A Text-Based Dialogue System using AIML

Daniel Christopher Peric 25811460

Andisheh Partovi 22999825

Introduction

We have used the Artificial Intelligence Markup Language (AIML) framework to create a text-based dialogue system (a chatterbot) that can handle a very short conversation about electricity. We have chosen a student persona for the chatterbot and imagined a scenario in which an energy consultant tries to help him save money by giving him some tips and answering his questions.

In this report, we first introduce the system and some of it capabilities (The Chatterbot), then discuss the procedure we undertook to develop the chatterbot (Procedure) and the evaluation approach we used to test it (Evaluation). We will also discuss different dialogue phenomena we handled in the system (Handling Different language and Dialogue Phenomena) and some of the limitations of the used framework (Framework Limitations).

The Chatterbot

We designed the chatterbot based on a scenario and a persona. We coded several attributes pertaining to the bot's persona and scenario (such as his and his roommate's name, their power bill amount, etc.) in the "*assignment2.py*" file.

We imagined the following scenario as the chatterbot's use case and based our design on it:

"*The user is an energy consultant from an organisation that wants to help people save money by reducing their energy consumption. He is going from door to door, talking to people, giving them tips on how to save money on power bills and answer their questions about how to save energy and money.*"

Therefore we coded the chatterbot to be able to answer some questions regarding his power bill and electricity consumption, ask questions regarding some of his appliances (a fridge and an Xbox), and receive tips on saving energy.

The procedure

1. We started off by investigating AIML and reading about electricity. We looked at the existing AIML modules on Alicebot website[1] to familiarise ourselves with its syntax and capabilities. From these libraries we decided to use the "adverbs library" that removes the adverbs from the user utterances and the "greetings library" that just includes greetings and farewells (the copyright notices are in the AIML file).

[1] http://www.alicebot.org/aiml/aaa/

2. We had several conversations with each other in a text chat environment where one of us had the role of the student and the other, the consultant. One of these dialogues is provided in the Appendix I.

3. After that, we distilled the dialogues by removing duplicate utterances (across multiple dialogues), and combining similar questions and answers. As we want a natural human-like chatterbot, we didn't modify the 'system' utterances to look more mechanical. At this stage, we also identified some topics that we needed to code for the chatterbot.

4. We implemented the first prototype of the system using this small corpus by adding AIML tags to utterances, adding alternative ways of saying the same questions, and abstracting out the keywords for our selected topics. We also added a few number of common synonyms of the keywords using a thesaurus dictionary. Given the nature of AIML, we tried to merge as many questions as we can into one and took a greedy approach with responses. For instance, any question that includes the word "pay" will be responded with the amount of both summer bill and winter bill. This greedy answer that tries to give as much information as they are relevant in one go, is very helpful in reducing the number of questions that must be answered, although, it sometimes make the answers seem non-organic.

5. An AIML specific implementation step was dealing with topics. We used binary variables to keep track of discussed topics. Once a topic is discussed, its flag variable is set to true so the bot doesn't initiates another talk on the topic nor allows the user to bring it up. A limitation of AIML is that, these variables should be initialised in a category and if that category isn't called (the user never asked the question in that category), the variables will not hold any value. We placed the initialisation in the "HELLO" category and by adding a greetings as the first utterance of the bot, tried to encourage the user to say a greetings and therefore initialise the variables.

6. We did the WoZ experiment over a shared desktop chat environment. The wizard typed the utterances in and read the system responses. In AIML that the system is in fact just the database and the database size is massive, a WoZ approach where the wizard only consults the database, seemed cumbersome and without a point.

7. We analysed the log files and made major modifications to the database based on the dialogues in the previous step. A topic was added and the way the unknown utterances are handled was changed completely. So by making these modifications, we developed our second prototype at this stage.

8. We repeated steps 6 and 7 one more time, and this was our final modification of the chatterbot. With a system like AIML, one can continuously improve the system by adding more and more categories but given the time limitation, we had to stop at some point.

9. We performed the user testing and evaluation (See the Evaluation Section).

Handling Different Language and Dialogue Phenomena

The following are some of the language and dialogue phenomena and how we decided to handle them in our system:

1. **Policy and handling unknown utterance:** system has a strict policy as any known utterance must be hard-coded in the system. We have a wildcard category that catches all the unknown utterances and based on the discussed topics, tries to steer the conversation towards topics that are already coded. For instance, if the user hasn't discussed the fridge yet, the system returns a fridge-related interruption as the answer for any unknown utterance, hoping that this will prompt the user to talk about the fridge. When the first topic is done, it will move to the next, using cue phrases such as "by the way". When all the three topics are exhausted, the bot will mention that he has to go.
2. **Digression:** As alluded to in the previous point, any utterance that is not coded, is answered by a hint towards known topics. The unknown utterances could be digressions or simply be variations of the existing categories. Because there is no way to detect this in AIML, as it is not feasible to include all variations of all the categories, we have to treat these two situations uniformly.
3. **Initiatives:** We have taken a mixed initiative approach, as the bot must both ask and answer questions. Because we can provide the first utterance through the python driver, and we have leveraged that to encourage the user to greet the bot, the very first interaction is system initiative. After that, interactions will be user initiative until the user brings up one of the appliances in the bot's house, then the bot will take control of the conversation for two or three turns. Because this is a fixed rule, the system is not variable initiative. This mixed initiative approach is natural to this scenario, therefore we utilized it.
4. **Dealing with social obligations:** The system is coded to understand near to 80 different ways of greetings and farewells. This shows the core limitation of AIML that needs all the alternative ways of saying the same thing, be coded in. The system also includes answers to thanking.
5. **Grounding and Clarifications:** using the recurrence tool (srai), continuer utterances such as "yes" and "right", are ignored and mapped to whatever comes after them. We are losing some grounding information in this way, but these tricks are necessary in AIML. The bot cannot ask clarification questions either so basically we couldn't find any effective way to handle grounding in AIML. Considering that the system is text-based, fortunately, grounding is not as vital as spoken dialogue system.
6. **Pronouns:** this cannot really be handled in AIML as it doesn't perform any natural language processing. We tried to include pronouns in the categories with the "that" tag, just to give an illusion of intelligence.
7. **Grammar:** We decided not to use a grammar for the input or output utterances for two main reasons: first, systems with grammars do not feel natural or human and secondly, the aim of this chatterbot is to have a conversation rather than extracting or presenting information; ideally it should be able to handle any form of sentence structure.

Evaluation

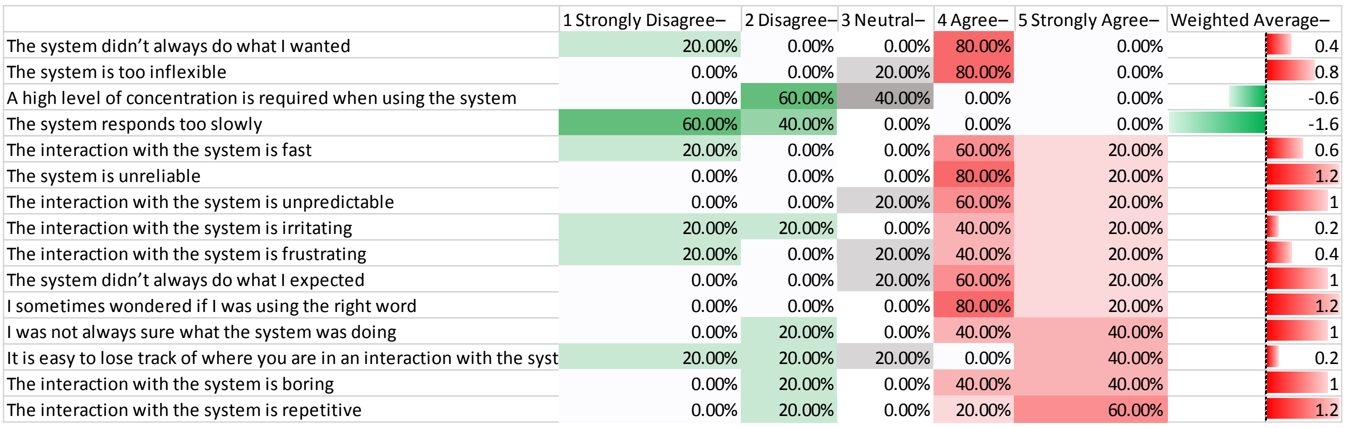
We tested the system with 8 users, giving them the same scenario (see the Chatterbot Section) and recorded their conversations in a log file. Table 1 summarises the recorded conversation length and the number of good/bad utterances for each user.

user 1		user 2	
conversation length	26	conversation length	12
bad utterances	61.54%	bad utterances	16.67%
good utterances	38.46%	good utterances	83.33%
user 3		user 4	
conversation length	19	conversation length	12
bad utterances	66.67%	bad utterances	0.00%
good utterances	33.33%	good utterances	100.00%
user 5		user 6	
conversation length	4	conversation length	33
bad utterances	0.00%	bad utterances	17.65%
good utterances	100.00%	good utterances	82.35%
user 7		user 8	
conversation length	30	conversation length	31
bad utterances	20.00%	bad utterances	25.00%
good utterances	80.00%	good utterances	75.00%
		Average	
		conversation length	20.87
		bad utterances	19.70%
		good utterances	80.30%

