IBM Watson Discovery – Node.js

Cognitive Solutions Application Development

IBM Global Business Partners
Duration: 90 minutes
Updated: Oct 25, 2019
Klaus-Peter Schlotter
kps@de.ibm.com

Version 5

Watson Library V5

Table of Contents

Overview

The IBM Watson Developer Cloud (WDC) offers a variety of services for developing cognitive applications. Each Watson service provides a Representational State Transfer (REST) Application Programming Interface (API) for interacting with the service. Some services, such as the *Speech to Text* service, provide additional interfaces.

The Watson Discovery service adds a cognitive search and content analytics engine to applications to identify patterns, trends and actionable insights that drive better decision-making. Securely unify structured and unstructured data with pre-enriched content, and use a simplified query language to eliminate the need for manual filtering of results.

The app created in this lab can run locally or in the IBM Cloud.

- In a first step, a Watson Discovery service instance is created on the IBM Cloud.
- Next, you will adjust the IBM Cloud demo to use the German document collections
- Then the chatbot from lab 3 will be enhanced with Discovery capabilities.

Objectives

- Learn how to use the Watson Discovery service on IBM Cloud
- Learn how to utilize the Watson Discovery service APIs in Node.js.

Prerequisites

Before you start the exercises in this guide, you will need to complete the following prerequisite tasks (see the setup guide).

- Create a IBM Cloud account
- Install the required software

Section 1: Create a Discovery service instance

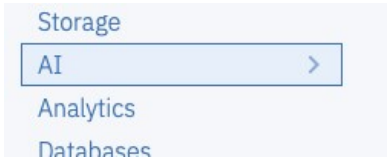
Step 1 In a web browser, navigate to the following URL

https://cloud.ibm.com

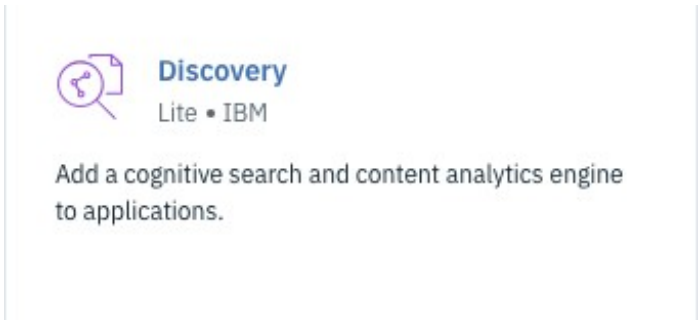
Step 2 Log in with your IBM Cloud credentials. This should be your IBMid.

Step 3 You should start on your dashboard which shows a list of your applications and services. Scroll down to the All Services section and click **Create Resource**.

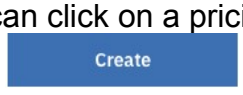
Step 4 On the left, under Services, **click** on *AI* to filter the list and only show cognitive services.



Step 5 **Click** on the Watson Discovery service.



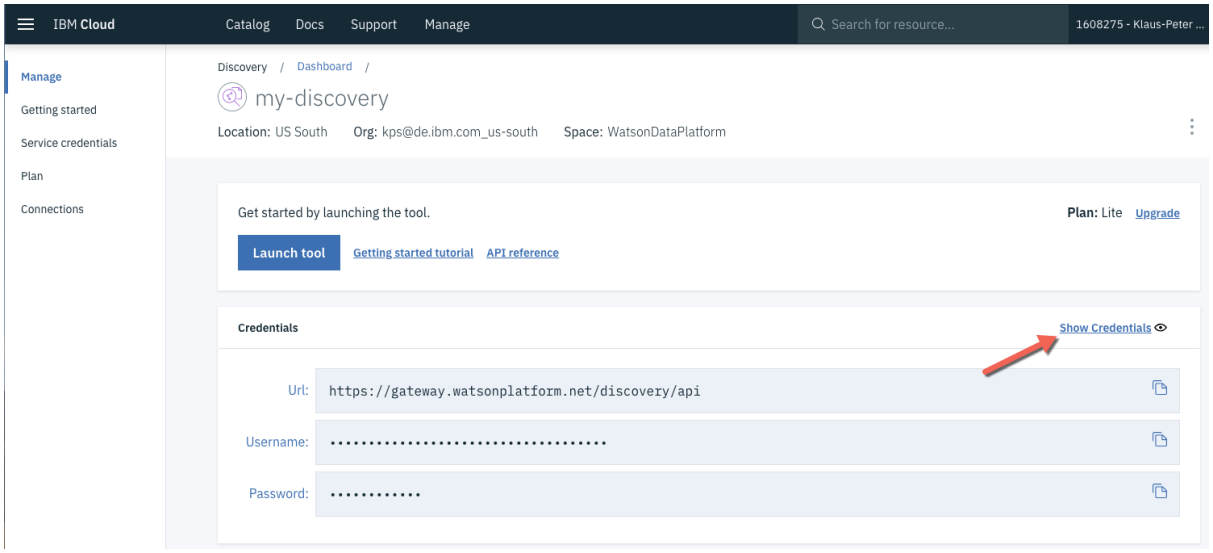
Step 6 Review the details for this service. At the top, there will be a description of the service. At the bottom, you can review the pricing plans. The Lite plan for this service provides no cost monthly allowances for workspaces, intents, and API calls. Enjoy your demo!

Step 7 At the top, you can enter information for your new service. In the middle, you can click on a pricing plan to select it. Fill out the fields as follows, then **click** 

at the bottom.

Field	Value
Service name	my-discovery
Selected Plan	Lite

Step 8 IBM Cloud has created a new service instance.



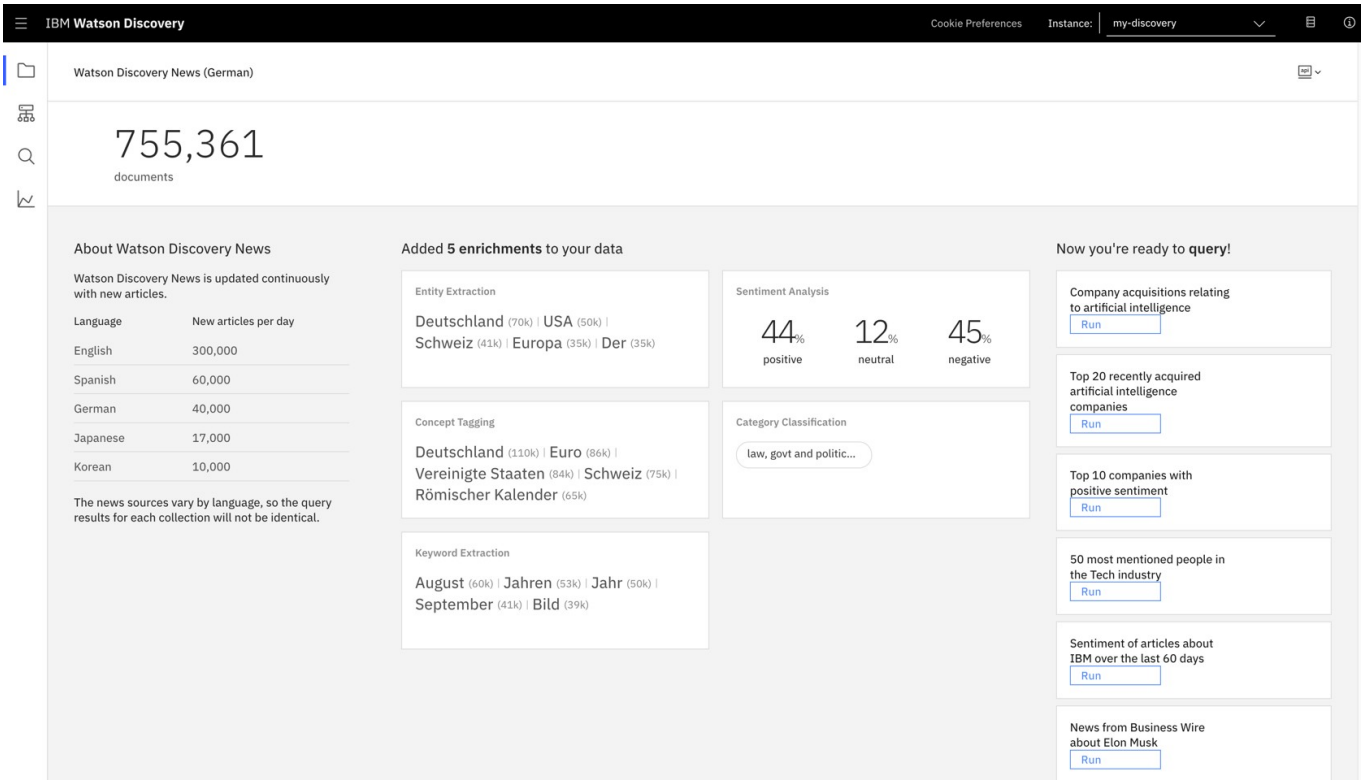
Step 9 In the *Credentials* section **click** Show Credentials . You should see the user-name and password for your service. Later in this exercise, you will enter these values into a JSON configuration file for your Node.js application. Feel free to copy them to your clipboard, to a text file, or just return to this section of the IBM Cloud web interface when the credentials are needed.

Step 10 **Click** the Launch tool button to open the Discovery service configuration tool.

