

Artificial Intelligence Neural Network

Software Version: 0.0.3

Paper by:
Dexter Rio Charles Shepherd

Problem

In the 21st century humans, lead busy lives. You don't have time to clean the house as you have to work and when you get home you have to cook. We get little time to ourselves.

If we are too busy and do not get enough rest research shows, we are more likely to have strokes and suffer breakdowns. Living in a messy house is also no way to live and can prevent you resting. Can we really be bothered to clean the house after a long day of work?

Elderly people need care, as do disabled people. When there is no family, around they need a carer but there are not enough carers around to cater to the persons need.

How can we do all these complicated jobs when we are too busy to do it, don't have enough people to do it and don't have the money for a servant?

The system in this paper is a learning algorithm, which learns specifically what the user teaches it. The system can apply its knowledge in similar situations. This system can help millions of people by making their lives easier.

The system can do whatever you can do as it learns through interaction. You can come home to a tidy home and a nice cooked meal every day so you can put your feet up and not have to worry about anything.

Another issue is that current AI on the internet is that people cannot teach it, cleverbot doesn't know who you are, how can you have a conversation with something which takes limited data in.

Planning and Design

Task

Building on V0.0.2, the AI, SHEP, in this version is edited to have many features to enhance its intelligence and ability to act like a human.

Firstly, the AI must be able to update its current data. This prevents the AI being stuck on old knowledge and not being able to update itself.

Secondly, the AI should have the ability to reply on the topic of conversation. This means SHEP can reply to vague comments by remembering what he is currently speaking about. A problem with many Chabots is that many of them do not remember what was said.

SHEP should be able to paragraphs into separate commands. When saying long sentences to SHEP, SHEP only looks at the beginning; consequently, the end is ignored. If SHEP could view other parts of a sentence and check them, it means SHEP can form his own paragraphs in response to the user.

When subjects are said, SHEP should store them as variables so he can call on them throughout conversation. This means SHEP can remember what has been said. A list could store the going in information, and another list store what is it. Example: variable1 [position] = weather, variable2 [position] = "sunny". SHEP cannot see the weather but he can pick up the information and remember what the weather is.

SHEP should know if the user has asked him something already. If you asked a human repeatedly "what are carrots", they would answer the first time, but then get bored and angry.

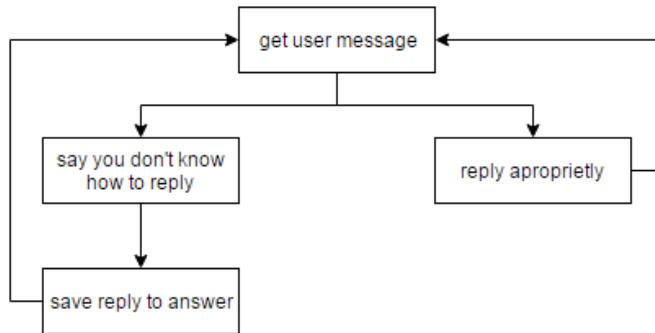
In a sentence with multiple triggers, subjects and/or commands, the system only registers the first one found. The AI should be able to ask the user, which word is wanted, and use that in the system pathway.

Requirement

The AI will be able to take in data and output the most appropriate response it can think of. The AI is able to navigate through a network of files to find the output. The system can learn from the user's teachings, and then append this to its data.

SHEP is able to remember what is said in the sentence and possibly deceive a human into thinking he is human. To test that I will have to set up an input/output system using HTML code on a random chat website, but this is not for this version.

User Interface



Success Criteria

I will know the system works when it can do everything my test plan says.

