When Large Language Models Meet Vector Databases: A Survey

1st Zhi Jing*
School of Computer Science
Carnegie Mellon University
Pittsburgh, US
zjing2@alumni.cmu.edu

2nd Yongye Su*
Department of Computer Science
Purdue University
West Lafayette, US
su311@purdue.edu

3rd Yikun Han*
Department of Statistics
University of Michigan
Ann Arbor, US
yikunhan@umich.edu

***Abstract*—This survey explores the synergistic potential of Large Language Models (LLMs) and Vector Databases (VecDBs), a burgeoning but rapidly evolving research area. With the proliferation of LLMs comes a host of challenges, including hallucinations, outdated knowledge, prohibitive commercial application costs, and memory issues. VecDBs emerge as a compelling solution to these issues by offering an efficient means to store, retrieve, and manage the high-dimensional vector representations intrinsic to LLM operations. Through this nuanced review, we delineate the foundational principles of LLMs and VecDBs and critically analyze their integration's impact on enhancing LLM functionalities. This discourse extends into a discussion on the speculative future developments in this domain, aiming to catalyze further research into optimizing the confluence of LLMs and VecDBs for advanced data handling and knowledge extraction capabilities.**

***Index Terms*—Large Language Models, Retrieval-Augmented Generation, Vector Databases**

I. Introduction

The field of Natural Language Processing (NLP) achieves a great milestone when ChatGPT unlocks the power of interactive smart assistants and people are offered the unprecedentedly tangible experience of talking to a machine program that is seemingly more knowledgeable than an ordinary person. This remarkable breakthrough is largely attributed to the advancements in LLMs, such as GPT [1], [2], T5 [3], and Llama [4]. They can process, understand, and generate human-like text. Thanks to the power of pre-training, which involves training a language model on a massive corpus of text data, LLMs can capture the complexities of human languages, including context, idioms, and even cultural references.

However, despite the impressive capabilities of LLMs, they are still under the shadows of doubt in certain aspects. 1) One major shortcoming is the problem of **hallucination**, where LLMs generate plausible but factually incorrect or faithfully nonsensical information [5]. There are many potential causes behind this. First, LLMs lack domain knowledge. The fact that LLMs are primarily trained on public datasets [6] inevitably leads to a limited ability to answer domain-specific questions that are out of the scope of their internal knowledge. Moreover, real-time knowledge updates are challenging for LLMs. Even if the questions are within the learning corpus of LLMs, their answers may still exhibit limitations because the internal knowledge may be outdated when the outside world is dynamic and keeps changing [7]. Last but not least, LLMs are found to be biased. The datasets used to train LLMs are large, which may introduce systematic errors [8]. Essentially, every dataset can be questioned with bias issues, including imitative falsehoods [9], duplicating biases [10], and social biases [11]. 2) Another disadvantage of incorporating LLMs commercially is the **expensive cost** [12]. For an average business entity, applying LLMs for business use is barely feasible. It is almost impossible for a non-tech company to customize and train a GPT model of its own because they don't have the resources and talents to conduct such a big project [13], while frequent API calls to third-party LLM providers like OpenAI can be extremely expensive, not to mention there are a very limited number of such providers in certain areas. 3) The **oblivion** problem of LLMs has been controversial because LLMs are found to tend to forget information from previous inputs. It is studied that LLMs also exhibit the behavior of catastrophic forgetting [14] as neural networks do [15].

While generative models like LLMs use vector data embeddings to represent unstructured data in our real world, users need a reliable data system to manage and retrieve a considerable amount of data with cost-efficiency. Yet another increasingly popular field that seems orthogonal to LLMs is purpose-built databases for AI that support vector data storage and its efficient retrieval at scale, also known as VecDB. With efficient combinations of VecDBs and LLMs, VecDBs can either be incorporated as an external knowledge base with efficient retrieval providing domain-specific knowledge for LLMs, a memory for LLMs saving previous related chat contents for each user's dialog tab, or a semantic cache, the aforementioned problems of LLMs can be seamlessly solved.

Since there are lacking papers that introduce LLMs in the view of VecDB, this survey aims to picture how VecDBs can be potential solutions to refine LLMs' known shortcomings in previous works and hopes to offer a unique perspective of future directions in the intersection that is fertile of research opportunities.

