

Creating an Artificially Intelligent Chat Bot Powered with Node.js

John Harden

Principal Software Engineer - Axia Technology Partners

Have you considered adding automated support to your organization? Would you like to be able to provide interactive applications that immerse the user? Are you interested in creating your own meme creation bot? Then this guide is for you!

The explosive growth of chat bots, paired with the potential for automation on multiple levels, has driven a huge demand for understanding & utilizing these automated bots successfully. This presentation focuses on creating an intelligent chat bot powered by Node.js. This document will guide you through the construction of a Meme bot communicating in Slack.

TABLE OF CONTENTS

page 1:	Cover Page
page 2:	Table of Contents
page 3:	(see below)
	<ul style="list-style-type: none"> i. Step 1: Downloading the necessary files ii. Setting up the Node Environment iii. Acquiring the Slack Bot integration Token and Adding it
page 4:	Creating a new API.AI Agent
page 5:	(see below)
	<ul style="list-style-type: none"> i. Making sure your bot is connected and works (step1.js) ii. Getting your bots information and the users/groups info (step2.js)
page 6:	(see below)
	<ul style="list-style-type: none"> i. Making sure your bot can see you (step3.js) ii. Making first contact! (step4.js) iii. Getting your bot to talk to you (step5.js)
page 7:	Setting up some intelligence with your bot
page 8:	Taking the intelligence and chaining intents
page 9:	Creating a meme entity
page 10:	Creating a meme entity continued...
page 11:	Finishing the intelligence
page 12:	Finishing the intelligence continued...
page 13:	Generate your meme!
page 14:	Generate your meme! continued...
page 15:	Getting your meme into slack! (Final step!) (step7.js)
page 16:	(see below)
	<ul style="list-style-type: none"> i. Challenging yourself ii. Reference/Contact Information

STEP 1: DOWNLOADING THE NECESSARY FILES

1. Utilize github to clone <https://github.com/johnharden/memebot>
- or -
2. Download at <https://github.com/johnharden/memebot/archive/master.zip>

STEP 1: SETTING UP THE NODE ENVIRONMENT

1. First create a new server or, if capable, install the node environment on your computer.
2. Run the following commands to install the environment: (Below is for Ubuntu 16.04), visit <https://nodejs.org/en/download/package-manager/> for your install guide

```
curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
sudo apt-get install -y nodejs
```

3. Install the required node packages for the project:

```
cd /path/to/node/project/

npm install apiai
npm install http
npm install slackbots
npm install request
```

4. You'll want to extract the files from the .zip belonging to this walkthrough in /path/to/node/project/. The compressed files should be as follows:
 - a. step1.js
 - b. step2.js
 - c. ...
 - d. step7.js
 - e. variables.js

STEP 2: ACQUIRING THE SLACK BOT INTEGRATION TOKEN AND ADDING IT

1. First login to slack at <https://slack.com/signin>
2. Once authenticated, navigate to <https://my.slack.com/services/new/bot>
3. For this demonstration we are going to name our bot **@memebot**
4. Once created you'll be redirected to a page where there is information to be filled out. Feel free to set up your icons, first & last name and a description of the bot although this is not necessary.

- There is a section for the API Token. This is private, do not expose/share this information. It can be found in this section:

Integration Settings

API Token

The library you are using will want an API token for your bot.

xxxb-153729982931-rcBOQXMGh56YnzKVFa7dxKk5

[Regenerate](#)



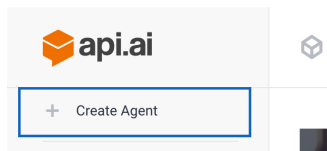
Be careful when sharing bot user tokens with applications. Do not publish bot user tokens in public code repositories. [Review token safety tips](#).

- Copy + Paste that code into the variables.js file where the parameter says

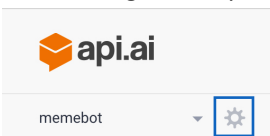
```
slack_bot_key: "{insert_slack_bot_key_here}"
```

STEP 3: CREATING A NEW API.AI AGENT

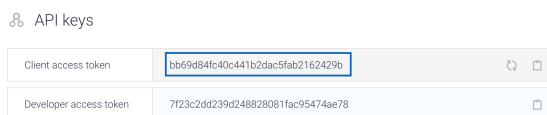
- First sign up for a new account at <https://api.ai/>
- When you sign up there is a short video explaining API.ai it is worth watching.
- Create a new Agent by clicking the button the upper left:



- Name your agent **memebot**, select whether you want it public or private and click the **SAVE** button.
- Upon creation you will see there are two **intents**. These will be your default intents, but we will remove these later in the demo.
 - intents represents a mapping between what a user says and what action should be taken by your software (learn more: <https://docs.api.ai/docs/concept-intents>)
- Click the cog next to your bot:



- In the section labeled API keys you'll want see your client access token:



- Copy + Paste that code into the variables.js file where the parameter says

```
api_ai_key: "{insert_api_ai_key_here}"
```

STEP 4: MAKING SURE YOUR BOT IS CONNECTED AND WORKS (STEP1.JS)

1. Enter a channel (this cannot done in DM) in slack that you want to monitor and type /invite @memebot
2. You'll want to move to /path/to/node/project and execute the first step of code

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node.step1.js
```

3. You'll likely see a lot of JSON information after the text "my bot works". This JSON information is the current presence information of your slack channels.
 - a. If you are seeing "Error: invalid_auth" verify both your slack api key and api.ai key are correct.
4. Exit the console and attempt to finish the below tasks
5. **TASKS:** (I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step2.js)
 - a. **TASK 1:** we need to create an associative array of both the groups[] and channels[] storing them as such groups = {id1:name, id2: name}
 - b. **TASK 2:** we need to go through the users and create an array of the users to store their current conversation + identify the bots information

STEP 5: GETTING YOUR BOTS INFORMATION AND THE USERS/GROUPS INFO (STEP2.JS)

1. After completing those tasks, jump into step2s code and see what has changed.
2. Execute step2s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node.step2.js
```

3. You should see the bot's information displayed. If you are not seeing this information, then you should consult the bot_params at the top of the page to make sure you have the slack_realname matching the name of your bot. Also ensure that your bot has been invited to the channel (see: step 4, line 1). An example of what you would want to see is:

```
{ id: 'U4HMFUWTD',
  team_id: 'T3X0XGQNR',
  name: 'memebot',
  deleted: false,
  status: null,
  color: '674b1b',
  real_name: 'Meme Bot',
  tz: null,
  tz_label: 'Pacific Daylight Time',
  tz_offset: -25200,
  profile:
    { bot_id: 'B4H2PGAGG',
      api_app_id: '',
      always_active: false,
      avatar_hash: 'fb4df999bd06a8f1377c_24.png',
      image_24: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_24.png',
      image_32: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_32.png',
      image_48: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_48.png',
      image_72: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_48.png',
      image_192: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_48.png',
      image_512: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_48.png',
      image_1024: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_48.png',
      image_original: 'https://avatars.slack-edge.com/2017-03-13/153730224931_fb4df999bd06a8f1377c_original.png',
      first_name: 'Meme',
      last_name: 'Bot',
      title: 'Nothing productive',
      real_name: 'Meme Bot',
      real_name_normalized: 'Meme Bot',
      fields: null },
    is_admin: false,
    is_owner: false,
    is_primary_owner: false,
    is_restricted: false,
    is_ultra_restricted: false,
    is_bot: true,
    presence: 'away' }
```

4. **TASKS:** *(I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step3.js)*
 - a. **TASK 3:** identify only the messages that are typed and remove all the clutter of the presence in the application
 - b. **TASK 4:** do a request to api.ai with the text to see what happens.

STEP 6: MAKING SURE YOUR BOT CAN SEE YOU (STEP3.JS)

1. After completing those tasks, jump into step3s code and see what has changed.
2. Execute step3s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node.step3.js
```

3. If everything looks correct and you are seeing the bot, then you should be able to type into the channel that you invited @memebot to earlier and see your messages appear in the console.
 - a. If you do not see these messages, double check @memebot is part of the channel by /invite @memebot
4. **TASKS:** *(I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step4.js)*
 - a. **TASK 5:** send the request to API.ai and get its response back

STEP 7: MAKING FIRST CONTACT! (STEP4.JS)

1. After completing those tasks, jump into step4s code and see what has changed.
2. Execute step4s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node.step4.js
```

3. If everything looks correct and you are seeing the bot, then you should be able to type into the channel you invited @memebot to earlier and see responses to your message in the console. Example responses are "What was that?", "I didn't get that. Can you say it again", etc.
 - a. If you are not seeing example responses like this, then there is likely something wrong with the api.ai connection, verify the token is correct in variables.js
4. **TASKS:** *(I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step5.js)*
 - a. **TASK 6:** we need to build a function using slackbots npm to respond in the correct channel
 - b. **TASK 7:** create function to take response_speech and send it back to the user

