Capstone Project

Startup Project 2nd submission

Group 10

Dino Meng - 20241265
Lourenço Passeiro - 20221838
Miguel Marques - 20221839
Peter Lekszycki - 20221840
Tomás Gonçalves - 20221894

1st December 2024

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Chatbot Data Sources

First part of the deliverable

X

Universities Data: Based on the following dataset containing websites of every university in the world: https://www.kaggle.com/datasets/thedevastator/all-universities-in-the-world
We selected a small sample of the dataset (50 universities), and most of them are portuguese universities.
Manual adjustments:

- We manually added the University of Trieste to the dataset;
- The non-Portuguese universities are manually selected to be European, for practical purposes (as universities from other continents can have a different course scheme and organization).

The sampling process is stored in the `sample_dataset.ipynb` notebook, with the random state for replicability purposes.

Courses, Requisites, Scholarships, Subjects, Internationals Data: To obtain deeper information about the educational offer of each university, we selected an even smaller sample from the previous sample (10 universities) and manually extracted the information from each website.

Manual adjustments:

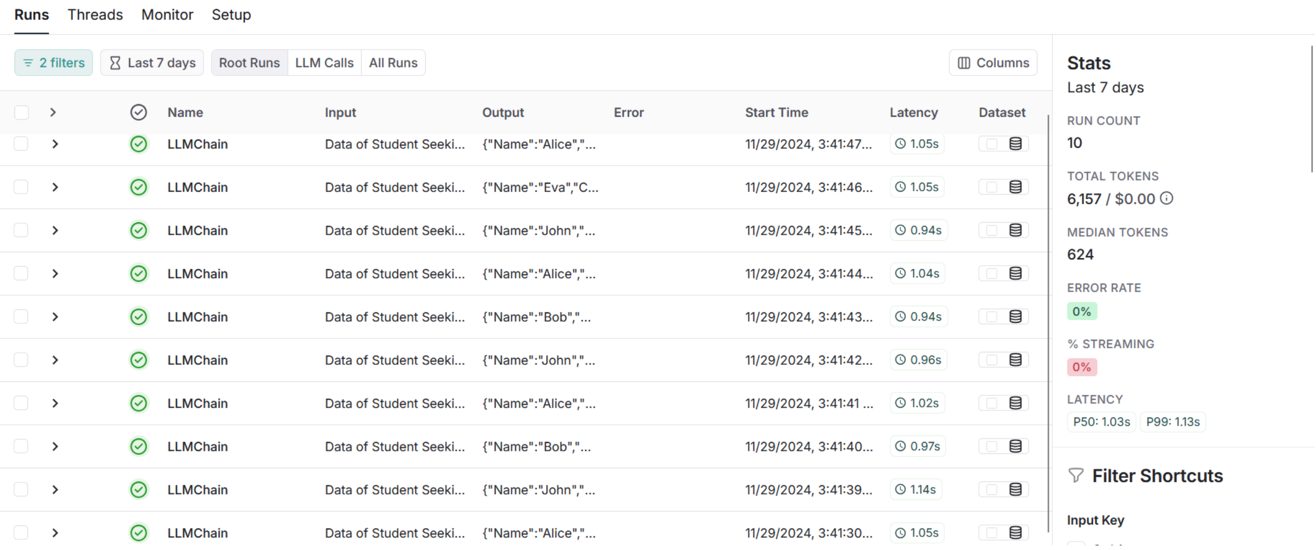
- We manually selected the New University of Lisbon (NOVA) and University of Trieste.
- We replaced Instituto Superior D. Afonso III - INUAF with another Portuguese university as it seems to be closed

Considering that each university should have at least 5 courses, this should be enough to populate the database for testing purposes.
The selected websites are as follows:

- http://www.ipportalegre.pt/
- http://www.ispa.pt/
- http://www.unitus.it/
- http://www.bme.hu/
- http://www.uni-dubna.ru/
- http://www.ipcb.pt/
- http://www.eshte.pt/
- https://www.ipl.pt/

- http://www.units.it/
- http://www.unl.pt/

Users Data: Synthetically generated 10 users, each with at least 5-10 preferences and areas
- Total tokens used: 6157
- On average, each call costed €0.0004, meaning that in total the whole run costed around €0.004
- Script is called `synthetic_users.ipynb`, and the generated data is stored in `users.json`



	Name	Input	Output	Error	Start Time	Latency	Dataset
✓	LLMChain	Data of Student Seeki...	{"Name":"Alice","...		11/29/2024, 3:41:47...	1.05s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Eva","C...		11/29/2024, 3:41:46...	1.05s	
✓	LLMChain	Data of Student Seeki...	{"Name":"John","...		11/29/2024, 3:41:45...	0.94s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Alice","...		11/29/2024, 3:41:44...	1.04s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Bob","...		11/29/2024, 3:41:43...	0.94s	
✓	LLMChain	Data of Student Seeki...	{"Name":"John","...		11/29/2024, 3:41:42...	0.96s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Alice","...		11/29/2024, 3:41:41 ...	1.02s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Bob","...		11/29/2024, 3:41:40...	0.97s	
✓	LLMChain	Data of Student Seeki...	{"Name":"John","...		11/29/2024, 3:41:39...	1.14s	
✓	LLMChain	Data of Student Seeki...	{"Name":"Alice","...		11/29/2024, 3:41:30...	1.05s	

