FREE eBook

LEARNING ibm-watson-cognitive

Free unaffiliated eBook created from **Stack Overflow contributors.**

#ibm-watson-cognitive

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ibm-watson-cognitive

It is an unofficial and free ibm-watson-cognitive ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ibm-watson-cognitive.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ibm-watson-cognitive

Remarks

This topic provides basic instructions for obtaining credentials for Watson services and provides relevant links for each service and the Watson Developer Cloud SDKs.

Watson services homepages:

- AlchemyLanguage
- AlchemyData News
- Conversation
- Discovery
- Document Conversion
- Language Translation
- Natural Language Classifier
- Natural Language Understanding
- Personality Insights
- Retrieve and Rank
- Speech to Text
- Text to Speech
- Tone Analyzer
- Tradeoff Analytics
- Visual Recognition

Versions

Version	Release Date
1.0.0	2016-05-05

Examples

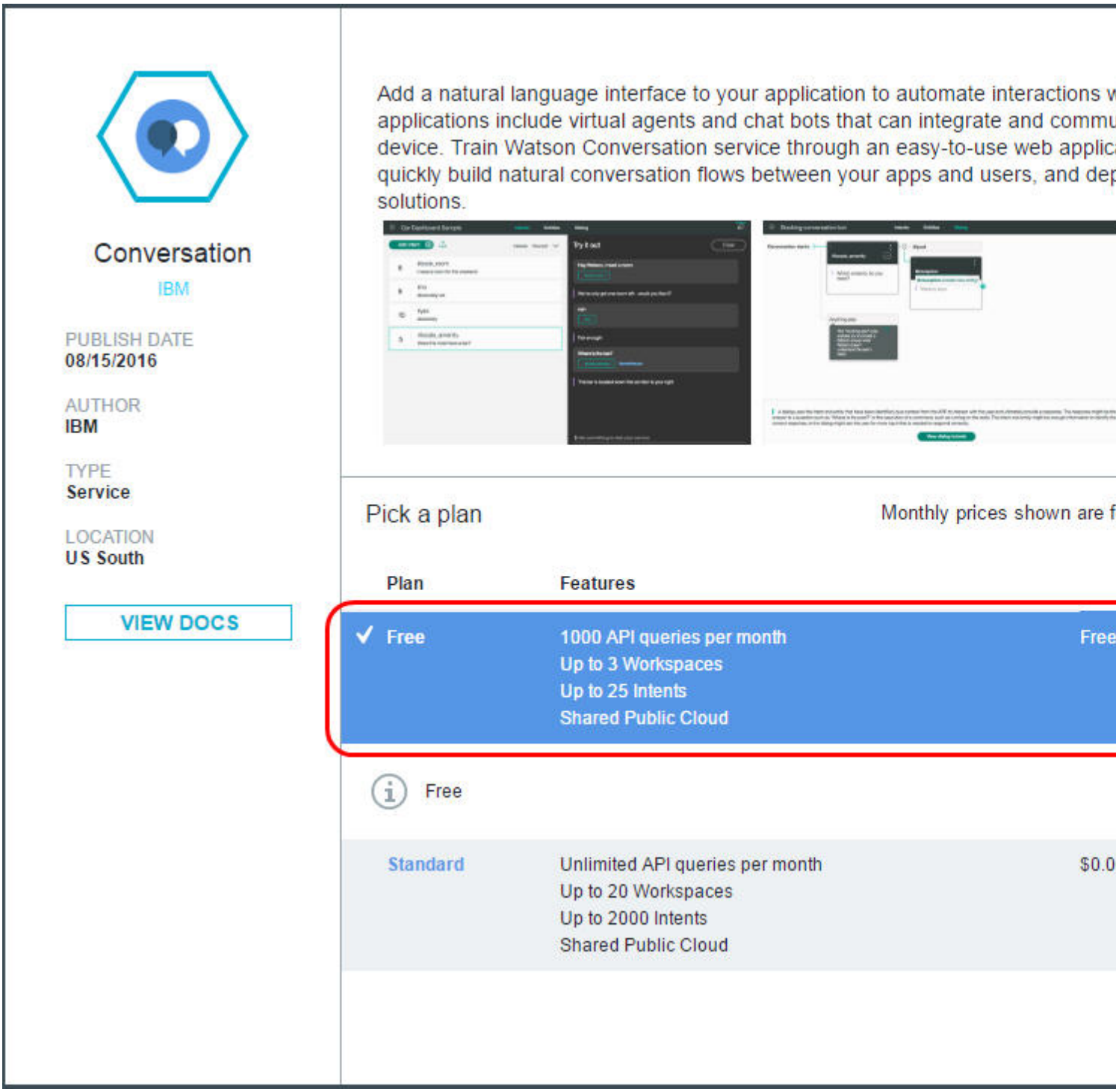
Getting API credentials

To authenticate to Watson services, you need credentials for each service that you plan to use. Depending on the service, you will need to pass a username and password with Basic Authentication, or you will need to pass an API key in a parameter for each request you make.

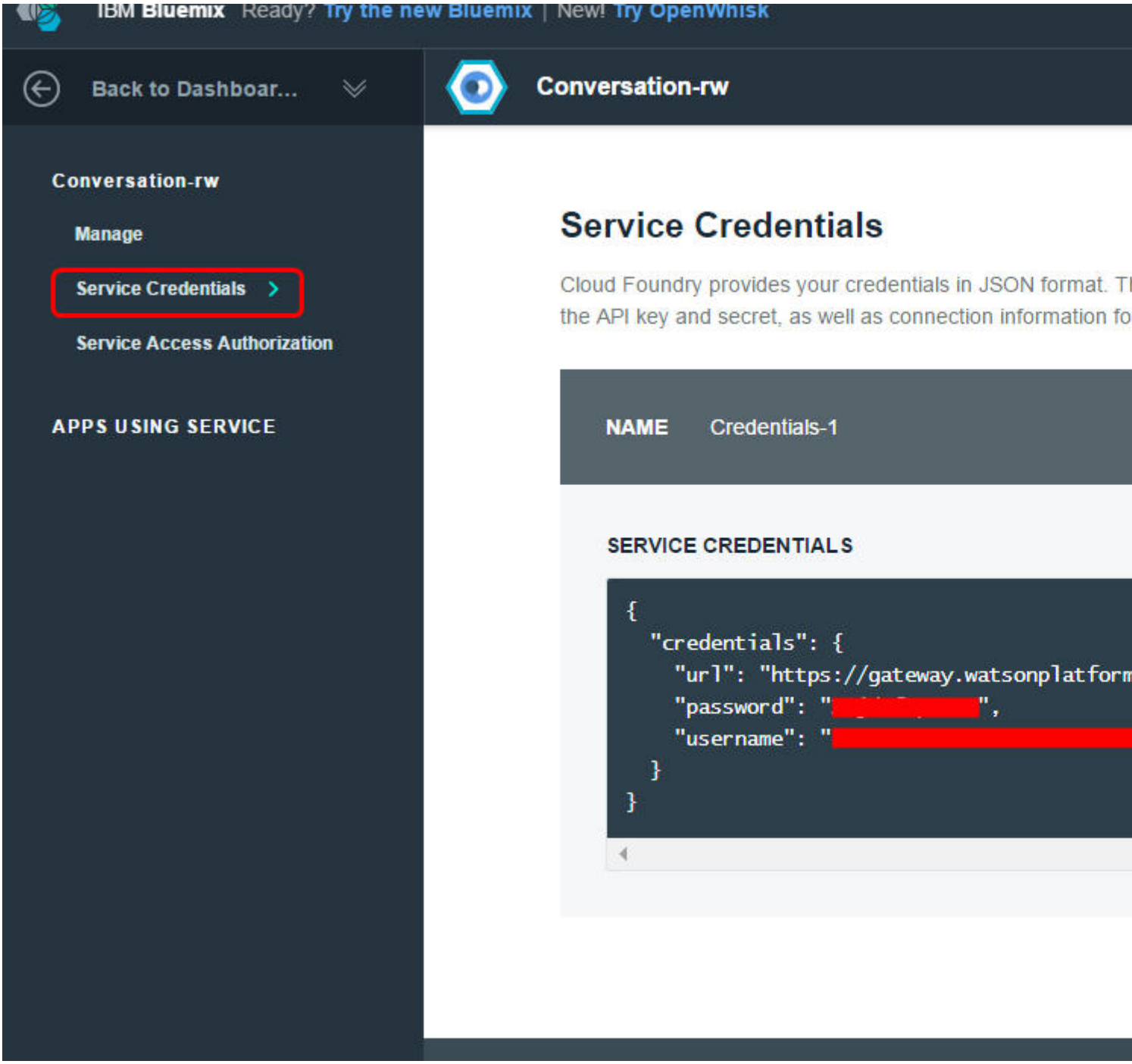
How to get credentials for a Watson service:

