Justifying the Use of an Adjective of a Notion

K.H. Cheng
Computer Science Department, University of Houston, Houston, Texas, USA

Abstract – To communicate sensibly with humans, it is important to provide a sensible answer with a justification. In this paper, we tackle the problem of finding the correct answer to a decision question that involves an adjective of a notion. We provide a simple solution to find the answer to this classification problem and provide the reasons to justify the answer. Since these answers are subjective, the answer is more convincing if a reason can be provided. In addition, providing a justifiable answer allows the program to make its own decision and becomes more autonomous. The solution depends on the program’s ability to acquire all the necessary knowledge, to locate the appropriate knowledge when needed, and to infer relationships among categories.

Keywords: adjective, notion, reasoning, knowledge representation, natural language processing.

1 Introduction

It is desirable for a computer program to possess the capability to communicate with humans using a natural language, specifically English. A decision question is a kind of questions asking for the confirmation of the given. It has the sequential grammatical structure: main verb followed by complete subject and then by subject complement, omitting the question mark punctuation to simplify our presentation. The complete subject defines the target of the question, and the subject complement defines the given to be verified. In this paper, we tackle the problem of finding the correct answer to a decision question that involves an adjective for a qualitative aspect, referred to as a notion. This corresponds to the classification problem of a category defined by an adjective and a category. The answer is true if the target can satisfy the criterion of that classification. For example, a human is important if he/she has some important achievement. Note that since notions are qualitative and the answer depends on learned criteria, the answer is subjective. In other words, if the same program is loaded into different robots, one robot may disagree with another robot even based on the same set of facts because their learned criteria may be different. As a result, some researchers [1] suggest to combine opinions from multiple sources to answer such questions. In that case, the answer is not based on the knowledge known by the robot, but by using other persons’ opinions. This approach may be useful if the robot is ignorant on the subject matter. However, Statistical methods were used in [1] to classify the sense of the opinions instead of truly understanding what each opinion actually means. This problem is difficult not because it needs complicated Mathematics, but instead it involves a lot of knowledge. In addition, each piece of knowledge can be learned in multiple ways, and so can be represented and stored differently in the knowledge base. Furthermore, there are many different criteria, and each criterion may need a different piece of knowledge. Since the criteria can also be taught in multiple ways, the problem becomes how to link what is needed with what is learned, so that finding what is needed can be accomplished easily.

Providing a solution to this problem is important because an autonomous robot may have to decide on many issues involving notions. For instances, the robot may need to know why a task needs to be done before it decides to perform the task. It may also prioritize the tasks that it has to work based on their importance or urgency. It may also have to determine that the source of the knowledge is trustworthy, so that facts acquired from that source can be trusted. In each case, the robot can justify its own decision and subsequent action by using our provided solution. Note that the criterion to justify the use of an adjective may contain many sub-criteria, and it is sufficient to simply use logical operators “and”, “or”, and “not” to combine them to define any criterion. In this paper, we will only work on situations such that the criterion is satisfied as long as one sub-criterion is satisfied, i.e., the criterion is the logical or of the sub-criteria. In other words, each sub-criterion is a sufficient criterion and satisfying a sufficient criterion will provide the reason to justify the answer, true. A sub-criterion may also be qualitative and it itself is also a classification, so the problem of finding a satisfied criterion is recursive. For example, an important achievement is one of the criteria for a human to be important, and receiving a prestigious award or making a scientific discovery may each be considered as accomplishing an important achievement. As in all incremental learning situations, it is possible to have knowledge that still needs to be learned. The answer is true only if at least one criterion is satisfied, and false if all criteria are not satisfied and no unknown knowledge. In case no criteria is satisfied but there is any missing knowledge including the way to decide a criterion, the answer will be unknown. In other words, combining logical results and their associated reasons uses the 3-valued logic on the logical or operator [2,3]. We provide a simple solution to find the answer to this classification problem. Our solution solves the problem by using common sense knowledge and the inference capability on hierarchical relationships among categories. Since these answers are subjective, our solution also provide the reasons to justify
them, making the answers more convincing. In addition, when the answer is an unexpected false or unknown instead of true, the provided reasons may serve as a teaching tool to help human teacher or student to easily identify missing knowledge, which needs to be learned.

