MOVIE CHAT AGENT

Stefanos Konstantinou, 21-738-778

1 Overview of the agent

In this project a conversational agent has been implemented to answer movie related questions. The personality of the agent is considered to be a rebellious teenager that is to be a bit cranky. Not really into chit-chatting. The agent's behaviour seems NOT not be keen on spelling mistakes e.g., the agent won't respond to a question if it is asked for the "drirektor" of a movie. Moreover the agent expects people's names and surnames with the first character capitalised. But fortunately, the agent is willing to accept the entities (movies, people) to not be exactly written like they are stored in the knowledge graph. Although cranky, the agent is actually very knowledgeable. It is able to know hundreds of thousands of facts, something a human teenager is not.

To get in more technical depth, the agent first tries to identify entities in the given input sentence. The entities recognised by the agent consist of humans with an occupation that is related to movie production and also, movies of various kinds e.g., animated films, feature films... Next, filtering out the entities from the input sentence, the agent tries to identify the relation between the given question and the entities identified previously e.g., searching for a director for a given movie. To elaborate, the relations act as the intentions of the user asking the question. Moreover, an important note is that the relations, that can be identified, are finite and strictly limited to properties of a film, but also the agent is able to find images and make recommendations.

Although being a cranky teenager and being backed by a whole knowledge graph in its head, in case its knowledge retrieves a wrong answer, it is willing to give a listen to the crowd, and attempt to fix itself. Additionally, it is able to make guesses by using embeddings of the entities contained in the knowledge graph. Smart teenage bot!

2 How the five types of questions are solved in the agent

In this assignment, there are 5 types of questions and the agent is able to answer all 5 types of questions.

2.1 Factual Questions

The agent was first implemented to answer factual questions. The agent uses it's knowledge graph to retrieve the URI IDs of the entities and the relations identified by its Deep Learning token classification models. Next the URI IDs are used to run a query in its knowledge graph and retrieve an answer.

2.2 Embedding Questions

The next type of question that the agent was implemented to answer was through embedding questions. In this question the agent is perceived to be making guesses. It considers the hyper-dimensional direction of each entity embedding (by adding the relation factor as well) and uses euclidean distance metric to retrieve the closest neighbouring entities. In general, there has been a difficulty in understanding when to use factual and when to use embeddings, consequently, both are activated always and the embeddings return one answer more than the factual answer. But in case the Factual answer is not there, the embeddings will always be.

2.3 Recommendation Questions

For the recommendation questions, a similar technique to embedding questions is used, but only for movie recommendations. The agent uses embeddings of entities and retrieves the 3 movies that are considered to be most similar. Additionally the agent is able to make recommendations based on genre, and genre-actor. For this, the agent searches for a random movie in its knowledge graph that satisfies the given criteria.

2.4 Multimedia Questions Questions

Basically for this kind of questions the agent is able to respond with images. Important to clarify, if not understood implicitly, the agent is able to distinguish between movies and people. Thankfully, this fact made it easy for the agent to complete this task with no worry. The agent loads up a file with IMDB IDs mapped to particular images. By identifying the IMDB ID of an entity from its knowledge graph, it is able to retrieve the right image and screen it to the Speakeasy API using the API's Codes. For movies, it retrieves movie posters and for people, just any picture it finds that is not a poster.

2.5 Crowdsourcing Questions

Before implementing this type of question, the malicious workers had to be removed (people that tend to respond with no factual integrity). This type of question is answered when the factual result consists of only one answer or no answer. The scenario for only the only one answer was basically because the crowd dataset was not very complete or reliable on multiple answer questions. Whenever the crowd dataset returns an answer that is deemed CORRECT, with a rate of STRICTLY ABOVE 0.5, the answer replaces the factual answer. Although the dataset was cleaned from malicious workers, the validity of many of the answers are still questionable.

3 Adopted techniques

3.1 NER

In order to retrieve entities, 2 pre-trained Flair models [ABV18] for Token classification on NER task were used. One large model for plan A, and one smaller for plan B - back up (in case of mismatching or no identification of entity). Flair models were chosen since they prove to be capable and always near the top if not at the top on the benchmark leader-boards. They perform excellently on token classification due to their character based encoding. Even-though, Flair models are doing an incredible job, they still find it difficult to identify entities (particularly people) when the first character of the entity is not capitalised.

3.2 POS tag

In order to retrieve relations (Intents), a pre-trained Flair model on POS is used. The nouns identified are considered the key words for identifying the relation. Also verbs are converted to nouns e.g., 'directed' $\rightarrow$ 'director'. The nouns are filtered and only the ones relevant to film properties are considered.

3.3 Pattern Matching

Besides the POS tagging, some relations such as 'costume designer' (others aswell) were found using pattern matching. This was done because these kind of relations are made up of more than one words and it would be easy to confuse them with other relations e.g., 'producer', 'executive producer'. Also pattern matching was used to match genres since NER models don't. To keep it more reliable, the agent is able to understand synonyms of genres and relations identified by pattern matching e.g., 'funny movie' means 'comedy move'.

3.4 Bi-Encoder

In order for the agent to identify an entity's URI ID with no need for the input to be exactly the same as the name of the entity in the knowledge graph, a Bi-encoder model [RG19] was used to encode all the entity text names (and loaded at boot up). Each time an entity is identified by the NER model, the entity is encoded by the Bi-encoder model and then euclidean distance is applied on all pre-embedded entity text names, and manages to retrieve all the URI IDs that match the entity. The Bi-encoder uses an MPNet model [STQ+20]. MPNet model is suitable for similarity tasks because of the Permuted Language modelling Task that was pre-trained on.

4 Extra remarks

In the sumbission file, there exists a jupyter notebook file. In there, it contains code that is included AND NOT INCLUDED in the agent's python files. The included code is the code that was developed for the agent functionality (it was built modular). The non included code, are parts (such as: cleaning the crowd data, creating embeddings for entity names, downloading models and store them locally...) that are then loaded in the agent's classes when booted up. To boot up the agent run the 'Stefos_agent.py' file. Make sure to install any library dependencies and Download the models in case they are absent from the 'models' directory.

References

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019.

[STQ+20] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *CoRR*, abs/2004.09297, 2020.