COMS 493

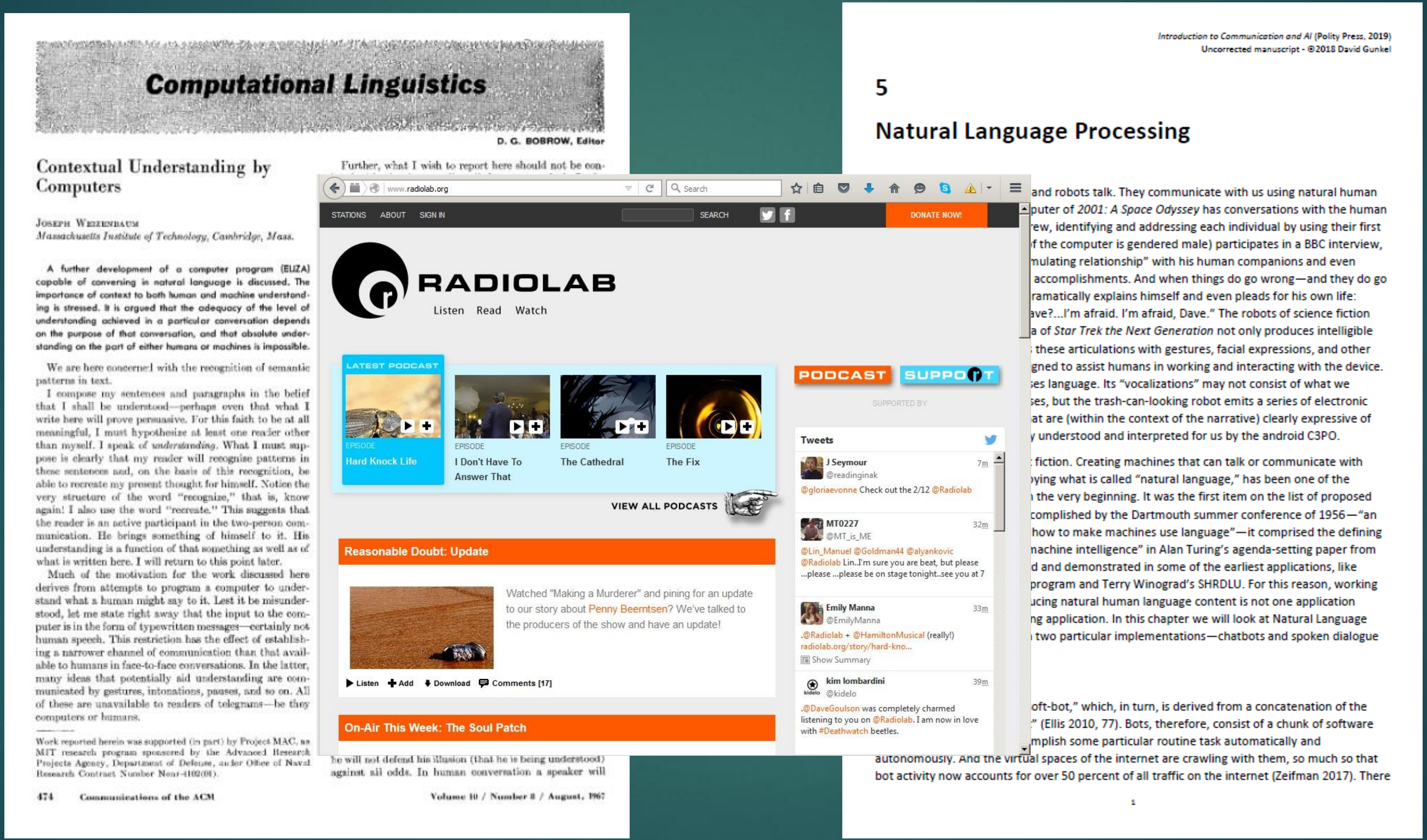
AI, ROBOTS & COMMUNICATION

Agenda

- Review
- Chatterbot Competition
- Preview

Review

Natural Language Processing (NLP)



Computational Linguistics

D. G. BOBROW, Editor

Contextual Understanding by Computers

Joseph Weizenbaum
Massachusetts Institute of Technology, Cambridge, Mass.

A further development of a computer program (ELIZA) capable of conversing in natural language is discussed. The importance of context to both human and machine understanding is stressed. It is argued that the adequacy of the level of understanding achieved in a particular conversation depends on the purpose of that conversation, and that absolute understanding on the part of either humans or machines is impossible.

We are here concerned with the recognition of semantic patterns in text.

I compose my sentences and paragraphs in the belief that I shall be understood—perhaps even that what I write here will prove persuasive. For this faith to be at all meaningful, I must hypothesize at least one reader other than myself. I speak of *understanding*. What I must suppose is clearly that my reader will recognise patterns in these sentences and, on the basis of this recognition, be able to recreate my present thought for himself. Notice the very structure of the word "recognize," that is, know again! I also use the word "recreate." This suggests that the reader is an active participant in the two-person communication. He brings something of himself to it. His understanding is a function of that something as well as of what is written here. I will return to this point later.

Much of the motivation for the work discussed here derives from attempts to program a computer to understand what a human might say to it. Lest it be misunderstood, let me state right away that the input to the computer is in the form of typewritten messages—certainly not human speech. This restriction has the effect of establishing a narrower channel of communication than that available to humans in face-to-face conversations. In the latter, many ideas that potentially aid understanding are communicated by gestures, intonations, pauses, and so on. All of these are unavailable to readers of telegrams—be they computers or humans.

Further, what I wish to report here should not be con-

he will not defend his illusion (that he is being understood) against all odds. In human conversation a speaker will

Work reported herein was supported (in part) by Project MAC, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01).

Reasonable Doubt: Update

Watched "Making a Murderer" and pining for an update to our story about Penny Beerntsen? We've talked to the producers of the show and have an update!

On-Air This Week: The Soul Patch

Tweets

J Seymour @readinginak 7m
@gloriaevonne Check out the 2/12 @Radiolab

MT0227 @MT_is_ME 32m
@Lin_Manuel @Goldman44 @alyankovic @Radiolab Lin..I'm sure you are beat, but please ...please ...please be on stage tonight..see you at 7

Emily Manna @EmilyManna 33m
.@Radiolab + @HamiltonMusical (really!) radiolab.org/story/hard-kno...
Show Summary

kim lombardini @kidelo 39m
.@DaveGoulson was completely charmed listening to you on @Radiolab. I am now in love with #Deathwatch beetles.

5

Natural Language Processing

and robots talk. They communicate with us using natural human
puter of *2001: A Space Odyssey* has conversations with the human
rew, identifying and addressing each individual by using their first
f the computer is gendered male) participates in a BBC interview,
mulating relationship" with his human companions and even
accomplishments. And when things do go wrong—and they do go
ramatically explains himself and even pleads for his own life:
ave?...I'm afraid. I'm afraid, Dave." The robots of science fiction
a of *Star Trek the Next Generation* not only produces intelligible
these articulations with gestures, facial expressions, and other
gned to assist humans in working and interacting with the device.
ses language. Its "vocalizations" may not consist of what we
ses, but the trash-can-looking robot emits a series of electronic
at are (within the context of the narrative) clearly expressive of
y understood and interpreted for us by the android C3PO.

fiction. Creating machines that can talk or communicate with
ying what is called "natural language," has been one of the
the very beginning. It was the first item on the list of proposed
complished by the Dartmouth summer conference of 1956—"an
how to make machines use language"—it comprised the defining
nachine intelligence" in Alan Turing's agenda-setting paper from
d and demonstrated in some of the earliest applications, like
program and Terry Winograd's SHRDLU. For this reason, working
ucing natural human language content is not one application
ng application. In this chapter we will look at Natural Language
two particular implementations—chatbots and spoken dialogue

oft-bot," which, in turn, is derived from a concatenation of the
" (Ellis 2010, 77). Bots, therefore, consist of a chunk of software
mplish some particular routine task automatically and
autonomously. And the virtual spaces of the internet are crawling with them, so much so that
bot activity now accounts for over 50 percent of all traffic on the internet (Zeifman 2017). There

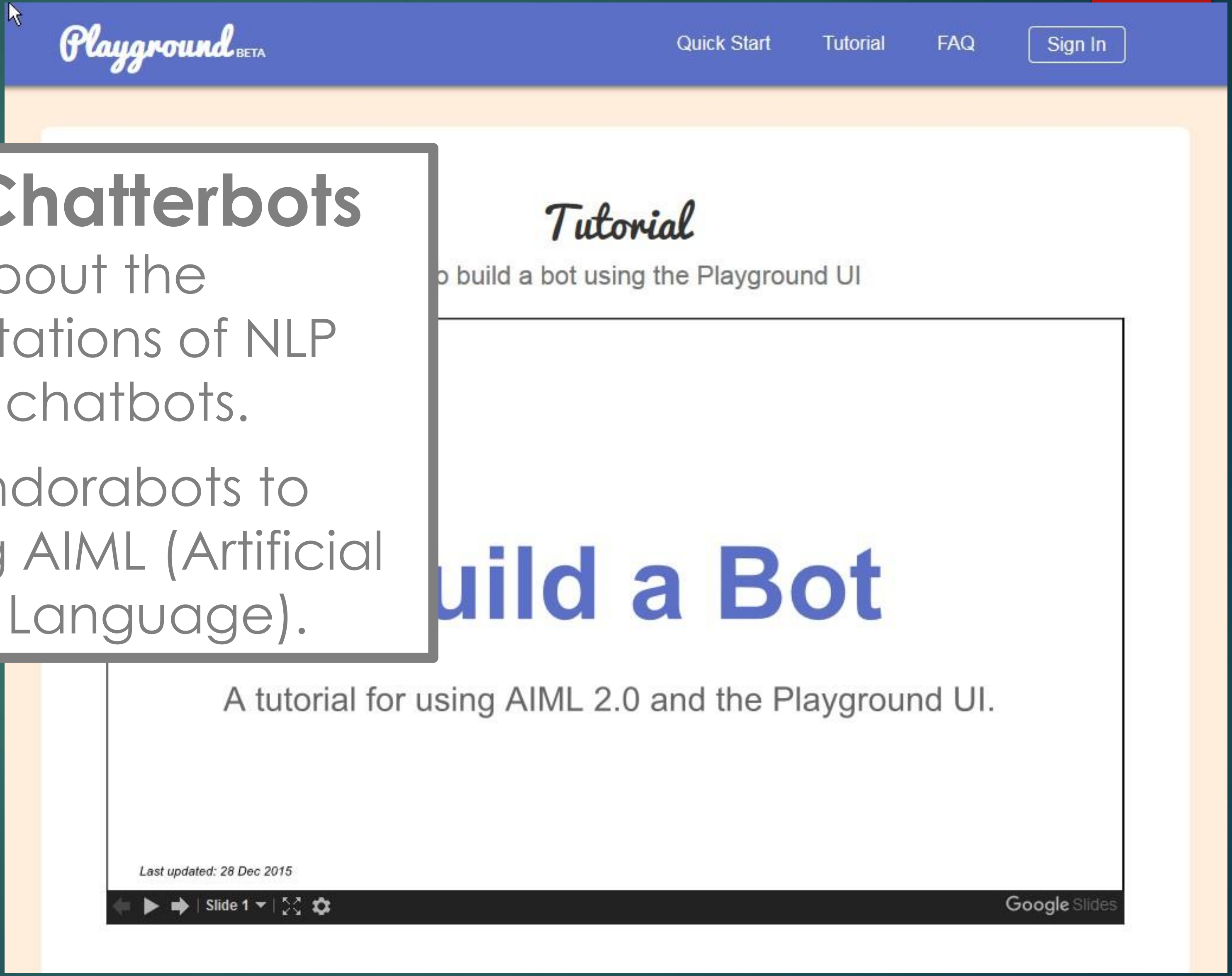
Review

https://home.pandorabots.com

Experiment with Chatterbots

Objective – Learn about the capabilities and limitations of NLP by building our own chatbots.

