# Accelerating Contextualization in AI Large Language Models Using Vector Databases

Raad Bin Tareaf, Mohammed AbuJarour, Tom Engelman, Philipp Liermann, Jesse Klotz
XU Exponential University of Applied Sciences
August-Bebel-Str. 26-53, 14482 Potsdam, Germany
Email: {r.bintareaf, m.abujarour, t.engelman, p.liermann, j.klotz}@xu-university.de

***Abstract*—In this study, we focus on evaluating the performance of machine learning models, specifically, language models, in interpreting and understanding a subset of Berlin Parliament textual documents. With the ultimate goal of identifying the most effective and reliable model, we conduct comparative analyses involving OpenAI's gpt-3.5-turbo and our in-house model. The evaluation framework incorporates Vicuna-13B, a refined model developed through fine-tuning and specifically designed for complex performance assessment of chatbots. Despite limitations tied to the scope of the dataset and hardware constraints, our results indicate that both models exhibit consistent performance metrics, suggesting their practical applicability in real-world settings. Moreover, the study emphasizes the potential of incorporating vector databases for contextual data retrieval as a future avenue for enhancing the efficiency and understanding of language models. The paper concludes by proposing specific optimization pathways for future work, particularly in terms of software parameter tuning and model fine-tuning, to elevate the capabilities of existing models.**

**Keywords: Machine Learning, Language Models, OpenAI GPT-3.5-Turbo, Vector Databases, Text Parsing, Vicuna-13B.**

## I. Introduction

In recent years, Generative Artificial Intelligence models, particularly Large Language Models (LLMs), have made significant strides in capabilities [1]. These state-of-the-art models are generally trained on voluminous collections of textual and code-based data [2]. Their broad range of applications includes natural language generation, machine translation, creative content authoring, and provision of knowledge-based responses [3]–[7].

Nevertheless, these models present challenges in scalability and efficiency [8]. They are computationally expensive to train and have an insatiable appetite for extensive data sets. Consequently, adding custom data to further contextualize or specialize these models often becomes a daunting task, limiting their applicability in specialized domains.

Vector databases, designed for the efficient storage and querying of high-dimensional vector data, offer a compelling solution to this limitation [9]. These databases can effectively manage the embedding vectors, which encapsulate semantic and syntactic information of textual data [10]. Such an approach streamlines the process of finding 'nearest neighbors' for a given vector— a pivotal step in data embedding.

In this study, we introduce an approach to augment existing LLMs with custom data sets, leveraging the efficiency of vector databases. Our empirical evaluations demonstrate that our approach not only expedites the embedding process but also enhances the contextual relevance of the LLM across multiple tasks without the need for retraining the original model.

Section II provides an overview of the historical developments in the field of large language models and vector databases. In Section III, we discuss the data used for contextualizing a pre-trained model, along with the hardware configurations and architecture of our pipeline. Section IV delves into the experimental results, comparing the performances of various models. Finally, Section V concludes the study, offering directions for future research and suggesting potential areas for expansion.

## II. Literature Review

Large Language Models (LLMs) represent a specialized subset of Artificial Intelligence (AI) technologies, trained on extensive corpora of text and programming code [11] [12]. These advanced models have the capability to produce text that closely mimics human language, perform language translation, generate a range of creative content, and provide informative responses to queries. Demonstrating proficiency in numerous tasks that were once considered solely within human expertise, LLMs are undergoing rapid advancements in sophistication [8] [13].

The roots of LLMs can be traced to the formative years of AI. As early as the 1950s, researchers initiated the development of statistical language models, laying the foundational blocks for machine translation by successfully translating a small corpus of 60 Russian sentences into English [14]. Although rudimentary, these early models displayed promising results.

The 1980s witnessed a transition towards neural network-based language models, which exhibited higher complexity and efficacy compared to their statistical predecessors [15] [16]. Such models excelled in predicting the likelihood of word sequences, thereby elevating the performance metrics of language models.

The 2010s marked a watershed moment for the development of LLMs, spurred by the advent of voluminous datasets comprising text and code. The sheer scale and complexity of these datasets enabled the training of increasingly intricate LLMs. A pivotal moment for public accessibility to LLM technology occurred with the release of OpenAI's Generative Pre-trained Transformer (GPT) series, commencing with GPT in 2018, followed by the more advanced GPT-2 in 2019, and culminating in the release of GPT-3 in 2020 [13].