Stats
Last 7 days

RUN COUNT
10

TOTAL TOKENS
6,157 / $0.00

MEDIAN TOKENS
624

ERROR RATE
0%

% STREAMING
0%

LATENCY
P50: 1.03s P99: 1.13s

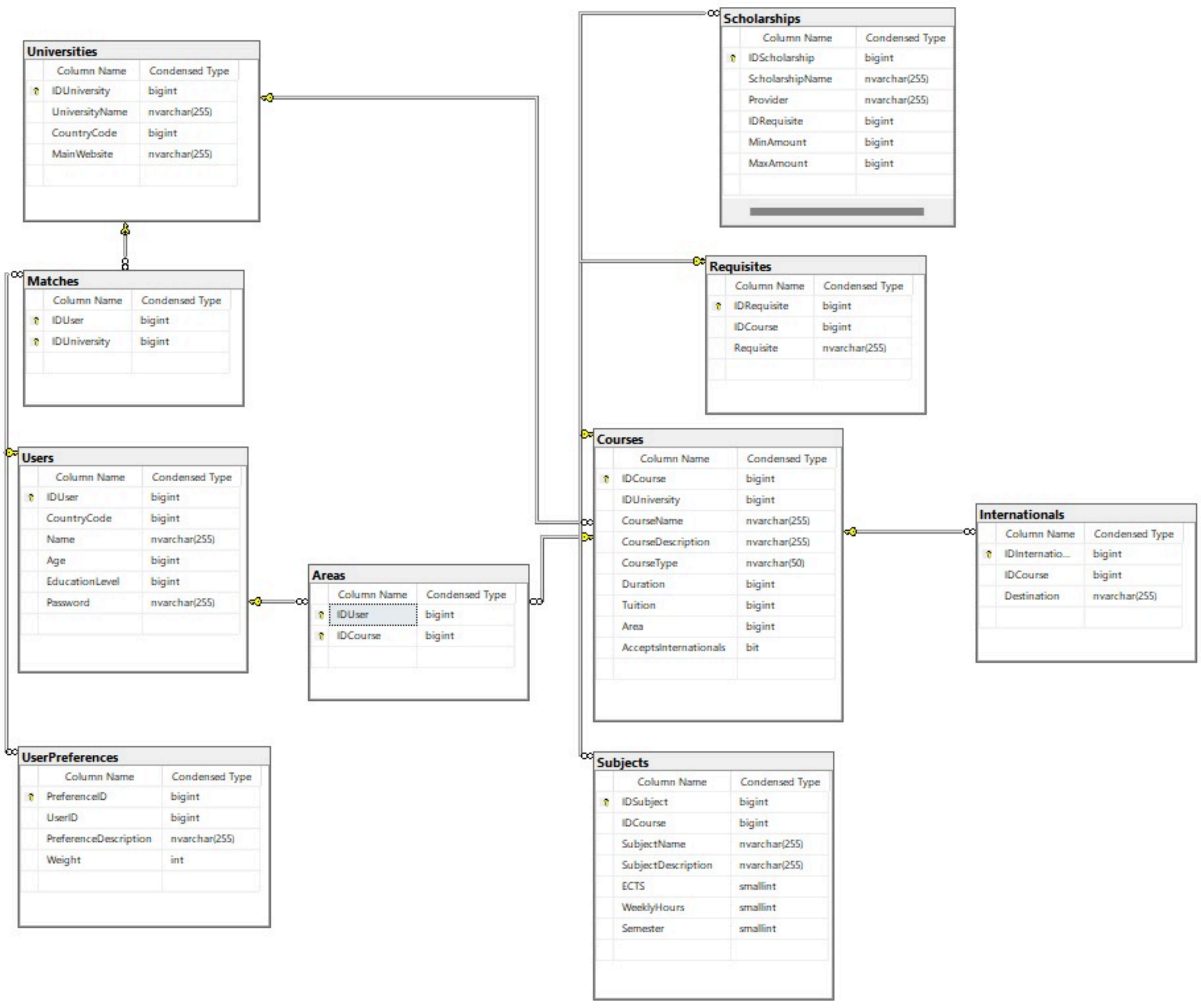
Filter Shortcuts

Input Key

PDF Data: 3 PDF files about the company and chatbot are manually written, and the user base can upload as many PDF files as they want for RAG purposes (see chatbot architecture).

Chatbot Database Schema

Second part of the deliverable



Universities

Column Name	Condensed Type
IDUniversity	bigint
UniversityName	nvarchar(255)
CountryCode	bigint
MainWebsite	nvarchar(255)

Matches

Column Name	Condensed Type
IDUser	bigint
IDUniversity	bigint

Users

Column Name	Condensed Type
IDUser	bigint
CountryCode	bigint
Name	nvarchar(255)
Age	bigint
EducationLevel	bigint
Password	nvarchar(255)

Areas

Column Name	Condensed Type
IDUser	bigint
IDCourse	bigint

UserPreferences

Column Name	Condensed Type
PreferenceID	bigint
UserID	bigint
PreferenceDescription	nvarchar(255)
Weight	int