TABLE I
COMPARISON OF MAINSTREAM ALL-LEVEL DATABASES SUPPORTING VECTOR DATA (THE VERSION NUMBER REPRESENTS THE FIRST VERSION THAT SUPPORTS VECTOR SEARCH AND ITS RELEASE DATE).
ABBREVIATIONS NOTE: *Ftx.*, *Rel.*, *Vec.* STAND FOR FULL-TEXT SEARCH, RELATIONAL DATABASE, AND VECTOR SEARCH.

Name	Supported Data		Supported Query		Vector Index	
(Version with year)	Vec. Dim.	Type	Filter	Multi-Vec.	Graph	IVF
ChromaDB (2022)	1536	Vec.	✓		✓	
Manu (2022)	32768	Vec.+ Ftx.	✓	✓	✓	✓
Milvus (2021)	32768	Vec.+ Ftx.	✓	✓	✓	✓
Pinecone (2019)	20000	Vec.	✓	✓	✓	✓
Weaviate (2019)	65535	Vec.+ Ftx.	✓	✓	✓	
Qdrant (2021)	4096	Vec.+ Ftx.	✓	✓	✓	
Amazon OpenSearch (v2.9, 2023)	16,000	Ftx.	✓	✓	✓	✓
Elastic Search (v8.0, 2022)	4096	Ftx.	✓	✓	✓	✓
AnalyticDB-V (2020)	$\geq$512	Rel.+Ftx.	✓		✓	✓
PostgreSQL-pgvector (2021)	2000	Rel.+ Ftx.	✓	✓	✓	✓
MongoDB Atlas (v6.0, 2023)	2048	NoSQL+ Ftx.	✓		✓	
MyScale (2023)	1536	Rel.	✓		✓	

II. BACKGROUND

A. Large Language Models

Over the last half-decade, we have witnessed the groundbreaking success of LLMs, marking a significant milestone in the field of NLP. LLMs have revolutionized our views on the approaches to understanding and generating human languages by machines.

1) Development of Language Models: With the advent of neural networks, the field of NLP has undergone a transformative shift, which began with the introduction of Recurrent Neural Networks (RNNs) [16]. RNNs provide a way to process sequences of words and capture temporal dependencies in text. And 2 of its famous variants, Gated Recurrent Units (GRUs) [17] and Long Short-Term Memory networks (LSTMs) [18] address the limitations of RNNs in handling problems of vanishing and exploding gradients.

The next pivotal milestone for language models is the widespread adoption of the Transformer architecture [19] which has set a new standard for language models. The multi-head self-attention modules and cross-attention modules in the encoders and decoders enable the model to capture long-range dependencies, parallel processing, and contextual understanding. Furthermore, the model starts to rapidly grow in size in the Transformer era. More importantly, the Transformer represents a paradigm shift in the NLP field, and the success of the Transformer architecture becomes the significant backbone of LLMs.

2) Are We There Yet? Challenges Faced by Pure LLMs: Although LLMs have achieved remarkable success, they face significant challenges that cannot be overlooked. Addressing these challenges is crucial for the continued advancement and practical application of these models.

One major challenge is **hallucinations** in LLMs [20], which undermine the reliability of their outputs. LLMs often generate plausible yet factually incorrect or nonsensical information [5]. Several factors contribute to this issue. First, LLMs lack domain-specific knowledge, primarily due to their training on public datasets [6], limiting their ability to answer domain-specific questions. Additionally, updating LLMs with real-time knowledge is a challenge, leading to outdated internal knowledge in a rapidly changing world [7].

Another notable concern is the **substantial computational resources** required for training and operating LLMs, which raises environmental and accessibility concerns [21] and underscores the importance of sustainable AI practices.

Transitioning to VecDBs could potentially address these challenges. VecDBs, with their efficient data structuring and retrieval capabilities, offer a promising solution for enhancing model editing flexibility, reducing hallucinations by improving domain-specific responses, and optimizing computational resource utilization, thereby aligning more closely with the principles of sustainable AI.

B. Vector Databases, the V-factor

1) A distinct database optimized for vector data: While LLMs like ChatGPT are relatively new, database management systems (DBMS) have been thoroughly developed and applied in many aspects for the last 60 years. DBMS are well-recognized for their consistent stability and universality for structured data with fixed formats that excel well with computer hierarchical computing architecture and storage. However, the development and wide application of deep learning models such as convolutional neural networks [22] and transformers [23], enable the embedding of unstructured multi-modal data like images and text mapped into corresponding fixed-length vector representations, which include the **high-dimensional semantic features** of original data, and the semantic similarities are naturally represented by distances between vectors, requiring a new type of DBMS that is specifically designed for handling vector data operations, especially search and storage.

