Building Multimodal Retrieval Augmented Generation (MM-RAG): A general purpose tool for combining Text and Images in Unique and Engaging Manner

Maryam Bandali, Leelavathi Chandrappa,
Lumi Nariyama Hofstad, Ricky Stanley D'Cruze

KTH, DM2586 VT24- Generative AI for Media and Design

March 2024

Abstract

Large Language Models are great at storing a lot of information, but they struggle with rare information and become computationally expensive as they get larger. Models such as RAG have tried to solve this by adding external knowledge, but they only retrieve text. In this project, we aim to build a Multimodal RAG that combines text and images for better understanding. In our architecture, we combine the capabilities of CLIP and OpenAI vision model to perform tasks such as image-text retrieval. In his project we embedded over 2300 images with their filenames, covering various categories like weather data, factory scenes, landscapes, etc. When querying, the application presents search results showing images along with their cosine distances. We submitted the six most similar images to the OpenAI Vision model to generate brief descriptions comprising three lines for each image.

Keywords

Multimodal Retrieval Augmented Generation(MM-RAG), LLM, OPEN AI Vision, CLIP, Weaviate

1 Introduction

In recent years, Generative AI systems have demonstrated their capabilities to generate accurate and relevant content. However, with the continuous expansion of network sizes, models frequently struggle to encode rare or infrequent information. Generating accurate and relevant content poses a significant challenge, particularly when it requires domain-specific knowledge. Some key challenges faced by Large Language Models (**LLMs**) are that they lack real-time updates and may provide outdated information. Additionally, They might exhibit hallucinations, generating details that do not exist in the training data.

One approach that has helped address these limitations is Retrieval Augmented Generation (**RAG**). RAG is a technique that allows generative models to access external knowledge sources, including documents, databases, or web pages, and use them as additional inputs for generating responses. RAG converts the data into vectors and then it is embedded into a vector space, which allows for efficient retrieval of information. By using vector embedding, it becomes possible to represent questions and queries as vectors and find data points in the vector space that are closest in meaning to the query.

Our objective in this project is to develop a Multimodal Retrieval Augmented Generation (**MM-RAG**). MM-RAG extends the capabilities of RAG to incorporate not just text but also images and audio. we utilized the contrastive model for encoding both textual and visual inputs into a vector representation, and the resulting vectors are stored in an open-access database. Additionally, to enrich the multimodal dataset and enhance the content, MM-RAG incorporates an OpenAI vision model for generating descriptive captions for images. This method not only provides additional context but also enriches the overall user experience by enabling more storytelling capabilities. Overall, the MM-RAG represents a significant advancement in multimodal content synthesis, providing a robust toolkit that leverages the combined potential of text and images.

2 Background

In [5], the concept of RAG was introduced for the first time. The RAG was proposed to overcome the limitations of LLMs in knowledge-intensive Natural Language Processing (**NLP**)

tasks. LLMs have the limitations of precisely accessing and manipulating factual knowledge. RAG overcomes this limitation by engaging parametric and non-parametric memories. Parametric memory is the knowledge stored in the neural network's weights obtained during the process of learning by LLM itself. However, non-parametric memory refers to an external knowledge source. The core idea behind RAG is that there is less space in LLMs' prompts to augment additional data. In other words, it is labor-intensive to provide them with all domain specific knowledge whenever we have a question. So, the most relevant knowledge is used to augment the prompts with LLM which then generates the answer. A vanilla RAG consists of three steps: Knowledge retrieval, augmentation and generation. In the first step, the user query is embedded in the same vector space as external knowledge source to retrieve the most similar context. Next, the query and the retrieved context is augmented into a prompt template and then the prompt is passed to the LLM[4]. The workflow of a RAG model is shown in 1