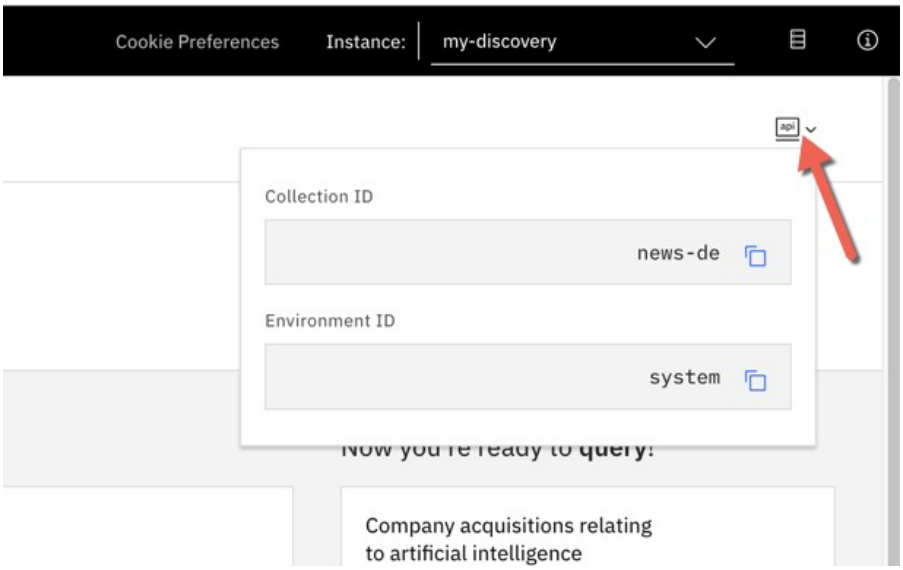
Step 11 **Select** your language and **click** on the *Watson Discovery News* collection, which is available in every service instance, to open it.

This collection is provided by IBM and gets updated daily, in German with approximately 40000 news articles per day containing a 60 day period of news (~ 3 Million entries).

On the overview page of the document collection you can see infos such as *Document count*, *Collection Info*, and *Insights from enriched data*. This enriched data is added by Watson when the documents get ingested.

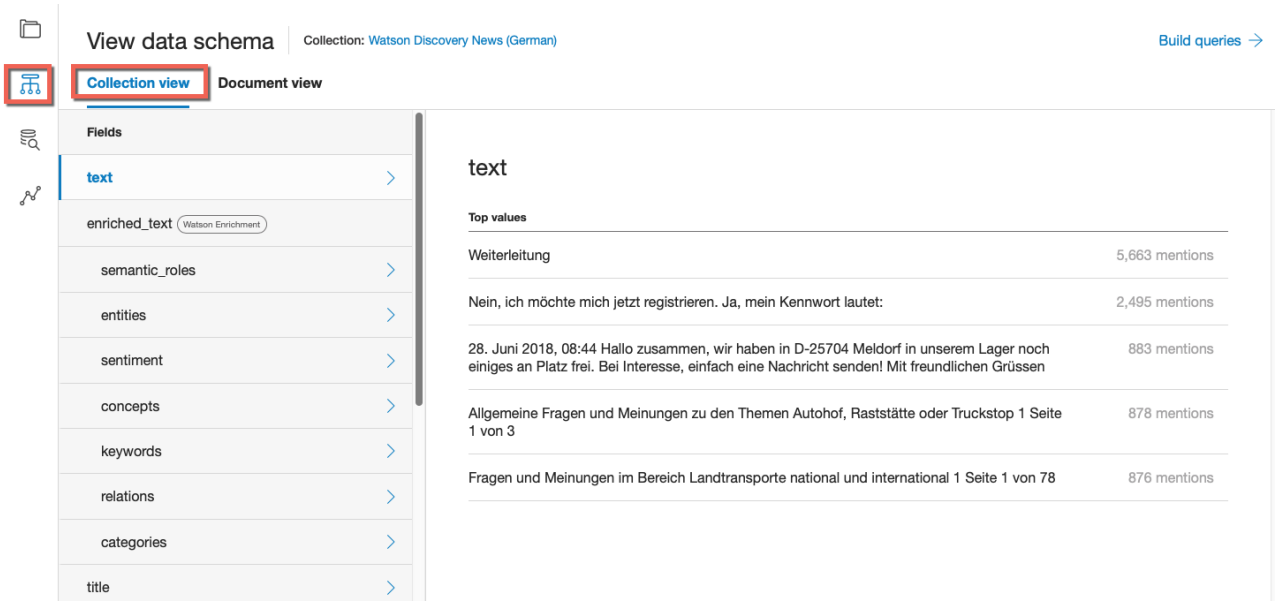


The *Use this collection in API* information is needed to access this information from your application, *Collection Id* and *Environment Id*.

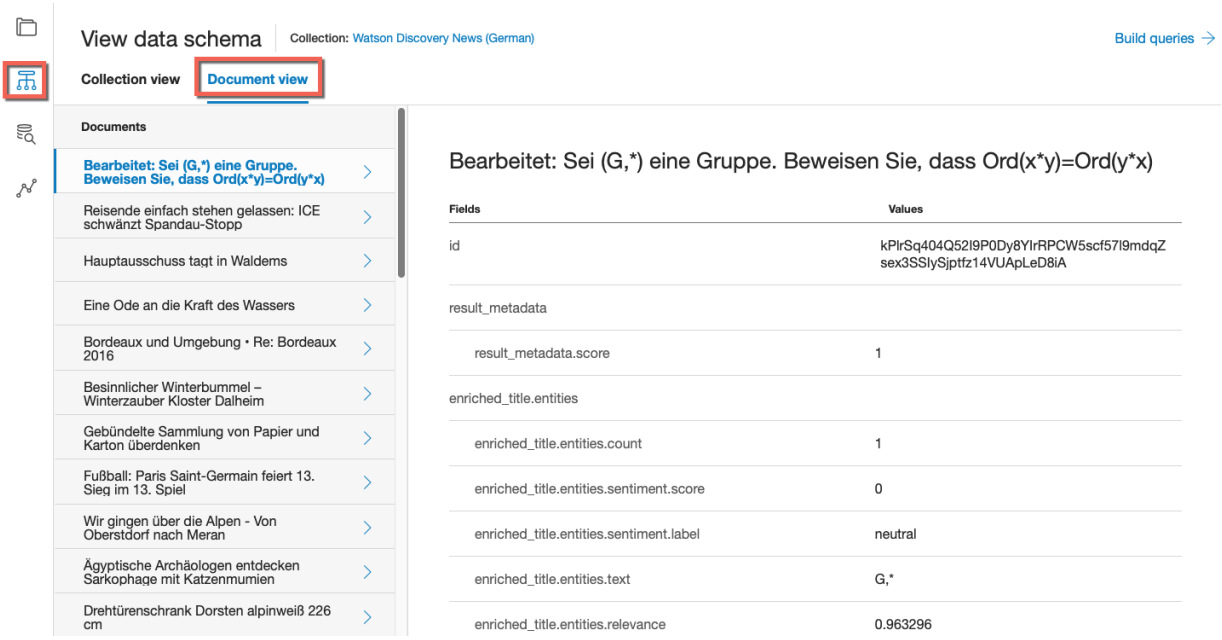


Step 12 On the *View data schema* page you can look at your collection in a

a) *Collections View* which shows you a summary of the Watson Enrichments on the collection scope

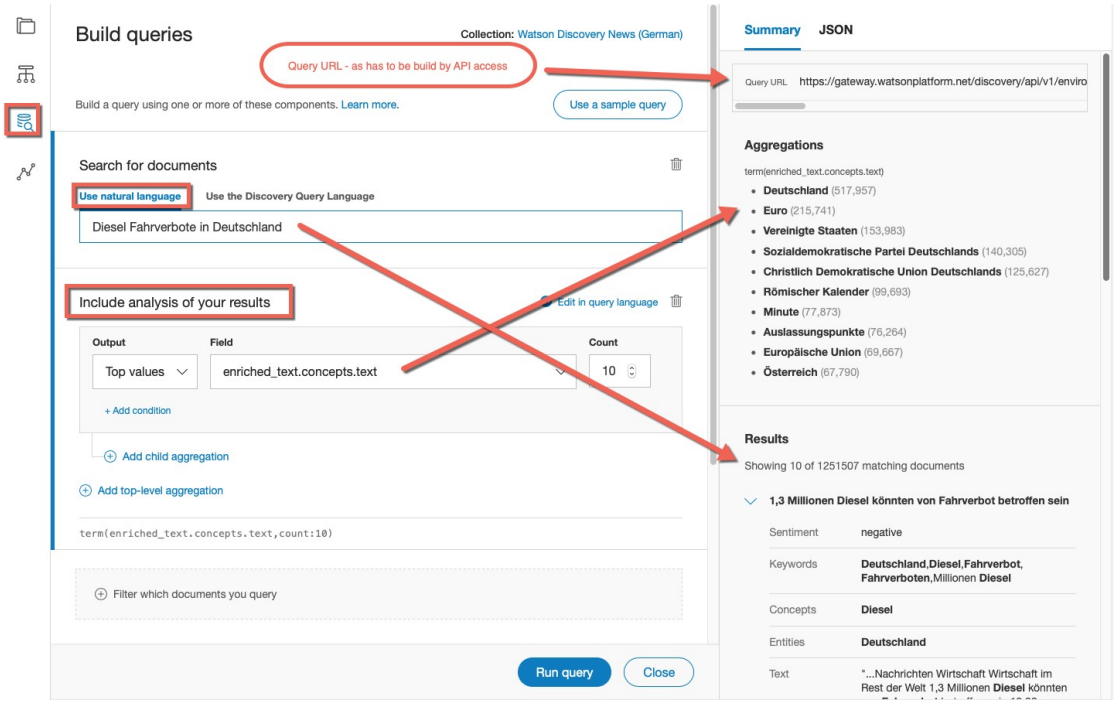


b) *Document View* page which shows you Watson Enrichments on a document level.