1. Sign up for Bluemix and log in.

2. Go to the service page for your desired Watson service:
    - AlchemyLanguage and AlchemyData News
    - Conversation
    - Dialog
    - Document Conversion
    - Language Translation
    - Natural Language Classifier
    - Personality Insights
    - Retrieve and Rank
    - Speech to Text
    - Text to Speech
    - Tone Analyzer
    - Tradeoff Analytics
    - Visual Recognition
3. Select your desired plan, and click CREATE:



4. Click the "Service Credentials" button from your service dashboard page to view your credentials. If you aren't taken to the service dashboard automatically, go to your Bluemix dashboard and click on your desired service instance.



Calling Watson APIs with curl

Depending on the service, you will either need to use Basic Authentication with a `username` and `password` or pass an `apikey` as a parameter in each request.

Some services also support token authentication.

GET using Tone Analyzer:

```
curl -X GET \
-u "username":"password" \
-d "version=2016-05-19" \
-d "text=Hey! Welcome to Watson Tone Analyzer!" \
"https://gateway.watsonplatform.net/tone-analyzer/api/v3/tone
```

POST using AlchemyLanguage:

```
curl -X POST \
-d "apikey=YOUR_API_KEY" \
-d "url=www.ibm.com" \
"https://gateway-a.watsonplatform.net/calls/url/URLGetRankedKeywords"
```

Using Watson Developer Cloud SDKs

The quickest way to get started with Watson services is to use the Watson Developer Cloud SDKs. The following GitHub repositories contain installation instructions and basic usage examples:

- Android
- iOS
- Java
- Node.js
- Python
- Unity

For example, here's how to make an AlchemyLanguage API call with the Node.js SDK:

Install the SDK:

```
$ npm install watson-developer-cloud
```

Save the following code to a file (we'll call it *app.js*). Make sure you replace `API_KEY` with your API key.

```
// Instantiate the service
var AlchemyLanguageV1= require('watson-developer-cloud/alchemy-language/v1');
var alchemy_language = AlchemyLanguageV1({
  api_key: 'API_KEY'
})

var parameters = {
  extract: [
    'entities',
    'keywords'
  ]
  url: 'https://www.ibm.com/us-en/'
};

alchemy_language.combined(parameters, function (err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response, null, 2));
});
```

Run the app:

```
$ node app.js
```

Read Getting started with ibm-watson-cognitive online: https://riptutorial.com/ibm-watson-cognitive/topic/607/getting-started-with-ibm-watson-cognitive

Chapter 2: AlchemyLanguage

Remarks

AlchemyLanguage is a collection of text analysis methods that provide deeper insight into your text or HTML content. See the Getting Started topic to learn how to get started with AlchemyLanguage and other Watson services. For more AlchemyLanguage details and examples, see the API reference and documentation.

Size limits

- HTML content before text cleaning: **600 KB**
- Source text, after text cleaning: **50 KB**
- Calls that use Custom Models: **5 KB**

Language support

To see which languages are supported for each function, refer to each function's entry in the API reference.

Language detection

By default, AlchemyLanguage automatically detects the language of your source text. You can manually specify the language of your content with the `language` query parameter. (e.g. `language=spanish`)

Text cleaning

When you use an HTML or URL function of the API, AlchemyLanguage cleans the content to prepare the source text for the analysis. The `sourceText` parameter allows you to customize the cleaning process with the following options:

- `cleaned_or_raw` (default) -- Removes website elements such as links, ads, etc. If cleaning fails, raw web page text is used
- `cleaned`-- Removes website elements such as links, ads, etc.
- `raw` -- Uses raw web page text with no cleaning
- `cquery` -- Uses the visual constraints query that you specify in the `cquery` parameter. See the documentation for details about visual constraints queries.
- `xpath` -- Uses the XPath query that you specify in the `xpath` parameter
- `xpath_or_raw` -- Uses the results of an XPath query, falling back to plain text if the XPath

query returns nothing
- `cleaned_and_xpath` -- Uses the results of an XPath query on cleaned web page text

Examples

Combined Call: use multiple functions in a single API call (Node.js)

The Combined Call method allows you to use multiple AlchemyLanguage functions in one request. This example uses a Combined Call to get entities and keywords from the IBM website and returns sentiment information for each result.

