

Enhancing Commercial Viability and Trustworthiness: Content Validity for LLM-Generated Outputs

Thirunithesh P R
II Year MBA
Thiagarajar School of Management
Madurai, India
thirunithesh07@tsm.ac.in

Dr. Nataraj Balasubramanian
Assistant Professor
Thiagarajar School of Management
Madurai, India
bnataraj@tsm.ac.in

Abstract—The purpose of this paper is to add content validity to the LLM generated outputs in the area of personalized content generation to increase its commercial viability. It is indeed to provide content validity to the output of LLM's to improve their trustworthiness which will eventually help organizations to confidently rely on LLM api based applications for content generation tasks. This approach constrains the knowledge of the LLM to an user created knowledge base. Whenever a new data is added to the knowledge base, the LLM Api is integrated using Langchain to identify the source or reference of data. Reference is stored along with the data in form of vectors in a vector database. During content generation, semantic search is used to fetch only the text chunks relevant to the topic from knowledge base and the knowledge of LLM will be constrained to text chunks fetched during semantic search to provide the output along with reference.

Keywords—Large Language Model, Vector Embeddings, Semantic Search, Content Validity, Generative AI, LangChain, Hallucination, NLP, Content writing

I. INTRODUCTION

Content creation jobs, such as blog writers and content writers, play a pivotal role in the digital marketing and e-commerce domains where information and engagement are currency. These professionals are the architects of the online narrative, crafting compelling, informative, and relevant pieces that captivate audiences and drive brand success. Blog writers specialize in the art of storytelling, weaving narratives that resonate with readers and provide valuable insights, while content writers excel at producing concise, informative, and SEO-optimized materials that help businesses rank higher on search engines and establish authority in their niches.

These roles demand not only exceptional writing skills but also the ability to adapt to ever-evolving digital

trends and consumer preferences, making them integral components of today's content-driven world. As the demand for high-quality online content continues to soar, LLMs can be leveraged to automate these content generation jobs by creating personalized content creation application by integrating LLM api with the help of LangChain open-source library. It will result in high quality personalized content creation in a cost-effective way

In today's dynamic landscape, Large Language Models (LLMs) have emerged as transformative innovations that are reshaping how we interact with and harness the potential of natural language. These sophisticated machine learning models, often powered by neural networks, have revolutionized various fields, from natural language processing and comprehension to content generation and decision support systems. Notably, LLMs like GPT-3 have garnered widespread attention for their extraordinary ability to autonomously generate coherent and contextually relevant text, making them indispensable tools in areas such as chatbots, automated content creation, and even creative writing.

Furthermore, LLMs have played a pivotal role in transcending language barriers, enabling real-time translation and facilitating global communication. Their applications extend across sectors like healthcare, finance, and education, where they assist in data analysis, information retrieval, and the creation of personalized learning experiences. However, the proliferation of LLM-generated content lacks reliability because of issues like privacy concerns, the spread of misinformation because of hallucination and LLM malfunctioning because of prompt injections. These issues prompt essential discussions about responsible AI development and usage.

This research aims to provide a model-based approach to improve the reliability of the LLM Api integrated applications by providing source validity to the LLM

generated output and tries to address the hallucination issue which may possibly expand its areas of application of LLM based applications and automation.

II. PROBLEM STATEMENT

Though the LLM generated content are far better and creative than the contents of the professional content writers, it still lacks trustworthiness in the business arena.

A. Challenge

Various innovative LLM based solutions lacks commercial viability in the business organizations because of the reliability issues associated with the large language models.

B. Lack of Reliability

Usually, people trust the information or data based on the credibility of its source or the credibility of the references cited. But LLM models fails to provide the source or reference details for the generated outputs along with the hallucination issues which in turn questions its reliability.

C. Need for the Research

Providing source references to the LLM generated outputs may possibly enhance the reliability of LLM Api integrated applications which in turn improves the commercial viability of such applications. These type applications will be very much useful in automating the content writing jobs.

III. METHODOLOGY

This research tries to create a dynamic vector knowledge base to store user provided credible source files in a vector database and utilize the concept of embedding based data retrieval to constrain the knowledge of the LLM to the vector knowledge base to generate the required content. It leverages the similarity search technique to identify and fetch the relevant data from the knowledge base. The LLM model will generate will be fine-tuned to provide output along with references only using the knowledge of the data fetched from similarity search.