In the realm of database technologies tailored for machine learning applications, vector databases occupy a unique niche. Specifically engineered for the storage and querying of high-dimensional vectors, these databases are gaining traction in the academic and industrial communities alike. High-dimensional vectors encapsulate a broad range of data modalities, encompassing text, images, and auditory signals, among others [17]. The cornerstone technology in the area of vector databases is FAISS, an indexing and search system that was open-sourced by Facebook in 2017 [18].

Vector databases employ a myriad of sophisticated indexing and search algorithms designed for high-dimensionality [9] [10]. These algorithmic approaches facilitate the efficient and precise retrieval of data, thereby solving one of the major challenges associated with the high-dimensional data spaces generated by various machine learning models, including Large Language Models (LLMs) [19]. Given their capability to handle enormous volumes of complex data while maintaining query efficiency, vector databases are finding increased adoption for a plethora of applications, such as document retrieval, recommendation systems, and even aiding in the training regimes of generative models like LLMs.

Their surge in popularity can be attributed to their prowess in managing the high dimensionality that characterizes many forms of contemporary data. Moreover, their performance in returning accurate and fast search results has proven to be advantageous for several machine learning and natural language processing tasks [21] [40] [41] [22]. Thus, the potential for integrating vector databases with existing LLMs to expedite tasks such as similarity search or contextual data retrieval is an emerging avenue that beckons further scholarly exploration.

## III. Methodology & Implementation

Incorporating custom, context-relevant data into Large Language Models (LLMs) enhances their performance and specificity for specialized tasks [23]–[25]. Vector databases serve as an efficient medium for this process, offering scalable storage and rapid query of high-dimensional data vectors [17]. Their use not only speeds up the embedding process but also improves the model's accuracy [20], making the integration of vector databases a promising avenue in advancing LLM capabilities.

We present our comprehensive methodology, including the types of data we have chosen to embed into the LLM, the hardware configurations employed, the architecture of the pipeline we have constructed, and the final model that was ultimately utilized

### A. *Data*

The primary data corpus for this study was obtained from the PARDOK database [26], an expansive digital archive housing a range of documents affiliated with the Berlin Parliament. These documents encompass parliamentary inquiries, legislative initiatives, committee reports, and transcripts of plenary sessions. For the purposes of this investigation, we specifically focused on parliamentary inquiries. The dataset culled from PARDOK consisted of approximately 6,000 documents, all of which were scraped from the PARDOK website and subsequently stored in a dedicated file system folder.

### B. *Hardware Configuration*

- System Specifications:

The computational backbone of the prototype operates on consumer-grade hardware components. The server executing the prototype boasts an Intel Core i5-13400f processor, an RTX 3090 graphics card with 24 GB of memory, 64 GB DDR4 RAM, and a storage capacity of 2 TB provided by an NVME SSD.

- VRAM Utilization:

The VRAM consumption is allocated across various components of the pipeline. Specifically, the Text Inference Model utilizes 15334 MiB of VRAM, while the Embeddings Model uses 3232 MiB.

### C. *Pipeline Architecture*

In response to the limitations inherent in existing Large Language Models (LLMs) with respect to input size and context management, we have architected a specialized pipeline designed for handling voluminous textual data. Conventional LLMs like ChatGPT are constrained by a maximum token limit—approximately 16,000 tokens [27] in current public versions—restricting their ability to manage expansive contexts or documents.

A prevalent workaround entails passing summarized documents as the model's context. However, this approach compromises the granularity of the information, which may be essential for resolving specific queries. Moreover, summarization does not efficiently utilize the available context size and still faces token limitations.

Our pipeline adopts a distinct approach that leverages a vector database to identify and select salient information pertaining to a query or statement. This method enables the retrieval of pertinent data from a virtually limitless pool of documents. Upon initialization, the pipeline ingests PDF documents and subsequently responds to queries concerning these documents by querying the vector database to identify relevant context. This mechanism not only alleviates the token constraints but also facilitates a more accurate and expansive contextual understanding for the LLM.

This methodology offers a scalable solution for managing extensive datasets and ensures that pertinent details are not lost, thereby providing a more comprehensive and nuanced response to queries.

### 1) *Text Splitter Implementation*

For the efficient population of our vector database, the raw PDF documents need to be transformed into a textual format. This transformation is accomplished through the PyPDFLoader function [28], a feature available in the Langchain library. The function is configured to convert an entire folder of PDFs into segmented lists of paragraphs.

