IBM Watson Cheat Sheet

Challenge yourself! How much do you remember from the workshop?

(Answers are provided at the bottom of the page)

1. What type of system is IBM Watson?
2. Name three tasks that AI wants to solve?
3. How many AI techniques does Watson bring together to be able to answer queries?
4. In what games has AI performance surpassed human performance?

Answers to the quiz

1. IBM Watson is a question-answering system
2. Tasks can include: reasoning, planning, natural language processing and understanding, learning, perception. Can you think of any more that should be on this list?
3. More than 100!
4. We mentioned Go, Jeopardy!, and chess in the workshop. Poker and Starcraft are proving much harder - do you have any ideas about why this might be?

Resources and Tips

Predictive analytics and statistical modelling

- Check out www.coursera.org and www.udemy.com - both have highly rated Data Science courses
- Download datasets for free to explore
    - University of California at Irvine: https://archive.ics.uci.edu/ml/index.php
    - This curated this: https://github.com/awesomedata/awesome-public-datasets

Data Visualisation

- Learn to use a data visualisation program like Tableau
- Learn Python or R (check out Sudo's Data Science in R workshop if you need help getting started!)
- Get hold of some data you find interesting from the resources above
- Jupyter Notebook is a very useful tool for exploratory work in Python - you can mix text, code, and images together in a report-style webpage

Natural Language Processing and Understanding

- Python is the most useful language to know for building NLP systems. Check out www.codeacademy.com for a free, beginner's friendly course.
- Take some computational linguistics or NLP classes
- Try building your own chatbot!

General machine learning and AI

- Create your own project using IBM Watson! See the **API** section for more information.

- If you know (or are planning on learning) Python, check out the **scikit-learn** machine learning library. It's very simple to get started with, the documentation is clear, and lots of people use it all over the world (so there are tons of stackoverflow posts if you get stuck!)

API

If you're interested in creating your own project using Watson Conversation and the IBM Watson API, this is the section for you! Below are some helpful tips for getting started. You can also find some beginner-friendly tutorials here:

- https://console.bluemix.net/docs/services/conversation/getting-started.html
- https://www.ibm.com/blogs/watson/2016/12/build-chat-bot/
- https://www.ibm.com/cloud/garage/tutorials/watson_conversation_support/

Getting started:

1. Register and login to IBM Cloud account
2. Provision an instance of the Watson Conversation on BlueMix
    a. *Watson Conversation* allows you to create applications that understand natural language and interact with customers in a conversational manner
    b. *Watson Conversation* is available through Bluemix, so you need to login to Bluemix and create a conversation service in the catalog. Select Watson in the left category to find your service.
    c. In the Bluemix dashboard, go to the service you created in the step above and click the Launch tool button to go to the tool.
3. Then create a conversation workspace by importing ./resources/conversation_workspace.json into Conversation workspaces
    a. See instructions in the **Import a workspace** section below
4. Install client and server dependencies
5. Create .env files in the project with the following contents

Creating a workspace

Workspace consists of:

1. *Intent:* The purpose of the word you want to say (E.g. book a room)
2. *Entity:* Used to provide a specific context for an intent. (E.g. booking a room would define an entity as a conference room)
3. *Dialog:* Branch of the conversation that defines what the application will do when it recognizes the defined intent and entity.

Import a workspace

Copy and paste the following code into a blank page in your favourite text editor (Sudo recommends Sublime Text) and save it as a .json file.

```
APP_ID=parks-conversation
PORT=3004
LOG_LEVEL=debug
SESSION_SECRET=test
WATSON_CONVERSATION_API_ROOT=https://gateway.watsonplatform.net/conversation/api
```

```
WATSON_CONVERSATION_VERSION=v1
WATSON_CONVERSATION_VERSION_DATE=2016-07-11
WATSON_CONVERSATION_USERNAME=<USERNAME>
WATSON_CONVERSATION_PASSWORD=<PASSWORD>
WATSON_CONVERSATION_WORKSPACE_ID=<WORKSPACE
```

Creating an intent

Choose your intent

Example: Choosing a PositiveAnswer value will give you options like “absolutely” or “of course” - when you’re using values in Watson *remember that the values are describing the actions you’re requesting.*
Create a new button to create an entity with the name of the room as shown below.

```
intents [0] == 'Help' or intents == 'Help'
```

Short grammar	SpEL grammar
#help	intent == 'help'
! #help	intent! = 'help'
NOT #help	intent! = 'help'
#help or #i_am_lost	(intent == 'help' || intent == 'I_am_lost')

Creating an entity

Create a new conference room (your entity) and then create an entity with the name of the room as shown below. The value is the name of the room to be used on each system and the synonyms are all words that the user can use to call a specific room.

Enable these two entities after creating them.

Creating a dialog

conversation_start
#greeting
#reservation
Anything_else

Terminology

- **Node**: A node is a component of a Dialog. This means one interaction in conversation with the user.
- **Condition**: The element that constitutes a node responds to the condition if it satisfies the condition based on the user's input.
- **Response**: The element that constitutes the node and is activated when the user's input matches the condition. You can return a simple string or you can run a more complex process.
- **Branch**: A series of Dialog Nodes, meaning dialog with the user, is called a branch. The branch will be used when the user's input matches the condition of the primary node.
- **Turn**: One cycle in which the user makes an input and responds to it is called a turn. A branch can have one or more turns.
- **Base node**: The top node (first node) of the branch is called the primary node.
- **Child node**: A node that is not a primary node is called a child node. The child node is used when the previous node needs more user input or process to give a final response.
- **Synchronous node** or **sibling node**: A node that is an alternative node to another node. All primary nodes are synchronous nodes of other primary nodes.
- **Node menu**: Allows you to erase a node or add functionality to it as an element of a node.

When creating nodes, remember that you must use Spring Expression Language (SpEL).
The following are global variables used for IBM Cloud and Watson.

intents []	List of Intents
entities []	List of entities
input	The JSON object that contains the user's input
output	A JSON object containing the output of Watson
context	A JSON object that contains the conversation context between the user and Watson
conversation_start	Can be used as a condition of welcome message as a Boolean value that will be *true* when starting the conversation for the first time
anything_else	When the user input value does not match any intent, the output value of the node using this Condition is used

For string types:

- String.length ()
- String.toUpperCase ()
- String.toLowerCase ()
- String.endsWith (string)
- String.startsWith (string)
- String.isEmpty ()
- String.trim ()
- String.substring (int beginindex, int endindex)
- String.append (object)
- String.matches (String regexp)
- String.extract (String regexp, Integer groupIndex)
- String.split (String regexp)

For functions available for numeric types:

- Number.toInt ()
- Number.toLong ()
- Number.toDouble ()

For functions available for the JSONObject type:

- JSONObject.has (string)
- JSONObject.remove (string)

For functions that are available for the JSONArray type:

- JSONArray.size ()
- JSONArray.append (object)
- JSONArray.remove (integer)
- JSONArray.removeValue (object)
- JSONArray.get (integer)
- JSONArray.getRandomItem ()
- JSONArray.join (string delimiter)
- JSONArray.contains (object value)
- JSONArray.set (integer index, object value)