You Just Need Another LLM
Improving LMTutor embedding model with LLMs

Baraa Zekeria [*]
A17137018
bzekeria@ucsd.edu

Divyansh Srivastava [*]
A59012150
ddivyansh@ucsd.edu

Reventh Sharma [*]
A59026450
resharma@ucsd.edu

Abstract

Large language models (LLMs) trained on massive internet-scale datasets are increasingly being used to improve human efficiency on tasks including knowledge retrieval, coding, and teaching. These models provide an out-of-box method to boost performance on tasks that previously required manual collection of large amounts of domain-specific data. Recently, *LMTutor* has been proposed as a platform to improve question-answering capability from pre-defined documents by leveraging LLMs. However, LMTutor requires training an embedding model on a limited dataset to select relevant documents for LLM prompt, limiting the embedding model's capability to capture intent from a diverse set of question-document pairs. In this project, we show that LLM embedding can be effectively used to represent diverse range of documents and queries, thereby removing the need to train an embedding model. We demonstrate that embeddings from LLMs such as Vicuna and LLama are better than sentence transformer embeddings when the corpus contains similar documents. We also perform a large scale analysis of embeddings from various hidden layers in LLMs. Our code is available on github.

1 Introduction

The advent of Large Language Models (LLMs) [4, 1, 16] has accelerated the development of applications requiring human-like text generation and communication. Recently, LMTutor, a Language Model (LM)-powered tutor, has been proposed as a platform to improve question-answering capability from a set of pre-defined documents by leveraging LLMs. LMTutor uses a sentence transformer to embed documents and queries in a vector space, and later the retrieved documents are added to the prompt as context and passed to LLM for generating answer based on context. The overall pipeline for LMTutor is shown in Fig 1. LMTutor currently uses an Instruct Embedding model to embed documents and queries. Answers for given queries are obtained by prompting an LLM model with documents relevant to that query. This helps in the generation of relevant answers by the LLM. However, document retrieval can be enhanced by capturing the context hidden in queries and documents. Hence, we propose the use of an LLM-based embedding model for retrieval since LLMs are known to perform well in a zero-shot manner even on queries that were not seen during their training stage [10]. We argue that LMTutor will benefit from the contextual understanding and semantic richness offered by LLMs, thereby elevating its performance on a wide range of natural language processing tasks such as question-answering, which is the main goal of LMTutor. We believe that this enhancement

[*]Equal contribution. Ordering by name

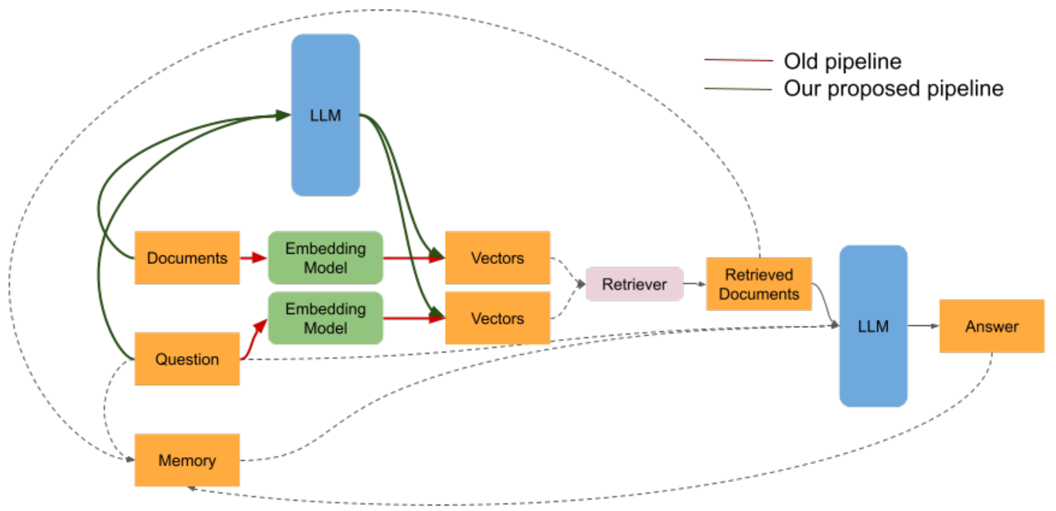


Figure 1: Overview of our LMTutor pipeline. LMTutor originally uses a sentence transformer to embed documents and queries in a vector space (red arrows), and later the retrieved documents are added to the prompt as context and passed to LLM for generating answer based on context. Our proposed change is to replace existing embedding model (green arrows) with a pre-trained LLM such as LLama2. Source: DSC250 lecture slides

