THE UNIVERSITY *of* EDINBURGH

The Learning, Teaching and Web Services Division (LTW)

Teacherbot Project

Pandorabot Playground Documentation

Author: Kathrin Haag

May 23, 2016

Contents

1 Introduction

The Pandorabot Playground provides a Graphical User Interface to create new and change existing rules in your Teacherbot's knowledge database. This documentation will help you create and maintain a Teacherbot with the Pandorabot Playground GUI. As a pre-requisite to fully benefit from this documentation, we recommend to read the Pandorabot Playground Tutorial, which you can find at https://playground.pandorabots.com/en/tutorial/. Our documentation is an extension to this and provides additional information and useful tips and tricks, but you will have to read up on the basics in the Playground Tutorial if you want to create more complex rules. You can easily spend a few days on the extensive tutorial, but to get you going with the basics it might only need a couple of hours.

The Pandorabot platform is structured into two different components: the **Developer Account** for server access to upload the bot's architecture and implement it into various systems, and the **Playground** for testing the bot's knowledgebase. The Pandorabot Playground is independent of our Developer Account. You have to create a separate account for yourself on the Pandorabot Playground in order to train and test the knowledge base of your Teacherbot. The Playground allows testing your knowledge base and adding more knowledge to it, but it doesn't allow you to upload the database to the server. The Pandorabot Playground can not be used for API access to the Teacherbot core on the Pandorabot server. In order for your changes to go live, you have to download your files from the Pandorabot Playground and upload it to the server via the Developer Account. Although this might seem tedious, it makes sure that changes you make on the Playground will not be effective on the live system straight away and your system can be changed and tested without publishing the new knowledge online. However, this also means that people cannot work on changing the same database simultaneously.

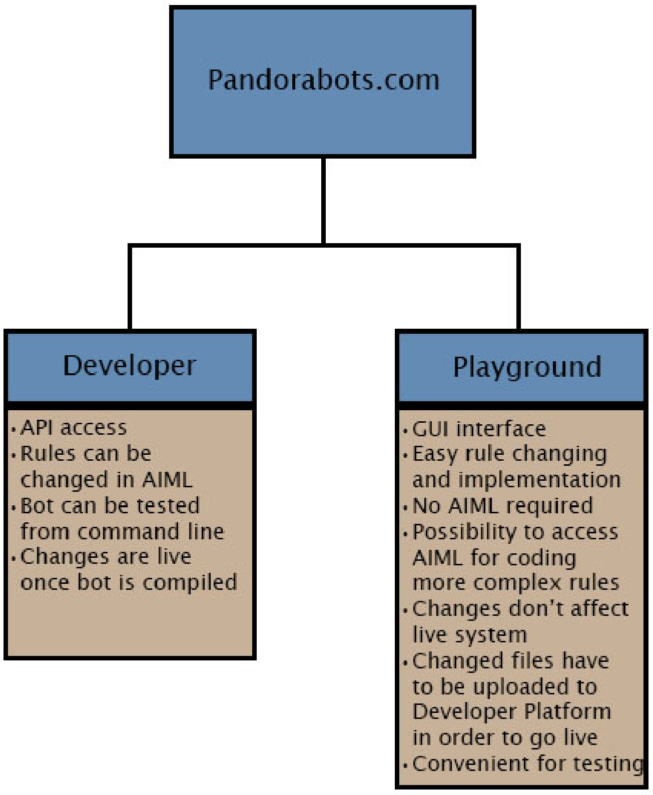
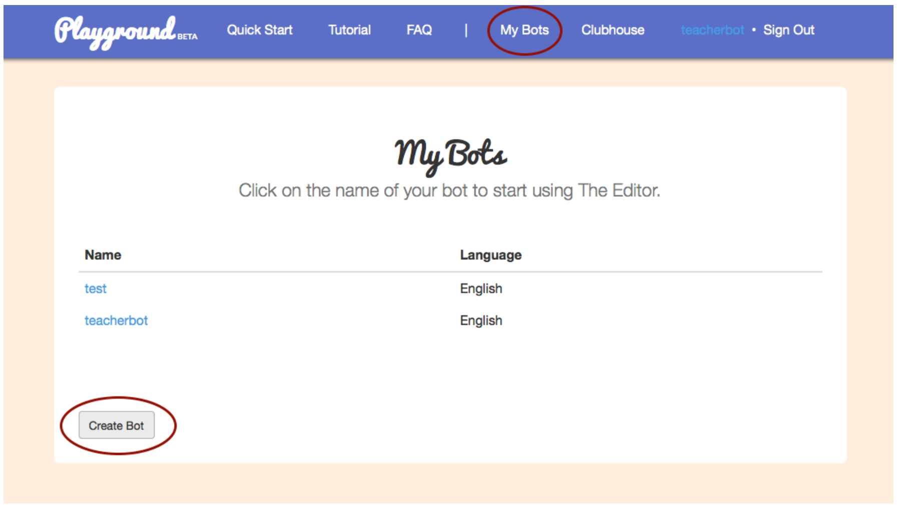

Pandorabots.com
Developer
· API access
· Rules can be changed in AIML
· Bot can be tested from command line
· Changes are live once bot is compiled
Playground
· GUI interface
· Easy rule changing and implementation
· No AIML required
· Possibility to access AIML for coding more complex rules
· Changes don't affect live system
· Changed files have to be uploaded to Developer Platform in order to go live
· Convenient for testing

Figure 1: Overview of the Pandorabot framework.

2 Creating a Teacherbot on the Playground

2.1 First steps

To get started, go to the Pandorabot Playground website at https://playground.pandorabots.com and register for an account. Once you have done this, click on **My Bots** in the blue header menu and create a new Bot by clicking on **Create Bot**.