> This example requires AlchemyLanguage service credentials and Node.js.

1. Use a command-line interface to install the Watson Developer Cloud Node.js SDK:

```
$ npm install watson-developer-cloud
```

2. Save the following code to an *app.js* file in the same directory. Make sure you replace `API_KEY` with your AlchemyAPI key:

```
var AlchemyLanguageV1 = require('watson-developer-cloud/alchemy-language/v1');
var alchemy_language = AlchemyLanguageV1({
  api_key: 'API_KEY'
})

var parameters = {
  extract: 'entities,keywords',
  sentiment: 1,
  url: 'https://www.ibm.com/us-en/'
};

alchemy_language.combined(parameters, function (err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response, null, 2));
});
```

3. Run the app:

```
$ node app.js
```

Sentiment Analysis: get sentiment information for specific phrases in text (Node.js)

AlchemyLanguage's Targeted Sentiment feature can search your content for target phrases and return sentiment information for each result.

> This example requires AlchemyLanguage service credentials and Node.js

1. Use a command-line interface to install the Watson Developer Cloud Node.js SDK:

```
$ npm install watson-developer-cloud
```

2. Save the following code to an *app.js* file in the same directory. Make sure you replace `API_KEY` with your AlchemyAPI key:

```
var AlchemyLanguageV1 = require('watson-developer-cloud/alchemy-language/v1');
var alchemy_language = new AlchemyLanguageV1({
  api_key: 'API_KEY'
})

var parameters = {
  text: 'Grapes are the best! I hate peaches.',
  targets: [
    'grapes',
    'peaches'
  ]
};

alchemy_language.sentiment(parameters, function (err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response, null, 2));
});
```

3. Run the app:

```
$ node app.js
```

Concepts: identify concepts from a webpage (Node.js)

AlchemyLanguage can detect general concepts referenced in your content. The service returns Linked Data links for each concept and a URL to a relevant website when possible.

> This example requires AlchemyLanguage service credentials and Node.js

1. Use a command-line interface to install the Watson Developer Cloud Node.js SDK:

```
$ npm install watson-developer-cloud
```

2. Save the following code into an *app.js* file in the same directory. Make sure you replace `API_KEY` with your AlchemyAPI key.

```
var AlchemyLanguageV1 = require('watson-developer-cloud/alchemy-language/v1');
var alchemy_language = new AlchemyLanguageV1({
  api_key: 'API_KEY'
})

var parameters = {
  url: 'http://www.cnn.com'
};
```

```
alchemy_language.concepts(parameters, function (err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response, null, 2));
});
```

3. Run the app:

```
$ node app.js
```

Read AlchemyLanguage online: https://riptutorial.com/ibm-watson-cognitive/topic/6817/alchemylanguage

Chapter 3: Retrieve and Rank

Remarks

The Solrj client and the Java SDK are independent so you can update them individually. Always make sure you use the latest version of the Java SDK.

See the GitHub release page for updates https://github.com/watson-developer-cloud/java-sdk/releases

Examples

Search and Rank using the Retrieve and Rank in Java

Install the required dependencies:

```
'org.apache.solr:solr-solrj:5.5.1'
'org.apache.httpcomponents:httpclient:4.3.6'
'com.ibm.watson.developer_cloud:java-sdk:3.2.0'
```

