NLP Part 3

Understand how a computer attempts to process human language

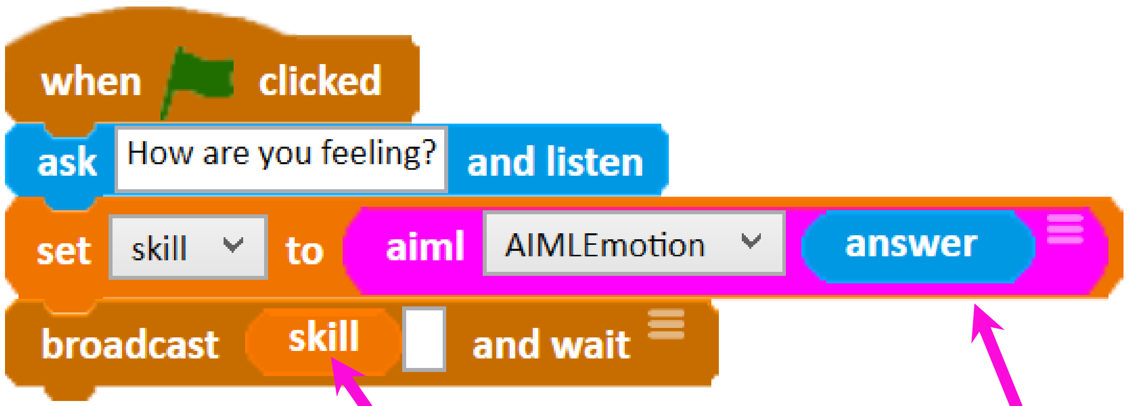
In NLP Part 1 we looked at the advantages and disadvantages of different ways of asking our robot for the time. We found that by reducing sentences to single words or pairs of words, the robot had a good chance of responding appropriately to our instructions.

In NLP Part 2 we used a decision tree to identify our emotions, then classify and recognise them.

Now let’s take a look at a different method for breaking down text to one key word in order to trigger a skill.

Download the example program here.

Can you tell what this program will do when you click the green flag?

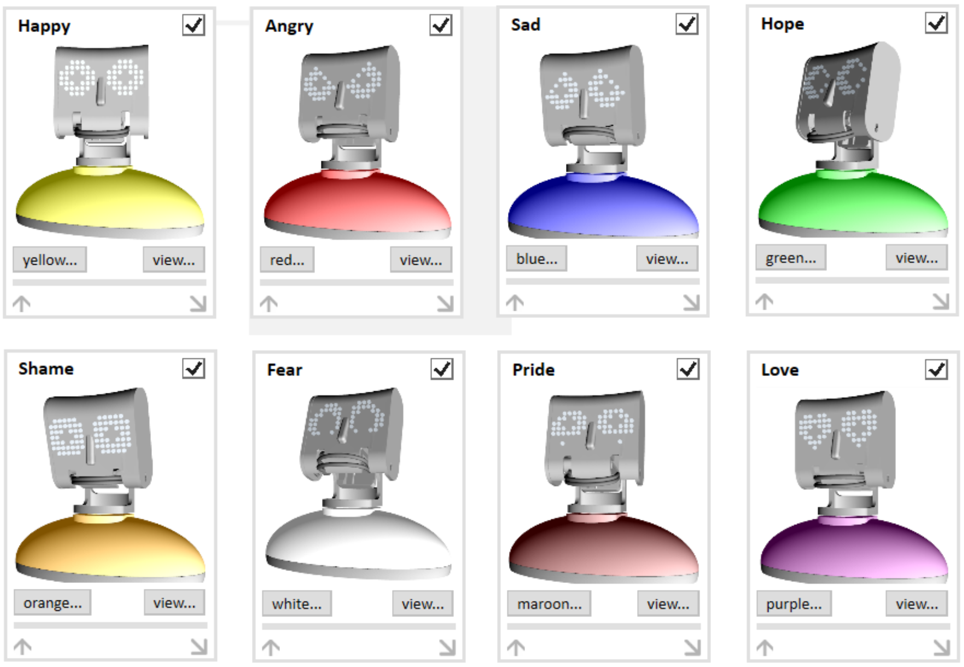


```
when [green flag] clicked
ask [How are you feeling?] and listen
set [skill] to (aiml [AIMLEmotion] (answer))
broadcast (skill) [] and wait
```