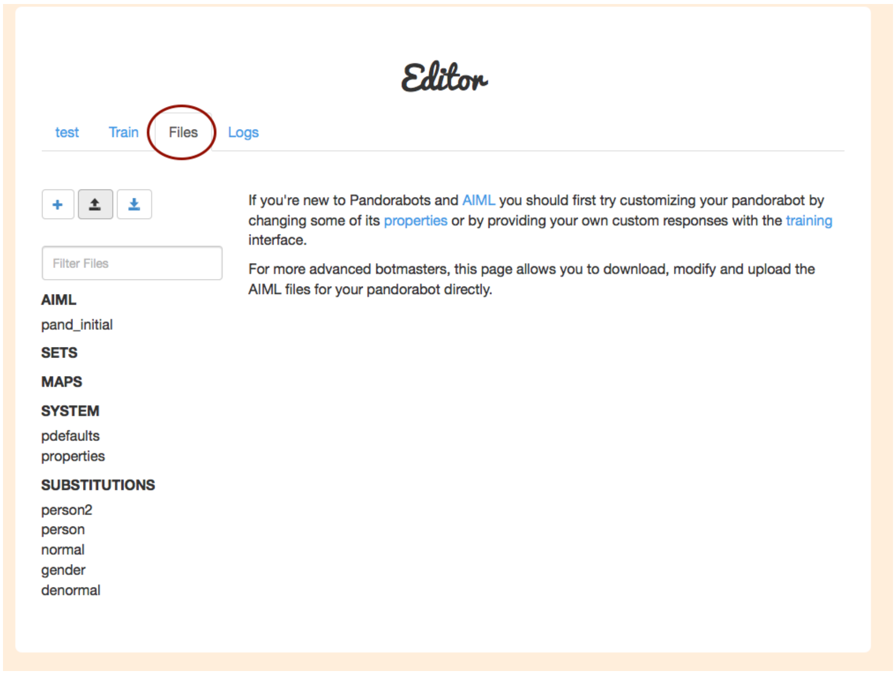


Click on the name of your bot to start using The Editor.

Name	Language
test	English
teacherbot	English

Figure 2: Creating a new Bot instance.

Now you can start filling your Teacherbot with content. Click on the name of the Teacherbot you just created and inspect the Editor that shows up. Go to **Files**.



If you're new to Pandorabots and AIML you should first try customizing your pandorabot by changing some of its properties or by providing your own custom responses with the training interface.

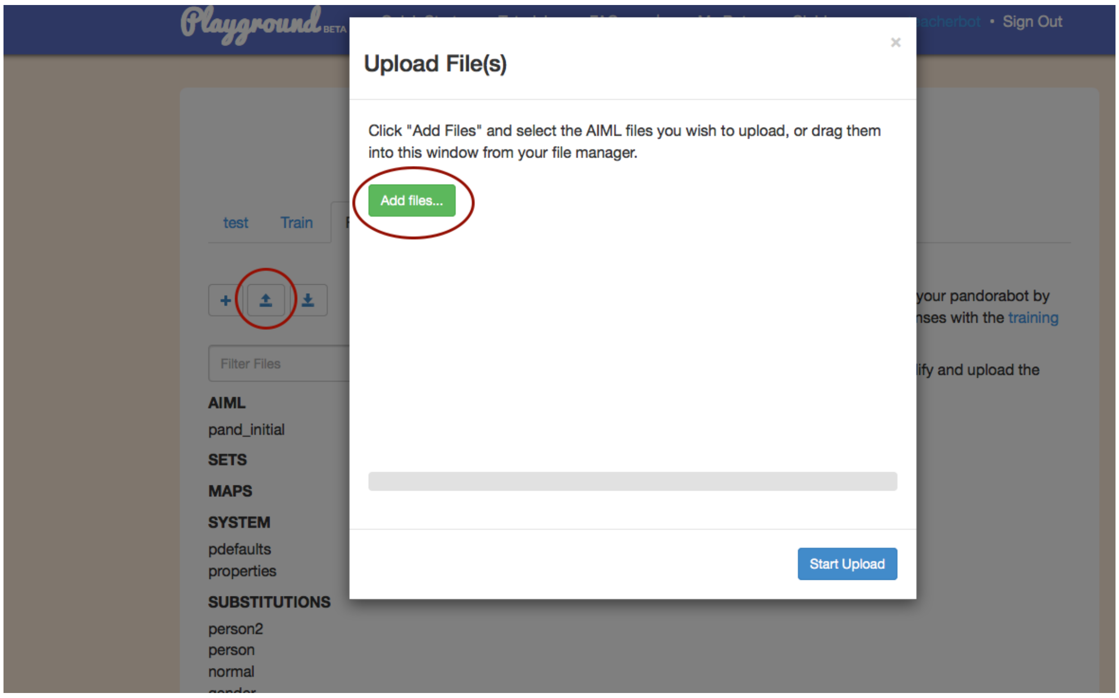
For more advanced botmasters, this page allows you to download, modify and upload the AIML files for your pandorabot directly.

Figure 3: The Pandorabot Editor.

You will see that some default files will already be there. You could start by building a system from scratch, but if you'd like to implement some chit-chat, you can import the open-source database Rosie (https://github.com/pandorabots/rosie). This knowledge base forms a general base for any Teacherbot and contains some basic small talk and answers to user inputs that test the Teacherbot's intelligence. For a more detailed description of Rosie's package contents, have a look here: http://blog.pandorabots.com/rosie-customizable-base-content/. Rosie also comes with a customisable **properties** file. We will explain in more detail what this means later, for now you only need to remember that there are certain characteristics of your Teacherbot that you can personalise. We have already changed this file to make it more Teacherbot-like and improved some other bits and pieces, and we therefore recommend to use our package instead (http://bit.ly/1qxY2IS). On the given URL you will find a **master** folder with two sub-folders: **lib-pandorabot** and **lib-teacherbot**. **lib-pandorabot** contains Rosie's knowledge and **lib-teacherbot** additional Teacherbot knowledge that we found useful. This folder will constantly be uploaded with new knowledge. Alternatively, you can create your own bot from scratch.

2.2 Uploading knowledge

In order to upload new knowledge to your Teacherbot, click on the **Upload Files** button in the Editor and then **Add Files** in the pop-up window. Select all files from the Rosie Teacherbot package (optional) and wait until all files have been uploaded. You are now ready to customize your Teacherbot! Alternatively, you can start building your own knowledge base if you do not want to include any pre-defined knowledge. However, if you'd like to practise using the Playground, the package is a good way to start and some examples we provide in these guidelines assume that you have it installed.