In our specific implementation, each document is divided into blocks of text, each comprising approximately 500 words with a 20-word overlap as shown in Table I. This design decision was made after meticulous consideration. We observed that too short a text block could lead to loss of contextual information, especially if a sentence is broken between blocks or if subsequent sentences refer to earlier content in the block. Conversely, making the text blocks excessively large risks compromising the quality of the generated embeddings, as we found during preliminary experiments.

TABLE I: Statistics of Text Transformation and Vector Database Population

| Characteristic | Description |
|---|---|
| Text Block Size | Approximately 500 words |
| Overlap Words | 20 words |
| Text Source | PDF Documents |
| Transformation Function | PyPDFLoader |
| Library Used | Langchain |
| Trade-off Considerations | Context vs. Embedding Quality |
| Database Population Efficiency | High |

After a series of basic tests, it was empirically determined that a block size of around 500 words offers the most balanced trade-off between maintaining context and ensuring effective embeddings. This approach allows us to efficiently populate the vector database without sacrificing the quality or integrity of the information being processed.

### 2) *Embeddings*

For the embedding phase of our pipeline, we employed an SBERT (Sentence-BERT) model, as introduced by Reimers & Gurevych [29]. Specifically, we utilized the *All-Roberta-Large-v1* model accessible through Hugging Face's repository [30]. The model is designed to handle an input sequence length of up to 514 tokens and returns a feature vector with 1024 dimensions. Although there are other long-former models capable of processing larger token sequences, exceeding 4096 tokens, we chose the aforementioned model for its balanced capabilities. This choice was particularly influenced by the absence of GPU support in our vector database for similarity searching, which raised concerns about the computational efficiency of higher-dimensional embeddings when operated on a CPU. Preliminary testing validated the efficacy of the 1024-dimensional vectors for similarity searches, which were subsequently stored in the vector database for future querying.

### 3) *Vector Database Configuration*

For our study, we used Milvus as our vector database due to its comprehensive documentation and its robustness as an open-source platform. The vector database schema includes fields such as ID, embeddings, sentences, timestamp, category, active status, vector name, author ID, language, source, and creation timestamp, with specific data types and details tailored for each field.

a. **Dimensionality**: The dimensionality of the embeddings in the vector database is set to 1024. This is to ensure compatibility with the output dimensions generated by our selected SBERT model.
b. **Indexing Method**: We employed the **IVF_FLAT** indexing algorithm to optimize query time. The IVF_FLAT (Inverted File with Flat Structure) algorithm is generally used as part of approximate nearest neighbor search in high-dimensional spaces. The technique is often associated with large-scale information retrieval and has been widely discussed in the literature related to computer vision, machine learning, and databases [32]. This technique partitions the dataset into smaller clusters and calculates distances between cluster centroids and input vectors, thereby reducing computational overhead

during queries as mathmaticall shown below:

$$c_i = \arg\min_c \sum_{x \in C_i} \|x - c\|^2 \tag{1}$$

$$d(x,q) = \|x - q\|_2 = \sqrt{\sum_{i=1}^{D}(x_i - q_i)^2} \tag{2}$$

(1) represent the clustering step mathematically, typically achieved using k-means. During the search step, the distance between a query vector q and the vectors in the closest cluster Ci is often calculated using the distance (2) where D is the dimensionality of the vectors.

c. **Metric Type**: For the distance metric, we used the L2 norm, commonly known as the Euclidean distance [31]. This is mathematically expressed as below.

$$d(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{3}$$

Leveraging this metric and the **IVF_FLAT** indexing method, Milvus efficiently ranks the top n vectors that are most relevant to the input vector.

d. **Query Execution**: Upon receiving an input vector, the database performs a similarity search, yielding a list of top-ranking paragraphs from all the documents stored.

### *4) Algorithmic Overview for Configuration and Query Execution*

**Algorithm 1** Vector Database Configuration and Query Execution

```
1: Input: Input Vector $Q$, Number of Top Matches to Retrieve $n$
2: Initialize Milvus Vector Database:
3: Set Dimensionality to 1024
4: Set Indexing Method to IVF_FLAT
5: Set Metric Type to L2
6: function EMBEDPARAGRAPHSINTOVECTORS
7:     Embed paragraphs using SBERT
8:     Store vectors in Milvus Database
9: end function
10: function PERFORMQUERY($Q, n$)
11:     Partition vectors into clusters                ▷ k-means for centroids
12:     for each cluster $C_i$ do
13:         $c_i = \arg\min_c \sum_{x \in C_i} \|x - c\|^2$
14:     end for
15:     Find closest cluster to $Q$                        ▷ Use L2 norm
16:     $C_{\text{min}}$ = cluster closest to $Q$
17:     Search $C_{\text{min}}$ for top $n$ vectors                ▷ Use L2 norm
18:     Sort vectors based on distance
19:     Retrieve top $n$ closest vectors
20:     Output corresponding paragraphs
21: end function
22: Execute PERFORMQUERY($Q, n$)
```