The variable "skill" will be set to a word from your answer. The AIML block is responsible for identifying the word

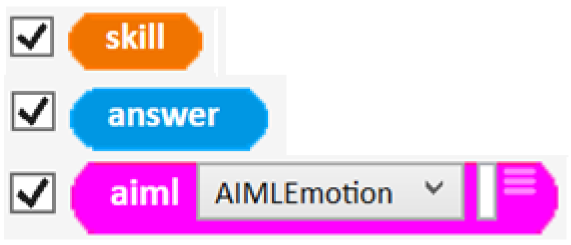
The "skill" is broadcast and will trigger one of a set of multiblocks. If you are unsure of how multiblocks work then take a look at our learning resource on them.

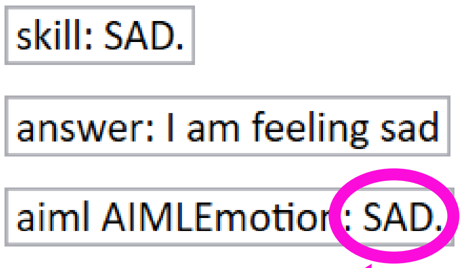
If you tell the robot that “I am feeling sad” then the “Sad” multiblock will run.



I’m feeling blue

To help you understand what the robot is doing, make sure you have the "skill" variable, the "answer" sensing block and the AIMLEmotion block ticked in your palette area.




skill: SAD.

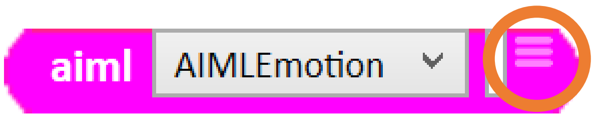
answer: I am feeling sad

aiml AIMLEmotion: SAD.

This is the output from the AIMLEmotion block. It has disregarded the start of the sentence, "I am feeling" and chosen "sad"

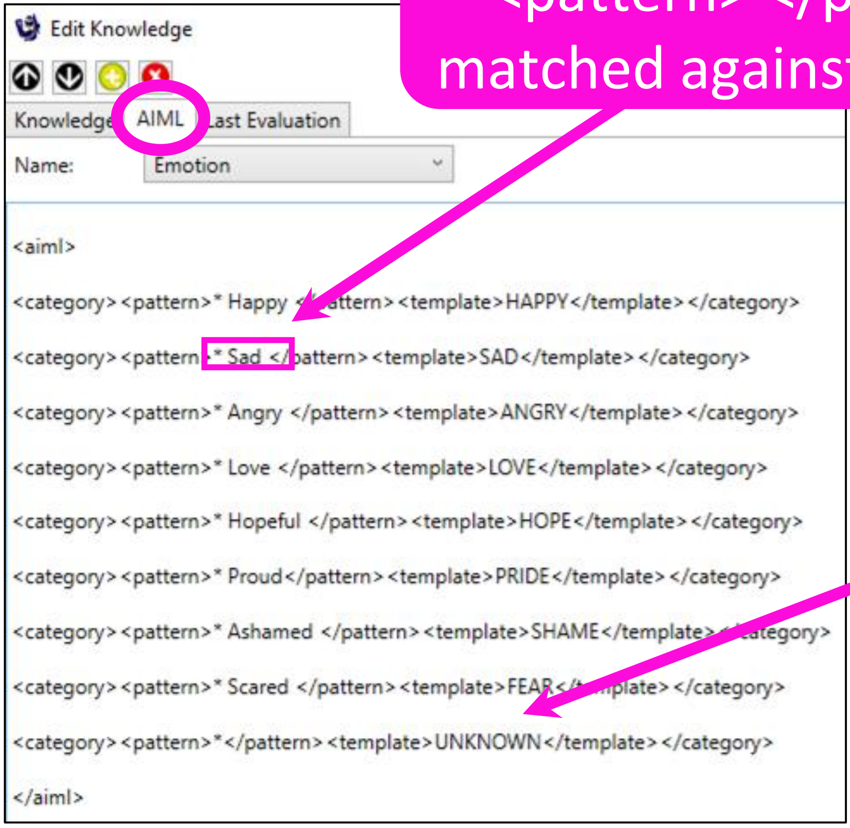
The AIML block is how we are reducing the sentence to one key word. AIML stands for Artificial Intelligence Markup Language. The AIML file that the block is using is called AIMLEmotion. Can you predict how it is selecting the emotion from the sentence?