Procedure – Use Pandorabots to program a bot using AIML (Artificial Intelligence Markup Language).



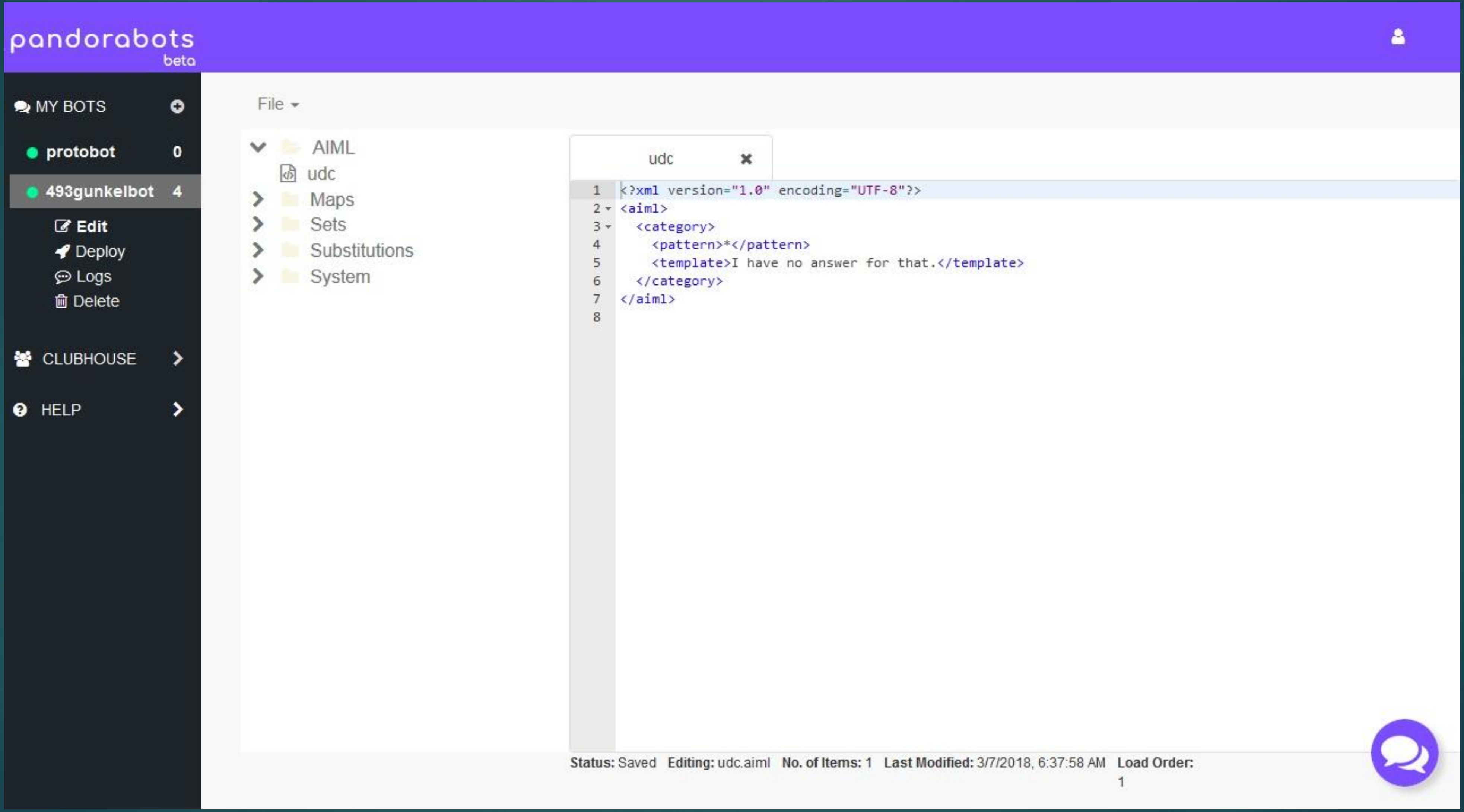
o build a bot using the Playground UI

uild a Bot

A tutorial for using AIML 2.0 and the Playground UI.

Last updated: 28 Dec 2015

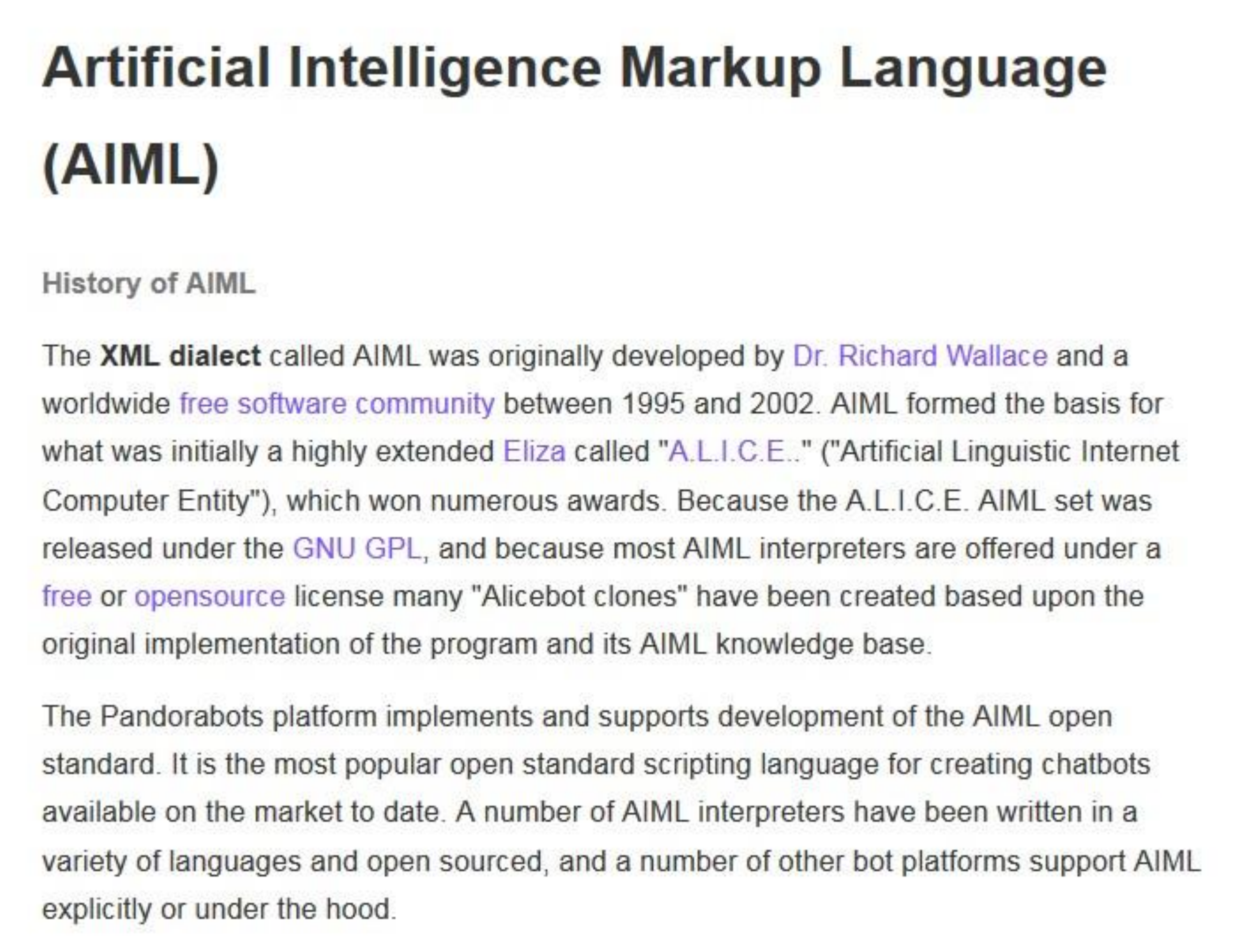
Review - AIML



```
<?xml version="1.0" encoding="UTF-8"?>
<aiml>
  <category>
    <pattern>*</pattern>
    <template>I have no answer for that.</template>
  </category>
</aiml>
```

Status: Saved **Editing:** udc.aiml **No. of Items:** 1 **Last Modified:** 3/7/2018, 6:37:58 AM **Load Order:** 1

Review - AIML



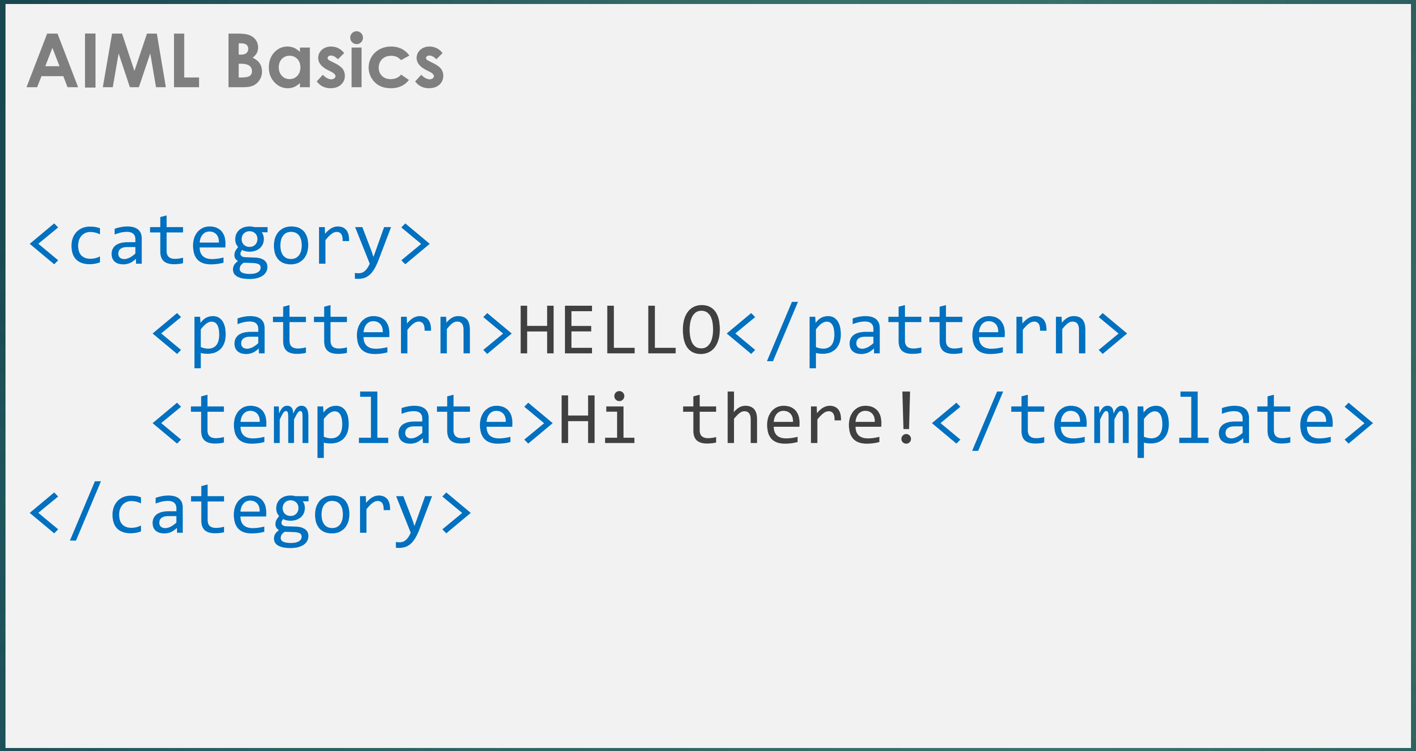
Artificial Intelligence Markup Language (AIML)