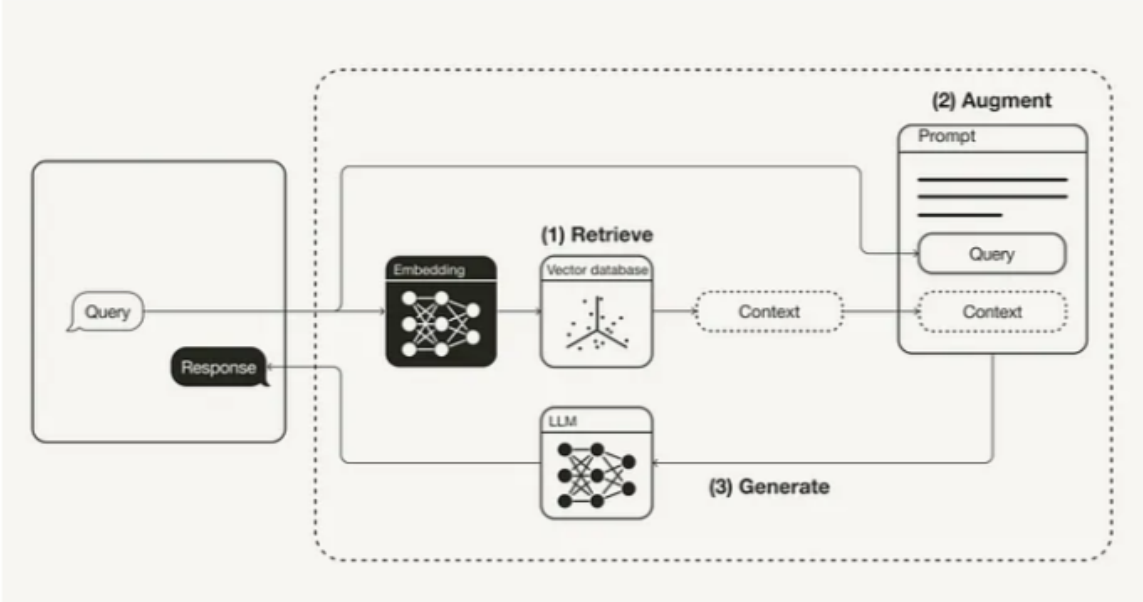

Figure 1: The workflow of a RAG [4]

The idea behind MM-RAG was introduced in [14]. MM-RAG is the RAG that not only includes texts but also images or audios. In fact, the RAG models are only capable of learning from textual data; however, there exists a huge amount of data in other modalities that can be fed into the model. [14] engages a non-parametric multi modal memory in which both

textual and visual data is stored. As a result, we can embed images in the vector space, the same way we embed the text, and then retrieve and pass them to the LLM. LLM read the relevant textual and visual information that is augmented into a prompt before generating a response. Thus, there is no need to fine tune the model every time the information is updated. Besides, the LLMs are supercharged with data of visual formats[6].

In a MM-RAG application, when a question is asked, it is converted to a vector semantically near the information that contains the answer in the vector space. Therefore, the texts and images with similar meanings can be realized by the model. As we mentioned before, in a RAG application first, all textual data is embedded in the vector space using an embedding model. Subsequently, the query is embedded using the same model and retrieval of the information takes place based on the computation of vectors in the vector space. In a MM-RAG application, the same procedure is performed with multi modal data which means even images can be converted into the vectors. As a result, the user can query textual contents to retrieve visual contents or query visual contents to get semantically similar visual contents[6]. In this case, the difference with the workflow of a vanilla RAG is as follows: the user can query the model in both text and/or image modalities. The retrieval context can be both text and/or images. In augmentation phase, the prompt can be synthesized with text and/or images and in generation step, the response to the user query can be returned in both image and/or text formats.

