

Big Data in Public & Social Services' Project

REPORT PROGETTO
AIDA & SkillGraph

Alessio Sanvito 844785
Andrea Ermellino 844623
Daniele Potertì 844892

Fabrizio Merlo 847203
Simone Radaelli 845065
Simone Sanvito 844794

Appello del 10/07/2023

Anno Accademico 2022–2023

Contents

1	Introduction	3
2	Main goals/objectives	3
3	Scraping	4
4	Chatbot Agent (AIDA)	4
4.1	Data Pre-processing	4
4.2	VectorStore Creation	5
4.3	Document Retrieval Chain	5
4.4	Agent	6
4.4.1	ReAct	6
4.4.2	Plan & Execute	7
4.5	Evaluation Steps & User Tests	7
4.5.1	First Test	7
4.5.2	Second Version	11
4.5.3	Hallucination problem	11
5	SkillGraph	12
5.1	Data Pre-processing - SUA and ESCO	12
5.2	GraphDB Construction	13
5.2.1	Nodes	13
5.2.2	Relationships	14
5.2.3	Graph Structure	14
5.3	Integrate GraphDB and LLM	15
5.4	Web Application	15
6	Conclusions and Results	17
	References	18

1 Introduction

This project introduces two distinct systems that utilize the same foundational data (SUA documents): an AI-driven chatbot platform and a comprehensive graph mapping. Both tools are designed to independently streamline the data retrieval process in the educational landscape, particularly for university students and stakeholders.

The AI platform, based on Large Language Models (LLMs) like GPT-3.5, leverages its impressive capabilities for text comprehension and generation to deliver accurate, context-specific academic information. It allows students to explore various degree programs offered by Italian universities, making this academic information accessible and user-friendly.

On the other hand, the graph tool provides a mapping of skills offered by each Italian university degree program. This is aligned with the European Skills, Competencies, qualifications, and Occupations (ESCO) framework, helping students comprehend the potential career paths associated with their academic choices.

Both the use of LLMs in the chatbot and the graph database aim to support how students and stakeholders access and understand university-related information, thereby empowering them to make informed academic decisions that align with their long-term career objectives. Through this potent combination, students are granted a valuable resource offering insights into the practical applications of their chosen fields of study.

2 Main goals/objectives

The main goals of this project are two:

1. Create a **Chat Bot System** that uses a **Large Language Model** and an **embedding database** to answer questions made up by some users. In this way, this implementation can innovate and ease the process of orientation for students that have finished high school or have a bachelor's degree.
2. Create a **Graph DataBase** that can match the skills supplied by a degree course with the skills needed for some occupations (jobs). Moreover, it will be possible to **query the graph with natural language queries** (and not necessarily by means of Cypher queries) by taking advantage of the usage of a **Large Language Model**, which enables the conversion of natural language into Cypher queries for a Neo4j graph database.

3 Scraping

In order to collect the necessary data from the website, a **web scraping** script was implemented using Python and the Selenium library. The scraping process involved several steps; all the SUA Documents were retrieved and scraped from the *University - L'università italiana a portata di click* site.

The first choice has been to use Selenium as a tool to interact with the website and retrieve the desired information and together with it the Chrome Web-Driver was used to enable programmatic control of the Google Chrome browser.

To organize the scraped data, a directory structure was created to store the information for each university. The scraping script iterated through the list of available universities and created a directory for each university. The name of each directory was derived from the corresponding university name, with any special characters or symbols sanitized to ensure compatibility with the file system.

The scraping process selects a specific university from a drop-down menu on a website, collects course data from the resulting table, and downloads additional details for each course, including the related PDF file. After gathering the necessary information, the script closes the course detail tab, returns to the main tab, and continues with the next course, ensuring efficient navigation and retrieval of information.

In conclusion, the implementation of the web scraping script using Selenium successfully retrieved the desired course information from the target website, enabling efficient data collection and organization for further analysis and decision-making.