History of AIML

The **XML dialect** called AIML was originally developed by Dr. Richard Wallace and a worldwide free software community between 1995 and 2002. AIML formed the basis for what was initially a highly extended Eliza called "A.L.I.C.E.." ("Artificial Linguistic Internet Computer Entity"), which won numerous awards. Because the A.L.I.C.E. AIML set was released under the GNU GPL, and because most AIML interpreters are offered under a free or opensource license many "Alicebot clones" have been created based upon the original implementation of the program and its AIML knowledge base.

The Pandorabots platform implements and supports development of the AIML open standard. It is the most popular open standard scripting language for creating chatbots available on the market to date. A number of AIML interpreters have been written in a variety of languages and open sourced, and a number of other bot platforms support AIML explicitly or under the hood.

Review - AIML



AIML Basics

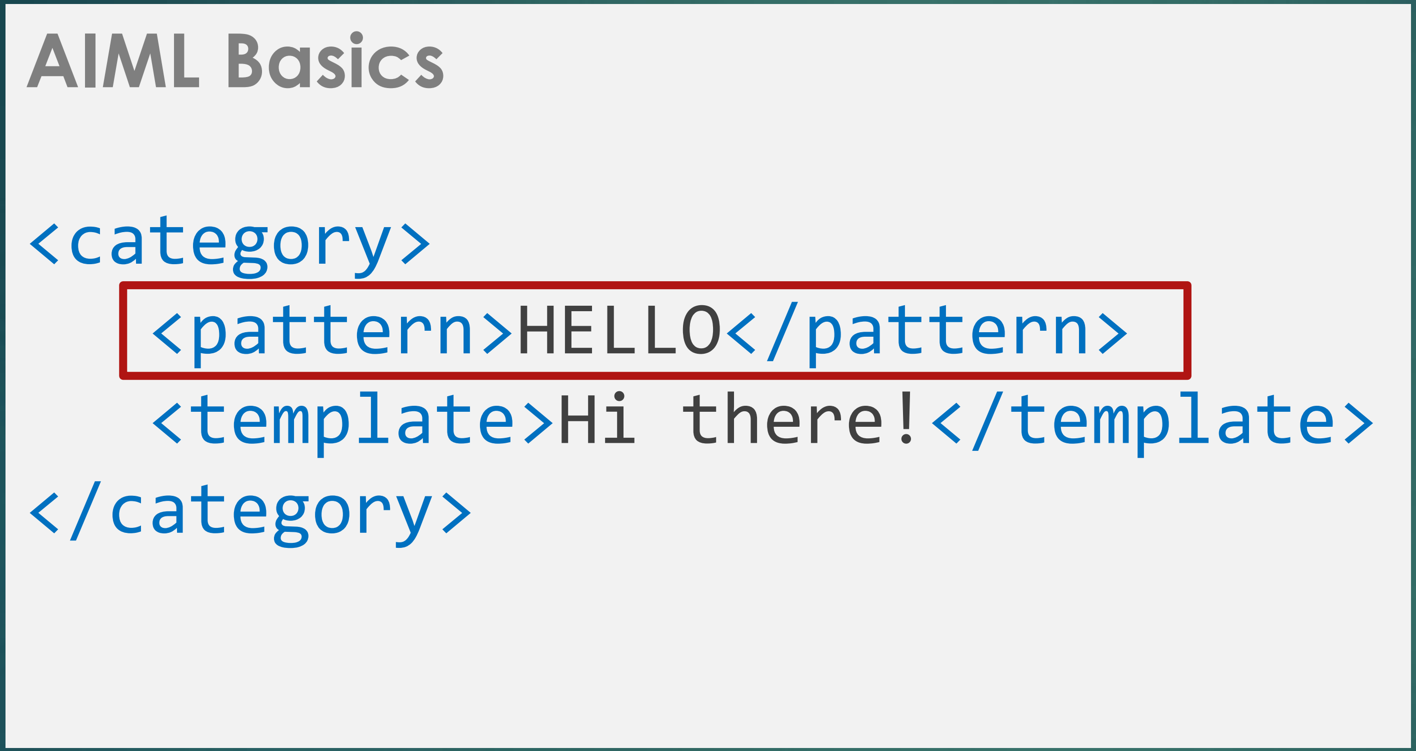
```
<category>
   <pattern>HELLO</pattern>
   <template>Hi there!</template>
</category>
```

The basic unit of knowledge in AIML is called a **category**.

Each category consists of an input question, an output answer, and an optional context.

The question, or stimulus, is called the **pattern**. The answer, or response, is called the **template**.

Review - AIML



AIML Basics

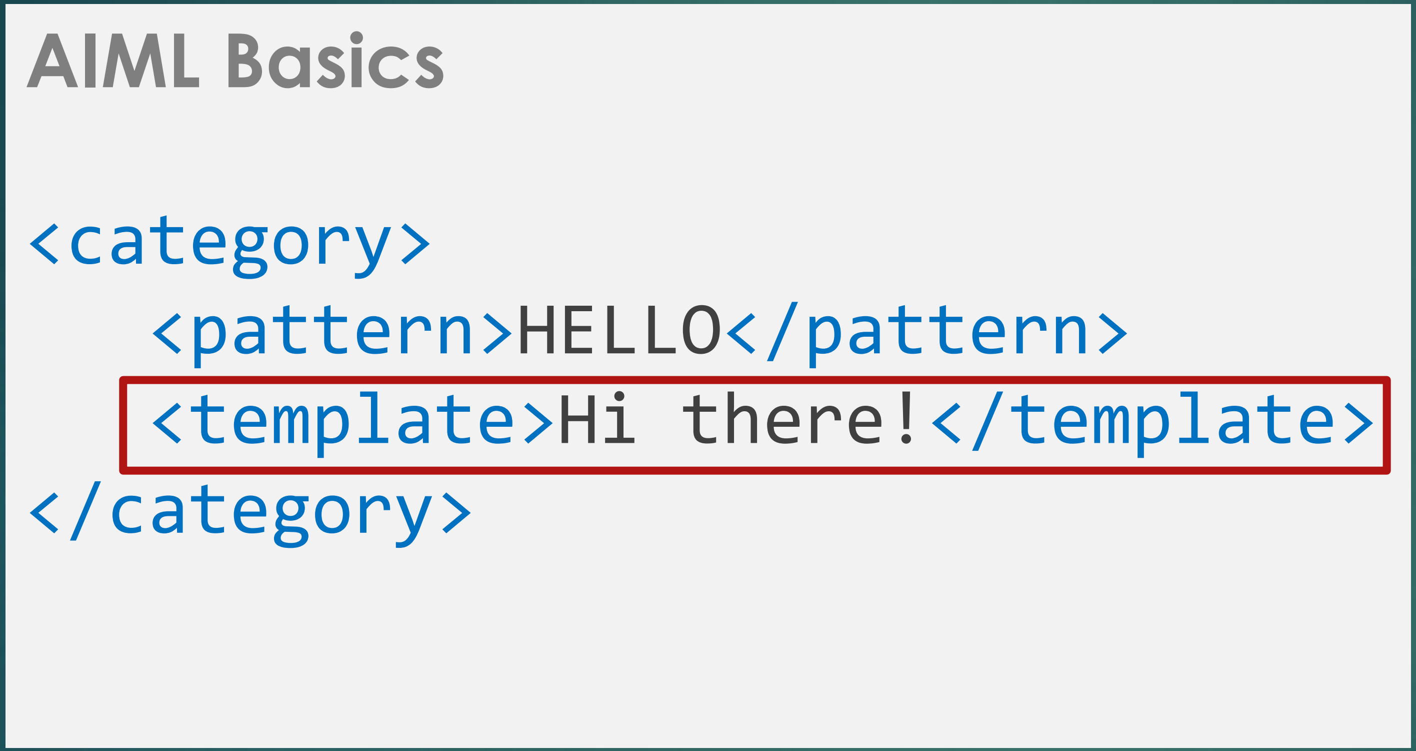
```
<category>
   <pattern>HELLO</pattern>
   <template>Hi there!</template>
</category>
```

The AIML **pattern** language is simple, consisting only of words, spaces, and wildcard symbols like *.

The words may consist of letters and numerals, but no other characters (no punctuation).

The pattern language is not case sensitive.

