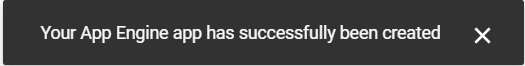
**Task 2. Initialize App Engine and Cloud Datastore**

1. On the **Navigation menu** (gcp-menu.png), click **App Engine > Dashboard.**
2. Under the Welcome dialog box, click **Create Application.**
3. Accept the default region, and then click **Create app**.
4. For **Language**, select **Python** from the dropdown. You can leave the default, "Standard", for **Environment**.
5. Click **Next** to initialize App Engine. It may take a few minutes to initialize. Look for the message box, "Your App Engine app has successfully been created", which may be near the bottom of your browser window. 

You have now successfully initialized App Engine and Cloud Datastore.

1. Click "x" to close the message box

**Task 3. Get started with Cloud Datalab**

This lab uses several Cloud Datalab notebooks, each of which has a primary function as described below:

**Pre-Processing Notebooks**

Let's first review the purpose of each of the notebooks. They are used to extract information from the HR Manual and create a knowledge base that is used by the chatbot to answer questions. Each of them will be run one time in the order listed here:

* **ProcessHandbook.ipynb** performs "semi-structured" analysis on the HR Manual text file. It alternately extracts topic "headings" and associated "policy text" from the file and stores these as key-value pairs in Cloud Datastore to give the chatbot a basic vocabulary.
* **ProcessSynonyms.ipynb** uses several Python data science libraries and the Cloud Natural Language API to generate synonyms for topics, which gives the chatbot an expanded vocabulary.
* **DialogFlow.ipynb** uses Dialogflow's "Entity" API to write the topics to Dialogflow's Entity module, which makes these words available to the chatbot as a data type.

**Set up Cloud Datalab**

To launch Cloud Datalab:

1. From the [Google Cloud Platform Console](https://console.cloud.google.com/), make sure your project is selected and click **Cloud Shell** (cloud-shell-icon.png) to start the shell.

Cloud Datalab is set up on a Compute Engine VM. For that, you need to specify the project and the zone where the VM is created. You'll use Cloud Shell as the client to run the installation commands.

1. In Cloud Shell, execute the following command:

gcloud config set core/project $GOOGLE\_CLOUD\_PROJECT

1. Run the following command to list zones:

gcloud compute zones list

Notice the output from running this command is in the format zone, region, then status (Zone | Region | Status).

The following are regions which currently support the functionality we want from our Datalab instance: **us-east1, us-central1, asia-east1, europe-west1**.

1. In preparation for the next step, identify which region is geographically closest to you and choose a zone from that region. For instance, if the region "us-west2" is closest to you, you might choose "us-west2-a" for the zone.

Next you'll create a Cloud Datalab instance (virtual machine) with the zone you chose and your project name.

1. Run the following command to create an instance and connection to a Datalab in the zone you selected:

datalab create mydatalabvm --zone <ZONE>

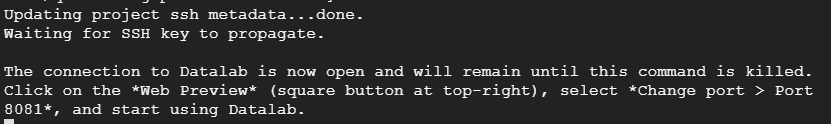
Replace <ZONE> with the zone name you picked from the previous step.

Datalab may take several minutes to complete.

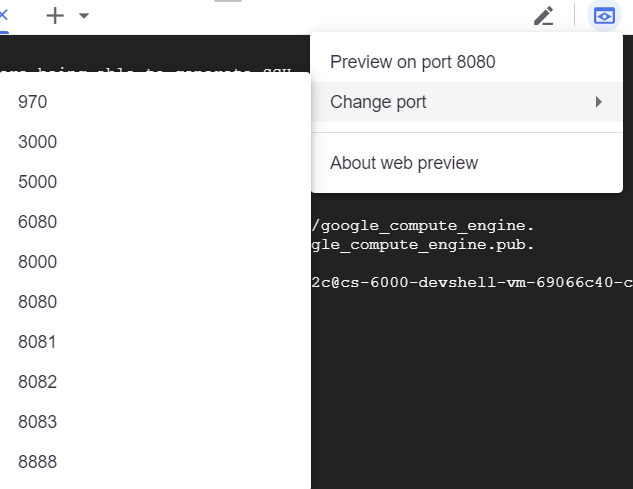
1. If the command prompts you to create SSH keys, type **Y**.
2. If the command prompts you for an RSA key password, press ENTER twice to confirm.