Test plan

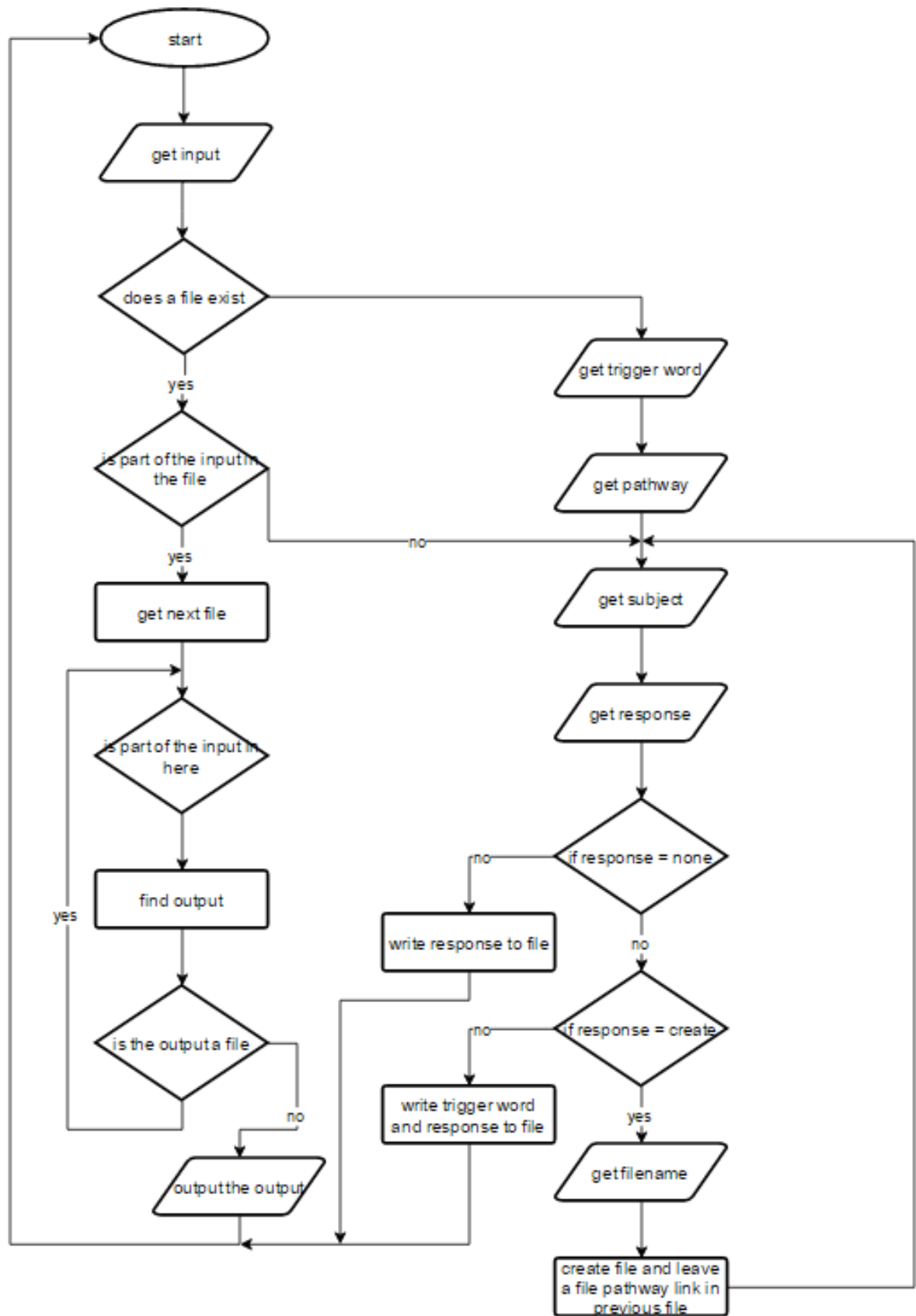
Test Number	Test	Expected result
1	To check SHEP can learn information, then change it and SHEP updates his data.	When you ask something, Shep answers, and then when you teach it the new answer Shep answers that.
2	To check SHEP can answer based on the topic of information, the user will ask, "what is the weather like" to which "weather" is the subject. After SHEP's response, the user will say, "it is sunny".	Any response other than "I fail to see the subject we are discussing" means this worked.
3	The user enters two points into one sentence.	SHEP responds to the first one, separates it with a "." Then responds to the second one in the output message.
4	SHEP can learn volatile subjects and answer to them.	When the user asks, "what is the weather", SHEP responds, "I believe the weather is:" and whatever the user said.
5	SHEP knows if you have previously mentioned something.	When you say something, and then again, it responds "we have already said this, I say again:" and then the output.