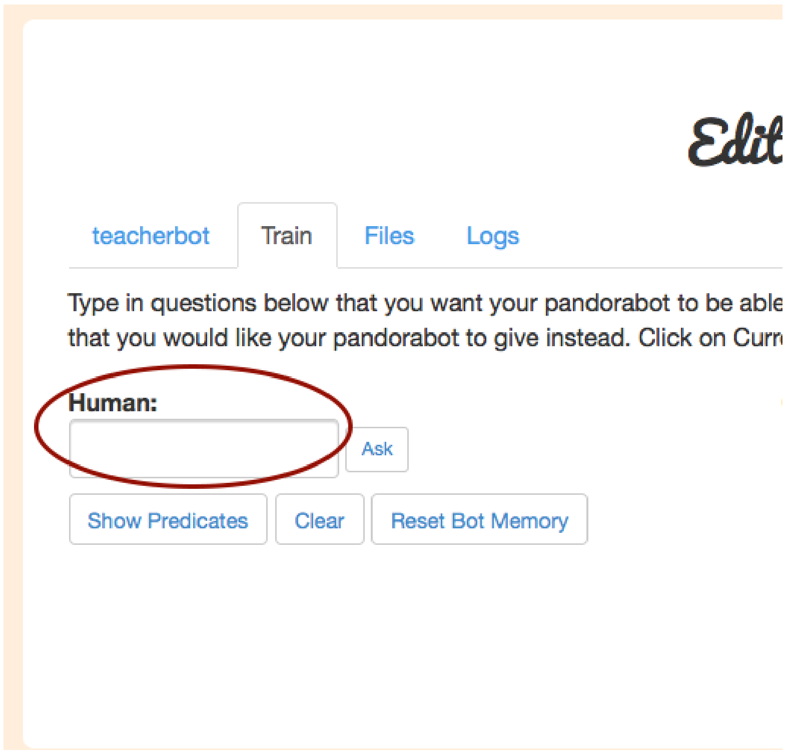
Click "Add Files" and select the AIML files you wish to upload, or drag them into this window from your file manager.

Figure 4: Uploading knowledge to the Playground.

2.3 Adding Knowledge to the Teacherbot

2.3.1 Changing Responses

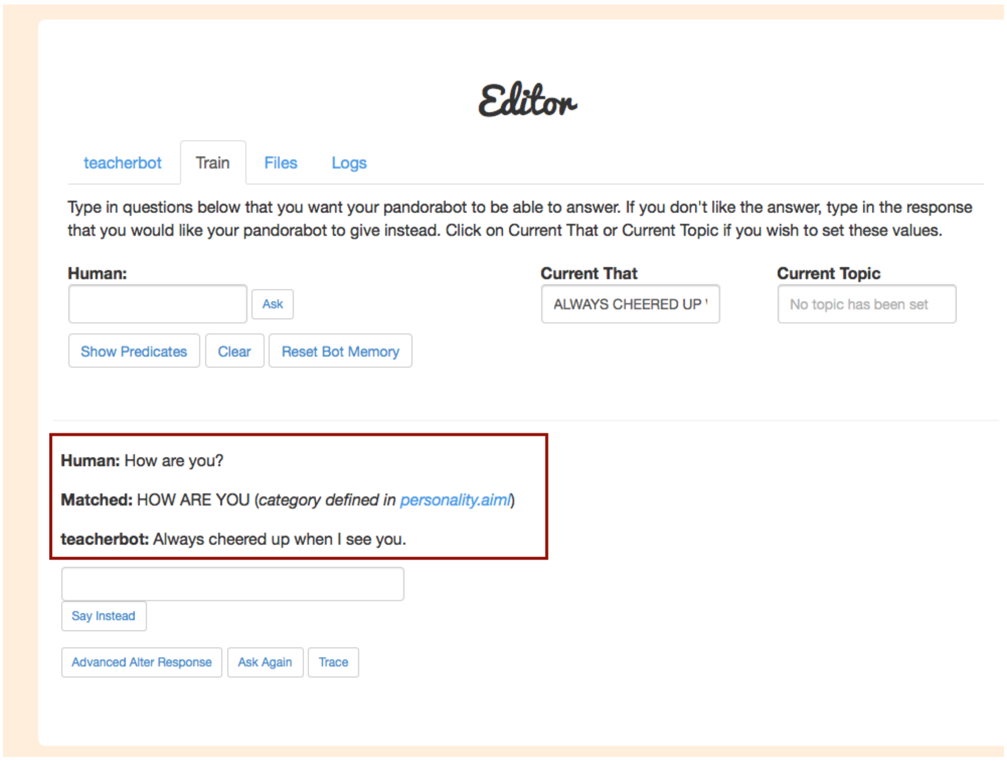
If you are already familiar with AIML and just want to upload your scripts to the Playground, you can skip this part. The following guidelines are for working with the GUI. There are several ways to add knowledge to your Teacherbot using the GUI. We will start with the easy option of changing responses and then gradually move on to more complex matters. Go to **Train** in the Playground Editor.



Type in questions below that you want your pandorabot to be able
that you would like your pandorabot to give instead. Click on Curr

Figure 5: Test interface

On the left side of the window is an input field with the heading **Human**. Ignore the two input fields on the right hand side for now (**Current That** and **Current Topic**), you can read about this in the Pandorabot tutorial. You can test your knowledge base by submitting an input. For example, enter the question *How are you?* and click **Ask**. (Note that this example will only work if you upload the Rosie Teacherbot files.)



Type in questions below that you want your pandorabot to be able to answer. If you don't like the answer, type in the response that you would like your pandorabot to give instead. Click on Current That or Current Topic if you wish to set these values.

Human:

Current That ALWAYS CHEERED UP

Current Topic No topic has been set

Human: How are you?

Matched: HOW ARE YOU (*category defined in personality.aiml*)

teacherbot: Always cheered up when I see you.

