International Journal of Social Science and Economic Research

ISSN: 2455-8834

Volume:06, Issue:02 "February 2021"

INCORPORATING ABSTRACT KNOWLEDGE STRUCTURES IN MACHINE LEARNING: IMPROVING QUESTION ANSWERING, PROBLEM SOLVING AND TEACHING IN PERSONAL ASSISTANTS AND EDUCATIONAL SOFTWARE

John Leddo[1] and Ivy Liang[2]

[1]Director of Research at MyEdMaster

[2]researcher at MyEdMaster

MyEdMaster, LLC, Herndon, Virginia, USA

DOI: 10.46609/IJSSER.2021.v06i02.017 URL: https://doi.org/10.46609/IJSSER.2021.v06i02.017

ABSTRACT

Artificial intelligence (AI) and machine learning(ML) are finding their way into many applications, ranging from personal assistants to learning systems to educational software. We believe that each of these applications has inherent limitations that keep it from realizing its full potential. Personal assistants answer questions by matching queries to text strings or metadata from websites. When responses are complex, users are often given links to websites rather than answers to their queries. Many learning systems use ML algorithms that focus on statistical data analysis, thus limiting its usefulness to classification or statistical predictions. Most AI-based educational software is labor intensive to create as the underlying concept knowledge must be hard-coded into the software. The limitations that all of these applications have in common is an inability to process information (learn) and form abstract knowledge representations that support higher level cognitive tasks. The present paper describes a framework and resulting software that can read a mathematics lesson on 2-step equations that is written in English (simulating the result of a search engine retrieval), learn the underlying concepts and then apply its knowledge to answer questions, solve problems, correct the work of others who are solving similar problems, and solve problems and correct the work of others for the topic of 1-step equations, even though the material it learned made no mention of this topic.

Key Words: artificial intelligence, machine learning, personal assistants, educational software, natural language processing, problem solving, question answering, knowledge modeling

Copyright © IJSSER 2021, All rights reserved

Introduction

Each day, countless people access personal assistants and have those assistants process requests for them. Requests may include tasks like “set an alarm for 5 pm” or answering questions like “who was the first president of the United States?” For requests that involve providing information, it can be argued that most of the effort involved in creating personal assistants has involved understanding the questions being asked and retrieving appropriate information for answering the questions. Several technologies are used to perform these tasks, most notably natural language processing (NLP), artificial intelligence (AI) and machine learning (ML). In cases where the personal assistant uses a voice interface to interact with the user, voice recognition and speech synthesis also play important roles.

While recent advances have created impressive applications, there are still inherent limitations in what has been achieved. Consider the following scenario. You take out your Apple iPhone and say, “Hey, Siri! Teach me two-step equations.” A moment later, Siri responds with links to a few websites and a reply that this is what it found in response to the query. Doing the same with Google Assistant on an Android phone produces a slightly better result in that Google Assistant returns with extracted text on two-step equations that it retrieved from a website. From these two episodes, we can reach a few conclusions. First, at the time of the query, neither Siri nor Google Android knew how to solve two-step equations. If they did, they would have responded by saying something like “The first step in solving two-step equations is to…” Instead, they searched the web for relevant information. The second conclusion is that after retrieving the information, neither personal assistant has actually learned how to do two-step equations. Otherwise, it would be teaching the concept rather than passing along websites.

Now compare the same scenario but with the request made to a human. If you ask a math teacher to teach you two-step equations, the teacher will do so. Suppose, however, you asked someone who did not know what two-step equations were. That person would likely do something similar to what Siri or Google Assistant did. The person would go online, search for websites that explain what two-step equations are and retrieve the relevant information. However, the person would not stop there. The person would then read the website material and learn the concept. If the person understood the material, we would expect at least two things to occur. First, that person would now be able to solve two-step equations himself or herself. Second, that person could check your work when you solve two-step equations and validate your correct steps and correct any mistakes that you make. If the person is reasonably insightful, he or she would also realize that by learning how to solve two-step equations, s/he would now know how to solve one-step equations as well even though the person was neither asked about them or explicitly taught about them in the retrieved website text.

