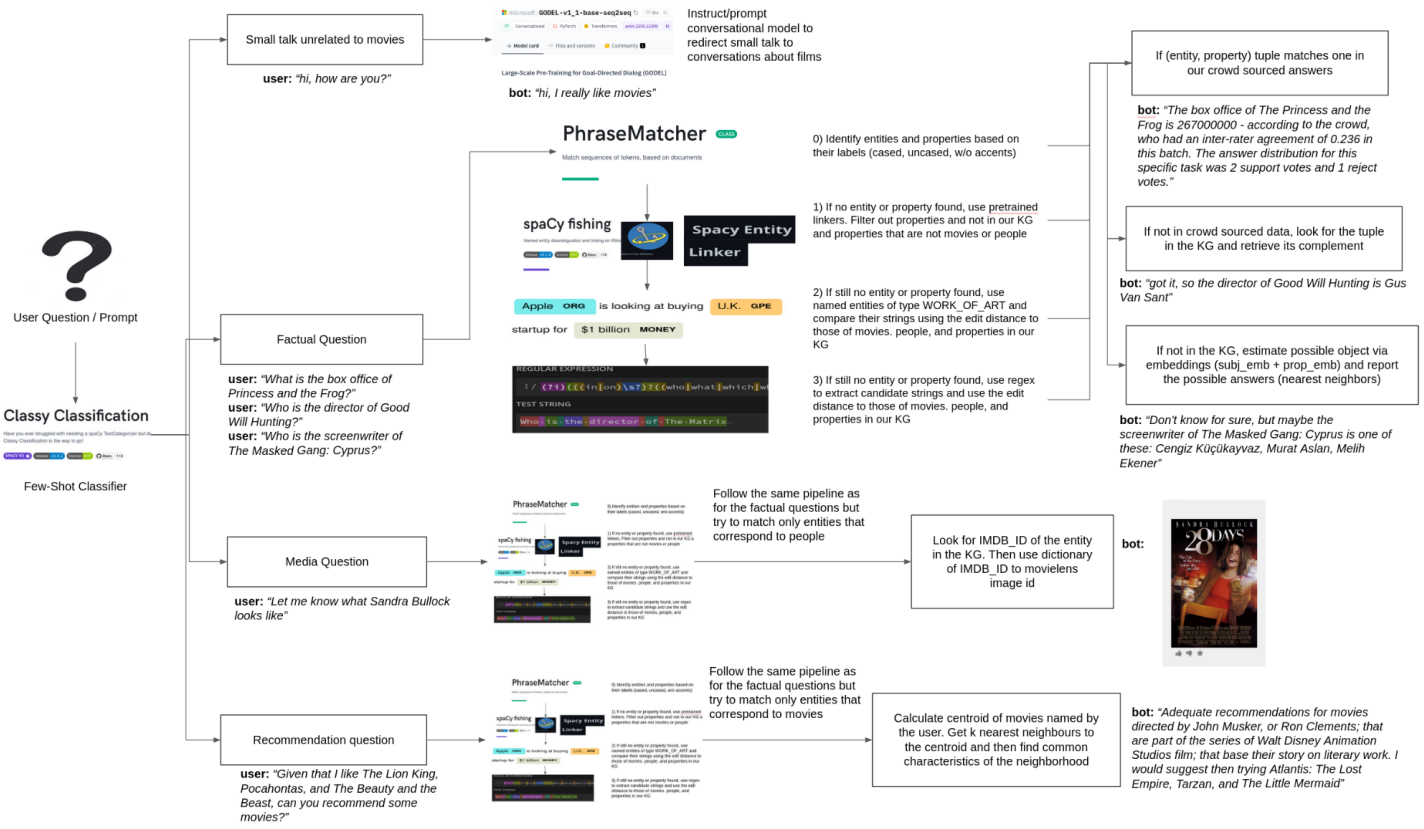
Class Project: Advanced Topics in Artificial Intelligence HS22

Juan Bermeo

Overview

Given a user's prompt, we classify the intent of the prompt to decide how it should be processed. We have four types of user intents: small talk, factual question, media question, and recommendation question. The prompt will be processed depending on the type of intent and the information available in the knowledge graph. The agent processes a single prompt at a time and previous prompts do not affect how a given prompt is handled. Next is a flow diagram that describes how our agent works:



Agent Components

Intent Classification: We use a spacy project that leverages sentence-transformers to create good text classifiers with only a few examples.

Small Talk: We use an instruction based LLM created by Microsoft called GODEL in which we can directly prompt what we want the LLM to do. We instruct it to redirect the topic of conversation to movies and also prompt the dialog with a previous interaction in which redirection to discussion about movies is already attempted.

Answer Format: For each interaction/intent type we have a set of template answer structures which we sample from. This is to provide variation and a less mechanical interaction.

Entity/Property Matching: Depending on the type of intent, we want to identify which entities (people and/or films) and/or properties from our knowledge graph (KG) are present in the user's prompt. To do so, we followed the steps next:

1. Generate different string representations for the entities and properties of interest in our knowledge graph. Relevant entities were those of instance type person or of one of the entity types that correspond to films. As for the string representations: for properties, we queried their aliases from wikidata; for entities, we added an uncased label, uncased without accents, and uncased without accents, points, or hyphens.
2. Use PhraseMatchers from spacy to match exactly the different string representations.
3. Use two pretrained entity linking models from spacy named entity Linker and entityfishing. Coalesce and filter the list of entities and properties found.
4. If we do not have one entity and/or one property from wikidata by this point, then we use spacy's NER model and check if the strings for PERSON and/or WORK_OF_ART entities found have a fuzzy match to any of the labels of interest.
5. If this fails, we have a set of regex patterns for each interaction type to extract relevant substrings from the user's prompt and again match them via fuzzy matching.

Crowd sourced data cleaning: We removed users that had a low overall score, always took an unlikely short time to answer, or answered all questions in the same way. We also used a correction offered if at least two workers agreed on its value. See crowd_sourcing.ipynb

Factual Questions: We try to find at least one property and at least one entity of type person or film. If at least one of each is not found, or more than one of either is found, we prompt the user to rephrase the question in simpler terms. If one of each is found, we verify if the tuple is present in our cleaned version of the crowd sourced data. If it is present, we report the answer from that dataset in the required format. If it is not present, we look for the tuple in our KG. If the tuple is in our KG, we report the object(s) found. If the tuple is not present in our KG, we use the given KG embeddings to find a plausible representation of the object (analogical reasoning) and report its nearest neighbors. If no embedding is found for the property or the referred entity, we tell the user we do not know the answer to the question.

Media Questions: We follow our entity/property matching approach but focus only on entities related to people. If a relevant entity is found, we query their imdb id from the KG and then map it to an image of the movielens. To map from imdb_id to movielens image ids, we looked for images in which only the given imdb_id was listed.

Recommendation Questions: We follow our entity/property matching approach but focus only on entities related to films. We calculate the centroid of the embeddings of the movies named by the user for which we found their wikidata entity names and corresponding embeddings. We then get the k nearest neighbors to the centroid and find common properties of the neighborhood. If a property is common to more than x% of k neighbors (the threshold is different for each property), we report this as a recommended characteristic as well as the names of the nearest neighboring movies.