Review - AIML



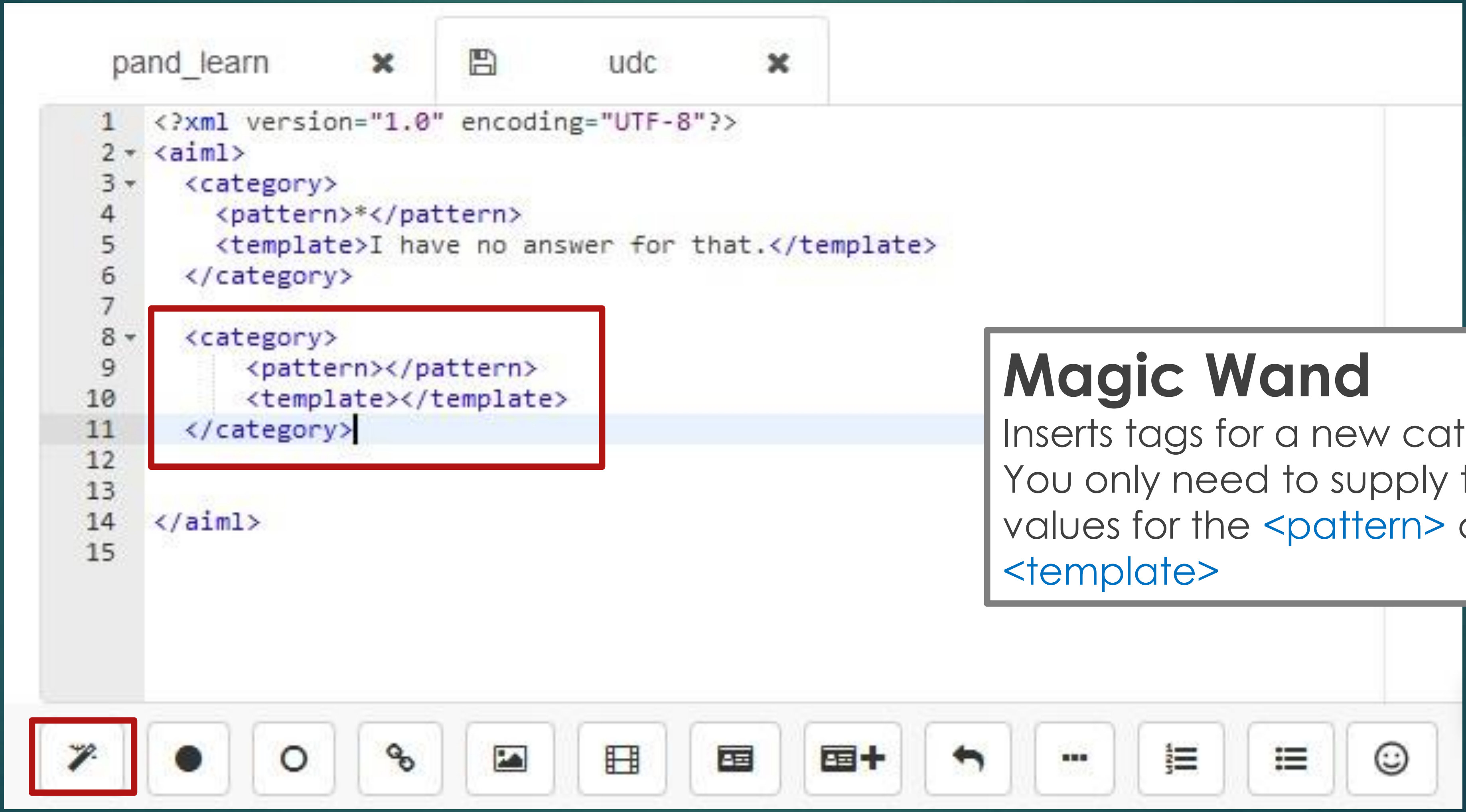
AIML Basics

```
<category>
   <pattern>HELLO</pattern>
   <template>Hi there!</template>
</category>
```

Template defines the bot's response to the matched pattern.

Case does matter in the template! Your bot’s response will be returned to the user exactly as written between the template tags.

Review - AIML



```
<?xml version="1.0" encoding="UTF-8"?>
<aiml>
  <category>
    <pattern>*</pattern>
    <template>I have no answer for that.</template>
  </category>

  <category>
      <pattern></pattern>
      <template></template>
  </category>


</aiml>
```

Magic Wand

Inserts tags for a new category. You only need to supply the text values for the <pattern> and the <template>

Review - AIML

```
<category>
 <pattern>*</pattern>
 <template>
  <random>
    <li>What was that?</li>
    <li>I don’t understand</li>
    <li>Can you say that more clearly?</li>
  </random>
 </template>
</category>
```

Randomized Responses

Review - AIML

Wild Cards

```
<category>
   <pattern>MY NAME IS *</pattern>
   <template>         </template>
</category>
```

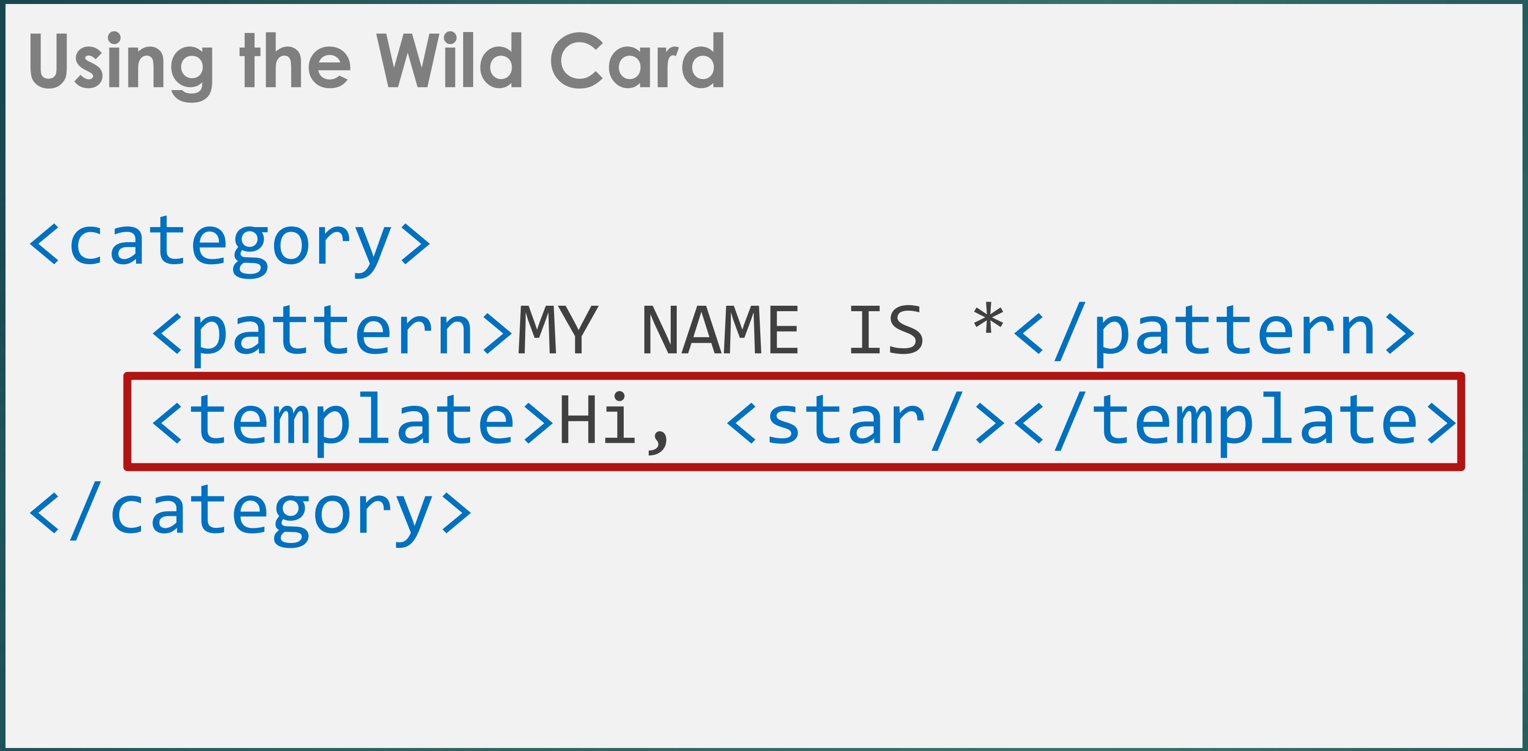
The * captures 1 or more words in the user input.

This pattern would match all of the following inputs:

Hello there!
Hello Kathy.
Hello my good friend.

The ^ captures 0 or more words in the user input. So it would capture all of the above plus "My name is"

Review - AIML



Using the Wild Card

```
<category>
   <pattern>MY NAME IS *</pattern>
   <template>Hi, <star/></template>
</category>
```

Echo the word(s) captured by the * wild card with the **<star/>** tag.

This tag inserts what ever word or words were captured from the user input with the *

USER: My Name is *Zenon*.
BOT: Hi, *Zenon*.

Review - AIML

Variables (Predicates)

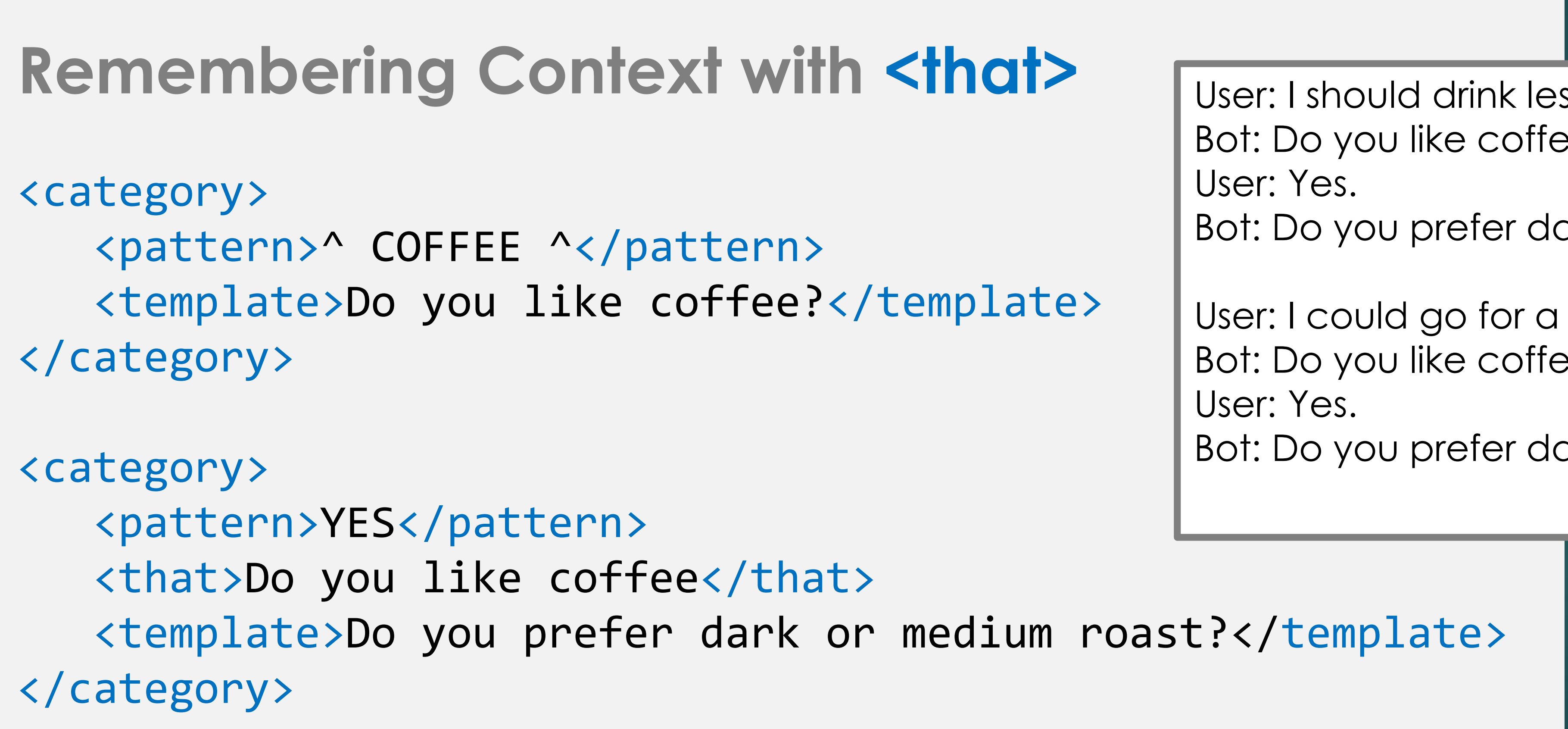
```
<category>
   <pattern>MY NAME IS *</pattern>
   <template>Hi, <set name=“name”><star/></set></template>
</category>
```

Set a Variable

```
<category>
   <pattern>WHAT IS MY NAME</pattern>
   <template>Your name is <get name=“name”/></template>
</category>
```

Recall a Variable

Review - AIML



Remembering Context with <that>

```
<category>
   <pattern>^ COFFEE ^</pattern>
   <template>Do you like coffee?</template>
</category>

<category>
   <pattern>YES</pattern>
   <that>Do you like coffee</that>
   <template>Do you prefer dark or medium roast?</template>
</category>
```

User: I should drink less coffee.
Bot: Do you like coffee?
User: Yes.
Bot: Do you prefer dark or medium roast?