Arguably, we should want this to be the next evolution in personal assistants. Right now, personal assistants retrieve and provide information. For simple factual queries like “What time is it?”, retrieving information like “It is 6 o’clock” is sufficient because such information is readily understood. However, what if the query involves something more complex like how to solve certain types of equations, how to fix a flat tire, or what is the difference between a fruit and a vegetable? Mere retrieval of information may be insufficient since information is useless until someone learns it. In the future, when presented with queries for which the personal assistant has no prior knowledge, the assistant should not only retrieve relevant information but also learn that information itself. In this way, the assistant could both teach that knowledge to the user or act on it to perform tasks for the user.

Currently, machine learning is the field that seeks to address this type of issue. There are countless applications in which ML is used to first learn a concept and then perform a task using that concept. For example, neural nets have been used in pattern/object recognition (Tan et al., 2020), statistical aggregation of data from multiple sensors has been used to provide early warning for earthquakes (Fauvel et al., 2020), and ML has been applied to NLP to create chatbots (Chen et al., 2018)

The challenge here is that current ML models do not mirror how humans learn. Even ML models that use neural nets operate at a low level of cognition.Unlike humans, ML models require enormous amounts of data to learn from and cannot explain how it arrives it its conclusion (for example, an image classifier may correctly classify an image but cannot explain why it made that decision) (File Cloud, 2018). When people learn, they do more than crunch data or make low-level connections between concept features. They also engage in high level reasoning, using existing knowledge and reasoning strategies to help them learn know concepts. In many ways, this represents more the “old school” paradigms of AI when AI systems were built around constructs such as scripts (Schank and Abelson, 1977) and frames (Minksy, 1975).

Creating personal assistants that effectively respond to the query “teach me two-step equations” may be found by blending new and older AI paradigms. Consider how each paradigm might respond to the query. A new personal assistant might use voice, speech recognition, ML and AI to first process the speaker’s voice in order to extract the words that are spoken. Then, the assistant would perform a key word analysis to determine that “two-step equations” was the key concept being inquired about. From there, the assistant would search online for matching content.

Generally, personal assistants stop at information retrieval and presentation to the user. There are ML systems that go farther. For example, SQaUD, a question and answering system produced at Stanford answers questions about text by retrieving segments from the testitself

(Rajpurkar et al., 2018). Moreover, ML has been used in image processing for such things as processing checks or medical diagnosis. In those cases, one can argue that the system is actually learning the content and not just retrieving it and passing it along. A closer look, however, reveals that such systems do not model human learning processes. For example, a medical diagnosis system might be able to look at an image and say whether a person has cancer (cf., Hong et al., 2020). However, that system acquired its ability through statistical analysis of thousands of other images accompanied by feedback on whether the image really represented a cancerous or non-cancerous condition. The system could not articulate why it made the decision, whereas a human doctor could look at the same image and explain his or her judgment. Moreover, the doctor did not learn through thousands of trials or performing statistical analysis.

In summary, newer approaches to AI and ML have the strength of being flexible enough to apply similar paradigms to a wide class of problems while achieving a relatively high degree of accuracy for the tasks they perform. Their limitations seem to be that the tasks they can do, e.g., pattern matching, are rather low level. Similarly, their method of learning, usually involving crunching large quantities of data, does not match how humans learn, which often involves applying higher order thinking skills to process content.

Older methods of AI seem to have complementary strengths and weaknesses. In some of the older AI systems, subject matter knowledge and reasoning algorithms would already be built into the system. Subject matter input such as stories or newspaper articles would serve to populate generic slots in these systems' knowledge structures with specific details. For example, a restaurant "script" would "know" that a person ordered, ate and paid while at a restaurant (Schank et al., 1975). When a story was read about a trip to a restaurant, the script would be populated with details such as who went to the restaurant and what was ordered.

The benefit of this older approach to AI is that these systems could answer questions like "Why did Bob go to a restaurant?" or "Why did Bob pay after he ate?" even though information about these questions was not directly in the story. This capability is consistent with the fact that people have a lot of pre-existing knowledge they can apply to situations and is something that many current AI and machine learning systems would have trouble accomplishing. On the other hand, the older AI systems were entirely limited to the pre-programmed knowledge that it already had and could, therefore, not handle new content.