This model uses the features of the open-source Lang Chain package to interact with the large language model. It combines with the openAI python library for prompting and fine tuning the LLM model.

A. Embedding Based Data Retrieval

Vector embedding-based information retrieval is a powerful technique that leverages mathematical representations of text to enhance search and recommendation systems. In this approach, documents and queries are transformed into dense vectors in a high-dimensional space, allowing for efficient and accurate retrieval of relevant information.

Vector embedding-based information retrieval has revolutionized the way we search for and retrieve information. By representing text as numerical vectors, we can capture semantic relationships and similarities between documents and queries. Techniques like Word2Vec and Doc2Vec have enabled the conversion of words and entire documents into continuous-valued vectors. These embeddings enable more precise matching and ranking of documents, leading to improved search results across various applications, from search engines to recommendation systems.

One of the key advantages of vector embedding-based information retrieval is its versatility. It can be applied to a wide range of tasks, including document retrieval, sentiment analysis, and content recommendation. By comparing vector representations, we can identify relevant documents, detect sentiments, and suggest personalized content. This technology has been particularly instrumental in e-commerce, where it powers product recommendations based on user behavior and product descriptions. It has also found utility in healthcare, aiding in the retrieval of relevant medical records and research papers.

B. Similarity Search

Similarity search is a fundamental technique in information retrieval and data analysis, enabling the identification of items that closely resemble a given query item. This process is essential in various fields, including information retrieval, recommendation systems, bioinformatics, and image analysis. In similarity search, algorithms and techniques are employed to measure the likeness between data points, which could be text documents, images, DNA sequences, or any other type of structured or unstructured data. These algorithms use various similarity metrics, such as cosine similarity, Euclidean distance, or Jaccard similarity, to quantify the resemblance between objects. Leveraging data indexing and search structures, similarity search enables the efficient retrieval of items that match user-defined criteria, playing a critical role in tasks like content recommendation, plagiarism detection, and pattern recognition. As the volume of data continues to grow exponentially, the importance of effective similarity search methodologies becomes increasingly evident in making sense of vast datasets and finding relevant information amidst the data deluge.

C. LLM Fine Tuning

One of the standout features of the LLM API is the level of customization and control it offers to developers. Users can fine-tune the output by providing prompts and instructions, ensuring that the generated content aligns with their specific requirements. This level of control enables developers to make the AI model generate content in a particular style, tone, or voice, ensuring consistency with their brand or application's personality. Additionally, developers can set parameters to limit the length of the generated text or specify the format, making it adaptable to various content formats such as short tweets, long articles, or

even code snippets. The level of creativity of the LLM Models can also be adjusted with the temperature parameter.

IV. DATA ANALYSIS AND MODELLING

The model can be divided into two major functions

- Adding Source files to the Knowledge Base (Vector Database)
- Content Generation using the Knowledge Base

A. Data Extraction

The source files or reference files uploaded by the user will be stored in a destination folder. These files are received and converted into pdf files initially and the textual data in those files can be extracted using the python package like PyPDF2 and stored in either a list or a variable based on the number of the files.

B. Split Text into Chunks

The uploaded files need to be split up into small text chunks because of the token limit in the LLM model. The text spitting is carried out using the character text splitter available in python. The text is split into a chunk of 1000 characters with an overlap of 200 characters to inculcate text cohesion. The text chunk should not be so big as it may possibly hit the token limit of the LLM Api.

C. Source and Reference Identification

The references and sources available in these text chunks are then extracted in form of python list by using the LLM model. The text davinci – 002 LLM of open AI is used

in this research for identifying the references. The identified references are then added to the end of each of the text chunks of a particular source files. If the LLM model fails to identify any references then the file name will be added.

D. Vectorization

These small text chunks with references are then converted into vector embeddings of 1536 dimensions using the embed function available in the openai python library. This research uses the ada V2 model for vectorization. The LLM Api key should be used to access the embedding model to convert the text chunks into vector embeddings

This can also be performed using various python packages like Word2Vec, Doc2Vec, GloVe etc. These python packages don't require the use of LLM but the dimension of vectors created varies according to the embedding technique used and it needs to be same as that of the vector database.

E. Upserting source files into vector database

The next step will be adding these vector embeddings into a vector database. The pinecone vector database is used in this research. The connection with the vector database will be initialized by using the api key and the vector database needs to be created and the dimensions of the vector database needs to match the dimensions of the embeddings created in the vectorization. The embeddings will be inserted into the vector database using an upsert query. These upsert query are the queries which are used to add the embeddings into the vector database. The syntax of this query may vary slightly based on the vector database used.