Scholarships

Column Name	Condensed Type
IDScholarship	bigint
ScholarshipName	nvarchar(255)
Provider	nvarchar(255)
IDRequisite	bigint
MinAmount	bigint
MaxAmount	bigint

Requisites

Column Name	Condensed Type
IDRequisite	bigint
IDCourse	bigint
Requisite	nvarchar(255)

Courses

Column Name	Condensed Type
IDCourse	bigint
IDUniversity	bigint
CourseName	nvarchar(255)
CourseDescription	nvarchar(255)
CourseType	nvarchar(50)
Duration	bigint
Tuition	bigint
Area	bigint
AcceptsInternationals	bit

Internationals

Column Name	Condensed Type
IDInternatio...	bigint
IDCourse	bigint
Destination	nvarchar(255)

Subjects

Column Name	Condensed Type
IDSubject	bigint
IDCourse	bigint
SubjectName	nvarchar(255)
SubjectDescription	nvarchar(255)
ECTS	smallint
WeeklyHours	smallint
Semester	smallint

Users: Core table representing users, the primary stakeholders, whose data drives recommendations and matches.

- EducationLevel is a numerical attribute which represents their current educational level. 0 is for high school diploma (meaning he's looking for a bachelor's degree or equivalent), 1 for BsC diploma (meaning he's looking for a master's degree or equivalent), 2 for MsC diploma (meaning he's looking for a PhD or postgraduate courses)

User Preferences: The user preferences table is essential for providing a personalized experience to each user, suggesting a university and/or course that most fits their needs.

- Weight is a number which represents the "important" of a preference. All weights for an user must sum to 100.

- PreferecenceDescription is a short sentence describing the preference

Universities: Essential to representing institutions and linking them to their respective range of courses.
Courses: Central to educational offerings, representing the academic programs each university provides.

- AcceptsInternationals is an indicator which describes whether a course accepts international students or not.

Subjects: This table is important to detail more each course, allowing students to evaluate the areas of study and workload within a course.
Areas: Bridges the connection between users and courses by storing the existent areas of study.
Requisites: Ensures users meet course entry requirements, preventing any mismatch in course recommendations.

- They can be either requisites to courses or requisites to get a scholarship.

Scholarships: Provides funding opportunities for users, enhancing accessibility and motivation for education.
Internationals: This table facilitates international opportunities by identifying the courses open to such exchange programs.
Matches: Makes the final connection between users and universities, enabling personalized matching insights.

Chatbot Architecture

Third part of the deliverable

x

Features

Manage Personal Information (CREATE, UPDATE, DELETE)

As a student seeking university,

I want to manage the personal information about me, by accessing, adding, removing or updating them

So that the chatbot can know more about me and make informed recommendations

Query for Scholarships and International Options (READ)

As a student seeking university,

I want to query for scholarships or international opportunities offered by universities,

So that I can understand my financial options or international options and determine which universities fit my budget.

University Information (READ)

As a student seeking university,

I want to to be able to retrieve detailed information about an university or a course, and know whether it is for me or not (according to my preferences and requisites),

So that I can make an informed decision on my education

University Course Recommendation (READ + WRITE + DELETE)

As a student seeking university,

I want to be able to retrieve university course recommendations according to my preferences, requisites and the information available in the database, then add them to my possible matches into the database (and remove the old recommendations if they exist)

So that I can know which are my potential choices

Query Matches (READ)

As a student seeking university,

I want to be able to retrieve university recommendations that have been previously made for me, if they exist,

So that I can remember which were my potential choices recommended by UniMatch

Extracting Information from Text Sources (RAG + Web Scraping)

As a student seeking university,

I want to retrieve information from a source of information about an university or a course, such as a website or a "fact sheet" in form of PDF,

So that I can receive an analysis of the document, tailored to my preferences

Company Information (RAG)

As anyone (students, potential investors, curious people, et cetera...),

I want to know more about the company and the chatbot, such as the company values and mission, or pointers on how to use the chatbot

So that I can gain more information about the company and/or receive a clear idea on the usage of chatbot

Chatting

As anyone,

I want to simply chat with the bot

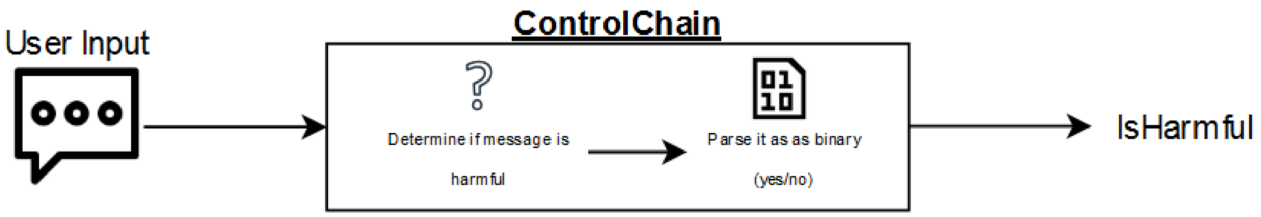
So that I can interact with it as if it were a real human

