second while automatically managing the distribution of indexes and queries across the cluster for smooth operations. Cross-clustering, high availability, automatic node recovery, data rebalancing, horizontal scalability, and has the capability to operate in a distributed architecture that was designed from the ground up to offer constant peace of mind are some of its important features.

Vespa: Vespa is a free and open-source data serving engine that enables users to store, explore, arrange, and make machine-learned decisions over vast amounts of data at serving time. Vespa is a platform that is capable for these while ensuring outstanding availability and it is faster for large data volumes that need to be distributed over several nodes and reviewed in parallel. The client receives write acknowledgements in milliseconds. Large tensors and vectors, as well as any combination of structured filters, free text search operators, and vector search operators, can be included in queries. Even if a match is playing simultaneously on multiple machines, it is still included.

Vald: A distributed, scalable, and quick dense vector search engine is called Vald. It uses the fastest ANN Algorithm NGT, which was designed with cloud-native architecture in mind, to find neighbors. Vald provides horizontal scaling, automated vector indexing, and index backup, enabling it to search through trillions of feature vector data. It's easy to use and incredibly adjustable, such as the Ingress/Egress filter, which you can tailor to work with the gRPC interface. Disaster recovery is made possible by Vald's automatic backups through Object Storage or Persistent Volume. It gives out vector indices to many agents, each of which keeps their own index. Each index is stored in many agents by the tool, which replicates indexes. Rebalance the duplicate automatically when a Vald agent fails. So, it is adaptable to choose the large vector dimensions, replicas, and so forth. Python, Golang, Java, Node.js, and more programming languages are supported.

ScaNN: Scalable Nearest Neighbours, or ScaNN, is a technique for quickly looking for vector similarity. The brand-new compression technique suggested by Google's ScaNN greatly improves accuracy. According to ann-benchmarks.com, this enables it to outperform competing vector similarity search libraries by a factor of two. Additionally, it offers other distance functions including Euclidean distance in addition to search space trimming, quantization, and Maximum Inner Product Search. The AVX2-compatible x86 processor is the target audience for the implementation.

PgVector: This is an extension of PostgreSQL, used to look for vector similarities. In the end, PgVector enables you to centrally store all application data. To using any language with the client, it does accurate and approximated

closest neighbor search and distance, inner product, and cosine distance. Its users also get to benefit from other wonderful capabilities like point-in-time recovery, JOINS, and ACID compliance.

Faiss (Facebook AI Research)**:** It is used for quick dense vector similarity search and grouping. There are techniques for exploring vector set arrays of any size. Additionally, it includes code for parameter adjusting and evaluation. Based on an index type, Faiss provides a method for searching through a set of vectors. Simple baselines are some index kinds, like precision search. One of the specific feature of the Faiss is to return up to k-th nearest neighbor is one of its unique features. Instead of a basic Euclidean search, the largest inner product search is used. The range search feature returns all located a certain distance from the query location. Instead of keeping them in memory, one can save the search indices on permanent storage.

II. LITERAURE SURVEY

Vector search algorithms have a long research history, and most works focus on efficient approximate search on large-scale datasets. Vector search algorithms are cited in the reference [3 and 17]. Existing algorithms can be roughly classified into four categories, i.e., space partitioning tree (SPT), locality sensitive hashing (LSH), and vector quantization (VQ) and proximity graph (PG). SPT algorithms divide the space into areas, and use tree structures to quickly narrow down search results to some areas.

Embeddings vectors are used to encode and decode input and output texts, and can vary in size and dimension. / Embeddings can help the model understand the relationships between tokens, and generate relevant and coherent texts. They are used for text classification, summarization, translation, and generation, as well as image and code generation. Vector embeddings are cited in the references [7, 9, 12, and 14].

The citation [1] is referred from Microsoft learning page for vector database and its terminologies. S. Singla et al. [2] have studied the raster and vector databases for concurrent processing of large scale data. Authors of [4] have used R language and tried to optimize their algorithm to improved accuracy of diabetic dataset. J. Pavon et al. [5] have presented Vector Indexed Architecture for sparse-dense computation of matrices. Researchers of [6] proposed presented Approximate Nearest Neighbor Accelerator with Google ScaNN and Facebook Faiss. By fusing the advantages of a specialized dataflow pipeline and effective data reuse, they have done good research for nearest neighbor search.