As mentioned above, we need a vector data base to mathematically represent and infer similarities for the data objects in a vector space. In a vector data base, the unstructured data such as images and audios are encoded based on an embedding models in high dimensional vectors called vector embedding. The vector embeddings are grouped based on the semantic similarities in the data objects[3], makes it efficient for managing data with different formats. Thus, an interesting property of the vector embedding is that if we vectorize a question, it will end up in the vector space near the data that contains the answer. We use the open-source **Weaviate** vector data base to this end[13]. It represents an open-source vector database carefully crafted to accommodate the archival of diverse data objects alongside the vector embedding produced by machine learning models of choice. Figure 2 shows a joint embedding space of a multimodal model that understands both text and images. Notably, its architecture exhibits a capacity for seamless scalability, adeptly managing vast repositories

comprising billions of data objects. Serving as a cloud-native entity, Weaviate operates as a modular, real-time vector search engine, strategically engineered to augment the scalability and operational efficiency of machine learning frameworks. Users interface with Weaviate through multiple framework, including GraphQL, REST, and a spectrum of language-specific clients, thereby ensuring versatile accessibility and integration capabilities[12].

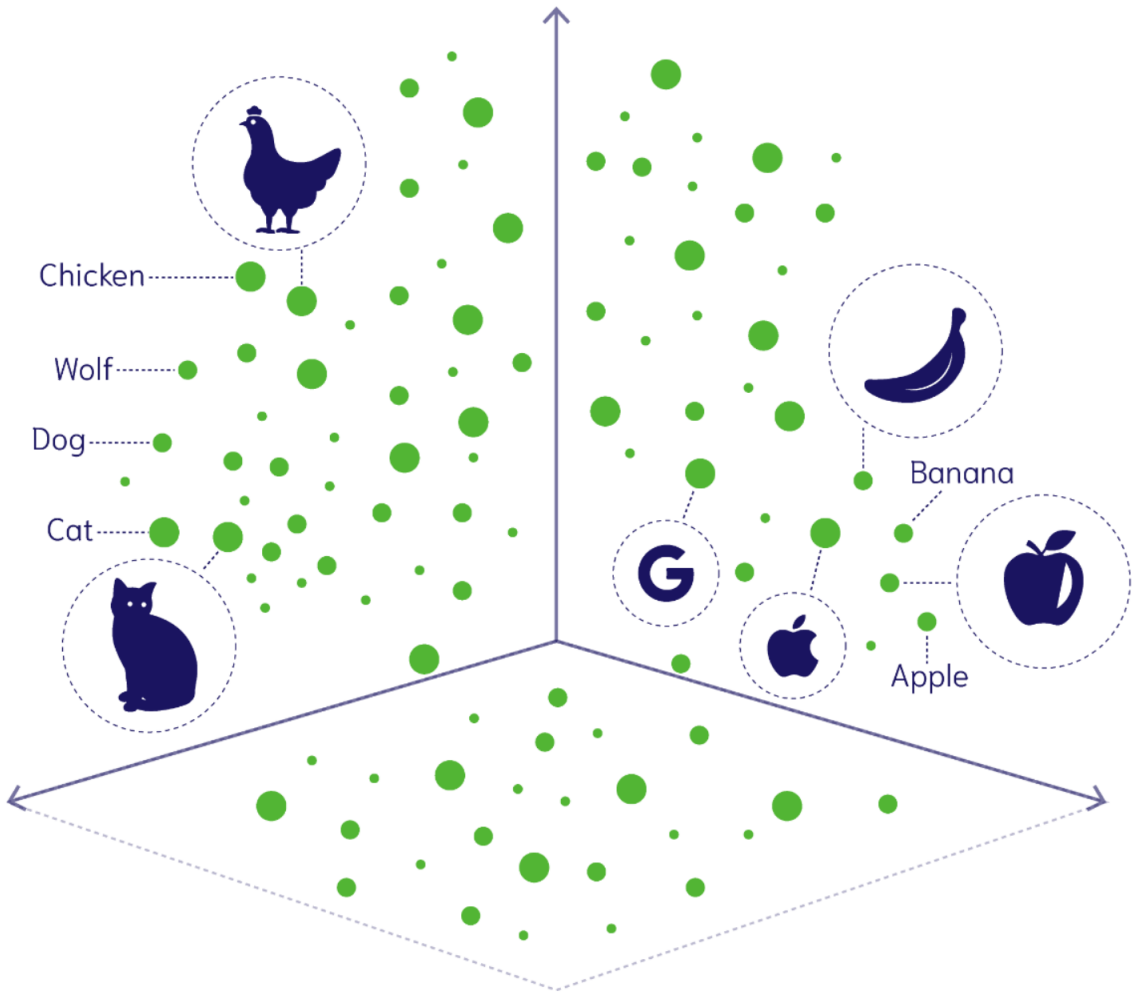

Figure 2: vector embedding of multimodal model[13]