X

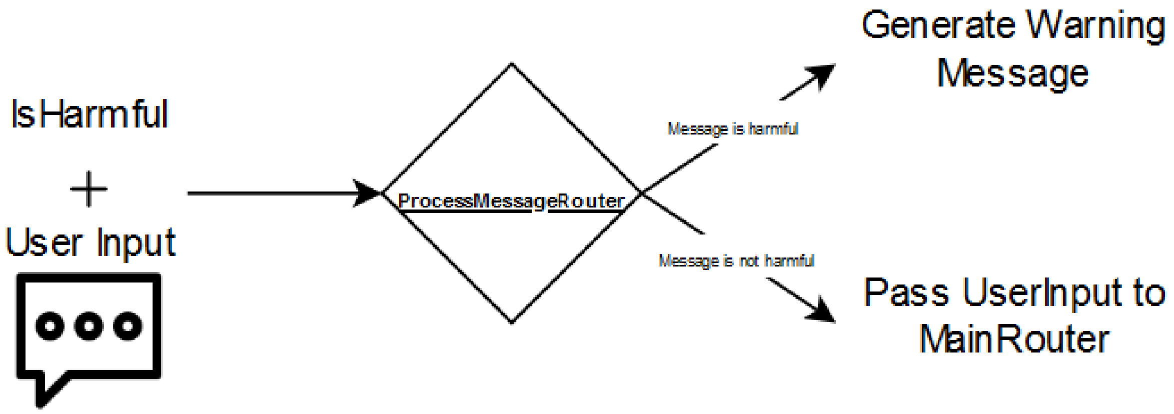
Chains

~~Input Handling~~

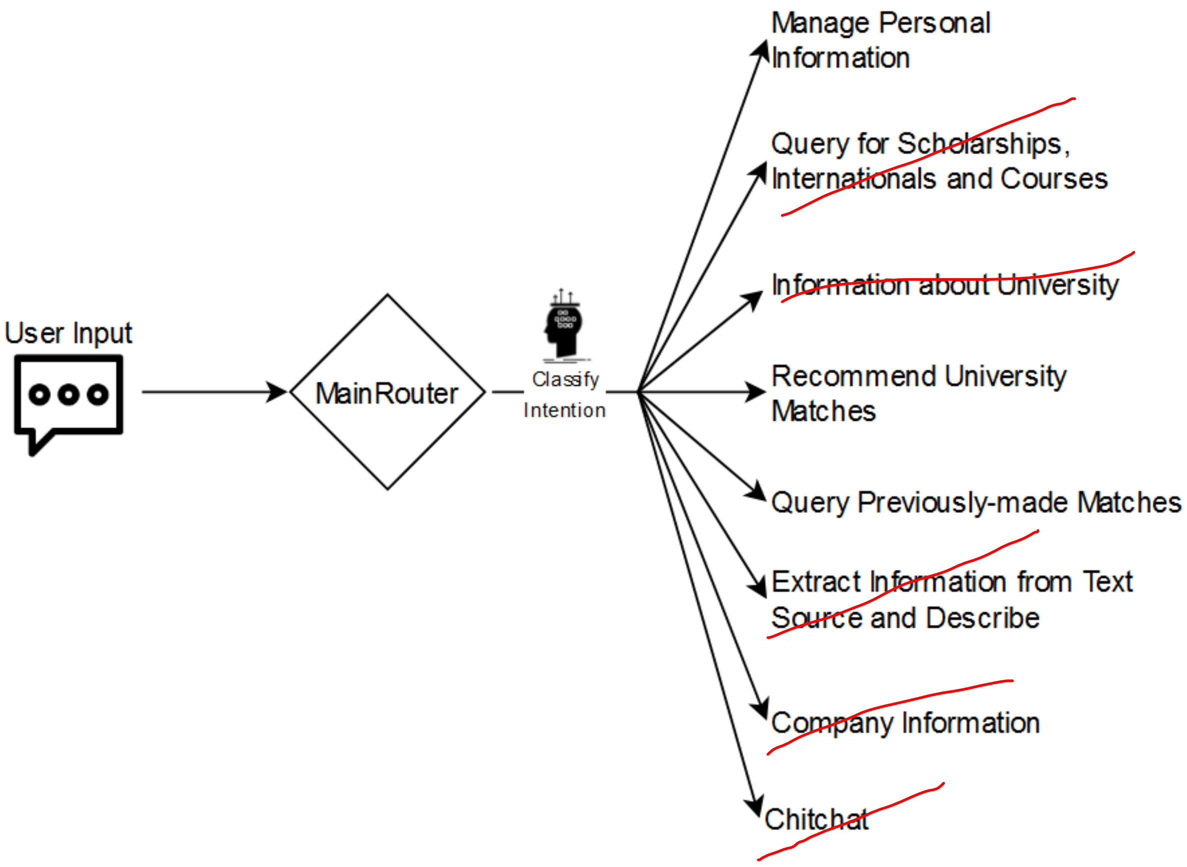
~~ControlChain~~: Make sure that user input does not represent an attempt to either request for harmful information or to commit injections. Return result as a boolean (1 for harmful input, 0 for acceptable input)


User Input
ControlChain
Determine if message is harmful
Parse it as as binary (yes/no)
IsHarmful

~~ProcessMessageRouter:~~ Accept input or not, redirecting them to different chains. If the message is determined to be harmful, generate a warning message to discourage the user with further attempts. If not, then redirect the message to the main router.