4 Chatbot Agent (AIDA)

The chatbot system offers an approach to assist students in their post-high school or post-bachelor degree journey by combining the capabilities of a Large Language Model and an embedding database. It simplifies the process of orientation by providing accurate, comprehensive, and personalized responses to the students' questions, helping them make informed decisions about their future paths.

4.1 Data Pre-processing

So at this point, having all the SUA documents for all universities, for this task, only the SUA documents of the Università degli Studi di Milano-Bicocca were taken into account.

In addition, for an enhanced user experience and to ensure the delivery of precise and accurate information, the reliance on the SUA documents from

Università degli Studi di Milano-Bicocca was complemented with data from the syllabi of the courses of the degree programs offered by Bicocca University: these syllabi were obtained through a data scraping task carried out by external individuals who kindly shared the data with our group.

For each document created, relevant information was extracted from the metadata. This information was then enhanced by adding additional details, such as the source of the syllabus file (syllabus file path), the name of the degree course (nome cdl), the course code (codice cdl), the access type (tipo accesso), the type of degree course (tipo cds), the type of degree award (tipo rilascio), and the document type (tipo documento). As a result, a complete metadata file specific to each degree program was generated. An informative header is created using the extracted metadata and the course name. The header is added to the beginning of the document’s page content of all documents, along with additional formatting.

4.2 VectorStore Creation

All of this information has been saved in a vector store. A **ChromaDB vector store** is typically used when there is a need to efficiently store and retrieve high-dimensional vector data. It is particularly useful when dealing with large-scale datasets and complex data structures.

In the context of our task, the usage of a ChromaDB vector store can be justified for several reasons. Firstly, the scraping process resulted in a significant amount of data, including the syllabi of all courses taught at Bicocca University. These syllabi likely contain diverse and high-dimensional information, making a vector store a suitable choice for efficient storage and retrieval.

By structuring this data as vectors, it becomes easier to perform various operations such as similarity calculations and searching.

In order to save the documents in the vector store, every document considered has been split every 256 tokens with an overlap of 20 tokens.

Then, all the documents were added to the vector DB.

4.3 Document Retrieval Chain

Google Researchers have demonstrated that the introduction of a chain of thought, comprising a sequence of intermediate reasoning steps, effectively enhances the capacity of large language models to engage in intricate reasoning. As a result, these models naturally exhibit improved reasoning abilities, leading to enhanced performance across arithmetic, commonsense, and symbolic reasoning tasks (Wei et al., 2022).

A custom ConversationalRetrievalQA was developed by adapting the original one available in the Langchain library, guided by the Chain-of-Thought Prompting.

The chain of operations begins by integrating the chat history and the question, creating a unified standalone question. Subsequently, it retrieves pertinent documents using the retriever module (SUA and Syllabus Documents on ChromaDB). Finally, the retrieved documents, along with the question, are fed into a question-answering chain, which generates a response.

4.4 Agent

4.4.1 ReAct

Utilizing an agent, the application of Large Language Models (LLMs) is investigated to concurrently generate reasoning pathways and task-oriented actions, creating a symbiotic relationship between them. Reasoning pathways aid the model in formulating, monitoring, and revising action strategies while also managing anomalies. Concurrently, actions serve as a conduit for interacting with external entities such as knowledge databases or different environments, thereby accumulating supplementary information (Yao et al., 2022). By incorporating the ReAct Prompting methodology (shown in Figure 1), it is possible to mitigate the common issues of false information generation and error cascading inherent in sequential reasoning. This is achieved through multiple interactions with the Document Retrieval Chain Tool.

