Dustin Jones

Ashok Goel

CS-7637

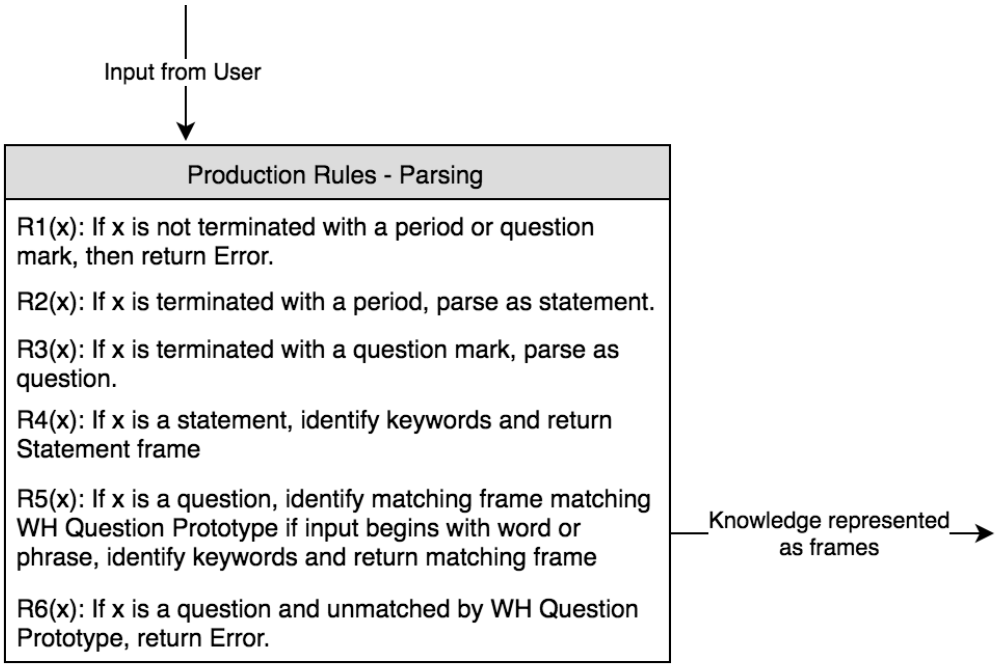
2016-10-09

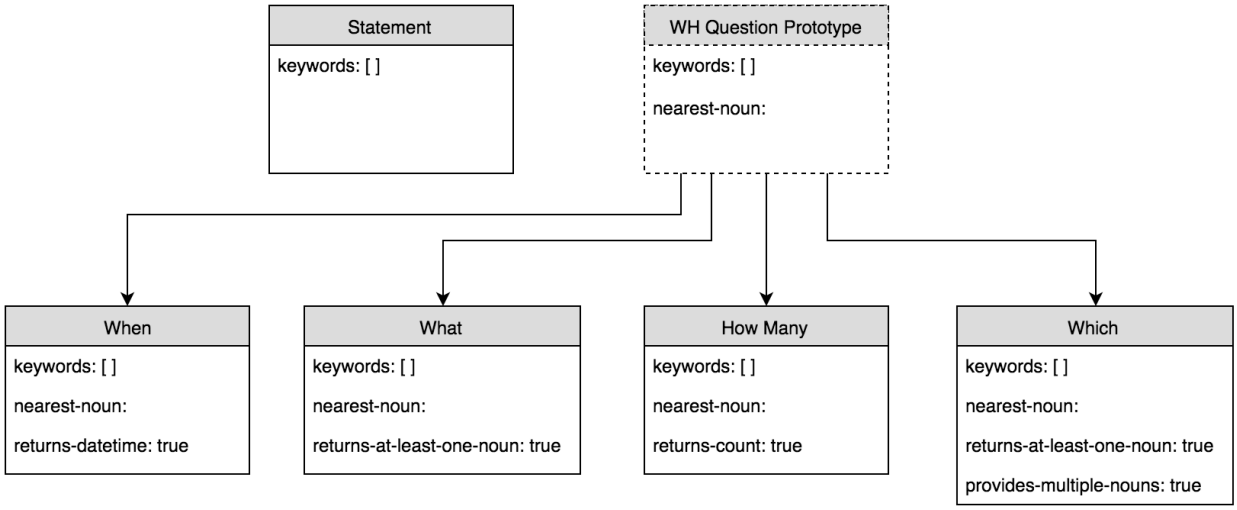
Mid-Term Examination

With the goal of designing an AI that is capable of natural language processing using methods of KBAI, my agent will be discussing the Georgia Tech OMSCS program. The design of the agent will be broken into 3 major steps: parsing input into a computable form, acting on the input – interpreting it and deciding upon an action, and building an output from the decided upon action.