Figure 6: Test interface - response

You will see a dialogue showing up below the input field. This contains:

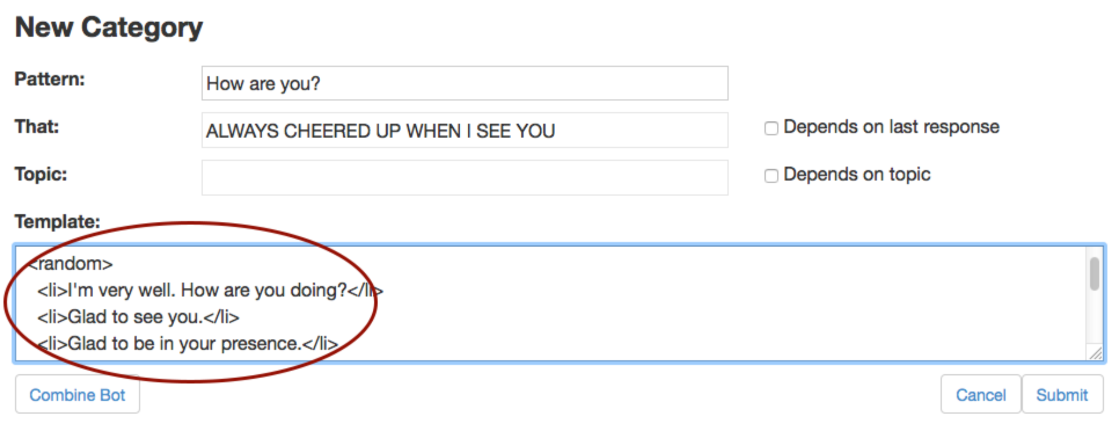
1. Your question *How are you?*

2. The rule it matched

3. The Teacherbot's response to your question

Next to the matched rule you will also find a link to the file in which the rule is defined, in this case **personality.aiml**. You can click on the file name to inspect the file and see what the rules look like in AIML. If you feel comfortable doing so, you can also change the Teacherbot's responses in this file, but if that looks too cryptic, don't worry. To change a Teacherbot's response in the GUI, enter an alternative response in the field above **Say Instead** and click on **Say Instead**. The new response is saved, and the old one deleted.

In order to make your Teacherbot appear more intelligent, you can also add a variety of different responses that will be randomized, so the Teacherbot doesn't always give the same answer when asked *How are you?*. In order to do this, click on **Advanced Alter Response**.


New Category

Pattern: How are you?

That: ALWAYS CHEERED UP WHEN I SEE YOU

☐ Depends on last response

Topic:

☐ Depends on topic

Template:

```
<random>
 <li>I'm very well. How are you doing?</li>
 <li>Glad to see you.</li>
 <li>Glad to be in your presence.</li>
```

Figure 7: Adding alternative responses

Another window appears at the bottom of the editor, you want to change the content of the **Template** box. Alternative responses are between <li></li> tags and these are nested between the tags <random></random> within your template. For example, here is a set of randomised responses to the question *How are you?*:

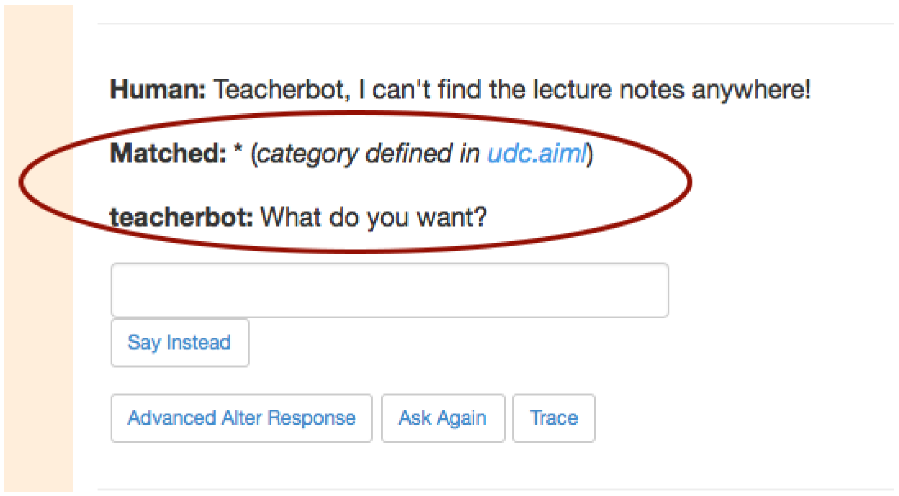
```
<template>
  <random>
    <li>I'm very well. How are you doing?</li>
    <li>Glad to see you.</li>
    <li>Always cheered up when I see you.</li>
  </random>
</template>
```

Make sure that the randomised responses always follow this pattern. Now that you have learned some basic AIML, you can also go to the .aiml file directly and make changes within the file. This is much faster because you don't have to ask a question first before you can edit responses, and it works in exactly the same way!

2.3.2 Adding new questions and responses

Adding your own set of questions and answers works similar to changing existing responses. Go to **Train** in the Playground Editor and type the question you would like the Teacherbot to understand into the entry field on the left. For now, let's take the example *Teacherbot, I can't find the lecture notes anywhere!*

If no matching rule exists, this input will match *, and anything that matches * will run to the **Ultimate Default Category**, which means the Teacherbot will point out that it didn't understand the user.

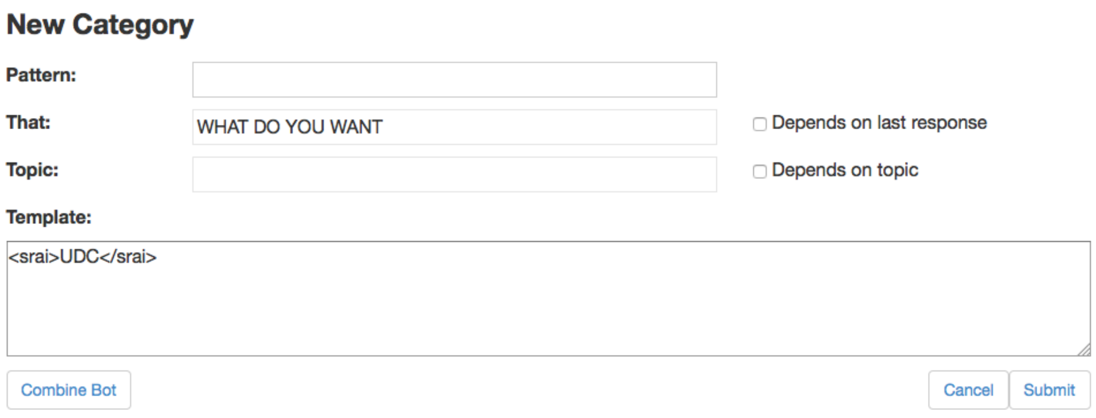

Human: Teacherbot, I can't find the lecture notes anywhere!

