Zachary Waters

Ashok Goel

CS 4635 Knowledge-Based AI

September 23, 2019

# Project 1 Reflection

**Block Diagram:**

See Page 2.

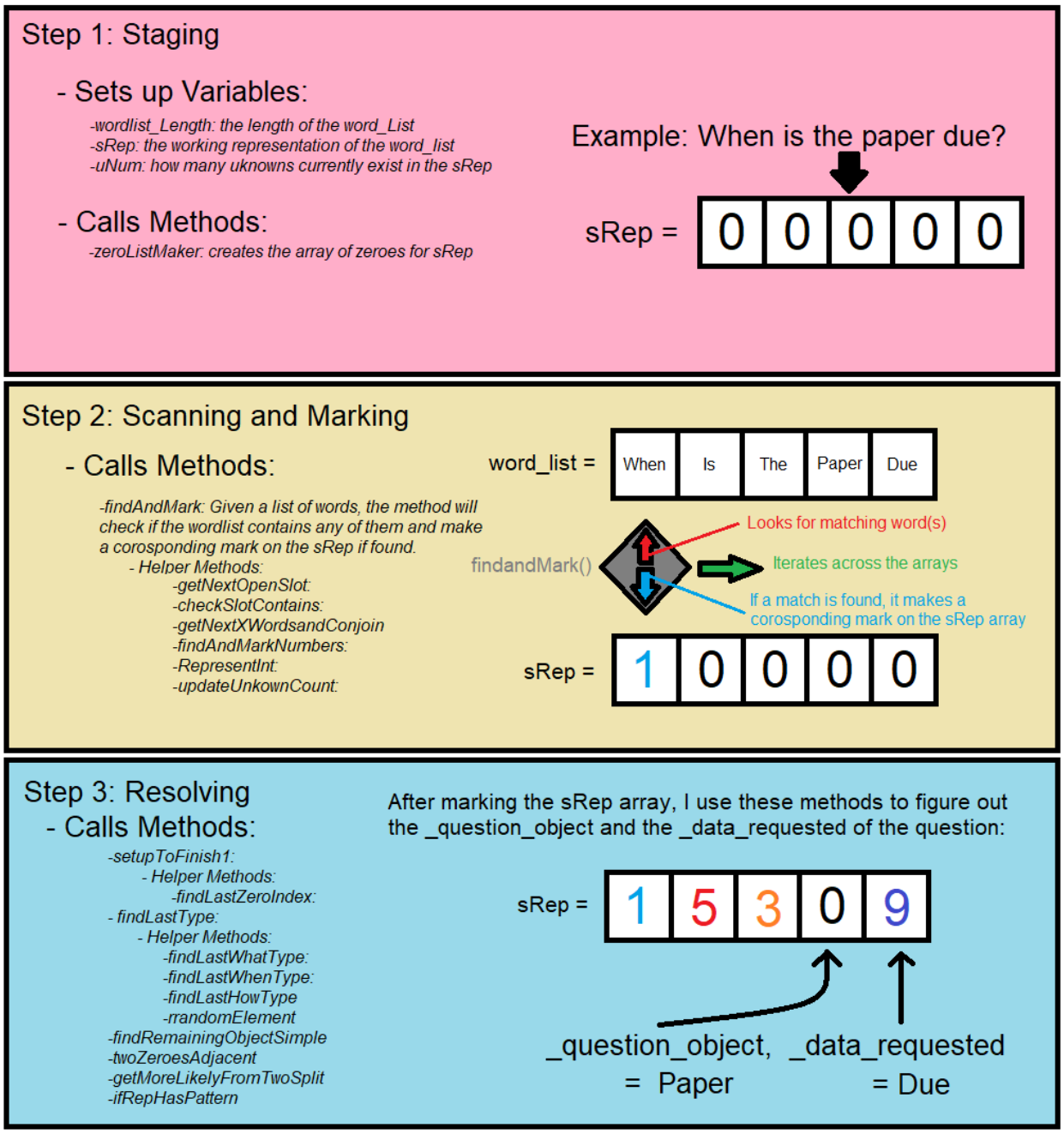
**Block Diagram Breakdown:**

As shown in my block diagram, my agent's functionality can be broken down into a 3-step process. Step 1 is the staging process, where I set up the initial variables to be used for the later steps. The most important thing established by this step is the creation of sRep. sRep is an array of matching size to the provided word_list. Being created by the zeroListMaker method, it naturally starts entirely fully of zeroes. However later in Step 2 the sRep zeroes are replaced with numeric values that represent the "type" of word that sits in the matching word_list index. Let me move onto Step 2 to elaborate.

In Step 2, I use the method findAndMark to find each word's "type" in the word_list and then write that type's signifying number in the matching sRep location. For example articles, such as "the", "a", and "an", are a type that I look for. So, I provide the findAndMark method with a list of the words I'm looking for, and then assign a number that I want it to be marked as. In my example above those articles are marked with the number 3. I repeat this process multiple times for a multitude of types such as: ordinals, numbers, personal pronouns, possessive determiners, modal verbs, etc. Each one of these types is given a unique identifying number for future identification, in Step 3.

