2024 AI CHALLENGE: SHOP SAVVY BOT

TEAM MEMBERS:

- FARHAN: k214808@nu.edu.pk
- NOMAN: k225320@nu.edu.pk/noman_ejaz96@hotmail.com
- HAREEM: k225438@nu.edu.pk/harim36148@gmail.com

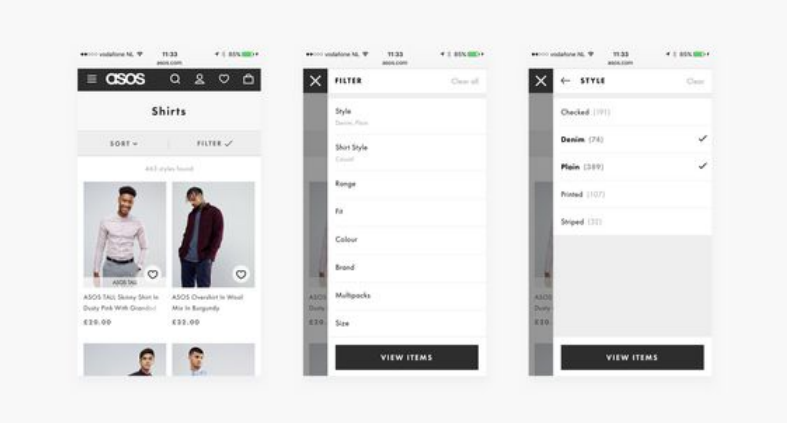
Introduction:

In the realm of E-commerce, consumers are inundated with choices, leading to decision fatigue and suboptimal purchasing decisions. Despite the availability of tons of products, users often struggle to find items that align with their preferences, needs, and budget.

PROBLEM:

Traditional approaches of building front end applications and presenting product details and search buttons for filtering through UI fall short in

- In providing Dynamic Filtering Mechanisms
- In Providing personalized recommendations.
- In Understanding nuanced user preferences in different languages
    - In the cases where understanding of multiple languages is required Multilingual apps are built today but this increases the development time and overall project cost!
- In providing video & voice based searches.
    - If the user likes a product displayed in a youtube video which he recently watched. Currently there is no straightforward method for him to instruct existing applications either web based or mobile based to search their databases for that specific product featured in the video.
- In the Pakistani E-commerce market voice chat based search is currently missing.



SOLUTION:

There is a pressing need for an advanced shopping assistant powered by Large Language Models (LLM) technology, to revolutionize the e-commerce landscape of Pakistan that addresses the following issues!

- Simplified Filtering Mechanisms
    - Instead of providing filter buttons for each of the possible options to search for the products. The user can simply present their queries in natural language and the underlying LLMs will use it to retrieve appropriate products from the database.
- In understanding user queries across different languages
    - LLMs can implicitly provide simple consistent front-end UI across different languages without any additional efforts
- Providing Video Based Searches
    - Well this is the main highlight of our project and we have tested it successfully! If the user likes a product displayed in a youtube video then he can simply instruct LLMs in natural language to search their databases for that specific product featured in the video.
- Voice based searches.
    - Using the whisper OPEN AI model we can easily transcribe user voice queries and use them to search the Databases to find out relevant products.

Target Community & Potential Reach:

- **Online Shoppers:**
    - Individuals who frequently engage in online shopping across various e-commerce platforms
- **Consumers seeking personalized recommendations:**
    - Customers who value personalized product recommendations based on their unique preferences, rather than generic suggestions.
- **E-commerce platform developers:**
    - Companies and developers in the e-commerce industry interested in enhancing their platforms with innovative technologies to improve user experience and increase customer satisfaction
- **Marketing professionals:**
    - Individuals responsible for marketing strategies and customer engagement within e-commerce companies, seeking tools and technologies to better understand and cater to consumer preferences.

Methodologies:

There are two approaches mentioned below to integrate LLMs in E-commerce product search! Both have their own pros and cons. We have come up with a third approach to combine benefits of both of them and reduce the impact of cons.