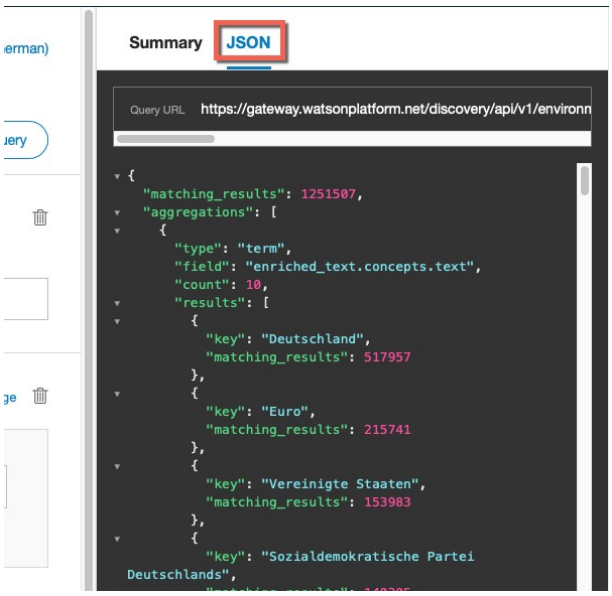


See the documentation of the service for more information.

Step 13 On the *Build queries* page you can search your collection f.e. *Use natural language* and include an analysis of the Watson Enrichment.



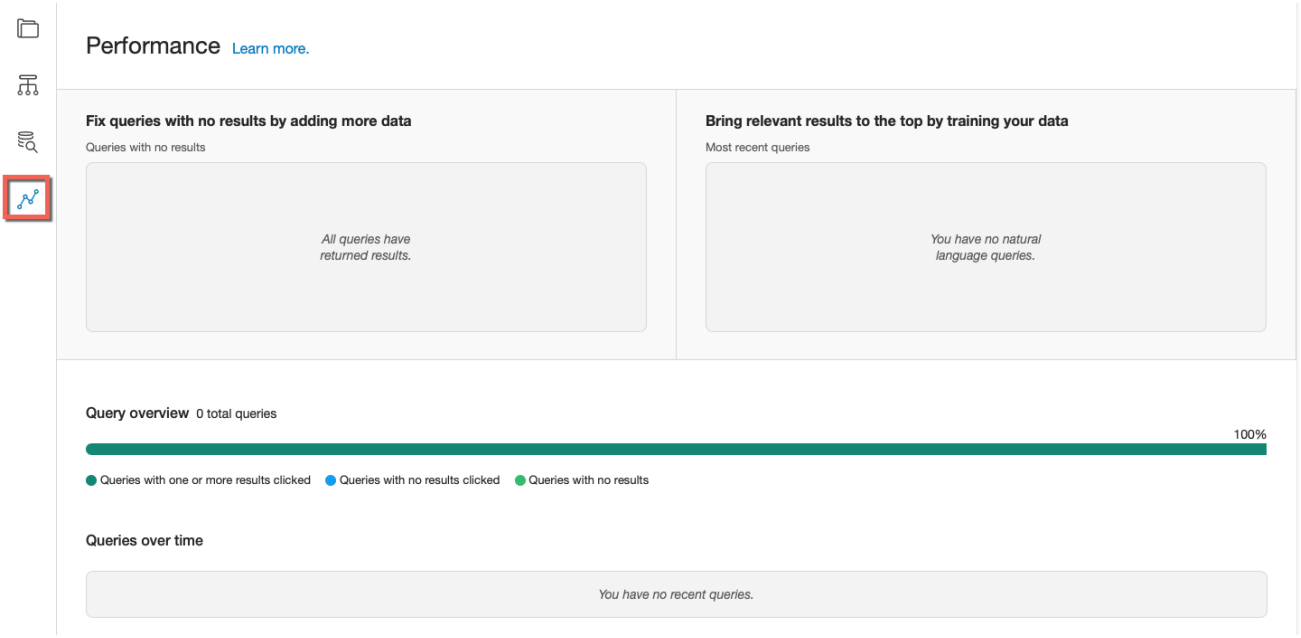
Step 14 Click the JSON tab at the top right then you see the result that you would get when you call this from the API.



Here is a nice video of the tutorial listed at the end of this document on how to work with the Discovery enriched-data to build powerful queries using this IBM News collection.
https://www.youtube.com/watch?v=N-HaIpPGde0&feature=youtu.be

More information can be found in the documentation.

Step 15 On the *Performance* page you can see how your collection performs over time. Right now, there is no data available.



Section 2 Use the German News collection with the Demo App (OPTIONAL)

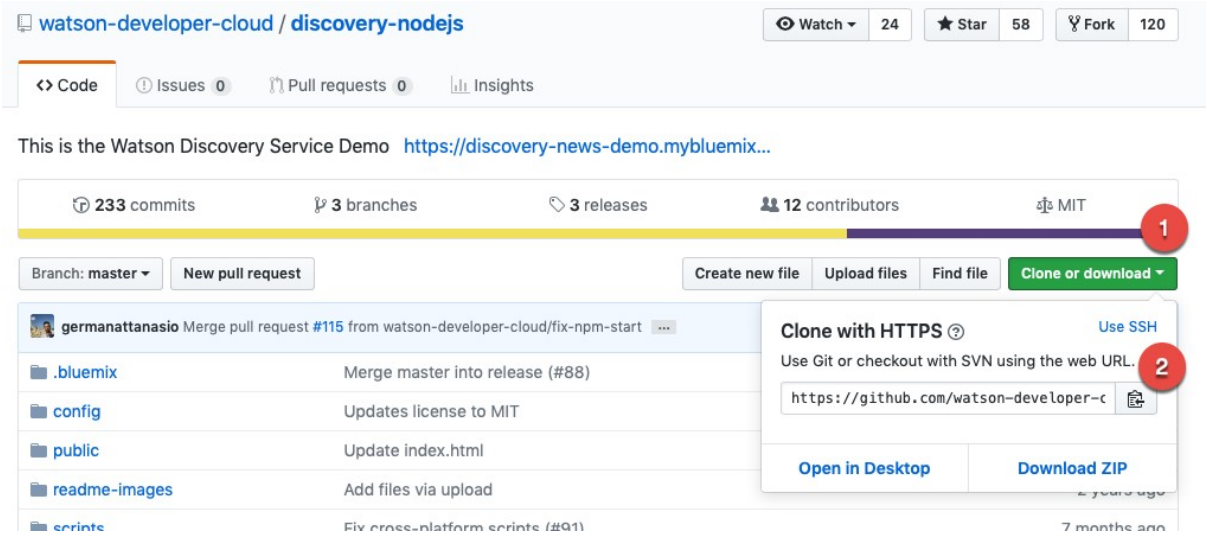
The Watson Discovery Demo can be found here. It is an English web application that uses the IBM Discovery News collection in the English version. In the following we want to switch this collection to German.

Step 16 In a web browser open the URL

https://discovery-news-demo.ng.bluemix.net

Step 17 On the top of the page **click** the *Fork on GitHub* link.

Step 18 On GitHub **copy** the clone URL to the clipboard.



Step 19 In a terminal window in a folder where you want to store your projects type

git clone <paste the url here>

f.e. `git clone https://github.com/watson-developer-cloud/discovery-nodejs.git`

Step 20 **Move** into the new folder created by the clone command. It should be named *discovery-nodejs*.

Step 21 Copy the .env.example and create a file called .env. **Type**

```
cp .env.example .env
```

Step 22 Open the project in Visual Studio code or with your preferred IDE.

Step 23 In the newly created .env file fill in your credentials (Username/password or APIKEY; Step 9) and also *Environment Id* and the Collection Id (Step 11). See the readme of the GitHub page for username/password and APIKEY.

Step 24 **Open** the file *app.js* in the project's root folder. On line 3 you can leave the NEWS_ENVIRONMENT_ID as "system" and adjust line 4 NEWS_COLLECTION_ID to "news-de".

```
const NEWS_ENVIRONMENT_ID = 'system';
const NEWS_COLLECTION_ID = 'news-de';
```

Step 25 **Save** and **close** the *app.js*.

Step 26 **Open** the file *query-builder.js*.

Step 27 **Change** the language in line 24 to

```
params.filter = `${fields.language}:(de)`;
```

Step 28 Save and close the file query-builder.js.

Step 29 In the terminal type to following commands to build and start the application.