There are systems that focus on answering questions such as IBM’s Watson [4,5] and PENG Light [6]. IBM’s Watson has the ability to rank several plausible answers using a machine-learning framework, but it only expects Jeopardy-style questions formed as declarative statements and responds in the form of an interrogative sentence. PENG Light uses a First-Order Predicate Calculus [7] semantic representation with annotations for syntactic information. However, it needs an interface to a compatible logical reasoning system, and any modifications to the parsing grammar needs changes to the rules for sentence generation which threatens the coherency between reading and writing. Recently, we have developed a natural language communication system [8] for a learning program system, ALPS [9]. It handles any sentences typically used in a conversation based on the learned grammar. Since it automatically generates the role sequences [10] for writing when the parsing grammar is learned, so coherency is kept. By learning a small subset of the English language, ALPS is able to understand declarative sentences using action verbs [11], forms-of-be verbs [12], pronouns [13], and time adverbs [14]. It can also find the correct answer to decision questions when the subject complement is a noun [15], and an adjective [16]. Although [16] can also handle adjectives for a notion, it can only done so when the target has a known description for that notion. In other words, if the notion value of the target is unknown, it cannot find the answer even other supporting knowledge is available. For example, it cannot deduce that John is happy because he just graduated from college. In addition, it did not provide a reason to justify the answer is correct.

A major problem that needs to be solved is the knowledge organization problem since many different kinds of knowledge is involved such as notion, adjective, criterion, individual, and fact about individuals. The problem is to decide where and how to store the relevant knowledge so that when needed, each can be retrieved successfully and efficiently. Our solution stores the relevant knowledge in their natural places within the knowledge base. For example, the criterion is stored within a notion so that it can be found naturally. Our algorithm is very simple, and it does not use complicated Mathematics that is needed by Statistical methods. A related problem is to decide which knowledge to look for since there are a lot of sub-knowledge within each knowledge object. By following the presented solution, one can recognize that this is easily solved since we find based on what we need. Another problem is the multiple representation problem. In our learning system, knowledge is stored in a way based on how the knowledge is acquired. Since knowledge may be acquired in many different ways, the same fact may be stored in several different ways. The problem is how to provide the same answer even if the fact may be stored in a variety of ways. In order to give the same answer, we need to solve two sub-problems. The first sub-problem is how to get a fact no matter how and where it is stored. The second sub-problem is how to transform different representations into a single representation so that it can be used uniformly. Other relevant problems for answering a decision question such as the word-sense disambiguation problem and the missing reference problem have already been solved adequately in [16], and so will not be discussed here.

The rest of the paper is organized as follows. Section 2 discusses how different kinds of knowledge are organized within the knowledge base, how and why they are indexed in a certain way to allow efficient and natural retrieval of the required knowledge. Section 3 presents the algorithm to find the correct answer to the decision question, i.e., to find a satisfied criterion, if existed. Section 4 demonstrates how the algorithm works to find the answer to the question “Is target an important human”. Several examples are used to show how the algorithm arrives at the answer by using the appropriate piece of knowledge. Section 5 concludes the paper, and briefly presents some issues that still need to be resolved.

2 Knowledge Organization

The discussion on universally applicable notions such as importance, and the use of a linear scale to maintain the senses and the relative magnitude among the adjectives of a notion can be found in [16]. Here, we will only discuss the additional knowledge that are required to handle the reasoning problem. First, although the same adjective may be used to describe objects of different categories, the criteria that needs to be satisfied are different. It is obvious that the criteria to classify a human, achievement, event, job position, and legislation as important are different. In addition, criteria for different adjectives of a notion when used on a category, such as ugly and beautiful woman, may also be different. As a result, each classification is defined by an adjective and a category, and there may be many criteria within a notion, each for a specific classification. In order to access the correct criterion within a notion, the search key is simply the classification’s name which combines the names of the adjective and the category such as important achievement and prestigious award.