Authors of [8] advocates for raster-based Deep Learning strategies to recognize highway interchanges converting vectors to small images using CNN (Convolutional Neural Network). Neelima and S. Mehrotra [10] tried to achieve feature reduction by loosely clustered similar features using graph search. They used different Machine Learning algorithms to evaluate their proposal. Authors of [11and 13] described the word embedding methods. Researchers of [15] have represented hybrid analytical engine developed at Alibaba and proposed a novel algorithm to improved the accuracy on large scale vectors having massive unstructured data. Authors of [16] designed a novel scheme for extending the index type of PostgreSQL to achieve high performance.

Researchers of [18] have proposed a fast and accurate locality-sensitivity hashing for queries in high-dimensional datasets. To achieve accurate distance estimation they also proposed a tunable confidence interval. Dr. Nivas Jeevanandam [19] described Exclusive vector databases for Large Language Model on site of govt. of India. Rentong Guo et al in [20] have presented their Manu model for cloud native vector databases.

A query result in multiple vector representations means that that query must be semantically similar in numerous ways. Try doing this with image models, different dimensionality language models or your data for the next steps. In this presentation the difference between some existing model and the proposed fine-tuned model has been verified.

High performance: The effectiveness of vector databases' query processing is crucial. Implementations should be heavily hardware optimized for optimal performance. Additionally, the framework should be carefully planned to reduce system overheads associated with query serving.

Strong adaptability: Customers now employ vector databases in a range of settings, from small-scale cloud deployments to large-scale laptop prototyping. A vector database should lessen the burden of moving code and data between environments and offer a consistent user experience.

III. METHODOLOGY

Important and renowned search tools are available for vector databases.

Azure cognitive search: There is no specific vector database service provided by Azure. On Azure, however, external vector databases can be used by simply deploying them. Cognitive Search can integrate with other Azure services across the Azure platform in the form of indexers that automate data ingestion/retrieval from Azure data sources, skill-sets that incorporate consumable AI from Azure AI services, like image and natural language processing, or custom AI that one creates in.

DuckDB: OLAP (Online Analytical Processing), often known as analytical query workloads, are supported by DuckDB. These workloads are characterized by sophisticated, time-consuming queries, such as joins over many large tables or aggregations over full tables, which process huge sections of the stored dataset. The data is also anticipated to undergo very significant changes, such as the addition of numerous rows or the simultaneous modification or addition of sizable chunks of tables. The query execution engine in DuckDB is columnar-vectorized, meaning that queries are still interpreted but a big group of values (a "vector") are processed all at once.

Sqlite-vss: Based on Faiss, the SQLite extension sqlite-vss (SQLite Vector Similarity Search) gives SQLite the ability to perform vector searches. It can be applied to the development of recommendations, questions-and-answers tools, and semantic search engines.

Redis-VSS: One of a vector database's most important features is vector similarity search (VSS). It involves searching a vector database for data points that resemble a

specific query vector. Recommendation systems, image and video search, natural language processing, and anomaly detection are common applications for VSS. Because of its real-time search performance, integrated fault tolerance and resilience, reduced architectural and application complexity, and flexibility across clouds and locations, Redis Enterprise is recognized as the vector database solution for every organization.

Engineering the vector values using domain knowledge is one method of producing vector embeddings. Feature engineering is the term used for this. For instance, in one can use the model to quantify a variety of aspects, including form, colour, and semantically significant regions in an image. Due to its strong data manipulation capabilities and the availability of modules for working with vector data, Python is a well-liked language for working with vector databases.