will enable LMTutor to provide more accurate, context-aware, and informative document selection, making it a more effective tool for aiding students in various educational and informational (course logistics) scenarios. We summarize our contributions in this work below:

1. We demonstrate that LLM Embeddings are better than sentence transformers when corpus contains similar documents. In particular, we show that LLM embeddings from LLama-based models are better at capturing the context of the document and query.
2. We do a large scale comparison of embeddings from LLama-based models and different hidden layers in the model.
3. We qualitatively demonstrate a scenario where LLM embeddings are better than sentence transformer embeddings.

2 Related work

Q&A Systems Question-answering is an area in Natural Language Processing (NLP) that answers questions from humans, usually in a chatbot-type setting. These systems can be categorized into two types: First, open-domain or generic systems[3, 16, 1, 4], which tend to answer any general question in the wild and are usually trained on large-scale datasets to learn the semantics of dialogue. Second, closed-domain or retrieval-based systems[17, 5, 6], which utilize pre-defined paragraphs or text corpora and extract relevant answers based on some similarity metrics. The recent development of the NLP-based deep learning approach has led to exponential advancements in the field of question-answering. LLMs, including ChatGPT[1] and Llama, can be characterized into open-domain systems and have shown a human-level understanding of concepts in dialogues. Further, closed-domain methods, including AQUA (Vargas et al., 2010)[17], which uses a novel similarity algorithm based on ontological structures, and a recent method from Hao et al. (2018)[[6]] which uses cross- attention to calculate mutual influence between question-answer representation have been highly successful. LMTutor could be categorized as a hybrid system that uses an embedding model to generate vectors for the pre-defined text and question, uses a similarity metric to get relevant content similar to closed-domain systems, and then passes the retrieved content and question as a prompt to the open-domain systems (LLMs). We argue that LLM embedding can represent semantic concepts better than the embedding model trained on a limited dataset, and using LLMs as the embedding model can significantly improve the performance of LMTutor.

ITS Systems and QA Intelligent Tutoring Systems (ITS) are crucial in catering to diverse student needs in the age of advanced LLMs and online courses. However, LLMs often generate inaccurate responses due to their limited domain knowledge and analytical reasoning capabilities. Integrating a knowledge base (KB) with LLMs, referred to as KB-LLM intelligent tutors, has shown improvements in accuracy and pedagogical effectiveness [2]. Another approach involves fine-tuning LLMs with subject-specific data, as demonstrated by VICUNA LLM (SPOCK) in a biology course, resulting in high relevance scores [14]. Our proposed framework combines elements from the CLASS-inspired and KB-inspired approaches. It utilizes LLM-based embeddings to identify relevant documents for a query (retrieval step) and employs the KB-inspired framework, i.e. relevant documents are added as knowledge base, to generate natural language responses from these documents.

Semantic embeddings of DL models Transformer based Deep Language models capture different abstraction of tokens within each hidden layer. For example, Roger et.al [13] concludes in their review article that lower layers of BERT capture linear word order information, while semantic relation is spread through all layers of the model. Since information is distributed throughout the deep architecture of LLMs, our study will experiment on different embedding layers to visualize knowledge captured and derive the combination of layers within LLM to use for encoding. Applying LLMs to word embeddings can be likened to a related concept of sentence embeddings. Jiang et.al [8] study addresses the ongoing research area of applying LLMs specifically to sentence embeddings, introducing a novel in-context learning-based method aimed at enhancing the performance of sentence embeddings. Using prompt-based representation method for autoregressive models, construction of a demonstration set for in-context learning, and systematic scaling of LLMs to varying model sizes, the authors present potential improvements LLMs can bring to the generation of high-quality sentence embeddings without resorting to fine-tuning—a critical facet aligning with this project's objectives in the realm of LLM applications to word embeddings.