It could take several more minutes for Datalab to finish propagation of the SSH keys, etc. Before continuing, wait for the Datalab to finish launching. It's ready when you see the following:

**Result**



Note: The connection from the Cloud Shell to the Datalab instance can time out due to inactivity. If you lose connection, you may be able to restore it as follows. Find the instance name from the drop down menu **Compute Engine > VM Instances**, then enter the command: **datalab connect mydatalabvm**in your new Cloud Shell.

1. To use your new Datalab connection and open your browser to the Cloud Datalab notebook listing page, click **Web preview** (web-preview-icon.png).
2. Click **Change port** and select **8081**. 

Cloud Datalab should open in a new browser tab.

**Download and open lab notebooks**

1. To open a new notebook click **+Notebook**.
2. Paste this code into the notebook cell:

%bash

git clone https://github.com/GoogleCloudPlatform/training-data-analyst

rm -rf training-data-analyst/.git

1. Click **Run** or press **SHIFT + ENTER**.

**View the HR Manual sample text file**

1. From the Cloud Datalab console, return to the **datalab** home page (this is running in a separate browser tab).
2. Navigate to **training-data-analyst > courses > dialogflow-chatbot > notebooks > CAHRC\_HR\_Manual.txt** to view the HR Manual we will use as our "data source" to build the chatbot from. Scroll through and explore the contents of the file. If it doesn't appear yet, try refreshing the page.
3. You will notice that the file is organized in a semi-structured format, with topic headings followed by blocks of text describing each topic:

Employment Equity

[THE ORGANIZATION] is an equal opportunity employer and employs personnel without regard to race, ancestry, place of origin, colour, ethnic origin, language, citizenship, creed, religion, gender, sexual orientation, age, marital status, physical and/or mental handicap or financial ability. While remaining alert and sensitive to the issue of fair and equitable treatment for all, [THE ORGANIZATION] has a special concern with the participation and advancement of members of four designated groups that have traditionally been disadvantaged in employment: women, visible minorities, aboriginal peoples and persons with disabilities.

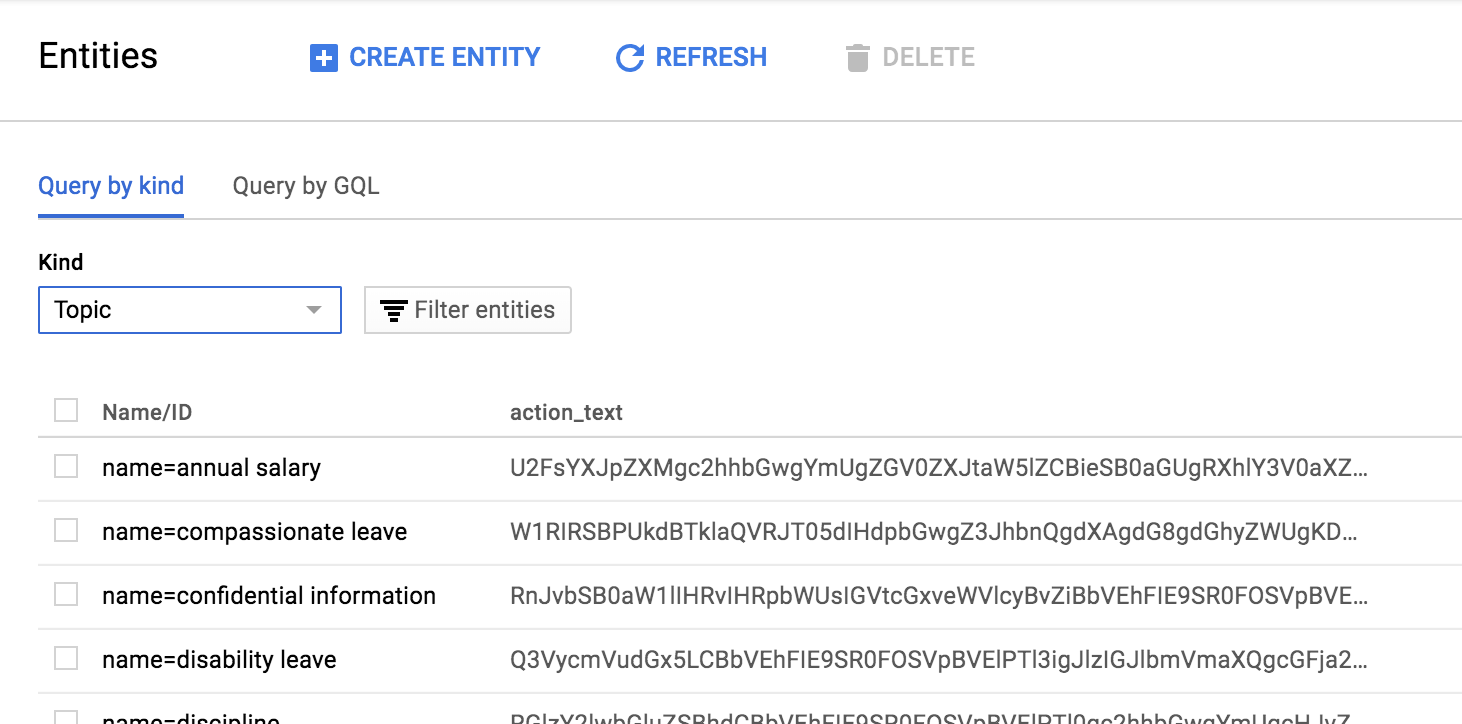
The first Python Notebook will use this "semi-structured format" to begin extraction of meaning from the document.