Click on the three lines on the block to find out.



When you click on the three lines, the “Edit Knowledge” window will open. Click on the AIML tab.

The characters in the <pattern> </pattern> are matched against your answer

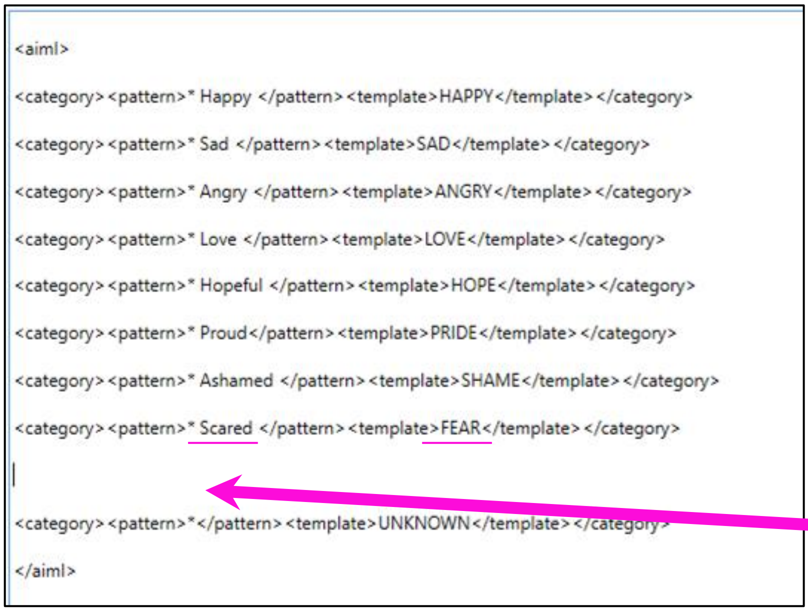


```
<aiml>
<category><pattern>* Happy </pattern><template>HAPPY</template></category>
<category><pattern>* Sad </pattern><template>SAD</template></category>
<category><pattern>* Angry </pattern><template>ANGRY</template></category>
<category><pattern>* Love </pattern><template>LOVE</template></category>
<category><pattern>* Hopeful </pattern><template>HOPE</template></category>
<category><pattern>* Proud</pattern><template>PRIDE</template></category>
<category><pattern>* Ashamed </pattern><template>SHAME</template></category>
<category><pattern>* Scared </pattern><template>FEAR</template></category>
<category><pattern>*</pattern><template>UNKNOWN</template></category>
</aiml>
```

The words in uppercase are the possible outputs from the block, including “Unknown”

In AIML code the “*” tag is used to match wildcard characters. So in our example, the block is only trying to match the emotions listed, in the form they are listed and only if they are preceded by something (like “I am feeling…”). If it doesn’t find an emotion it knows to match then it will return “Unknown”.

You can easily edit the code yourself to improve it. Try adding a new emotion to the list. Don't worry, it won't let you save a mistake.