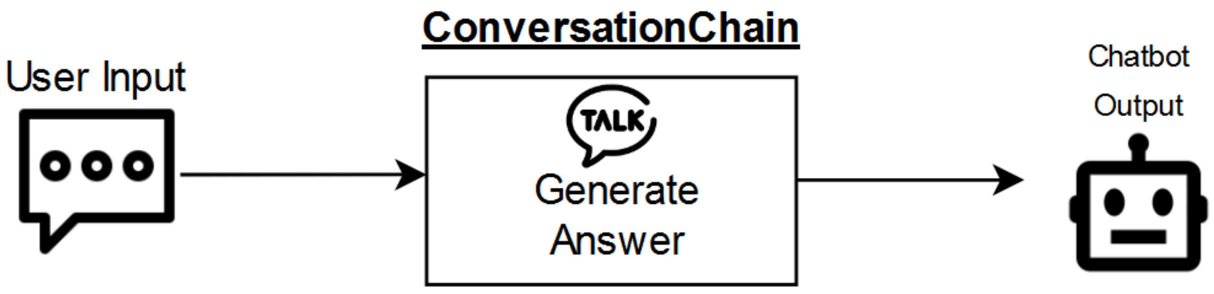


MainRouter: Classify the given message into one of the intents, and call the agent to handle the intent.

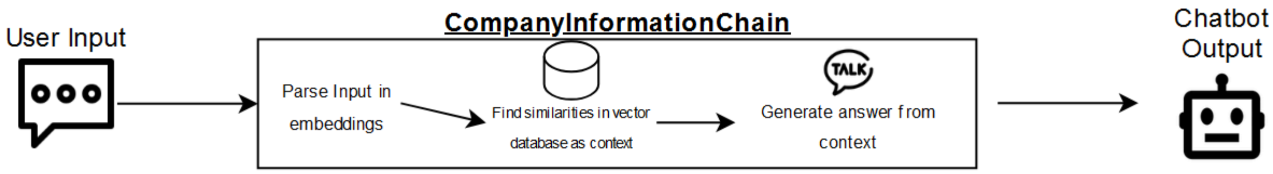

User Input
MainRouter
Classify Intention
Manage Personal Information
~~Query for Scholarships, Internationals and Courses~~
~~Information about University~~
Recommend University Matches
Query Previously-made Matches
~~Extract Information from Text Source and Describe~~
~~Company Information~~
~~Chitchat~~

Handle Conversation and Chatting

~~ConversationChain~~: Process the message as a conversation item, attempting to redirect users towards a pertinent question.

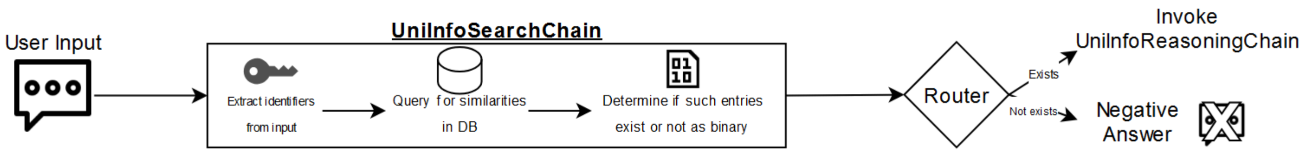


~~CompanyInformationChain~~: Process the message as a question about the company, and answer it with a pertinent answer. This step leverages RAG as it involves accessing a vector database through Pinecone.

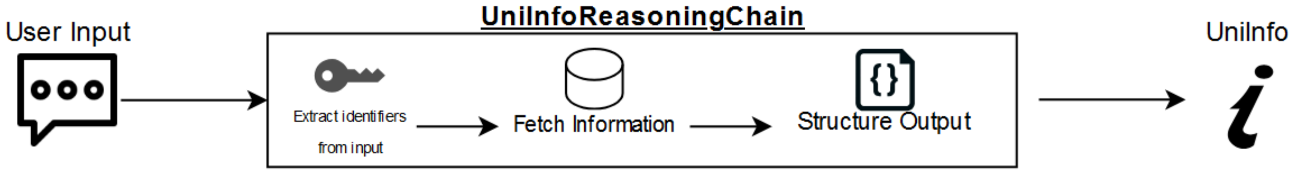

CompanyInformationChain
User Input
Parse Input in embeddings
Find similarities in vector database as context
TALK
Generate answer from context
Chatbot Output

University Information

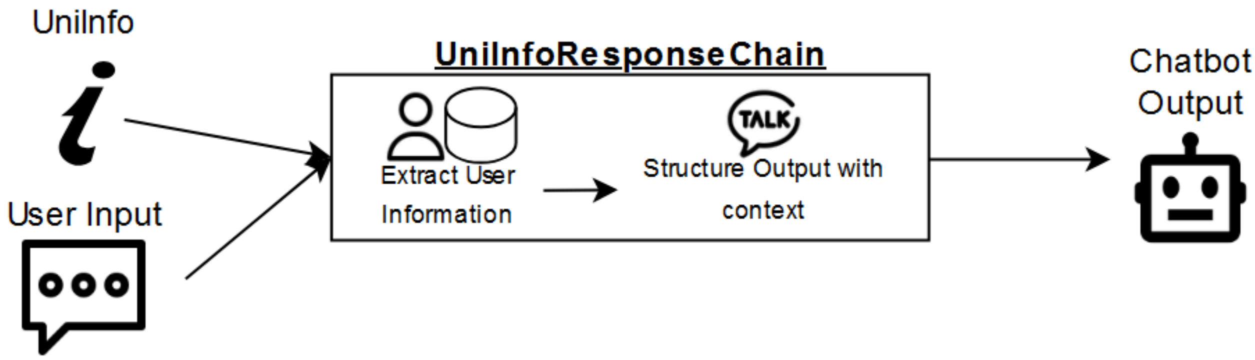
~~UniInfoSearchChain~~: From user input (info. about university or course), attempt to search it in the database. If it exists, return 1; otherwise 0. Its main goal is to avoid "hallucinations" by the bot.