1. Close the browser tab and return to the main Cloud Datalab tab (running in a separate browser tab).

**Task 4. Open and execute the ProcessHandbook notebook**

This first Python Notebook extracts heading "topics" along with their associated content "action text" from the HR manual text file we just reviewed - and stores these as key-value pairs in Cloud Datastore. This notebook should only be run one time.

1. From the Cloud Datalab console, return to the **datalab** home page and select **training-data-analyst > courses > dialogflow-chatbot > notebooks > ProcessHandbook.ipynb**.
2. Click the dropdown arrow next to **Clear**, and then click **Clear all Cells**.
3. Execute the cells individually and observe the results. To move through the cells conveniently, press **SHIFT + ENTER** and wait for each cell to complete before continuing. Code cell completion is indicated by a blue bar to the left of the cell.
4. Once you have executed the last cell on the page, you are finished running this particular notebook and you have extracted the topics from the HR Manual document.
5. Now we will review the database you have created. Switch back to the GCP Console (this is probably the first tab in your browser), and on the **Navigation menu**, click **Datastore > Entities**.
6. Under **Kind**, select **Topic**. Review the topics that were created. Note that the **action\_text** column is automatically encoded by Cloud Datastore. When you query the data, it will automatically be decoded for you:



**Task 5. Open and execute the ProcessSynonyms notebook**

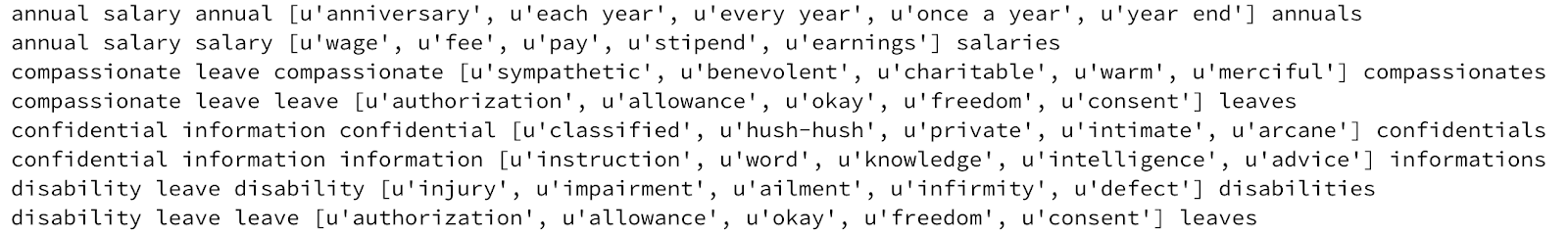
This second notebook uses Natural Language Processing (NLP) to create additional "synonyms" for the topics extracted in Task 1. These techniques are documented in the notebook itself.

1. From the Cloud Datalab console, return to the **datalab** home page and select **training-data-analyst > courses > dialogflow-chatbot > notebooks > ProcessSynonyms.ipynb**.
2. Click the dropdown arrow next to **Clear**, and then click **Clear all Cells**.
3. Execute the cells individually and observe the results.