In Step 3, I use the pattern of numbers created in sRep to determine what the question's object and the data requested. Because I am unable to directly find and mark the nouns in the sentence, due to the inefficiency of looking through the entire list of nouns at every index, I use a more heuristic process. By marking everything that isn't a noun with a number, I know that the subject must be one of the remaining zeroes left in sRep. I can find the subject by the process of elimination (method setupToFinish1), or if there is an ordinal or number adjacent to it, or more complicated patterns handled by ifRepHasPattern. I use the findLastType method to calculate the _data_requested of the question. It works by looking at the initial question word (when, where, how, and what) and calling the corresponding helper method that further examines the patterns within sRep to determine the data type being requested.

**Block Diagram Representation**:



**Usage of Frames:**

Due to the nature of this first project, and how I went about solving it, I believe that my agent's use of frames is not immediately recognizable. Despite this, however, my agent can be thought of constructing a frame with two slots for each of the questions. These slots are the _data_requested and _question_object variables. These slots are what will allow a more complete version of the agent to determine what information to gather in order to answer the user's question.

**Relationship to Human Cognition:**

I believe that my program acts relatively similarly to human cognition. My program looks for common patterns in language to determine the subject of the question. For example, in the phrase "what is the weight for the test", the subject is "test" and the pattern that illustrates that is "for the". To elaborate on this one example, let's say we replaced "test" with a word that wouldn't make any sense in its place, like "were". We now have "what is the weight for the were", both a human reader and my agent would struggle at comprehending this question, as they would expect a noun to be in "were" position.

**Testing my Agent:**

I created approximately 80 test questions for my program. I did this by starting with a single question for each combination of question word (when, where, how, and what) and possible data requested type. I wrote this list in Excel, with each word taking up a column, so I could easily track the length of my questions. From each question I would then create a more generalized form. For example, I started by replacing the subject of the sentence with a more general <subject> identifier tag, which would stand in for any noun that could be in its place. I continued to repeat this process, making each question more and more general until I list of word types in the pattern of the original question, I would then repeat this process with a new question. By the end I had constructed enough patterns that I could generate many unique test questions for my agent, well as create a system to generally reverse engineer the subject from most questions. Images are provided below showing an example of my question patterns that I had constructed.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | where | #BEING-SYNONYMS | #NOUN-P1 | | | |
| 2 | where | #BEING-SYNONYMS | #ARTICLE | #NOUN-P1 | | |
| 3 | where | #BEING-SYNONYMS | #POSSSESSIVE | #NOUN-P1 | | |
| 4 | where | #DOFORMS | #PERSONALPRONOUN | #SUBMIT-SYNONYM 1 | #ARTICLE | #NOUN-P1 |
| 5 | where | #BEING-SYNONYMS | #ARTICLE | #NOUN-P1 | #NOUN-P2 | |
| 6 | where | #BEING-SYNONYMS | #ARTICLE | #NOUN-P1 | | |
| 7 | where | #BEING-SYNONYMS | #ARTICLE | #NOUN-P1 | #NOUN-P2 | |
| 8 | where | #MODALVERB | #PERSONALPRONOUN | #SUBMIT-SYNONYM 1 | #ARTICLE | #NOUN-P1 |
| 9 | where | #NEGMODALVERB | #PERSONALPRONOUN | #SUBMIT-SYNONYM 1 | #ARTICLE | #NOUN-P1 |

| K | L | M | N |
|---|---|---|---|
| #NEGMODALVERB 1 | #NEGMODALVERB 2 | #DOFORMS | #BEING-SYNONYMS |
| won't | will not | do | is |
| can't | can not | did | was |
| don't | do not | does | will |
| cannot | | | will be |
| couldn't | could not | | were |
| hasn't | has not | | are |
| isn't | is not | | had been |
| shouldn't | should not | | been |
| wasn't | was not | | has been |
| hadn't | had not | | |
| didn't | did not | | |
| doesn't | does not | | |
| aren't | are not | | |
| | | | |
| #POSSSESSIVE | #PERSONALPRONOUN | #ARTICLE | |
| my | I | the | |
| our | we | a | |
| | | an | |
| | | | |
| #SUBMIT-SYNONYM 1 | #SUBMIT-SYNONYM 2 | | |
| submit | turn in | | |
| deliver | send in | | |
| | hand in | | |

**My Agents Strengths and Weaknesses:**

my agent is best at identifying the subject of a sentence using its pattern matching system. It also contains a rather robust collection of words and their various possible synonymic phrases such that it that can easily adjust to many possible variations of the same structure of sentence. My agent is at its weakest in determining the data type being requested. Its pattern structure is not as helpful when many of the same patterns can be used to ask for widely different types of things. Another flaw that my system has is that it struggles to handle the many multitude of unique verbs that can be thrown at it. For example, the question "What time can <object> be <Verb>" can be extremely challenging for my agent to identify its data request type entirely hinges upon what verb is specifically being used in that situation. For example, if the verb "download" was used, the type would be releasedate, while "submitted" would be due date. Trying to account for all these cases became an impossible task that I had to forgo in the sake of finishing my project on time.