3 Methodology

In this section, we describe our approach to using LLMs as an embedding model for LMTutor. We first describe the LMTutor framework and general problem formulation in Section 3.1, describe architecture of LMTutor's original embedding model and planned LLM replacements in Section 3.2, and finally describe our approach to using LLMs in Section 3.3.

3.1 Formulation

Let D be the set of documents (e.g., course content) and q be a query/question with relevant answer in the documents. Let the document containing the answer to q be $d_q^* \in D$. Let E_θ be the embedding model parameterized by θ and L_ϕ be the LLM parameterized by ϕ. The goal of LMTutor is to generate natural language response a to q by finding d_q^*. The overall pipeline for the proposed LMTutor approach can be divided into three main components:

- **Document embeddings and vector store generation**: The embedding model E_θ is used to generate embeddings for each document $d \in D$. The embeddings are then stored in a vector store V_D, which is a dictionary mapping document IDs to their embeddings.

$$V_D = \{E_\theta(d_i)\}_{i=1,2,\ldots,|D|} \tag{1}$$

- **Document retrieval based on query**: The embedding model E_θ is used to generate an embedding for the question q. It is possible to use the same embedding model as in the previous step, or a different model. For example, the Two Tower model [7] is a representation-based ranker architecture, which independently computes embeddings for the query and documents and estimates their similarity via interaction between them at the output layer. The document d with the highest similarity to q is retrieved from the vector store V_D. LMTutor uses FAISS [9] for efficient similarity search.

$$\begin{aligned} d^* &= \arg\max_{d \in D} \text{sim}(E_\theta(q), E_\theta(d)) \\ &= \arg\max_{d \in D} \text{sim}(q, d) \end{aligned}$$

  Note that *sim* can be any function to compare two embedding vectors, for example, cosine similarity.

- **Answer generation**: The language model L_ϕ is used to generate a response a to the question q based on the retrieved document d^* as context. The idea is to use the language model for human-like responses to the question.

$$a = L_\phi(q, d^*)$$

3.2 Architecture

LMTutor consists of two models: Embedding model E_θ, which is used to generate document/query embeddings to find documents similar to query, and Language model, which is used to generate answers from retrieved documents. Embedding model used by LMTutor is recently released Instructor[15], where every text input is embedded together with instructions explaining the use case. This single embedding model is used to generate embeddings for both documents and queries. For the large language model, LMTutor uses Vicuna[18] (which is fine-tuned from LLaMA) with supervised instruction fine-tuning. We hypothesize that the large language model can be used to replace the embedding model in LMTutor, since they are trained on a much larger dataset and can potentially capture more information about the documents. We will look at two popular LLM models for this project:

- **LLaMa2**: LLaMA works by taking a sequence of words as an input and predicts a next word to recursively generate text. Their smallest model, LLaMA-7B, has 7 billion parameters and is trained on one trillion tokens. The architecure of llama consists of an embedding layer to map tokens to embeddings, followed by arbitray number of *LlamaDecoderLayer* blocks, which are composed of a self-attention layer, a feed-forward layer, and a layer normalization layer. The output of the last *LlamaDecoderLayer* is passed through a linear layer to predict the next token.
- **Vicuna**: Vicuna is a large language model finetuned with LLaMA, but with a different training objective. Vicuna is trained using supervised instruction fine-tuning, where the model is trained to predict the next word given the previous words and the instruction. Our aim to use Vicuna is to understand if instruction-fine tuned models can be a better replacement than the original counterparts.