Second, the multiple sub-criteria of a classification are stored in a data structure containing two lists. The first list is an ordered linear list since some sub-criteria are more important than others. Our solution will follow this order so more important sub-criteria will be investigated first. The second list uses the C++ map by using the name of each criterion as the search key to avoid duplication. In addition, in order to avoid learning various knowledge in a strict sequence, a sub-criterion can be unknown when it is learned as a criterion for another classification, and the details on how to satisfy it can be learned at a later time. To do this, a temporary knowledge is created and stored, and it contains adjective and category as its sub-knowledge, both are also
temporary. The name of each temporary knowledge is the same as the eventual actual knowledge. To locate the actual criterion from the temporary one, we first use the adjective name to locate the actual adjective. From the adjective, the notion is uniquely located since adjectives for notions only describe one single notion, unlike adjectives of quantitative aspects. Finally, the actual criterion can be located within the notion using the name of the temporary one. If not found, then the criterion is not taught yet, and since the program does not know how to satisfy the sub-criterion, the answer for using this sub-criterion is unknown.

Third, since human may use remembered facts to answer questions, our program may also be taught a list of examples in addition to a list of criteria. In order for an example to be accepted by the program, it must be justified by one criterion, which will be used to justify a true answer if the example matches the target of the question. For example, Albert Einstein may be taught as an important human since he has an important achievement. The list of examples is stored in a list implemented by using the C++ map. By using the name as the search key, an example can be efficiently located and duplication can be avoided.

Finally, facts for individuals are stored within each individual based on how they are acquired. One capability of ALPS is to acquire a sub-knowledge of any knowledge by name, which serves as the search key for the acquired sub-knowledge within the knowledge. For example, the fact that Newton discovers gravity may be stored within Newton in several ways. If it is acquired by learning what he discovered, it is stored as discover: gravity, i.e., stored under the key discover with knowledge gravity. Alternatively, if it is acquired by learning his achievement, it is stored as achievement: discover gravity. Other facts and achievements such as inventions and innovations may all be stored similarly.

3 Algorithm to find the correct answer

Our solution extends the algorithm in [16] to answer a decision question that involves an adjective. Our solution starts when the target is not known to have a description for the notion, so the answer is determined based on the learned criterion of the classification. The answer is true if any sub-criterion is satisfied by the target, it is false if all sub-criteria are not satisfied and there are no unknown knowledge. Otherwise, the answer is unknown which may happen if the knowledge on how to satisfy any sub-criterion or about the target or sub-target is unknown. As a result, if the answer is either false or unknown, our algorithm will traverse all the criteria and their sub-criteria. Our algorithm makes use of 5 new functions, and we will only present the major logical steps omitting details for easy understanding. For example, each time an intermediate answer is obtained by using a sub-criterion, instructions to set the variable notDone and to collect the reasons for the answer are omitted. In addition, some necessary arguments of the functions such as uninitialized variables are also omitted. For example, the logical variable, notDone, is omitted but necessary to record the progress of the algorithm. They will be included in the argument list as needed to explain the solution.

Given a decision question that has the form “Is the target a classification”, if all the criteria of the classification are stored in a central place instead of distributed among many criteria, it amounts to a tree of criteria. Our solution is equivalent to traversing the tree in a depth-first manner. Our solution first calls satisfy(target,classification), which is a function to determine if the given target can be satisfiable to be called the classification. If some easy ways fail to obtain a true answer, then satSome(target, theCriteria) is called to determine if the target satisfies some criterion of the classification. For each criterion, it will attempt to find an appropriate internal knowledge value of the target, and stored that in targetValue. If the targetValue is found, the satisfy(targetValue, criterion) is called recursively since the problem repeats itself, namely, can the targetValue be satisfied to be called the criterion. Otherwise, it calls satSome(target, subCriteria) recursively where subCriteria is the sub-criterion of the criterion. The function satisfy() is shown in Fig. 1. It first calls isA(), a function that uses multiple methods to determine whether or not a target is a category [15]. If the algorithm is not done, i.e., the answer is not true, it searches within the notion to locate the learned criteria for the classification. For example, if important human is the classification, it will search within the notion of importance for any criterion learned for important human. If no criterion has been learned, then the answer is unknown. Otherwise, if criterion is found, it first searches the example list of this classification. The answer is true if the target is a known example, with its stored reason used as the reason to justify the answer. If the target is not an example, it calls satSome(), a function to decide whether the