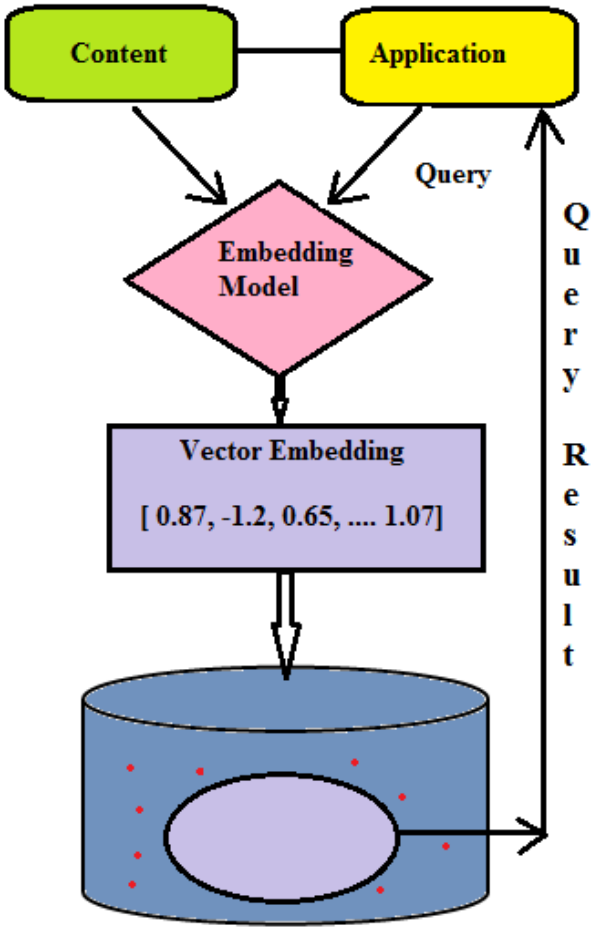

Figure 2. Vector Embedding Model

An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness, and large distances suggest low relatedness. Contents and application can fetch the query through embedding model. Query results can be sent back to application. With embeddings, a vector can be represented as [0.87, -1.2. 0.65,…,1.07] as shown in Figure 2.

Another model is also proposed here for cloud-based vector databases. In order to accomplish fine-grained decoupling between the system components, the model uses a service-oriented design. The model contains four layers, namely the access layer, coordinator layer, worker layer, and storage layer, as shown in Figure 3.

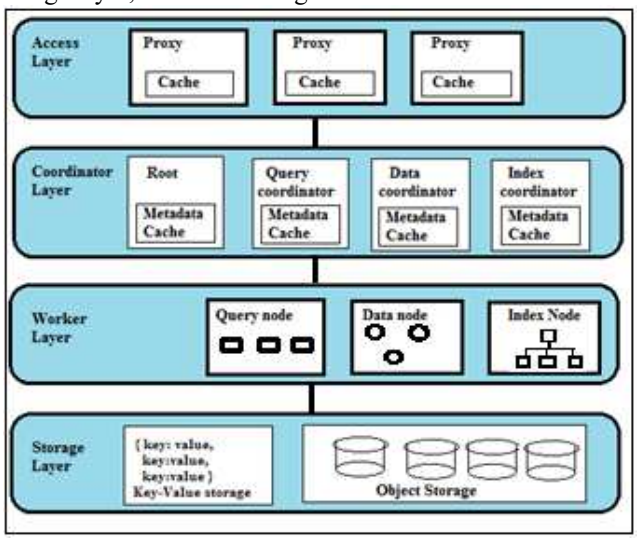

Figure 3. Cloud based vector embedding model

The user endpoints in the access layer are stateless proxies. Receiving requests from clients, distributing those requests to the appropriate processing components, and aggregating incomplete search results before returning to clients are all tasks they perform concurrently. The coordinator layer preserves collection metadata, regulates system status, and coordinates system components for processing tasks. Each of the four coordinators is in charge of a separate task. The actual computation is done at the worker layer. The worker nodes are stateless; to complete tasks, they fetch read-only copies of the contents and do not require communication with one another. This makes it possible to easily scale up worker nodes that are computationally expensive and intense. System status, metadata, the collections, and related indexes are all persistent thanks to the storage layer.

Multi-vector search, attribute filtering, and traditional vector search are all supported by the model. The distance/similarity function for traditional vector search can be angular distance, inner product, or Euclidean distance. When the distribution of segments among the query nodes changes (which may occur during scaling, load-balancing, query node failure, and recovery) query nodes consult the Binlog for data. In order to coordinate failure recovery and scaling, the query coordinator oversees segment distribution and keeps an eye on the query nodes' workload and liveness. When a query node fails, the segments it handled and any matching indexes (if any exist) are loaded onto the healthy query nodes.

