Task 2 - From data to AI
AIBeSa

Authors: Beloslava Malakova (TU/e: 1923404), Alicja Gwiazda(TU/e:2017830), Stanimir DImitrov(TU/e: 1932217), Aleksandra Nowińska (TU/e: 2008580)

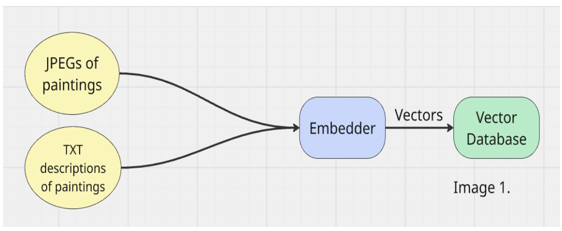
1. Collect data:
    - Download data from a preferred source (clarifications in class or via email/canvas to the lecturer)
    - Data exploration:
        - Volume?
        - Velocity?
        - Variety?
        - Veracity?
2. Steps to train your (supervised) system:
    - Make sure data is aligned and you have a source (input) and a target (output, labels) side.
    - Preprocess the data !
    - Split the data into train, test and dev sets !
    - Clean the data by removing information you don't think is relevant, by removing or filling in missing data, etc. !
    - Train the system
    - Evaluate the system !
3. Form the intelligence - if necessary, combine trained models (each of which solves one part of the task) to cover the complete intelligence
4. Prepare a maximum 1 page A4 outlining the above.
    - Writing things down will allow you to see if something is wrong and if something else is needed.
    - Have three sections:
        - Data
        - Model (statistics about the mode, e.g. evaluation metrics, training time, inference time, number of epochs, etc.)
        - Software used for the development, including software for data processing and preparation.

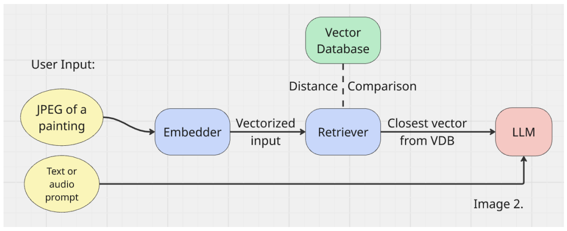
Data

For our project, we work with a subset of existing open-source art datasets from Kaggle, focusing only on information relevant to one selected exhibition. The datasets are partially sampled and curated to fit the needs of our stakeholder (a curator). This means filtering the data down to include only the artist name and a corresponding image of the artwork, which represent the minimum requirements for building an AI system that can support an exhibition setting. We used datasets such as *Best Artworks of All Time*, which contain thousands of high-quality images. From these, we create a reduced dataset, suitable for one exhibition scenario. The **volume** of the full datasets contains over 8,000 images, but for our task, we curate a smaller sample (dozens to a few hundred images). The **variety** of the data includes multiple modalities: tabular metadata (artist names, time periods, genres) and visual data (artwork images). Regarding **velocity**, the data is static rather than streaming, making it suitable for one-time curation and offline model training. And for **veracity**, no missing data or corrupted files.

Model (Pipeline)

Our project's core idea is to leverage foundational models and RAG to tackle our problem. We first remove all unnecessary data from the dataset like "number of images per artist". We then use a subset of the large dataset to create a vector database in which each embedding consists of an image and a small text description. This image preprocessing is done by an multimodal embedding model like OpenAI's CLIP. Then the newly created vector database symbolizes one art gallery exhibition. You can see the process of embedding visualized in Image 1.





In Image 2 you can see the following process visualised:

After the vector database (consisting of all the art in the exhibition) is created we are creating a LangChain RAG pipeline in which the customer can input an image and ask a question about it (example can be "From which period is this artist"). Then this image is vectorized and a retriever finds the closest image to it in the vector database. The information related to the image (author, era, small description) is then put in a json file. This json file is then sent to an LLM model with a specific system prompt that is instructed to answer the user's question in a tour guide style. The output of the model (information about the artwork) is then presented to the user.

Software Used for Development (Python is the main programming language used)

Kaggle - Used for gathering the dataset ; ***Pandas*** - Used for cleaning the dataset of unnecessary information ; ***Weaviate*** - Vector database ; ***Embedder and Retriever*** - both are integrated in Weaviate for multimodality tasks (image + text) ; ***Gemini API*** - Using gemini models to generate a cohesive tour guide style answer ; ***Streamlit/Gradio*** - Creating UI for the app ; ***LangChain*** - Orchestrating the RAG pipeline and the connection to the UI ***VS Code + Github*** - Programing environment and version control.

References

a. https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time/data
b. https://www.kaggle.com/datasets/steubk/wikiart