Information as Machine Knowledge

June Benedict Parreno
De La Salle University
june_parreno@dlsu.edu.ph

Abstract: Artificial Intelligence, or A.I. for short, is a rising topic in the field of computer science that deals with finding ways to enable machines to think for themselves, to the point of almost mimicking human thought and expression. Since the 1950's, computer scientists and philosophers alike are finding way to enable computers to act and think like human beings when it comes to tasks it is assigned to do. A paper by Hubert Dreyfus, however, argued that the notion of computers being able to replicate human intelligence is not possible because it will require vast amounts of data about the known world. With the advent of files and databases it is now possible, storing massive amounts of information about the known world now becomes a relatively trivial task. But even with this, Dreyfus' statement still raises a question: are these information or data can be really considered as knowledge? Or are these just bits and pieces of data stored by the machine?I will argue that the information gained by the machine is more than mere data, but rather it is knowledge. I will argue based on the paper by John McCarthy and Patrick J Hayes entitled "Some Philosophical Problems from the Standpoint of Artificial Intelligence", specifically on the part that defines formalisms that can be used by AI to formulate knowledge. An analysis based on those papers will be made to see if these information given to the machine with A.I. are knowledge. Possible problems for the formalisms will also be discussed.

Keywords: Artificial Intelligence, Epistemology

1. Introduction

Artificial intelligence is an emerging field of computer science that existed since the 1950's with the advent of computers. This field of computer science deals with the attempt to programmatically mimic human thought and allow computers to think for themselves when it comes to making decisions. As early as 1950's, with the release of Alan Turing's paper "Computing Machinery and Intelligence?" which contains the infamous Turing Test, Computer Scientists and Philosophers alike seek to find ways to enable computers to act like humans and exhibit human-like intelligence on the tasks they are assigned to do. Some Philosophers though, like Hubert Dreyfus, argued that the notion of computers being able to replicate human intelligence is not possible. In fact, in Dreyfus' paper, which is ironically titled "Artificial Intelligence", asserted that unless computers are able to store and manage vast amounts of data about the known world, it would be difficult, if not impossible for the computer to fully simulate human thought. (Dreyfus, 1974) With the creation of databases and files however, the task of storing large amounts of information has become a rather trivial task. But even if computers are now able to store massive amounts of information, Dreyfus' statement still raises a question; are these information given to the computer with AI be considered as knowledge?

I will tackle the issue of whether or not the information given to a machine with artificial intelligence can be considered as knowledge. I will attempt to argue in support of this claim, primarily through the use of Formalisms as a way to represent knowledge as presented by John McCarthy and Patrick Hayes' paper "Some Philosophical Problems from the Standpoint of Artificial Intelligence.".

The first part of this paper will deal with how machines can gather information about the outside world. A framework of this has been presented by Diana Forsythe in her paper "Engineering Knowledge: The Construction of Knowledge in Artificial Intelligence." and will be tackled briefly in this paper. The second part of this paper will deal more with the Formalisms McCarthy and Hayes present in their paper, as well as an analysis of their formalisms and whether or not these formalisms can make information gathered by the machine real knowledge. The final part of the paper will tackle possible problems when it comes to these formalisms.

2. Gathering Information for Machines

Before machines with artificial intelligence can act their tasks on their own, they must have some form of information about the world around them, their specific tasks, and how to interact with people or other things. As an example, in her paper, Diana Forsythe (1993) describes 3 steps for the building of expert systems, a form of application that employs the use of artificial intelligence. To directly quote her:

> "Building an expert system typically involves the following steps:
> 1. collecting information from one or more human informants and/or from documentary sources;
> 2. ordering that information into procedures (such as rules and constraints) relevant to the operations that the prospective system is intended to perform; and
> 3. designing or adapting a computer program to apply these rules and constraints in performing the designated operations."

The first step is self-explanatory. It is simply the gathering of data and information about the system, what the system needs to know, it's surroundings, the specific functions that the system needs to do, and other important information that is vital for the system in order for it to function properly. The second step will be the focus of the discussion in this paper, for it deals with the formulation of rules, conventions and formalisms which the expert system will use in order to interpret the information and data given to him. The third step is simply the implementation of these sets through programming an application for the system to use.

Now that there is a need for rules and formalisms to be made for machines with artificial intelligence to follow, what then, will be the formalisms and rules that are needed for machines in order for them to know things? Or to put it in Aaron Sloman's words, based on his paper "Epistemology and Artificial Intelligence", "What formalism, or set of formalisms, is used for expressing such concepts and combining them into beliefs, rules, intentions, etc?" (Sloman, 1979) He raises the question of how to formulate concepts, information and other bits and pieces of data and what rulesets or formalisms that the AI must use in order for it to know or to act. The question of formalisms will be discussed in the next part of the paper.

