My summer internship at Trinity College Dublin involved the usage of LLMs to develop a programming assistant for novice programmers – one that assists in the process of learning rather than providing solutions to its users. Post the Exploratory Data Analysis (EDA), a model that uses a localised LLM along with embedding content provided to it was chosen as the final approach.

Embeddings are content that are provided to an LLM along with the user's query in order to provide for contextual knowledge that either behaves as the source or boosts the already existent knowledge in that regard. These have multiple parameters and are stored in the form of multidimensional vectors, ideally within a vector database. I chose to visualise all the vectors stored within the vector database to obtain a visual reference of if / how different parts within the database were linked with each other, with the final goal of identifying independent clusters within the data. This would further assist in the process of linking various pieces of content stored in the database alongside making provisions for search functionalities to enhance the outcome of the visualisation.

1 Dataset

The dataset (MKB, n.d.) chosen for this visualisation has been created by scraping tutorial content from the publicly available content for Pytch – an ongoing research project at TCD aiming to bridge the gap between Scratch and Python.

Text based content has been scraped from the available game based tutorials on the Pytch (TCD, n.d.) webapp. This has been done by building and setting up a development environment for the Pytch webapp, which has been designed to populate two empty git repos with all the tutorial content (present in markdown format) as and when the webapp loads. The text based content also contain code blocks, which have been indicated by their git commit hashes. A script has been designed to check out the repository at each of the git commit hashes and to load the code present within itself back to the text content. Finally, a text file containing all the text and code blocks was generated as the complete raw dataset for this visualisation. The content from this text file was pre-processed (discussed in further sections) and the resultant multidimensional vector embeddings were treated as the source for the visualisation post dimensionality reduction.

This dataset consists of data containing item and position data types. Each vector embedding contains multidimensional features along with text based tutorial content, thus representing the position (spatial) and item data types respectively. Based on the data types, the dataset can be mapped to a Geometry based dataset type.

Even though each of the vectors consists of numeric coordinates / features and text based content, no operations can be performed on them. The data contained within this dataset belongs to the categorical / qualitative data attribute type, as one of the main goals of this visualisation is to identify the presence of clusters within the dataset.

The data contained within this dataset consists of about 250 multidimensional vector embeddings. Each of these vector embeddings contain multiple dimensions / features. It would not be possible to visualise all of these points without reducing the number of dimensions to a humanely representable / viewable format.

Post dimensionality reduction, the visualisation of these vectors would help understand the correlation between various embedding content, thus allowing us to better understand the dataset whilst also identifying clusters within the data. Potentially, the path of a query within a vector database could also be visualised.

2 Tools / Technologies used

The following steps have been taken to pre-process the above created dataset (text file containing tutorial content and code):

- Partitioning of the content into elements of a list based on subsections present within tutorials taken from the Pytch webapp.
- Feeding each of the subsections into the OpenAI Embeddings endpoint, which converts the content into vectors consisting of text based content along with multidimensional positional / spatial data.
- Dimensionality reduction of each of the vector embeddings into 3D vectors based on their Principal Component Analysis (PCA) on an array created from the cosine distances between each of the vectors.
- Appending additional attributes to the dataset to include additional representable data such as presence of code, length of content post scaling factor induction, etc.

The following tools and technologies have been used to pre-process and visualise the data contained within this dataset:

- **Pre-processing** – Python – Pandas, NumPy, SkLearn, Git, OpenAI GPT Embeddings Endpoint.
- **Visualisation** – Python – Plotly (3D Visualisation), Flask (Python based web application)
- **Deployment** – Python – Azure Web Apps Git Workflow Deployment (created a webapp to which the visualisation has been deployed)

Note: The webapp runs off a free Azure Web Apps subscription due to which it takes a LONG time to load and consists of a total of 1 hour of CPU run time per day.

3 Tasks

This visualisation aims at consuming and querying the data contained within the created dataset. Visualising the multidimensional vector embeddings helps discover correlations between themselves alongside exploring various patterns within the data. Provisions have also been made to query the content contained within these vectors to identify the locations of various topics and subtopics within the dataset.