Matched: * (*category defined in udc.aiml*)

teacherbot: What do you want?

Figure 8: Ultimate Default Category response

We want to avoid this and add a new response to the Teacherbot. Click on **Advanced Alter Response**.


New Category

Pattern:		
That:	WHAT DO YOU WANT	☐ Depends on last response
Topic:		☐ Depends on topic

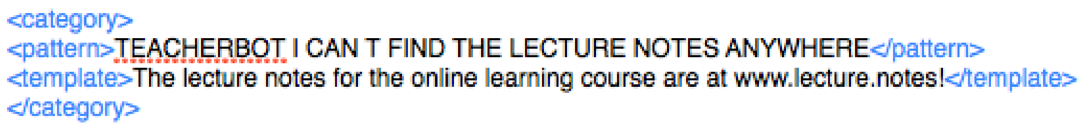
Template:

```
<srai>UDC</srai>
```

Figure 9: Adding a new response

Remove all existing content from the entry fields. This is very important if you want your input/output to work! If you are not able to remove content from the **THAT** field, tick the box **Depends on last response**, then remove the content from the **THAT** field, untick the box **Depends on last response**.

Add your question to the **Pattern** field if it's not already in there. Then type your desired response into the **Template** box. You can add alternative responses using the <li></li>tags as explained in the previous sub-section **Changing Responses**. When you're done, click **Submit**. You can have a look at the .aiml that contains your new rule by going to the **Files** tab, then look for the file **pand_learn** in the AIML folder. This file gets filled with all the rules you create via the GUI. If you create many different rules, you could consider creating various AIML files that cover specific topics. You can also create AIML files with a text editor and then upload them to your Playground, you don't have to use the GUI.



```
<category>
<pattern>TEACHERBOT I CAN T FIND THE LECTURE NOTES ANYWHERE</pattern>
<template>The lecture notes for the online learning course are at www.lecture.notes!</template>
</category>
```

Figure 10: A very basic example rule. Note that punctuation gets automatically removed and the text is converted to uppercase when using the GUI.

2.3.3 More Complex Patterns

In the previous sub-section we learnt how to add new knowledge to our database. However, the pattern we used to match the user input is very basic. In fact, it is so basic that only inputs that look exactly the same will be understood by the Teacherbot. In our example case, only if the user inputs *Teacherbot, I can't find the lecture notes anywhere*, the Teacherbot will be able to understand what the user wants. This is not practical. What if the user asks, *Where are the lecture notes?* We would probably want the same response from the Teacherbot in this case. Let's see how we can solve this.

We could come up with a large amount of possible patterns that the user might input as a question to the Teacherbot. We could add each of these as a pattern/template pair. But then we would end up with a large number of pattern/template pairs. While this is easy to do, it is difficult to maintain. Imagine you want to change the response to that question because the location of your lecture notes changed. You would have to change the response in each of your pattern/template pairs!

Thankfully, the Pandorabot Platform offers a much neater solution. However, you have to dive a bit into the rules and conditions you have at your disposal, which is why we recommend you read the Pandorabot Playground Tutorial.

If you have read the tutorial, you have come across the notion of **wildcards**. So if you have a variety of different user inputs in mind that are somewhat similar, you can use wildcards to write your **pattern**. Remember that wildcards can represent one or more user inputs (if * is used) or zero or more user inputs (if ˆ

is used). Note that there should be a white space between the wildcard and the preceding/following words. Let's look at an example.

ˆ *WHERE * LECTURE NOTES* ˆ

If you use this pattern, your Teacherbot will understand the following user inputs:

- *Where are the lecture notes?*
- *Teacherbot, where are the lecture notes?*
- *Hey Teacherbot, where can I find the lecture notes for the online learning course?*

If you used * instead of ˆ at the beginning and the end of the pattern, then it would only match if something precedes *WHERE* and follows *NOTES*. The ˆ makes surrounding text optional because it matches zero or more inputs. It might be useful to use ˆ at the beginning and end of each pattern, but it depends on how similar your input patterns are. If you code too loosely, a pattern might match inputs it shouldn't match. For example, if you offer several classes and a student asks for material about a specific subject, you might want to include this in the pattern instead of using a wildcard.

This answers the question how to make patterns more flexible, but we still don't know how to make things easier if two user inputs are very different. In our case, the examples are

- *Teacherbot, I can't find the lecture notes anywhere!*
- *Where are the lecture notes?*

There is a very neat solution to this. We can write one question/answer or pattern/template pair. Then we can write another pattern and link back to the first one. We can do this as many times as we like and always put a link to our main pattern/template pair. This way, if we wanted to change the Teacherbot's reply, we only have to do it once. Let's have a closer look at this.

