Building an Intelligent Conversational Agent

Kartikey Sharma (20-744-595)

December 14, 2022

1 Introduction

The architecture of the chatbot mainly comprises the following components:

1. **Data Preparation**: Since the dataset also includes turtle files and the triples-like structure, various data frames were created to make life easier and to search for the answers in a more robust manner.
2. **Pre-processing**: Helps to get rid of unwanted/less important words from the question. *SpellChecker()* and *word tokenize* and stopwords from *nltk corpus* are used
3. **Entity Extractor**: Detects the entities such as the movie name and the human name from the question
4. **Property Extractor**: Detects the name of the property (ex. actor, director, nationality, etc)
5. **Intent Detection**: Detects if the question asked is related to the image or the recommendations for a movie or a person. Additional intent decides if the question is asking for the plot of the movie, user comments for a movie or a list of the top 10 IMDb movies.
6. **Sentiment Detection**: Using the *zero-shot-classifier*, the model detects if the user is greeting the chatbot, or if the user liked or disliked the answer based on the replies to the answers. For instance, the bot would understand that the user liked the answer if he/she will write " *amazing*" or "*great answer*" The follow-up answers are curated for each of these sentiments. This is useful in maintaining a nice flow to the conversation.

2 Architecture and Methodology

2.1 Entity Extractor

After the question is received, the pre-processing is done on it. The major component in this part is the use of the transformer models *dslim/bert-base-NER* This NER model provides the label *"PER"* for the person name and *"MISC"* for the movie entities. The scores for an entity belonging to both these classes are also provided. The logic implemented compares the extracted NER class to the movie name/ human name which is extracted separately from the Knowledge Graph (KG). If the scores are similar for both the movie/human, a scores comparison is done. Even after extracting the NER, in some cases movie names are correctly extracted. Hence, *difflib* is used to find the closest entity to the ones I extracted from the graph. The last resort is to apply the *tokenise* and then do the regex substring matching.

2.2 Property Extractor

Once the entities are extracted, the name of the entities are removed from the question and the remaining question is fed to the property extractor function. The intent is then identified of the property. As shown in the figure 1, intent can be the property related to the *location, director(director of photography, art director, etc), rating(FSK, MPAA, etc)* . After the intent is detected, they are mapped directly to the property codes and property names. For the remaining ones, *difflib* is used to match the nearest property from the list of properties.

2.3 Factual Questions(including answers from KG, Crowdsource and Embeddings)

This includes extracting the results from the Knowledge Graph first. If the result is found, then it is compared with the results from embeddings as a cross-validation step. If the result is not found in the KG, the crowdsource dataset is used to find the answer. For curating the crowdsource, *groupby* was used to aggregate the number of correct and incorrect answers. The Fleiss Kappa value is also calculated for the given dataset to find the inter-rater agreement. Lastly, if the KG as well as crowdsource data is unable to extract the answer, TransE embeddings are used to provide the result.

