Evaluation Event Final - *red-dragon*

Yuxuan Wang, Ömer Yildirim

December 8, 2024

1 Introduction

The red-dragon agent is a movie-focused chatbot designed to answer factual, embedding, recommendation, multimedia, and crowdsourcing questions. It integrates several key components: a fine-tuned Named Entity Recognition (NER) module to identify movie or person titles from user input, a question classifier to determine the user intent, and integration with multiple data sources including knowledge graph, embedding space, movie net and crowdsourcing to retrieve accurate information. The system then employs a response generator to deliver human-like answers. For close-form questions, the agent will provide human-like answers from a knowledge graph or embedding-based techniques with similarity calculations. Additionally, it provides movie recommendations using clustering and nearest-neighbor approaches. Multimedia requests are supported by matching entities to their corresponding images. The agent also employs crowdsourced data with inter-rater agreement scores to augment the knowledge graph, providing accurate and transparent answers.

2 Capabilities

The red-dragon agent consists of the following components:

- **NER Module**: The Named Entity Recognition (NER) module combines a fine-tuned bert-base-NER [1] and fuzzy matching techniques to extract entities from user queries. In cases where the NER model has low confidence or user input contains typos, fuzzy matching is employed to find the closest matching entities from the knowledge graph. This dual approach ensures robust and accurate entity recognition, which is the foundation for subsequent processing modules.

- **Question Classifier**: A pre-trained Support Vector Machine (SVM) classifier [2] to classify user questions into 3 categories: factual, recommendation, and multimedia. The classification pipeline can accurately distinguish different types of questions, so the agent can process user requests with the appropriate module.

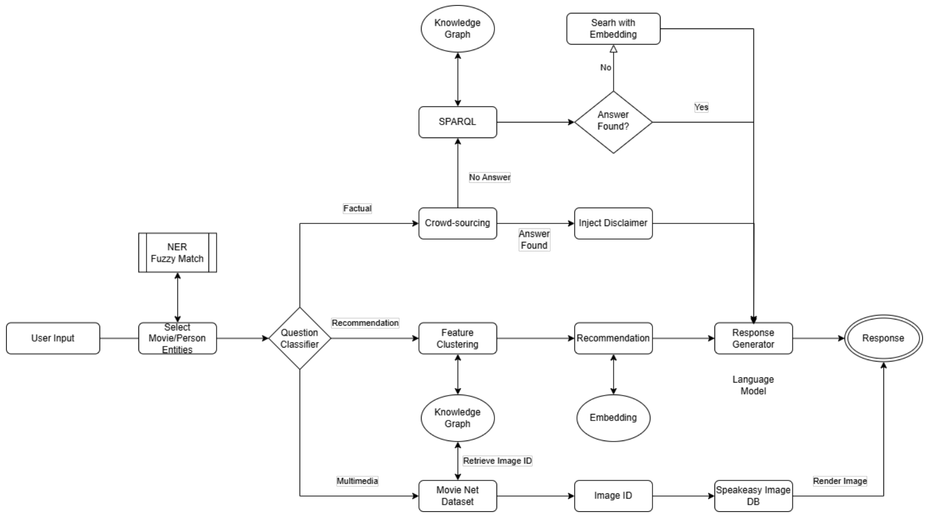

Figure 1: Response Generation Pipeline of the Agent

- **Knowledge Graph Integration with SPARQL Query**: Constructs SPARQL queries [3] based on recognized entities to retrieve relevant information from the movie knowledge graph. Format the retrieved information for the Response Generator.
- **Embedding Space**: The embedding space module allows the agent to answer questions using embedding-based reasoning, particularly when SPARQL queries are insufficient. It computes combined embeddings for movies and relations, identifying the best match based on similarity. This module also supports similarity-based queries for recommendations and attribute-specific analysis, enhancing the agent's flexibility and accuracy.
- **Recommendation**: This module handles the logic related to movie recommendations, it interacts with Embedding Handler and SPARQL Query Processor to get necessary data, and eventually recommends movies based on recommendation algorithms.
- **Multimedia**: Handles requests for images related to movies or cast members by extracting entities, retrieving corresponding IMDb IDs, and searching a pre-loaded image dataset and returns an image ID that can be recognized and displayed by the Speakeasy system.
- **Crowdsourcing**: Integrates crowd-sourced data to supplement or override knowledge graph results. It identifies matching entities and relationships in a pre-processed, cleaned dataset, returning answers along with transparent disclaimers on the inter-rater agreement.
- **Humam-like Response Generator**: This module is the final output

layer, it formats the retrieved information using Llama 3.2 [4] language models to generate human-like responses to the user.