3. Formalisms as a Basis for Knowledge.

McCarthy and Hayes (1969) in their paper "Some Philosophical Problems from the Standpoint of Artificial Intelligence", defined several terms in order to develop their formalism on how computers can acquire knowledge from information. First are situations. A situation *s* is the state of the universe at a given time. A set of situations can be denoted by *Sit*. A situation cannot be fully described, but facts can be used to know more about the sitution. In order to gain more information about situations, the notion of "fluents" is introduced.

Fluents are functions which are present in *Sit* of situations. There are two kinds of fluents: propositional and situational. Propositional fluents are fluents whose values are either true or false, while situational fluents are fluents whose values are within the range of *Sit*. A sample of a fluent would be the function $\mathbf{has}(e,a,s)$, where in *e* is a person named Eloise and *a* is an Apple, thus if converted into everyday language, we can get the situation "Eloise has an apple".

Fluents can be used to assert the causality of a situation. A Fluent F(π,s) asserts that the situation *s* will be followed by another situation s' that satisfies a fluent π, where π is a propositional fluent. Following the previous example of Eloise and the Apple, An example of this would be the construct $\mathbf{has}(e,a,s) \rightarrow \mathbf{eat}(e,a,s)$, which if converted to everyday language is "If Eloise has an Apple, then Eloise will eat the Apple."

A situational fluent defined as $\mathbf{result}(p,\sigma,s)$ is used for situations where actions are involved. Actions are represented by *σ*. The value of result is a situation resulting from *p* doing *σ*. Should *σ* fail to terminate, then the fluent is undefined. Following the definitions given in the previous examples and defining *b* as a basket, a sample of this formulation would be the construct $\mathbf{has}(b,a,s)$ ^ $\mathbf{near}(e,b,s) \rightarrow \mathbf{eats}(a,\mathbf{result}(e,\mathbf{gets}(e,a),s))$, which if translated to everyday language is "If a basket has an apple and Eloise is near the basket, then she will eat the apple if she gets it."

Actions are used in the construction of strategies, which are essentially a combination of actions. In a programming sense, this can be considered as an algorithm. McCarthy in his paper represented a strategy as a block of programming code. In this case, using McCarthy's example the following statements and fluents are considered as part of a strategy

(McCarthy & Hayes, 1969):

```
begin
    move(box, under-bananas);
    climb(box);
    reach-for(bananas)
end;
```

which in common language is translated into "move the box underneath the bananas, climb the box and reach for the bananas."

Finally, we can now relate these concepts to Knowledge. McCarthy did this by citing the example of knowing a combination of a safe in order to open it. (McCarthy & Hayes, 1969) Given the *sf* as the safe, *c* as the combination, *p* as the person trying to open the safe and *s* as the situation, he cited the following construct: **fits(***c*,*sf***)** ^ **at(***p*,*sf*,*s***)** → **open(***sf*,**result(***p*,**opensf(***sf*,*c***)**,*s***))**, which if translated to everyday language would mean "If a person is at a safe and a combination fits the safe, then the combination safe will open." However he later modified this statement in order to reflect that the safe is infact a combination safe and that in order for the safe to be opened, a combination is needed. Thus he ended up revising the construct and added 2 new fluents: **csafe(***sf***)** and **combination(***sf***)** in order to fomulate a new construct: **at(***p*,*sf*,*s***)** ^ **csafe(***sf***)** → **open(***sf*,**result(***p*,**opensf(***sf*,**combination(***sf***))**,*s***))** which if translated to everyday language is "If a person is at a safe and the safe is a combination safe, then if he has the right combination then he is able to open the safe." He further refined this construct in order to add in the notion of feasibility, meaning if it is possible for a person or a machine to do an action or a strategy. With this he introduced the fluent **idea-of-combination(***p*,*sf*,*s***)** which determines if the person *p* has even the faintest idea of what the combination for the combination safe *sf* is. He then equated this fluent with the combination fluent introduced earlier, thus **idea-of-combination(***p*,*sf*,*s***)** = **combination(***sf***)**.

Applying all these to a machine with some degree of artificial intelligence, lets follow the example of the combination lock. The machine should have an idea of what the combination for the combination lock is. If in his memory he does not have information about the combination, he cannot enter the code, unless there is a application inside of him that would allow him to guess the code repeatedly. However, if beforehand he was fed that information regarding the combination, he will be able to unlock the combination safe following the constructs made above. Thus, the machine is able to turn information given to him into knowledge, albeit that knowledge is only limited to those related to the tasks given to him and the constructs and formalisms that were defined within him.