UniInfoSearchChain
User Input
Extract identifiers from input
Query for similarities in DB
Determine if such entries exist or not as binary
Router
Exists
Invoke UniInfoReasoningChain
Not exists
Negative Answer

UniInfoReasoningChain: From user input (information about university or course), identify the university, fetch it from the SQLite database and parse it as an opportune structure.

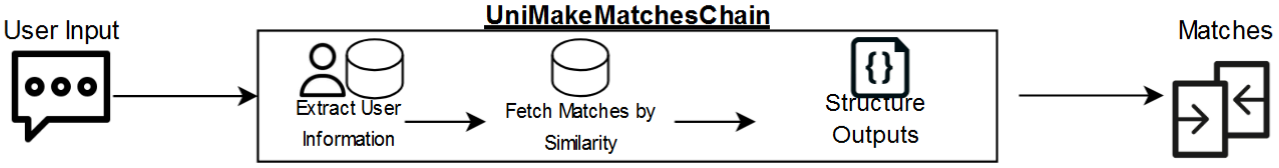


~~UniInfoResponseChain~~: Given a `UniInfo` entity, structure it as a chatbot output, tailoring it to user's preferences if possible

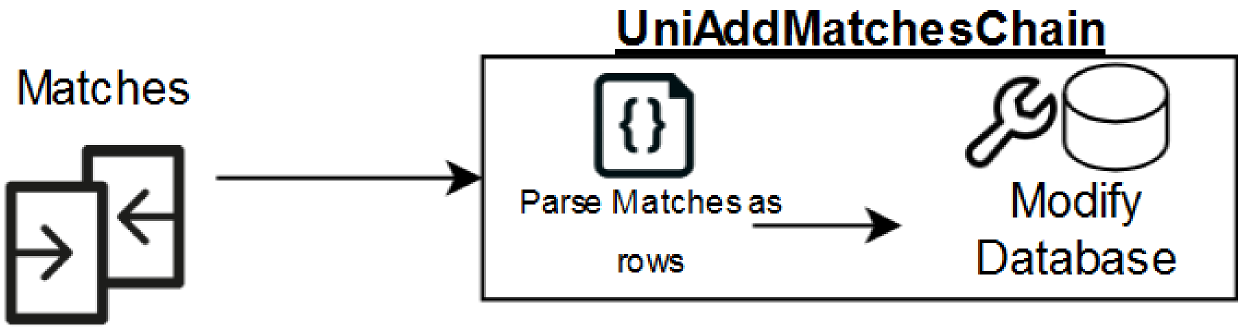


Make & Query Matches

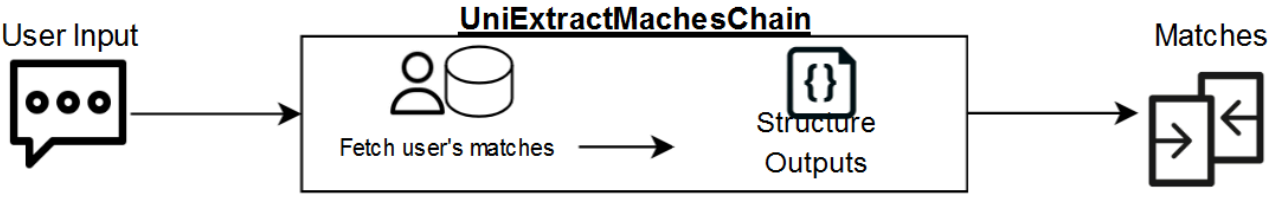
UniMakeMatchesChain: Given user, create a list of possible matches, taking into consideration its input and its preferences



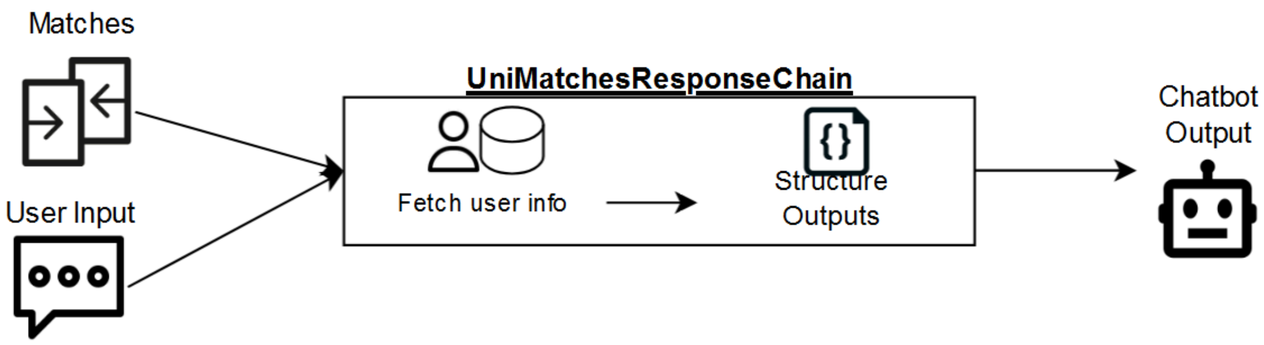
UniAddMatchesChain: Given `Matches`, add the data into the `Matches` table of the Database through SQL queries and delete the old ones, implementing some sort form of memory about user's potential matches.