Parsing input into a computable form

Input to this agent will be limited to defined keyword and phrase matching represented by frames and governed by production rules.





The input will initially be classified as either a statement or as a question. This will allow the agent to determine what actions need to be taken with the input. If the input is a statement, its resultant frame will be stored in short term memory, if it is a question, memory will be queried to provide a response to the user.

A dictionary of keywords and phrases will be stored in memory for use during the parsing process, and will be known as the keyword dictionary. Each entry in the dictionary will have a corresponding part of speech (noun, verb, adjective), a list of synonyms, and a list of antonyms. This dictionary will be prepopulated by words and phrases relevant to the discussion topic, sourced from the Oxford English Dictionary. During the parsing process, any entries from the keyword dictionary that are found to exist in the input will be stored in the keywords slot of the resulting frame. If no matching keywords are found, the agent returns Error.

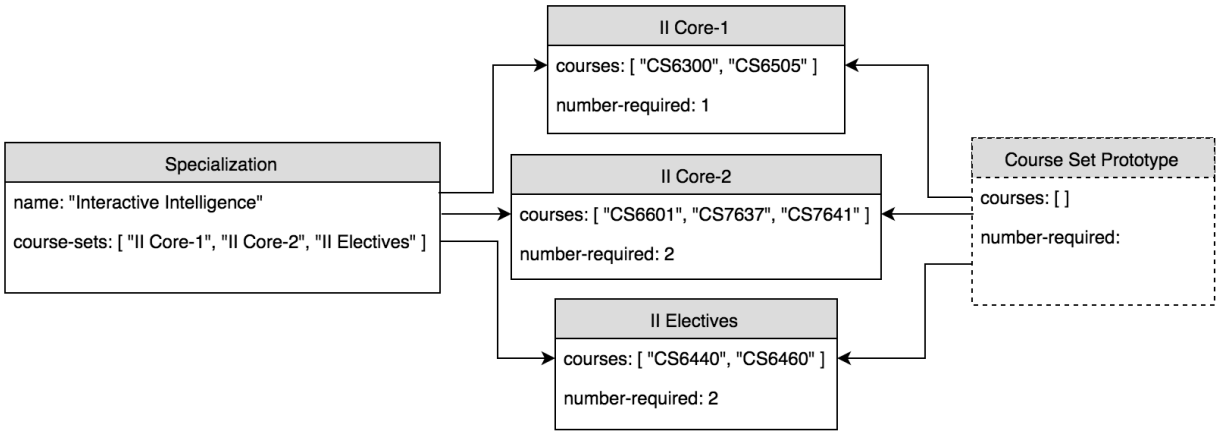
For questions, the input will be searched for the noun, or adjective-noun pair nearest to the WH Question Prototype type that exist in the keyword dictionary and stored in the nearest-noun slot of the resulting frame. For example, if the input was: “What is the hardest class offered this