<table>
<thead>
<tr>
<th>satisfy(target,targetValue,classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans = target → isA(classification)</td>
</tr>
<tr>
<td>if (notDone)</td>
</tr>
<tr>
<td>theCriteria = notion → search(&quot;criteria&quot;,classification)</td>
</tr>
<tr>
<td>if (no criteria)</td>
</tr>
<tr>
<td>ans = unknown</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>exampleList = theCriteria → search(&quot;example&quot;)</td>
</tr>
<tr>
<td>anExample = exampleList → search(target)</td>
</tr>
<tr>
<td>if (found) ans = true</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>ans = satSome(target,targetValue, theCriteria)</td>
</tr>
<tr>
<td>if (notDone)</td>
</tr>
<tr>
<td>ans = similar(exampleList)</td>
</tr>
<tr>
<td>return ans</td>
</tr>
</tbody>
</table>

Figure 1. A function to determine if the target can be satisfied to be called the classification
target satisfies some criterion of the learned criteria. Finally, if it is still not done, it calls similar(), a function to check if the target is similar to any learned example. To decide whether the target is similar to an example, we simply use the isA() function to determine if the target is a sub-category of the example. For instance, suppose one criterion of important human is important occupation. If specialist is an example of important occupation, then if the target’s occupation is cardiologist, then if cardiologist is known to be a specialist, cardiologist is an important occupation since it is similar to specialist. As a result, the target is determined to be an important human since the criterion important occupation is satisfied. We assume that there are a lot more examples than the number of criteria because each criterion may have many examples, so the execution time of going through the examples should be much longer than going through the criteria. Hence similar() is only executed if no criteria has been satisfied. In fact, this function will never be needed once the robot is “fully” educated since any true answer should have already been discovered by simply using the criteria.

Both satSome() and similar() functions simply go through a list sequentially, a list of criteria and examples respectively. We will only present the satSome() function, which is shown in Fig. 2. One can easily see that it simply uses a loop to go through the list of criteria sequentially. Whenever the target satisfies a criteria, the answer is true, and the decision is done. To check if the target satisfies a specific criterion, the function satA() is called. However, although there is no duplication within each list of criteria, the same criterion may be added to multiple criteria creating a cycle of criteria, thus causing an infinite loop when executed. We can prevent an infinite execution by ensuring that no criterion can be examined more than once. To achieve that, before each criterion is tried, it is inserted into a tried-list. If it has already been tried, the insertion will fail. On the other hand, if the criterion has not been tried before, the insertion will be successful. So it will be tried, and its insertion will prevent it to be tried again.

The function satA() is shown in Fig. 3. First, it calls the tempAClass() function, which determines whether or not the target satisfies the criterion using knowledge obtained from the target. This obtained value may either be passed down through the argument, targetValue, or obtained by using the current criterion passed to tempAClass(). The details on how it works will be presented later. Now if it fails to obtain a true answer, for the case if targetValue is not null, it was obtained from a higher level criterion. In this case, it will be passed down to the next level criterion by calling the satisfy() function. This call will use the original tried list since it is the continuation of the current problem but using a sub-criterion, so any tried criterion should not be tried again to avoid an infinite loop. On the other hand, if targetValue is null, the function will attempt to obtain the target knowledge using the category of this criterion. If the target has knowledge about the category of the criterion, thus setting the value of targetValue, then it also calls the satisfy() function to decide if the targetValue can be called the criterion, but this call will require a new tried list since it starts a completely new problem. For example, suppose the current criterion is scientific discovery, a sub-criterion for a higher level criterion called important achievement. If the target has an achievement knowledge of discover gravity, assume this achievement is not recognized as an important achievement, and so has to be passed along to be tested by a lower level sub-criterion. Suppose the next level criterion is important discovery, once again assume it is also not recognized as important discovery, and so will once again being pass down to the next lower level criterion through

```
Figure 2. A function to check if target satisfies any criterion