STEP 8: GETTING YOUR BOT TO TALK TO YOU! (STEP5.JS)

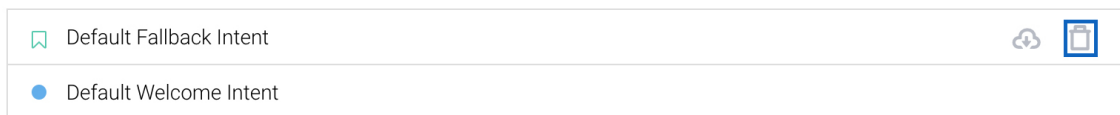
1. After completing those tasks, jump into step5s code and see what has changed.
2. Execute step5s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node.step5.js
```

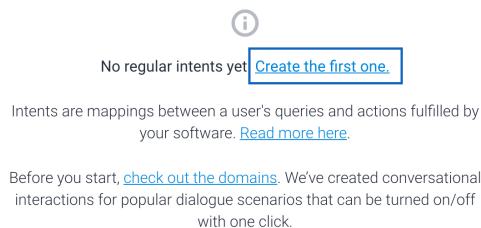
3. If everything looks correct and you are seeing the bot, then you should be able to type into the channel you invited @memebot to earlier and be able to converse back and forth. At this point, no matter what you type, you will get a response from the confused bot. This is good! This means that you are able to communicate back and forth with your bot.
4. **TASKS:** (I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step6.js)
 - a. **TASK 8:** make your agent more intelligent
 - b. **TASK 9:** build handlers to retain the information from intent to intent

STEP 9: SETTING UP SOME INTELLIGENCE WITH YOUR BOT! (TASK 8)

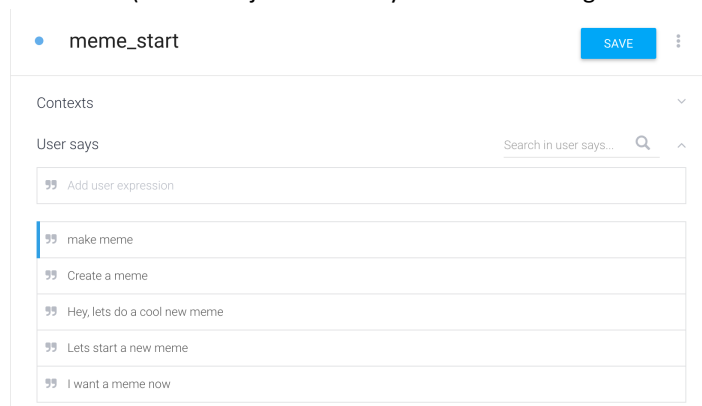
1. First, we'll need to make our AI a little bit more intelligent! We'll start with wiping all its knowledge (intents). Navigate to <https://console.api.ai/>
2. Delete the Default Fallback Intent & Default Welcome Intents from the interface by clicking on the trashcans to the right of the intents.




3. When they are all removed you should be prompted to create a new one. Click "Create the first one"



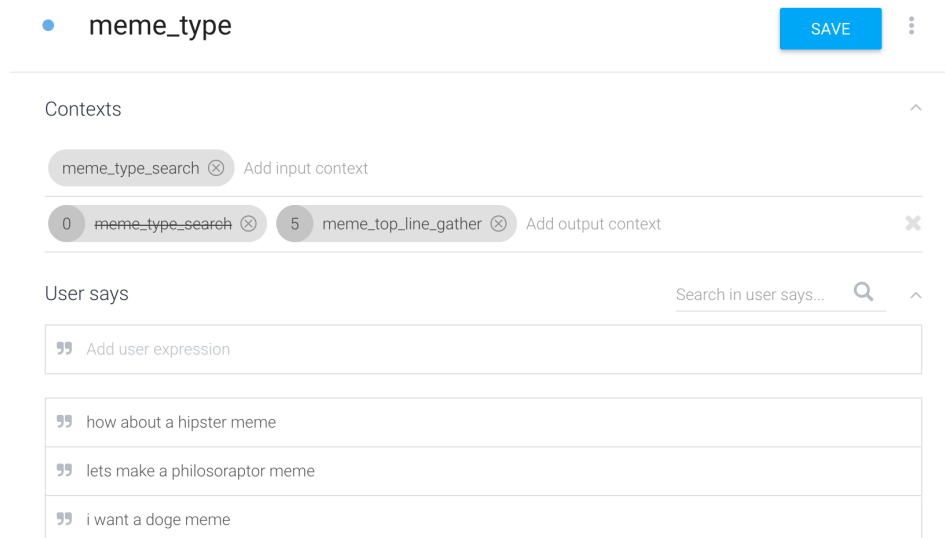
4. Upon creating a new intent you'll have a lot of blank information. Let's start with a basic one telling the bot we want to create a new meme. Visit <https://docs.api.ai/docs/concept-intents> for more information to learn on intents.
 - a. For the "Intent name" (at the top) name it "meme_start"
 - b. Where it has "User Says" fill out a few ways you would want to start a new meme. My examples are below: (These are just basic ways to start teaching the Machine Learning AI)




- c. At the bottom of the screen, where it says text response, put a response to know that we've hit the `meme_start` intent (I used: "Cool! Lets get started! What kind of meme?")
 - d. Save your meme by clicking the  button at the top of the screen.
5. Upon saving your meme, start back up `step5.js` (see above). Now say one of your phrases into slack while `step5.js` is running. (At this stage, the AI will start to match for this intent). For instance, if you used my examples it will also match on "can we make a meme"

STEP 10: TAKING THE INTELLIGENCE AND "CHAINING" INTENTS (TASK 8)

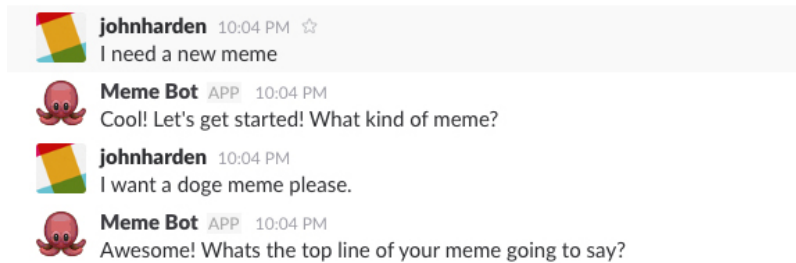
1. Now that we have an intelligent bot that can interpret our request to create a meme, let's create some actions using contexts. Visit <https://docs.api.ai/docs/concept-contexts> for more information to learn on concepts.