### *5) Model Configuration*

We employed a Falcon-based fine-tuning approach for our language model, specifically the *OpenAssistant/falcon-7b-sft-mix-2000 model* [33] [34]. This model has been fine-tuned for the German language and is tailored for instructional tasks. Despite its capacity for additional fine-tuning, we chose not to further optimize it for our specific use case. We reason that extensive fine-tuning may not significantly impact the quality of generated summaries and answers, as the primary role of the model in our system is to effectively summarize and recall paragraphs in response to queries.

#### *a) Choice of Model*

Before the release of Lama2 [35], Falcon models were among the few commercially licensed solutions available. Although the 40b Falcon model topped the Huggingface LLM leaderboard, we opted for the 7b version due to hardware limitations—specifically, the 40b model could not be accommodated by our available GPU resources.

#### *b) Prompt Structure*

Upon retrieving the top-n most relevant paragraphs from our vector database, we construct a prompt to pass to the model as follows:

```
<|prompter|>
Use the following information as context:
{context}
to answer the following question:
{question}
Answer:<|endoftext|><|assistant|>
```

This specific token structure is crucial, as the model has been fine-tuned using similar tokens.

#### *c) Deployment*

For deployment, we utilized *Huggingface's text-generation-inference* (TGI) [36], which provided us with a high-performance, scalable, and cost-effective platform for serving large language models. TGI also offers valuable features like quantization and continuous batching, which will be particularly beneficial when scaling to larger models, such as LAMA2-70b, that require workload parallelization.

## IV. RESULTS

For our experiment, we initially created 75 questions using an AI model (*OpenAI gpt-3.5-model*) from 7 sample PDF documents. These questions were posed to our custom model equipped with a Vector Database, as well as to the *OpenAI gpt-3.5-model*. We recorded these results for analysis. Further, we also had these answers evaluated and scored by the *OpenAI gpt-3.5-model*, the details of which were saved for final comparison.

The generated dataset contains 75 records comprising questions and answers related to parliamentary inquiries. Each record includes the following columns: *Question*, *Answer*, *Status Code*, and *Response Time*. The benchmark dataset can be accessed via the link at the bottom of this page[1]. The questions were generated using a tool called *Humata*[2], which employs the *OpenAI's gpt-3.5-turbo* model and indexes text chunks into a vector database provided by *Supabase*[3].

Our dataset consists of 72 questions and 71 answers, with some repetitions noted. Assessing the quality and relevance of these answers requires a nuanced approach, potentially involving more sophisticated linguistic analysis. Every query in our study returned a successful status code of 200, with response times between approximately 0.132 and 0.209 seconds.

In comparing our model to alternatives, we sought a counterpart to OpenAI's models that could be locally hosted and meet commercial licensing conditions. Our evaluations involved generating and analyzing responses from our model for both accuracy and diversity, using OpenAI's GPT-3.5 Turbo model as a benchmark.

The unique challenges of chatbot performance, such as linguistic comprehension and context relevance, were addressed using *Vicuna-13B* [38]. Preliminary insights from *Vicuna-13B*, which aligns closely in quality with leading models like OpenAI ChatGPT, prompted further consideration of models like GPT-4 as potential evaluation tools [37], [39].

Our findings alongside responses from our model (A1) and OpenAI's GPT-3.5 Turbo (A2) to our questions, with 73 distinct reasoning explanations documented. Initial data suggests GPT-3.5 Turbo has an edge over A1, as depicted in Figure 1 and reflected in quality scores. Yet, A1 demonstrated faster inference times. While these initial results are based on Vicuna-13B's evaluations, they highlight the need for a deeper, comprehensive analysis. Both models, however, exhibited consistent performance metrics.

## V. Conclusion

The consistent performance of both models further emphasizes the untapped potential of leveraging vector databases in enhancing language models like LLMs. Vector databases could offer a more structured and efficient way to manage and query the high-dimensional data, thereby potentially improving the models' understanding and processing speed. This opens new avenues for optimizing and scaling up language models for specific tasks that require rapid and accurate text analysis and interpretation.