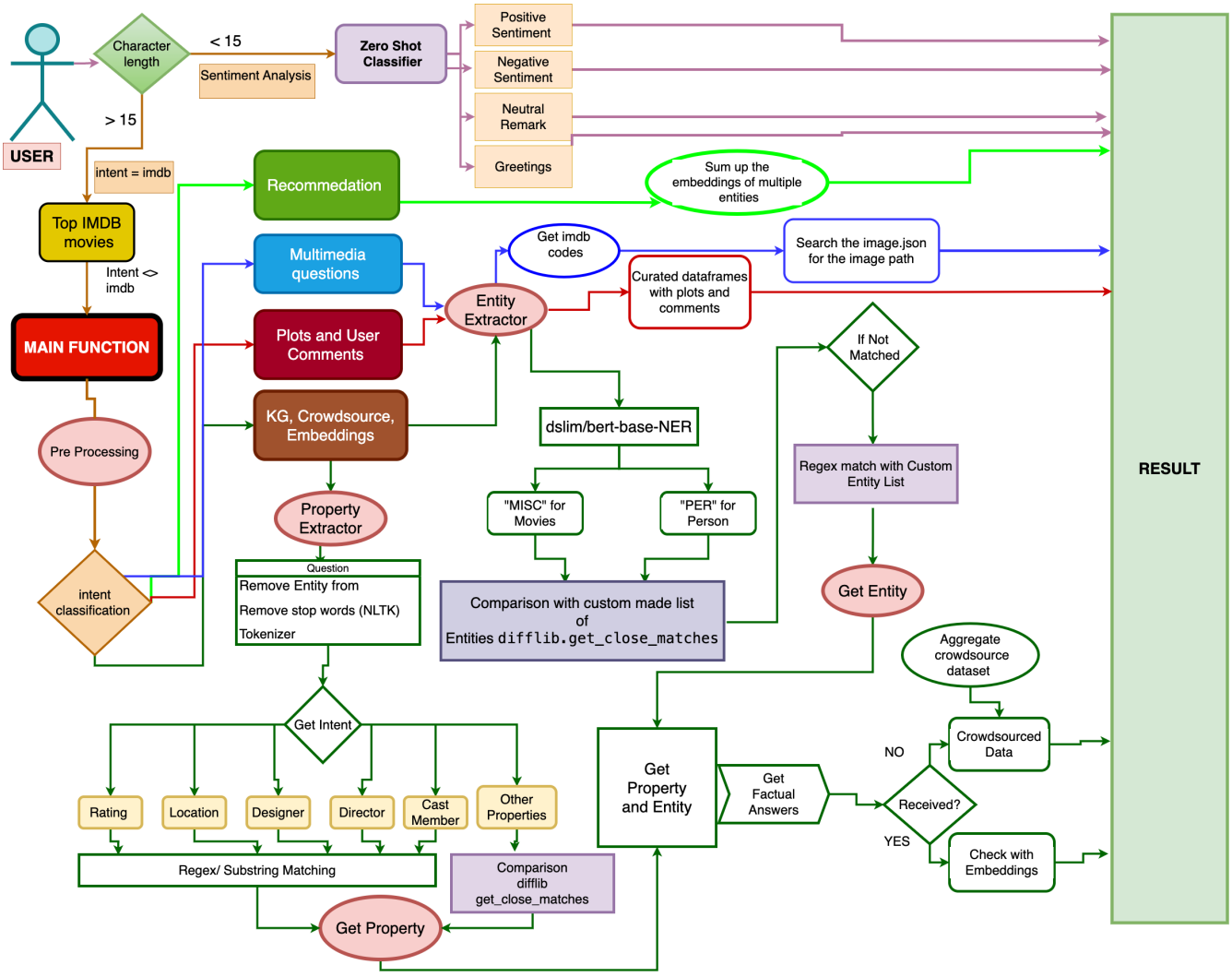

Figure 1: Architecure of the workflow

2.4 Multimedia Questions

The intent is analysed based on some regex matching with a curated list of image-related words. *SpellChecker()* is used before the matching. This makes the approach quite robust. Once we have the intent, the questions go to the *get image()* to render the image. This function extracts the entity codes using the *entityextractor()*. Then, the image directory location is extracted to render the image

2.5 Recommender Questions

The intent is identified similarly to the multimedia questions. All the entities are extracted using *entityextractor()*. Then the embedding values of all these entities are summed up before calculating to the pairwise distances. Using the TransE embeddings, the top results are returned.

3 Results & Discussion

From the extensive tests, that were done, the model turns out to be really robust. The performance of the property extractor, in particular, stands out and the accuracy of the multimedia and recommender questions was also very nice. The extraction of the entity is not flawless even after using multiple approaches. The combination of all the approaches was used and evaluated and then this current approach in figure 1 was finalised. One thing that could have been explored was to train the custom NER model to boost performance.