Table 1. Conversation Length and the Number of Good/Bad Utterences per User

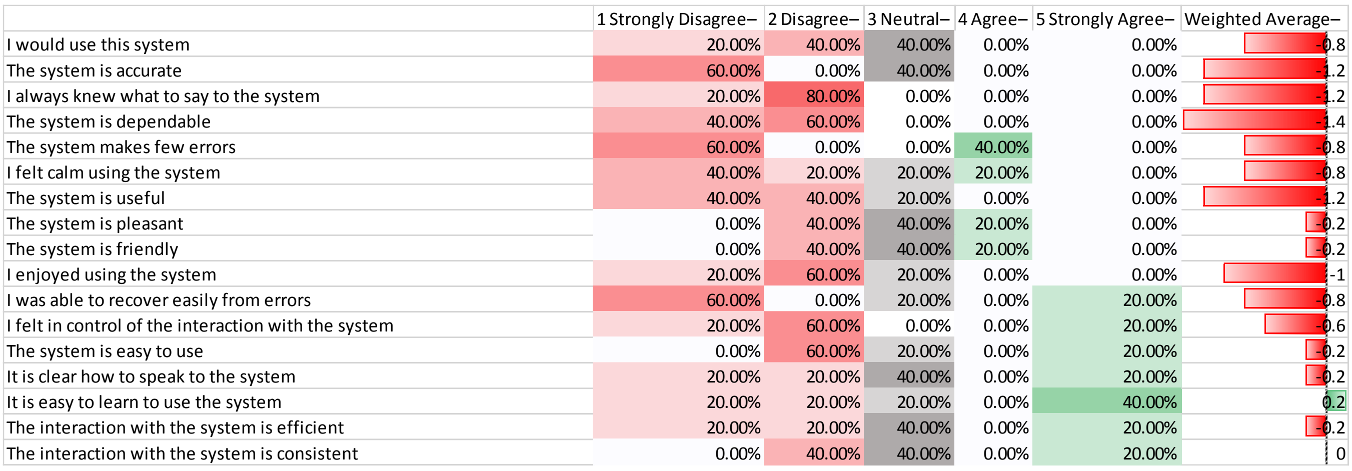
As for a subjective evaluation, the users were asked to fill out a survey after the experiment in order to measure their satisfaction with the system. The questionnaire is developed by Hone and Graham (2000)[2] as part of the SASSI evaluation framework and uses the Likert scale. The average results for questions are colour-coded and presented in Figures 2a and 2b.

[2] Hone, K. S., & Graham, R. (2000). Towards a tool for the subjective assessment of speech system interfaces (SASSI). *Natural Language Engineering*, *6*(3&4), 287-303.



	1 Strongly Disagree–	2 Disagree–	3 Neutral–	4 Agree–	5 Strongly Agree–	Weighted Average–
The system didn’t always do what I wanted	20.00%	0.00%	0.00%	80.00%	0.00%	0.4
The system is too inflexible	0.00%	0.00%	20.00%	80.00%	0.00%	0.8
A high level of concentration is required when using the system	0.00%	60.00%	40.00%	0.00%	0.00%	-0.6
The system responds too slowly	60.00%	40.00%	0.00%	0.00%	0.00%	-1.6
The interaction with the system is fast	20.00%	0.00%	0.00%	60.00%	20.00%	0.6
The system is unreliable	0.00%	0.00%	0.00%	80.00%	20.00%	1.2
The interaction with the system is unpredictable	0.00%	0.00%	20.00%	60.00%	20.00%	1
The interaction with the system is irritating	20.00%	20.00%	0.00%	40.00%	20.00%	0.2
The interaction with the system is frustrating	20.00%	0.00%	20.00%	40.00%	20.00%	0.4
The system didn’t always do what I expected	0.00%	0.00%	20.00%	60.00%	20.00%	1
I sometimes wondered if I was using the right word	0.00%	0.00%	0.00%	80.00%	20.00%	1.2
I was not always sure what the system was doing	0.00%	20.00%	0.00%	40.00%	40.00%	1
It is easy to lose track of where you are in an interaction with the syst	20.00%	20.00%	20.00%	0.00%	40.00%	0.2
The interaction with the system is boring	0.00%	20.00%	0.00%	40.00%	40.00%	1
The interaction with the system is repetitive	0.00%	20.00%	0.00%	20.00%	60.00%	1.2