In many ways, today's AI-based educational systems are based on this older model. In software like Carnegie Learning's Cognitive Tutor (Aleven et al., 2009) and even our own (Leddo et al., 2019), the content being taught has already been encoded into a knowledge representation framework such as ACT-R or INKS (for INtegrated Knowledge Structure) and the teaching engine draws on this knowledge in its teaching. Any modifications to the knowledge base must be re-coded into the software. Even educational software that uses ML such as Squirrel AI or ALEKS does so in order to evaluate which existing lesson to provide students, not to update the concepts that are being taught.

We believe it is desirable for personal assistants and educational software to be able to learn subject matter as humans do and then incorporate that subject matter into its teaching process. This capability would give the technology the power to expand what it can answer, teach or do in response to user requests. The remainder of the paper presents a software framework we have created to accomplish this and then presents a trace of that software in action.

Our Software Paradigm

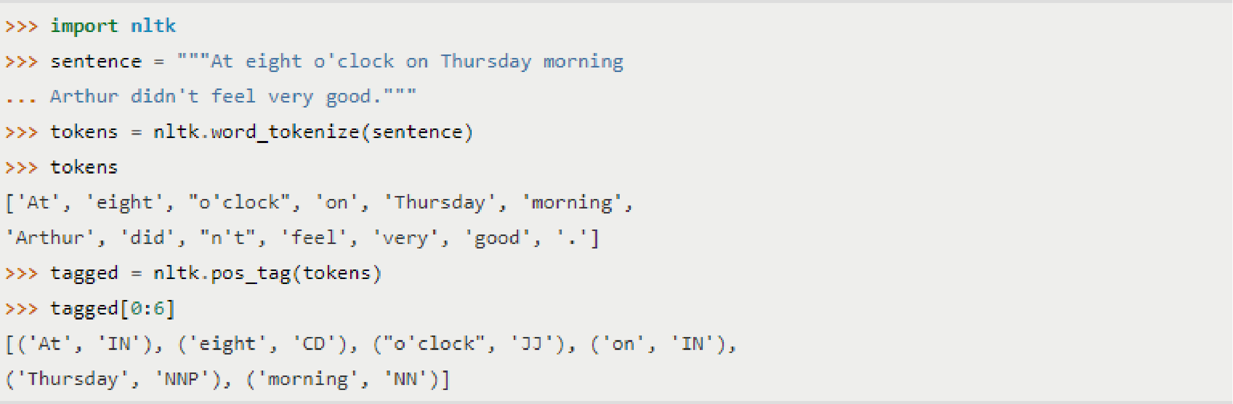
The software is written in Python. The software is designed to read and learn a mathematical problem-solving process that it has never seen before. However, as a human would do, it can bring some abstract knowledge about learning mathematics to the task, so that, in essence, it is building its domain knowledge from the top-down rather than bottom-up as most machine learning programs do. The main innovation here is that the software uses a meta-MOP (Schank, 1982) as its driving force in processing the text it reads.

In this case, the meta-MOP is for solving equations, and it represents a generalization or abstraction of how to solve specific types of equations. The meta-MOP "knows" that when one solves an equation, one starts with a statement of the problem that is to be solved. Next, one executes a series of procedures, and finally, one arrives at an answer of the form x = some value. As humans, we also know that when we learn how to solve problems, we are generally taught in terms of specific problems that we inductively generalize. Therefore, we realize that any numbers we see in practice problems are really placeholders for any other number that a problem could be based on and that the letters we see are generally placeholders for variables. The = symbol is used to indicate that what we are learning applies to equations. Using this meta-knowledge, the software is able to process a lesson on how to solve equations.

The software starts by receiving the lesson. In this case, the lesson is for solving two-step equations. It is presented below:

If the equation is 2x + 3 = 15, you subtract 3 from both sides of the equation.
This gives you 2x + 3 - 3 = 15 - 3, which simplifies to 2x = 12.
Then you divide both sides by 2.
This gives you 2x/2 = 12/2.
This gives you the final answer of x = 6.