2.1 Workflow Details

- **Factual Questions**
  The agent provides precise answers by querying the knowledge graph and formatting responses in a natural, human-like manner, with the following three steps:

  1. In the first step, we employ a fine-tuned bert-base-NER [1] to recognize entities like movie titles from the user input. We have tuned the bert-base-NER based on our movie dataset to improve accuracy. Additionally, it performs a fuzzy search [5] to improve the matching accuracy of the movie titles with those in the knowledge graph.

  2. In the second step, the matched title is used to construct a SPARQL query to retrieve relevant movie information such as release dates, directors, awards, etc.

  3. In the final step, the retrieved and formatted information is fed into the Llama 3.2 [4] language model to generate a natural language response that provides the answer to the user in a concise format.

- **Embedding Questions**
  When a SPARQL query does not return any results, the agent utilizes an embedding-based approach to answer the question. This process involves the following four steps:

  1. In the first step, same as the factual questions, we make sure we have identified the movie entity by NER and fuzzy search.

  2. In the second step, we will use perform relation extraction. The user's query is processed to determine the intended relation (e.g., director, release date), and match the user intent to a specific relation in our dataset.

  3. In the third step, we will perform a combined embedding calculation. We compute a combined embedding to represent the query (movie + relation).

  4. In the final step, the combined embedding is used to find the closest match among all entity embeddings using pairwise distances. The entity with the highest similarity score is selected as the answer.

- **Recommendation Questions**
  The agent answers recommendation questions by leveraging both the Knowledge Graph and embeddings. It uses the following workflow: 1. In the first step, similar to factual and embedding questions, we use the fine-tuned NER model and fuzzy search to extract movie entities from the user's query.

2. In the second step, the agent retrieves relevant features of the movies (e.g., director, production company, or genre) using SPARQL queries to the knowledge graph. These features provide the basis for finding similar movies. Additionally, the system uses a whitelist of meaningful relations (e.g., director, screenwriter, genre) to ensure relevance.

3. In the third step, the agent employs two methods for generating recommendations:

- **K-means Clustering**: Features of the liked movies are represented in embedding space. The embeddings are clustered using K-means to identify significant clusters, and the features closest to the cluster centroids will be selected and movies close to these features will be recommended.

- **KNN-Based Similarity**: The agent calculates the mean embedding of the liked movies and finds movies with the K smallest pairwise distances to this mean embedding in the embedding space. This ensures that the recommended movies are semantically similar to the liked ones.

4. In the final step, the system filters and ranks the recommended movies to remove duplicates and irrelevant results. The features are also used to explain what the recommendation is based on.

- **Multimedia Questions**
  The agent is capable of answering multimedia questions such as movie posters and images of cast members, with the following workflow:

  1. In the first step, the system uses the NER module to extract movie and person entities from the user query. It uses both fine-tuned and base NER pipelines to extract movie entity and person entities, ensuring entity extraction accuracy.

  2. In the second step, the recognized movie or person entity is matched to the IMDb ID stored in the knowledge graph. This step involves querying the graph using a SPARQL template to retrieve the corresponding IMDb ID.

  3. In the third step, the system uses the IMDb ID to locate image assets in a pre-loaded dataset of images (images.json). For movies, it prioritizes posters, and then publicity images. For persons, it prioritizes event images, followed by publicity images.

  4. In the final step, the retrieved image ID is formatted into a response string that the speakeasy platform can recognize, so that it can show the image to the end user. If no image is found, the system informs the user accordingly.

- **Crowdsourcing Questions**
  The agent also integrates with a crowdsourcing dataset to augment the knowledge graph, offering reliable answers supported by community-driven data, which involves the following four steps:

1. In the first step, the agent extracts the movie entity from the user query using the NER module and matches it to an entity in the knowledge graph.

2. In the second step, it attempts to match the user query intent to a relationship (e.g., director, box office) presented in the crowdsourcing dataset. If a match is found, the system checks the crowd-sourced dataset for relevant answers.

3. In the third step, if a valid match is found in the crowd-sourced dataset, the chatbot will prioritize the crowdsourcing answer and override the result of the knowledge graph. If no match is found, the system proceeds to use the knowledge graph or embedding-based methods.