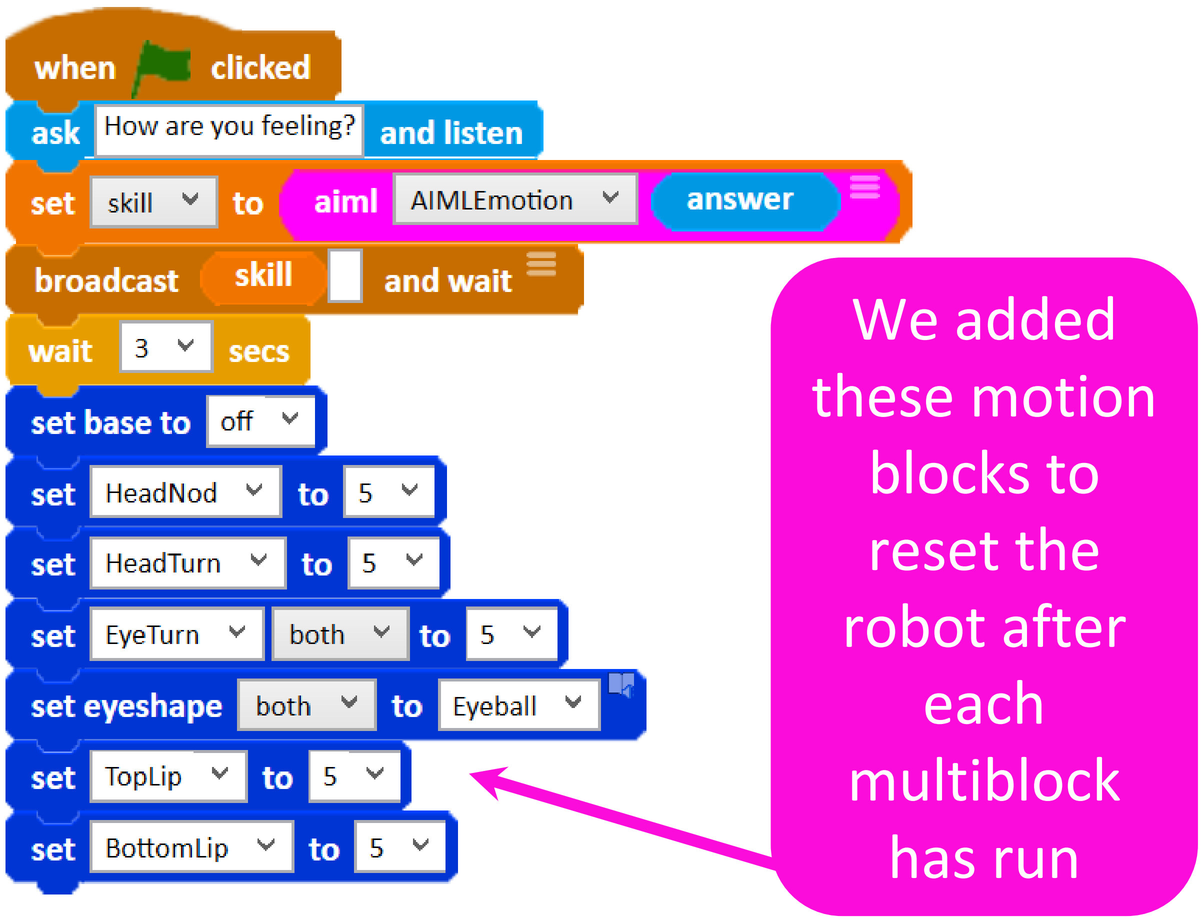
```
<aiml>
<category><pattern>* Happy </pattern><template>HAPPY</template></category>
<category><pattern>* Sad </pattern><template>SAD</template></category>
<category><pattern>* Angry </pattern><template>ANGRY</template></category>
<category><pattern>* Love </pattern><template>LOVE</template></category>
<category><pattern>* Hopeful </pattern><template>HOPE</template></category>
<category><pattern>* Proud</pattern><template>PRIDE</template></category>
<category><pattern>* Ashamed </pattern><template>SHAME</template></category>
<category><pattern>* Scared </pattern><template>FEAR</template></category>

<category><pattern>*</pattern><template>UNKNOWN</template></category>
</aiml>
```

Simply copy and paste an existing line of the code here and change the emotion and output to your new one

Now close the window and try running the program again, saying your new emotion.