The software starts by emulating the machine learning model used in Natural Language Toolkit or NLTK. Specifically, the initial steps of this code used the word_tokenize function, one that splits a string input into a list by, separated by each word/operator. An example of the word_tokenize function can be seen below (https://www.nltk.org/):



```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

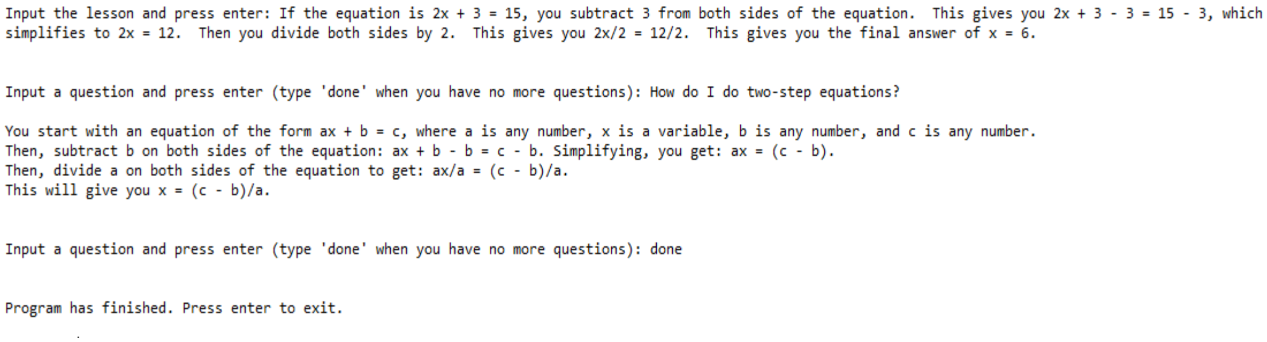
Using a method similar to the word_tokenize function, the program converts this string input into a list output. From there, the program is able to find each step of solving a two-step equation by location the "=" to indicate an equation/step. Each step of solving a two-step equation is then extracted and stored in a separate list.

Once the variables are tokenized (as the system "knows" that math formulas are generic and do not apply solely to single equations), the system populates its meta-MOP. Because the software sees an equal sign (=) in the initial equation, it assumes that what is being learned is how to solve an equation. It takes the initial step in the lesson as the type of problem being solve and the last step in the lesson as the answer. It tasks the middle steps as the process for arriving at the solution.

When asked to solve a problem on its own, the software merely substitutes the values in the problem it is presented with the tokenized variables in its representation. It then executes the procedure it learned, step by step, until it arrives at the answer. A similar approach is taken when evaluating a student's work. The software maps the student's problem and steps onto its representation. If there is a mismatch in the steps, it displays the correct step.

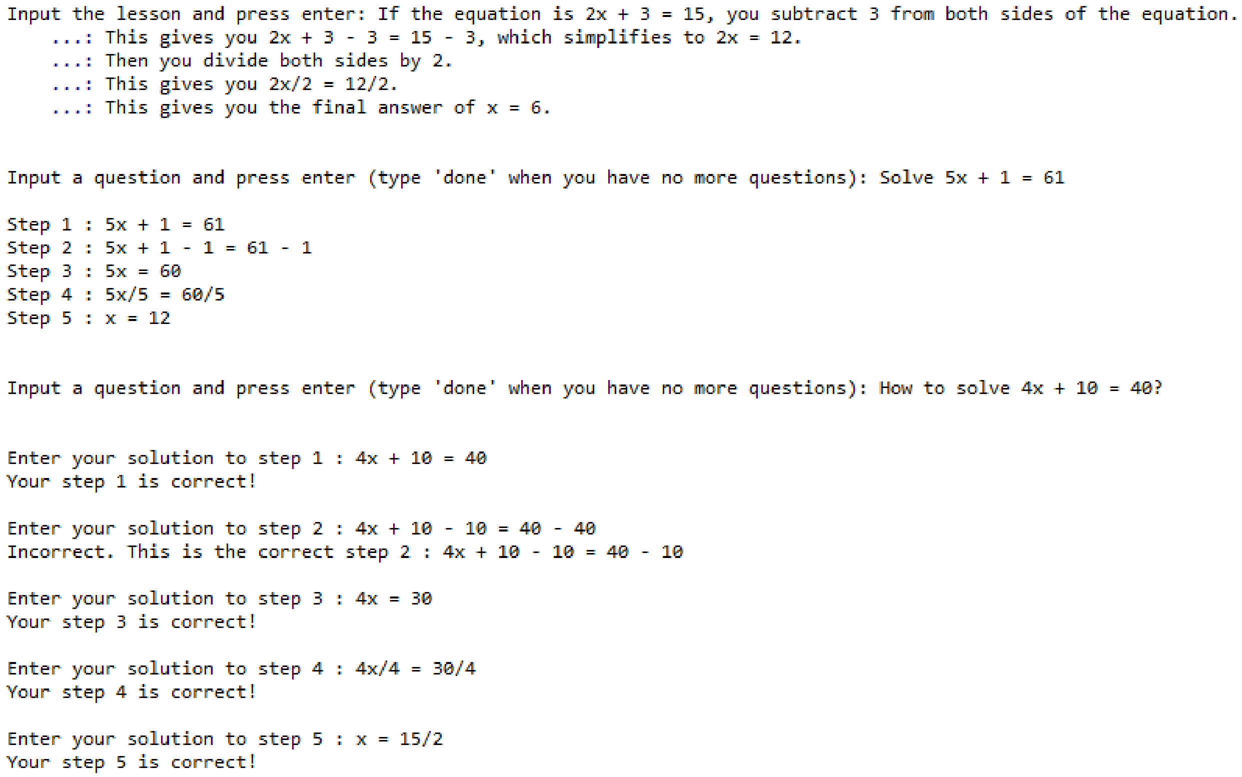
Results

The previous section describes the design of an AI/machine learning system that reads a lesson, written in English, and then performs the following tasks: answers questions based on what it read, solves problems using what it learned, corrects the work of a user solving the same problems, and extending what it learned to solving problems that use subsets of the knowledge but are not explicitly part of what it learned. Below we present the results of a test of this system on the subject matter of learning how to solve two-step equations of the form ax + b = c.