6	A sentence with multiple triggers is entered into SHEP.	A box comes up and asks which word is wanted
7	A sentence with multiple subjects is entered into SHEP.	A box comes up and asks which word is wanted
8	A sentence with multiple commands is entered into SHEP.	A box comes up and asks which word is wanted
9	A word, which is not in the string, is entered into the system.	The box says "invalid" and re-opens itself.
10	A normal sentence is entered into the system which is saved	The saved reply is output.
11	The user enters a sentences not in the data to update	The AI prompts the user it does not exist and resets.
12	The user does not enter a trigger, subject or a command	The AI tells the user that none is present
13	The user enters nothing	The AI says nothing is inputted and resets
14	The AI saves its data	Anything it learns can be called upon and it will output what you told it to
15	The user enters something without a subject after saying something is a subject but nothing is in the data for the sentence	The AI checks with the old subject, but when nothing is found it does nothing
16	You can add words to the variable memory	The user types it into a box and then it comes up in the variable list in the console.

Hardware

A computer with python to run the code on (Pc, Raspberry Pi)

Flowchart



Pseudocode

Check to see if the user has entered it before

If the users message = the gathered conversation string THEN
 OUTPUT ("this has been said before, I say again: " + output message)

User has more than one trigger

Loop until all triggers are found
 If trigger found in string THEN
 Add to list
 E = trigger
 Position += 1
 ENDIF
If there is more than 1 trigger THEN
 Loop:
 Open up the selection box
 E = INPUT
 If E is invalid THEN
 Loop back
 Else
 ENDLOOP

Trigger = E

User has more than one subject

Loop until all subjects are found
 If subject found in string THEN
 Add to list
 E = subject
 Position += 1
 ENDIF
If there is more than 1 subject THEN
 Loop:
 Open up the selection box
 E = INPUT
 If E is invalid THEN
 Loop back
 Else
 ENDLOOP

Subject = E

User has more than one trigger

Loop until all commands are found
 If commands found in string THEN
 Add to list
 E = commands
 Position += 1

```

ENDIF
If there is more than 1 commands THEN
  Loop:
    Open up the selection box
    E = INPUT
    If E is invalid THEN
      Loop back
    Else
      ENDLOOP

Commands = E

```

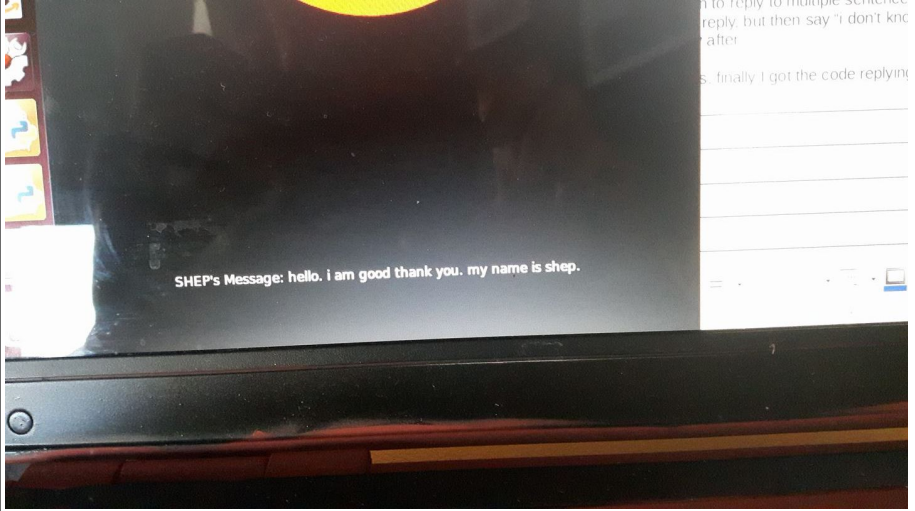
Coded Solution

Development

Development diary

Date	What I worked on
30/05/2017	<p>Today I started work on this development paper. I already had many ideas to improve SHEP, however when talking to him today I noticed a problem. This problem was that SHEP would get words mixed up, for example:</p> <p>“what is your name” “what” = trigger – question “name” = subject – name “Your” and “is” are both commands, it picked “is” as it is the first one. This was annoying because when I said “my name is Dexter” it replied “my name is Shep”. Now this works in conversation, however, it is following the wrong command word. This made me think about adding the function of the code finding all relevant words, and then asking the user which word is wanted.</p> <p>This version is going to have some complicated programming and difficulties. I can already see myself up late at night, drinking high caffeine drinks, and banging my head against a table because there is a bug in SHEP’s code.</p> <p>I will attempt the easier stuff first, then develop the complicated stuff</p>