There are many kinds of databases, whereas VecDBs are the only category of databases that naturally support diverse

unstructured data with efficient storage, indexing, and retrieval. As a result of various blooming applications on the cloud, various data sources exist in many different formats and diverse places. Unlike traditional databases that require structured or semi-structured data that must convey a few restrictions and formats, VecDBs are purpose-designed for storing the deep learning embedding of various unstructured data that has emerged in real-world applications. On the other hand, distinct from traditional DBMSs that search for exact values within databases, the semantic feature of vectors in VecDBs needs an approximate search for vectors, which searches approximate top-k nearest-distance neighbors within the high-dimensional base vector data space that do not necessarily require exact matches, representing top-k semantically close data.

2) Efficient vector retrieval within VDBMS: With the unstructured base data embedded into vectors with high dimensionality, calculating the k-nearest neighbors of a given query vector data can be expensive since it requires distance computation to every point in the dataset and maintaining the top-k results. Such an operation would result in a time complexity of $O(dN+N\log k)$, where d is the dimensionality and N is the number of vectors, for exhaustively searching top-k results using pair-wise distance calculation and a heap to keep top-k results.

Since brute-force search is time-consuming and computationally expensive, this calls for a more efficient search and storage technique with satisfying accuracy. Vector indexing can solve both of these problems, which optimized for ANN search within VDBMS include tree-based [24], [25], hash-based [26], and graph-based methods [27]. At the same time, the ANN Benchmarks [1] has also showcased the great performance gap between brute-force search and index-enhanced ANN search, as well as those indexing techniques combining different vector quantization methods such as Product Quantization (PQ) [28]. All these operations could be efficiently done based on the vector indexes, which are optimally designed collections of vectors deployed together for approximate nearest neighbor (ANN) search. The ANN Benchmarks have also showcased the great performance gap between brute-force search and index-enhanced ANN search. Also, according to them, among all these indexes, the graph-based Hierarchical Navigable Small Worlds (HNSW) provides state-of-the-art performance and capability with great universality and is widely used within most VecDB management systems, including the examples we mentioned in Table I.

3) A feasible solution of knowledge base for LLMs: Traditional full-text and keyword search engines, such as Elastic Search and Amazon OpenSearch, are based on term frequency metrics like BM25, which has proved practical for many search applications. However, such search techniques require significant investment in time and expertise to tune them to account for the meaning or relevance of the terms searched, you can't search it like you mean it. On the other hand, they lack multi-modal storage and retrieval support, as VecDBs do.

Nevertheless, VecDBs are capable of solving the aforementioned problem by simply transforming data into vectors and leveraging their retrieval efficiency; thus, the emergence of VecDBs (as shown in Table I) has greatly influenced machine learning systems and their multi-modal applications. An intuitive application using vector search is searching images with an image on search engines built on VecDBs [29] like Google Image Search [2], which uses one input image to find similar images on the internet. While VecDBs boast their unique search capability and efficiency for unstructured data, they are just a back-end factor for manifold applications. To maximize their abilities, this leads us to an interesting question: since LLMs use embeddings to represent text as vectors, can developers combine VecDBs and LLMs to overcome the aforementioned challenges that are inherited in LLM applications? The answer is yes.

As a kind of database encapsulated with vector search in vector data that represents real-world information within high dimensionalities, VecDBs are well-capable for retrieval applications [30] incorporating LLMs because LLM applications are generally read-intensive, not requiring many write-related changes, especially data deletes [31]. On the other hand, VecDBs can efficiently manage and warehouse vector data required and generated by LLMs, thus providing a solid data cornerstone for both LLMs and their applications. While LLMs are often limited by domain knowledge that cannot be uploaded or distributed due to security and privacy concerns, domain knowledge is often unstructured data that can easily be embedded into VecDBs for efficient local retrieval and further integration with generative AIs. Moreover, the computing and storage resources of VecDBs are way cheaper than LLMs, since they do require costly GPUs and TPUs, thus achieving a cost-effective way of fast retrieval and durable (non-volatile) storage.

III. Synergizing LLMs with VecDBs: Enhancements and Innovations

With recent rapid development in both areas, applications that utilize the combination of the two are also growing rapidly, and the most fruitful one is Retrieval-Augmented Generation (RAG), where VecDBs play a crucial role in enabling cost-effective data storage and efficient retrieval.