When dealing with a text document, one converts all the words and sentences into embeddings. The document will then have a semantic representation as a collection of vectors. These vectors accurately depict the content and setting of each word or sentence. When words or phrases have been embedded, semantic similarity can be determined using these. Calculating how near the vectors are to one another is a typical method for determining how similar two embeddings are to one another. By computing the cosine similarity, the distance between two vectors can be determined.

Vector embeddings must be compared using vectors of the same length and dimension. It has been done using the sentence transformer model. The most recent iteration of the transformers for natural language processing using Tensorflow 2.0 and PyTorch is called BART (Bidirectional Abstractive Representation from Transformers) [19]. None of the dimensions, for instance, will be able to convey concepts like a part of speech, the amount of words in a phrase, whether a word is a proper noun, etc.

Common use cases of vector data bases are depicted in Figure 4.

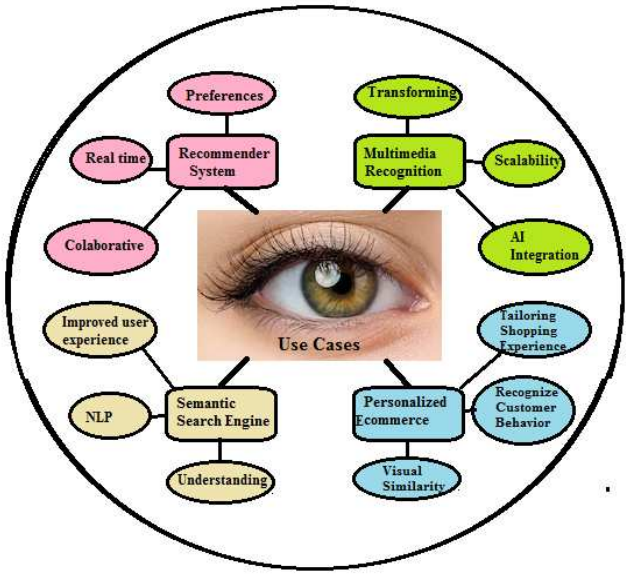

Figure 4. Common Use-cases

*1) **Recommendation Systems:***

- **Recognizing Preferences:** Vector databases offer accurate and pertinent recommendations by representing user preferences and item properties as vectors. It's the magic of vectors at work whether it's recommending a film on a streaming service or a dish in a food delivery app.
- **Real-time Recommendations:** Recommendation systems need to be quick. In this regard, vector databases shine because they enable real-time customization that users adore.
- **Collaborative Filtering:** By incorporating strategies like collaborative filtering, vector databases may provide better suggestions by taking into account user behavior and connections between objects.

*2) **Multimedia Recognition***

- **Transforming Media into Data:** Complex media can be analyzed, categorized, and identified by turning images and sounds into vectors. It's comparable to training a machine to see and hear!
- **AI Integration:** Advanced image and speech recognition capabilities are made possible by the integration of vector databases into bigger AI systems. Vector databases are essential for voice controls in smart devices and facial recognition in security.
- **Scalability:** With the expansion of media content, vector databases offer the scalability necessary to manage huge amounts of audio and image data without sacrificing performance.

*3) **Semantic Search Engines***

- **Understanding Context:** Semantic search engines work to comprehend the context and intent behind a search query in addition to matching keywords. These engines can deliver more precise and pertinent results by modeling words and sentences as vectors.
- Natural Language Processing (NLP): By using NLP strategies, vector databases help search engines comprehend human language more naturally, improving the conversational and intuitive nature of interactions.
- Improved User Experience: As a result, the search engine is smarter and more responsive, finding what you're looking for while also comprehending why you're looking for it.

*4) **Personalization in E-commerce***

- Tailored shopping experience: Imagine entering a store where everything is set up to suit your tastes and preferences. This is what is meant by "tailored shopping experiences." Vector databases are facilitating this in the world of online buying. It's all about from product recommendations to custom discounts.
- **Visual Similarity:** By expressing product photos as vectors helps to find products that resemble a particular item. It is like to having a personal stylist at your disposal.
- **Recognizing Consumer Behavior:** E-commerce platforms may design more interesting and fulfilling buying experiences using vectors by analyzing consumer behavior and preferences.