	last to make sure the system still runs through.
16/06/2017	<p>I have finally finished my exams! Further more, I can work on SHEP throughout my holidays, as well as go out (because I am not a total shut in).</p> <p>Tonight I worked on SHEP's code so the main code function works as a return function. A new function gathers what is found within the main function, this builds the foundationd of all the aspects of this version I want to add in.</p> <p>I will need to do some reforming of the code still, such as the trigger gathering all the possible triggers, same going for subjects and commands. This will then be fed into the main function to check through. I may do this by reforming the current functions (trigger, subjec, command) and adding a new function to form all possible sentences, these sentences are fed back into a list, where it is repeatedly tested.</p>
18/06/2017	<p>Today, so far, I have made SHEP's vocabulary checking functions work to the standard I want them to. It now desplas all the possible vocabulary in a list. There are still a few bugs in the command function, however I can work through these.</p> <p>I need the system to then desplay possible outcomes for the sentences and repeatedly look for the answer till it either finds one or does not.</p> <p>Another idea I came up with today, is to organise the data into folders, rather than files. This means that data wont be crossed, for example; a statement about a subject, and a question about the same subject will lead down the same route even though different outcomes are expected.</p> <p>After many bugs, I finally got the code to sort out the subjects and commands into trigger folders. The folders have nicely organised the data and I am over all happy with it.</p> <p>I now must get back to working on the vocab function.</p>

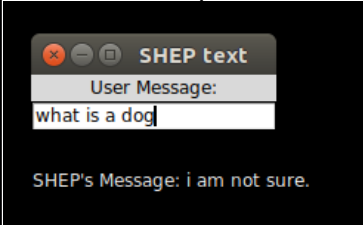
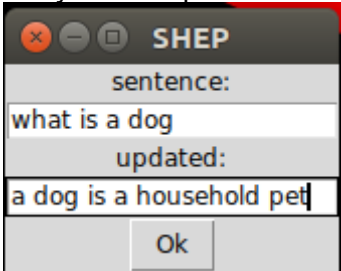
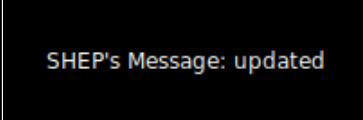
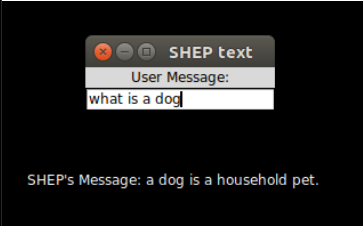
	<p>I worked on the function to reply to multiple sentences in one. The AI so far has managed to reply, but then say "i don't know how to repond to that" shortly after.</p>  <p>1. After hundreds of errors, finally I got the code replying to multiple sentences.</p>
19/06/2017	<p>I have added a save and open function. This allows the user to save the conversation they are having with SHEP, and then open it up to continue. The benefits of this is that if you are descussing a subject, SHEP will know what to talk about.</p> <p>It had many bugs creating this saving process but I have got it to a working stage. I have not added ability to look for the last sentence said yet, this will be added to the open function.</p> <p>I have added the ability to understand repetition. When asking SHEP the same things, SHEP responds "as I have previously stated, " and then the output.</p>
22/06/2017	<p>I have added the update function. When going to edit, an option of update appears in the list. The user can select this, type in the existing sentence, the new output wanted and update the data.</p> <p>The code works and fully updates the string. I wanted the normal text to also be used by typing in "/update>" however it would not open up a new tab.</p> <p>I would like the update function to be a bit more natrual in conversation, however for now it is fine. Perhaps I could make it so when the AI says something wrong, the user says "no instead say>"and type in the new word. This will be possible in the near future.</p> <p>Today I added more tests to the test plan, this means I can proove the AI is bug free.</p>
23/06/2017	<p>Today I added the use of old subject function. When you have a</p>

