ATAI - Report - Final

Christian Skorski, 18-700-284 Konstantin Moser, 17-706-169

December 11, 2023

1 Introduction

Our agent uses multiple techniques learned in the course to answer questions or show pictures in the domain of movies, and give matching movie recommendations for the user. We use three methods to reply to closed questions: Knowledge graph querying, cross-checking of the former with crowd-sourced data, and embeddings. We use singular value decomposition on a user ratings dataset (strictly a subset of the movies in the provided data) and simple genre- and release-based filtering to recommend movies to the user, based on the liked movies they shared in their question to the agent. The whole system is designed and built using the object-oriented architecture, making it well-organized, readable, and easily maintainable.

2 Expected Input & Output

2.1 Querying

The agent initially attempts to find specific movie information by executing SPARQL queries on the graph. If an answer is found from these queries, the agent does not proceed to search for an answer using embeddings.
Input: Who is the screenwriter of The Shawshank Redemption?
Output: The screenwriters of The Shawshank Redemption are Frank Darabont and Stephen King.

2.2 Embeddings

If the agent can't find an answer with queries, it will try to find one using embeddings.
Input: Who is the screenwriter of The Masked Gang: Cyprus?
Output: The answer suggested by embeddings: Melih Ekener, Murat Aslan, Mehmet Ali Erbil, Cengiz Küçükayvaz.

2.3 Multimedia

If the user asks to see an image of an actor or a movie the agent will provide up to three pictures to answer the question.
Input: What does Leonardo DiCaprio look like?
Output:

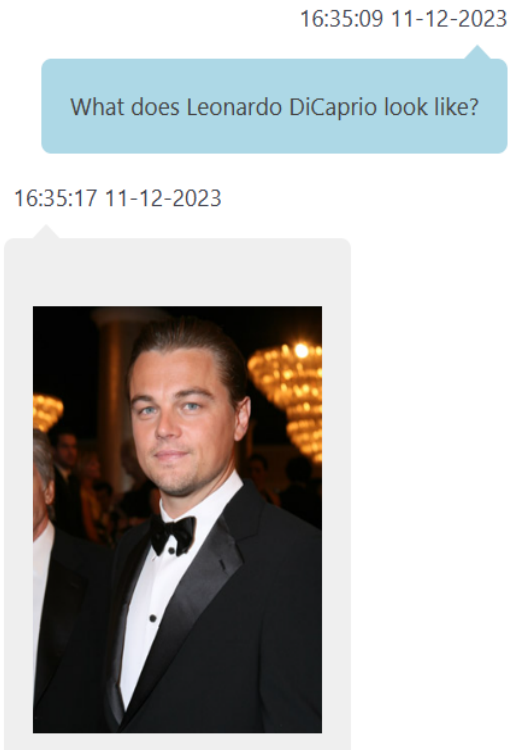

Figure 1: The agent showing the user what Leonardo DiCaprio looks like.

2.4 Recommendations

Input: Given that I like Inception and Interstellar, what could I watch today?
Output: Based on your liked movies, I think you might also like Sherlock: The Abominable Bride (2016), Source Code (2011), Girl Who Played with Fire, The (Flickan som lekte med elden) (2009), The Double (2011), Jack Reacher: Never Go Back (2016), and Blackhat (2015).

3 General Workflow

The agent's general workflow for generating responses to user questions involves several key steps:

Question Classification: The agent begins by classifying the type of the user's question using a zero-shot classifier. This helps to determine if the question is about movie recommendations, specific movie information, or multimedia content like actor images or movie scenes.

Multimedia Questions Handling: If the question is identified as a multimedia question, the agent further classifies it to understand whether it's asking for movie scenes, posters, behind-the-scenes images, or actor pictures. For movie-related images, the agent extracts the movie title from the question, retrieves the movie's IMDb ID, and then finds the relevant images from the database based on the specified type (still frames, posters, or behind-the-scenes). For actor images, the agent