User: I could go for a coffee or espresso.
Bot: Do you like coffee?
User: Yes.
Bot: Do you prefer dark or medium roast?

Review - AIML

Conditional Statement (version 1)

```
<category>
   <pattern>TODAY IS *</pattern>
   <template>
     <think><set name=“today”> <star/></set></think>
     <condition name=“today”>
       <li value=“Monday”>The start of a new week.</li>
       <li value=“Tuesday”>Tuesday already?</li>
       <li value=”Wednesday>Halfway to the weekend!</li>
         . . .
     </condition>
   </template>
</category>
```

Using the **<condition>** tag, a bot can respond differently to the same input depending on the value of a variable.

The concept is the same as the **IF / ELSE** statements we were using with Javascript

If the User writes “Today is Tuesday” The bot responds with “Tuesday already.”
If the User writes “Today is Monday” The bot responds with “The start of a new week”

Review - AIML

Conditional Statement (version 2)

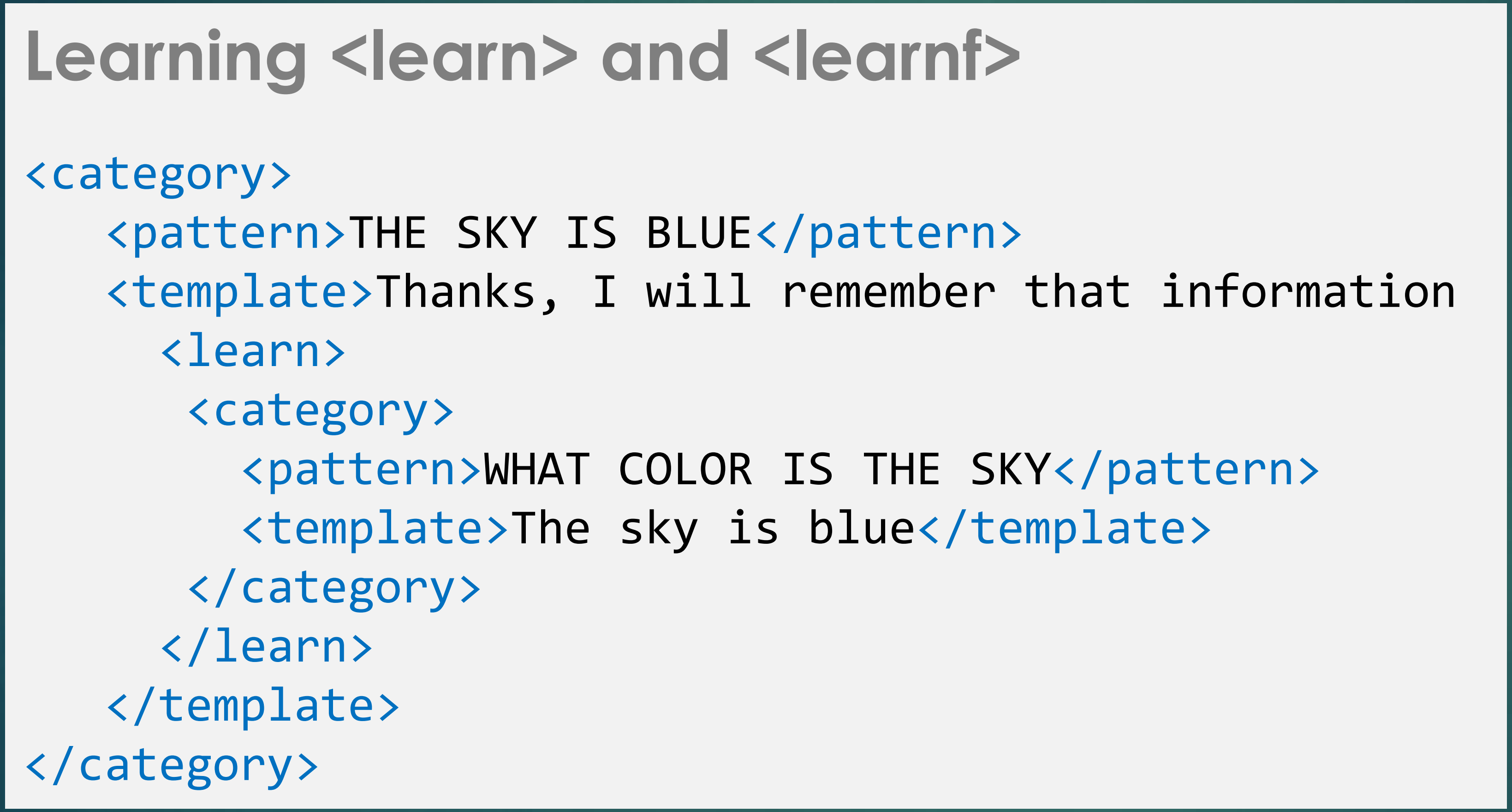
```
<category>
   <pattern>WHAT IS MY NAME</pattern>
   <template>
     <condition name=“name”>
       <li value=“*”>Your name is <get name=“name”></li>
       <li>You haven’t told me your name yet!</li>
     </condition>
   </template>
</category>
```

Another way to use the conditional statement is to check the status of a variable to see whether it has been set or not.

If it has been set, then there is one response.

If it has not been set, then there is a different response.

Review - AIML



Learning <learn> and <learnf>

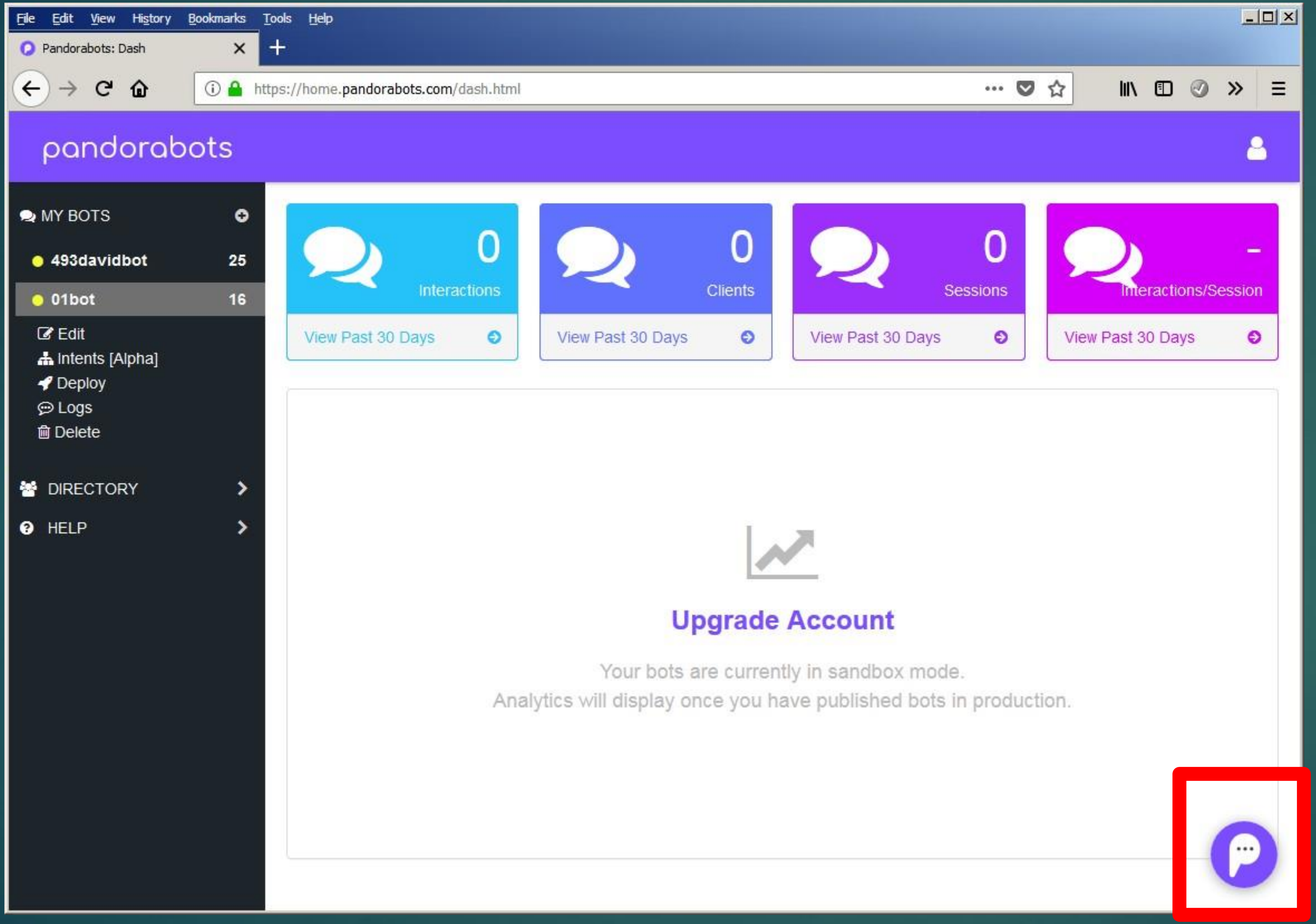
```
<category>
   <pattern>THE SKY IS BLUE</pattern>
   <template>Thanks, I will remember that information
     <learn>
      <category>
        <pattern>WHAT COLOR IS THE SKY</pattern>
        <template>The sky is blue</template>
      </category>
     </learn>
   </template>
</category>
```

With <learn> users can generate new categories from a conversation.

Categories learned using <learn> will only be accessible during that conversation, and will eventually be cleared.

Categories learned using <learnf> will be written to a new AIML file and can be accessed by anyone talking with your bot.