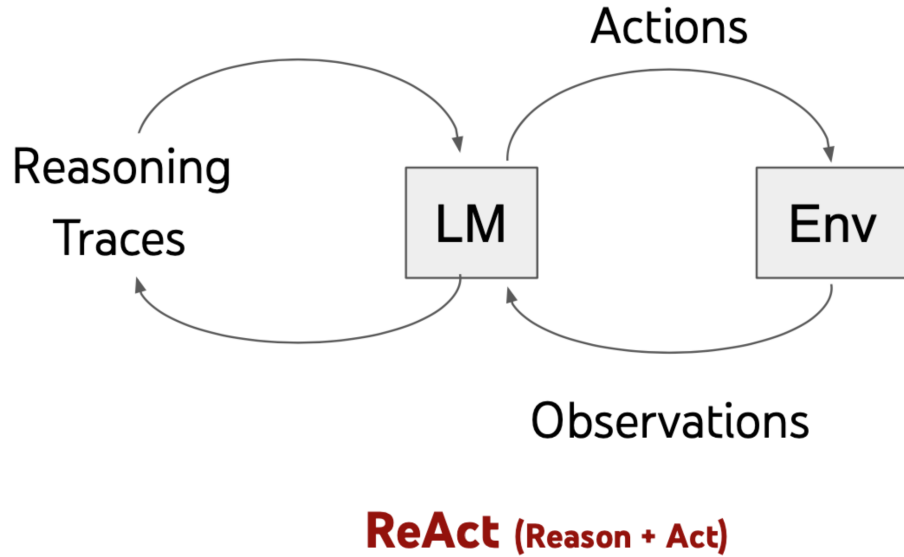


Figure 1: ReAct Framework

4.4.2 Plan & Execute

In line with the procedures proposed by (Wang, Lan, Lan, & Lim, 2023) in their study, the 'Plan & Execute' prompting approach was integrated into AIDA. The initial step of this process involves decomposing the primary task into smaller, more manageable units, and creating a comprehensive plan to address each segment methodically. This systematic planning is fundamental to the effective utilization of the 'Plan & Execute' method, as it structures the tasks at hand more efficiently.

Once the planning phase is completed, the execution phase is initiated. In this stage, the individual segments are worked upon by the pre-determined plan. Every action is based on a prior project, thereby promoting efficiency and providing very specific answers to general questions.

This sequential approach to handling tasks is proven to deliver improved results that are more specific and targeted. It aids in narrowing down broad tasks into precise actions, ultimately leading to an output more aligned with the intended goal. However, it's essential to note that this method still has some drawbacks, such as hallucinations, long response times, and expensive LLM calls.

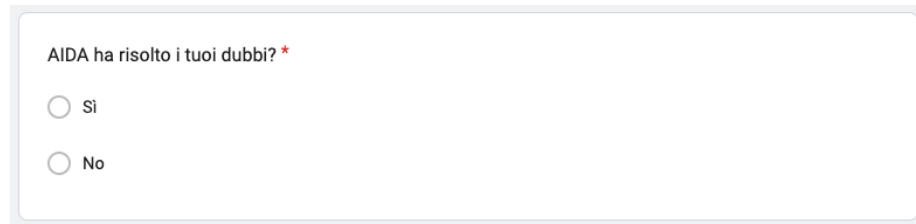
4.5 Evaluation Steps & User Tests

To evaluate the work done on the chatbot, a test was done to assess how satisfied a user was with the responses received. Users were asked to provide feedback regarding their experience with the chatbot and to fill out a questionnaire.

4.5.1 First Test

An α **version of AIDA** was given to a tiny circle of people for testing. 14 people took part in the AIDA First Test, each of whom was asked to fill in a questionnaire at the end of the test.

The AIDA First Test questionnaire was structured as follows:



AIDA ha risolto i tuoi dubbi? *

☐ Si

☐ No

Figure 2: AIDA First Test questionnaire Question 1

Quanto è stata esaustiva nelle risposte? *

	1	2	3	4	
Per niente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Moltissimo

Figure 3: AIDA First Test questionnaire Question 2

AIDA ha capito cosa stavi cercando? *

- ☐ Mi ha risposto subito
- ☐ Ho dovuto specificare meglio in una domanda successiva
- ☐ Ho dovuto ripetere più volte la mia richiesta
- ☐ Non l'ha proprio capito
- ☐ Altro...