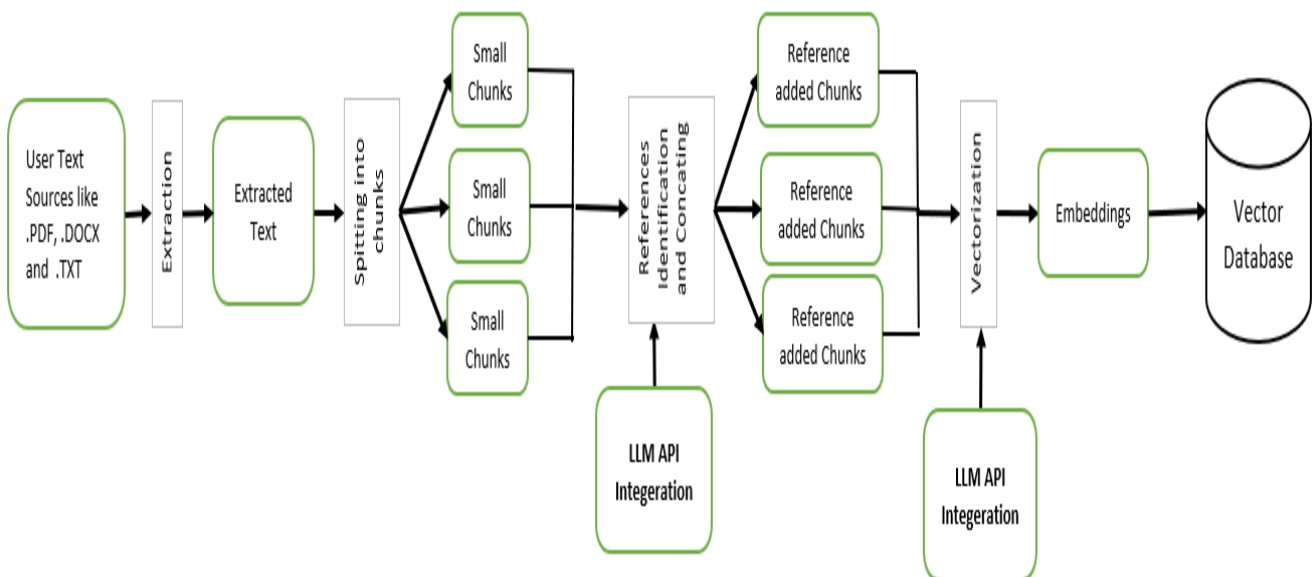


Fig. 1 Adding Source files to the Knowledge Base

F. Content Generation

This research primarily fine tunes the LLM model to generate blog content based on the title provided by the user. The title is received from the user using python's flask and the title is stored in a variable. The title is then converted into vector embeddings using the ada V2 model.

The vector embeddings are used in the similarity search to fetch the text chunks relevant to the title. The vector - distance calculation method can be chosen during the vector database creation. There are different distance calculation methods such as Euclidean, cosine can be used. The number of text chunks to be fetched can be passed on as a parameter.

The number of the text chunks to be generated can also be decided based on the similarity score threshold value. The knowledge of the LLM model will be constrained to the data fetched and the LLM model will be prompted to generate required number of sub topics for the title as per the required length only based on provided knowledge.

The large language model may sometimes reach the maximum token limit per minute because of consecutive content generation for different sub topics. Thus it is indeed to make sure that it doesn't reach the maximum threshold allocated by the LLM model. This threshold may vary according to the LLM used. This issue can be evaded by including a time wait between iterations of the sub-topic content generations. This can be achieved by using the sleep function available in the python date library.

The sub topics are then again treated as a separate heading. These sub topics will be converted into vector embeddings again and the similarity search is carried out to fetch the relevant data to generate the content of the sub topic.

The content for the topic is generated based on the data fetched for the sub topic from the knowledge base. This content generation process will be iterated to each of the sub topics individually. In this research the model is prompted to generate content for the blogs. These prompts can be personalized as per the user's needs.

The references from the text chunk used to generate the content will be passed on to the LLM model to identify the references and sources associated with these text chunks. The output is taken in the form of python list. The duplicate references were removed from the references python list.