1. Transforming User Queries to Executable Code

The first commonly used approach is to simply convert user queries into an executable code in SQL or Cypher language.

a. Pros: Good in Result Aggregation and performing Mathematical Operations
b. Cons: Bad at handling synonyms, user spelling mistakes, or a query containing vocabulary from two or more than two languages!

For instance consider the following Example:

User Query: *Show me the price of Paneer*

**Here the user meant Cheese from the word Paneer.*

Problem: if we translate the user query to sql, we will end up with the following result, which will return zero results when executed on a database. Because the product is saved with an English name called “Cheese”

SELECT price FROM product WHERE name=”Paneer”

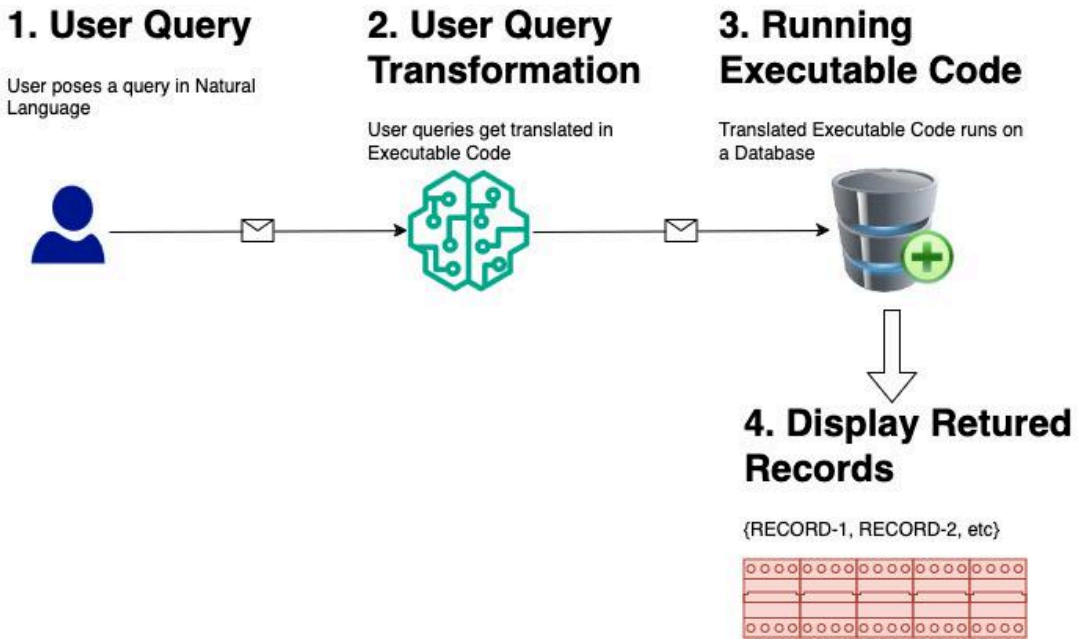


Figure: The diagram shows the traditional method of transforming user given text SQL base queries to retrieve relevant products.

2. RAG to Retrieve Matching Records from the Database

The second popular approach now a days is to Retrieval Augmented Generation (RAG) to fetch most suitable records from the database

a. Pros: Good in processing synonyms, user spelling mistakes, or a query containing vocabulary from two or more than two languages.

b. Cons: Bad at Result Aggregation and Performing Mathematical Operations

For instance consider the following Example:

User Query: *Show me total sale amount of my store in the month of January and February*

Problem: *If we directly apply the RAG to the records of the products table in the database. It will successfully be able to retrieve sales records from the database but struggles in satisfying filters of the provided months and providing the SUM operator over them!*
In our experiment, we have observed that it ends up with too many False Positives.