Figure 4: AIDA First Test questionnaire Question 3

Quanto sei soddisfatto dell'assistenza di AIDA? *

	1	2	3	4	
Per niente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Moltissimo

Figure 5: AIDA First Test questionnaire Question 4

At the end of the questionnaire, the user was able to give overall feedback on AIDA usage.

Despite being an α version, the impact of AIDA on users was good. The results extrapolated from the questionnaire are summarised below:

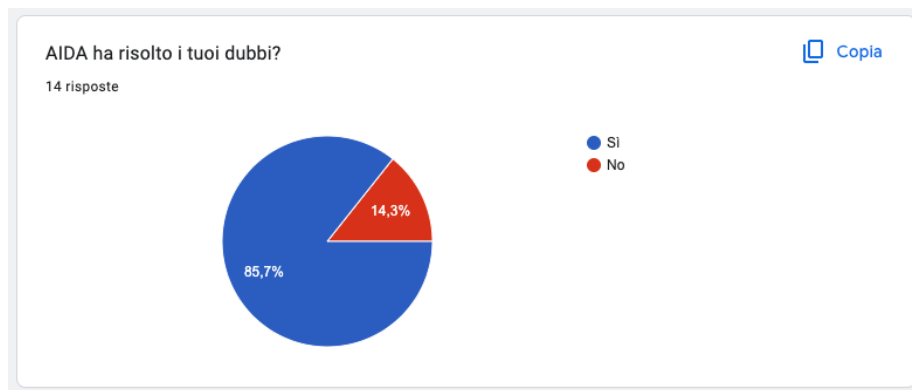


Figure 6: AIDA First Test questionnaire Results question 1

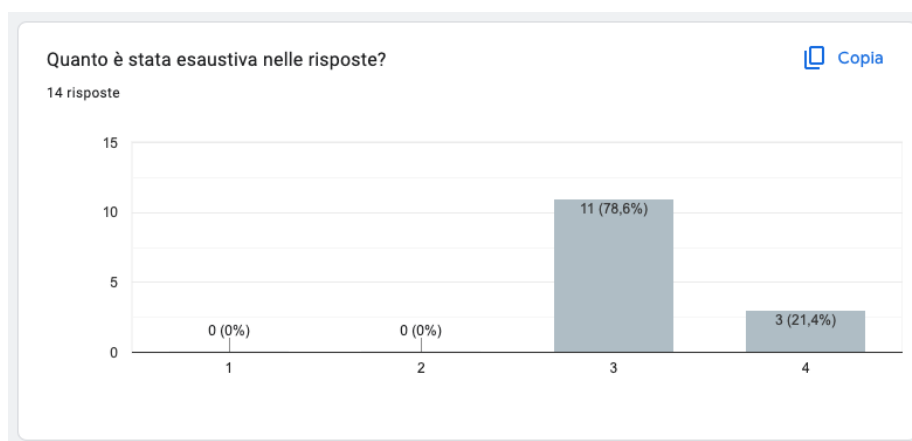


Figure 7: AIDA First Test questionnaire Results question 2

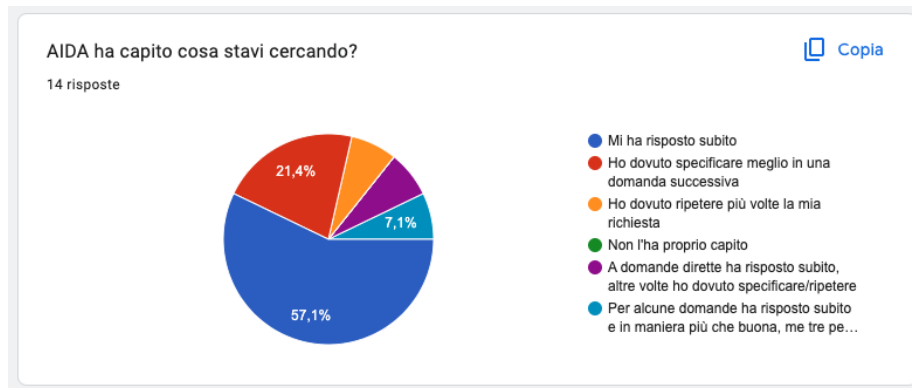


Figure 8: AIDA First Test questionnaire Results question 3

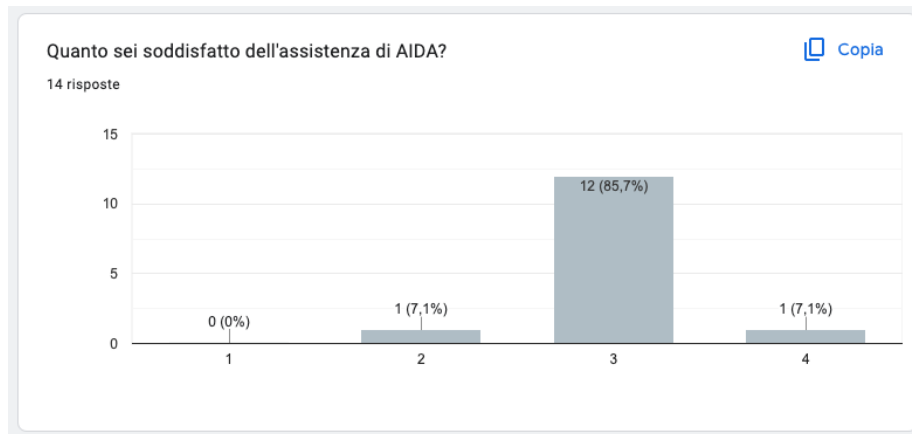


Figure 9: AIDA First Test questionnaire Results question 4

The overall user experience was good, questionnaire results show that AIDA was comprehensive in its answers, quick in most cases to understand the users' requests, and managed to resolve their doubts.

Users' feedback showed that the AIDA version gave poor answers to questions concerning degree program planning (which teachings are offered in a given academic year), so the β **AIDA version** was developed to improve this aspect.

4.5.2 Second Version

Users had reported that one area for improvement was artificial intelligence’s knowledge of the exam program and its temporal organization. To address this improvement point, we searched the SUA files to see if any relevant information was available.

Additionally, we have discovered some tables that provide additional information on the courses offered by Bicocca University. These files contain details such as the names of all courses, their sectors, course years, teaching names, professors’ names, credits, and hours. It is important to note that these tables were present in the SUA documents but required preprocessing and separate loading due to encoding issues.