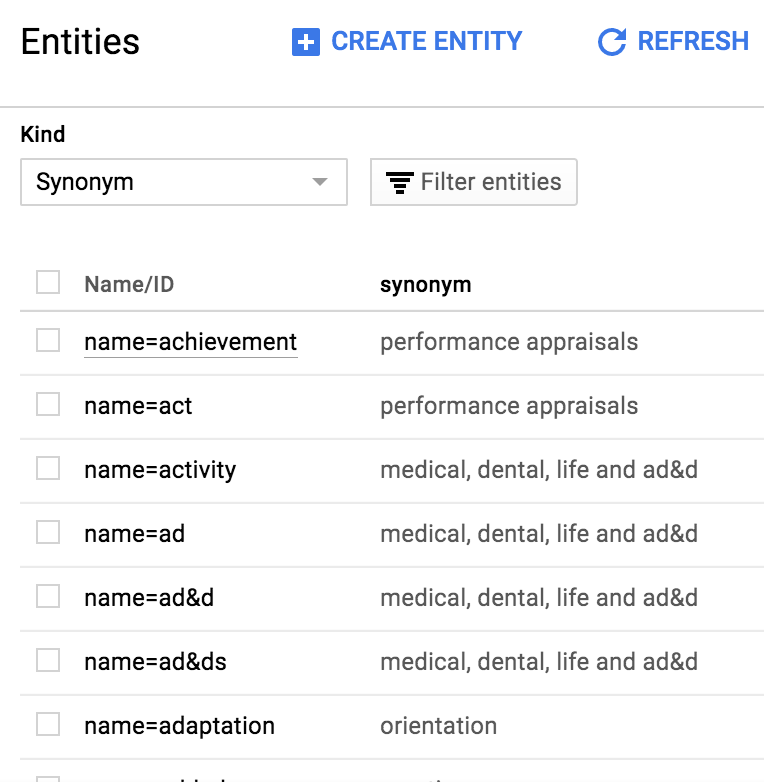
To move through the cells conveniently, press **SHIFT + ENTER** and wait for each cell to complete before continuing. Code cell completion is indicated by a blue bar to the left of the cell.

This notebook uses the Google Cloud Natural Language API. If you haven't used this API in this project before, [enable it](https://support.google.com/cloud/answer/6158841?hl=en) before running the cells.

1. A convenient way to progress through the cells is **Shift + Enter** waiting for each cell to complete before progressing. Code cell completion is indicated by a blue bar on the left of the cell and any output printed below.



1. In the Cloud console, navigate to **Storage > Datastore > Entities**
2. Under **Kind** select **Synonym**. It may take several minutes for this data to appear in the web console. Try refreshing the page after waiting a minute or two if the data doesn't show up right away.
3. Review the synonyms that were created. Note that there are multiple synonyms for each topic you created in the previous notebook. The synonyms are listed under the Name/ID property, and the topics they refer to are listed under the synonym property.



The knowledge base for your chatbot is now complete! A variety of semi-structured and unstructured analysis techniques were used to extract knowledge for the HR manual in text format.

1. Close the Datalab tab(s).

Task 6. Login to Dialogflow

Task 7. Create a Dialogflow chatbot (Agent)

Task 8. Create a "Topic" entity

An entity is essentially a data type in Dialogflow that you can use to parameterize conversations. In previous labs, we started with Intents. Here, we do it a little differently since we already have a knowledge base. You create an entity called "Topic" that encapsulates all possible HR topics this chatbot can discuss.

1. On the main Dialogflow menu, click **Entities**.
2. Click **Create Entity**.
3. For **Entity name**, type **Topic**.
4. Select **Allow automated expansion**.

This allows your chatbot to recognize Topic values that are not explicitly listed in your data model.

1. Clear / uncheck **Define synonyms**. Your webhook handles synonyms instead.
2. Click **Enter value** and type **test**.

You add more values for Topic in an automated fashion in the next task, but Dialogflow doesn't allow you to save the entity without at least a single value.

1. Click **Save**.

## Task 9. Import topic entities from Cloud Datastore to Dialogflow

This third notebook imports topic entries from Cloud Datastore to Dialogflow.