3.3 Problem Statement

Our primary objective in this project is to analyze the embeddings of Large Language Models (LLMs) and assess their potential as a superior and training-free replacement for the existing LMTutor's embedding model. More precisely, we will replace the embedding model E_θ with an LLM and evaluate the performance of the embedding model and the overall LMTutor framework. In this section, we will mathematically describe the architecture of LLama-based models along with a mathematical formulation on how we obtain embeddings from these models. Let E_θ be a LLama-based model. Let p represent input document/query. There are four types of layers in LLaMA-based models:

- Tokenizer T: The input prompt p is tokenized into a sequence of tokens $t_1, t_2, ..., t_n$. Note that there are n tokens in the sequence.
- Embedding layer M: The tokens are embedded into a sequence of embeddings $m_1, m_2, ..., m_n$. This layer is esentially a lookup table, where each token is mapped to a vector of size d i.e. $m_i \in \mathbb{R}^d$. The output of this layer is a 2D tensor of size $n \times d$.
- LLaMA Decoder Layers H_i: The output of the embedding layer is passed through K LLaMA decoder layers. The output of the i^{th} layer is $h_i \in n \times \mathbb{R}^d$.
- Output probability layer O: The output of the last LLaMA decoder layer is passed through a linear layer to predict the next token. The output of this layer is a probability distribution over the vocabulary of size v i.e. $o \in n \times \mathbb{R}^v$. The probability over the last token is used to get the next word in the sequence.

The overall LLM model can be represented as:

$$E_\theta(p) = O(H_K(H_{K-1}(...(H_1(E(T(p)))))) \tag{2}$$

Now we will describe how we obtain embeddings V from the LLaMA-based models for a query/document p. In this project, we will look at three kinds of embeddings from the LLaMA-based models:

Layer-i We sum the embeddings of all the tokens from the i^{th} LLama decoder layer to get the embedding for the input. The output of this layer is a 2D tensor of size $n \times d$ and the embedding we obtain is of size d as formulated below:

$$V(p) = \sum_{j=1}^{n} H_i(H_{i-1}(...(H_1(E(T(p))))_j \tag{3}$$

Token In this method, we take the embedding of tokens from the token embedding layer. The idea here is to understand if the initial embedding of tokens is sufficient to capture the semantics of the input. The output of this layer is a 2D tensor of size $n \times d$ and the embedding we obtain is of size d as formulated below:

$$V(p) = \sum_{j=1}^{n} E(T(p)) \tag{4}$$

Last-Token In this method, we modify the prompt and obtain prompt p' as follows: *This sentence: p means in one word:* and take the embedding of the last token from the last LLama decoder layer. The idea here is to understand if the last token embedding is sufficient to capture the semantics of the input if we engineer the prompt to force model to capture the semantics of the input in the next predicted word.

$$V(p) = H_K(H_{K-1}(...(H_1(E(T(p'))))))_n \tag{5}$$

In the next section, we will describe the experiments to evaluate the performance of LLaMA-based models as an embedding model for LMTutor based on the above embedding formulations.

4 Experiments

In this section, we replace the embedding model E_θ with various LLMs, and evaluate the performance of the embedding model and the overall LMTutor framework. We first describe our experimental setup with dataset and evaluation metrics in Section 4.1, evaluate popular LLMs as embedding models in Section 4.2, and finally evaluate embeddings of different layers in Section 4.3.

4.1 Setup

Our experiments run on a server with 16 CPU cores, 16 GB RAM, and 1 Nvidia Tesla V100 GPU. We use langchain and transformers library for downloading and using pretrained LLMs. Our codebase builds on top of the original LMTutor codebase, which is available at here. Section 4.1.1 describes the datasets used for evaluation, and Section 4.1.2 describes the evaluation metrics used in our experiments.

4.1.1 Dataset

We use publicly available context-based question answering datasets for evaluating the performance of LLMs as embedding models. More precisely, we use context as the documents, and the question as the query and evaluate the performance of LLMs in retrieving the correct document for a given query. The dataset used for evaluation is:

1. **SQuAD1.1** [12]: The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. We use validation dataset of SQUAD which consists of 10,570 questions and contains 2067 unique contexts or documents.