```
Input the lesson and press enter: If the equation is 2x + 3 = 15, you subtract 3 from both sides of the equation.  This gives you 2x + 3 - 3 = 15 - 3, which
simplifies to 2x = 12.  Then you divide both sides by 2.  This gives you 2x/2 = 12/2.  This gives you the final answer of x = 6.


Input a question and press enter (type 'done' when you have no more questions): How do I do two-step equations?

You start with an equation of the form ax + b = c, where a is any number, x is a variable, b is any number, and c is any number.
Then, subtract b on both sides of the equation: ax + b - b = c - b. Simplifying, you get: ax = (c - b).
Then, divide a on both sides of the equation to get: ax/a = (c - b)/a.
This will give you x = (c - b)/a.


Input a question and press enter (type 'done' when you have no more questions): done


Program has finished. Press enter to exit.
```

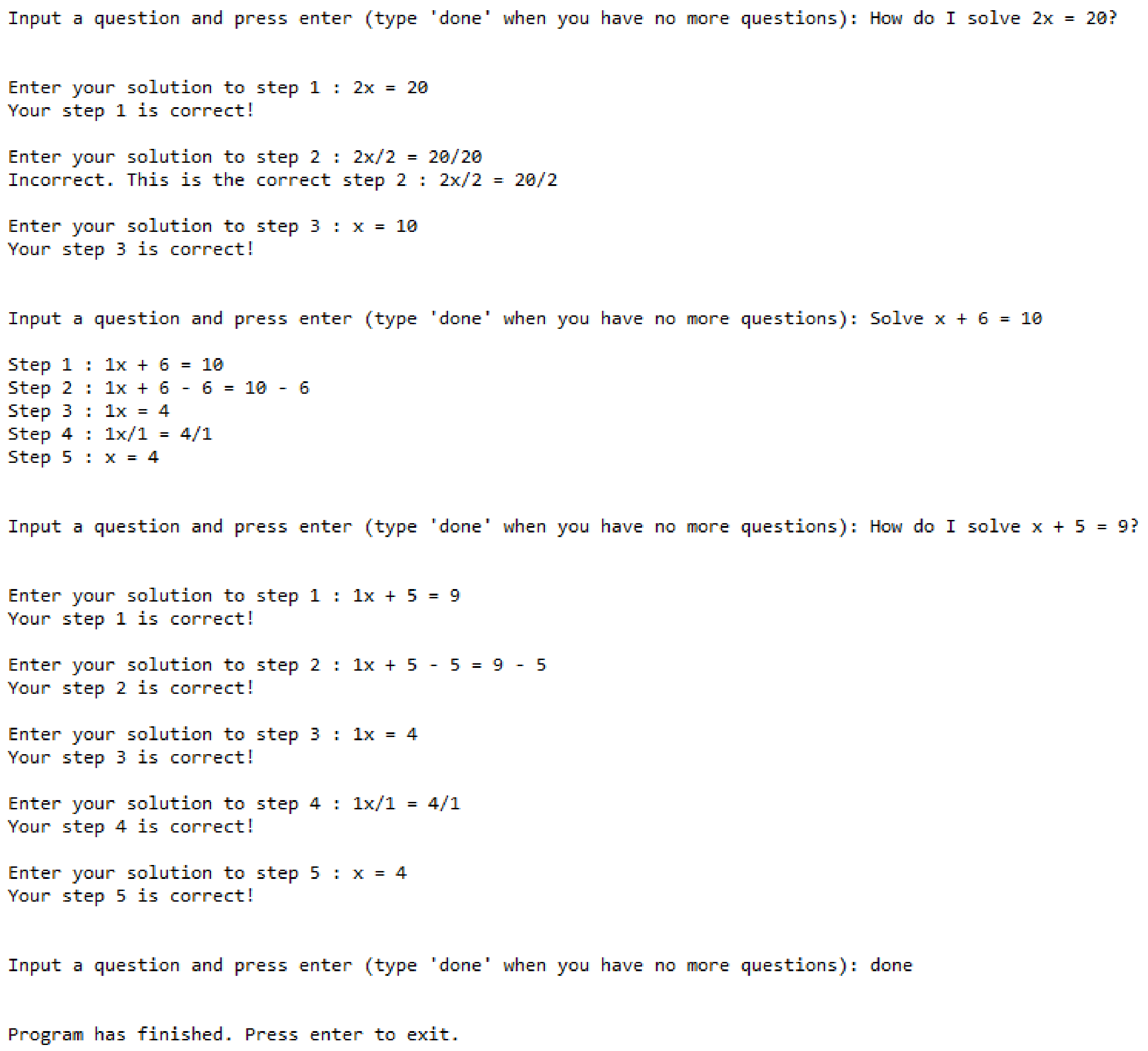
In this first screen shot, the user inputs the lesson, which the software reads. The user then asks the software the question, “How do I do two-step equations?” The software provides an answer. However, the answer is not achieved by extracting text from the initial input lesson. Rather, the software relies on its internal representation of the knowledge to construct an answer, which is why the format of the answer is generalized and not tied to the specific numbers that it learned from. This also demonstrates that the software now “understands” the process of solving two-step equations at a conceptual level, which is a benefit of using the meta-MOP as the driver of its learning.



```
Input the lesson and press enter: If the equation is 2x + 3 = 15, you subtract 3 from both sides of the equation.
   ...: This gives you 2x + 3 - 3 = 15 - 3, which simplifies to 2x = 12.
   ...: Then you divide both sides by 2.
   ...: This gives you 2x/2 = 12/2.
   ...: This gives you the final answer of x = 6.


Input a question and press enter (type 'done' when you have no more questions): Solve 5x + 1 = 61

Step 1 : 5x + 1 = 61
Step 2 : 5x + 1 - 1 = 61 - 1
Step 3 : 5x = 60
Step 4 : 5x/5 = 60/5
Step 5 : x = 12


Input a question and press enter (type 'done' when you have no more questions): How to solve 4x + 10 = 40?


Enter your solution to step 1 : 4x + 10 = 40
Your step 1 is correct!

Enter your solution to step 2 : 4x + 10 - 10 = 40 - 40
Incorrect. This is the correct step 2 : 4x + 10 - 10 = 40 - 10

Enter your solution to step 3 : 4x = 30
Your step 3 is correct!

Enter your solution to step 4 : 4x/4 = 30/4
Your step 4 is correct!