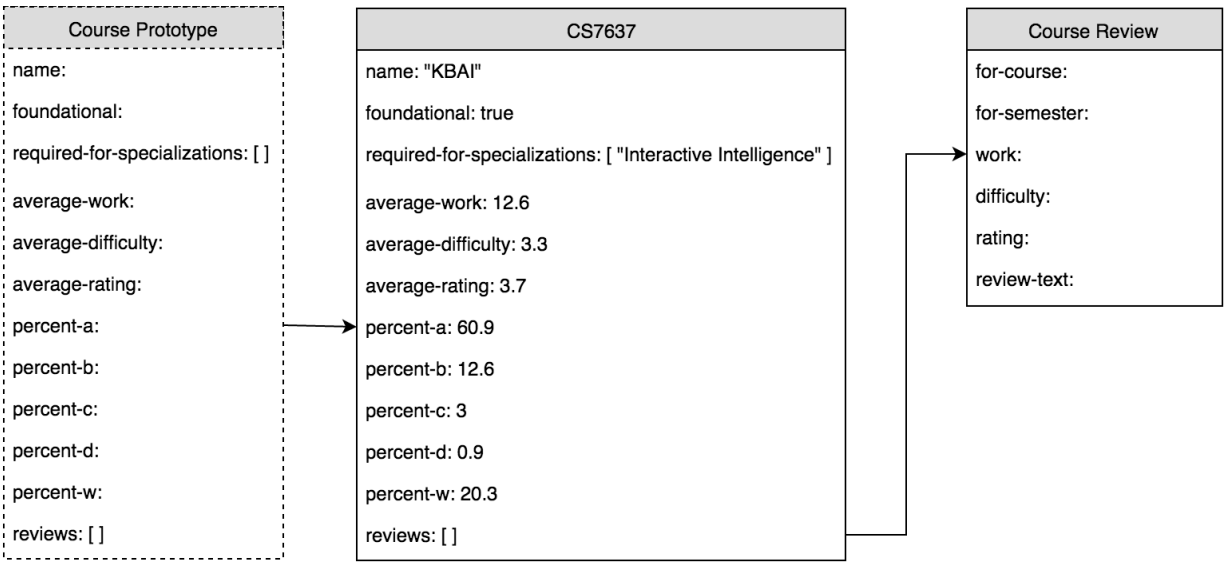
semester?", where the words "hardest", "course", and "semester" appeared in the dictionary (having identified "class" a synonym for "course"), the resultant frame would look as it does below.

What
keywords: [hardest, course, semester]
nearest-noun: "hardest course"
returns-at-least-one-noun: true

Acting on the input

Along with the predefined keyword dictionary, the agent will make use of three additional data sources. The first is a predefined set of data gathered from the GA Tech website about the OMSCS program cataloging data on specializations, associated required classes, and all available classes. The second is a dump of data from the OMS Central website (https://omscentral.com/), using both the pre-calculated statistical data for courses such as average difficulty, average rating, and grade distribution, along with textual course reviews. The third is a dump of data from the OMSCS Google+ page with general discussions of the OMSCS program, of courses and specializations.