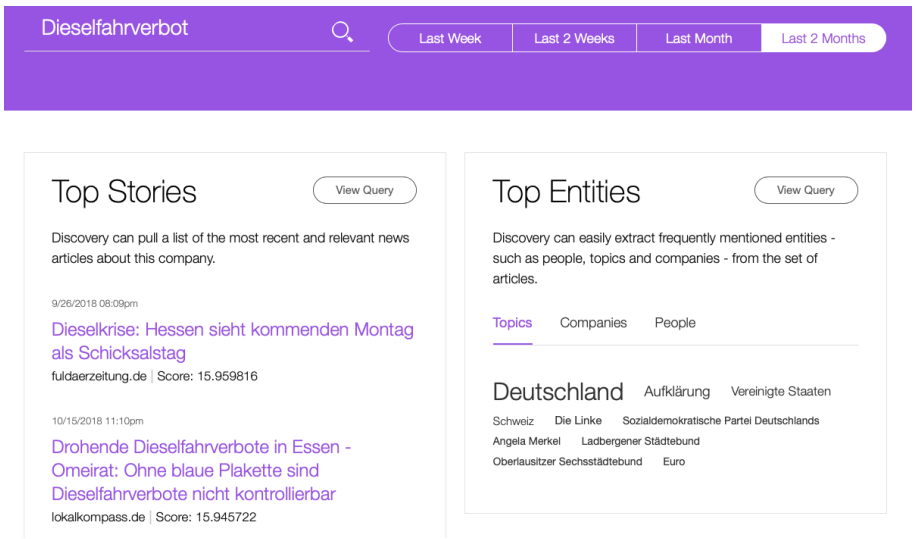
```
npm install
```

```
npm start
```

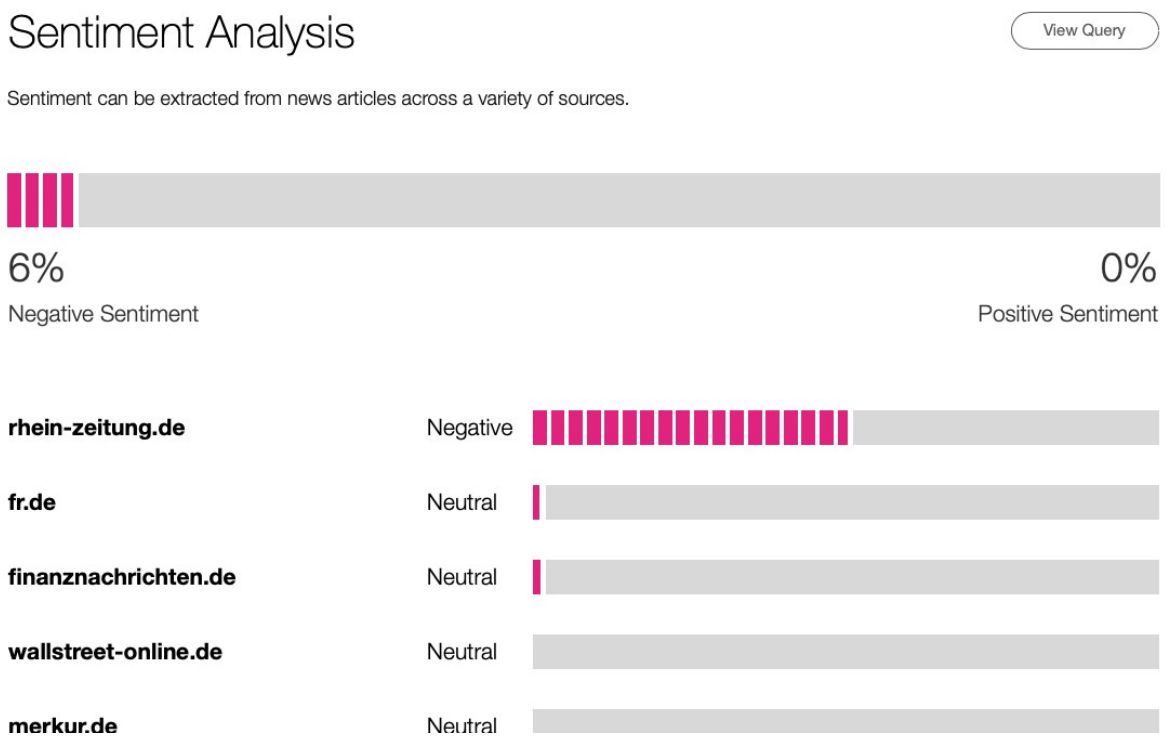
Step 30 In the web browser open

http://localhost:3000

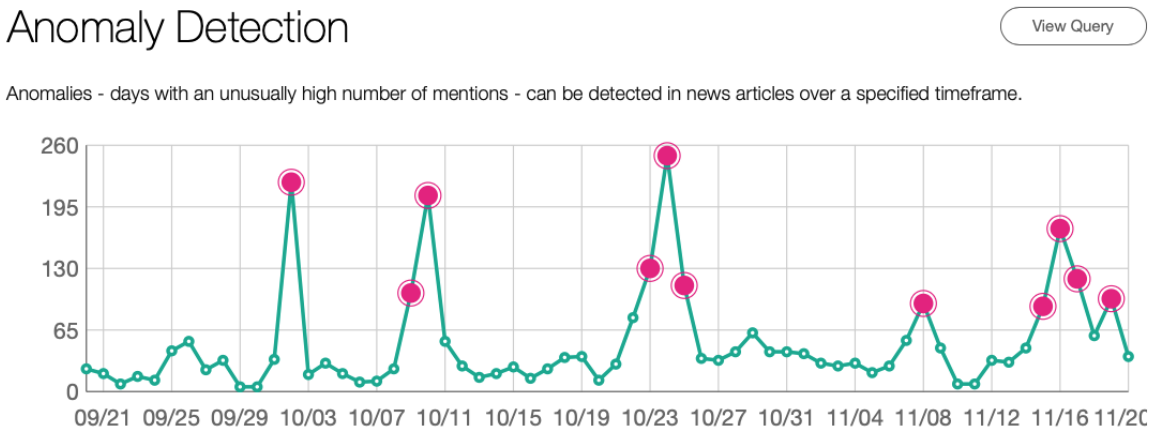
The web application that opens is still in English but now you can search for German text. As of November 2018 "Dieselfahrverbot" might be an intersting topic.



For example you can also see Sentiment analysis



and Anomaly Detection



You can use this constantly updated data in your applications.

In the next section we will enhance the Chatbot from Lab 3 with a custom document collection from the Discovery service.

Section 3 Integrate Discovery into a Chatbot (API)

In this section we will integrate the Discovery Service into the Chatbot created in Lab 3.

If you have not done **Lab3** and you want to execute all the steps here, just clone the git branch of lab3 with the following command:

```
git clone -b lab3_csadConversation https://github.com/iic-dach/csadConversation.git
```

If you do not want to do all the coding steps described in the following, you can clone the completed repository from GitHub. Then just add your credentials in *config.js*.

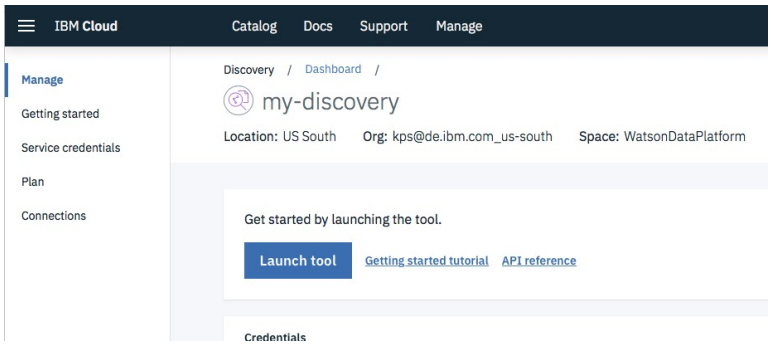
```
git clone https://github.com/iic-dach/csadConversation.git
```

Step 31 As a starting point for this section you should follow Lab 3 Section 1 and 3. These section will list shortcuts on how to get started quickly such as importing the Conversation workspace or cloning the app source code.

Create a Watson Discovery Document Collection

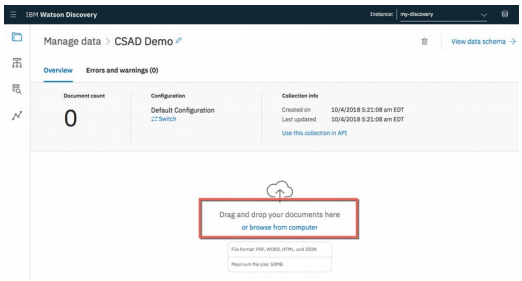
Step 32 Download the zip file manualdocs.zip from github (**click** the Download button) and extract it into a folder. It's a web manual for a Ford car.

Step 33 In your cloud console dashboard **click** on the "my-discovery" service created above and **click** the Launch tool button.

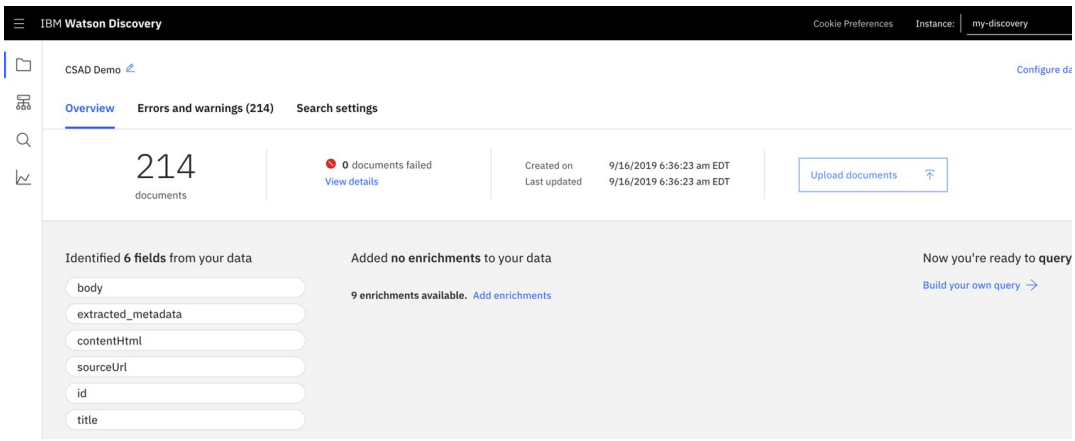


Step 34 On the *Manage data* page **click** Upload your own data .
As collection Name **enter** *CSAD Demo* and click Create.

Step 35 The new collection opens and you can either **browse** or **drag 'n' drop** all the documents extracted above (234 json files).

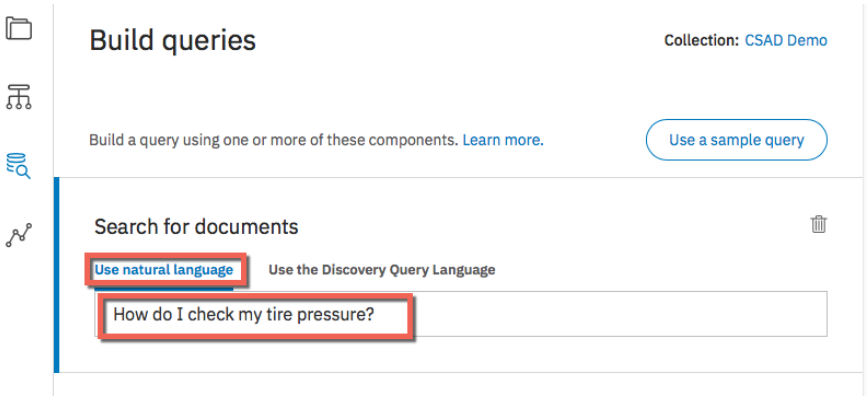


Step 36 The Discovery Service now indexes the documents. This may take a few minutes.