Enter your solution to step 5 : x = 15/2
Your step 5 is correct!
```

The second session trace above also starts by showing the lesson that is input into the software. After that, the software prompts the user for a question. The user enters a two-step equation, 5x + 1 = 61, which is different than the one that the software learned from. The software shows its solution path to the answer.

When complete, the software prompts the user for another question. This time the user asks the software for help solving the equation 4x + 10 = 40, which is alsodifferent than what the initial lesson contained. Upon receiving the equation, the software prompts the user to enter his/her steps. After each step, the software evaluates the step and if there is a mistake (as in step 2), the software provides the correct step. Note that the corrected step uses the actual numbers from the problem rather than showing a generic second step for solving two step equations.



```
Input a question and press enter (type 'done' when you have no more questions): How do I solve 2x = 20?


Enter your solution to step 1 : 2x = 20
Your step 1 is correct!

Enter your solution to step 2 : 2x/2 = 20/20
Incorrect. This is the correct step 2 : 2x/2 = 20/2

Enter your solution to step 3 : x = 10
Your step 3 is correct!


Input a question and press enter (type 'done' when you have no more questions): Solve x + 6 = 10

Step 1 : 1x + 6 = 10
Step 2 : 1x + 6 - 6 = 10 - 6
Step 3 : 1x = 4
Step 4 : 1x/1 = 4/1
Step 5 : x = 4


Input a question and press enter (type 'done' when you have no more questions): How do I solve x + 5 = 9?


Enter your solution to step 1 : 1x + 5 = 9
Your step 1 is correct!

Enter your solution to step 2 : 1x + 5 - 5 = 9 - 5
Your step 2 is correct!

Enter your solution to step 3 : 1x = 4
Your step 3 is correct!

Enter your solution to step 4 : 1x/1 = 4/1
Your step 4 is correct!

Enter your solution to step 5 : x = 4
Your step 5 is correct!


Input a question and press enter (type 'done' when you have no more questions): done