Vector databases are utilized with numerous use cases in various fields like NLP, Computer Vision, and recommendation systems for semantic comprehension and data matching [1]. These databases enable large language models to produce coherent, relevant text using Plugins of Artificial Intelligence. Complex language models often face issues like producing irrelevant or erroneous information, missing consistency, repeating or contradicting themselves, or being biased. To address these issues, a vector database can store data on subjects, keywords, truths, opinions, and sources. This information can be sent along with an AI plugin and a large language model to create useful and captivating content in line with the user's aim and style.

IV. Test Cases And Results

Text data is frequently represented in NLP as large vectors utilizing methods like word embeddings or transformer-based models. These vectors represent the semantic content of the text, and the spacing between them shows how semantically similar two pieces of text are. These text vectors can be stored in vector databases, which also offer effective similarity search capabilities. This is very helpful in applications like document retrieval. Finding the top-k vectors (i.e., database vectors) in a collection that are most similar to a query vector is known as a "similarity search." The Nearest Neighbor Search (NNS) problem is another name for it. The similarity of vectors can be defined using a variety of similarity functions [7].

Vectordb is a light weight Pythonic vector database offers a comprehensive suite of CRUD (Create, Read, Update, Delete) operations and robust scalability options. Sentence transformer models have been used where input data is in the form of 100 sentences. The three existing models are compared with the proposed fine-tuned model for same sentences.

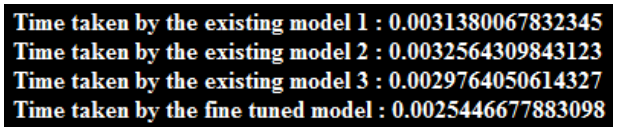

Figure 5. Comparison of time

It can be seen in Figure 5 that time for the 3 original models and the proposed fine-tuned model. The proposed model takes little less time for the similarity search.

The latest version of the Transformer package [34] has been introduced and several NLP (Natural Language Processing) tasks have been implemented successfully. Going in deep of NMT (Neural Machine Translator) Positional Encoding has been also analyzed and executed successfully for base of this paper. The paragraph of the English Language has been paraphrased successfully by a sentence having several meanings by stressing the word. The multilingual sentiment analysis for the similar sentences of English and French has been done successfully which matching results are 94% to 98%. The largest improvement has been observed by training the pre-trained models of the Transformer. Future scope is to explore new methods for question-answer and for translation of natural language to programming languages.

V. CONCLUSION

Demonstration and Implementation of vector embedding vector databases are done successfully with fine tuned model. The proposed model takes little less time also. Many AI applications, such as recommender systems, picture and video recognition, natural language processing (NLP), and anomaly detection, depend on vector databases. These databases provide accurate and efficient search and analysis of big datasets by storing and maintaining data as high-dimensional vectors. This improves automation and allows for the early discovery of abnormalities. Vector databases provide the speedy identification of items most pertinent to users' preferences in the world of recommender systems. At the same time, effective object and face recognition is made possible by image and video recognition. By storing and organizing word and sentence information as vectors, vector databases play an important part in natural language processing (NLP). They make it possible to quickly identify odd patterns or behaviors in anomaly detection.

Vector databases can help detect unusual activities or behaviors in various areas, such as cyber-security, fraud detection or industrial equipment monitoring. These databases can quickly identify patterns that deviate from the norm by representing data as vectors. AI models integrated with vector databases can then flag these anomalies and trigger alerts or mitigation measures, ensuring timely and effective responses.

REFERENCES

[1]. Evchaki, JimPaine, matthewbolanos, sneha-afk, shpathak-msft. What is a vector database? https://learn.microsoft.com/en-us/semantic-kernel/memories/vector-db 07/31/2023

[2]. S. Singla, A. Eldawy, T. Diao, A. Mukhopadhyay and E. Scudiero, "Experimental Study of Big Raster and Vector Database Systems," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 2243-2248, doi: 10.1109/ICDE51399.2021.00231.

[3]. W. Du, Z. Wang and J. Ai, "Fast Search of Massive High-Dimensional Vectors Similarity," 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Shenzhen, China, 2020, pp. 67-70, doi: 10.1109/AEMCSE50948.2020.00022.

[4]. M. Sanghar, V. K. Shukla, A. Verma and P. Sharma, "Implementation of Support Vector Machines Algorithm through R-Language for Diabetes Database Testing," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 746-751, doi: 10.1109/Confluence51648.2021.9377124.