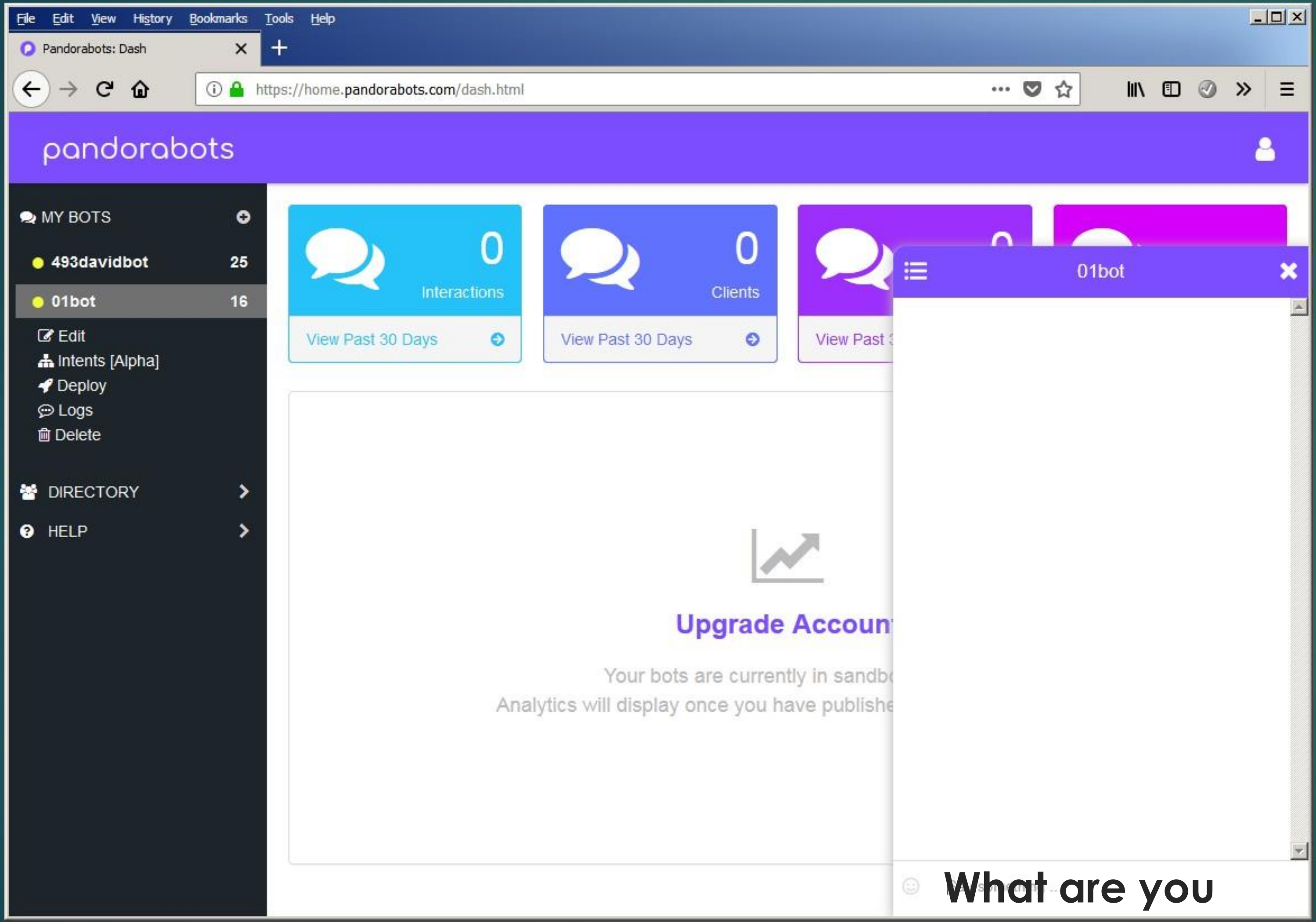
Review - AIML



Bot Training via Chat Widget

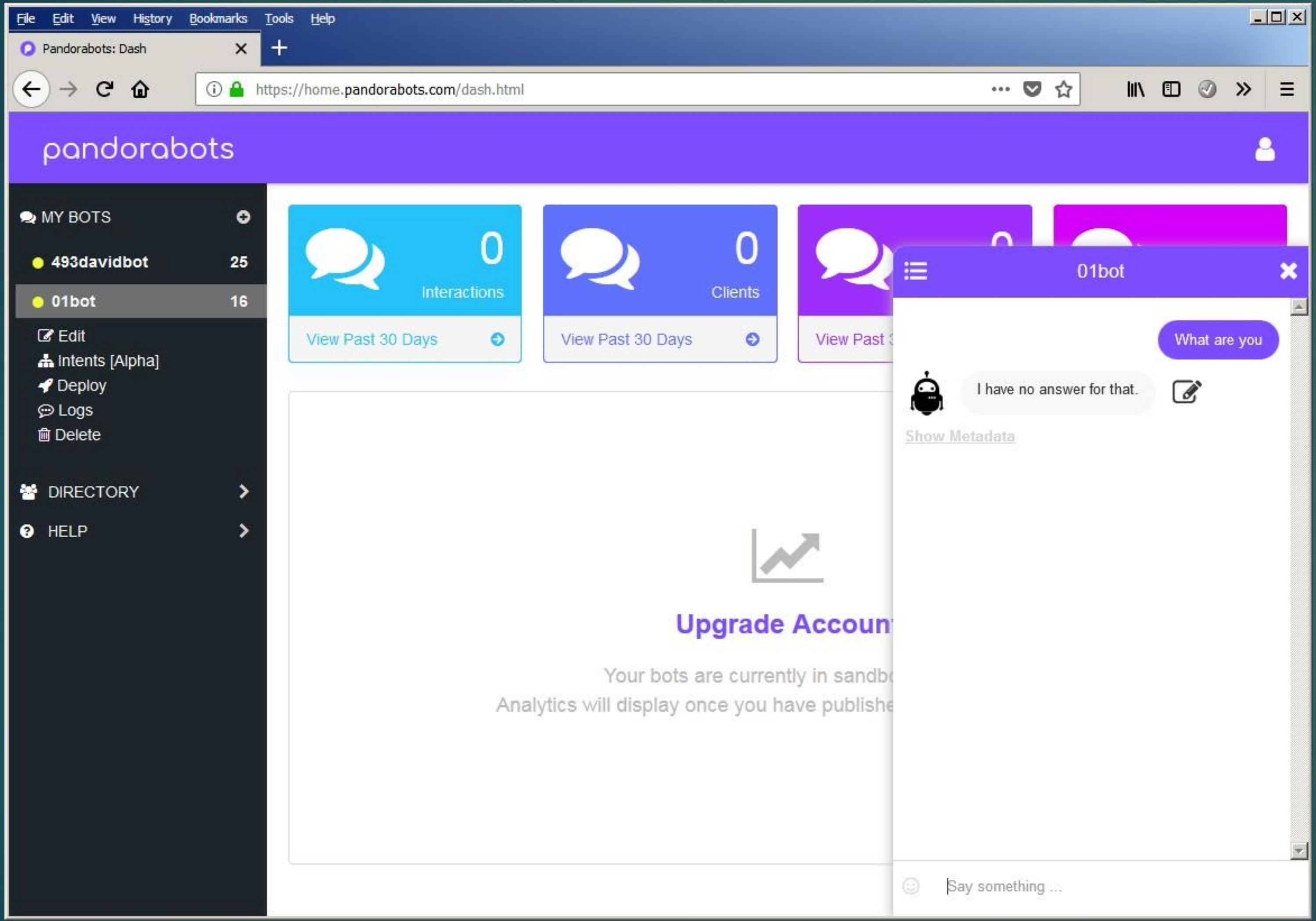
Clicking on this icon will allow you to chat with your bot and make modifications to the bot’s behavior without needing to write the code in AIML

Review - AIML



What are you

Review - AIML

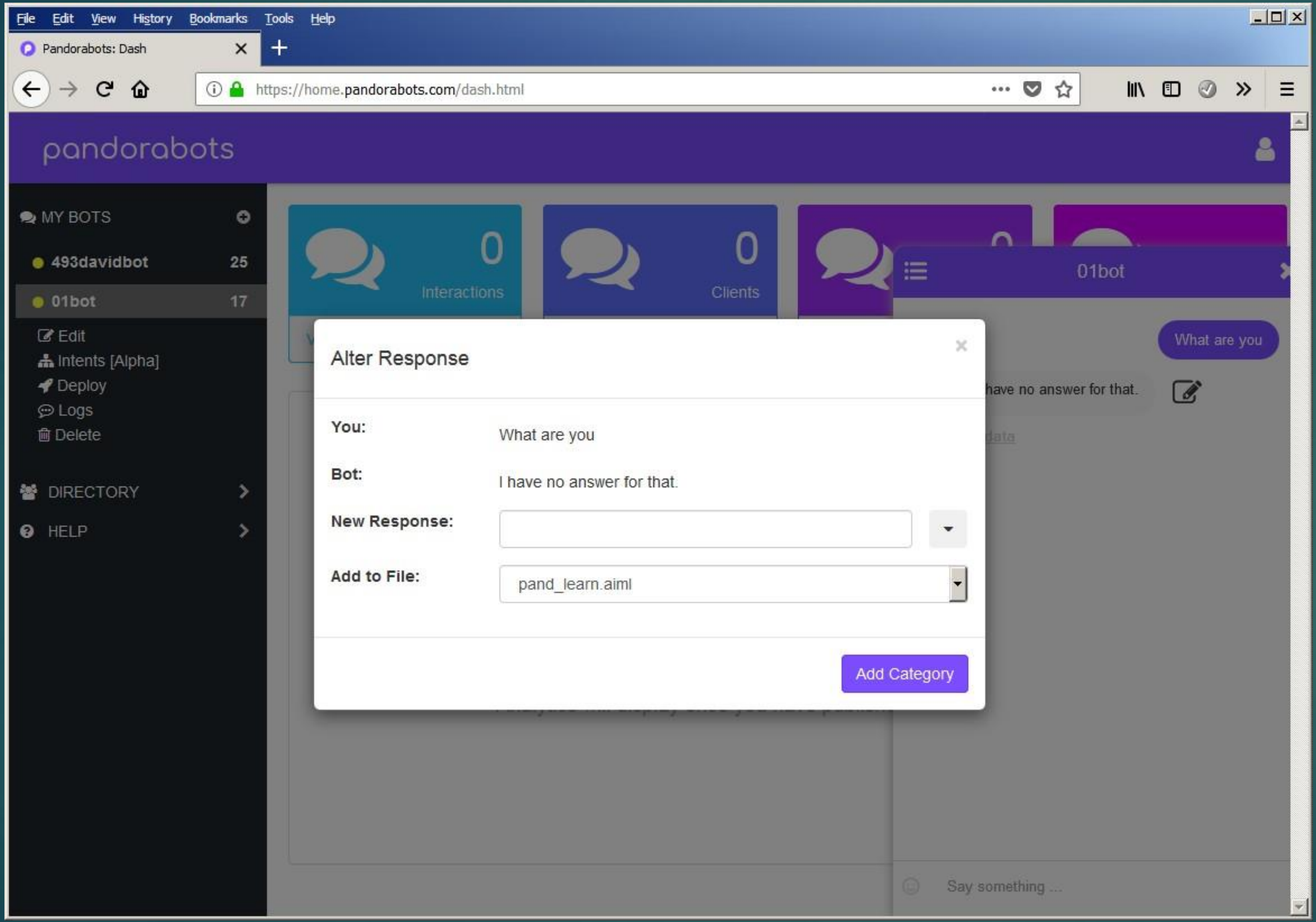


Your bots are currently in sandb
Analytics will display once you have publishe

What are you

I have no answer for that.

