310: Chatbots

Dan Goldsmith

May 2018

Chatbot

Introduction

- So yesterday we talked about AI in general
  - Not very practical
  - Today lets do something a bit more interesting

Chatbot

- As we talked about yesterday Chatbots are a growing area of intelligent agents
- Two Main forms:
  - Those that imitate Humans: Turing Test
  - Those that provide a service: Assistants

Chatbot Requirements: Imitate Humans

- Natural Language processing
  - Also reply in Realistic Way
- Ability to deal with multiple conversation topics
- Ability to remember previous information.

Chatbot Requirements: Assistants

- Natural Language processing (May be limited to domain)
- Expert Domain Knowledge
- Ability to communicate with other API's

Our Task

- We are going to make a simple assistant:
  - Domain area is up to you
- Ask questions and get some sort of response.

First Version

Back to the Theory:

- Recall our definition of an Intelligent Agent:
  - A System situated in some *environment* that is capable of taking *autonomous action* to meet its design goals.
- What is the Environment: The Base Program
- What is the actions: To parse and respond to messages
- Design Goals: To fulfil some service to the customer

Design Goals:

- We are making a simple Chatbot. What are the system goals?
  - To respond to input.
  - To be able to respond to different inputs.
  - To Answer questions on a specific subject.
  - To pretend to be human?

How will we achieve these goals?

- Take Text input
- Parse It into relevant tokens
- Make a decision based on the tokens presented
- Send a response.

The Basic Framework

- So What are we going to need:
  - Main Loop
    - Receiving Input
    - Decision Making
    - Formulate Reply
    - Display Output

Basic Framework: Code

<github.com/djgoldsmith/shape/.. >

Iteration 1:

Problems with the Basic Bot

- A Really simple program for I/O
- Although having a Chatbot that just repeats a set of phrases barely meets our definition of an Intelligent Agent.

Problems with the Basic bot

- Lets Review our Goals
  - [X] To respond to input.
  - [ ] To be able to respond to different inputs.
  - [ ] To Answer questions on a specific subject.
  - [ ] To pretend to be human?

Responding to different Inputs

- So lets modify the program to respond to some different inputs
- First we want a list of topics that we can respond to, and various responses to them

What Inputs:

- Its up to you. I will use:
  - Hello
  - What is your Name
  - Who are you
  - Where are you from
  - What is the weather like
  - What is your favourite place
  - Who is your favourite actor

What Outputs:

- For the moment we will just leave it as a single output for each question.

Code

<chatter bot 2>

Chatterbot2: Issues

- Doesn't like punctuation
  - We can fix this by stripping punctuation
- Can only respond to "Known" questions
  - Provide a default answer
- Has a limited range of responses
  - Have more results available

Chatterbot2: Issues

- Lets' add a default answer together

```
 //Function to Calculate response.
static String findMatch(String str) {
String result = "Im sorry, I don't understand you"; //Default Response
for(int i = 0; i < KnowledgeBase.length; ++i) {
    if(KnowledgeBase[i][0].equalsIgnoreCase(str)) {
    result = KnowledgeBase[i][1];
    break;
    }
}
return result;
}
```

Chatterbot2 Issues:

- Your turn to fix the rest of the problems:
- Hint: Punctuation (replaceAll function)
- Hint Limited Responses:
  - Have a range of answers and pick one at random

What outputs: Hello

- And some varied outputs to the questions:
- Hello
  - Hello
  - Hi
  - Hello, How are you

What Outputs: What is your name?

- What is your name
  - A Chatbot
  - Name
  - Name. What is your name?

