Rule-/retrieval-based Q&A subsystem

# Purpose

As explained in Section 4 of the Design Report, the team decided to design the Q&A subsystem to first try to match incoming queries with a set of predefined rules and then fall back to a machine learning model if no good match exists. This hybrid approach covers a wide range of queries with the machine learning model while leaving room for fine-tuning with rules, forming a comprehensive solution.

# Design

Rule-based chatbots take users along different paths in a tree of questions and answers based on their responses, with retrieval-based chatbots automatically gathering some of the information presented. The team’s goal is to adapt these concepts to let the user ask natural questions and have chatbot find and ask for any missing information needed for a specific answer, rather than restrict the user to answering a fixed series of prompts.

The design made by Mitesh and Sandor is to build a tree where each node represents a category of information users may request, along with some associated keywords. The children of the tree’s root are the broadest categories, such as assignments, course guidelines, and PE/CE information, and the user’s initial query is sorted into one of these nodes by keyword similarity. (The exact similarity measure is not finalized.)

Within each broad category, the desired answer depends on several parameters. When asking about an assignment, the user may give the assignment name and ask for its due date, but they may also want the assignment’s template, rubric, or other information. Therefore, queries sorted into the assignments node are searched for the assignment name and the specific intent. If one is missing, the bot asks for it, looks in a list of valid values for the best match to the user’s reply, and serves the predefined answer associated with it. If both are missing, the bot asks for both before proceeding.

But there is a variety of information on the Capstone Support wiki. Sometimes, it’s better to assume a default value when information is missing from the query, such as returning a list of *all* assignments when the assignment name is absent. The system will allow the rules to be flexibly described for several types of nodes, so the bot makes reasonable decisions about what to show at every step.

# Implementation

What remains now is to decide which kinds of information on the wiki need different node types and implement each type with its different rules in Python while fleshing out the database of predefined answers. The team must also decide on a good measure of keyword similarity between queries and nodes. Alternative approaches to the rule tree are also possible, such as simply making it mirror the structure of the wiki. These should be evaluated before committing to the design described here.