Review - AIML



Alter Response

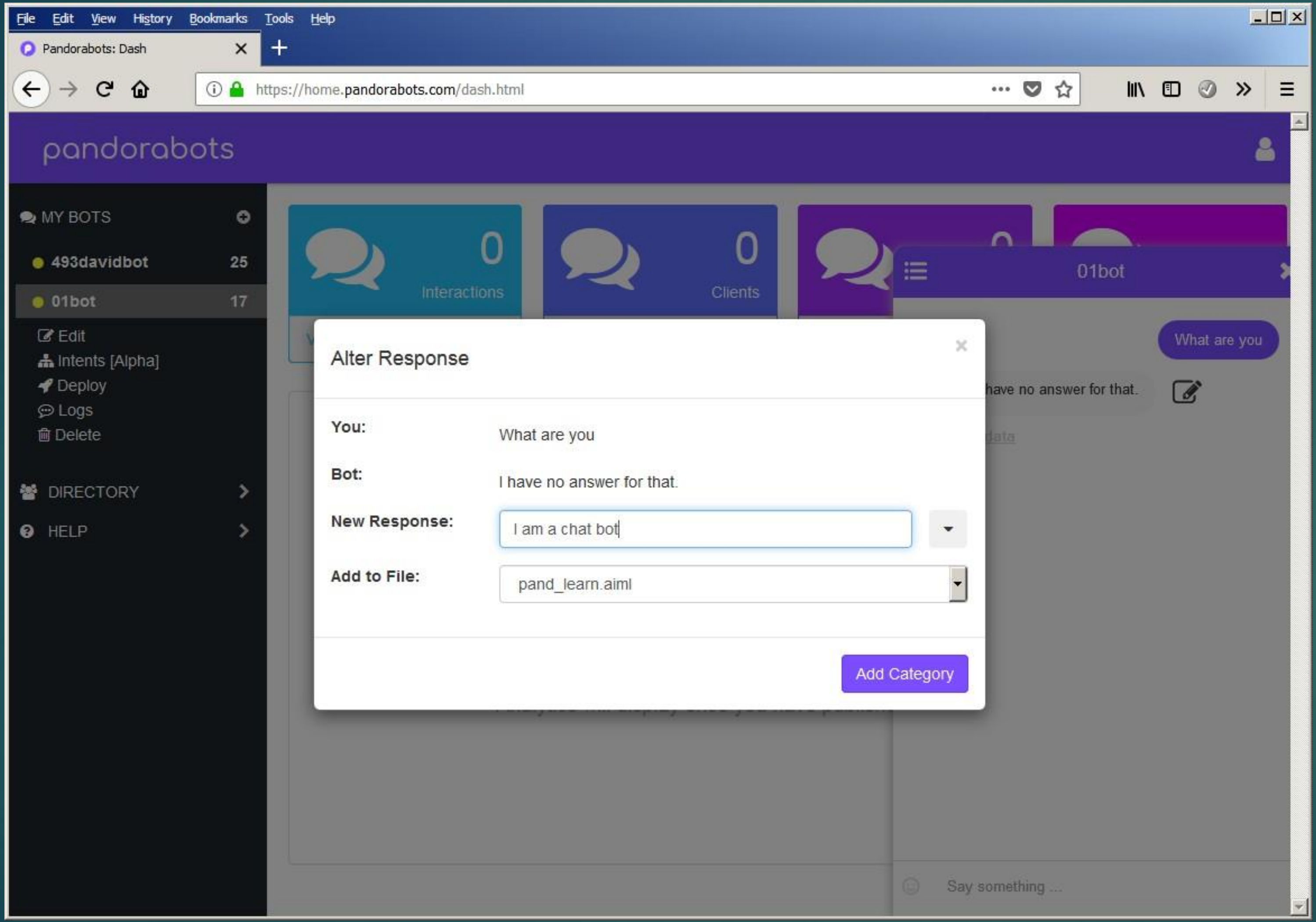
You: What are you

Bot: I have no answer for that.

New Response:

Add to File: pand_learn.aiml

Review - AIML



Alter Response

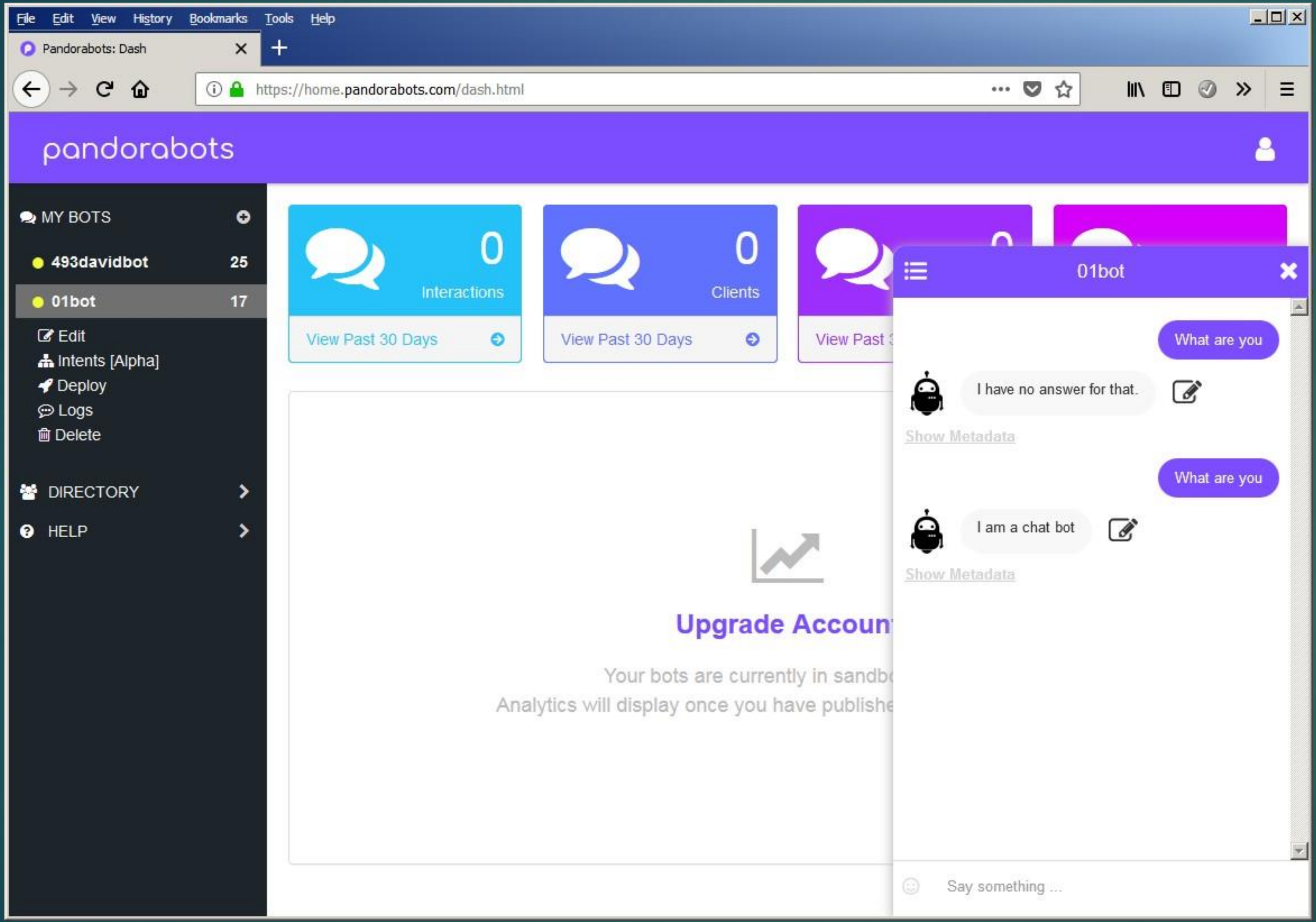
You: What are you

Bot: I have no answer for that.

New Response: I am a chat bot

Add to File: pand_learn.aiml

Review - AIML

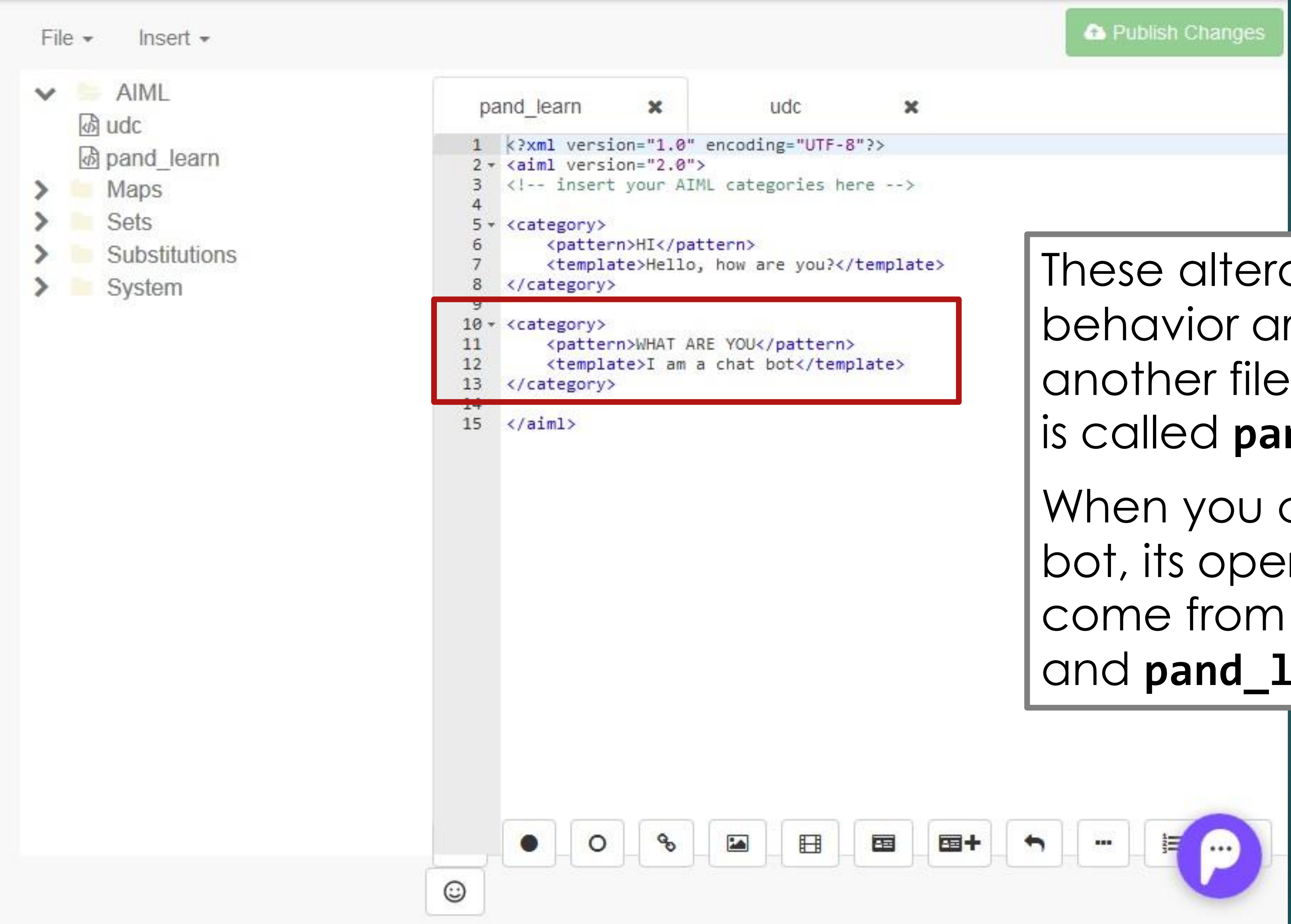


What are you

I have no answer for that.