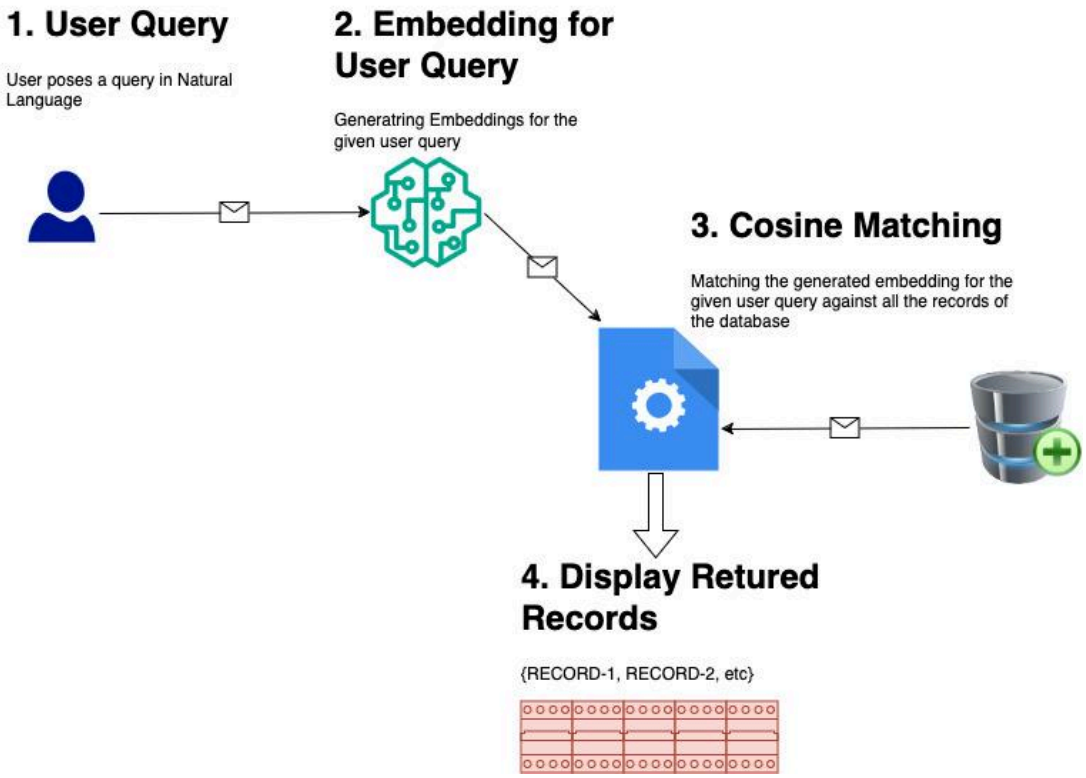


Figure: The diagram shows the architecture of implementing basic RAG architecture to retrieve relevant products based on user given queries.

3. RAG to Retrieve the Matching Records and Transforming User Queries into Executable Code such as (SQL/Cypher)

Our Proposed Methodology: In our design we have combined the best of the previously mentioned approaches!
In our design, we first use the RAG approach to retrieve the matching records from the database and then plug in the retrieved records as context to generate a suitable executable query/code! Which then can be utilized to find the relevant products from the database.