```

```
\begin{verbatim}
satA(target,targetValue,aCrit)
    ans = tempAClass(target,targetValue,aCrit)
    if (notDone)
        \{ adj = aCrit \rightarrow search(“adjective”) 
        critCategory = aCrit \rightarrow search(“category”) 
        subNotion = adj \rightarrow search(“aspect”) 
        if (targetValue is null)
            \{ targetValue = target \rightarrow search(critCategory) 
            if (targetValue is not null)
                \{ ans = satisfy(targetValue,targetValue,aCrit) 
                if (targetValue is not null)
                    \{ ans = satisfy(targetValue,targetValue,aCrit) 
                else
                    \{ ans = unknown 
                    \}
                \}
            else
                \{ ans = unknown 
                \}
            \}
        \}
    else // targetValue already found at upper level
        ans = satisfy(targetValue,targetValue,aCrit)
    \}
return ans
\end{verbatim}

```

```
Figure 3. A function to check if target satisfies a criterion

```
the satisfy() function. Now if this lower level criterion is scientific discovery, then by the time execution reaches the tempAClass() function, assume it can be recognized as scientific discovery, so it is an important discovery, and hence an important achievement. Now, for the case if $target$Value is still null, using the adjective will lead our algorithm to the notion of the criterion, and our algorithm can then search within the notion any learned criterion for the criterion, setting the variable subCriteria. If subCriteria is null, i.e., details on how to satisfy the criterion has yet to be learned, then the answer is unknown. If subCriteria is not null, then call satSome() recursively using target and subCriteria, i.e., to decide whether the target satisfy some learned criterion stored inside subCriteria.

The final function tempAClass() is shown in Fig. 4. If the classification, a criterion, is a category that has based-on knowledge about the classification, it can be used to classify the target as long as the target has the required based-on knowledge. As discussed earlier, the same fact may be stored within the target in multiple ways. Currently, we use three ways to find that based-on knowledge within the target. The first way is obtained by using the category of an upper level criterion. In this case, the $target$Value passed into the function is null. As discussed in Section 1, the appropriate based-on knowledge has to be extracted from it. If that based-on knowledge has not been obtained from the target at an upper level criterion, then the $target$Value passed into the function is null. Now, to obtain the based-on knowledge from the target, one way is to use the category of the criterion, another way is to use the based-on requirement for the classification. Now, if all three ways fails to find the required based-on knowledge, the answer is unknown. Otherwise, a temporary category is created with the found based-on knowledge added into it, then it can be called to use the isA() function to determine whether it is a sub-category of the classification. The details on how this works and why a temporary category needs to be created will be explained in two examples in Section 4.

### 4 Examples for a test question

We tested our algorithm by using the question of the form: “Is target important”. As discussed in [16], if the target is an object, then the reference is determined to be the category of the target. In addition, we also tested our solution using another form on several other classifications such as “Is pharmacist an important occupation” or “Is Nobel Prize a prestigious award”. We also performed several tests initially to verify the correctness of the program using some simple input test files. The first test involves getting a “false” answer for the given question. Once the program acquired a decent amount of knowledge, it is difficult to get the answer “false” unless it has complete knowledge, i.e., every criterion must be completely learned, must not be satisfied by the target, and the required knowledge of each target for each criterion must be known. The second test involves getting a “true” answer when the target matches exactly with an example of the classification, which is easily verified. The next test is to make sure that duplicate criteria will not cause the solution to execute infinitely, and that the designed mechanism will not accidentally prevent the use of a criterion if a new sub-question is generated. We deliberately teach the same criterion in several criteria list at different levels to make sure our algorithm will not investigate the same criterion twice to prevent an infinite loop. Finally an “unknown” answer comes up very frequently. It happens when the target does not satisfy any criteria, i.e., it is not true, and there is some missing knowledge such as certain criteria has not been learned yet or the required knowledge about the target or some intermediate target is unknown.