A. VecDB as an External Knowledge Base: Retrieval-Augmented Generation (RAG)

1) Development of RAG Paradigm: As the release of ChatGPT casts a spotlight on LLMs to the public world, there is an increasing need to use AI chatbots as query and retrieval agents. But simply loading users' private data as input to LLMs is found to be incompetent in real-world applications. LLMs have been constrained by their limited token counts and the high costs associated with training and fine-tuning

[1]https://ann-benchmarks.com/

[2]https://cloud.google.com/vertex-ai/docs/vector-search/overview

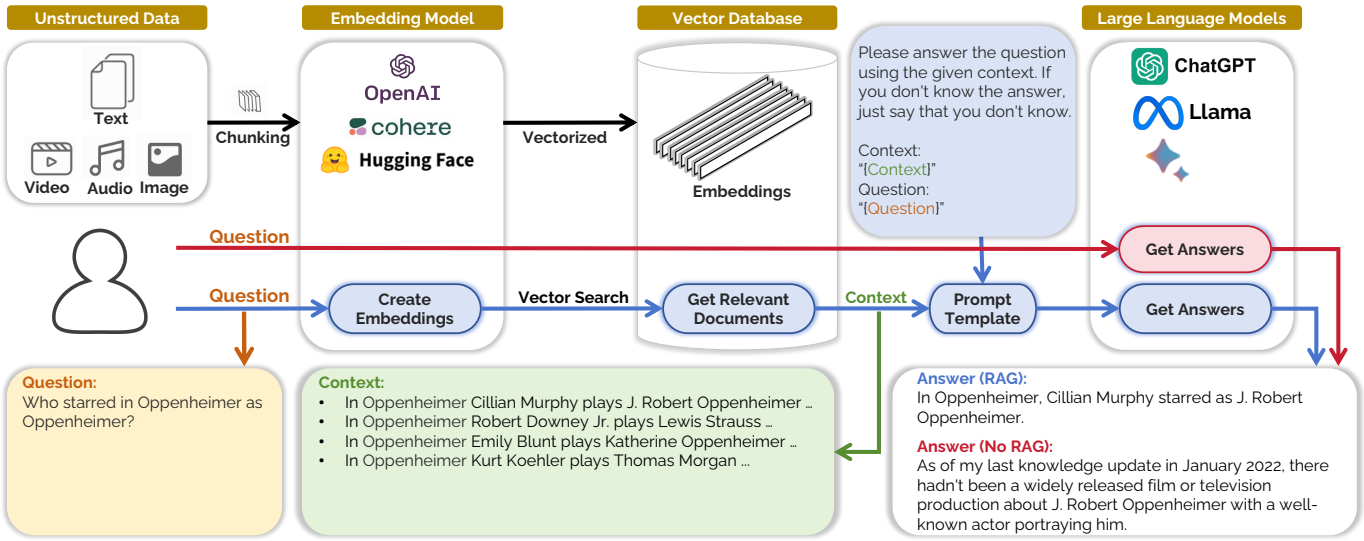

Fig. 1. A sample RAG framework that uses VecDBs.

for every alteration of data, especially when dealing with personalized or business-specific responses that require real-time data updates. To address such concerns and needs, retrieval-augmented generation (RAG) emerges as a novel solution that addresses the challenges faced by LLMs in integrating and processing large and dynamic data in external databases, where VecDBs offer a solution by acting as external memory for LLMs. They allow for the segmentation of private data by converting them into vectors and storing these vectors efficiently for a quick retrieval process. Integrating with VecDBs enables LLMs to access and synthesize enormous amounts of data without the need for constant re-training, thereby overcoming their inherent limitations.

The concept of RAG is devised to be a paradigm, and a common workflow of RAG is illustrated in Figure 1. There are essentially three main parts to a full run of the system: data storage, retrieval, and generation.

Data storage means establishing reliable external memory like VecDBs, and there are detailed recipes for this process [32]. It starts with data preprocessing, during which the original data is collected, cleaned, integrated, transformed, normalized, and standardized. The processed data is chunked into smaller segments because of the contextual limitations of LLMs. These segments of data are then converted by an embedding model into vectors, the semantic representations of the data, which are stored in VecDBs and will be used for vector search in later steps. A well-developed VecDB will properly index the data and optimize the retrieval process. The retrieval part starts with a user asking a question in the form of prompts to the same embedding model, which has generated the vector representation of the stored data and gained the vector embeddings of the question. The next step of the process is vector searching and scoring inside the VecDBs, which essentially involves computing similarity scores among vectors, and the database then identifies and retrieves the data segments that have the highest similarity scores (top K in most RAG systems) compared to the query vector. These retrieved segments are then converted back from their vector format to their original form, which is typically the text of documents. In the generation part, LLMs are involved in generating the final answers. The retrieved documents, along with the user's question, are incorporated into a specifically chosen and designed prompt template. This template selection is based on the task type and aims to effectively merge the context with the question to form a coherent prompt. The selected LLM is provided with the prompt and generates the final answer.