We use Contrastive Language–Image Pre-training (CLIP) neural network model to encode images into vector embeddings. The learning of the visual concepts by CLIP happens based on natural language supervision[8]. The CLIP model stands out from other visual encoders due to its superior generalization performance, which is why we have decided to use it in

our case. Traditional visual encoders are often trained on manually labeled datasets, limiting their ability to generalize to unseen objects and concepts. In contrast, CLIP is trained on a massive dataset of 400 million noisy image-text pairs crawled from the internet, allowing it to learn visual concepts with language supervision in a scalable manner[10].

In this project, we aim to build a MM-RAG application and several multi modal use cases. The data set that we use is a set of images. The goal is that with this model, we can do jobs such as image to image, text to image, or image to text retrieval.

3 Methodology

Figure 3 features a schematic representation of an architecture for managing and interacting with a vector database, incorporating a semantic search and a generative AI component. The architecture is delineated in three principal steps, each with dedicated flows and technology implementation. There are different stages of a RAG application:

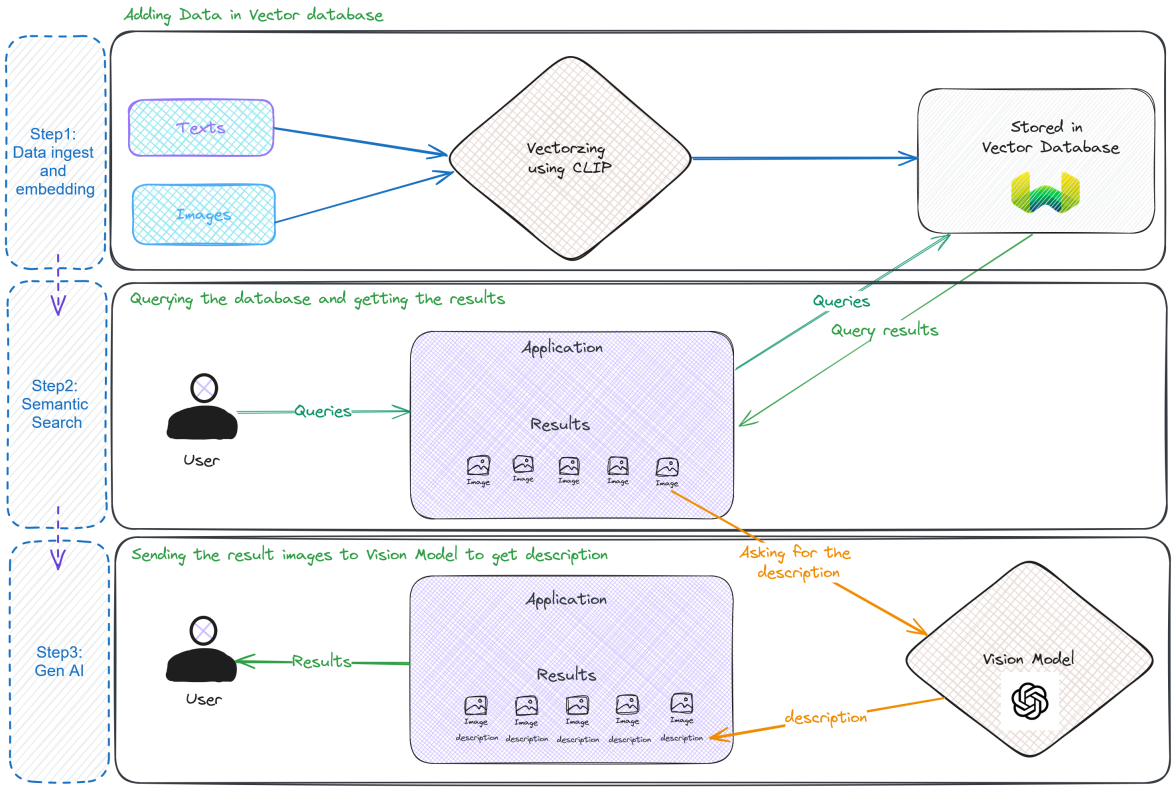

Figure 3: The workflow and architecture of the application

Step 1: Data Ingest and Embedding In the initial step, two types of data—texts and images—are ingested into the system. Subsequently, these data are vectored using CLIP in particular "sentence-transformers/clip-ViT-B-32-multilingual-v1". This model is designed to accommodate multiple languages. It facilitates the alignment of text in over 50 languages and images into a unified dense vector space, thereby ensuring proximity between images and corresponding textual descriptions. Its applications encompass image retrieval, wherein users sift through extensive image databases, as well as multi-lingual zero-shot image classification, where image categorizations are informed by textual labels.[9] Following the embedding process of both textual and visual elements, the resultant vectors are archived within Weaviate.