1. In the Cloud Datalab console, return to the **datalab** home page and select **training-data-analyst > courses > dialogflow-chatbot > notebooks > DialogFlow.ipynb**.
2. Click the dropdown arrow next to **Clear**, and then click **Clear all Cells**.
3. Execute the first cell to install the Dialogflow SDK on Cloud Datalab (via updated pip package installer).
4. You must restart Python on your notebook server. Click the dropdown next to the **Reset** button in the Cloud Datalab menu bar, and click **Interrupt Execution**. Then click the **Reset** button and click **Restart**.
5. Execute the remaining cells in the notebook.

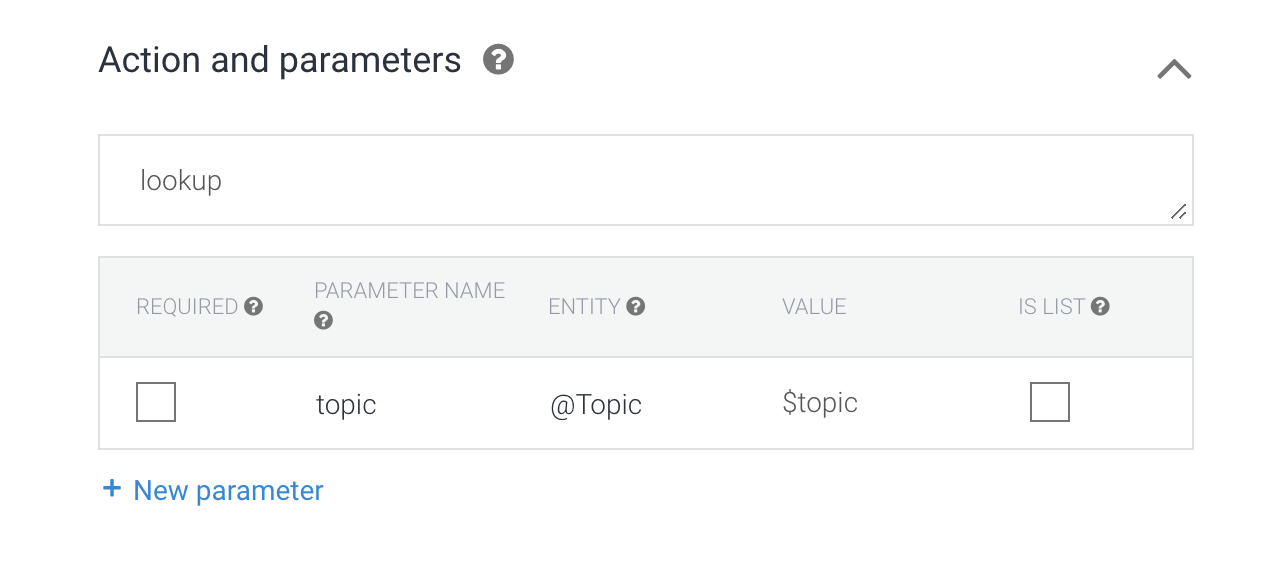
These make API calls to Dialogflow to upload your topics. A list of topics should appear in the notebook output.

1. Return to the Cloud Datastore console and open the **Entities** tab from the left-hand menu.
2. Click **Topic**. If you are already on the Topic page, you may need to refresh your browser.

Your entries from Cloud Datastore now populate the Topic entity. Your API calls populated DialogFlow with the topics you extracted from the document!

**Task 10. Create and train your intent**

1. On the Dialogflow menu, click **Intents**. You only need one intent for your HR chatbot. This intent will respond to requests for information about different HR topics.
2. Click **Create Intent**.
3. For **Intent name**,type **Topic**.
4. Click **Add Parameters and Action**.
5. For **Enter action name**, type **lookup**.
6. In the Parameters table, for **Parameter Name** type **topic**. For **Entity** type **@Topic**. And for **value**, type **$topic**. This creates a "lookup" action that passes a "topic" parameter to your backend process (webhook), which retrieves information on this topic from the HR Manual.