4. Additionally, the system includes a disclaimer summarizing inter-rater agreement and the vote distribution for the selected answer, enhancing transparency.

3 Adopted Methods

- BERT-base-NER [1]
  BERT-base-NER is a BERT model specialized in Named Entity Recognition and achieves good performance on the recognition task. It plays a crucial role for our agent to identify the movie that the user is mentioning, which is the basis for future steps. Additionally, we have generated a labeled dataset containing movie names and fine-tuned the model. The fine-tuned model is trained by a labeled dataset of movie titles from knowledge and demonstrates high accuracy in recognizing movie names.

- **TheFuzz for Fuzzy Matching**
  To improve entity matching accuracy, we use TheFuzz [5] to find the closest match for movie titles for the recognized movie within the knowledge graph, ensuring more precise retrieval of information.

- **SPARQL Query Execution with rdflib**
  RDFLib [6] is a Python package for working with RDF. It is used to interact with the knowledge graph and execute SPARQL queries to retrieve structured data for the identified movie entities.

- **Meta Llama/Llama-3.2-1B**
  The Llama-3.2 model [4] is an open-source language model available on Hugging Face . We use this model to generate natural language responses based on the information retrieved from the knowledge graph. The reasoning and language capabilities of the model made it well-suited for generating user-friendly answers in cases where hardcoded logic could be limited.

- **SVM-based Question Classification** To determine the type of user query (factual, recommendation, or multimedia), we employ a Support Vector Machine (SVM) classifier [2]. The training dataset, generated and labeled with the assistance of ChatGPT, provides a diverse range of movie-related questions. We use TF-IDF vectorization to transform the text into a suitable numerical representation, and then train an SVM model on these features. This approach results in a robust classification pipeline that can accurately distinguish among query categories, thereby directing the user request to the appropriate processing module.

- **K-Means Clustering** [1]
  K-Means clustering [7] is used in the recommendation system to group common features (e.g., directors, production companies) extracted from liked movies into clusters. The centroid of each cluster represents a shared characteristic, and movies close to these centroids are selected as recommendations. This method ensures that recommendations are contextually aligned with user preferences.

- **K-Nearest Neighbors (KNN)** [2]
  K-Nearest Neighbors [8] is used to recommend movies based on the proximity of their embeddings to the combined embedding of the user-provided movies. By computing pairwise distances between embeddings, the system identifies movies that are most similar to the user's input. KNN provides robust recommendations by leveraging semantic similarity in the embedding space.

- **Fleiss Kappa Inter-rater Agreement Score**
  We employ Fleiss Kappa inter-rater agreement score [9] to ensure the reliability of crowd-sourced answers. This metric measures the consistency among multiple annotators who provided judgments on the correctness of the information. By calculating the support and reject votes for each movie and relation pair, the agent determines the level of agreement between contributors. The resulting score enables the system to include transparent disclaimers in its responses, improving the overall quality of the agent's answers.

4 Examples

- **Factual Questions**
  Given the question "When was "The Godfather" released? ":

  1. The agent first extracts the name "The Godfather" using fine-tuned name entity recognition based on the provided knowledge graph.
  2. It then constructs a SPARQL query to retrieve the release date from the knowledge graph.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`
[2] `https://scikit-learn.org/stable/modules/neighbors.html`

3. Finally, it uses the Llama 3.2 model to generate the response: "The Godfather was released in 1972." This is correct, as The Godfather was released on 1972-03-15 according to the knowledge graph.

- **Embedding Questions**
  Given the question "Who is the director of The Masked Gang: Cyprus?":

  1. The agent first recognizes "The Masked Gang: Cyprus" using fine-tuned name entity recognition based on the provided knowledge graph.
  2. It extracts the intended relation "director."
  3. Next, it calculates the pairwise distance in the embedding space to find the closest match.
  4. The result is "Cengiz Küçükayvaz," which is correct as the provided embedding answer.

- **Recommendation Questions**
  Given the query "Given that I like The Lion King, Pocahontas, and The Beauty and the Beast, can you recommend some movies? ":

  1. The agent first recognizes "The Lion King", "Pocahontas", and "The Beauty and the Beast" using fine-tuned name entity recognition based on the provided knowledge graph.
  2. It extracts relevant features for the provided movie, such as "Walt Disney Pictures," "Linda Woolverton," by querying the Knowledge Graph with SPARQL. These features are used as the basis for recommendations.
  3. The agent then calculates the mean embedding of the movies and features recommend movies based on them
  4. The recommendations are filtered and ranked to ensure relevance and avoid duplicates. The example result as follows: *"Adequate recommendations will be related to "Walt Disney Pictures," "Linda Woolverton,", therefore we recommend the following movies: The Lion King, Aladdin and Tarzan"* which is correct as they are all Disney movies with similar topic and production years.