4 Encoding Channels and Idioms

The following encoding channels have been used within this visualisation:

- **Position** – Dimensionality reduction with PCA based on cosine distances has been used in order to positionally encode the multidimensional vector embeddings into three dimensions, thus allowing it to be represented and visualised.
- **Size** – Size of each of the datapoints within the 3D Scatterplot are based on the length of the content present within each of themselves. This has also been normalised based on a scaling factor to ensure that the size of the datapoints are maintained throughout.
- **Colour hue** – Colour has been used to represent the presence / absence of code blocks within the text based subcontent contained in each of the datapoints.
- **Interactiveness** – Interactivity has been added to the visualisation to allow users to view the 3D scatterplot at different angles, along with zoom, pan and rotate capabilities. A search functionality has also been added to the visualisation, which allows the user to search for

content within the vector embeddings and the top searches based on cosine distance change colour temporarily to indicate the search results before reverting to their original colours.

The above encoding channels have sought to best serve the purpose of representing various aspects of this visualisation, especially with the search function which allows for a temporary change of colour to indicate search results. Position helps determine the closeness of two datapoints / subtopics to each other. Upon hovering, each of the datapoints also reveal their contents. Size helps determine the relationship between contents of different length whereas colour helps determine how the presence / absence of code affects the spread of datapoints throughout the visualised space.

5 Novelty

With the rise of LLMs, improvements in efficiency and personalisation in relation to LLMs has and will continue to stay one of the most invested and interesting topics for a long period of time. Embeddings and fine tuning have served to be the best ways to quickly tune LLMs towards personalisation of results. To create embeddings that assist these models to generate more accurate and personalised results, a visualisation of the embedding content is highly necessary and beneficial as it helps discover and identify patterns and correlations within the dataset. There exists a spectrum of visualisers within the market. Nomic AI (AI, n.d.) has developed Atlas to help create simple 2D visualisations of multidimensional vector embeddings without any effort whatsoever whereas TensorFlow (Tensorflow, n.d.) has developed Projector to help create complex 3D visualisations of multidimensional vector embeddings.

Our visualisation helps bridge the gap between both of these visualisers along with the presence of additional features that make it novel in its use case. The presence of various different visual encoding channels along with a 3D interactive view and the provision for search functionalities along with temporary result highlighting combine to form a novel visualisation of vector embeddings.

6 Critical Analysis

The strengths and weaknesses of the above data visualisation are as follows:

Strengths

- **Contextual Understanding**: The visualisation focuses on understanding the contextual relationships between vector embeddings by incorporating both text-based content and multidimensional positional data. This approach enhances the interpretability of the embeddings.
- **Interactivity**: The inclusion of interactive features improves user engagement and exploration of the dataset.
- **Multiple Encoding Channels**: The use of various encoding channels provides a comprehensive representation of different aspects of the data.
- **Application of Dimensionality Reduction**: The use of PCA for dimensionality reduction is a robust technique that helps in visualizing high-dimensional data in a more manageable 3D space, making it accessible for human interpretation.

Weaknesses

- **Performance Issues:** The mention of long loading times and limited CPU run time per day for the web application is a significant drawback.
- **Limited Dataset Information:** More information on the dataset characteristics would aid in a more nuanced understanding of the visualized data.

- **Sparse Information on Evaluation Metrics:** There is a lack of information on how the effectiveness or quality of the visualisation is measured or evaluated. Metrics such as interpretability, ease of use, or user feedback are essential for a comprehensive analysis.
- **Limited Insight into Cluster Identification:** While the document mentions the goal of identifying clusters within the data, it doesn't delve into how these clusters are defined or identified post-dimensionality reduction.