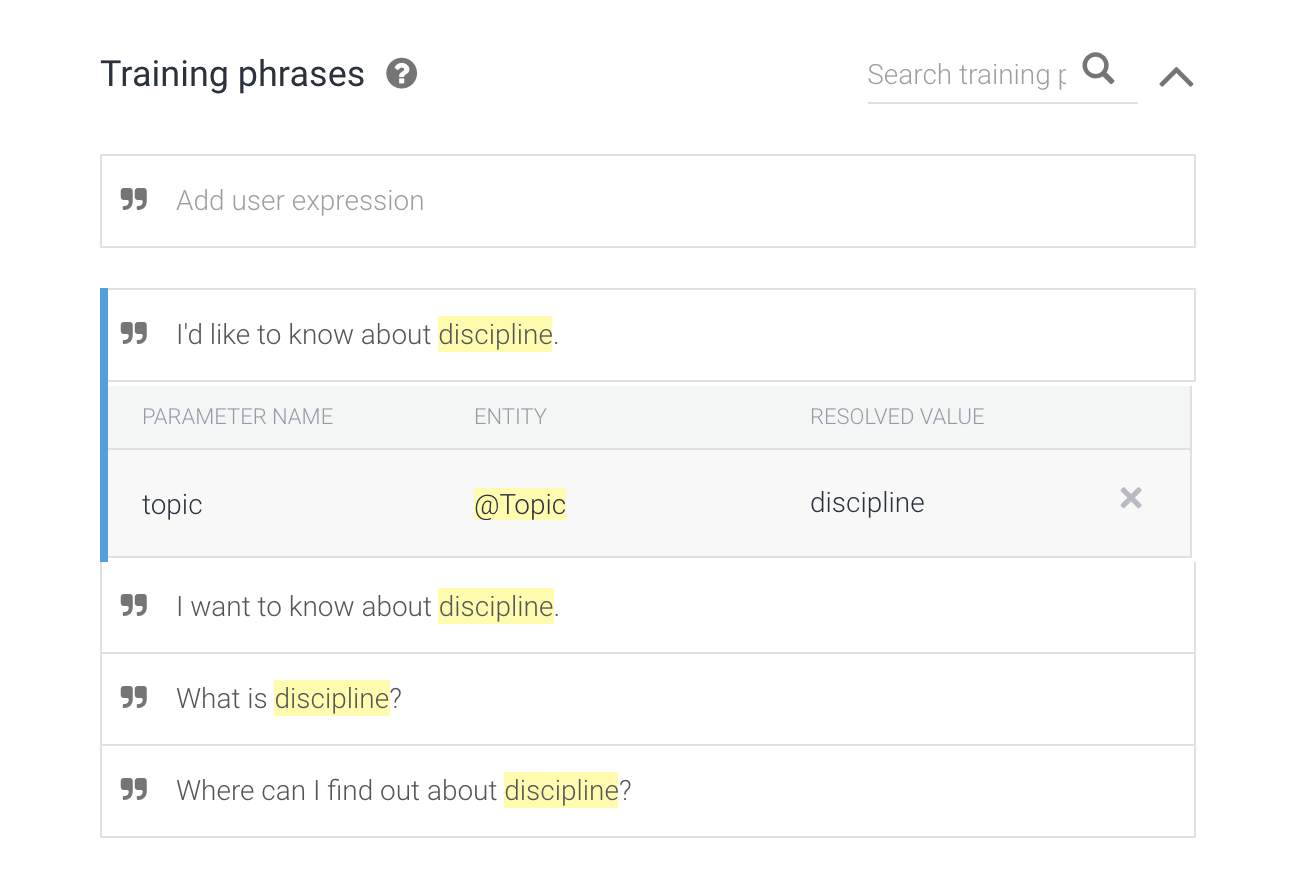


1. Click to expand the **Fulfillment** section, and then click on **Enable Fulfillment**.
2. Toggle enable the **Enable webhook call for this intent** option. Leave the **Enable webhook call for slot filling** option off.

If you don't see the webhook option in the Dialogflow menu, Click to expand the **Fulfillment** section to see the options for Enabling Webhook call.