The overall functioning is satisfying. The major issue that has been encountered is that of hallucination, a well-known problem in the field of artificial intelligence.

4.5.3 Hallucination problem

As the definition of Wikipedia says (Wikipedia contributors, 2023) in artificial intelligence (AI), a "hallucination" or "artificial hallucination" is when an AI confidently gives an answer that isn’t backed up by the data it was trained on. For instance, an AI chatbot might incorrectly say that Tesla made \$13.6 billion, even though this number isn’t based on any real data.

This term is similar to the human experience of hallucination, but in AI, it refers to unjustified answers or beliefs. Some people argue that calling it an "AI hallucination" wrongly makes computers seem human-like.

The term started being used more around 2022 when big language models like ChatGPT were launched. Users noticed that these AI often made up believable but random lies in their responses. By 2023, these frequent "hallucinations" were seen as a big issue in AI technology. An example of "hallucination" can be seen in Figure 10.



summarise this article <https://www.nytimes.com/2023/03/11/technology/chatgpt-prompts-to-avoid-content-filters.html>



The article discusses how ChatGPT, an AI language model, is being used to circumvent content filters that are designed to detect and block certain types of content. By generating prompts that are less likely to trigger these filters, ChatGPT can be used to create content that might otherwise be censored. While this technology could be useful for those seeking to express themselves freely, it also raises concerns about the potential misuse of AI-generated content, as well as the difficulty of regulating such content in the future. The article concludes by noting that there is still much to learn about the impact of AI language models like ChatGPT on the way we communicate online.