Next, we will explain several cases on how our algorithm obtain the answer “true” for the example question “Is target important” when the target is a human. First, assume one criterion is “prestigious award” and the target is known to have a prestigious award such as Nobel Prize. In addition, assume this is the only criterion the target will satisfy, i.e., all intermediate answers are not true. By the time our algorithm works on this criterion, it will look within the target for award, the category of the criterion. Once the $target$Value is obtained, the new question is whether or not the award is prestigious, so it calls satisfy(Nobel Prize, prestigious award). This call to satisfy() will need a new and different tried list since asking whether a human is an important human and an award is a prestigious award are two totally independent questions. Without a different tried list, potentially it can prevent a criterion from being tested if that criterion happens to be required by both questions. In

```plaintext
<table>
<thead>
<tr>
<th>tempAClass(target,targetValue,classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans = null</td>
</tr>
<tr>
<td>if (aCat=aCriterion isA(&quot;category&quot;))</td>
</tr>
<tr>
<td>basedOn = classification search(&quot;based on&quot;)</td>
</tr>
<tr>
<td>if (aCat and basedOn is not null)</td>
</tr>
<tr>
<td>if (targetValue is null)</td>
</tr>
<tr>
<td>{ basedOnValue = target search(category)</td>
</tr>
<tr>
<td>if (basedOnValue is not null)</td>
</tr>
<tr>
<td>basedOnValue = target search(basedOn)</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>{ if (firstPartTargetValue == basedOn)</td>
</tr>
<tr>
<td>basedOnValue = lastPartTargetValue</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>if (basedOnValue not null)</td>
</tr>
<tr>
<td>tempCat = new category</td>
</tr>
<tr>
<td>tempCat add(basedOn,basedOnValue)</td>
</tr>
<tr>
<td>ans = tempCat isA(classification)</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>ans = unknown</td>
</tr>
<tr>
<td>return ans</td>
</tr>
</tbody>
</table>
```

Figure 4. A function to determine if the target satisfies the classification using knowledge about the target.
addition, since the notion for important and prestigious are different, the notion for prestigious have to be identified from the adjective, and passed as an argument of satisfy(). Now, instead of using some criterion for prestigious award, we assume that it is common sense to know Nobel Prize is prestigious, i.e., it is an example of prestigious award. Hence, by using common sense knowledge, knowledge about the target, and criteria about a classification, our algorithm determines that the target is an important human, and can give the reason to support the answer.

Next, let us consider the case about Newton who discovered gravity. Assume that one of the criteria for an important human is important achievement, and one criterion of important achievement is scientific discovery. Note that, unlike the earlier example explained in Section 3, this example does not have the additional level of important discovery before scientific discovery. Assume the current criterion to be tested is scientific discovery. If the fact is taught as the discovery of Newton, i.e., discovery: gravity, then gravity can be obtained by using discovery. However, if it is stored as discover: gravity, then gravity can be located by the key discover assuming that scientific discovery is known to be classified based on what is discovered, i.e., based on: discover. For both cases, the fact is obtained in tempAClass(). Alternatively if the fact is stored as achievement: discover gravity, it has already been retrieved when important achievement is the criterion. Assuming that it is not recognized as an important achievement, so it is passed down through the satisfy() function to the next level criterion, i.e., scientific discovery. Next the algorithm isolate gravity from discover gravity by using what scientific discovery is based on, i.e., discover. Now no matter how the fact is learned, the same knowledge, gravity, is obtained. Note that all these apply to many other criteria such as important invention and prestigious nomination. The use of the based on knowledge for a classification, e.g., discover, to isolate the substring is required because it will not mix that up with other achievements such as invent airplane or nominated Academy award. Next, the problem is to recognize that discovery of gravity is a scientific discovery. Our algorithm creates a temporary category for discovery of gravity that simply contains the sub-knowledge discover: gravity, i.e., it is about discovering gravity. Finally, one of the methods used in isA() function [15], namely the based-on method, can easily determine that discovery of gravity is a scientific discovery. So Newton is an important human since he has important achievement of discovering gravity which is a scientific discovery. Note that a temporary category is created because it is unreasonable to assume that every discovery, invention, nomination, etc. is stored permanently in the knowledge base such as “nominated: best T.A.” or “awarded: best employee of the month”. This case and the case explained in Section 3 for the satA() function illustrate that no matter how many intermediate levels of criteria does the classification has, what adjectives and notions are involved, and how the fact about the target is taught and stored, our algorithm can still locate the fact and use that to determine the correct answer. Besides some of the earlier mentioned knowledge, it also makes use of the two sub-knowledge stored inside scientific discovery, based on: discover and discover: Science, i.e., it is classified based on what is discovered and that has to be about Science, and gravity is a sub-category of Physics which is Science.