4.1.2 Evaluation Metrics

We follow the evaluation metrics used in past work [12, 11] to evaluate the performance of LLMs as embedding models. Following the notation defined in Section 3.1, let D_q^k be the set of top-k retrieved documents for a given query q and d^* be the groundtruth document for q. Further, define the rank(q) as the position of d_q^* in the retrieved documents.The evaluation metrics used in our experiments are:

- **Recall@k**: The percentage of queries for which the groundtruth document is present in the top-k retrieved documents. We use FAISS to directly retrieve the top-k documents from the embedding space closest to the query embedding. Then, Recall@k is defined as:

$$\text{Recall@k} = \frac{\sum_{q \in Q}(d_q^* \in D_q^k)}{|Q|}$$

- **Mean Reciprocal Rank (MRR)**: Rank metrics provide insights into the relative position of true positive predictions within the ranked list. Then MRR can be defined as:

$$\text{MRR} = \frac{\sum_{q \in Q} \frac{1}{\text{rank}(q)}}{|Q|}$$

4.2 Evaluating LLMs as embedding models

We compare LLama-7B and Vicuna-7B models for retrieving the correct document for given questions within the **SQUAD 1.1** dataset. Table 1 shows the result of comparing the Instructor model (the baseline embedding model used in LMTutor) with embeddings generated with equation 3 on LLama2-7B and Vicuna-7B. We observe that the recall@k and MRR is consistently better for LLama based models. In particular, we observe an improvement by $\sim 8\%$ in recall score with LLama-7B model.

Table 1: Comparison of embeddings from Last hidden state of Vicuna-7B, LLama-7B and Instructor(baseline) for document retrieval on SQUAD 1.1. We observe that LLama-based embeddings perform better than Instructor model and able to beat performance by as much as 8%

Model	Parameters	Recall%@1	Recall%@5	Recall%@10	MRR(10^{-2})
Instructor (baseline)	335M	0.14	0.76	2.3	5.5
Vicuna	7B	0.58	3.82	9.65	5.3
Llama2	7B	**0.79**	**4.31**	**10.24**	**5.7**

4.3 Evaluating different hidden layers as embedding layer of LLMs

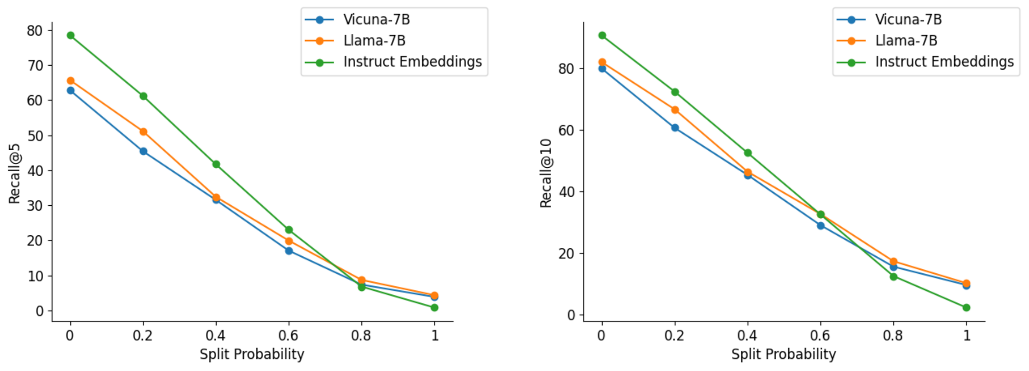
Given the deep structure of LLMs, it is crucial to identify which layer within the model can capture the most semantic context of the document/query. To determine this, we evaluate the last three hidden states of Vicuna-7B and LLama-7B on the **SQUAD 1.1** following Equation 3 and direct token embeddings following Equation 4. The results are shown in Table 2. We observe that the last hidden layer (which is the 32^{nd} layer in LLama-7B and Vicuna-7B) captures the most context for the question in LLMs. We believe this is because the last hidden layer encapsulates the maximum amount of information required to generate an answer, as it is followed by the distribution generation function which produces output tokens. Furthermore, we observe that the tokenization layer also performs better than the baseline Instructor model, proving that the LLMs are able to capture more text features from first tokenization layer itself.