The initial title can be passed in as a prompt to Image generation models like DALL-E to generate copyright free images required for our blogs. There are much more GAN's which can be used instead of DALL-E.

The content generated for various subtopic along with the image generated from the GAN and the references stored in the python list are converted into a word file and is formatted and provided as an output file to the user. This research uses this model to generate blog content but this content generation can be personalized as per needs of various business users and customers.

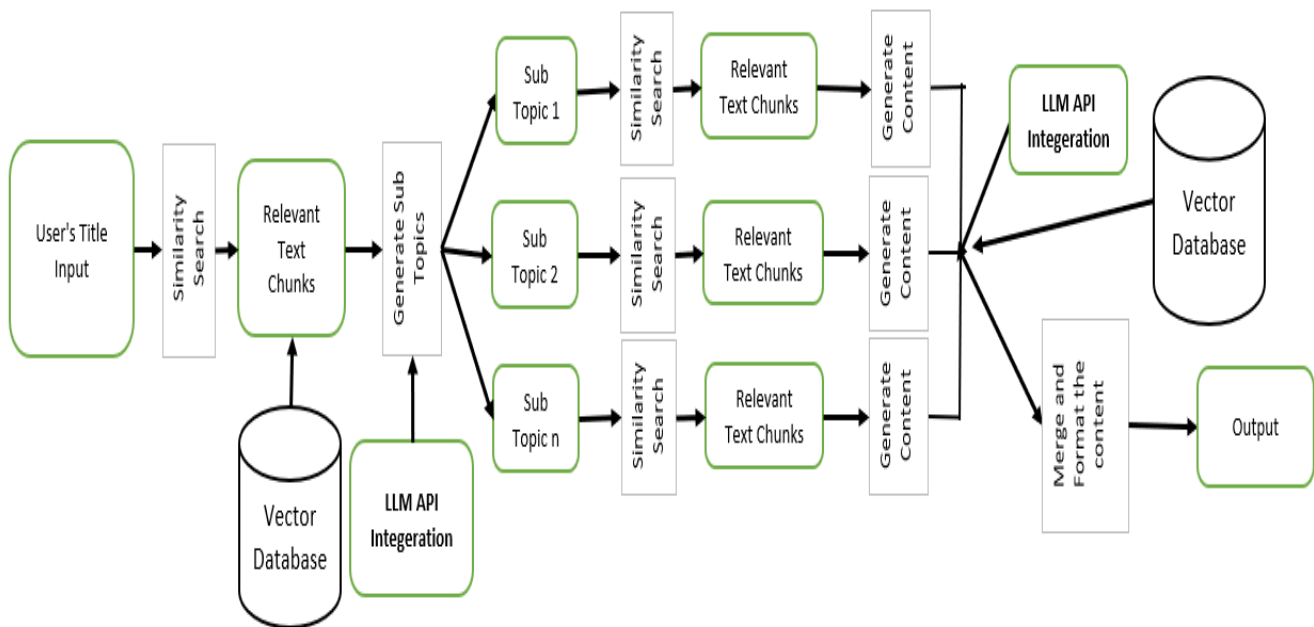


Fig. 2 Content Generation using the Knowledge Base

V. RESULTS AND INTERPRETATIONS

This model along with source validity tries to address the hallucination issues of the LLM model. It also tries to eliminate the possible malfunctions that may arise due to adverse prompt injections. This model will be very much helpful in reducing the cost and time taken for content generation to a larger extent

A. Content Validity

The references provided in the end of each of the blogs will enhance the credibility and provides authenticity to the content. This in turn will improve the commercial viability of the LLM- based content generation.

B. Hallucination

This model tries to include a mechanism like exception handling in the content generation prompts and strictly instructs the model to generate the content based only on the provided knowledge to avoid hallucination. This exception handling mechanism instructs the LLM model to give a user defined message like 'I don't know' if the data sources provided are not sufficient to generate the content.

If more than 25% of the topics return the user defined message then the content generation process is stopped and a message to include relevant source files is displayed to the user.

C. Prompt Injection

This model also follows effective prompt engineering mechanism to avoid prompt injection issues. It restricts the user inputs from directly prompting into the LLM which in turn reduces the chances of prompt overriding and prevents the LLM from malfunctioning in the case of adverse prompts.

VI. IMPLEMENTATION AND DEPLOYMENT

This model can be integrated in the LLM Api based content generation models. These content generation applications can be provided in both cloud-based or a personal storage-based solution. The only requirement will be an internet connectivity and an LLM Api subscription.