Our final case is about some target who got a nomination for an Academy award. Assume that one criterion for an important human is prestigious honor, and one criterion for prestigious honor is prestigious nomination. In addition, the category prestigious nomination has been taught to be classified based on being nominated to a prestigious award, i.e., based on: nominated, and nominated: prestigious award. Then similar to the previous example, for all 3 ways of learning that fact about the target, namely, nomination: Academy award, nominated: Academy award, and honor: nominated Academy award; our algorithm is able to obtain the required fact. Now, after creating the temporary category that contains the sub-knowledge of nominated: Academy award, the isA() function is used to find out that a nomination of an Academy award can be classified as a prestigious nomination, so a prestigious honor, and the target is found to be an important human.

5 Conclusion

We have described in this paper how to find the correct answer to a decision question that involves an adjective of a notion. Our solution is simple and based on knowledge learned by the program instead of using complicated Mathematics to guess the answer. Our solution depends on the program’s ability to acquire all the required knowledge, to locate the appropriate knowledge when needed, and to infer relationships among categories. Our solution also provides the reasons to justify the answer, so the answer is more persuasive. By providing a reason may allow a teacher to easily discover what knowledge is missing, so that appropriate knowledge may be taught to remedy that. Conversely, it may also be used as a teaching tool to help human students to learn. Finally, providing a justifiable answer allows the program to make its own decision and becomes more autonomous.

Many issues still remain to be solved. One easy issue involves getting all reasons instead of just one, or the criterion is formed by using logical and instead of logical or. It can be solved easily by using additional arguments to indicate what is needed and with some minor modifications to the algorithm. More complicated logical expressions may require the integration of this solution with an existing solution to handle conditions. Another issue is related to synonyms. Since these classifications are made up of two terms and each term may have several synonyms, the total number of classifications with similar meaning is quite
substantial. For example, accomplishment and achievement are synonyms; significant, important, major and noteworthy are also synonyms. Currently, our algorithm only uses what is being taught, and so will be unable to find the needed criterion using all other synonym names. However, it is unreasonable to teach the same set of criteria for all combinations of synonyms. One expensive solution is to have the program automatically check all combinations for each criterion since it has both knowledge names separately. However, this issue is not limited to synonyms of individual words, e.g., reputable and respectable are not synonyms of important but when combined with many categories creates terms with similar meanings. Another issue is that there are many more ways a fact may be taught and stored, e.g., receive: a nomination to Academy award or get: an Olympic silver medal. Each may not be an obvious way to teach achievement, but the problem is how to obtain the fact if those are the only known knowledge about the target, and more sophisticated ways are required to isolate the needed word/sub-knowledge from the retrieved knowledge. In addition, many classifications need a lot of thought in order to teach them meaningfully so that it can be used by the program. Another issue is that a target may have several pieces of knowledge such as multiple awards or job positions. One may have to choose the best one or go through all of them. Still another issue is our assumption that the program does not have complete knowledge, so when there is missing knowledge about the target, such as no discovery, the correct answer is unknown. However, the answer should be false if the resume or biography of the target is available. A complete solution to the problem should also consider the scenario the program is currently facing, so that the program can provide the correct answer under each specific circumstance. Finally, during the testing of our program, the provided reasons allow us to discover a lot of missing knowledge when the answer is an unexpected false or unknown instead of true. The answer, although unexpected, is actually correct based on all knowledge known by the program. The provided reasons may serve as a teaching tool to help human teacher or student to easily identify missing knowledge that still needs to be learned. The amount of knowledge including common sense knowledge is very enormous. A formidable problem is how to create a comprehensive set of basic and common sense knowledge to be acquired by the program. Similar to human, this may require the program/robot to go to school to learn and be exposed to a variety of activities to gain experience. All these and many other issues may require a substantial amount of additional work in order to have a reasonable solution for the problem.

6 References