The code below assumes you have a Solr collection with documents and you have trained a ranker, otherwise follow this tutorial

```
public class RetrieveAndRankSolrJExample {

  private static HttpSolrClient solrClient;
  private static RetrieveAndRank service;

  private static String USERNAME = "<username>";
  private static String PASSWORD = "<password>";
  private static String SOLR_CLUSTER_ID = "<your-solr-cluster-id>";
  private static String SOLR_COLLECTION_NAME = "<your-collection-name>";
  private static String RANKER_ID = "<ranker-id>";

  public static void main(String[] args) throws SolrServerException, IOException {

    // create the retrieve and rank instance
    service = new RetrieveAndRank();
    service.setUsernameAndPassword(USERNAME, PASSWORD);

    // create the solr client
    String solrUrl = service.getSolrUrl(SOLR_CLUSTER_ID);
    solrClient = new HttpSolrClient(solrUrl, createHttpClient(solrUrl, USERNAME, PASSWORD));

    // build the query
    SolrQuery query = new SolrQuery("*:*");
    query.setRequestHandler("/fcselect");
    query.set("ranker_id", RANKER_ID);

    // execute the query
    QueryResponse response = solrClient.query(SOLR_COLLECTION_NAME, query);
    System.out.println("Found " + response.getResults().size() + " documents!");
```

```
    System.out.println(response);
  }

  private static HttpClient createHttpClient(String uri, String username, String password) {
    final URI scopeUri = URI.create(uri);

    final BasicCredentialsProvider credentialsProvider = new BasicCredentialsProvider();
    credentialsProvider.setCredentials(new AuthScope(scopeUri.getHost(), scopeUri.getPort()),
        new UsernamePasswordCredentials(username, password));

    final HttpClientBuilder builder = HttpClientBuilder.create()
        .setMaxConnTotal(128)
        .setMaxConnPerRoute(32)

.setDefaultRequestConfig(RequestConfig.copy(RequestConfig.DEFAULT).setRedirectsEnabled(true).build())

        .setDefaultCredentialsProvider(credentialsProvider)
        .addInterceptorFirst(new PreemptiveAuthInterceptor());
    return builder.build();
  }

  private static class PreemptiveAuthInterceptor implements HttpRequestInterceptor {
    public void process(final HttpRequest request, final HttpContext context) throws
HttpException {
      final AuthState authState = (AuthState)
context.getAttribute(HttpClientContext.TARGET_AUTH_STATE);

      if (authState.getAuthScheme() == null) {
        final CredentialsProvider credsProvider = (CredentialsProvider) context
            .getAttribute(HttpClientContext.CREDS_PROVIDER);
        final HttpHost targetHost = (HttpHost)
context.getAttribute(HttpCoreContext.HTTP_TARGET_HOST);
        final Credentials creds = credsProvider.getCredentials(new
AuthScope(targetHost.getHostName(),
            targetHost.getPort()));
        if (creds == null) {
          throw new HttpException("No creds provided for preemptive auth.");
        }
        authState.update(new BasicScheme(), creds);
      }
    }
  }
}
```

Read Retrieve and Rank online: https://riptutorial.com/ibm-watson-cognitive/topic/6053/retrieve-and-rank

Chapter 4: Speech to Text

Remarks

IBM Watson Speech to Text offers a variety of options for transcribing audio in various languages and formats:

- **WebSockets** – establish a persistent connection over the WebSocket protocol for continuous transcription
- **Sessionless** – transcribe audio without the overhead of establishing and maintaining a session
- **Sessions** – create long multi-turn exchanges with the service or establish multiple parallel conversations with a particular instance of the service
- **Asynchronous** – provides a non-blocking HTTP interface for transcribing audio. You can register a callback URL to be notified of job status and results, or you can poll the service to learn job status and retrieve results manually.

See the Getting Started topic to learn how to get started with Speech to Text and other Watson services. For more Speech to Text details and examples, see the API reference and the documentation.

Examples

Recognizing an audio file using WebSockets in Java

Using the `Java-SDK 3.0.1`

```
CountDownLatch lock = new CountDownLatch(1);

SpeechToText service = new SpeechToText();
service.setUsernameAndPassword("<username>", "<password>");

FileInputStream audio = new FileInputStream("filename.wav");

RecognizeOptions options = new RecognizeOptions.Builder()
    .continuous(true)
    .interimResults(true)
    .contentType(HttpMediaType.AUDIO_WAV)
    .build();

service.recognizeUsingWebSocket(audio, options, new BaseRecognizeCallback() {
  @Override
  public void onTranscription(SpeechResults speechResults) {
    System.out.println(speechResults);
    if (speechResults.isFinal())
      lock.countDown();
  }
```

```
});

lock.await(1, TimeUnit.MINUTES);
```

Transcribing an audio file using WebSockets (Node.js)

This example shows how to use the IBM Watson Speech to Text service to recognize the type of an audio file and produce a transcription of the spoken text in that file.