Let's suppose we have already created a pattern for

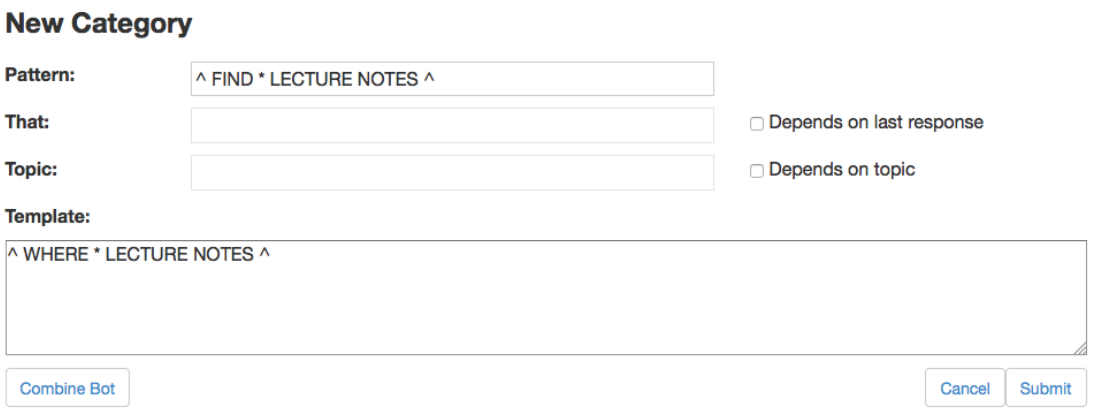
ˆ *WHERE * LECTURE NOTES* ˆ

We go to the **Train** tab in the Playground Editor. Type another pattern into the entry field, for example

ˆ *FIND * LECTURE NOTES* ˆ

Then, click on **Advanced Alter Response** and in the **Template** box write:

*<srai>^ WHERE * LECTURE NOTES ^</srai>*


New Category

Pattern: ^ FIND * LECTURE NOTES ^

That:

Topic:

Template:

^ WHERE * LECTURE NOTES ^

Figure 11: Linking to another template

The <srai></srai>tags are used to link to the response of the pattern *^ WHERE * LECTURE NOTES ^*. Therefore you have to make sure that the text between the <srai></srai>tags matches exactly the same pattern as that of your main pattern you are linking to. You can have a look at the file **pand_learn** in the AIML folder if you want to learn how to write this in AIML.

2.3.4 Advanced: Sets

Sometimes, if you have many alternatives for a possible user input, typing all these different patterns becomes a bit tedious. Imagine, you have a student asking for lecture notes, they could ask:

- Where are the lecture notes?
- Where is the lecture material?
- Where are the slides from the lecture XY?

Instead of making a pattern/template pair for all of these, wouldn't it be nice if we could use a box of synonyms and put everything into one pattern? **Sets** allow us to do this. **Sets** are files that contain lists of items from the same category, for example *animals*, *colours* or *learning material*. You can find **Sets** under the **Files** tab. If you are not using the Rosie Teacherbot package, then this category will be empty, but you can create a new **Set** file by clicking on **Create File** (the

"PLUS" symbol). Make sure to select **Set** from the drop-down menu. A blank page appears on the right hand side along with a box called **item**.

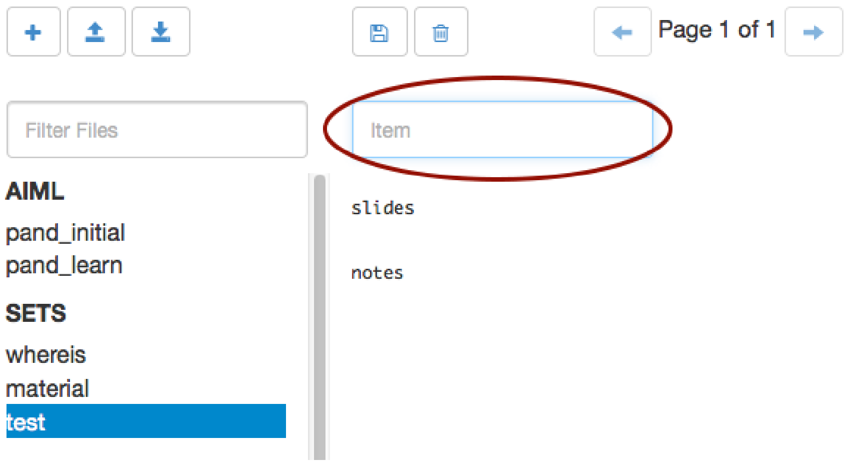


Figure 12: Sets

You can add words to your set by typing a word into the **item** box and pressing **enter**. Words have to be added one by one (one word per line in the Editor). Only group words together into one set that have a similar meaning. If you want to add another set you have to create a new file. Make sure to add the singular and plural version of each word and if you notice from your log files that users frequently misspell a word, it is also a good idea to include words with spelling errors. Let's suppose you have filled a file called **lecturematerial** with the words *slides,notes,resources*. You include your **Set** like this:

ˆ *WHERE * <set>lecturematerial </set>* ˆ

Now your Teacherbot is able to understand all inputs that refer to different words for *lecture material* in your **Set**. You could make a different **Set** for words or phrases that users might use to ask for where to find something. For example, make a **Set** called *whereis* and include *I can't find*, *where is*, *where do I find*, *help me find* and anything else you can think of. Be careful not to use patterns that are too strict to allow for some flexibility but don't make them too flexible to avoid ambiguity. Now you can use the following in your pattern:

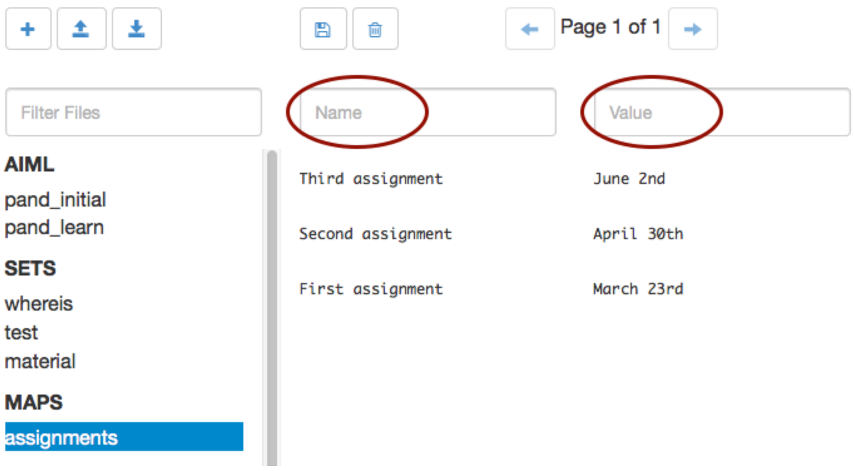
ˆ *<set>whereis</set> * <set>lecturematerial</set>* ˆ

It is useful to include wildcards in the actual pattern to allow for the recognition of full sentences. Your Teacherbot is now able to understand many similar user inputs! **Sets** can be used for a variety of scenarios. Be creative and see what you can come up with!