UniExtractMatchesChain: Given user, extract its matches if they exist, as a list of dictionaries.



UniMatchesResponseChain: Given `Matches`, structure the output as a numbered list containing information about universities and courses. User input can help the bot to structure the answer in a more precise way.



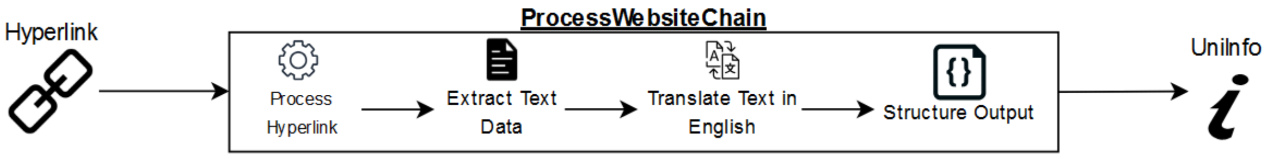
Example:

```
(User with registered preferences in mathematics)
> Can you reccomend me some bachelor's courses, involving topics about AI?

(Chatbot)
Considering your interest in mathematics and your desire to pursue bachelor's courses involving topics about AI, here are some university programs that integrate both fields:
1. **Carnegie Mellon University**: Bachelor of Science in Artificial Intelligence
2. **Indiana University**: Bachelor of Arts in Artificial Intelligence
3. **University of Nebraska Omaha**: Bachelor of Science in Artificial Intelligence
4. **Esade Business School**: Bachelor in Artificial Intelligence for Business
5. **IE University**: Bachelor in Computer Science and Artificial Intelligence
These programs offer a strong foundation in AI, complemented by mathematical coursework, aligning well with your interests.
```

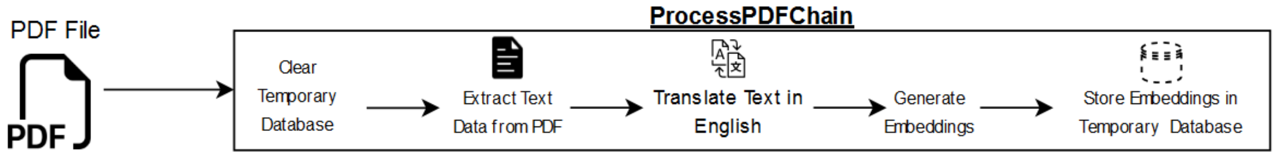
Process Websites and User-loaded PDFs

ProcessWebsiteChain: Given user input containing a link to a university or course website, process the website and extract information with web scraping tools



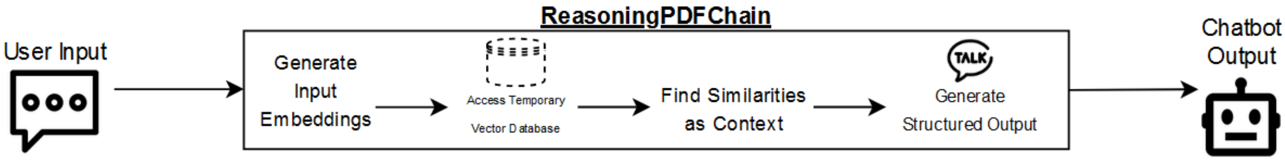
- *Note: this will process only the given link and won't explore eventual inner links, in order to prevent excessive token usage and potential harmful "injections". If an user needs to process more links, they will have to provide them as separate inputs*

ProcessPDFChain: Given one (or more) PDF file(s) uploaded by the user, process it and store it in a temporary vector database. Partially leverages RAG as it involves using a vector database in Pinecone.


PDF File
PDF
ProcessPDFChain
Clear Temporary Database
Extract Text Data from PDF
Translate Text in English
Generate Embeddings
Store Embeddings in Temporary Database

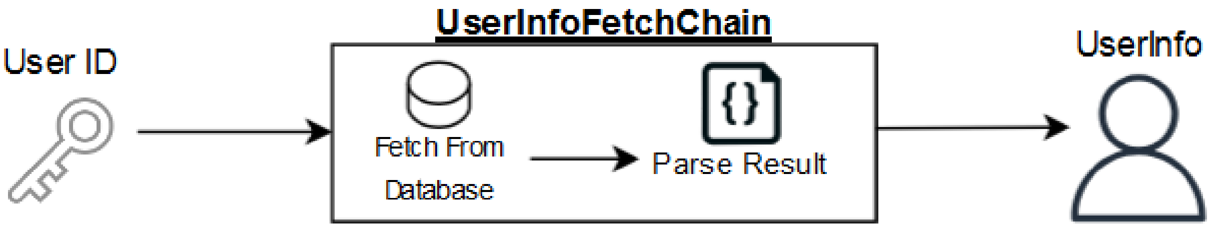
- *Note: if there are more users using the same vector database at the same time, it will be necessary to implement process lock methods. For the sake of simplicity, we will assume that only one user uses this feature at a time.*

~~**ReasoningPDFChain**~~: Given the temporary vector database for PDFs and some user inputs (or default queries), extract and process relevant information as `UniInfo` instances. Partially leverages RAG as it involves using a vector database in Pinecone.