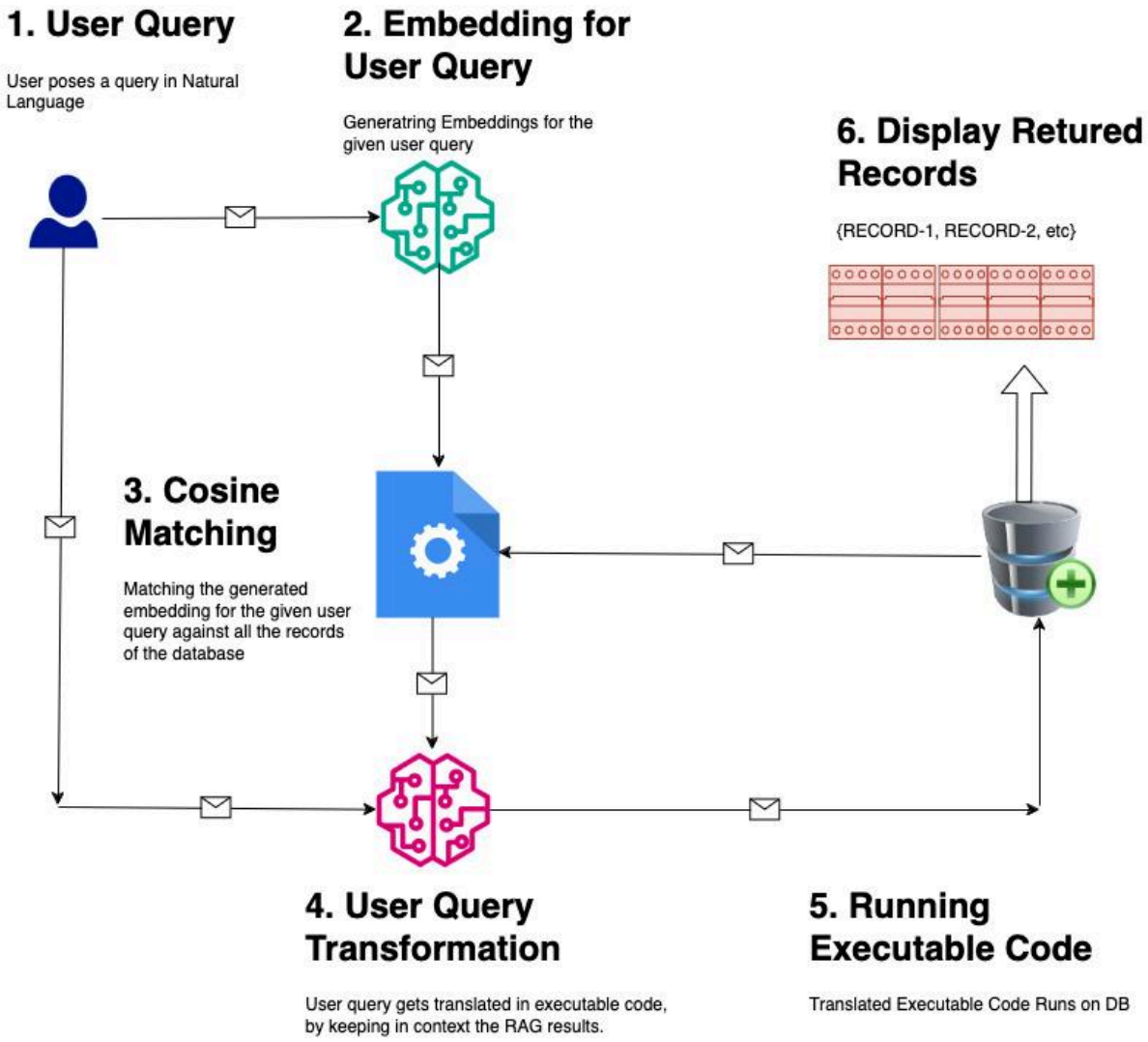


Figure: The diagram shows a birds eye view of our approach.

RAG: Implementation of Advance RAG:

1. Generate multiple variants of query in English.
2. Calculation of the most optimum value “K” for a given user query to execute RAG based search.

RAG: Generate Multiple Variants of User Query in English:

We have observed that most of the LLMs, especially the open source one, are more tuned with the English language. Additionally, most often the data stored in the databases are also in the English Language! Plus, users often do not fully express their desired intent in the query. Therefore we have used MultiQuery Retrieval in order to implement our RAG base approach and improve our results!