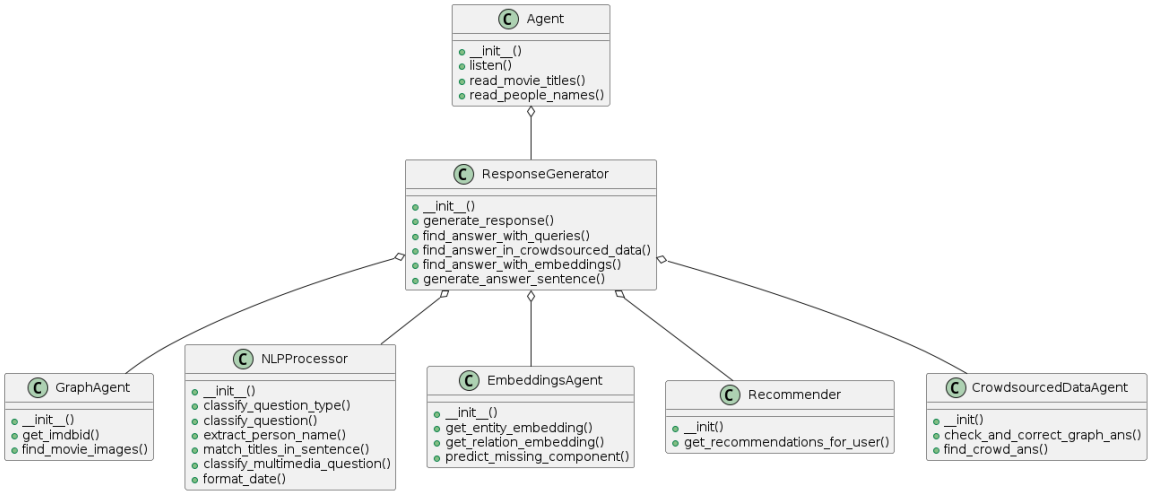

Figure 2: Class diagram of our final bot

extracts the actor's name, gets the IMDb ID, and retrieves images where the actor is the only one listed in the cast.

Movie-specific Questions: If the question asks for specific information about a movie, the agent identifies the intent of the question (like director, release year, etc.). The agent first tries to find answers through SPARQL queries on the graph database. If an answer is found, it stops there. If the query doesn't yield an answer, the agent then attempts to find an answer using embeddings.

Movie Recommendations: For recommendation-related questions, the agent identifies movies mentioned in the question and treats them as highly rated by the user. The agent then uses these ratings to generate movie recommendations, leveraging the SVD algorithm.

Response Generation: The agent generates a natural language response based on the findings from the above steps. If no information is found, the agent politely informs the user that it couldn't find the necessary information. Throughout this process, the agent utilizes various libraries like Rdflib for graph database queries, Scikit-Learn for machine learning models, and RapidFuzz for fuzzy matching of movie and actor names.

4 Object Oriented Design

4.1 GraphAgent (graph_agent.py)

- Executes SPARQL queries, focusing on movie data retrieval from Wikidata.
- Runs diverse SPARQL queries for flexible interaction with graph databases.
- Offers methods for getting entity URLs, details by label, and names from IDs.
- Designed for easy extension to support more query types and data schemas, highlighting adaptability.

- Offers the functionality to accurately retrieve the IMDb IDs of actors and movies. These IDs are then used to filter and select relevant images from the 'images.json' database.
- The agent includes methods designed to return appropriate images in response to user queries. These methods locate and provide images that match the content requested by the user.

4.2 NLPProcessor (nlp_processor.py)

- Manages natural language processing (NLP) tasks for identifying movie-related entities in text and classifying the type of information requested about movies.
- Utilizes pre-trained transformers for classification and RapidFuzz for entity recognition.
- Classifies the type and intent of a question using zero-shot classification.
- Can identify movie titles and actor names within a sentence with RapidFuzz even if they are misspelled.

4.3 EmbeddingsAgent (embeddings_agent.py)

- Manages embeddings for entities and relations in RDF triplets.
- Provides functionality to retrieve embeddings for specific entities and relations.
- Capable of predicting missing components in RDF triplets based on available embeddings.

4.4 Recommender (recommender.py)

- Initializes by loading and preprocessing movie and rating data to facilitate recommendation generation.
- Utilizes the SVD algorithm from the Surprise library to fit a model to the ratings data.
- Generates movie recommendations based on user preferences, filtered by similarity in genres and release years.

4.5 ResponseGenerator (response_generator.py)

- Generates natural language responses to user queries. It integrates outputs from the instances of GraphAgent, NLPProcessor, EmbeddingsAgent, and Recommender.
- The `generate_response` method if responsible for the workflow by classifying the user's question, identifying movie titles, and obtaining information through graph queries and embeddings.
- Capable of handling different types of queries including recommendations or specific movie information requests.
- Uses the `generate_answer_sentence` method to construct well-formed responses based on the query results and user's question intent.