2. Every time an intent is triggered there are "input contexts" and "output contexts". For our example, dig back into `meme_start` and add an output context of "meme_type_search". To do this first, expand the "Contexts" at the top then add the output context.
3. We can now have a follow-up question after we've already established we want to start a meme. On the left, create a new intent by clicking the + sign.
 - a. For the "Intent name" (at the top) name it "meme_type"
 - b. Expand "Contexts" and add an "input context" of "meme_type_search". It **should** start auto populating if you saved from earlier. If it did not, return to step 10, line 2.
 - c. After you add the input context, it will automatically add itself to output. Click on the "5" and turn this into 0. This will remove this context once the "meme_type" intent is hit.
 - d. Add a new output context called "meme_top_line_gather".
 - e. Where it has "User Says" fill out a few different kind of memes. My examples are below: (These are just basic memes to start teaching the Machine Learning AI)



The screenshot shows the Google Assistant console interface for creating a new intent named "meme_type". At the top, there is a "SAVE" button. Below it, the "Contexts" section is expanded, showing two contexts: "meme_type_search" (input context) and "meme_top_line_gather" (output context). The "User says" section is also expanded, showing a list of user expressions: "how about a hipster meme", "lets make a philosoraptor meme", and "i want a doge meme".

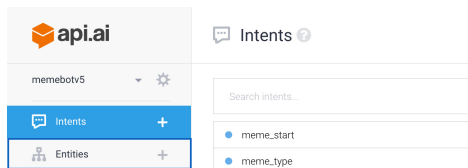
- f. At the bottom of the screen where it says text response, put a response to know that we've hit the `meme_type` intent (I used: "Awesome! Whats the top line of your meme going to say?")
 - g. Save your intent by clicking the  button at the top of the screen.
4. At this point, you should be able to go back into slack, start a conversation by saying "can we make a meme", then following up with "I want a doge meme". (make sure `step5.js` is running)

5. If everything is working, your slack should look like this:

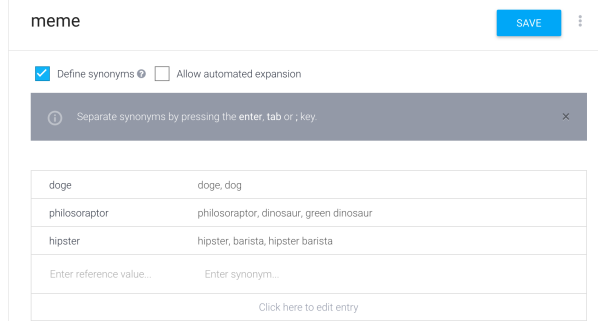


STEP 11: CREATING A MEME “ENTITY” (TASK 8)

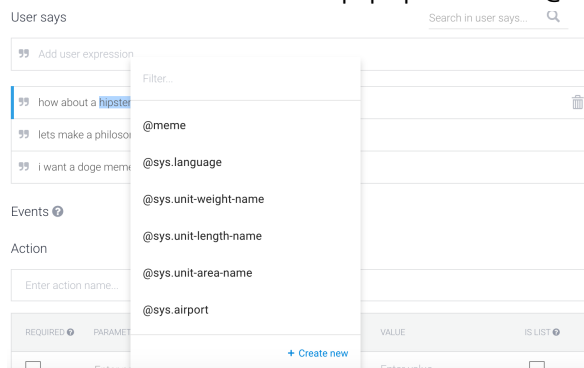
1. So we need to capture the kind of meme that we want in the node program. To do this, on the left side of the screen, click the plus sign next to entities: Visit <https://docs.api.ai/docs/concept-entities> to learn more on entities.



2. For memes we will need to enter all the types of memes that we would like our bot to support. Create the entities then add a few synonyms.