Program has finished. Press enter to exit.
```

The second screen shot from the session shows the software extending its knowledge of two-step equations to solving one-step equations. There are two forms of one step equations: one of the form ax = b that requires the use of multiplicative inverses to solve, and one of the form x + a = b that requires the use of additive inverses to solve. As with two-step equations, for each type of one-step equation, the software demonstrates its ability to solve problems and correct the steps performed by the user.

Discussion

The present software represents a next step in the evolution of personal assistants and educational software. Rather than merely retrieving information and presenting it to a user or extracting excerpts of retrieved text to answer questions, the present software demonstrates that it is able to process the text that has been presented to it, create an internal knowledge representation of that information and then use the information to answer questions about the material, solve problems and even correct the work of others. These features represent valuable contributions to the fields of information retrieval and question answering, general machine learning, and educational systems.

For the information retrieval and question answering functions of personal assistants, the present technology expands the current capabilities from key word search and processing to an ability to "understand" the text and apply reasoning algorithms to them, not just text matching. This was demonstrated in the present technology as the software was able to extend its knowledge of two-step equations in order to solve one-step equations, a topic never mentioned in the text, but a process that represented a subset of what was captured in its knowledge model that it constructed of the text. This has the potential to greatly enhance personal assistants by enabling them to help people learn the retrieved information, not simply present them with it. It also more closely matches how humans would respond to queries for which they did not already know the answer.

The present technology offers an enhancement to current machine learning systems that are largely driven by statistical analyses/data processing. Much of current machine learning technology is driven by pattern recognition, whether it be in image classification or big data analysis. Such machine learning has become very good at making predictions/classifications based on quantitative analysis. However, such conclusions tend to equate to low level cognitive processing like making associations (e.g., if person A buys product X, what other product will s/he buy?; if a medical image has these configuration is this associated with healthy or cancerous tissue?). Today's machine learning systems may be able to infer that people buy more ice cream in the summer than in the winter and even recommend that inventories of ice cream be increased in the summer, but they cannot explain why people like ice cream in the first place. There are systems that distinguish cancerous from health tissues, but can they explain why they call cancerous tissues cancerous and can they apply that knowledge to identify other tissue as unhealthy, even if it does not have cancer?

The present technology uses higher level reasoning skills such as meta-MOPs in order to process the information it is given. This enables the software to incorporate knowledge of goals or general cause and effect or even principles of logic to establish relationships (that may be non-statistical in nature) between concepts. Current machine learning systems may reliably predict that people pay when they go to restaurants and will conclude this based on processing many examples of restaurant dining. However, would they also predict, absent any new data, predict that a person will also have to pay to have his/her lawn mowed? A meta-MOP recognizes that restaurant episode as an example of a business transaction where money is exchanged for goods and services. Accordingly, we would expect a solid AI system that incorporates meta-MOPs to also predict that a person would have to pay to have his/her lawn mowed as this, too, is a business transaction.