Step 2: Semantic Search During the second step, a user interacts with the system through a query. The user inputs a query, which can be a text or an image, to an application, which then searches the vector database for relevant results. The results are returned to the application, and the user is presented with images that semantically nearest to the query. In our analysis, we employed the cosine distance [11] metric among the available methods for determining the similarity between two vectors. Cosine similarity assesses the angle formed between two vectors within a multi-dimensional space, operating on the premise that vectors pointing in similar directions denote similarity. Widely applied in NLP, this metric gauges document similarity irrespective of their magnitudes[1].

Step 3: Generative AI In the final step, the user requests further information about the returned images. The application then forwards these images to a OPEN AI vision model (gpt-4-vision-preview), which is currently available to developers access via chat completion API[2]. This model is adept at generating descriptions for images, utilizing advanced AI techniques to understand and articulate the content within an image. The descriptions generated by the vision model are then sent back to the application and ultimately provided to the user, adding an interpretative textual layer to the visual data.

Overall, figure 3 diagram exemplifies a sophisticated architecture utilizing contemporary technologies such as CLIP and OPEN AI vision models to provide a seamless user experience for searching and obtaining insights from a database of vectorized texts and images. It underscores the integration of modern AI capabilities in enhancing data accessibility and interpretability in complex systems.

All together we have collected 2300+ royalty free images. We planned to do on the

manufacturing domain use case in this project but collecting data for a particular domain was difficult for us. So we have gathered data from many different sources, like, kaggle , freepik etc. Although we have shown this app for general purpose use case but any industry can reuse the application according to their need.

4 Results

In the initial stage illustrated in Figure 3, approximately 2300+ images were subjected to embedding along with their respective filenames. These images encompass various categories such as weather data, factory work floor scenes, natural landscapes, flowers, food items, among others. A selection of sample images is depicted in Figure 4. Additionally, as demonstrated in Figure 5, the Python application developed by our team allows for the retrieval of images through free-text queries in over 50+ languages, as well as through image-based searches.