Future work can take several directions. Data expansion to include a more diverse set of documents, incorporating graphical elements in the analysis, and upgrading the hardware resources are immediate considerations.

[1]https://github.com/fipso/ml-itdz/blob/main/benchmarks/Benchmark.csv
[2]Humata - ChatGPT for all your files. (o. D.). https://www.humata.ai/
[3]Supabase - Open Source Firebase alternative. https://supabase.com/

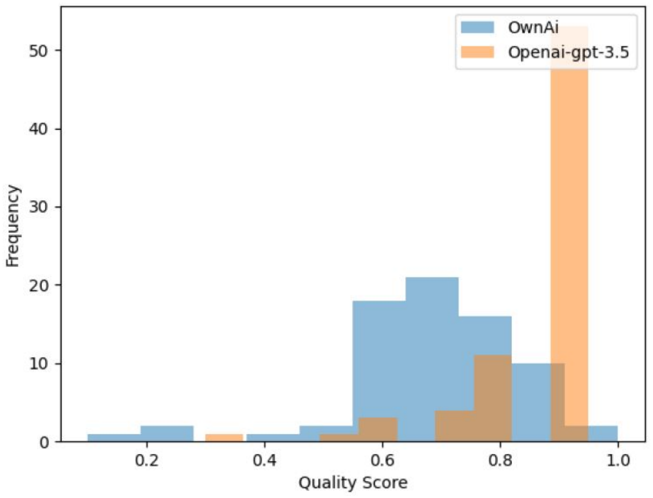

Fig. 1: Distribution of Quality Scores for Both AI Models

## References

[1] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal large language models," arXiv preprint arXiv:2203.15556, 2022.

[2] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," arXiv preprint arXiv:2211.09085, 2022.

[3] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," CoRR, vol. abs/1609.08144, 2016.

[4] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, Berlin, Germany, August 7-12, 2016. The Association for Computer Linguistics, 2016.

[5] J. Huang and K. C. Chang, "Towards reasoning in large language models: A survey," CoRR, vol. abs/2212.10403, 2022.

[6] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," arXiv preprint arXiv:2303.04226, 2023.

[7] J. Li, T. Tang, W. X. Zhao, and J. Wen, "Pretrained language model for text generation: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, Virtual Event / Montreal, Canada, 19-27 August 2021, Z. Zhou, Ed., pp. 4492–4499, ijcai.org, 2021.

[8] T. B. Brown, B. F. Mann, N. C. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. C. Wu, C. Winter, and D. Amodei, "Language models are

Few-Shot learners," arXiv, Cornell University, 2020. Online. Available: https://doi.org/10.48550/arxiv.2005.14165.

[9] Guo, R., Luan, X., Xiang, L., Yan, X., Yi, X., Luo, J., Cheng, Q., Xu, W., Luo, J., Liu, F., others (2022). Manu: a cloud native vector database management system. Proceedings of the VLDB Endowment, 15(12), 3548–3561.

[10] P. Mishra, "3 Ways Vector Databases Take Your LLM Use Cases to the Next Level," LinkedIn, 18-Jun-2023. Online. Available: https://www.linkedin.com/pulse/3-ways-vector-databases-take-your-llm-use-cases-next-level-mishra. Accessed: 20-09-2023.

[11] M. Shanahan, "Talking about large language models," arXiv preprint arXiv:2212.03551, 2022.

[12] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," arXiv preprint arXiv:2201.11903, 2022.

[13] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, and Y. Du, "A survey of large language models," arXiv preprint arXiv:2303.18223, 2023. Online. Available: https://arxiv.org/abs/2303.18223

[14] A. A. Patel and A. U. Arasanipalai, "Introduction to NLP," in *Applied Natural Language Processing in the Enterprise*, 1st ed., O'Reilly Media, Inc., May 2021. Online. Available: https://www.oreilly.com/library/view/applied-natural-language/9781492062561/ch01.html/.

[15] D. Khurana, A. Koli, K. Khatter, et al., "Natural language processing: state of the art, current trends and challenges," Multimed. Tools Appl., vol. 82, pp. 3713–3744, 2023. Online. Available: https://doi.org/10.1007/s11042-022-13428-4

[16] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland, May 22-27, 2022. Association for Computational Linguistics, 2022, pp. 8493–8502.