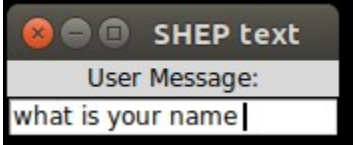
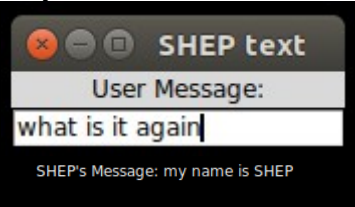
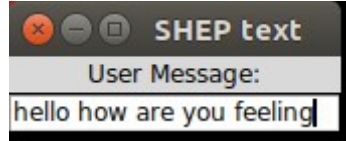
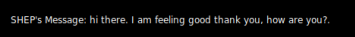
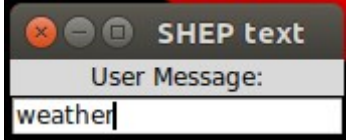
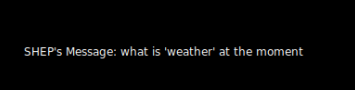
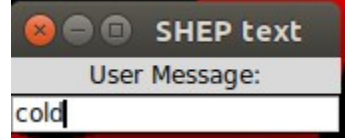
	<p>normal conversation with the AI, you can continue to discuss a point without using any subject words. "how are you feeling" with "feeling" as the subject - "how are you" does not have a subject but will carry on the "feeling" to get the same output as "how are you feeling".</p> <p>I decided to start a plan for designing the variable memory function. There will be certain subjects which are not constants, such as; weather, what is going on. Perhaps these variables could be added to a list and SHEP could call upon these variables when they come up in conversation.</p> <p>Unrelated to this version, I stumbled upon the idea to create a sort of log of what the user does. If I was to tell SHEP I was walking along Shorham harbour looking at boats, after stating what I was doing today, shep would create a log, and find out what kind of things I do to create topics to talk about. The only problem is I must think of a smart way to incorporate this into the program as a natural thing.</p> <p>If I program inbuilt commands into SHEP too much he is no-longer learning for himself. I must plan out a method to make SHEP naturally take notes, by understanding key points which take notes.</p> <p>When testing out SHEP, I came across a bug. When I say something, then ask something different, it responded the same as last time due to two similar words. This was annoying because it made me re-think the subject function. I may have to get rid of the function and re-think how or if I shall re-do it.</p> <p>I will rethink how it shall work, and return with a solution, or a decision to delete it.</p>
24/06/2017	<p>I added the variable abilities today, I came into many complications throughout the day. There were many bugs, but I managed to work through them.</p> <p>The AI now saves the variables and splits them into a list, it gathers the answers to them through conversation and then outputs them when asked.</p> <p>When adding multiple variables, I came upon a problem. The AI outputted "the weather is 24/06/2017". This was not the date nor weather. I managed to fix this by rearranging where loops were within the code.</p> <p>The AI now works with volatile data, this means all I have now is the multiple vocab functions I started at the beginning. I do have a few odds and ends to clear up, such as; making the old subject in sentence work better, making the update function work more naturally in conversation.</p>
29/06/2017	<p>When looking into the SHEP program, the vocab function proved itself to not fully work. With sentences such as "is that a dog" and "that is a dog", they both have two different meanings with words switched around. If I added one of them, the vocab function would check the</p>

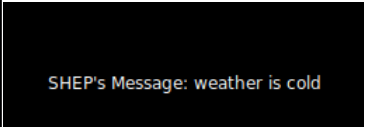

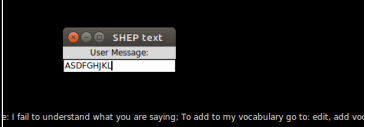


	<p>trigger words then output the answer to the one in the data.</p> <p>For example: "is that a dog" = "no" "that is a dog" = not in data – check for other trigger words – sentence="is a dog" - output = "no"</p> <p>I have completed all the tasks, other than the multiple vocab function. As I am no-longer doing that task, SHEP version 0.0.3 will be complete after I tidy up a few loose ends.</p> <p>These loose ends contain:</p> <ul style="list-style-type: none">• making updating more natrual.• Adding short cut keys to the menus.• Making the open funntion load the file into the program.
--	--