2) RAG with VecDBs: In the prototype of the RAG paradigm, we use such an idea to overcome the hallucination problems of LLMs by providing domain-specific information with accurate instructions like Chain-of-Thought [33], where VecDBs serve as an external knowledge base to warehouse domain-specific related information, then LLMs can easily reason with question-related information from massive data owned by users.

Even though LLMs such as ChatGPT now have user-specified GPTs for specific uses, with just prompt engineering, their knowledge bases are still limited to the training data provided by OpenAI. However, by using VecDBs, users can pre-filter whatever they want it to look at, whereas that is difficult with prompt-engineering GPT assistants.

Many research and industry examples justified RAG's profound effect. Azure AI Search (formerly Microsoft Cognitive Search) developed its vector search using Qdrant [3]. Another example is Pinecone's Canopy [4], an open-source framework that leverages Pinecone's VecDB to build and host production-ready RAG systems. Companies like Spotify and Yahoo have adopted Vespa, an AI-driven online VecDB, and Yahoo uses

[3]https://learn.microsoft.com/en-us/azure/search/vector-search-overview
[4]https://www.pinecone.io/blog/canopy-rag-framework/

it to help users directly chat with their mailbox, ask questions about their mailbox, and tell it to do things for them. This uses personalized data integrated with LLMs to form a RAG system [5]. These examples demonstrate the strength of RAG combined with VecDBs to address the unique challenges faced by business enterprises in managing and extracting value from diverse datasets but also showcase how the aforementioned difficulties faced by LLMs can be solved via integration with VecDBs.

B. VecDB as a Cost-effective Semantic Cache

The combination of VecDBs and LLM not only facilitates the profound application of LLM with RAG but also provides a new frontier for cost-effective end-to-end LLM applications [34].

a) Outrageous API Costs: LLM-based chatbots and agent systems heavily rely on LLM's output from API vendors; repeated or similar inquiries may lead to outrageous API costs.

b) API Bandwidth Limitations: Such chatbots and agent systems could also experience a bursty inquiry workload that may drown the system's bandwidth with explosive API calls coming within seconds, leading to the system's outage and reconfiguration.

One of the primary benefits of integrating VecDBs with LLMs is the significant reduction in data operational costs [35]. GPT-Cache [36], a VecDB that serves as a semantic cache, as shown in Figure 2, for instance, stores responses to previously asked queries and works as a cache before calling LLM APIs. This caching mechanism means that the system doesn't need to have API calls to wait for generated responses from scratch every time, reducing the number of costly API calls to the LLM. Moreover, this approach also speeds up the response time, enhancing the user experience.

VecDBs enhance the LLMs' ability to retrieve and utilize relevant information by indexing vast amounts of previous Q&A data and mapping them into a vector space. Instead of caches in computer systems that require exact hash-match, this allows more precise semantic matching of queries with existing knowledge and results in responses that are not only generated based on the LLM's training data but also informed by the most relevant and recent information available in the VecDBs.

C. VecDB as a Reliable Memory of GPTs

a) Oblivion: When using LLM Q&A applications like ChatGPT, LLMs are likely to completely forget the content and information of your previous conversation, even within the same chat tab.

To solve this problem, applications generally use VecDBs that serve as a robust memory layer [37] that addresses one of the intrinsic limitations of LLMs: the static nature of their knowledge. While LLMs excel in generating human-like text based on patterns learned during training, they cannot inherently update their knowledge base dynamically and, thus, may lack the few-shot learning ability. VecDBs bridge this gap by offering a storage solution that can be continually updated with new information, ensuring that the LLM's responses are informed by the most current and relevant data available.

[5]https://vespa.ai/

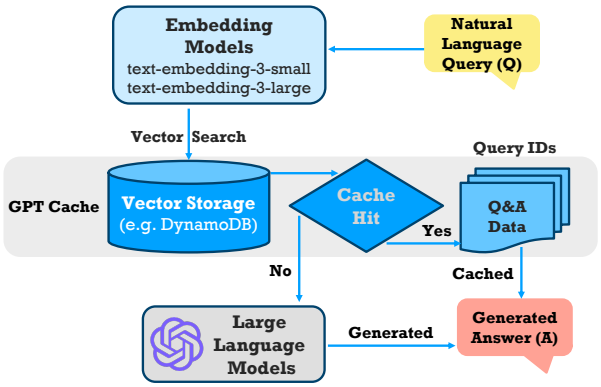

Fig. 2. An overview of semantic cache for GPTs that utilizes VecDBs.