A. Source Files Upload UI

This model requires an HTML file which can receive one or more text files (like .docx,.pdf,.txt) and transfer them into the destination folder. The destination folder to store the source files can be either present in a physical memory or a cloud storage (like Google Drive, OneDrive).

B. Vector Database

A vector database needs to be created to store the embeddings of the source files. The api key of the vector database along with its index will be used to connect with the vector database.

C. LLM Api

An LLM Api subscription is required to access the services of the large language model. This research uses two different LLM models. An organization account or personal account based on the user type needs to be created on the business or the service provider.

D. Python Packages

This model requires a list of python libraries to be installed for its functioning. Thus the system which hosts this service should be compatible and storage space to install these libraries. The system should have an ample configuration to run these packages.

VII. CONCLUSION

A. Summary

This research provided content validity to the LLM generated output which will increase the trustworthiness of the LLM based content generation applications and enhanced their reliability.

This research also sheds insights to address the one of the possible hallucination scenario use case and avoiding LLM malfunctions because of prompt injection in case of adverse prompts

B. Limitations

- Though this model improves the reliability by providing source validity, it doesn't provide any solutions to copyright the content generated by the LLM
- This model can be tested and improved only through trial ended error process as there is a lack of a metric to measure the performance of the LLM model.
- The content can be generated only in means of small chunks because of the token limit.
- The content generated may also contain discriminating context in case of adverse topics.

C. Scope of further study

- This research can be expanded further in large language models other than text generation.
- Explore new ways to solve the copyrighting problem associated with the LLM generated outputs
- Further research is also possible in the areas of ethical validity of the AI

VIII. REFERENCES

- [1]. Kamalloo, E., Zhang, X., Ogundepo, O., Thakur, N., Alfonso-Hermelo, D., Rezagholizadeh, M., & Lin, J. (2023, July 6). Evaluating embedding apis for Information Retrieval. arXiv.org.
<https://arxiv.org/abs/2305.06300>
- [2]. Li, Z., Peng, B., He, P., & Yan, X. (2023, August 17). Do you really follow me? adversarial instructions for evaluating the robustness of large language models. arXiv.org. <https://arxiv.org/abs/2308.10819>
- [3]. Oppenlaender, J. (2022). The creativity of text-to-image generation. Proceedings of the 25th International Academic Mindtrek Conference.
<https://doi.org/10.1145/3569219.3569352>
- [4]. Rygl, J., Pomikálek, J., Řehůřek, R., Růžička, M., Novotný, V., & Sojka, P. (2017). Semantic vector encoding and similarity search using fulltext search engines. Proceedings of the 2nd Workshop on Representation Learning for NLP.
<https://doi.org/10.18653/v1/w17-2611>
- [5]. Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D., & Perera, M. (2018). Legal document retrieval using document vector embeddings and deep learning. *Advances in Intelligent Systems and Computing*, 160–175.
https://doi.org/10.1007/978-3-030-01177-2_12
- [6]. Topsakal, O., & Akinci, T. C. (2023). Creating large language model applications utilizing LangChain: A primer on developing LLM Apps Fast. *International Conference on Applied Engineering and Natural Sciences*, 1(1), 1050–1056.
<https://doi.org/10.59287/icaens.1127>
- [7]. Vernikos, G., Bražinskas, A., Adamek, J., Mallinson, J., Severyn, A., & Malmi, E. (2023, May 22). Small language models improve giants by rewriting their outputs. arXiv.org.
<https://arxiv.org/abs/2305.13514>
- [8]. Viswanathan, V., Zhao, C., Bertsch, A., Wu, T., & Neubig, G. (2023, August 23). Prompt2Model: Generating deployable models from natural language instructions. arXiv.org.
<https://arxiv.org/abs/2308.12261>
- [9]. Yan, T., & Xu, T. (2023, May 6). Refining the responses of llms by themselves. arXiv.org.
<https://arxiv.org/abs/2305.04039>
- [10]. Ye, W., Ou, M., Li, T., chen, Y., Ma, X., Yanggong, Y., Wu, S., Fu, J., Chen, G., Wang, H., & Zhao, J. (2023, August 30). Assessing hidden risks of LLMS: An empirical study on robustness, consistency, and credibility. arXiv.org.
<https://arxiv.org/abs/2305.10235>