Zachary Waters

Ashok Goel

CS 4635 Knowledge-Based AI

October 28, 2019

Project 2 Reflection

Human Thought Process:

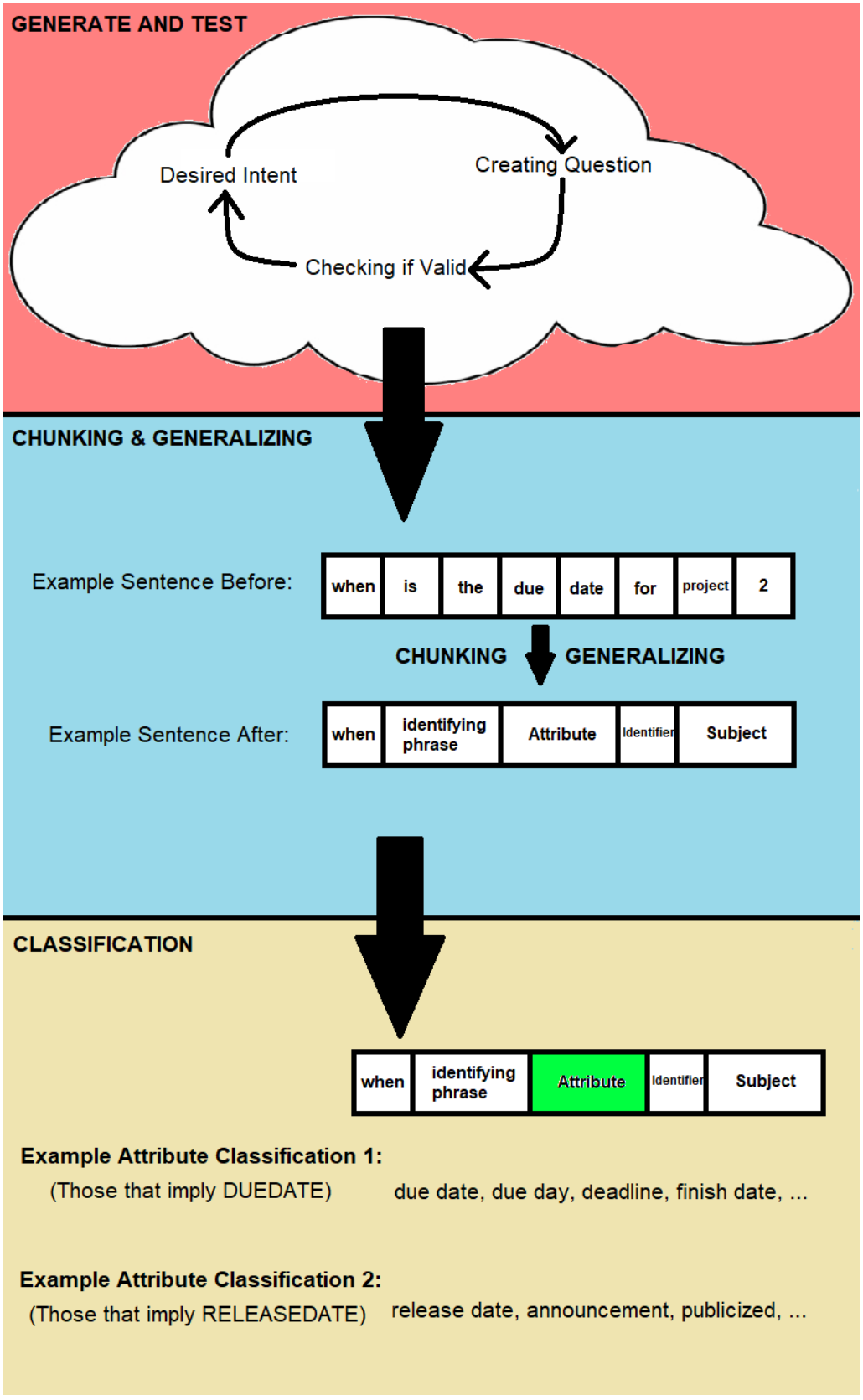
When I was classifying my example sentences, I ended up grouping them into two main categories. The first category contained questions that fit the more traditionally defined format of the previous project. This means that their sentences had a clearly defined subject and category, which made the process of determining its intent rather straightforward. An example of such a category would be "when is project 1 due". The second category contains questions that rely on tertiary information in order to identify their subject. As seen through this example: "when is the project released on week 3 due". To identify the intent of the second category, I am required to know the background information that defines that various objects of the domain space. In order to determine the intent of the previous example, I must know the release dates of the projects; to figure out which one was announced on the third week.