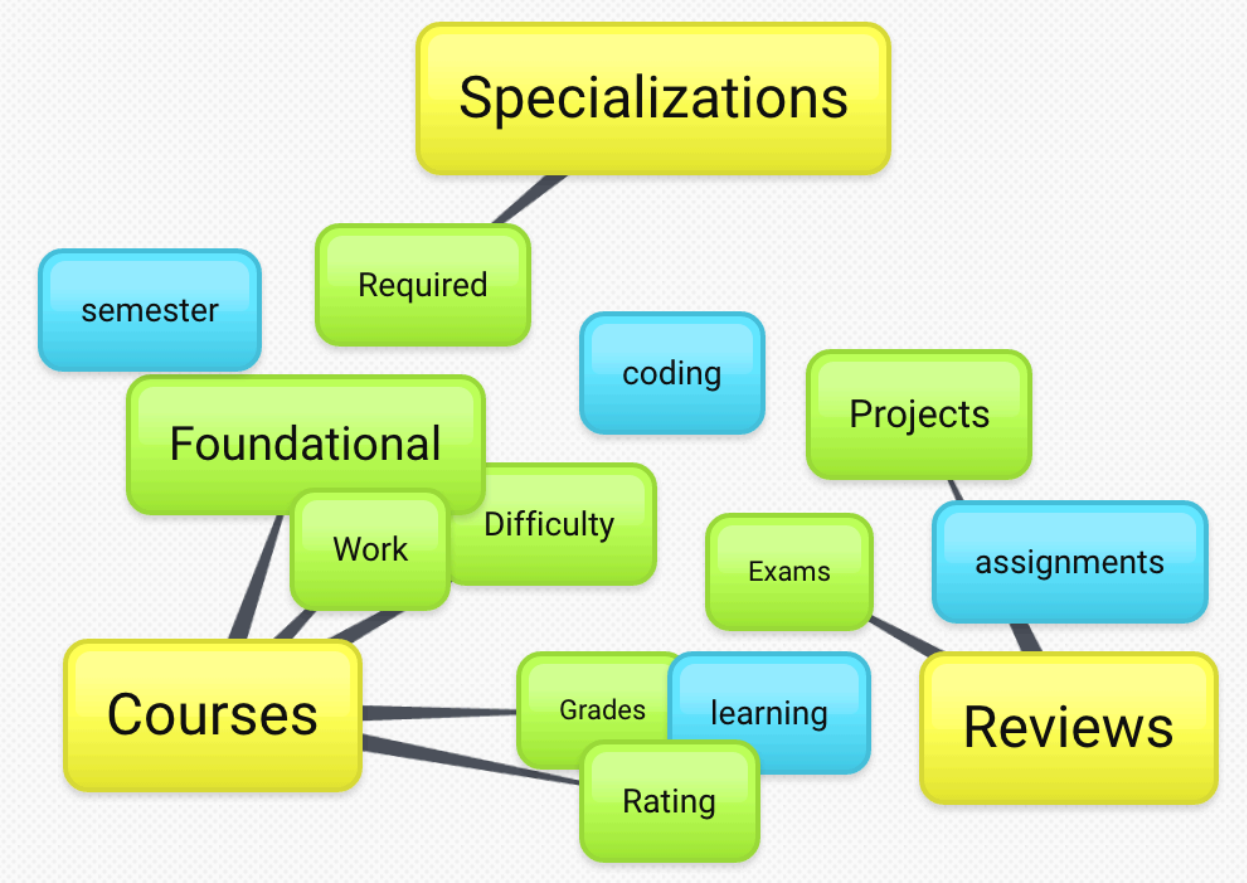




This data will provide the basis for constructing replies to inquiries by the user. Interpreting the intentions of the user will be done by comparing the resultant input frame using semantic networks with a library of cases that will evolve over time through use of incremental concept learning and case-based reasoning.

The initial decision making process will begin as a production system in order to give the agent a starting point for existing cases to examine. It will begin using the nearest-noun slot filler to identify if the user is asking about one of three categories: specializations, courses, or reviews. It will do this by searching and comparing each word in the nearest-noun slot against the category name and synonyms of that name in the keyword dictionary. If no match is found, the agent will return Error. Using the previous example input “What is the hardest class this semester?”, the agent will search the keyword dictionary entries, and match to “courses” because the word “class” exists as a synonym to that category.

Each category will have various keywords associated with it and be represented internally as a 3-dimensional graph. Each category will appear in the graph and will remain an equal distance from one another. Keywords will be placed around each category with relative distance to each determined by the frequency it is mentioned with regards to a category, and distance from other keywords determined by frequency to each other. The size of each keyword will be determined by the overall frequency at which it is used within the context of the agent. The data dumps will be used to prepopulate the graph with keywords, searching against the keyword dictionary and associated synonyms. For example, if visualized the graph might look like the below image after being populated with data. Yellow bubbles representing categories, green bubbles representing keywords for which the agent has data, and blue bubbles for keywords without data associated.



Using a weighted k-nearest-neighbor algorithm, the agent will use the keyword or keywords in the nearest-noun slot (weighted heaviest), along with the additional keywords in the keywords slot (each weighted according to the size of the keyword in the keyword-graph) to determine the nearest keyword for which it has data. Using the previous sample image, and the sample question "What is the hardest class this semester?", the agent would compute that the user is requesting data on the difficulty of a course or courses.

While being able to respond to a user with actual information is useful, it's not always possible to come up with a perfect answer. Much as humans also like to pretend they also know and understand the answers to all questions, more frequently than not there are other people that know a particular topic with more expertise. In fact, some of the most intelligent humans are those who can admit to not knowing something and instead refer you to someone who does know something. Since this agent also does not have perfect knowledge, it must be able to recognize and accommodate when it is unable to produce a satisfactory answer. As part of its behavior, this agent will record cases for which it does not have direct data (an example of direct data being difficulty ratings, or if a course is foundational) that it can reason about.