Figure 10: ChatGPT summarizing a non-existent New York Times article even without access to the Internet

5 SkillGraph

The second main task of the project was to create a **GraphDB** that could represent the relationships between universities, degree programs, occupations, and skills. So the graph captures how courses prepare students for certain occupations, the skills required for each occupation, and the universities that offer the corresponding degree programs.

5.1 Data Pre-processing - SUA and ESCO

To do this, **ESCO's datasets** were used. The European multilingual classification of Skills, Competences, and Occupations (ESCO) serves as a glossary of occupations and skills relevant to the EU labor market and education and training. ESCO provides descriptions of 3.008 occupations and 13.890 skills linked to these occupations, translated into 27 languages. The ESCO classification is composed of modules that contain elements such as occupations, knowledge, skills and competencies, qualifications, and the International Standard Classification of Occupations (ISCO) hierarchy. When combined and interrelated, these modules make up the whole classification.

For this process, all the available SUA documents of all the universities retrieved were considered (not only the SUA documents of the Università degli Studi di Milano-Bicocca). So in the process were used more than 5000 SUA documents.

In particular, the table A2.b of each SUA document was extracted, which represents the employment opportunities associated with the unique ISTAT code for each occupation/job; therefore, the job and the ISTAT code (explained as a

unique identifier for occupations) were saved in a table, capturing the relevant information from each university’s SUA document.

Some of the files used are all retrieved from ESCO’s site.

The datasets used in the graph creation were slightly adjusted to better align with the requirements needed and enhance the matching between user queries and the Cypher queries generated by the language model; this involved removing irrelevant numbers and alphanumeric strings from course names and converting all course names to uppercase to avoid issues with case sensitivity.

5.2 GraphDB Construction

The graph creation process involved first building the nodes with their respective properties. Subsequently, the nodes were interconnected to establish relationships, each with its corresponding properties.

To create the GraphDB, the following steps were followed:

5.2.1 Nodes

1. **Skills:** The skills data were read from the CSV file retrieved from ESCO’s site. Each row of the file represents a skill with various attributes such as concept type, concept URI (the URI of the skill), the preferred label of the skill, skill type, etc. For each row, a corresponding node of type ”Skill” was created in the GraphDB.
2. **Occupations with ISTAT mapping:** Another CSV file was read, containing occupation data along with ISTAT mapping information. The attributes of each occupation included concept type, concept URI (the URI of the occupation), the preferred label of the occupation, etc. Nodes of type ”Occupation” were created in GraphDB based on the data in the CSV file.
3. **Universities:** The unique university names were read from the specific CSV file that takes all the names of all universities retrieved with the scraping task. Each university name represents a node of type ”University” in the GraphDB.
4. **Courses:** The course data was obtained from the CSV file in which each row represents a course with attributes like course name, course code, work, and ISTAT code. As in the previous one, even the file used for the creation of these nodes was created using the information obtained by means of scraping. Nodes of type ”Course” were created in GraphDB based on the information in the CSV file.

5.2.2 Relationships

Relationships in the GraphDB:

1. **HAS_SKILL**: This connection establishes the relationship between occupations and their associated skills, with the creation of edges of type "HAS_SKILL". Each row contains information about the occupation URI and the skill URI, along with the relation type and skill type.
2. **OFFERS**: This connection establishes the relationship between universities and their offered courses. Each row contains the name of the university and the name of the course. Nodes representing the universities and courses were matched, and edges of the type "OFFERS" were created between them.
3. **LEADS_TO**: This relationship connects courses with their respective occupational outcomes. The matching phase was performed based on the course code and the Classification2 ID of occupations. For each pair of matching nodes, an edge of type "LEADS_TO" was created from the Course node to the Occupation node.