NOTE: If you use the Playground Editor to add knowledge to your Teacherbot rather than coding directly in AIML, the <set></set>tags will not be translated into AIML. In this case, they will show up as **SET LECTURENOTES SET** in the AIML file. We therefore advise not to build Sets with the GUI and code in AIML directly.

2.3.5 Advanced: Maps

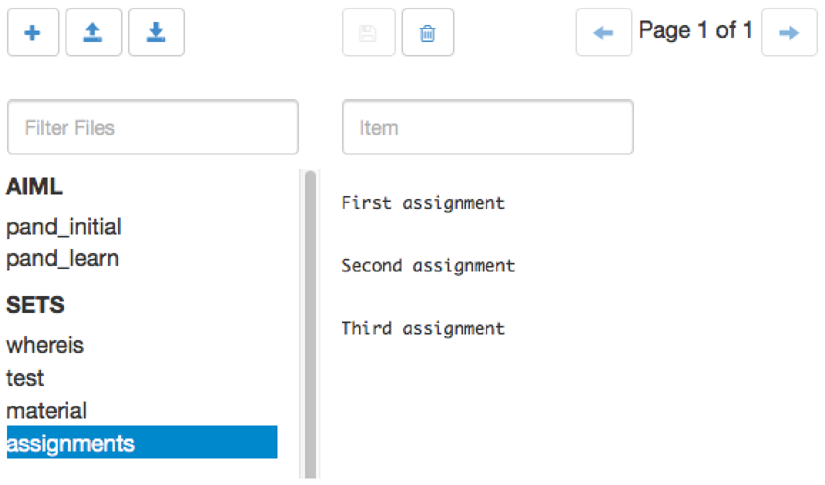
Maps function as look up tables or dictionaries. They are similar to **Sets** but have key-value pairs. They are useful if you have a list of events with corresponding dates (e.g. deadlines for different assignments) or a list of resources with corresponding web links. Let's look at an example. Imagine you are building a Teacherbot for a seminar and the students have to take 3 assignments. In the **Editor** under the **Files** tab, create a new file and select **Map** from the drop-down menu. You can add the name of your assignment into the **Name** field and the date of the deadline into the **Value** field. Press enter and then do the same for assignment 2 and assignment 3. Note that a name always needs a value, name and value only come in pairs!



Third assignment	June 2nd
Second assignment	April 30th
First assignment	March 23rd

Figure 13: Maps

The **Map** you just created only manages the response of the Teacherbot and functions as a lookup table. This way, the Teacherbot will know that Assignment 1 is on March 23rd but we have to make sure that the user input is understood as well. This is done with the help of a **Set**. You can create a set as explained previously and fill it with all your assignments:



First assignment

Second assignment

Third assignment

Figure 14: Assignment set

If there are several possible names for your assignments, include them all in your **Set** and make separate mappings for them in your **Map** file. Now you can write a pattern and template that includes your **Set** and **Map**. Go to the **Train** tab in the Pandorabot Playground. Into the **Ask** box, write

When is the deadline for the <set>assignments</set>

The Teacherbot will have no answer to this, so click on **Advanced Alter Response** and into the **Template** field write:

The deadline for <star/>is <map name="assignments"><star/></map>

The <star/>tags function as place holders for the missing words to be inserted. Think of it as a cloze with the <star/>tags being the blanks.

NOTE: If you use the Playground Editor to add knowledge to your Teacherbot rather than coding directly in AIML, the <set></set>tags will not be translated into AIML. In this case, they will show up as **SET ASSIGNMENT SET** in the AIML file. We therefore advise not to build Sets and Maps with the GUI and code in AIML directly.

Sometimes, a student might ask for an assignment deadline that is not in the **Maps** lookup table. In that case it is helpful to have a backup answer. For example, you could make a pattern that includes a wildcard, like this:

*WHEN IS THE DEADLINE FOR **

And then in the template you would write something like this:

I don't know when the deadline for <star/>is

Again, the <star/>functions as a place holder, but this time for anything the user inputs in place of the wildcard. For example, if the user asked *When is the deadline for the online learning exam?*, the Teacherbot would reply *I don't know when the deadline for the online learning exam is.* Make sure to phrase your template in a way that allows for syntactic variation in the user input.

Note: If you use more than one wildcard in your pattern, for example to make potential user inputs more flexible, you will have to **index** the correct wildcard. Otherwise the default reference for the output template is the first wildcard in the pattern. Let's look at an example.

If your pattern is:

ˆ *WHEN * DEADLINE FOR **

because you have three wildcards, your template should look like this:

I don't know when the deadline for <star index="3"/>is

2.3.6 Advanced: That and Topic

THAT and **TOPIC** allow your Teacherbot to remember parts of the conversation. This way you can set up a more dialogue like conversation and make your Teacherbot appear more intelligent. While **THAT** only spans over the current input/output pair, a **TOPIC** is remembered throughout the entire conversation. For short Twitter based queries, **TOPIC** is probably not relevant for you. You can read more about this in the Pandorabot Tutorial. We will give you an example of how to implement **THAT**.

Imagine a student is asking for the deadline of an assignment, but they don't specify what this assignment is, they only ask *What is the assignment deadline?*. We created a list of specific assignments and deadlines using **Maps** earlier, but the Teacherbot is not able to access it based on the student's very general question. Wouldn't it be good if the Teacherbot could ask for more information? This is possible using **That**.