- For the “Entity name” call it meme.
 - For the rows, you will want to decide which memes you want to support and put their names and synonyms. My examples are below:



- Save your entity by clicking the **SAVE** button at the top of the screen.
3. Now we want to return back to our intents tab and dig back into our “meme_type” intent. In the section where it says “User says” you should have a few examples. Inside those examples, highlight the actual meme terms and a box should pop up. Select the @meme option (you may have to filter to find it)



4. Do this for each “User says” element that has a meme. Once this is completed it should look something like this:

” how about a hipster meme
” lets make a philosoraptor meme
” i want a doge meme

5. Excellent! Now we’ve actually taught the system what a meme is we can use it programmatically in the node script.
6. Let’s name our new variable `meme_type`, below where it says “Action” update the parameter name from `meme` to `meme_type`

Action

Enter action name...

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	<code>meme_type</code>	<code>@meme</code>	<code>\$meme_type</code>	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

STEP 12: FINISHING THE INTELLIGENCE (TASK 8)

1. Now that we have the meme type we'll want the **meme top line** and the **meme bottom line**.
2. For the **meme top line**, create a new intent:
 - a. For the "Intent name" (at the top) name it "meme_top_line"
 - b. Your input context will be "meme_top_line_gather", set the corresponding "meme_top_line_gather" output context from 5 lifespan to 0.
 - c. Add an additional output context of "meme_bottom_line_gather"
 - d. Where it has "User Says" add @sys:any and press enter. Highlight the text like you did when identifying the "memes" and filter for @sys:any. You will get prompted to confirm. Press "Confirm".
 - i. @sys:any will actually filter for any piece of information that is typed at this point. There is no need for multiple "User Says" lines here.
 - e. Below in the actions area, rename your parameter from "any" to "meme_top_line".
 - f. Add a text response for "Okay! What is your bottom line?"
 - g. The full screen you should like this: (Once it does click the save button.)

The screenshot shows the Rasa X interface for configuring the 'meme_top_line' intent. The interface is divided into several sections: Contexts, User says, Events, Action, and Response.

Contexts: The 'meme_top_line_gather' context is set as the input context. The output context is 'meme_bottom_line_gather' with a lifespan of 5.

User says: The user expression '@sys:any' is entered and highlighted.

Events: No events are currently configured.