Did it output what you wanted it to?

Why not try making your own multiblock for the emotion you added?

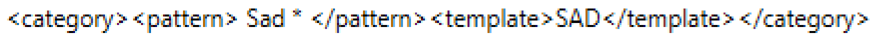


```
when [flag] clicked
ask [How are you feeling?] and listen
set [skill v] to (aiml [AIMLEmotion v] (answer))
broadcast (skill) [ ] and wait
wait [3 v] secs
set base to [off v]
set [HeadNod v] to [5 v]
set [HeadTurn v] to [5 v]
set [EyeTurn v] [both v] to [5 v]
set eyeshape [both v] to [Eyeball v]
set [TopLip v] to [5 v]
set [BottomLip v] to [5 v]
```

We added these motion blocks to reset the robot after each multiblock has run

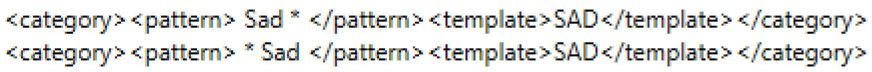
We have already pointed out the flaws in this AIML file. It only matches if the emotion is preceded by some wildcard characters, it only knows the emotions we have listed and it only knows them in the form we have provided.

How would you use the “*” tag to allow the block to match an emotion if it came at the start of a sentence? For example, Yoda might say, “Sad I am feeling.”



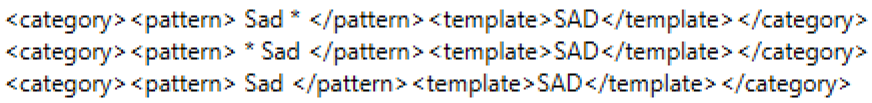
```
<category><pattern> Sad * </pattern><template>SAD</template></category>
```

All you have to do is move the “*” to the other side of the emotion.



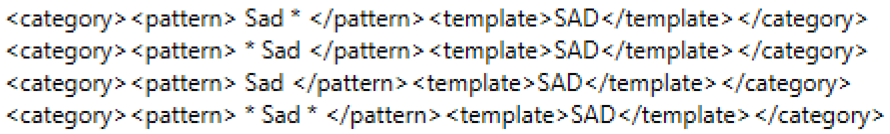
```
<category><pattern> Sad * </pattern><template>SAD</template></category>
<category><pattern> * Sad </pattern><template>SAD</template></category>
```

And if you have both versions in the code then it will match either way!



```
<category><pattern> Sad * </pattern><template>SAD</template></category>
<category><pattern> * Sad </pattern><template>SAD</template></category>
<category><pattern> Sad </pattern><template>SAD</template></category>
```

And if you have a line with no “*” at all then the block will match if you are monosyllabic too!



```
<category><pattern> Sad * </pattern><template>SAD</template></category>
<category><pattern> * Sad </pattern><template>SAD</template></category>
<category><pattern> Sad </pattern><template>SAD</template></category>
<category><pattern> * Sad * </pattern><template>SAD</template></category>
```

And finally, for the more verbose, if you have a “*” on either side of the emotion then it will match if the word is in the middle of the sentence.