The present technology can also enhance educational software. As we noted before, today's AI-based educational software is very labor-intensive to create. Each concept gets encoded into the software's database. This both limits the effectiveness of the instructional system to the encoded knowledge (and therefore makes it useful for teaching static rather than evolving concepts) and makes updating also labor intensive. Having software that could emulate a human teacher by reading, learning and then teaching would result in dynamic educational systems that could evolve to meet the needs of individual learners and changing subject matter, all without requiring extensive human intervention.

There is one more area in which the present technology may play an important role: the Internet of Things (IoT). In the future, it is widely expected that more and more devices will be intelligently interconnected and incorporate AI and ML. We can imagine refrigerators that monitor your food consumption and your shopping habits (as measured by removal and replenishment of food from the refrigerator) and lets you know when it is time to buy more items. We can imagine smart mirrors in your bathroom that conduct image processing of your body to check your health. Imagine a scenario in which a person is in the bathroom in the morning and asks the mirror, "Do I have acne?" If the mirror's technology is limited to pre-programmed conditions, it may not be able to answer that. However, an extension of the principles described in this paper would suggest that the mirror of the future would initiate a search on how to diagnose acne from facial images, learn how to do so, and then answer the user's question. Assistive robots that are asked to make blueberry pancakes for breakfast would not just retrieve the instructions from a website and pass them on to its owner but would read the instructions itself and then make the pancakes.

We believe the possibilities here are virtually unlimited. To fully exploit the technology herein, more work is needed to integrate it with search technologies so that the software can seek out its own information to learn from and also to integrate it with operational systems to enhance their functionality.

References

Aleven, V., McLaren, B. M., & Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access website for middle-school mathematics learning. *IEEE Transactions on Learning Technologies*, *2*(2), 64-78.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, ShreenathIyer, GirithejaSreenivasulu, Runxiang Cheng, Ashwin Bhandare, and Zhou Yu. 2018. Gunrock: Building a human-like social bot by leveraging large scale real user data. In Alexa Prize 2018.

Fauvel, K., Balouek-Thomert, D., Melgar, D., Silva,P., Simonet, A.,Antoniu, G.,Costan, A., Masson, V., Parashar, M., Ivan Rodero, I., and Termier, A, (2020). A Distributed Multi-Sensor Machine Learning Approach to Earthquake Early Warning. *The Thirty-Fourth AAAI Conference on Artificial Intelligence.*

FileCloud (2018). Top 5 Limitations of Machine Learning in an Enterprise Setting.

Retrieved from https://www.getfilecloud.com/blog/2018/06/top-5-limitations-of-machine-learning-in-an-enterprise-setting/#.YB7XCuhKjIV

Hong, R., Liu, W., DeLair, D. and Fenyo, D. (2020). Predicting Endometrial Cancer Subtypes and Molecular Features from Histopathology Images Using Multi-resolution Deep Learning Models. Retrieved from https://www.biorxiv.org/content/10.1101/2020.02.25.965038v1.full

Leddo, J., Guo, Y., Liang, Y., Joshi, R., Liang, I., Guo, W. and Bailey, S. (2019). Artificial Intelligence and Voice-powered Electronic Textbooks. *International Journal of Advanced Educational Research, 4(6),* 44-49.

Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision*. New York McGraw-Hill Book.

Rajpurkar, P., Robin Jia, R. &Percy Liang. P. Know what you don't know: Unanswerable questions for SQuAD. In Association for Computational Linguistics (ACL), 2018.

Schank, R. S. (1982). *Dynamic Memory* : *A theory of reminding and learning in computers and people*. New York: Cambridge University Press.

Schank, R.C. and Abelson, R.P. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Erlbaum.

Schank, R.C. et al. (1975). *SAM – A Story Understander*, Department of Computer Science Research Report No. 46, Yale University, New Haven, CT.

Tan, M., Pang, R. and Le, Q.V. (2020). EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*