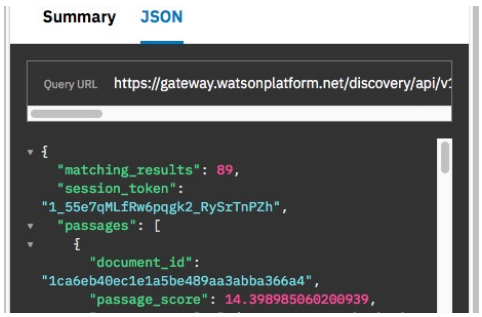
Step 37 On the left, click the button to open the *Build queries* page.

Step 38 **Enter** the following query: `How do I check my tire pressure?`



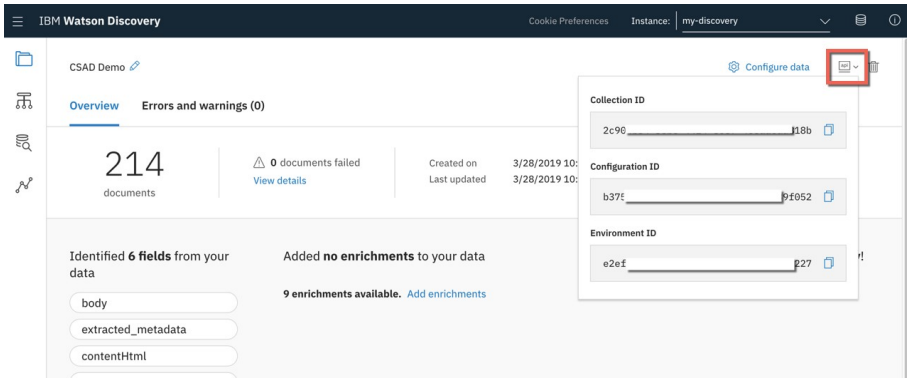
and press the Run query button.

On the left you can then see the results. Especially switch between the Summary and the JSON view of the results.



There is a lot more to explore here, but this is out of scope for this short tutorial. We just want to use the question submitted above later in our extended chatbot.

Step 39 On the top right of the *Overview* page **click**  to show the ids you have to copy for later use by the application in addition to the service credentials.



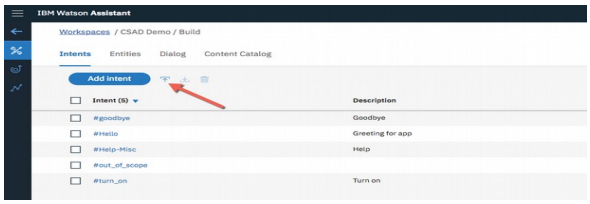
Update the Assistant Skill to integrate the Discovery Collection

Step 40 Download the out_of_scope_intents.csv from the Github repository to your project folder. Use the following in the terminal:

```
wget https://github.com/iic-dach/csadConversation/blob/master/resources/
Out_of_scope_intents.csv
```

Step 41 Go back to your Watson Assistant console that should still be open in the browser (Step 11)

Create a new intent *out_of_scope* and return to the list of intents. C**lick** the  button to import the user examples from the file downloaded above.



131 examples will be imported.

Step 42 On the *Dialog* tab **click** the menu ⋮ button on the *Turn on* node and **click** *Add node below*. **Name it** *Out of Scope*. In the field *If bot recognizes* **type** *#out_of_scope*.

> **Note:** Due to modularity we create the node in this location. Because this is also some kind of help question, it might be also appropriate, to insert this in the child tree of the *help* node.

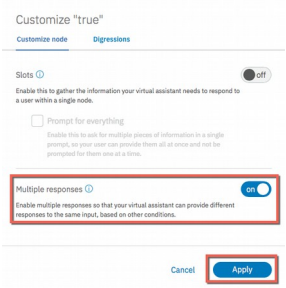
Step 43 Now **click** the menu ⋮ button on the *#out_of_scope* node. And **click** *Add child node*.

Step 44 In this child node,**name** it *Check out of Scope,* in **If bot recognizes**, just **type** *true*.

Step 45 Now **click** the menu ⋮ button on the *#out_of_scope* node. And **click** *Jump to (Recognizes Condition)*.

Step 46 **Click** the *Check out of Scope* node.

Step 47 **Click** Customize and **enable** *Multiple responses*. Then click Apply



Step 48 Now you can add more than one answer.

Add the following:

If bot recognizes	Respond with
intents[0].confidence>=0.94	That question is out of scope for this application, take a look at the Conversation Enhanced application to handle questions like these.
intents[0].confidence<0.94	Sorry I haven't learned answers to questions like this.

Step 49 Now click the Try it and **enter** the following question:

```
Hi, how do I check my tire pressure?
```

This returns the first answer because it matches an out_of_scope intent.

Step 50 Now enter another question:

```
How do I check my account balance?
```

This returns the second answer because it only matches an out_of_scope intent to some degree.

This is not a very good solution because we only check the natural language and we do not have any entities assigned here. It works when you enter

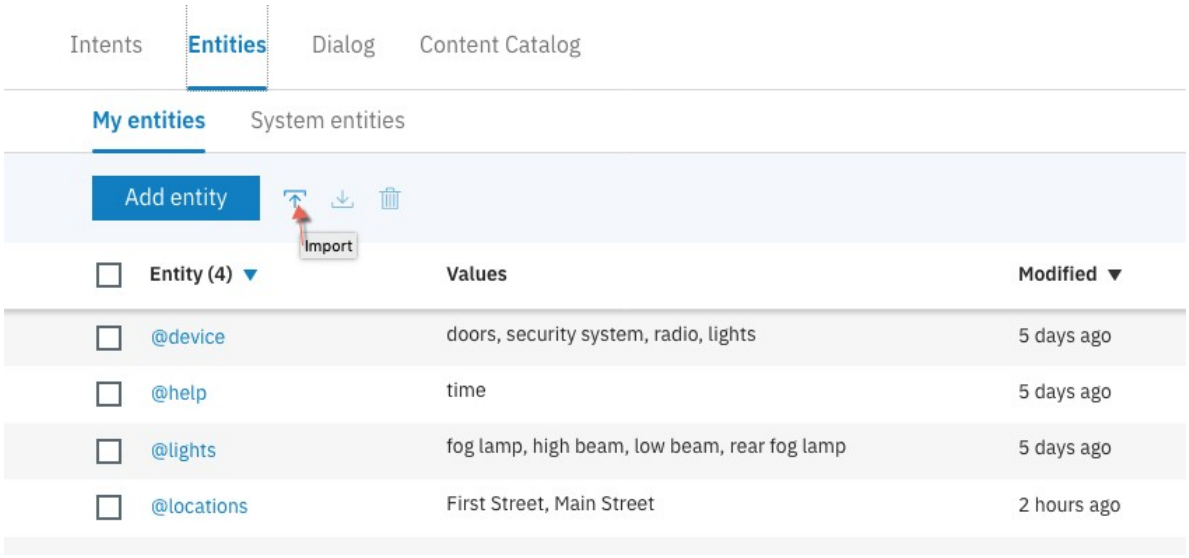
`How do I check my tire pressure` – Similar to `Hi, how do I check my tire pressure?`

`How do I check my blood pressure` Is very similar to the above but <94%

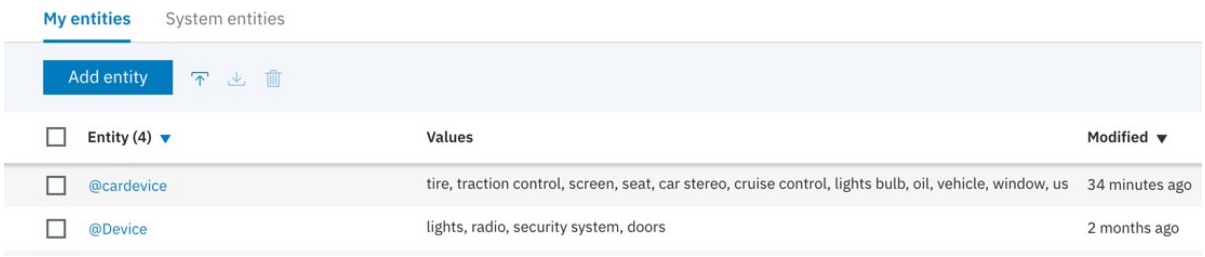
A solution would be just to add an entity @cardevice and add all the devices listed in the intent examples.

Step 51 In your Skill's Entities add an Entity by importing the cardevice_entity.csv CSV file.

```
wget https://github.com/iic-dach/csadConversation/blob/master/resources/
cardevice_entity.csv
```



This will add an Entity @cardevice



Step 52 Now change the *Check out of Scope* node (Step 48) to the following

If bot recognizes	Respond with
@cardevice	That question is out of scope for this application, take a look at the Conversation Enhanced application to handle questions like these.
true	Sorry I haven't learned answers to questions like this.

"true" always executes when the above condition is not met.

Step 53 The backend is now adjusted for integrating the Discovery service into our chatbot application.

Update the application to integrate with the Discovery Service

Step 54 Open the cloned repository in Visual Studio Code or continue with the application completed in Lab 3.