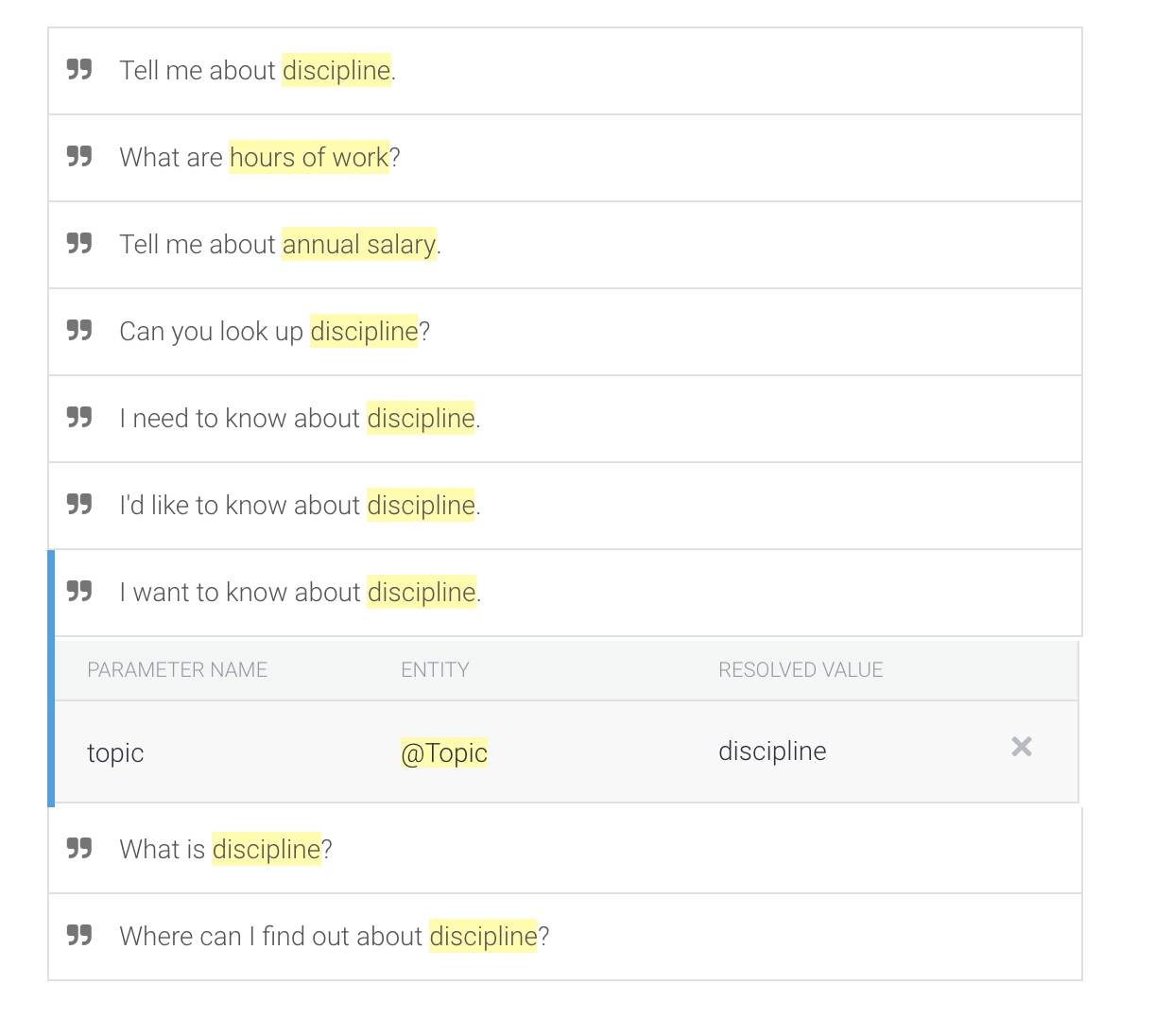
1. Scroll to the top of the page and click **Save**. You will now train your chatbot to carry on a conversation.
2. Under the **Training Phrases** section and click **Add Training Phrases.** Type in some sample sentences as shown below. Before pressing ENTER for each sample, highlight the word "discipline" (do not highlight punctuation) with your mouse to trigger a popup menu.

**Typing in sample phrases**



**Highlighting "discipline" to trigger popup menu**

1. Select **@Topic:topic** from the menu. This tells Dialogflow where in your example sentence to find your topic parameter. After specifying the topic parameter in your sentence, remember to press **ENTER** to add the sample sentence.
2. You can then clear out the **Add user expression** textbox and continue to add more examples. About 10 is a good number of examples; see suggested examples below. *(Be sure to include punctuation in your training examples as illustrated)*:



1. Click **Save**. Dialogflow will now train its agent based on your example intentions.
2. When Dialogflow indicates that training is complete (the settings gear stops spinning or a notification message is displayed), you are finished training your chatbot.

So far you have created an intent and an entity to match HR topics from our knowledge base. You have indicated in the agent's configuration that the agent's response for this intent will be fulfilled with a webhook, so in the next lab you will setup fulfillment to do a lookup from your knowledge base in Datastore. Since you don't want to repeat the steps to setup your agent's intent and entity in the next lab, the next task shows you how to export out your agent, so you can import it later.

END