The VecDB and LLM combination brings forth a synergy where the LLM provides context and understanding for user queries, while the VecDB offers a precise mechanism for storing and retrieving the relevant vectors. This collaborative approach allows for more accurate, relevant, and efficient responses to complex queries, which would be challenging for either system to address independently.

A VecDB integrated with an LLM facilitates real-time learning and adaptation. As new data is ingested into the VecDB, the LLM can immediately leverage this updated repository to refine its responses. This capability is pivotal for applications requiring up-to-the-minute accuracy, such as financial analysis, news dissemination, and personalized recommendations.

IV. Expanding Horizons: RAG Advancements

A. Multimodality of RAG

RAG has now evolved to handle a wide range of data types by lending the power of multimodal models. The impressive achievements of LLMs have inspired significant advancements in vision-language research. DALL-E from OpenAI introduced a Transformer-based approach for converting text to images, treating images as sequences of discrete tokens. Subsequent improvements in the text-to-image area [38] have been achieved through methods like model scaling, pre-training, and enhanced image quantization models. BLIP-2 [39] uses static image encoders with LLMs for efficient visual language pre-training, facilitating direct image-to-text transformations. Flamingo [40] presented a visual language model for text generation, showcasing remarkable adaptability and leading performance across various vision-and-language tasks. CM3 [41] trained a randomly masked model on a large HTML corpus and showed that the model is capable of generating images and text. FROMAGe [42] gains robust multimodal capabilities for few-shot learning solely from image-caption pairs, unlike other models that necessitate large-scale, interwoven image-text data from the websites.

To import speech data to RAG systems, Wav2Seq [43] allows for efficient pre-training without the need for transcriptions, using techniques like k-means clustering and byte-pair encoding to generate pseudo-subwords from speech. The Large Language and Speech Model (LLaSM) [44] is an end-to-end trained, large multi-modal speech-language model equipped with cross-modal conversational skills and proficient in understanding and responding to combined speech-and-language directives. Videos are also made available for certain types of RAG systems. Vid2Seq [45] enhances language models with specific temporal indicators for predicting event limits and textual descriptions in a single output sequence.

B. Retrieval Optimizations of RAG

To better harness the knowledge from various types of data, kNN-LMs [46] explore how incorporating nearest neighbor search into language models can enhance their ability to generalize by effectively leveraging memorized examples. EASE [47] is distinctive in its use of entities as a strong indicator of text semantics, providing rich training signals for sentence embedding. In-context RALM [48] proves that with the language model architecture unchanged, simply appending grounding documents to the input will improve the performance. SURGE [49] enhances dialogue generation by incorporating context-relevant sub-graphs from a knowledge graph. Another work that combines knowledge graphs and LLMs is RET-LLM [50], which is designed to equip LLMs with a general write-read memory unit. These studies have focused on retrieval granularity and data structuring levels, with coarse granularity providing more, but less precise, information. Conversely, structured text retrieval offers detailed information at the cost of efficiency.

To utilize both internal knowledge and external resources, SKR [51] improves LLMs' ability by enabling them to assess what they know and identify when to seek external information to answer questions more effectively. Selfmem [52] enhances retrieval-augmented text generation by creating an unbounded memory pool using the model's output and selecting the best output as memory for subsequent rounds. FLARE [53] uses predictions of upcoming sentences to anticipate future content and retrieve relevant documents for regenerating sentences, especially when they contain low-confidence tokens. Atlas [54] demonstrates impressive performance in tasks like question-answering and fact-checking, outperforming much larger models despite having fewer parameters. Many other works like these also aim to make the RAG system more efficient and competent.

V. Discussion: Challenges and Future Work

1) Are vector searches all you need?: Although vector data is an efficient representation of unstructured data, vector-based search methodologies are still not well suited for operations such as structured queries (SQL), post-query filtering, comprehensive full-text searches, relation-based graph, and keyword search mechanisms that are fundamental to conventional database systems and search engines. In the context of diverse applications for different users with multiple needs, vector search, or any single search approach, is not a one-size-fits-all solution. For example, for searching exact transaction data, users may have queries of very specific keywords to look up in a huge pool of data; giving them semantic approximate results totally would not make sense, resulting in a one-step forward yet two-step back. On the other hand, traditional DBMSs are well-developed for efficiently handling structured data within their transactional and analytical applications, and knowledge graph can represent the interation and logic between events and entities [55], in both cases, a dedicated VecDB is not yet ready for those queries.