4.4 Varying degree of retrieval difficulty

In this section, we try to study conditions where LLM embeddings outperform the instructor embedding model. We modify SQUAD1.1 data preprocessing to probabilistically generate one document per title or one document per context. In all the previous experiments we used one document per context, resulting in a total of 2067 documents. **Split probability** is defined to be 1 if each context is a document and 0 if each title is a document. We vary the split probability from 0 to 1 and evaluate the performance of the Instructor model and the LLMs. The results are shown in Figure 2. We

Table 2: Comparing embeddings across last three hidden layers and Token embeddings for Vicuna-7B and LLama-7B. We observe that the last hidden layer (which is the 32^{nd} layer in LLama-7B and Vicuna-7B) captures the most context for the question in LLMs. Here $R\%@k$ is Recall@k.

	Vicuna-7B					**LLama2-7B**			
Layer	R%@1	R%@5	R%@10	MRR(10^{-2})	Layer	R%@1	R%@5	R%@10	MRR(10^{-2})
Layer 32	**0.58**	**3.82**	**9.65**	**5.30**	Layer 32	**0.79**	**4.31**	**10.25**	**5.7**
Layer 31	0.50	3.22	7.58	4.44	Layer 31	0.7	3.43	8.34	3.7
Layer 30	0.46	1.94	3.95	1.61	Layer 30	0.2	1.24	2.54	1.2
Token	0.31	2.39	6.22	5.4	Token	0.34	2.05	6.07	5.5



(a) Recall@5 (b) Recall@10

Figure 2: Varying degree of retrieval difficulty with split probability. Split probability 1 implies one document per context (2000 documents) and split probability 0 implies one document per title(450 documents). We observe that LLM embeddings work better when split probability is greater than 0.75, implying that LLM embeddings are better at retrieving documents when the number of documents is large and contains similar documents.

observe that LLM embeddings work better when **split probability** is greater than 0.75, implying that LLM embeddings are better at retrieving documents when the number of documents is large and retrieval involves search through a corpus of similar documents. We believe this is because the LLM embeddings are able to capture the context better since they are trained on a much larger dataset and can capture much longer range of contexts compared to the Instructor model. Hence, LLMs are robust to searching from similar corpus of document to generate embeddings.

4.5 Query engineering

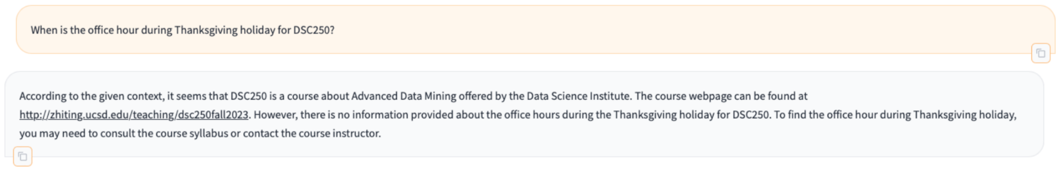
In this section, we try to modify the input query and see if we are able to obtain better embeddings directly from the last token. We follow the setting in Eq 4 and call this embedding method *Last-Token* and compare against the mean embeddings with Layer-32 for LLama-7B and Vicuna-7B. The results are shown in Table 3. We observe that though the *Last-Token* method performs better than the Instructor model, it is not able to beat the performance of the mean embeddings with Layer-32. We believe this is because the last token is not enough to capture the context of entire document or query as well as the mean embeddings.

4.6 Qualitative analysis of embeddings

In this section, we qualitatively analyze the embeddings generated by LLMs. We use the same setting as in Section 4.2 and compare the embeddings generated by the LMTutor and our embedding model. We ask the following question: *When is the office hour during Thanksgiving holiday for DSC250?*. The results are shown in Fig 3. We observe that the LMTutor is not able to generate the correct answer for the given query, whereas our embedding model is able to retrieve the correct answer.

Table 3: Result with query engineering by modifying prompt to *This sentence: p means in one word:* and using the last token as embedding. We observe that though the *Last-Token* method is not able to beat the performance of the mean embeddings with Layer-32 but comes closer to the performance of the mean embeddings for LLama-7B. Here $R\%@k$ is Recall@k.