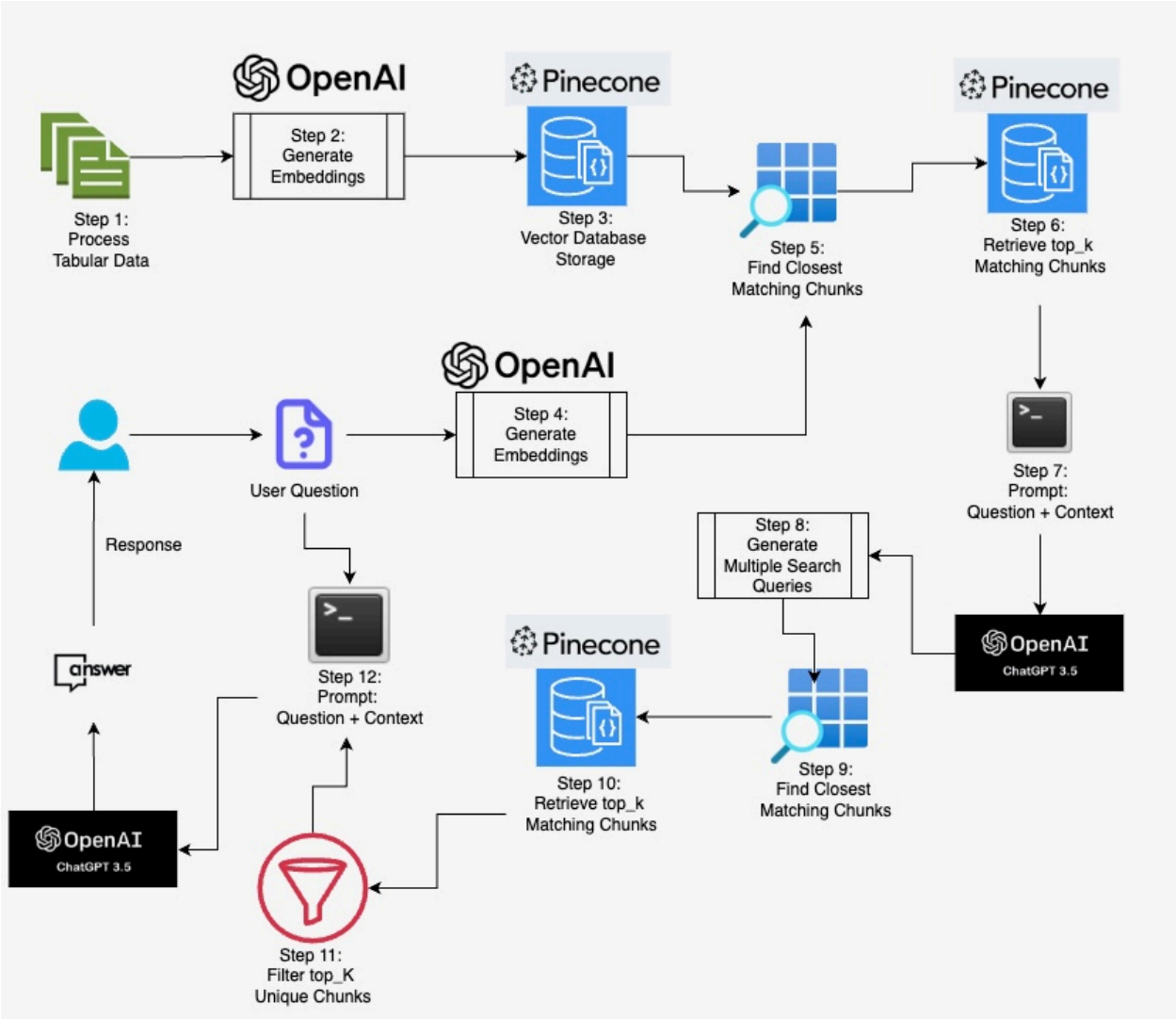


Figure: The diagram illustrates the details of our experiment of closed domain-tabular Q/A using MultiQuery Retrieval.