Action: The action name is 'meme_top_line'. The parameter 'meme_top_line' is set to the entity '@sys:any' with the value '\$meme_top_line'.

Response: A text response is configured with two variants: 'Okay! What is your bottom line?' and 'Enter a text response variant...'.

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	meme_top_line	@sys:any	\$meme_top_line	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

3. For the **meme_bottom_line**, create a new intent:
 - a. For the “Intent name” (at the top) name it “meme_bottom_line”
 - b. Your input context will be “meme_bottom_line_gather”, set the corresponding “meme_bottom_line_gather” output context from 5 lifespan to 0.
 - c. Where it has “User Says” add @sys:any and press enter. Highlight the text like you did when identifying the “memes” and filter for @sys:any. You will get prompted to confirm. Press “Confirm”.
 - d. @sys:any will actually filter for any piece of information that is typed at this point. There is no need for multiple “User Says” lines here.
 - e. Below in the actions area, rename your parameter from “any” to “meme_bottom_line”.
 - f. Add a text response for “We are generating your meme now!”
 - g. The full screen you should like this: (Once it does click the save button.)

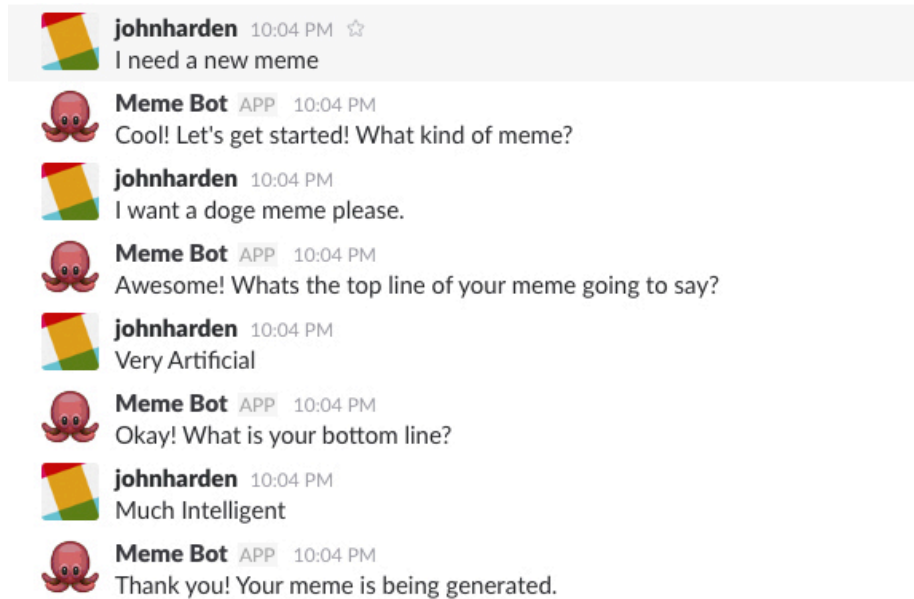
The screenshot shows the Rasa X interface for creating a new intent named "meme_bottom_line". The interface is divided into several sections:

- Intent Name:** "meme_bottom_line" (with a "SAVE" button and a menu icon).
- Contexts:**
 - Input Context:** "meme_bottom_line_gather" (with a "Add input context" button).
 - Output Context:** "0" (with a "Add output context" button).
- User says:**
 - Search in user says... (with a magnifying glass icon).
 - Buttons: "Add user expression" and "@sys:any" (highlighted in yellow).
- Events:** (with a question mark icon and a dropdown arrow).
- Action:**
 - Enter action name... (with a magnifying glass icon).
 - Table with parameters:
- Response:**
 - Text response (with a question mark icon and a trash icon).
 - 1 Thank you! Your meme is being generated.
 - 2 Enter a text response variant...

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	meme_bottom_line	@sys.any	\$meme_bottom_line	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

+ New parameter

- Now that both these intents are available, restart the step5.js script and your conversation should look something like this:



STEP 13: GENERATING YOUR MEME! (STEP6.JS)

- After completing those tasks, jump into step6s code and see what has changed.
- Execute step6s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node step6.js
```