  Multimedia Questions
  Given the question "Show me a picture of Halle Berry.":

  1. The agent first recognizes "Halle Berry" as a person entity using name entity recognition.
  2. It then employs the person entity with the SPARQL template, and retrieves the person ID for "Halle Berry" from the knowledge graph.
  3. Using the person ID, it finds the corresponding poster or publicity image of "Halle Berry" in the dataset.

4. The response is formatted as: “image:id”, the speakeasy platform will render the corresponding picture based on the id, which is expected as the user is querying the image of ”Halle Berry”

Crowdsourcing Questions
Given the question “What is the box office of The Princess and the Frog?”:

1. he agent first recognizes ”The Princess and the Frog” as a movie entity using fine-tuned name entity recognition based on the provided knowledge graph.
2. It then identifies the relationship ”box office” from the user query intent.
3. The agent then matches entity ”The Princess and the Frog” and relation ”box office” in a pre-processed and cleaned crowd-sourced dataset, where a malicious user answer has been removed, and inter-rater agreement score has been calculated using Fleiss Kappa score.
4. When the response is found, the result is : “The box office of The Princess and the Frog is 267000000. [Crowd, inter-rater agreement 0.236, The answer distribution for this specific task was 2 support votes, 1 reject vote]”, which is correct based on the given crowd-sourcing dataset and it overrides the answer from knowledge graph.

5 Additional Features

- **Fine-tuned bert-base-NER**
We fine-tuned the BERT-base NER model to improve its accuracy in recognizing movie-related entities. The BERT base model struggled to recognize movie titles effectively. To resolve this problem, we generated a labeled dataset with 60,000 entries, containing all the movie names from the provided knowledge graph, and built the pipeline to fine-tune this NER model. The fine-tuned BERT-base NER model performs extremely well on the movie dataset. It can accurately recognize movie names from input sentences.

- **Fuzzy matching**
The red-dragon agent employs fuzzy matching to further improve the accuracy of NER. It fuzzy matches the NER result with movie dataset and user query input. The fuzzy match finds top matched entity and handles user typos efficiently.

- **Intermediate Responses**
The agent provides intermediate responses, such as ”Still working on it,” when the LLM generation is slow, improving user experience by keeping the user informed.

- **Graph Cache**
  We implemented a mechanism to cache the knowledge graph into a binary pickle file, this significantly reduced the time for initiating graph load, which makes it more efficient during development and production.

- **K-Nearest Neighbors (KNN) for Embedding-Based Recommendations**
  The recommendation system also uses KNN to identify movies semantically similar to those the user likes. By computing pairwise distances in the embedding space, the system ensures that the recommendations closely match the user's interests.

- **Explainable Feature-Based Movie Recommendations**
  Our recommendation system employs K-Means clustering to extract features such as directors, production companies, and genres to generate personalized movie suggestions. This feature enhances user trust and satisfaction with the recommendations.

6 Conclusions

The red-dragon agent effectively answers factual, embedding, recommendation, multimedia, and crowdsourcing questions related to movies by integrating multiple NLP technologies and leveraging a knowledge graph for accurate information retrieval. The system demonstrates robust performance in extracting entities, understanding user queries, and generating concrete and human-like responses.

This project is a collaborative joint effort by our team. Yuxuan Wang focused on the fine-tuning of the Named Entity Recognition module and implementation for factual, embedding, and crowdsourcing questions. Ömer Yildirim Led the implementation of the question classification model, recommendation system, and multimedia questions. Furthermore, our team set up a collaborative repository and adopted development policies with high standards. All changes to the main branch were managed via pull request, requiring approval from another team member. This ensures the code quality and maintainability of the project.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[3] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008.

[4] Meta. meta-llama/llama-3.2-1b, September 2024.

[5] Adam Cohen. Thefuzz: Fuzzy string matching in python, 2021. Version 0.19.0.

[6] RDFLib Developers. Rdflib: A python library for working with rdf, 2024. Version 7.1.1.

[7] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[8] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[9] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.