I found certain questions to be more challenging to answer over others. These questions usually contained multiple qualifiers that made the question more challenging to decipher and classify. Such as "what project has weight 15% and released week 6". To answer such questions, I often had to start breaking down the sentence into smaller chunks to then individually validate and then combine for a total answer.

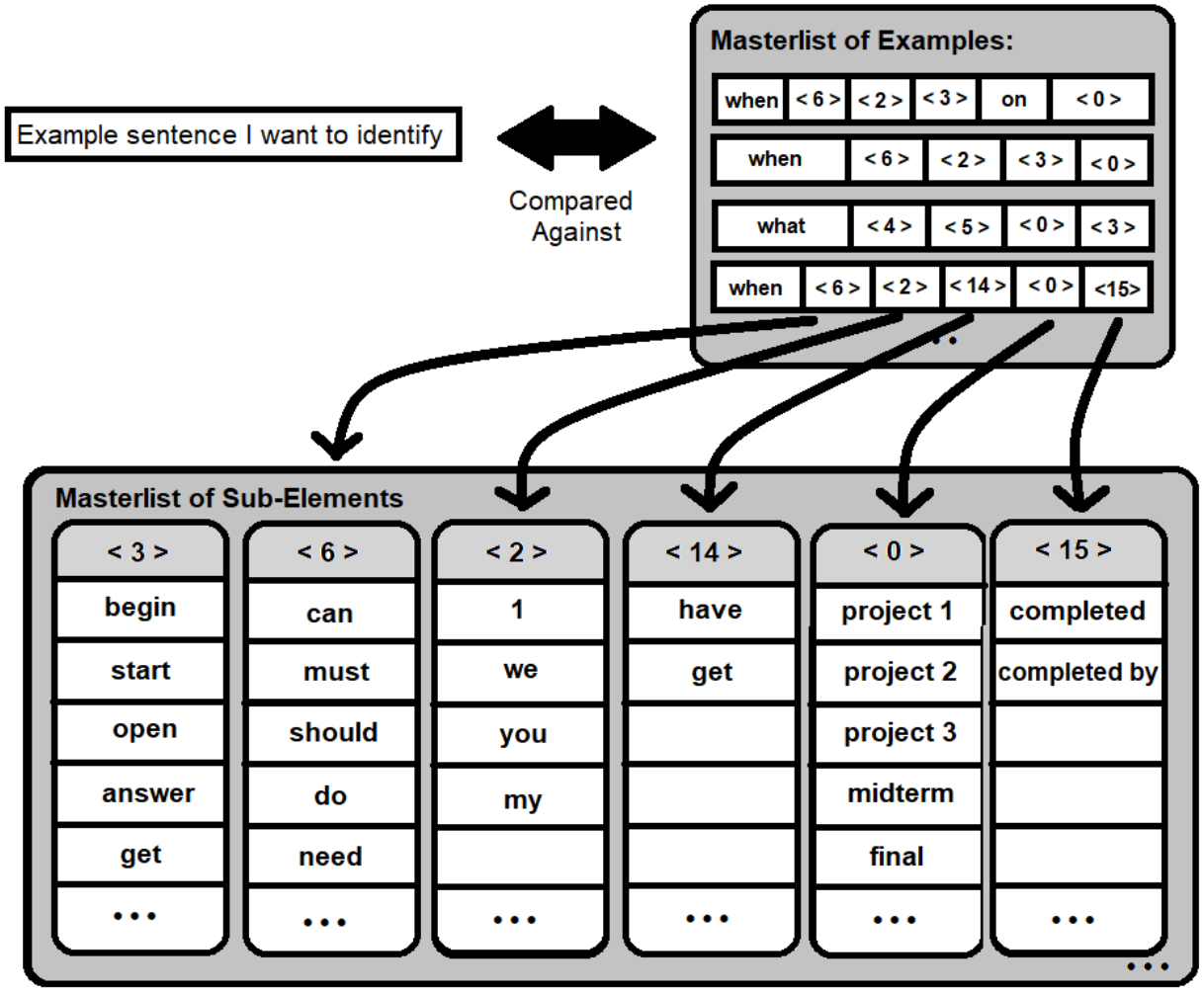
KBAI techniques:

When creating and classifying my own examples, the KBAI techniques I found myself using the most: were a combination of generate and test, chunking/generalizing, and classification. I used the process of generate and test when creating the questions themselves, as I would often start with the desired intent of a question and then work backwards towards developing the question's structure and verbiage. After creating a new question, I would then use chunking and generalizing to break down and conjoin various elements of the sentence into blocks that represent similarly attributed information. These blocks betoken the many variations of synonymic words and phrases that could fill their place. For example, you can often replace the subject of a sentence with another, and the question would remain entirely valid. As seen in the diagram, I represent the subject of a sentence with a more generalized block that is just referred to as SUBJECT. This block represents all possible subjects for the question, like project, assignment, and midterm. The sentence is comprised of various chunks that would each use different lists of various words that would be valid in their slots. I then use classification to sort these lists into categories by the types of intents they imply. As seen in the following diagram, I made a distinction by separating out the attributes that would imply the sentence is asking DUEDATE from a separate list of terms that implies RELEASEDATE.

KBAI Diagram:



Agent Diagram:



Comparisons and Contrasts:

I went through a great amount of effort to make sure that my agent's processes closely reflect my own mental procedure for when I was classifying and solving questions. My agent works by comparing a provided question against a master list of examples. Each example can be comprised of a multitude of generalized blocks, where each of these generalized blocks refers to a separate array of potential words held inside a separate master list. One of the major differences between my own mental KBAI process and my agent's is the amount of abstraction capable from a single generalized token. For example, in my KBAI process I can have my generalized "attribute" token simultaneously point to multiple different classification types. As shown in the KBAI diagram, my attribute block points to both a DUEDATE list and RELEASEDATE list. By contrast, my own programmatic agent is limited to pairing each generalized token with only a single sub-element list. This difference sadly comes about due to the limitation of my own programming, and if given the opportunity to recreate my agent, I would remove this flaw.

Correct Test Questions:

1. **When can I start on assignment 1**
2. **What project can we work on during week 6**
3. **What must I finish by week 3**
4. **What project will be available in week 15**
5. **How much time is there for submitting assignment 1**

Incorrect Test Questions:

1. **What project has weight 15% and released week 6**
2. **What are not the file submissions for project 2**
3. **What is the due date of the project on week 40**
4. **When can I write project 2**
5. **What are the due date announcements for project 2**

Incorrect Test Questions Explanation:

My agent incorrectly parses the first question on the list because it cannot perform the search required to answer the conjunction of multiple attribute types. My agent fails the second question because I did not have the time or resources to properly account for the many ways "not" could be used in a question. The third example fails because I ran into storage problems when trying to account for all the variations of invalid objects. My agent fails to properly answer the fourth problem because it lacks the specificity to know that projects are coded, not written. My agent fails the last one because it overgeneralizes and assumes that any question that contains announcements must be about general class announcements, giving it intent 41.

Metacognition:

This assignment showcased to me a valuable approach to solving problems. That of dividing more complicated questions into smaller more manageably solvable sub-questions. This was most clearly demonstrated to me when I was trying to solve questions that referred to their subject through the use tertiary attributes, like "when is the project released on week 3 due?". Because I don't have a strong recollection of the various project and their attributes, I had to treat this question like the combination of two smaller sub questions:

"what project was released on week 3?"
and
"when is that project due?"

And when solving these two subproblems, if the first one ever failed it would result in the failure of the entire combination. This became incredibly relevant when implementing my agent, as I quickly learned that the most efficient way to determine if a question was invalid was to check if its object was invalid. This meant similarly dividing the subject of the question into a smaller query that can be used to immediately determine if the entire question itself is valid.