This distinction underscores a potential gap in the functional alignment of VecDBs with previously well-established data retrieval paradigms, necessitating the development of hybrid search algorithms that can seamlessly integrate vector search with traditional relational data engine capabilities.

2) VDBMS with multi-modality: In real-world data representation and search, we may look for multi-modal data entries that are not restricted to a specific kind of structured, semi-structured, or unstructured data, but with any random combination of them, e.g., users can look for a knowledge graph from an interacted prompt of image and description. While searches using uni-modal data may lack the prospect of providing informative and more contextually appropriate search results, multi-modal data search and its hybrid processing also present a non-trivial challenge for VecDBs' storage and retrieval [56], since integrating and merging multi-modal data require efficient preprocessing with multi-modal encoders, multi-modal storage indexes, but also ranking weight assignment and multi-modal data fusion.

VI. Conclusion

In this paper, we present a systematic review of recent advances in combinations of LLMs and VecDBs. We summarize the rationales for viewing LLMs with VecDBs and also introduce certain applications that combine LLMs and VecDBs with distinct prototypes that categorize existing works from various perspectives and interdisciplinary studies. Our study also demonstrates both the research and engineering challenges in this fast-growing field and suggests directions for future work.

References

[1] T. Brown, B. Mann *et al.*, "Language models are few-shot learners," in *NeurIPS*, 2020, pp. 1877–1901.

[2] J. Achiam, S. Adler *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] C. Raffel, N. Shazeer *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, vol. 21, no. 1, pp. 5485–5551, 2020.

[4] H. Touvron, T. Lavril *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] L. Huang, W. Yu *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint arXiv:2311.05232*, 2023.

[6] G. Penedo, Q. Malartic *et al.*, "The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only," *arXiv preprint arXiv:2306.01116*, 2023.

[7] Y. Onoe, M. Zhang *et al.*, "Entity cloze by date: What LMs know about unseen entities," in *NAACL Findings*, 2022, pp. 693–702. [Online]. Available: https://aclanthology.org/2022.findings-naacl.52

[8] E. M. Bender, T. Gebru *et al.*, "On the dangers of stochastic parrots: Can language models be too big?" in *FAccT*, 2021, p. 610–623. [Online]. Available: https://doi.org/10.1145/3442188.3445922

[9] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," in *ACL*, S. Muresan, P. Nakov *et al.*, Eds., 2022, pp. 3214–3252.

[10] K. Lee, D. Ippolito *et al.*, "Deduplicating training data makes language models better," in *ACL*, 2022, pp. 8424–8445. [Online]. Available: https://aclanthology.org/2022.acl-long.577

[11] P. Narayanan Venkit, S. Gautam *et al.*, "Nationality bias in text generation," in *EACL*, 2023, pp. 116–122. [Online]. Available: https://aclanthology.org/2023.eacl-main.9

[12] R. Schwartz, J. Dodge *et al.*, "Green ai," *Commun. ACM*, vol. 63, no. 12, p. 54–63, 2020. [Online]. Available: https://doi.org/10.1145/3381831

[13] M. Musser, "A cost analysis of generative language models and influence operations," *arXiv preprint arXiv:2308.03740*, 2023.

[14] Y. Luo, Z. Yang *et al.*, "An empirical study of catastrophic forgetting in large language models during continual fine-tuning," *arXiv preprint arXiv:2308.08747*, 2023.

[15] R. Kemker, M. McClure *et al.*, "Measuring catastrophic forgetting in neural networks," *arXiv preprint arXiv:1708.0207*, 2017.

[16] W. Zaremba, I. Sutskever *et al.*, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[17] J. Chung, C. Gulcehre *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[18] X. Shi, Z. Chen *et al.*, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NeurIPS*, 2015.

[19] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[20] Z. Ji, N. Lee *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[21] E. Strubell, A. Ganesh *et al.*, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019.

[22] K. He, X. Zhang *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[23] J. Devlin, M.-W. Chang *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[24] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *VISAPP*, 2009, pp. 331–340.

[25] Y. Tao, K. Yi *et al.*, "Quality and efficiency in high dimensional nearest neighbor search," in *SIGMOD*, 2009, p. 563–576. [Online]. Available: https://doi.org/10.1145/1559845.1559905

[26] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Comm. ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[27] Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," *PAMI*, vol. 42, no. 4, pp. 824–836, 2018.