[17] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu et al., "Milvus: A Purpose-Built Vector Data Management System," in Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2614–2627.

[18] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2019.

[19] J. MSV, "How Large Language Models Fuel the Rise of Vector Databases," The New Stack, 16-Jun-2023. Online. Available: https://thenewstack.io/how-large-language-models-fuel-the-rise-of-vector-databases/. Accessed: 20-09-2023.

[20] M. Pandey, "10 Best Vector Database for Building LLMs," Analytics India Magazine, 28-Jun-2023. Online. Available: https://analyticsindiamag.com/10-best-vector-database-for-building-llms/. Accessed: 20-09-2023.

[21] A. Nisha, "What are Vector Databases and Why Are They Important for LLMs?" KDnuggets, 19-Jun-2023. Online. Available: https://www.kdnuggets.com/2023/06/vector-databases-important-llms.html. Accessed: 20-09-2023.

[22] L. Roque, "Document-Oriented Agents: A Journey with Vector Databases, LLMs, Langchain, FastAPI, and Docker," Towards Data Science, 12-Apr-2022. Online. Available: https://towardsdatascience.com/document-oriented-agents-a-journey-with-vector-databases-llms-langchain-fastapi-and-docker-be0efcd229f4. Accessed: 20-09-2023.

[23] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, Dec. 4-9, 2017, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., pp. 4299-4307.

[24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," CoRR, vol. abs/2203.02155, 2022.

[25] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. F. Christiano, and G. Irving, "Fine-tuning language models from human preferences," CoRR, vol. abs/1909.08593, 2019.

[26] Abgeordnetenhaus Berlin, "PARDOK," Accessed: 20-09-2023, Online. Available: https://pardok.parlament-berlin.de/portala/browse.tt.html.

[27] OpenAI Community, 'GPT-3.5 Turbo 16k Maximum Response Length,' OpenAI Community Forums, 2023. Online. Available: https://community.openai.com/t/gpt-3-5-turbo-16k-maximum-response-length/266264. Accessed: 20-09-2023."

[28] Langchain, "Document Loaders - PDF," Accessed: Accessed: 20-09-2023, Online. Available: https://python.langchain.com/docs/modules/data_connection/document.

[29] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," arXiv preprint arXiv:1908.10084, 2019.

[30] Hugging Face, "All Roberta Large v1 Model," Accessed: 20-09-2023, Online. Available:https://huggingface.co/sentence-transformers/all-roberta-large-v1.

[31] P.-E. Danielsson, "Euclidean distance mapping," in Computer Graphics and Image Processing, vol. 14, no. 3, pp. 227-248, 1980. Online. Available: https://www.sciencedirect.com/science/article/pii/0146664X80900544. doi: 10.1016/0146-664X(80)90054-4.

[32] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, "Product Quantization for Nearest Neighbor Search," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pp. 117-128, Jan. 2011. Online. Available: DOI:10.1109/TPAMI.2010.57

[33] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo, "Falcon-40B: an open large language model with state-of-the-art performance," 2023.

[34] Hugging Face, "Falcon-7B Model by TIIUAE," Accessed: 20-09-2023, Online. Available: https://huggingface.co/tiiuae/falcon-7b

[35] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, and A. Rodriguez, "Llama: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.

[36] Huggingface, "GitHub - Huggingface/Text-generation-Inference: Large Language Model Text Generation Inference," GitHub, [Online]. Available: urlhttps://github.com/huggingface/text-generation-inference.

[37] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, and H. Zhang, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," arXiv preprint arXiv:2306.05685, 2023.

[38] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality," Mar. 2023. Online. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[39] The Vicuna Team, "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality," Mar. 30, 2023. Online. Available: https://lmsys.org/blog/2023-03-30-vicuna/.

[40] R. B. Tareaf et al., "Aseds: Towards automatic social emotion detection system using facebook reactions," 2018 IEEE 20th international conference on high performance computing and communications; IEEE 16th international conference on smart city; IEEE 4th international conference on data science and systems (HPCC/SmartCity/DSS), IEEE, 2018.

[41] R. B. Tareaf, "MBTI BERT: A Transformer-Based Machine Learning Approach Using MBTI Model For Textual Inputs," 2022 IEEE 24th Int Conf on High Performance Computing  Communications; 8th Int Conf on Data Science  Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud  Big Data Systems  Application (HPCC/DSS/SmartCity/DependSys), IEEE, 2022.