When a user input frame is processed and compared against the keyword graph, it will use a limiting factor to prevent it from trying to correlate keywords that are too diverse, that is if a set of keywords and categories are too far from one another, it will instead use other means to identify relevant data for the user. The agent will instead compare the input keywords to those keywords used significantly within the OMSCS Google+ groups post data. Again if a post is

found to contain the relevant keywords, a link (or links) will be provided to the user to allow them to interpret the information on their own.

Building an output

In order to determine if the agent is answering the question posed by the user with a relevant response, the agent will always respond to a question with a question asking if the keyword category-keyword pair identified by the agent is what they are interested in, seeking a simple “Yes” or “No” response. If the response is affirmative, the keywords from the users input frame are moved closer to one another, and to the identified keyword in the keyword graph. If the response is negative, the identified keyword is temporarily disabled, the nearest neighbor algorithm recalculated, a new response is generated and returned, again asking for confirmation. If this second response is affirmative, the users input keywords will be moved closer to the newly identified keyword. If the second response is still negative, the agent will ask the user to rephrase their question.

By adjusting the positions of keywords within the keyword graph only in successful circumstances, the agent will be able to train for better results on subsequent attempts of questions asked that are similar. By not changing the keyword graph when a category-keyword pair is chosen incorrectly, the chance of negatively impacting the overall characteristics of the graph from questions with poor grammar are greatly reduced. These changes, or lack thereof have the effect of adapting the agent’s concept of the keywords associated with the various categories, effectively specializing the model each time a question is successfully answered.

Finally, after having successfully chosen a category-keyword pair, final analysis upon the data can commence. Basic lexical analysis upon the adjective-noun pair can be used to determine that if an adjective is present whether or not it is a 'sortable' adjective, meaning that there is some sort of order applied to the result or results that should be output. For example, if a word ends in "-est" it may indicate that it is a sortable quality. If the adjective identifies that the keyword should be sorted, the frame associated with the keyword can use the sort-and-select function with a predefined set of sorting keywords to process the data and select a result. Following our previous example of "What is the hardest class this semester?", the adjective would be identified as "hardest", and would be passed to the courses frame upon which the sort-and-select function would determine that "hardest" is a member of the qualifying sortable keywords and that it should sort courses in descending order by the average-difficulty slot, and select the first result. If no sortable keyword was found, as in the question "How hard is the course KBAI?", the value or values associated with that particular keyword are selected and returned.

Using a predefined list of responses, the agent will select the appropriate response based upon the characteristics of the original input frame, and the category-keyword pair selected previously using a production system. These responses will exist as 'fill-in-the-blank' style sentences, where each 'blank' has associated attributes to guide the allowable input. For example, for our sample question "What is the hardest course this semester?", the selected response might resemble "The __(adjective)__ course is __(course-name)__ with a rating of __(number)__". By limiting the agent to a predefined list of responses, the grammar of the responses can be much more directly controlled over other methods that try to compile a sentence from scratch.

This feedback cycle is crucial to the operation of the agent. As in a normal conversation, humans are able to discern behavioral cues and percepts to determine whether or not their message is being understood, or whether or not it's relevant to the conversation. Since those cues are typically unavailable via textual format, the simplest way to identify if a message is being understood is to ask.

As the feedback cycle continues between agent and user, the agent will record those perfectly successful conversations so that they can potentially be adapted more directly for later use. Continuing with our example question: "What is the hardest class this semester?", if the agent is successful in returning a result to the user, the agent will store the input keywords and the output response, whether it be returning a calculated answer based on data it has, or returning a link to information that is relevant to that user. The returned data will be tagged with the keywords in question so that it can be looked up more efficiently in the future.

As the agent gets more use it will be able to build its library of recorded cases. Maintaining and updating this library will however be challenging. One of the largest challenges might be how to deal with users of different backgrounds (such as geographical, socio-economic, experience) that require different levels of detail about information. Some users may only want to know high-level information about a particular topic, while other users may want the most absolutely detailed information. Being able to recognize these scenarios by keyword usage simply may not be sufficient to cover all of those use cases.