Step 55 In *config.js* enter the credentials and IDs from your Discovery service.
(If you cloned the lab3 repository you make a copy of the *config_sample.js)*

You can cut and paste from the lab05_codesnippets.txt file.

```
var config = {
  watson: {
    assistant: {
      username: "<yourServiceUsername>",
      password: "<yourServicePassword>",
      version: "2019-02-28",
      url: "<yourServiceUrl>",
      assistantId: "<yourAssistantId>"
    },
    discovery: {
      version: "2019-04-30",
      username: "<your discovery username>",
      password: "<your discovery password>",
      url: "<yourServiceUrl>"
    },
    discoveryEnv: {
      collectionId: "<your collectionId>",
      environmentId: "<your environmentId>"
    }
  }
};
module.exports = config;
```

```
var config = {
  watson: {
    assistant: {
      iam_apikey: "<yourApiKey>",
      version: "2019-02-28",
      url: "<yourServiceUrl>",
      assistantId: "<yourAssistantId>"
    },
    discovery: {
      version: "2019-04-30",
      iam_apikey: "<yourApiKey>",
      url: "<yourServiceUrl>"
    },
    discoveryEnv: {
      collectionId: "<your collectionId>",
      environmentId: "<your environmentId>"
    }
  }
};
module.exports = config;
```

Depending on your service, use *username/password* or *apikey*.

Step 56 In controllers*/watson.js* **add** a blank line after line 1.

Step 57 In line 2 add the following – this imports the discovery api

```
const DiscoveryV1 = require('ibm-watson/discovery/v1');
```

Step 58 **Add** a blank line after line 13

Step 59 In line 14 add the following – this creates an instance of Discovery

```
const watsonDiscovery = new DiscoveryV1({
  version: config.watson.discovery.version,
  authenticator: new IamAuthenticator({
    apikey: config.watson.discovery.iam_apikey
  }),
  url: config.watson.discovery.url
});
```

Step 60 Replace assistantMessage() with the following:

```
exports.postMessage = (req, res, next) => {
  let text = '';
  if (req.body.input) {
    text = req.body.input.text;
  }
  watsonAssistant.message({
    assistantId: config.watson.assistant.assistantId,
    sessionId: assistantSessionId,
    input: {
      'message_type': 'text',
      'text': text
    }
  }).then(assistantResult => {
    console.log(JSON.stringify(assistantResult, null, 2));
    if (text === '') {
      return res.json(assistantResult);
    }
    console.log("Detected input: " + text);
    if (assistantResult.result.output.intents.length > 0) {
      var intent = assistantResult.result.output.intents[0];
      console.log("Detected intent: " + intent.intent);
      console.log("Confidence: " + intent.confidence);
    }
    if (assistantResult.result.output.entities.length > 0) {
      var entity = assistantResult.result.output.entities[0];
      console.log("Detected entity: " + entity.entity);
      console.log("Value: " + entity.value);
      if ((entity.entity === 'help') && (entity.value === 'time')) {
        var msg = 'The current time is ' + new Date().toLocaleTimeString();
        console.log(msg);
        assistantResult.result.output.generic[0].text = msg;
      }
      if (intent != null && intent.intent === "out_of_scope"
              && assistantResult.result.output.entities.filter(val => val.entity === "cardevice").length > 0) {
        let discoveryParams = {
          'query': text,
          'environmentId': config.watson.discoveryEnv.environmentId,
          'collectionId': config.watson.discoveryEnv.collectionId,
          'passages': true,
          return: 'text, title, sourceUrl, passages'
        };
        watsonDiscovery.query(discoveryParams)
        .then(discoveryResult => {
          console.log(JSON.stringify(discoveryResult, null, 2));
          assistantResult.result.output.generic[0].text = discoveryResult.result.passages[0].passage_text;
          return res.json(assistantResult);
        }).catch(err => {
          console.log('error:', err);
        });
      } else {
        // console.log(JSON.stringify(result, null, 2));
        res.json(assistantResult);
      }
    } else {
        res.json(assistantResult);
    }
  }).catch(err => {
    console.log(err);
  })
}
```


Lines	Description
50	Successful return from Watson Assistant with assistantResult
56	Check that there is an intent
61	Check that there is an entity
65	Check for the help entity with a time value
70	Check for #out_of_scope with a @cardevice
79	Call the Discovery service with appropriate parameters
82	On a successful return set the first text passage as the Assistant return message
90	When no Discovery is called, return what's set as the return message.
94	Catch any remote call error (networking, etc.)

Step 61 Save the file.

Step 62 In the terminal execute

```
npm install
```
– only if you have cloned the repo.

```
npm start
```

Step 63 In the browser open

http://localhost:3000

Step 64 Enter some statements into the Assistant input line:

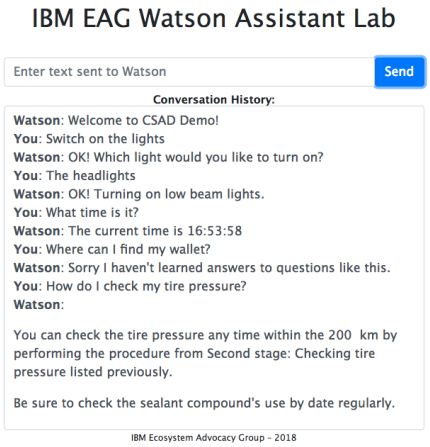
Switch on the lights

The headlights

What time is it?

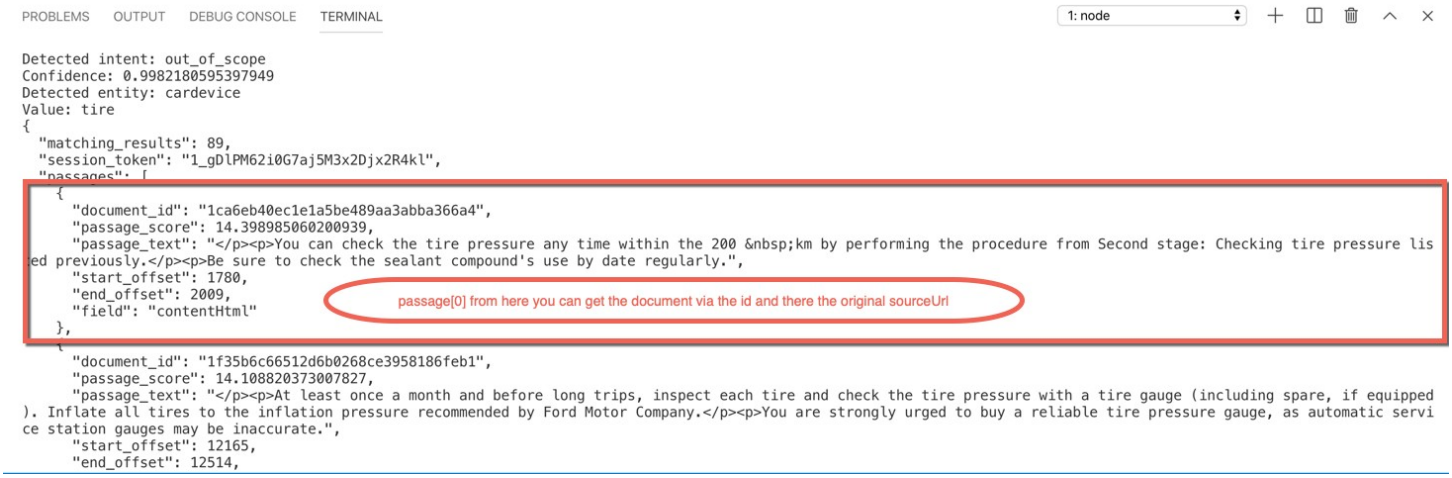
Where can I find my wallet?

How do I check my tire pressure?



The question for the tire pressure should return the same as in **Step 38**, the first passage.

In the server console you should see the JSON returned from Discovery:



Push your application to the IBM Cloud

See **Lab03 Step 127ff** on how to push your application to the IBM Cloud.

Section 4 Integrate Discovery into a Chatbot (Search Skill)

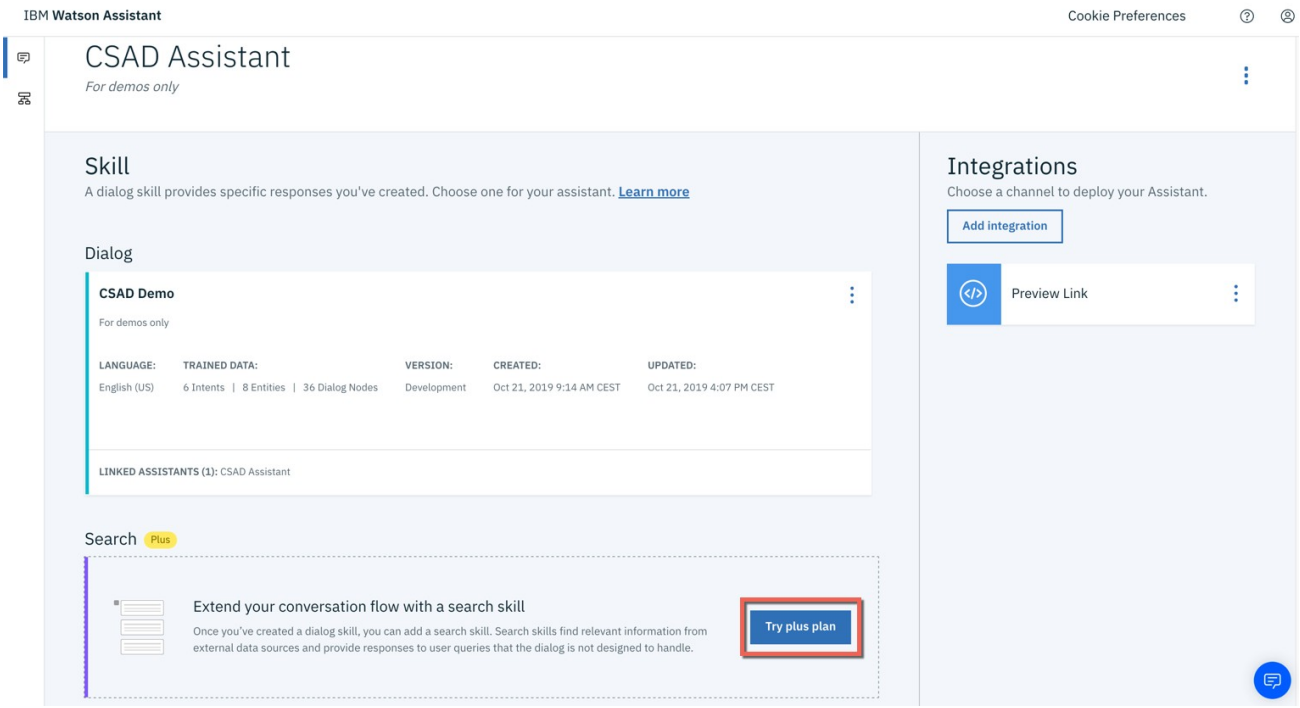
In the previous section we have seen in how to integrate the Watson Discovery service into a chatbot using the API.