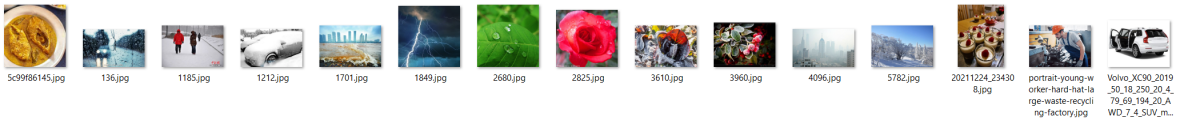

Figure 4: sample text and images embedding and stored in Weaviate

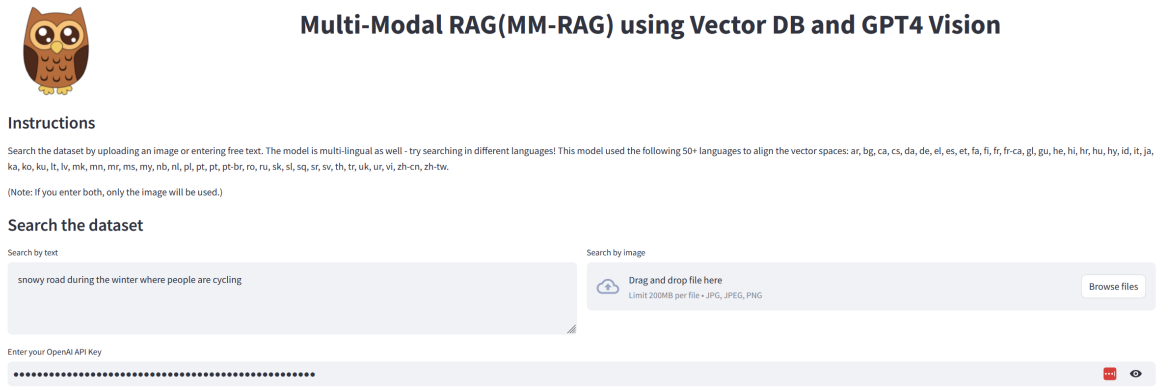

Figure 5: Python APP for the search with text or image

In Figure 6, the search results are presented, featuring images accompanied by their respective distances. Here, the term "distance" refers to the cosine distance between two vectors.

The cosine distance ranges from 0 to 2, with a value of 0 denoting identical vectors, 1 indicating no correlation, and 2 representing complete dissimilarity. The resulted images exhibit distances of 0.666, followed by 0.679 and 0.680, respectively. These values closely correspond to our query regarding "snowy road during the winter where people are cycling," as depicted in Figure 5.

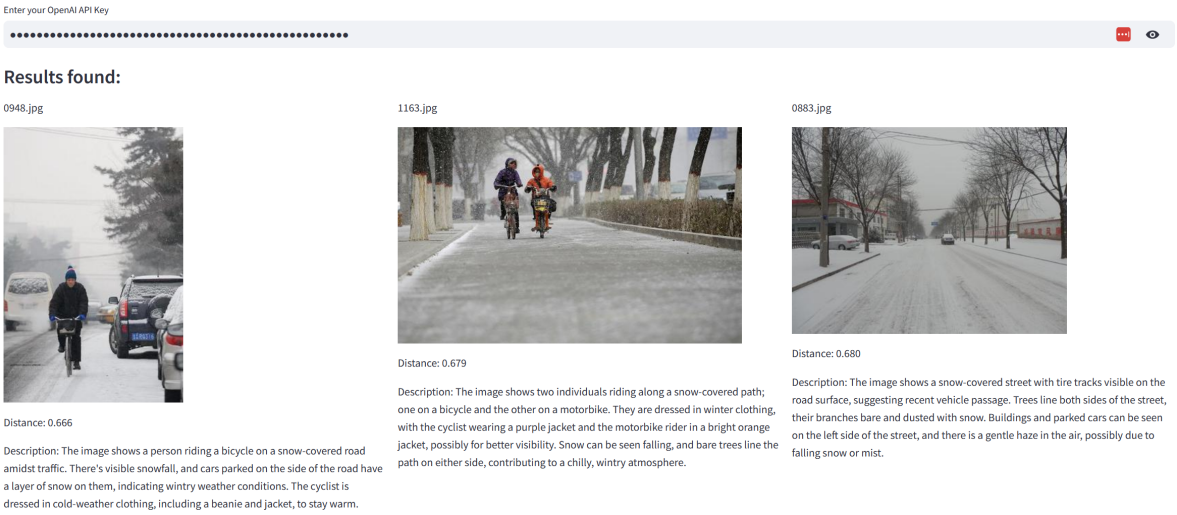

Figure 6: Query result of similarity search and description from OPEN AI Vision Model

Subsequently, the generative AI aspect is introduced, wherein the images retrieved from the search results are forwarded to an OPEN AI vision model. This model is then tasked with providing a descriptive analysis of the depicted image, as delineated in Figure 7.