5 Recommender

The Recommender class uses singular value decomposition on the MovieLens user ratings dataset and simple genre- and release-based filtering to recommend movies to the user, based on the liked movies they shared in their question to the agent. As discussed with the teaching assistants, the recommended movies are strictly a subset of the movies in the provided data. The recommender first initializes movie ratings and titles. After getting a question from the user, it gets converted into a list of tuples with 'title' and 'score', where the latter is set to 5.0. It is expected that the user only asks for recommendations based on movies he likes. The recommender then uses the Surprise library's Singular Value Decomposition (SVD) algorithm to fit the SVD model to the ratings data extended with the user's. Afterwards, it generates personalized movie recommendations based on user preferences, applying filters for genres and release years to tailor suggestions.

6 Crowd-sourced Data

The crowd-sourced data (CSD) agent has two public methods: One for correcting information found in the knowledge graph, and one for finding information not found in it. There are the following cases:

- An answer (object in the triple) is found in the KG:
  - A matching triple is found in the CSD, and it is correct. Return true for correctness to the response generator.
  - A matching triple is found in the CSD, but it is incorrect. Return false for correctness and the corrected triple if it is given by the raters.
- No answer (object in the triple) has been found in the KG:
  - An answer is found in the CSD, and it is correct. Return it to the response generator.
  - An answer is found in the CSD, but it is incorrect: If the KG subject and predicate are right but the Object is not, return the Object corrected by the raters.
  - An answer is found in the CSD, but it is incorrect. If the object is right but the subject or predicate is not, we do not return an answer from the crowd-sourced data. We could potentially match subject-predicate tuples also to corrected labels, or just add these as 'CORRECT' rows, but we found this too laborious for the scope of our project.
  - No answer is found in the CSD. Return nothing.

7 Limitations

7.1 Knowledge Graph Querying

If the user is asking for a specific information about a film the agent is limited to the following topics: director, screenwriter, release year, producer, director of photography, original language, genre, actor, costume designer, film editor, nominated for, based on, country of origin, filming location, award received and box office. This limitation arises because we're using a zero-shot classifier to determine the user's query intent. By restricting the zero-shot classifier to specific cases, we aim to minimize misclassifications.

7.2 Multimedia

The agent can display images of actors, movie still frames, behind-the-scenes movie set photos, and movie posters. When asked for an actor's images, it shows the first three found in the dataset. For movie scenes, it initially searches for 'still_frames'; similarly, it looks for specific types for posters and behind-the-scenes images. If no 'publicity' type images are available, it considers other types. However, it does not handle more specific queries, such as images of multiple actors or a specific actor in a particular movie.

7.3 Embeddings

The embeddings agent initially searches for entities with a cosine similarity below 0.86. If none are found, it then returns the entity having the smallest pairwise distance to the predicted location. Therefore, when searching for answers using embeddings, there is a possibility of retrieving incorrect answers.

7.4 Recommendations

The recommender's limitation is its rather small dataset, reducing the variety of movies it can suggest. This migth limit the system's ability to take into account a wider range of user preferences. However, despite its limited dataset, the recommender performs well by focusing on more famous movies.

Another limitation is that although

7.5 Crowd

8 Libraries

- **Python:** Serves as the foundational programming language.
- **Numpy:** Used across various classes for numerical computations, particularly in handling array operations.
- **Cosine Similarity:** Initially employed by the EmbeddingsAgent, this method searches for entities in the knowledge graph with a cosine similarity below a threshold to the input vectors. This approach helps in finding the most closely related entities or movie titles based on their vector representations.
- **Pairwise Distances:** Also utilized by the EmbeddingsAgent when cosine similarity does not yield suitable results. It calculates the distance between pairs of vectors, aiding in identifying the closest entity or relation in the knowledge graph relative to the predicted location, based on the embeddings of the other components.
- **Rdflib:** Utilized for querying and manipulating RDF graph data.
- **Scikit-Learn Surprise:** Utilized for the SVD algorithm to generate recommendations for the user.
- **RapidFuzz:** Used to search for movie and actor names in the graph, particularly useful in cases of misspellings. It enhances the system's ability to accurately match user queries to movie titles or actor names.