5.2.3 Graph Structure

So, the general structure that results after the graph creation is shown in Figure 11:

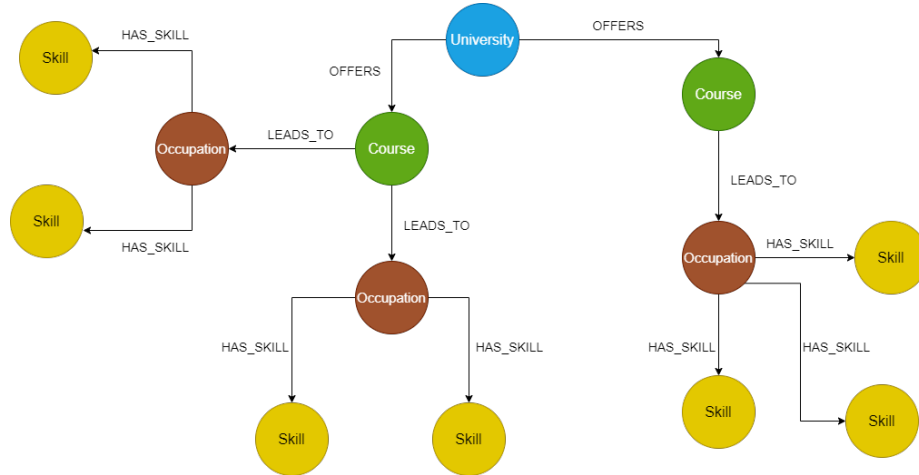


Figure 11: General and synthetic structure of the Graph DB

5.3 Integrate GraphDB and LLM

The translation capability of large language models for converting natural language into query language is highly promising. Since the graph DB was constructed in **Neo4j**, the query language is Cypher.

For this task, it was used a **GraphCypherQACChain**, a tool that permits the conversion of the natural language to a Cypher statement; then uses that Cypher statement to retrieve data from Neo4j and, at last, returns the retrieved information to the user in a natural language form. This two-way conversion process between natural language and database language not only enhances the overall accessibility of data retrieval but also greatly improves the user experience.

By utilizing the graph schema information, this approach implements a model that generates Cypher queries on that graph. At first, a connection to a Neo4j instance was established from Python, allowing for the retrieval of schema information during initialization. Subsequently, this graph schema information serves as input for the large language model.

APOC is a library that comprises around 450 procedures and functions that can be helpful in different areas, including collection manipulation, graph algorithms, and data conversion.

Regarding the specific function `apoc.meta.data()`, it is used to retrieve metadata information about the database. It provides details about the schema, labels, relationship types, and property keys used in the database. This metadata is used to inform and enhance the interaction between the language model and the graph database. It allows the language model to understand and work with the database schema, enabling more meaningful queries and insights.

5.4 Web Application

Additionally, a web app has been developed to offer users an interactive interface for querying the graph. To accomplish this, the **Streamlit framework** was employed. Streamlit is an open-source framework for rapidly creating interactive web applications. It is designed to simplify the process of developing and deploying data-driven applications.

The app consists of several pages (the Home Page can be seen in Figure 12), each serving a specific purpose. Here's a brief overview of the available pages:

1. **Home:** The home page of the application serves as an introductory section, providing users with an overview of the SkillGraph pipeline and its various operations. Then, the home page proceeds to describe the structure of the underlying graph that the application relies on. It provides users with a clear understanding of the nodes and relationships within the graph. Concluding the home page is a user guide section, which aims to assist users in navigating the application and understanding its different sections. It offers a brief description of each available page, providing users with a glimpse into the purpose and functionality of each section.