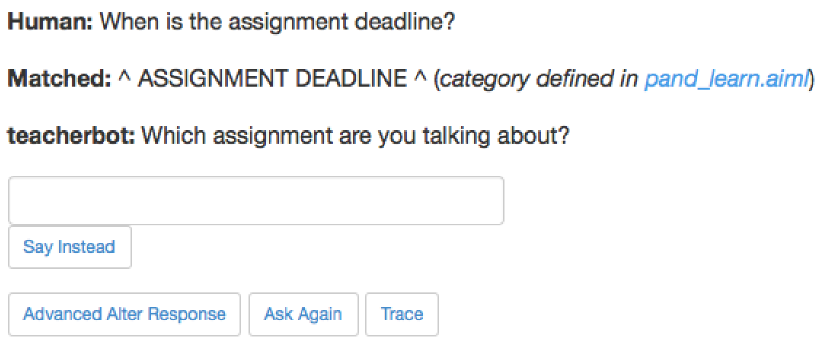
Go to the **Train** tab in the Pandorabot editor and make a pattern/template pair for the very general user input *What is the assignment deadline?*. The pattern could look like this:

ˆ *ASSIGNMENT DEADLINE* ˆ

The template should be a question, for example:

Which assignment are you talking about?

Test your pattern/template pair by typing a possible user input into the **Ask** field and click **Ask**. The output will look like what you see in Figure 15. So far it worked exactly like making a simple pattern/template pair.


Human: When is the assignment deadline?

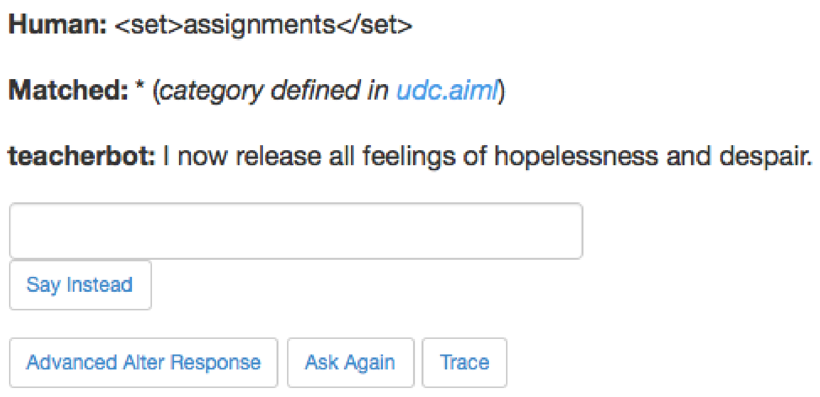
Matched: ^ ASSIGNMENT DEADLINE ^ (*category defined in pand_learn.aiml*)

teacherbot: Which assignment are you talking about?

Figure 15: Step 1 of formulating **THAT**: a simple template/pattern pair

Notice that, once you submitted the question *When is the assignment deadline?*, the **Current That** field next to the **Ask** box gets filled with **WHICH ASSIGNMENT ARE YOU TALKING ABOUT**, which is the response of the Teacherbot minus punctuation and written in upper case. This is to identify the current theme or **That** of the conversation.

Now you can add a follow-up user input, for example one that specifies the assignment. We created a **Set** for this earlier, so we can simply input *<set>assignments</set>* into the **Ask** field.

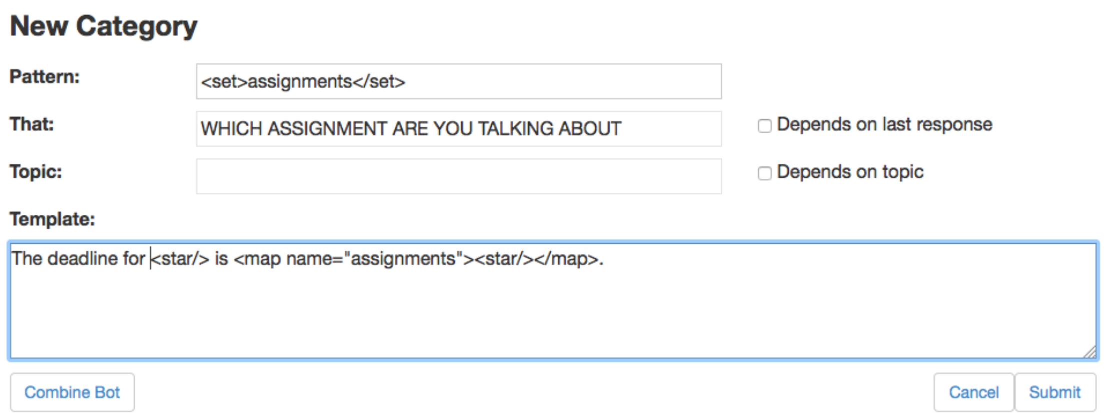

Human: <set>assignments</set>

Matched: * (*category defined in udc.aiml*)

teacherbot: I now release all feelings of hopelessness and despair.

Figure 16: Step 2 of formulating **THAT**: adding a follow-up user input

This will go straight to the **Ultimate Default Category** (Figure 16) because we haven't specified a response yet. Click on **Advanced Alter Response**. Leave everything as it is and add the response into the **Template** field as shown in Figure 17.



Pattern: <set>assignments</set>

That: WHICH ASSIGNMENT ARE YOU TALKING ABOUT

Topic:

Template:
```
The deadline for <star/> is <map name="assignments"><star/></map>.
```

Figure 17: Step 2 of formulating **THAT**: adding a follow-up response using **Current That**

Now we can have the following conversation with the Teacherbot:

User: *When is the assignment deadline?*

Teacherbot: *What assignment are you talking about?*

User: *The first assignment*

Teacherbot: *The deadline for the first assignment is on March 23rd.*

You can test this yourself in the Pandorabot Playground.

2.4 Other features

We have covered the most basic and some more advanced features of the Pandorabot framework. There are more things you can try and if you're interested, have a look at the Pandorabot tutorial.

3 Resources

Here is a list of useful information and tutorials for building a Teacherbot with the Pandorabot framework:

A quick and short introduction to Pandorabots and AIML: http://docs.pandorabots.com/tutorials/getting-started/

Details about **Rosie**, a package of general Bot knowledge: http://docs.pandorabots.com/tutorials/using-rosie/

An in-depth Pandorabot Tutorial, similar to this documentation, good for filling in gaps: https://playground.pandorabots.com/en/tutorial/

A text-editor for coding in AIML. Any text editor will do, but Brackets is good for coding and also free: `http://brackets.io`. The following extension is helpful for formatting your AIML: `https://github.com/brackets-beautify/brackets-beautify`.