Chatbot 2a:

CODE

Chatbot 3:

- Things are getting a little better, but we still have issues:
  - Our responses are sill random.
    - Remembering what has been said before would be good
  - We can only respond to Exact Phrases

Chatbot3 Improvements:

- 1) Let turn it into a Class, so we get some memory
- 2) We can use this to remember what has been said before
- 3) Lets also look at parsing the user input
  - Keyword Matching
  - Fuzzy Matching

Chatbot 3a: Remembering what has been said before

- We will keep it simple for the moment:
  - Lets just store the previous input and point out if the user is repeating themselves

```
public static boolean checkRepeat(){
//Check against last input
if (sInput.equals(lastInput)){
    return true;
}
//Update and return
lastInput = sInput;
return false;
}
```

Chatbot 3a: Repetition

```
try{
    getInput();
    cleanInput();
    System.out.println("\tDEBUG: "+sInput);
    if(sInput.equalsIgnoreCase("quit")) {
    System.out.println("BOT: It was nice talking,  goodbye");
    return false;
    }
    else {
    //Check for duplicates
    if(checkRepeat()){
        System.out.println("You are repeating yourself!");
    }
    else{
        sResponse = findMatch(sInput);
        System.out.println("BOT: " + sResponse);
    }
    }
```

Chatbot 3a: Exercise: Improving Repetition

- We only remember the last input
  - Perhaps remember the last N inputs
  - Only store if input was valid?

Parsing Input Keyword Matching:

- A very simple approach.
  - Instead of sentences we match keywords.
  - IE **“Name”** rather than **“What is your name”**
  - There is a problem with this:
    - “What is your name?”
    - “My name is Dan”

Parsing Input: Fuzzy Matching

- A very basic form of Language processing
- Rather than match the whole sentence, we look for keywords
  - Use some metric to determine if this is the question being asked.

What Metric:

- There are several metrics for assessing string similarity
  - Levenstein Distance (how many letters need to be changed)
  - Cosine Distance
- These can be difficult to implement, and give us some complex design choices
  - What is the threshold?

Token based metric:

- We are going to make use of Tokens.
  - Break each string into a series of keywords
  - Count matches For each element in the knowlegebase
    - The result with the most matches is the “winner”

Token based Metric (Chatbot 4)

```
static String findMatch(String str) {
result = "Im sorry, I don't understand you"; //Default Response
//Split input into an array based on spaces
String[] inArray = str.split(" ");
int bestIndex = 0; //The Best Match
int bestCount = 0;
for(int i = 0; i < KnowledgeBase.length; ++i) {
    //Tokenise the input
    String[] checkArray = KnowledgeBase[i][0].split(" ");
    int count = 0;
    //Count the matches
    for (int outer = 0; outer < checkArray.length; outer++){ //Outer Loop (
    for (int inner = 0; inner < inArray.length; inner++){ //Inner Loop (inp
        if (checkArray[outer].equalsIgnoreCase(inArray[inner])){
        count += 1;
        }
    }
    }
    //Now we work out the best answer
    if (count > bestCount){
    bestCount = count;
    bestIndex = i;
```

Issues with our current Token based Metric

- What if several strings have the same matches:
  - What is your Name
  - What is the weather like
  - What is your favourite place
- Reject if one of more strings has the “best” metric?
- Over to you.....

Other Issues with the token based metric:

- Computational Complexity. (total number of words for input) * (total number of options for all knowledge)
- Inappropriate for large datasets

Moving on from here:

- We could get the system to accept and remember inputs
  - Would require some metadata in the knowledge base
- We could try to improve the efficiency of the knowledge base
  - Decision trees?
  - Finite State Automita (like a compiler)

AIML

A Better approach

- We are going to make use of another markup language to represent our bots knowledge
- This gives us more flexibility in the inputs and outputs

AIML

- Artificial Intelligence Markup Language
- Designed 1995 for the ALICE bot
- https://www.pandorabots.com/pandora/talk?botid=a847934aae3456

AIML

- XML file defining the inputs and responses
- Is parsed by the Chatbot program to define converstaion
  - Similar to our List in the example above
- I haven't had time to put a chat bot together so we will use an existing one

Tutorial

Either: - https://playground.pandorabots.com/en/tutorial/ - https://www.tutorialspoint.com/aiml/aiml_quick_guide.htm

Tutorial: Getting Started

- Download the code from
- https://code.google.com/archive/p/program-ab/
- Unzip etc.

Tutorial: First Steps

```
$run.bat
--> Bot super 4846 completed 0 deleted 0 unfinished
28534 nodes 22294 singletons 4846 leaves 0 shortcuts 1394 r
Human: Hello
Robot: Hi how are you?
Human:
```

Tutorial Writing our own input