RAG: Calculation of the Most Optimum Value of "K"' (main novelty of our approach):

Well the main novelty of approach is the improvements that we included in the implementation of RAG with dynamic "K":

1. Calculation of the "K" value dynamically based on the given query and the data stored in the database.
    a. We have used clustering oriented techniques that are silhouette & elbow method to determine the appropriate value of "K" based on the given user query and the data stored in the database

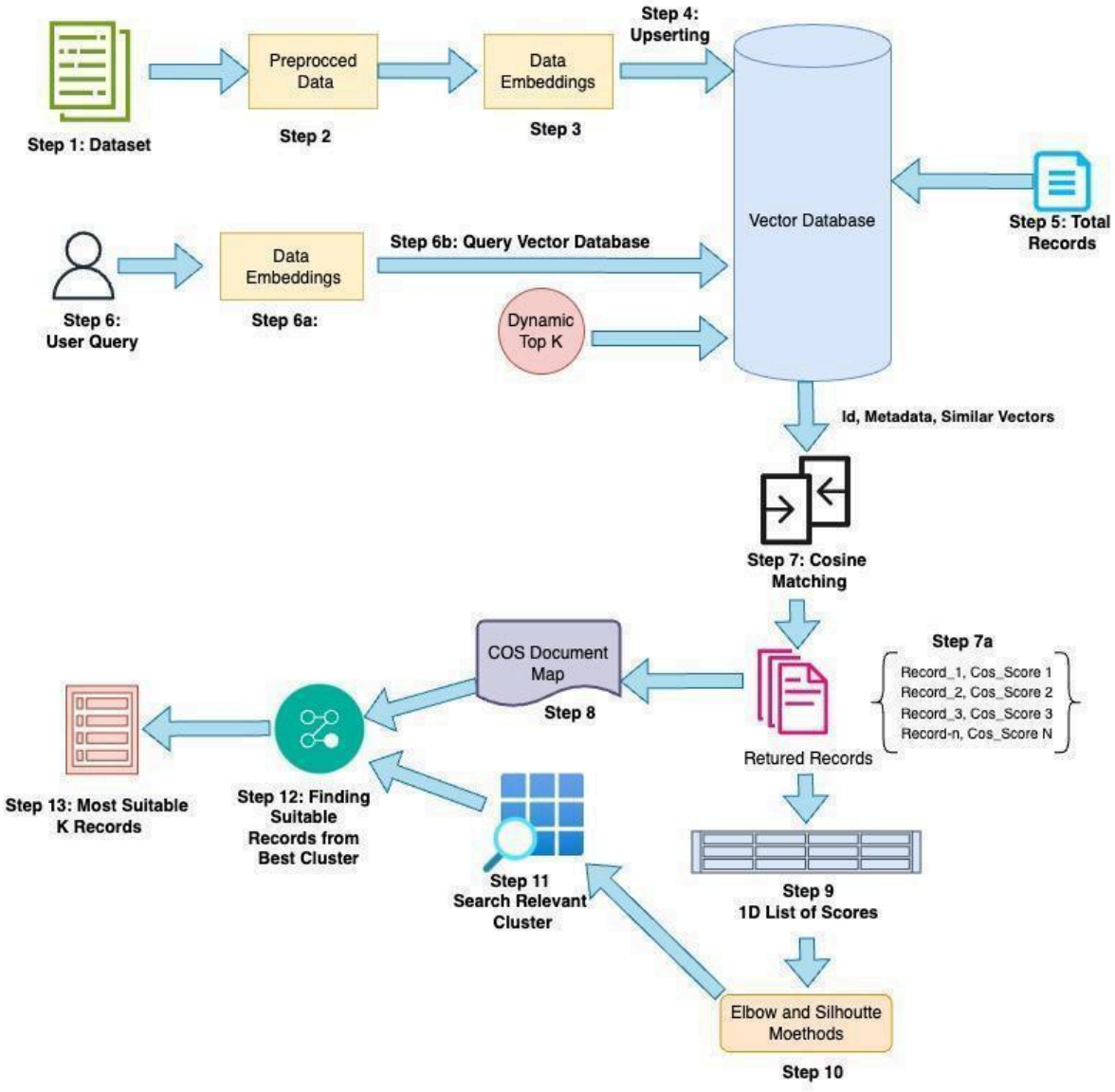


Figure: The above figure shows the retro architecture and the methodology of finding effective K value using similarity metric.