In the Watson Assistant service there is now a *Search Skill* option available that allows to directly integrate a Discovery document collection into a chatbot.

To use this search skill we have to upgrade our Discovery service instance to the **Plus Trial** plan.

Create the Search Skill

Step 65 In the browser **open** your *Watson Assistant* console and **click** Try plus plan



Step 66 Then **click** Start free trial and **click** Agree to start the 30 day trial period.

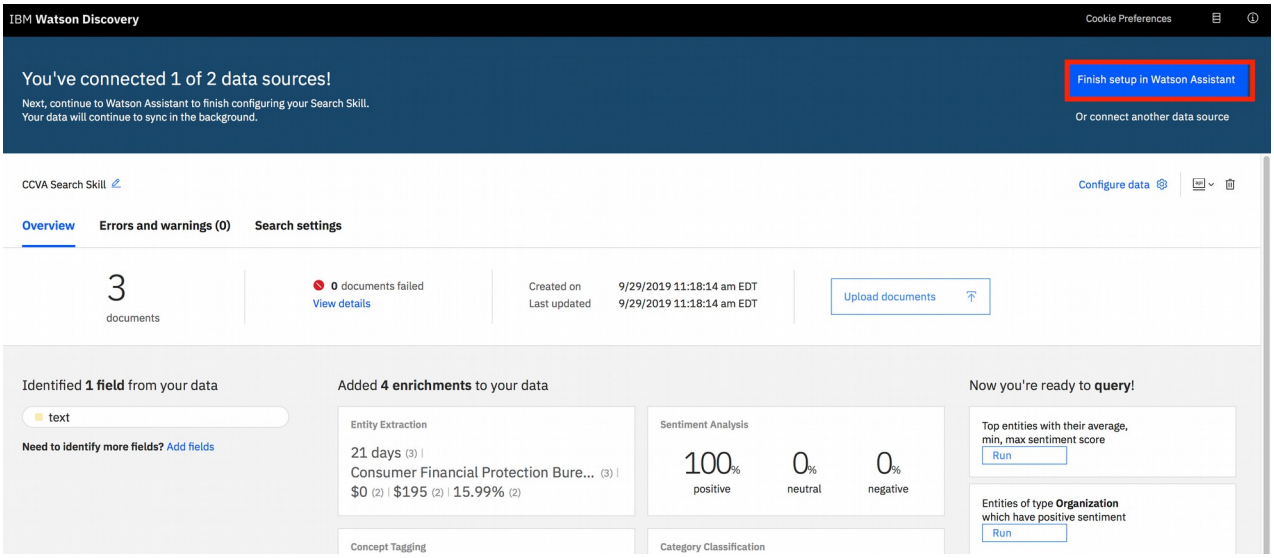
Step 67 After the upgrade **click** Add search skill.

Step 68 **Name it** f.e. `CSAD Search` and **click** *Continue*. Your previously created Discovery service will be displayed with the already defined collections.

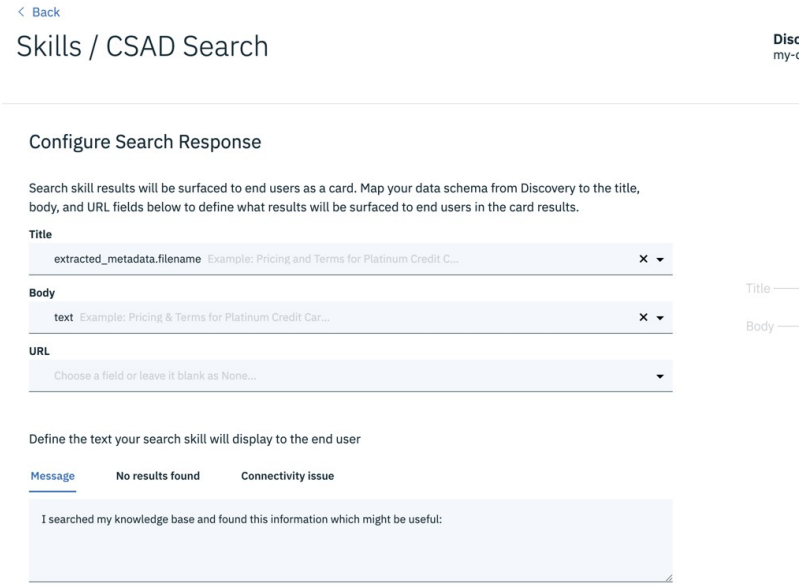
Step 69 **Click** Create new collection. Then **click** Let's get started. You cannot have more than two collection is your lite service.

Step 70 **Click** Upload documents. You can find these documents here or in the resources folder of the Github repository. **Name** your collection `CSAD Search`. Now upload the three PDF documents. **This may take a while!!**.

Step 71 Once the documents have been ingested, **click** *Finish setup in Watson Assistant*

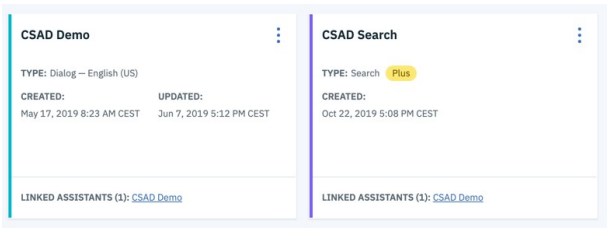


Step 72 Make sure the new collection is selected and **click** *Configure*.



For **Title** select the field: *extracted_metadata.filename*
For **Body**, select the filed: *text*.
For **url**, leave empty.

Then **click** *Create*.



Connect the Search Skill to a place in the dialog

Step 73 **Open** the *Intents* section of your Assistant skill

Step 74 **Click** Create intent enter the following values and **click** Create intent

Field	Value
Intent name	Credit_Card_Info
Description	Search Skill integration

Step 75 The user examples are phrases that will help Watson recognize the new intent. (Enter multiple examples by **pressing** "Enter" or by **clicking** the *Add example*). When finished, **click** ← at the top of the page.

Field	Value
User example	are there any international fees if i use my card in paris atm fees for my credit card fees for using my card abroad what are the apr fees for my silver credit card What are the fees for using an ATM in another country?

Step 76 **Open** the *Entities* section of you Assistant skill

Step 77 **Click** Create entity, **enter** `card_info` and **click** Create entity

Step 78 **Click** Add value. When finished, **click** ← at the top of the page.

Field	Value	Synonym
Entity name	card_info	
Value	APR	annual percentage rate
	cash advance	advanced cash, money advance
	international fees	International charges, foreign fees, foreign charges, fees while abroad, foreign exchange
	annual fee	yearly fee, annual membership, yearly membership, yearly cost, entry fee

Step 79 **Open** the *Dialog* section of your Assistant skill

Step 80 **Click** the menu ⋮ on the *Book a Table* node and then **click** *Add node below*, with the following values.

Field	Value
Name this node…	Information for a Card
If bot recognizes:	#Credit_Card_Info

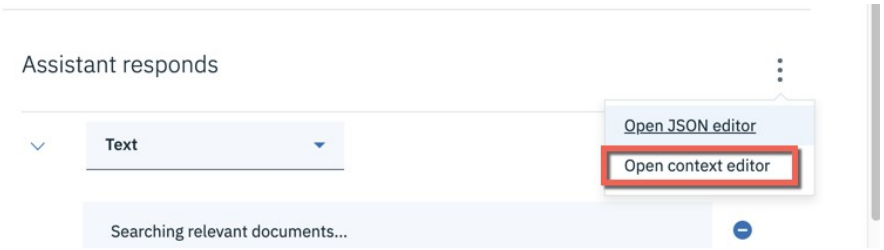
Step 81 **Click** Customize and **enable** *Multiple responses*. The click Apply

Step 82 Enter the following responses:

If Assistant recognizes	Respond with
@card_info:(annual fee)	Searching relevant documents…
@card_info:(international fees)	Searching relevant documents...
@card_info:(APR)	Searching relevant documents…
@card_info:(cash advance)	Searching relevant documents...
anything_else	Searching relevant documents...

