Evaluation Event II - *red-dragon*

Yuxuan Wang, Ömer Yildirim

October 28, 2024

1 Introduction

The red-dragon agent is a movie chatbot designed to answer factual and embedding questions. It provides interactive responses about movies with Natural Language Processing (NLP) techniques and knowledge graph queries. The agent can perform Named Entity Recognition (NER) to identify key entities such as movie titles and get the relevant data to generate informative responses. It can also provide intermediate responses to improve the user experience if the processing time is too long

2 Capabilities

The red-dragon agent consists of the following components:

- **NER Module**: Uses fine-tuned bert-base-NER [3] to extract entities from user queries, specifically focusing on movie names.
- **SPARQL Query Processor**: Constructs SPARQL queries [4] based on extracted movie names to retrieve relevant information from the movie knowledge graph. Format the retrieved information for the Response Generator.
- **Response Generator**: Formats the retrieved information using RedPajama [2] language models to generate human-like responses to the user.

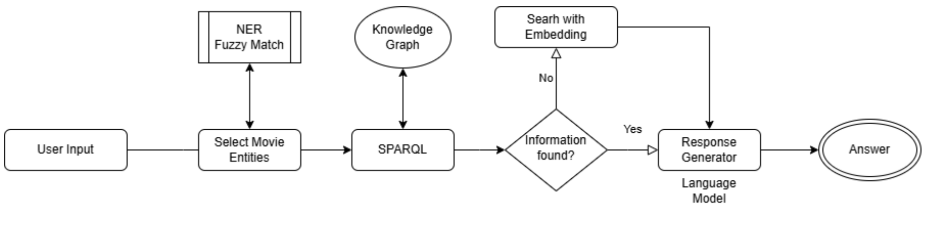

Figure 1: Response Generator Pipeline of the Agent

2.1 Answering Capabilities

- **Factual Questions**
  The agent answers factual questions in three steps:

  1. In the first step, we employ a fine-tuned bert-base-NER [3] 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 [1] 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 RedPajama 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, we use an embedding-based approach to answer the question. This process involves the following 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 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.

3 Adopted Methods

- bert-base-NER [3]
  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 to improve its performance on movie-related questions.

- **TheFuzz for Fuzzy Matching** [1]
  To improve entity matching accuracy, we use TheFuzz to find the clos-

[1]`https://github.com/seatgeek/thefuzz`

est match for movie titles for the recognized movie within the knowledge graph, ensuring more precise retrieval of information.

- **SPARQL Query Execution with rdflib** [2]
  RDFLib 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.

- **togethercomputer/RedPajama-INCITE-Chat-3B-v1**
  The RedPajama model is an open-source language model 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.

4 Examples

- Factual Questions
  Given the question "*When was "The Godfather" released?* " the agent first extracts the name "The Godfather" using NER. It then constructs a SPARQL query to retrieve the release date from the knowledge graph. In the end, it uses the RedPajama model to generate the response: "The Godfather was released in 1972." which is correct as The Godfather was released on 1972-03-15 according to knowledge graph

- Embedding Questions
  For example, given the question "Who is the director of The Masked Gang: Cyprus?". Firstly, "The Masked Gang: Cyprus" will be recognized from NER. Then we will extract the relation "director". In the end we will calculate the pairwise distance in the embedding space and find the result: "Cengiz Küçükayvaz", which is correct as the provided embedding answer.

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 original BERT model struggled to recognize movie titles effectively, so we generated a labeled dataset containing movie names and fine-tuned the model to improve its performance on movie-related questions.

- **Intermediate Responses**
  The agent provides intermediate responses, such as "Still working on it,"

[2] `https://rdflib.readthedocs.io/`

when the LLM generation is slow, improving user experience by keeping the user informed.

- **Custom Stop Criteria for LLM Output**
  Custom-stopping criteria have been implemented to ensure that responses are concise and relevant, avoiding overly verbose or irrelevant information.

- **Double Mechanism**
  The agent also provides a mechanism to output format without using LLM. This is done by Using intent recognition and hard-coded result formatting. This mechanism is currently disabled in code as it cannot handle more complex queries and sometimes misclassifies the intent.

- **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.

6 Conclusions

The red-dragon agent can answer factual and embedding questions related to movies by integrating multiple NLP components and leveraging a knowledge graph for accurate information retrieval. With future development, we would like to focus on the following new features and enhancements:

- Developing a new feature to handle recommendation questions.
- Optimizing inference speed to improve user experience.

References

[1] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, page 73–78. AAAI Press, 2003.

[2] T. Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. 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.

[4] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008.