Figure 2a. Negatively phrased questions of the survey



	1 Strongly Disagree–	2 Disagree–	3 Neutral–	4 Agree–	5 Strongly Agree–	Weighted Average–
I would use this system	20.00%	40.00%	40.00%	0.00%	0.00%	-0.8
The system is accurate	60.00%	0.00%	40.00%	0.00%	0.00%	-1.2
I always knew what to say to the system	20.00%	80.00%	0.00%	0.00%	0.00%	-1.2
The system is dependable	40.00%	60.00%	0.00%	0.00%	0.00%	-1.4
The system makes few errors	60.00%	0.00%	0.00%	40.00%	0.00%	-0.8
I felt calm using the system	40.00%	20.00%	20.00%	20.00%	0.00%	-0.8
The system is useful	40.00%	40.00%	20.00%	0.00%	0.00%	-1.2
The system is pleasant	0.00%	40.00%	40.00%	20.00%	0.00%	-0.2
The system is friendly	0.00%	40.00%	40.00%	20.00%	0.00%	-0.2
I enjoyed using the system	20.00%	60.00%	20.00%	0.00%	0.00%	-1
I was able to recover easily from errors	60.00%	0.00%	20.00%	0.00%	20.00%	-0.8
I felt in control of the interaction with the system	20.00%	60.00%	0.00%	0.00%	20.00%	-0.6
The system is easy to use	0.00%	60.00%	20.00%	0.00%	20.00%	-0.2
It is clear how to speak to the system	20.00%	20.00%	40.00%	0.00%	20.00%	-0.2
It is easy to learn to use the system	20.00%	20.00%	20.00%	0.00%	40.00%	0.2
The interaction with the system is efficient	20.00%	20.00%	40.00%	0.00%	20.00%	-0.2
The interaction with the system is consistent	0.00%	40.00%	40.00%	0.00%	20.00%	0

Figure 2b. Positively phrased questions of the survey

Most users had some issues with the system, many had a lot of issues, and only one user was able to complete the conversation successfully to a reasonable length. Even though User 4’s experience was good, she still lacked confidence in the system and did not feel comfortable interacting with it; this was because the answers, while technically correct, were just graceful recoveries.

Although the bot was correct approximately 80% of the time, this was found to be too low and as it can be seen in Figure 2, users were generally unhappy with the system’s performance.

The system performs better in smaller conversations and with conversational English so it failed for the users who used a more formal or less formal English. Of course, given unlimited time, this can be fixed by adding all possible synonyms and alternative forms of the utterances.

Framework Limitations

AIML is a very limited framework that does not handle natural language processing and it is just a database of pre-written questions and answers. It doesn’t even perform any similarity measures between unknown utterances and the ones in its database. In addition to these, there are also limitations we faces in its programmed capabilities, namely:

1. Not being able to initialise variables safely. As stated in the Procedure Section, there is no guarantee that the variables will be initialised if they are put in a category and there is no other place to put them.
2. The “*that”* tag has many limitations including its inability to accept punctuations and variables. This creates bugs in code every time the original answer or its variables change as the “that” utterance must be changed manually. Moreover, this significantly limits the randomization process, as if the randomised answer is used in “that”, all possible randomised forms must be coded separately.

Uncompleted Ideas

Our original idea was to automate the tedious task of typing in all the possible questions and answers; so we decided to choose a character from a TV show and build a large corpus of statements or questions and their answers from the show’s scripts. By using a similarity measure, we could have simply found the closest question or statement to the user’s input and returned the answer for that from the database. This approach also had the benefit of giving the bot a distinct personality. To undertake the task we decided on the show “The Big Bang Theory” and the character “Sheldon”. We chose a comedy to help improve the mood of the user when using the bot and help with easing frustration. “The Big Bang Theory” also had the benefit of having run for many seasons so there would be a large corpus to find questions and answers within.

We gathered and tagged the corpus and implemented the similarity measure script (based on our first assignment), however, eventually we did not proceed with this approach, because of the following reasons: firstly, it’s harder to develop and evaluate the system that doesn’t have any specific purpose and secondly, the answers usually made sense only in the context of the scenes or episodes and not otherwise.

Appendix I – Sample Human Dialogue

The following dialogue was recorded on a Google chat between the team members.

S: Hi
C: I am from a charity and I want to help you save money with your electricity bill
-save money, how?
-we have some general tips. but by asking some questions, I can get a better sense of your usage
-Oh, Ok that sounds cool
-so yeah one tip is unplug your phone after it's charged
-so should I unplug the whole charger or the phone
-just the phone is enough
-Ok
-Another tip is to check your fridge temperature. Make sure it's not too cold, especially if it's almost empty.
-Brrrr, my fridge is cold! What temp should it be set to
-between 0 and 5 degrees
-what is the correct temp for milk that is what I keep in it mostly
-let me google it for a second...between 1 and 4 should be good
-ok, thanks
-so how many people live in this household?
-2
-and how long each person spends at home on an average work day?
-14 hours
-what about weekends?
-all day
-ok so you never go out?
-seems that way. I work from homwe
-ah ok. so you should expect a big bill.
-no we jsut dont put on the lights at night we have LED lights
-cool. Those are actually more efficient.
-yes
-so just apply these tips and you will be alright
-OK thxs
-ok. Have a nice d