5 Discussion

The performance of the CLIP ViT model is assessed when it hasn't undergone specific training for a particular task, a scenario known as the "zero-shot" case, which is the case we're focusing on. In the study of [10], the researcher found that when it comes to spotting and understanding things in pictures, the CLIP ViT model doesn't do as well as other models. This is because it struggles to pinpoint where things are in the pictures. They measured this using a number called Average Precision (AP), and found that CLIP ViT's AP score was very low, indicating it had trouble spotting things compared to other models. On the other hand CLIP is good at some tasks without training, like recognizing objects, which we have done in our case using

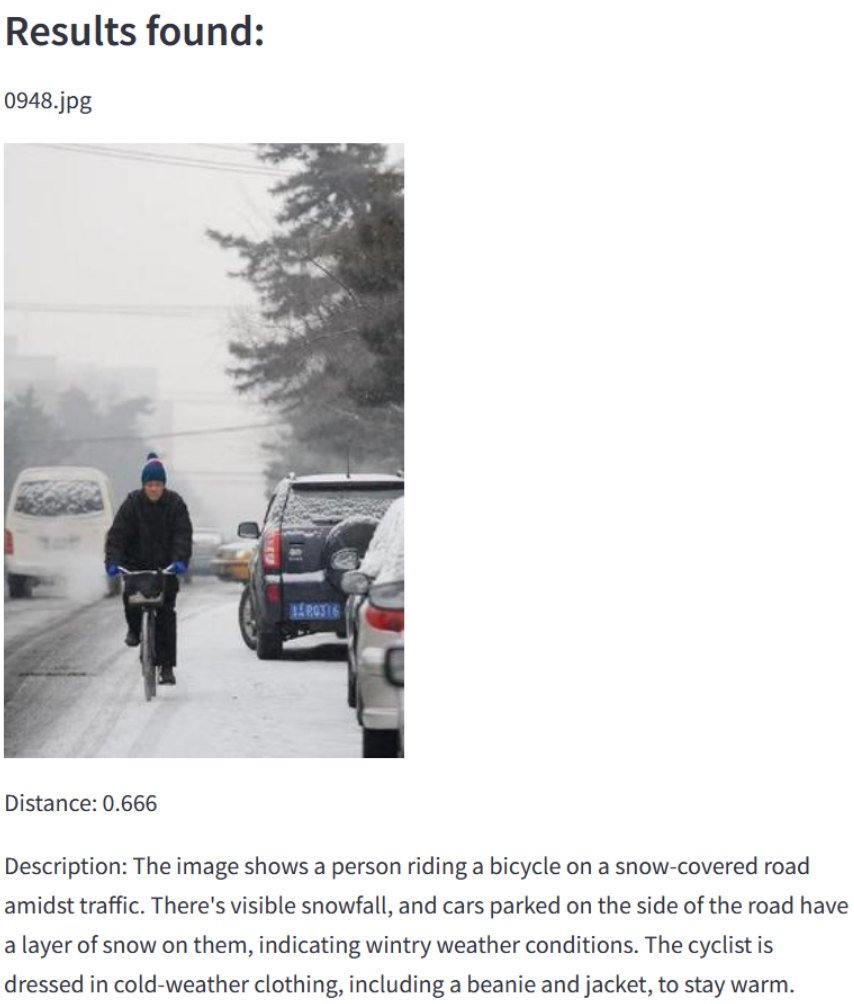

Figure 7: Description from OPEN AI Vision Model of the first image

vector similarity search. It seems to have a harder time with tasks that involve both seeing and understanding, such as answering questions about what's in a picture. To make CLIP ViT better at these kinds of tasks, more research is needed to figure out how to improve its abilities when it hasn't been trained for a specific task.