Now that the it is more or less settled that through formalisms information given to a machine can be considered as knowledge, the need to properly represent that information in a manner that a machine can understand it appears forth. There are several ways to do this, including hardcoded information, however, it would completely limit the machine to whatever knowledge it can discern from that source of information. A way to represent information is through the use of XML, or eXtensible Mark-up Language. XML allows developers to define their own tags for bits and pieces of information that can be stored into a file, be read and parsed through by a program machine. Using the correct combination of the combination safe as an example, through XML we can represent the data in this form:

<combination>*123456***</combination>**

The program used by the machine could then use this information to unlock the combination safe using the formalisms described above, albeit these will be represented in a form that is understanable by the computer. The machine could also use XML to represent other information about the combination safe such as it's location, the size of the safe, the contents and other data that can be deemed as important.

Databases could also be a good way of storing information to a machine. In tables in a database, information regarding a particular situation in an organized and clear manner. Such an example of a database table can be found on Table 3.1, still using the example of the combination safe:

Table 1: Sample Database Table of all Safes with their combination codes

SafeID	SafeName	CombinationCode	Status
1	Safe1	234812	Open
2	Safe2	532987	Close
3	Safe3	112233	Close

With this database table, the machine could simply pick up the code based on the corresponding SafeID and through accessing other database tables to pinpoint the location and other details of the safe, he may be able to open it using the formalisms formulated above.

4. Possible Problems with Formalisms

The use of formalisms and rules do have it's drawbacks. One of the drawbacks is that the information that can be fed into the machine is limited by the formalisms themselves. For instance, if the set of rules is made for a specific task, like processing speech from other people and creating replies based on those text inputs, if a command is given to the machine that is not within the scope of the rules made for the machine, then the machine will either end up ignoring it and focus only on the inputs that it already has knowledge and rules to base it on, or reply in a manner that meets the formalisms defined in the machine. Meaning if you give that machine a command (not simple speech) to make ice cream for example, then the machine will just consider it not as a command, but will just process it instead as speech and will give out a reply based on whatever formalisms it has.

Another problem with the use of formalisms is what McCarthy calls the "Frame Problem". McCarthy and Hayes (1969) described the problem as having to write conditions such that certain actions will not influence or change the value of certain fluents, to the point that for every *m* actions and *n* fluents, there is bound to be *mn* conditions. In other words, the number of possible conditions will be large due to the number of actions and fluents, and not all of those actions can affect some fluents at all, thus the strategy will end up containing unnecessary formalisms. McCarthy attempted to solve that problem by introducing the concept of the "Frame", wherein several fluents are assigned to a frame and thus the effect of an action is shown by revealing which fluents have their values changed and which are, presumably, unaffected by the action. There have been many other solutions to the frame problem, but they will not be discussed in detail in this paper.

A more serious concern in this approach is that it also assumes information given to a machine is similar to epistemic belief states. This means that the computer will end up getting beliefs that may or may not necessarily be true or justified. If that were the case then how could the computer justify the information that is given to them to be true? Even if the computer is able to turn that information into knowledge through formalisms, still justifying that information and verifying whether or not these are really true or not will prove to be a challenge. Plus this is if we are working with the presumption that belief states are a form of information. A different problem arises if belief states are not a form of information. This might require a different form of processing from the computer when it is fed with a belief. How can the computer translate a belief into knowledge? Does it need information prior to it receiving about anything related to that belief? How can the computer justify these beliefs by themselves? These and other questions are to be tackled should the discussion on belief systems as intelligence is further pursued.

5. Conclusion

In conclusion, through the formalisms one can define for a machine, the information given to a machine can be considered as knowledge. These formalisms allow the machine to be able to act accordingly depending on whatever rules are present for the machine to follow when it comes to given information. Although there is still the issue of these very same formalisms limiting machines with artificial intelligence to tasks related to them, still with the evolving pace of technology there may be a possiblity that an A.I. that is capable of doing all sorts of tasks without being limited by pre-set can be developed. It remains to be seen whether or not such a thing can be truly be as sentient and knowledgable as a human person.

6. Acknowledgements

I would like to thank Dr. Jeremiah Joven Joaquin and Dr. Mark Anthony Dacela for reviewing my paper as well as giving me the necessary comments and suggestions to help shape the paper into it's current form.

7. References

Dreyfus, H. L. (1974). Artificial Intelligence. *Annals of the American Academy of Political and Social Science,* 412, 21 – 23. http://www.jstor.org/stable/1040396

Forsythe, D. E. (1993). Engineering Knowledge: The Construction of Knowledge in Artificial Intelligence. *Social Studies of Science,* 23 No. 3, 445-477. http://www.jstor.org/stable/370256

McCarthy, J. and Hayes, P. (1969). "Some philosophical problems from the standpoint of artificial intelligence". *Machine Intelligence,* 4, 463--

502.

Sloman, J. (1979). "Epistemology and Artificial Intelligence." Retrieved from: http://www.cs.bham.ac.uk/research/projects/cogaff/sloman-epist-ai.pdf