- If everything looks correct and you are seeing the bot, then you should be able to type into the channel and have the above conversation. At the end of the conversation you should see

```
http://version1.api.memegenerator.net/Instance_Create?username={username}&password={password}&languageCode=en&generatorID=doge&text0=very artificial&text1=much intelligence

{ success: false, errorMessage: 'Input string was not in a correct format.' }
```

- The reason you are getting this error is because the memegenerator expects generatorID to be an actual ID and not the name of the meme.
- Return to the “Entities section”, dig into your meme entity. In the top right where the three dots are (next to save) click and dig down to “Enter raw mode”. Copy + Paste the following into the CSV mode:

```
"17","philosoraptor","green dinosaur","dinosaur"

"2454136","doge","dog"

"147426","hipster","barista","hipster barista"
```

6. These are the IDs associated to each memegenerator meme instance. Save then re-enter into editor mode.

- a. To get the IDs you can use the following cURL to search the IDs DB:

```
curl
"http://version1.api.memegenerator.net/Generators_Search?q=philosoraptor&pageIndex=0&pageSize=25"
```

- b. This will return JSON results in which you can find the generator ID. (For more information visit:

http://version1.api.memegenerator.net/#Instance_Create)

7. If you have the same conversation again you should see the API call in the console succeed.

```
http://version1.api.memegenerator.net/Instance_Create?username=test8&password=test8&languageCode=en&generatorID=24541366&text0=Very Artificial&text1=Much Intelligent
{ success: true,
  result:
    { generatorID: 2454136,
      displayName: 'DogeEEEE',
      urName: 'DogeEEEE',
      totalVotesScore: 0,
      imageID: 9872337,
      instanceID: 76089690,
      text0: 'Very Artificial',
      text1: 'Much Intelligent',
      commentsCount: 0,
      mgUserID: 14130400,
      username: 'test8',
      entityVotesSummary:
        { entityName: 'Instance',
          entityID: 76089690,
          totalVotesSum: 0,
          userID: null,
          userVoteScore: null },
      imageUrl: 'https://cdn.meme.am/cache/images/folder337/400x/9872337.jpg',
      instanceImageUrl: 'https://cdn.meme.am/cache/instances/folder690/400x/76089690.jpg',
      instanceUrl: 'https://memegenerator.net/instance/76089690' } }
```

8. **TASKS:** (I highly recommend trying these tasks before looking at the next steps code, the solutions for these tasks are in step7.js)

- a. **TASK 9:** post the image into the channel

STEP 13: GETTING YOUR MEME INTO SLACK (FINAL STEP!) (STEP7.JS)

1. After completing those tasks, jump into step6s code and see what has changed.
2. Execute step7s code with the following:

```
root@server:/~# cd /path/to/node/project/
root@server:/path/to/node/project/# node step7.js
```

3. If you have the above conversation the final slack conversation should look like this!



The screenshot shows a Slack conversation. At the top, a user named **johnharden** (with a profile picture of a colorful flag) sends a message at 10:04 PM: "I need a new meme". Below this, a bot named **Meme Bot** (with a profile picture of a red octopus) responds at 10:04 PM: "Cool! Let's get started! What kind of meme?". The user **johnharden** replies: "I want a doge meme please.". The bot responds: "Awesome! Whats the top line of your meme going to say?". The user replies: "Very Artificial". The bot responds: "Okay! What is your bottom line?". The user replies: "Much Intelligent". The bot responds: "Thank you! Your meme is being generated.".

Below the bot's last message, there is a section titled "Your meme sir." with a timestamp "Today at 10:04 PM (49KB)". It contains a meme image of a Shiba Inu dog. The text "VERY ARTIFICIAL" is overlaid at the top of the image, and "MUCH INTELLIGENT" is overlaid at the bottom. A small watermark "memegenerator.net" is visible at the bottom right of the image.

4. Congratulations! You've made your very own, artificially intelligent memebot with api.ai/node/slack!

STEP 14: CHALLENGING YOURSELF!

1. After completing this project, you should try and push your project even further, good examples of extra things to do are:
 - a. Add more memes:
 - i. Create more IDs in the entity list manually
 - ii. Utilize the memegenerators search API and api.ais Intent API to automatically populate many memes at once
 - b. Add an initial AI section that you can start a meme and already define the meme entity:
 - c. Create fallback intents that allow you to respond when an intent isn't met.
 - d. Create exit intents where you can stop during any process of the meme creation process.
 - e. Add size elements to your meme for exporting
 - f. Add the ability to create a meme in a "single line"
2. In the github is a memebotv3.zip file that you can upload as an agent that has a few of the above "extra features"

REFERENCE INFORMATION

API.ai docs : <https://docs.api.ai/docs>

Meme Generator API : <http://version1.api.memegenerator.net/>

Memebot Github : <https://github.com/johnharden/memebot>

Contact me at:

john@jjharden.com

Connect with me at LinkedIn:

<http://jjharden.com/>