[5]. J. Pavon et al., "VAQUERO: A Scratchpad-based Vector Accelerator for Query Processing," 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Montreal, QC, Canada, 2023, pp. 1289-1302, doi: 10.1109/HPCA56546.2023.10070958.

[6]. Y. Lee et al., "ANNA: Specialized Architecture for Approximate Nearest Neighbor Search," 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, Republic of, 2022, pp. 169-183, doi: 10.1109/HPCA53966.2022.00021.

[7]. Y. Zhao, A. Anand and G. Sharma, "Reviewer Recommendations Using Document Vector Embeddings and a Publisher Database: Implementation and Evaluation," in IEEE Access, vol. 10, pp. 21798-21811, 2022, doi: 10.1109/ACCESS.2022.3151640.

[8]. Touya, Guillaume and Lokhat, Imran. Deep Learning for Enrichment of Vector Spatial Databases: Application to Highway Interchange, 2020, Association for Computing Machinery,New York, USA, volume 6, issue 3, issn 2374-0353, https://doi.org/10.1145/3382080

[9]. V. Pandya and F. D. Troia, "Malware Detection through Contextualized Vector Embeddings," 2023 Silicon Valley Cybersecurity Conference (SVCC), San Jose, CA, USA, 2023, pp. 1-7, doi: 10.1109/SVCC56964.2023.10165170.

[10]. Neelima and S. Mehrotra, "A Comprehensive Review on Word Embedding Techniques," 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Coimbatore, India, 2023, pp. 538-543, doi: 10.1109/ICISCoIS56541.2023.10100347.

[11]. Q. Jiao and S. Zhang, "A Brief Survey of Word Embedding and Its Recent Development," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2021, pp. 1697-1701, doi: 10.1109/IAEAC50856.2021.9390956.

[12]. Agarwal, D. Uniyal, D. Toshniwal and D. Deb, "Dense Vector Embedding Based Approach to Identify Prominent Disseminators From Twitter Data Amid COVID-19 Outbreak," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 5, no. 3, pp. 308-320, June 2021, doi: 10.1109/TETCI.2021.3067661.

[13]. P. Sherki, S. Navali, R. Inturi and V. Vala, "Retaining Semantic Data in Binarized Word Embedding," 2021 IEEE 15th International Conference on Semantic Computing

(ICSC), Laguna Hills, CA, USA, 2021, pp. 130-133, doi: 10.1109/ICSC50631.2021.00031

[14]. Grohe, Martin. Word2vec, Node2vec, Graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data, 2020, isbn 9781450371087, Association for Computing Machinery, New York, USA, url https://doi.org/10.1145/3375395.3387641

[15]. Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine Towards Query Fusion for Structured and Unstructured Data. Proceedings of the VLDB Endowment (PVLDB) 13, 12 (2020), 3152–3165.

[16]. Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. PASE: PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension. In ACM Conference on Management of Data (SIGMOD). 2241–2253.

[17]. Weijie Zhao, Shulong Tan, and Ping Li. 2020. SONG: Approximate Nearest Neighbor Search on GPU. In International Conference on Data Engineering (ICDE). 1033–1044.

[18]. Bolong Zheng, Xi Zhao, LiangguiWeng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S. Jensen. 2020. PM-LSH: A Fast and Accurate LSH Framework for High-Dimensional Approximate NN Search. Proceedings of the VLDB Endowment (PVLDB) 13, 5 (2020), 643–655.

[19]. P. N. Singh and S. Behera, "The Transformers' Ability to Implement for Solving Intricacies of Language Processing," 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, 2022, pp. 1-7, doi: 10.1109/ASIANCON55314.2022.9909423.

[20]. P. N. Singh and T. P. Gowdar, "Reverse Image Search Improved by Deep Learning," 2021 IEEE Mysore Sub Section International Conference (MysuruCon), Hassan, India, 2021, pp. 596-600, doi: 10.1109/MysuruCon52639.2021.9641572.

[21]. Dr. Nivas Jeevanandam, Exclusive vector databases for LLMs. Jul 04, 2023, https://indiaai.gov.in/article/exclusive-vector-databases-for-llms

[22]. Rentong Guo et al. 2022. Manu: a cloud native vector database management system. Proc. VLDB Endow. 15, 12 (August 2022), 3548–3561. https://doi.org/10.14778/3554821.3554843