-
2. **App:** App page represents the main functionality of the application. It establishes a connection to the Neo4j graph database and the OpenAI API. Users can input text and submit it to receive a response from the OpenAI model. The code handles the interaction with the OpenAI API and retrieves the generated response. Additionally, it retrieves and displays information about the database schema, providing users with insights into the underlying data structure.
 3. **Course:** Course page displays course data. It allows users to filter courses based on the university and course name. The data is read from a CSV file, which contains information about various courses, and is presented in a table format. Users can interact with the table and explore different courses offered by universities.
 4. **Skills:** Skills page displays skills data. It enables users to filter skills based on the reuse level and skill type. The skill data is stored in a CSV file, and the code reads this file to retrieve the information. The displayed information is presented in a tabular format, allowing users to explore different skills and filter them based on their specific criteria.
 5. **Occupation:** Occupation page presents occupations data. It provides users with the ability to filter occupations based on various criteria. The occupation information is stored in a CSV file, and the code reads and displays this data in a table format. Users can interact with the table to explore different occupations and apply filters to narrow down the results based on their preferences.

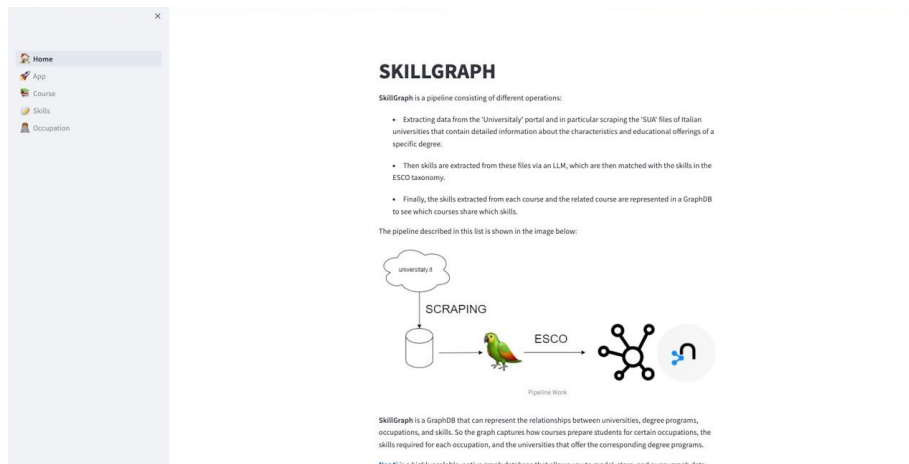


Figure 12: Home page of the Streamlit Webapp

6 Conclusions and Results

Both tasks offer ample opportunities for enhancement, with the most substantial augmentation undoubtedly originating from the utilization of GPT-4 as opposed to GPT-3.5-turbo. Nonetheless, given its unavailability, this remains a speculative hypothesis for the ongoing project.

The principal objective of this endeavour is to illustrate how Language Learning Models (LLMs) can significantly streamline numerous communication tasks. They can be harnessed to auto-generate written reports, email replies, and presentations, among other forms of text-based content, drastically accelerating the workflow and liberating individuals to concentrate on other critical tasks.

A vivid example can be seen in the customer service sector where AI's potential is extraordinary. By integrating an AI chat model, a customer service representative can be suggested optimal answers to customer inquiries, thereby amplifying the service quality and fostering simultaneous learning. This fusion of technology and human interaction accelerates the representative's evolution into an expert, a process significantly quicker than traditional training methods.

Nevertheless, there is a substantial requirement for further advancements to ensure that the information provided is not only precise but also specifically tailored to individual needs, thereby eliminating the chance of AI-induced hallucinations.

References

- Wang, X., Lan, H., Lan, L., & Lim. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- Wikipedia contributors. (2023). *Hallucination (artificial intelligence)* — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=Hallucination_\(artificial_intelligence\)&oldid=1162775450](https://en.wikipedia.org/w/index.php?title=Hallucination_(artificial_intelligence)&oldid=1162775450). ([Online; accessed 1-July-2023])
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.