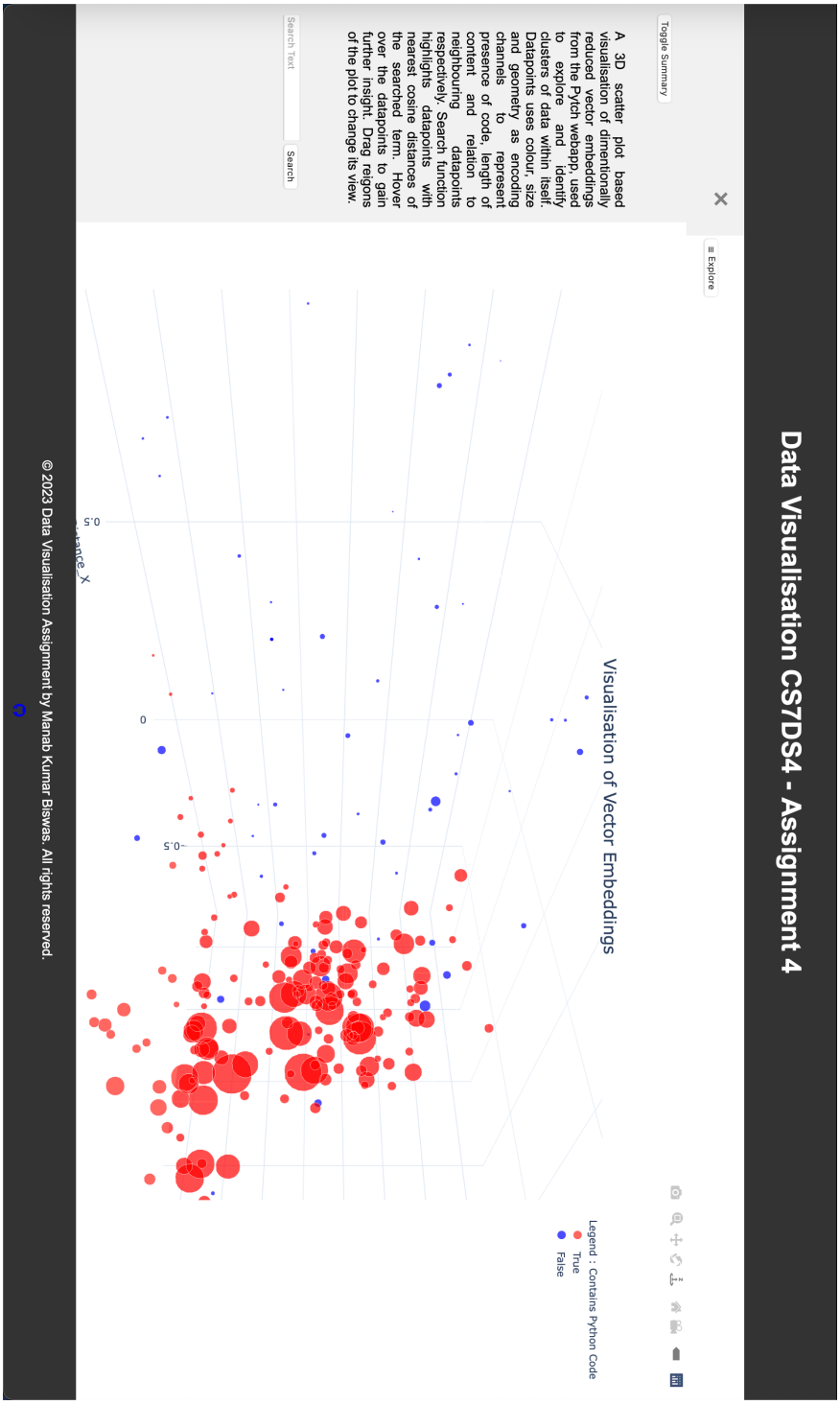

Figure 1 : Visualisation

References

AI, N. (n.d.). *Atlas*. Retrieved from Atlas - Nomic AI Visualiser: https://atlas.nomic.ai/

MKB. (n.d.). *Dataset*. Retrieved from GitHub Dataset: https://github.com/manab-kb/MKBDVA4

TCD. (n.d.). *Pytch*. Retrieved from Pytch Webapp: https://www.pytch.org/app/

Tensorflow. (n.d.). *Tensorflow*. Retrieved from Tensorflow Projector: https://projector.tensorflow.org/