What are you

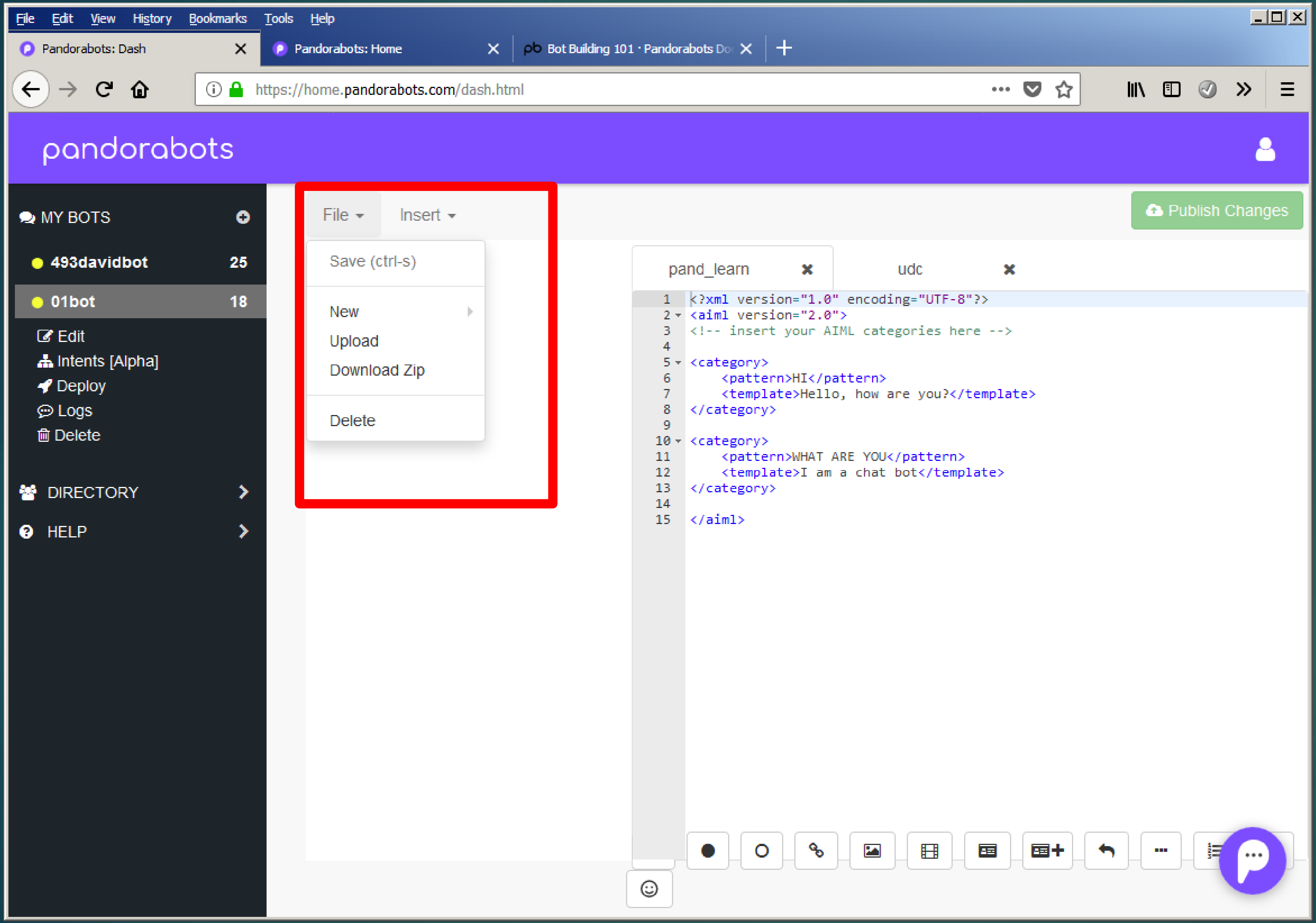
I am a chat bot



```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="2.0">
<!-- insert your AIML categories here -->

<category>
    <pattern>HI</pattern>
    <template>Hello, how are you?</template>
</category>

<category>
    <pattern>WHAT ARE YOU</pattern>
    <template>I am a chat bot</template>
</category>

</aiml>
```

These alterations to bot behavior are saved in another file under **AIML**. It is called **pand_learn**

When you chat with your bot, its operations will come from both the **udc** and **pand_learn** files



```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="2.0">
<!-- insert your AIML categories here -->

<category>
    <pattern>HI</pattern>
    <template>Hello, how are you?</template>
</category>

<category>
    <pattern>WHAT ARE YOU</pattern>
    <template>I am a chat bot</template>
</category>

</aiml>
```

Save all changes

And **save** often so you do not lose changes

Deploy your bot to the **Directory**

Test your bot to make sure it works

Competition

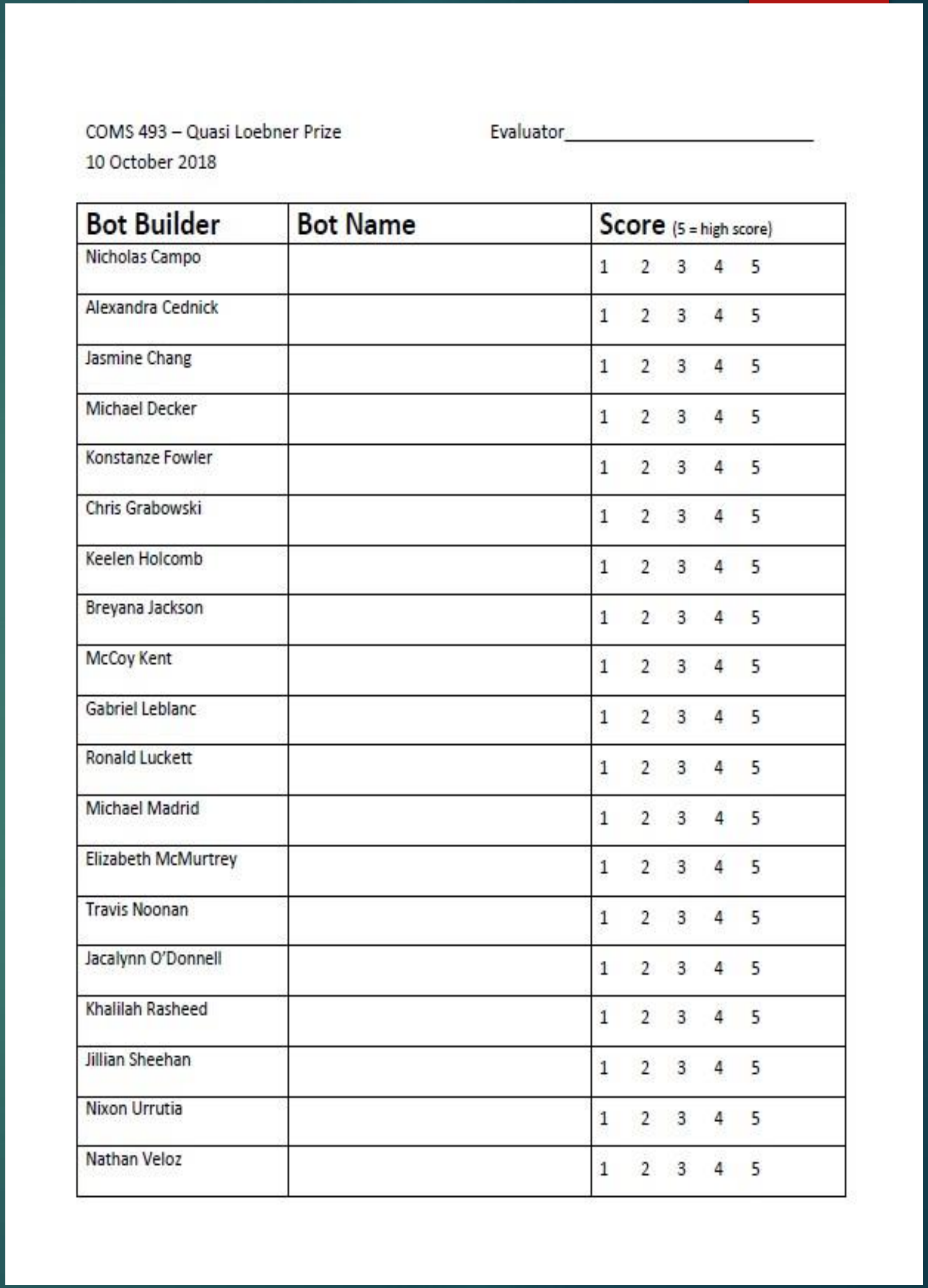
Quasi Loebner Prize

Procedure/Rules

1. Edit your bot; deploy it to the Directory
2. Interact with everyone's bot
3. Complete Evaluation Form

Timeframe

6:00-7:15 – Bot editing and deployment
7:15-8:15 – Chat with the bots
8:15-8:30 – Calculate results & Awards



COMS 493 – Quasi Loebner Prize
10 October 2018

Evaluator_______________________

Bot Builder	Bot Name	Score (5 = high score)
Nicholas Campo		1 2 3 4 5
Alexandra Cednick		1 2 3 4 5
Jasmine Chang		1 2 3 4 5
Michael Decker		1 2 3 4 5
Konstanze Fowler		1 2 3 4 5
Chris Grabowski		1 2 3 4 5
Keelen Holcomb		1 2 3 4 5
Breyana Jackson		1 2 3 4 5
McCoy Kent		1 2 3 4 5
Gabriel Leblanc		1 2 3 4 5
Ronald Luckett		1 2 3 4 5
Michael Madrid		1 2 3 4 5
Elizabeth McMurtrey		1 2 3 4 5
Travis Noonan		1 2 3 4 5
Jacalynn O'Donnell		1 2 3 4 5
Khalilah Rasheed		1 2 3 4 5
Jillian Sheehan		1 2 3 4 5
Nixon Urrutia		1 2 3 4 5
Nathan Veloz		1 2 3 4 5