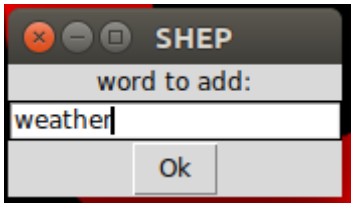
Testing

Test results

Test number	Test	Pass/fail	Comment
1	To check SHEP can learn information, then change it and SHEP updates his data.	pass	<p>I tested out the function and it worked as expected.</p>  <p>This shows my input and the system output ^</p>  <p>Above shows the system update box. The system then confirms the update has worked, as shown below.</p>   <p>Above shows the test to see if it updated, as you can see it has.</p>

2	To check SHEP can answer based on the topic of information, the user will ask, "what is the weather like" to which "weather" is the subject. After SHEP's response, the user will say, "it is sunny".	pass	 <p>I firstly entered the normal sentence. I then got the normal output.</p> <p>Then I entered a sentence with the trigger and command but not the subject.</p>  <p>As you can see it works fine, the subject "name" is carried on.</p>
3	The user enters two points into one sentence.	pass	 <p>Above shows the input of two sentences.</p> <p>The AI outputs two answers in the punctuated form of a paragraph.</p> 
4	SHEP can learn volatile subjects and answer to them.	pass	 <p>I inputed the subject.</p> <p>Shep gave me a message saying ^.</p>   <p>I told Shep it was cold.</p>

			<p>When I asked again Shep said:</p> 
5	SHEP knows if you have previously mentioned something.	pass	When the user says the same thing twice, the AI will recognize this, then tell the user it has happened.
6	A sentence with multiple triggers is entered into SHEP.	---	Function not used
7	A sentence with multiple subjects is entered into SHEP.	---	Function not used
8	A sentence with multiple commands is entered into SHEP.	---	Function not used
9	A word, which is not in the string, is entered into the system.	---	Function not used
10	A normal sentence is entered into the system which is saved	pass	<p>The AI outputs it to the user.</p> 
11	The user enters a sentence not in the data to update	pass	 <p>SHEP does not understand and cannot find vocabulary, so tells the user this.</p>
12	The user does not enter a trigger, subject or a command	pass	
13	The user enters nothing	pass	 <p>This was a similar test to test 12</p>
14	The AI saves its data	pass	The system could not

			function if this was not the case. When I ask shep something, shep outputs it.
15	The user enters something without a subject after saying something is a subject but nothing is in the data for the sentence		
16	You can add words to the variable memory	pass	<p>The user types it into a box and it is added to the list. <code>['weather', 'date', 'today']</code> Above shows the console's output. Below shows the box:</p> 

Future ideas

SHEP could write his own code, SHEP could open up the “.py” file and write into it. This means SHEP can expand his own ability and re-write how he works. In the far future, SHEP could have access to all his code and change what he needs to when. This means SHEP is capable of evolution, to an extent.

Evaluation

I feel like SHEP works in a good well, and the new file management is a better way to prevent data cross-overs. However, SHEP has many issues within the algorithm. The ability to check the input for multiple triggers, subjects, and commands was a failure. It prevent simular sentences from being added which had very different meanings. This being said, without this function sentences may use the wrong command words and answer wrong to different messages.

I believe that the new folder organization method will help prevent this, but I can see problems arising.

Some sentences have logic which this algorithm cannot understand. This logic could be multiple subjects. A basic sentence has the trigger, subject and command. More complex sentences have lots more layers. Possibly in the next version I could add the ability to add

more layers of vocabulary to SHEP. SHEP should be able to work with the 3 main, but if he detects another word in the next layer, SHEP will continue to go down that route and find a specific answer.

The next version should include:

- The ability to write a sector of its own code to help it think in situations and change small roles.
- The ability to add layers of vocabulary to answer complex sentences
- Can look at lots of data to solve a problem – this links to task 2.
 - Example learn to play chess
- Can learn/create equations to solve problems
 - Volume of a square
 - Answer a maths paper