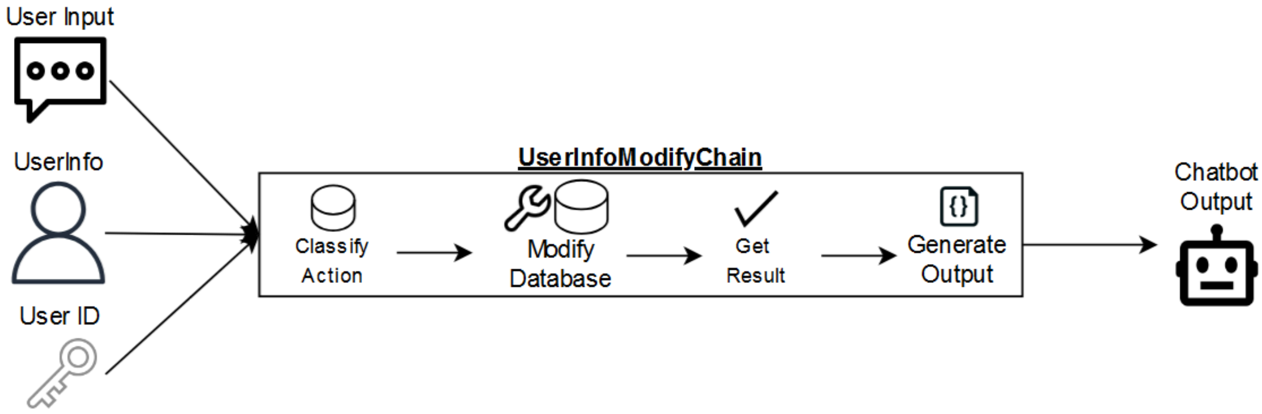

ReasoningPDFChain
User Input
Generate Input Embeddings
Access Temporary Vector Database
Find Similarities as Context
TALK
Generate Structured Output
Chatbot Output

Manage Personal Information

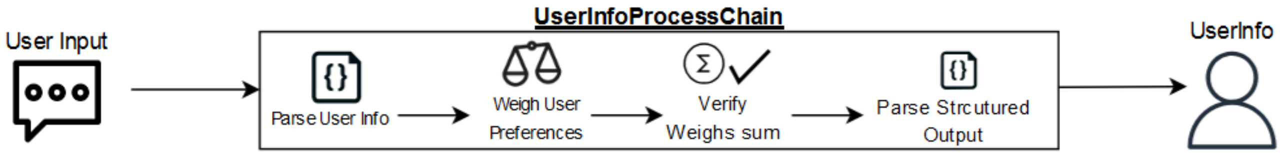
~~**UserInfoFetchChain**~~: From an user ID, extract the user's preferences from the database and parse the result in a structured output.



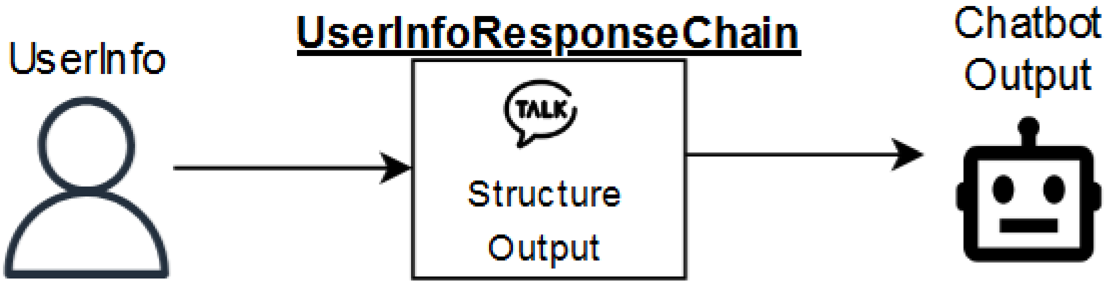
UserInfoModifyChain: From user input, parsed information about the user, modify the SQL database. It can add more entries, or remove some of them, or modify some of the already-existing entries. Outputs the result, i.e. whether an operation had success or not.



UserInfoProcessChain: Parse an user input to extract information about the user's preferences, so they can be eventually added, removed or modified from the database.



UserInfoResponseChain: Parse an user's information to chatbot output, so the user can know what the bot knows about the user.



X

Agents

PersonalInformationManager is the agent which handles intentions for managing user's personal information (namely preferences and areas), and is able to do *DML* operations on the database.

- Chains: UserInfoFetchChain, UserInfoModifyChain, UserInfoProcessChain, UserInfoResponseChain
- Tools: LangChain's built-in SQL DML tools (SQLDatabaseToolkit)