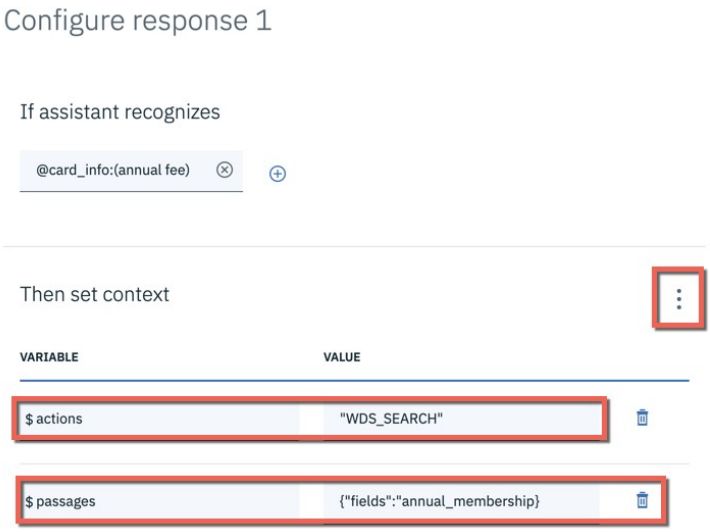
Step 83 Now customize the response for response 1, click ⚙

Step 84 Open the Context Editor

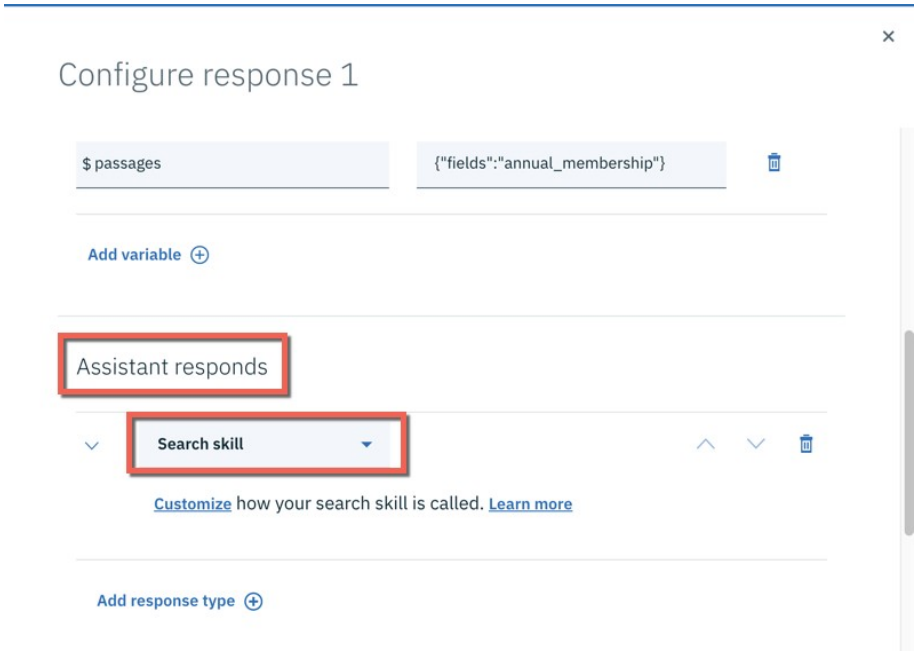


Step 85 Add two context variables

Variable	Value
actions	"WDS_SEARCH"
passages	{"fields":"annual_membership"}



Step 86 **Change** the *Assistant responds* option to *Search skill*. **Click** Save



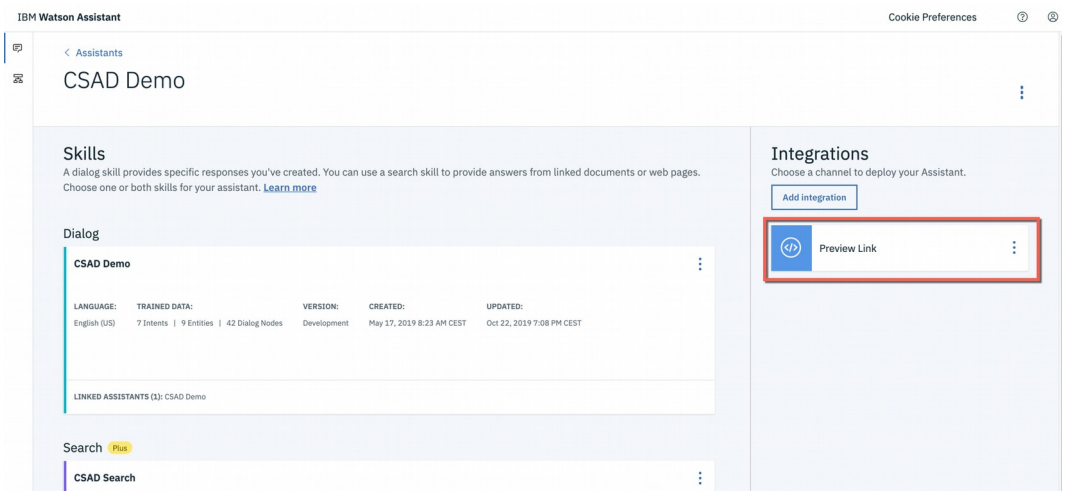
Step 87 Do the same for the second response, but for passages enter the following:

Variable	Value
actions	"WDS_SEARCH"
passages	{"fields":"international_fees"}

Step 88 Then **click** × to close the dialog.

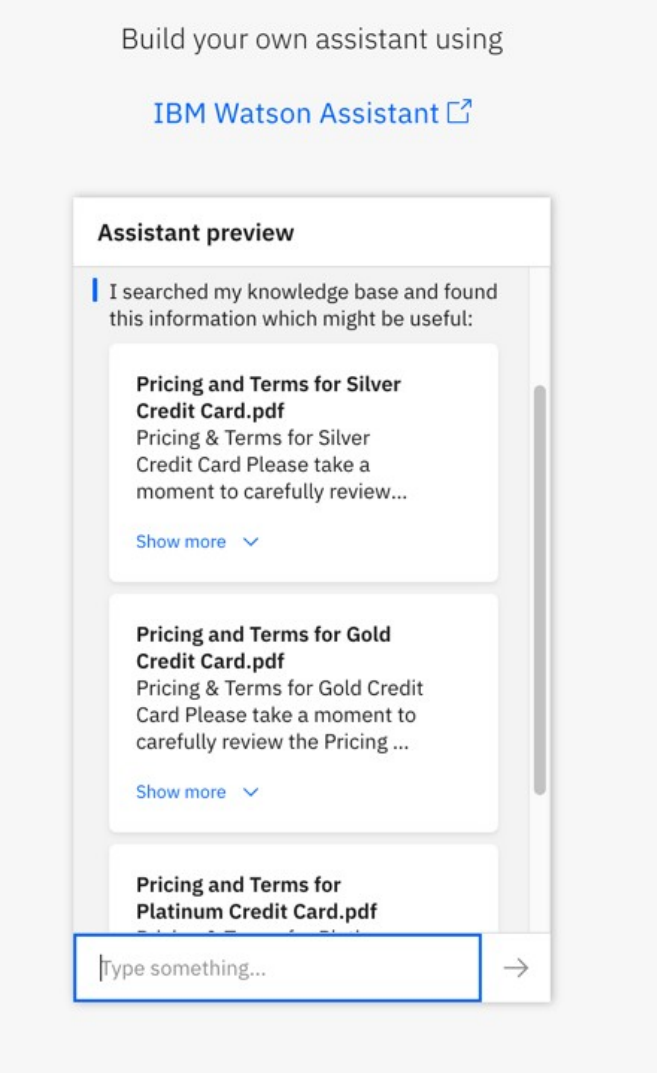
Step 89 All other responses just return "*Searching relevant documents...*"

Step 90 **Open** your Assistant and **click** the *Preview Link*. **Click** the *URL shown* to open the Assistant in the browser.



Step 91 Enter the following into your Assistant preview:

```
Foreign fees on my silver credit card
```



Updating the app to show the result from the search skill

So far the client side of our Nodejs application can only show text messages in a certain format. A search skill result has a different format and therefore we have to adjust the client side JavaScript.

In public/js/scripts.js (line 28) we update our chat log with the line

```
updateChatLog("Watson", json.result.output.generic[0].text);
```

In the object `json.result.output.generic[0]` there is an item `response_type` we have to check now. The search skill will have a `"response_type" : "search"`. This is a quite complex result (check the console log), where we just want to grab a simple text for simplicity reasons. For proper handling, as shown above in the preview above, you should create a JavaScript method to format this nicely.

Step 92 In Visual Studio open public/js/scripts.js.

Step 93 Replace the line

```
updateChatLog("Watson", json.result.output.generic[0].text);
```

with the following lines

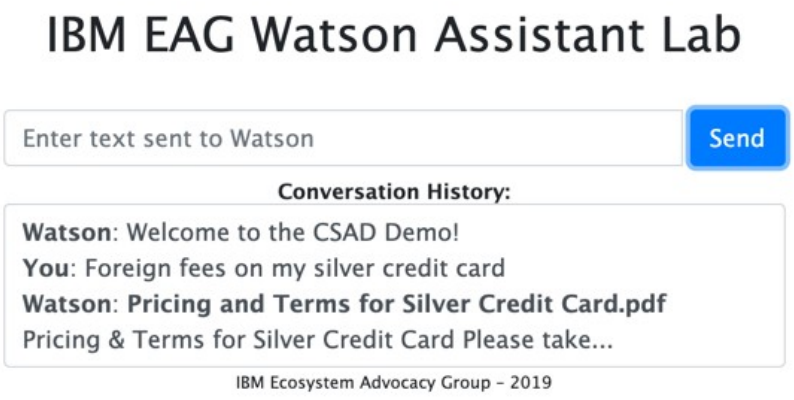
```
if (json.result.output.generic[0].response_type === "text") {
  updateChatLog("Watson", json.result.output.generic[0].text);
} else {
  let title = "<b>" + json.result.output.generic[0].results[0].title + "</b><br/>";
  text = title.concat(json.result.output.generic[0].results[0].body.substring(0, 50
)).concat('...');
  updateChatLog("Watson", text);
}
```

Step 94 **Start** the app with `npm start`.

Step 95 Open http://localhost:3000 in the browser.

Step 96 Enter `Foreign fees on my silver credit card` in the chat bot.

You see the title of the first document returned from the Search skill together with the first 50 characters of the document body.



Tutorials using the Watson Discovery Services

Watson Discovery News Alerting

In this Code Pattern, we will build a Node.js web application that will use the Watson Discovery Service to access Watson Discovery News.

Watson Discovery News is a default data collection that is associated with the Watson Discovery Service. It is a dataset of primarily English language news sources that is updated continuously, with approximately 300,000 new articles and blogs added daily.

https://www.youtube.com/watch?v=zFl-2FybDdY&feature=youtu.be
https://github.com/IBM/watson-discovery-news-alerting
discovering news alerting (The Build Query tool):
https://www.youtube.com/watch?v=N-HaIpPGde0&feature=youtu.be