- **Transformers:** Utilized for classification tasks in the NLPProcessor. Enables a pipeline to the Bart large language model to classify the intent of the user input.
- **Pandas:** Utilized by the Recommender and Crowdsource Class for data manipulation and analysis, mostly for handling tabular data.
- **Statsmodels:** Employed in CrowdsourcedDataAgent for statistical analysis, like calculating inter-rater agreement.

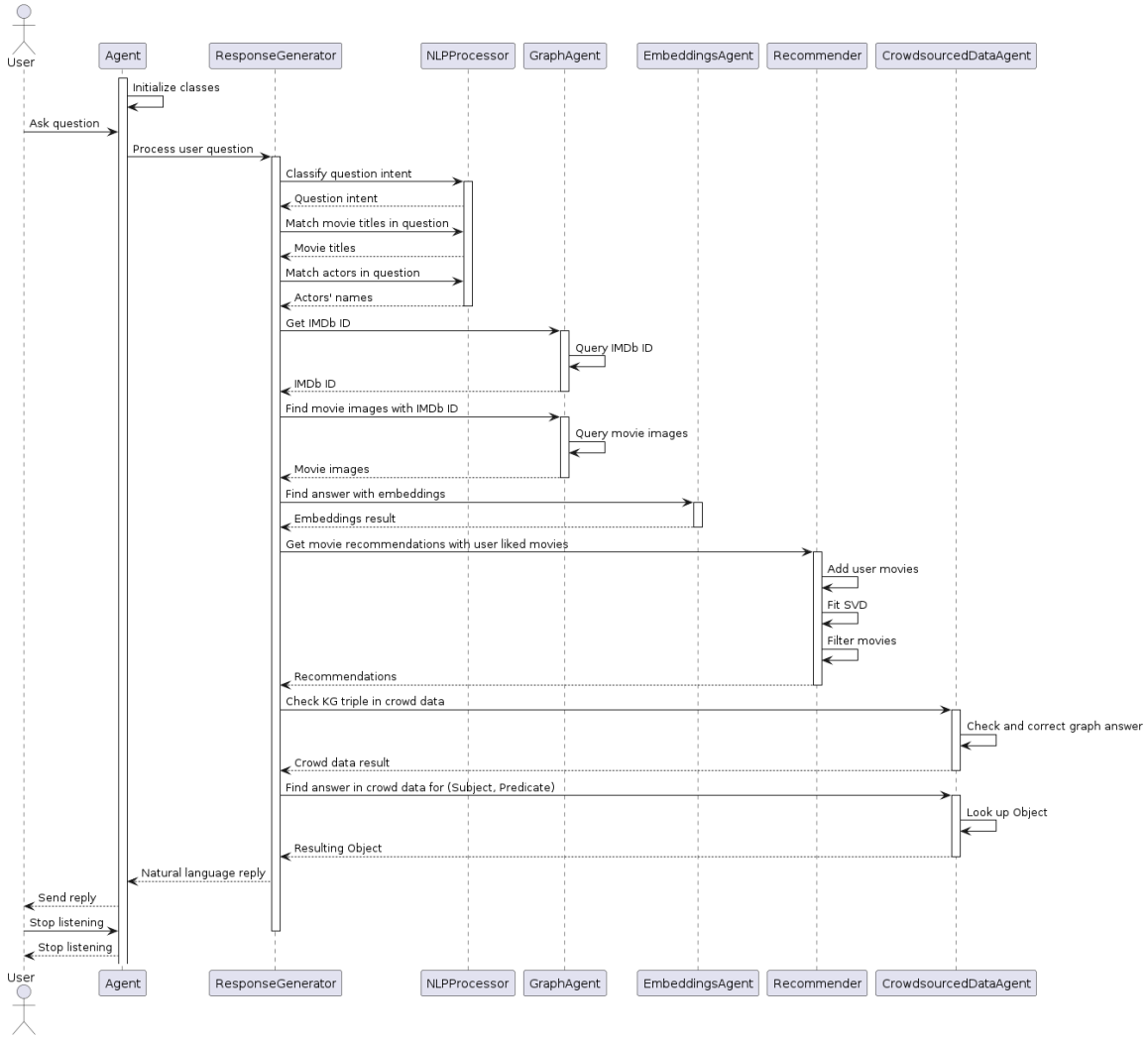

Figure 3: Sequence diagram of our bot's workflow.