[28] H. Jégou, M. Douze *et al.*, "Product Quantization for Nearest Neighbor Search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[29] J. Wang, X. Yi *et al.*, "Milvus: A purpose-built vector data management system," in *SIGMOD*, 2021, pp. 2614–2627.

[30] A. Asai, S. Min *et al.*, "Retrieval-based language models and applications," in *ACL*, 2023, pp. 41–46.

[31] J. J. Pan, J. Wang *et al.*, "Survey of vector database management systems," *arXiv preprint arXiv:2310.14021*, 2023.

[32] Y. Han, C. Liu *et al.*, "A comprehensive survey on vector database: Storage and retrieval technique, challenge," *arXiv preprint arXiv:2310.11703*, 2023.

[33] J. Wei, Y. Tay *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[34] Y. Su, Y. Sun, M. Zhang, and J. Wang, "Vexless: A serverless vector data management system using cloud functions," *Proc. ACM Manag. Data*, vol. 2, no. 3, May 2024. [Online]. Available: https://doi.org/10.1145/3654990

[35] V. Sanca and A. Ailamaki, "E-scan: Consuming contextual data with model plugins," in *VLDB Workshop*, 2023.

[36] F. Bang, "Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings," in *NLP-OSS Workshop*, 2023, pp. 212–218.

[37] Y. Zhang, Z. Yu *et al.*, "Long-term memory for large language models through topic-based vector database," in *IALP*, 2023, pp. 258–264.

[38] C. Zhang, C. Zhang *et al.*, "Text-to-image diffusion models in generative ai: A survey," *arXiv preprint arXiv:2303.07909*, 2023.

[39] J. L. Li, D. Li *et al.*, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[40] J. Alayrac, J. Donahue *et al.*, "Flamingo: a visual language model for few-shot learning," *arXiv preprint arXiv:2204.14198*, 2022.

[41] A. Aghajanyan, B. Huang *et al.*, "Cm3: A causal masked multimodal model of the internet," *arXiv preprint arXiv:2201.07520*, 2022.

[42] J. Y. Koh, R. Salakhutdinov *et al.*, "Grounding language models to images for multimodal inputs and outputs," in *ICML*, 2023.

[43] F. Wu, K. Kim *et al.*, "Wav2seq: Pre-training speech-to-text encoder-decoder models using pseudo languages," *arXiv preprint arXiv:2205.01086*, 2022.

[44] Y. Shu, S. Dong *et al.*, "Llasm: Large language and speech model," *arXiv preprint arXiv:2308.15930*, 2023.

[45] A. Yang, A. Nagrani *et al.*, "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning," *arXiv preprint arXiv:2302.14115*, 2023.

[46] U. Khandelwal, L. Omer *et al.*, "Generalization through memorization: Nearest neighbor language models," 2020.

[47] S. Nishikawa, R. Ri *et al.*, "EASE: Entity-aware contrastive learning of sentence embedding," in *NAACL*, 2022, pp. 3870–3885. [Online]. Available: https://aclanthology.org/2022.naacl-main.284

[48] O. Ram, Y. Levine *et al.*, "In-context retrieval-augmented language models," *TACL*, vol. 11, pp. 1316–1331, 2023. [Online]. Available: https://aclanthology.org/2023.tacl-1.75

[49] M. Kang and J. M. Kwak, "Knowledge graph-augmented language models for knowledge-grounded dialogue generation," *arXiv preprint arXiv:2305.18846*, 2023.

[50] A. Modarressi, A. Imani *et al.*, "Ret-llm: Towards a general read-write memory for large language models," *arXiv preprint arXiv:2305.14322*, 2023.

[51] W. Yu, D. Iter *et al.*, "Generate rather than retrieve: Large language models are strong context generators," *arXiv preprint arXiv:2209.10063*, 2023.

[52] C. Xin, L. Di *et al.*, "Lift yourself up: Retrieval-augmented text generation with self memory," *arXiv preprint arXiv:2305.02437*, 2023.

[53] Z. Jiang, F. F. Xu *et al.*, "Active retrieval augmented generation," *arXiv preprint arXiv:2305.06983*, 2023.

[54] G. Izacard, L. Patrick *et al.*, "Atlas: Few-shot learning with retrieval augmented language models," *arXiv preprint arXiv:2208.03299*, 2022.

[55] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[56] M. Wang and X. K. et al., "Must: An effective and scalable framework for multimodal search of target modality," *arXiv preprint arXiv:2312.06397*, 2023.