Connecting the LLMs with GraphDatabase:

In order to build a shopping assistant, we not only need to display the results of the search to the user but also recommend other products from the store which may interest him! There are two widely known techniques for product recommendation that are Content Based Filtering and Collaborative Filtering! Both of them are widely supported by many graph Database
In our solution we have used Neo4j, a graph Database with Content Based Filtering to recommend suitable products to the user based on related content available to his/her previous purchases.

Product Recommender = LLMs + GraphDatabase = Salesman!!

Technologies Used:

- Neo4j graph Database!
    - We choose Graph Database because of its inherent flexibility in consuming and storing data records from multiple sources!
    - Another challenge arises in product recommendation! Both collaborative and Content Based Filtering can be easily implemented via Neo4j graph database
- Ollama + Hugging-Face Models + Open AI With LANGHAIN
    - Well our app supports the following models! Available on the Ollama/Hugging-Face/Open AI. In the code we just need to specify which model we want to use

        - Embedding Model:
            - Mxbai-embed-large:latest (Available on OLLAMA)
            - Text-embedding-ada-002 (Available on Open AI)
        - LLMs
            - LLAMA-3 (Available on OLLAMA)
            - CodeGemma (Available on OLLAMA)
            - GPT-3.5 (Available on Open AI)
            - CLIP (Available on Open AI + Hugging Face Transformer models)
    - Streamlit app
        - To build the Front End

Evaluation Metrics:

We have just evaluated the quality of the returned outputs by our approach on the FeTaQA dataset with the actual records that need to be returned against a given query with the Recall/Precision/Accuracy/F1-Score/Specificity measures.

Value of K	Technique	Recall	Precision	Accuracy	F1-Score	Specificity
k=3	Baseline RETRO	32.12	65.95	66.96	41.79	87.91
k=3	RETRO + MQR	48.93	58.6	66.09	50.58	74.35
k=5	Baseline RETRO	45.1	57.25	66.37	48.58	75.63
k=5	RETRO + MQR	62.17	51.19	62.62	53.5	57.11
k=7	Baseline RETRO	55.57	51.67	64.28	51.44	62.46
k=7	RETRO + MQR	71.23	47.36	58.81	54.43	43.38

Table: The table shows the comparison of the scores of finding suitable records for a given query with basic RAG and hard coded value of "K" with our approach of including Multi-Query-Retrieval (MQR).

Method	Recall	Precision	Accuracy	F1-Score	Specificity
Baseline RAG with k=3	32.12	65.92	66.68	41.78	87.92
Baseline RAG with k=5	45.16	57.23	66.36	48.55	75.63
Baseline RAG with k=10	67.74	46.88	59.77	53.27	44.62
RAG with Elbow Method	40.35	**69.33**	65.10	45.18	79.85
RAG with Silhouette Method	**48.76**	65.77	60.43	46.44	68.16

Table: The table shows the comparison of the scores of finding suitable records for a given query with basic RAG and hard coded value of "K" with our approach of determining the suitable value of "K" at the runtime with clustering based methods.

If the product were to become successful, how do you see it evolving in the future?

Well if our idea of shopping assistant bots becomes successful, Then I think there is an evolution of big ecommerce Apps like Daraz / Ali Express or even Amazon

.

Because the idea doesn't stop at the chat interface it can easily scale to verbal telephonic type conversations with machines. Just like nowadays a normal human does with the product support team! Which requires no download of apps from the play store no clicking UI buttons Or filters. Just a simple phone call to order the thing that the user needs based on his region and personal preferences.