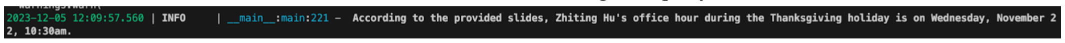
	Vicuna-7B					**LLama2-7B**			
Layer	R%@1	R%@5	R%@10	MRR(10^{-2})	Layer	R%@1	R%@5	R%@10	MRR(10^{-2})
Layer-32 (base)	**0.58**	**3.82**	**9.65**	**5.3**	Layer-32 (base)	**0.79**	**4.31**	**10.24**	**5.7**
Last-Token	0.43	2.54	6.39	3.1	Last-Token	0.69	3.78	9.63	5.34



When is the office hour during Thanksgiving holiday for DSC250?

According to the given context, it seems that DSC250 is a course about Advanced Data Mining offered by the Data Science Institute. The course webpage can be found at http://zhiting.ucsd.edu/teaching/dsc250fall2023. However, there is no information provided about the office hours during the Thanksgiving holiday for DSC250. To find the office hour during Thanksgiving holiday, you may need to consult the course syllabus or contact the course instructor.

(a) LMTutor results for a given query



```
2023-12-05 12:09:57.560 | INFO     | __main__:main:221 -  According to the provided slides, Zhiting Hu's office hour during the Thanksgiving holiday is on Wednesday, November 2
2, 10:30am.
```

(b) Our results for the same query

Figure 3: Qualitative analysis of our proposed embeddings. We demonstrate that LLMs embeddings are able to retrieve the correct answer for the given query.

5 Conclusion and Future Work

Our integration of LLMs into the LMTutor framework has demonstrated promising improvements in question-answering capabilities. Both Vicuna-7B and LLama2-7B have proven to be an effective replacements to the existing embedding model, particularly showcasing enhanced document retrieval with higher recall rates. It highlights the significance of LLMs in capturing contextual information and semantic nuances, leading to improved generation of relevant answers. However, the impact of LLM architecture and layer choice on performance metrics should be carefully considered. Our exploration of query engineering strategies, such as summarization prompts and input query modifications, provides valuable insights for optimizing LMTutor's question-answering process by improving contextual understanding.

Looking ahead, fine-tuning LLMs for specific educational domains, like social sciences or engineering, holds potential for further performance gains. The integration of user feedback remains crucial for continuous improvement, ensuring LMTutor evolves to meet dynamic user needs. Future work involves comparing the 13B/70B parameter model with Instruct Embeddings to gain insights into model architecture effectiveness. Experimenting with diverse prompt variations and validating LLM embeddings on varied datasets will contribute to assessing LMTutor's generalizability. Exploring unsupervised fine-tuning approaches, such as mapping the final layer to a linear layer, offers additional avenues for refining LLM embeddings within the LMTutor framework. These recommendations aim to refine and extend LMTutor's capabilities, positioning it as a versatile tool for diverse educational and information-seeking needs. Through continuous refinement, user feedback integration, and exploration of advanced model architectures, LMTutor is poised to become a more effective and adaptable resource for learners across various domains.

6 Acknowledgements

We appreciate our Teaching Assistants (Meng Song and Vyshnavi Sankaran Krishnan) and Prof. Zhiting Hu for their detailed feedback, enhancing the quality of our work. Special thanks to the LMTutor team (Hao Zhang, Anze Xie, Pushkar Bhuse, Yuheng Zha, Tiffany Yu, and Licheng Hu) for creating this platform and collaborating with us during project development.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Blake Castleman and Mehmet Kerem Turkcan. Examining the influence of varied levels of domain knowledge base inclusion in gpt-based intelligent tutors. 2023.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[5] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. Kbqa: learning question answering over qa corpora and knowledge bases. *arXiv preprint arXiv:1903.02419*, 2019.

[6] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, 2017.

[7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[8] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models, 2023.

[9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[10] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[11] Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating web-based question answering systems. May 2002.

[12] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[13] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. 2020.

[14] Shashank Sonkar, Lucy Liu, Debshila Basu Mallick, and Richard G. Baraniuk. Class meet spock: An education tutoring chatbot based on learning science principles. 2023.

[15] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings, 2023.

[16] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[17] Maria Vargas-Vera and Miltiadis D Lytras. Aqua: A closed-domain question answering system. *Information systems management*, 27(3):217–225, 2010.

[18] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.