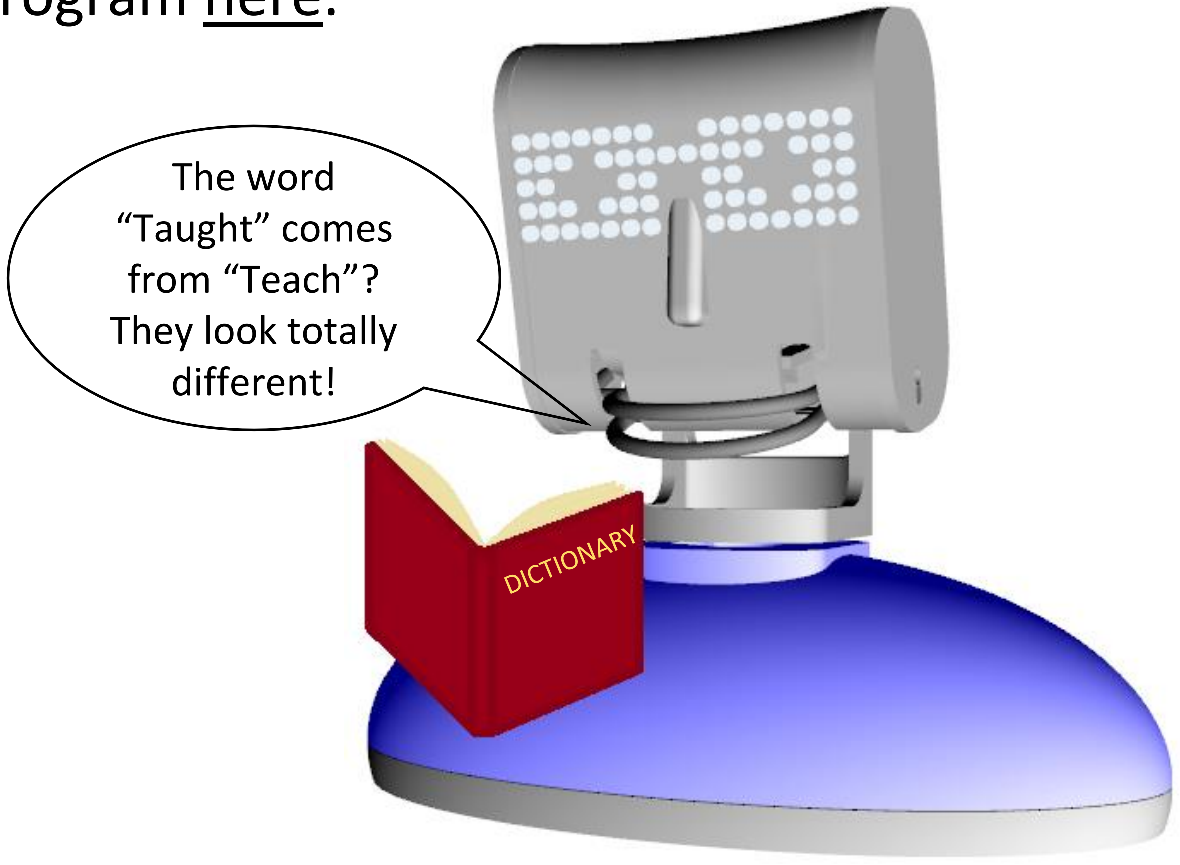
Okay, but what about the fact it only recognises the single form of the word? There are many forms of the word “sad”, like “sadness” and “sadly”. And that’s before we get onto synonyms like “unhappy” and “miserable” and all of their different forms!

The only option is to include all of the variations in the code. But you can make them all share the same output.

```
<category><pattern>* Sad </pattern><template>SAD</template></category>
<category><pattern>* Sadness  </pattern><template>SAD</template></category>
<category><pattern>* Sadly  </pattern><template>SAD</template></category>
<category><pattern>* Unhappy  </pattern><template>SAD</template></category>
<category><pattern>* Miserable  </pattern><template>SAD</template></category>
```

This is an important part of NLP and it is called lemmatisation. Lemmatisation is the process of matching all the different forms of a word back to the root word.

You can download our lemmatisation program here.


The word "Taught" comes from "Teach"? They look totally different!
DICTIONARY