One of the limitations of our study is the absence of empirical evidence regarding the effectiveness and accuracy of the descriptions provided by the OPEN AI Vision model for the resultant images. Future research endeavors could address this limitation by employing evaluation metrics such as BLEU (Bilingual Evaluation Understudy) score. BLEU score quantifies the quality of predicted text, termed as the candidate, in comparison to a predefined

set of reference texts[7], or by utilizing alternative evaluation methodologies. This approach would facilitate a more comprehensive assessment of the performance and reliability of the AI-driven image description generation process.

In this project, our focus did not entail innovating algorithms or devising novel models to address the case at hand. Rather, our approach involved leveraging existing models and technology to streamline processes and enhance efficiency and accuracy. Presently, our solution caters to general-purpose applications; however, it holds potential for adaptation and utilization within specialized domains by other practitioners in the future. An illustrative scenario could involve the detection of machine failures within manufacturing industries through the utilization of image datasets. In this context, operators could capture images of a specific machine malfunction and upload them into a system designed to identify similar instances. Subsequently, the system would use a vision model to analyze the uploaded images and generate textual descriptions, thus facilitating rapid diagnosis and problem solving of machinery issues. Similarly, in the telecommunication domain, the application can be used by the field engineers to help diagnose the hardware failures of the high radio base stations.

6 Conclusion

In this project, we propose to implement an application by using a Multimodal RAG for general purpose use cases. We explore "How can a MM-RAG system be tailored to assist users ?" through a combination of textual instructions and images.

Our application is built by leveraging existing models and technology. A MM-RAG system involves finding textual and/ or visual data that semantically match in the augmentation phase, and then passing the data to a LLM such as GPTV4 to extract the answer which is synthesized in the form of text/image as the response to the user query in the generation step. This process helps optimize the output of LLMs by packing relevant information into a prompt before its generation without using a fine-tuning technique.

However, there is still absence of empirical evidence to evaluate the effectiveness and accuracy of the descriptions provided by the OPEN AI Vision model for the resultant images. It could be a potential future research direction as well as a key consideration when building such a system.

7 Abbreviations

In this section, we introduce a list of abbreviations that are used throughout the article. Table 1 contains all the abbreviations.

Abbreviation	Definition
MML	Mutimodal Machine Learing
RAG	Retrieval Augmented Generation
LLM	Large Language Model
NLP	Natural Language Processing
CLIP ViT	Contrastive Language–Image Pre-training Vision Transformer
MM-RAG	Mutimodal Retrieval Augmented Generation
GPTV4	Generative pre-trained transformers Version 4

Table 1: Abbreviation

References

[1] Erika Cardenas. Distance metrics in vector search. https://weaviate.io, 2023. Accessed: Feb 23, 2024.

[2] Open AI Documentation. Vision. https://platform.openai.com, 2023. Accessed: Feb 23, 2024.

[3] IBM. What is a vector data base? https://ibm.com, 2023.

[4] Leonie Monigatti. Retrieval-Augmented Generation (RAG): From Theory to LangChain Implementation. https://towardsdatascience.com, 2023.

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[6] LIamaIndex. Multi-modal Retrieval Augmented Generation with LlamaIndex. https://www.youtube.com/@LlamaIndex, 2023.

[7] Priyanka. Evaluation metrics in natural language processing — bleu. https://medium.com, 2022. Accessed: Feb 28, 2024.

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[9] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[10] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks?, 2021.

[11] Milana Shkhanukova. Cosine distance and cosine similarity. https://medium.com, 2023. Accessed: Feb 28, 2024.

[12] Vishnu Sivan. Experimenting with vector databases: Chromadb, pinecone, weaviate and pgvector. https://medium.com, 2023. Accessed: Feb 23, 2024.

[13] Weaviate, B.V. Built with Docusaurus. Weaviate Official Documentation. https://weaviate.io, 2023.

[14] Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Rich James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Retrieval-augmented multi-modal language modeling. *arXiv preprint arXiv:2211.12561*, 2022.