> This example requires Speech to Text service credentials and Node.js

1. Install the npm module for the Watson Developer Cloud Node.js SDK:

```
$ npm install watson-developer-cloud
```

2. Create a JavaScript file (for example, *app.js*) and copy the following code into it. Make sure you enter the `username` and `password` for your Speech to Text service instance.

```
var SpeechToTextV1 = require('watson-developer-cloud/speech-to-text/v1');
var fs = require('fs');

var speech_to_text = new SpeechToTextV1({
  username: 'INSERT YOUR USERNAME FOR THE SERVICE HERE',
  password: 'INSERT YOUR PASSWORD FOR THE SERVICE HERE',
  url: 'https://stream.watsonplatform.net/speech-to-text/api'
});

var params = {
  content_type: 'audio/flac'
};

// Create the stream,
var recognizeStream = speech_to_text.createRecognizeStream(params);

// pipe in some audio,
fs.createReadStream('0001.flac').pipe(recognizeStream);

// and pipe out the transcription.
recognizeStream.pipe(fs.createWriteStream('transcription.txt'));

// To get strings instead of Buffers from received `data` events:
recognizeStream.setEncoding('utf8');

// Listen for 'data' events for just the final text.
// Listen for 'results' events to get the raw JSON with interim results, timings, etc.
['data', 'results', 'error', 'connection-close'].forEach(function(eventName) {
  recognizeStream.on(eventName, console.log.bind(console, eventName + ' event: '));
});
```

3. Save the sample audio file *0001.flac* to the same directory. This example code is set up to process **FLAC** files, but you could modify the `params` section of the sample code to obtain transcriptions from audio files in other formats. Supported formats include **WAV** (type `audio/wav`), **OGG** (type `audio/ogg`) and others. See the Speech to Text API reference for a complete list.

4. Run the application (use the name of the file that contains the example code)

```
$ node app.js
```

After running the application, you will find the transcribed text from your audio file in the file *transcription.txt* in the directory from which you ran the application.

Read Speech to Text online: https://riptutorial.com/ibm-watson-cognitive/topic/675/speech-to-text

Chapter 5: Visual Recognition

Examples

Get a list of custom classifiers

This lists all of the custom classifiers you have trained.

```
'use strict';

let watson = require('watson-developer-cloud');

var visualRecognition = watson.visual_recognition({
  version: 'v3',
  api_key: process.env['API_KEY'],
  version_date:'2016-05-19'
});


let url =
"https://upload.wikimedia.org/wikipedia/commons/1/1c/Chris_Evans_filming_Captain_America_in_DC_cropped.


visualRecognition.classify({url: url}, function(error, results) {
  console.log(JSON.stringify(results,null,2));
});
```

Get information about a specific custom classifier

This returns information about a specific classifier ID you have trained. This includes information about its current status (i.e., if it is ready or not).

```
'use strict';

let watson = require('watson-developer-cloud');

var visualRecognition = watson.visual_recognition({
  version: 'v3',
  api_key: process.env.API_KEY,
  version_date:'2016-05-19'
});


visualRecognition.getClassifier({classifier_id: 'DogBreeds_1162972348'}, function(error,
results) {
  console.log(JSON.stringify(results,null,2));
});
```

Train a custom classifier

Training a custom classifier requires a corpus of images organized into groups. In this example, I have a bunch of images of apples in one ZIP file, a bunch of images of bananas in another ZIP

file, and a third group of images of things that are *not* fruits for a *negative set*. Once a custom classifier is created, it will be in state `training` and you'll have to use the classifier ID to check if it is ready (using the 'Get information about a specific custom classifier' example).

```
'use strict';

let watson = require('watson-developer-cloud');
let fs = require('fs');

var visualRecognition = watson.visual_recognition({
  version: 'v3',
  api_key: process.env.API_KEY,
  version_date:'2016-05-19'
});


let custom_classifier = {
  apple_positive_examples: fs.createReadStream('./apples.zip'),
  banana_positive_examples: fs.createReadStream('./bananas.zip'),
  negative_examples: fs.createReadStream('./non-fruits.zip'),
  name: 'The Name of My Classifier'
}

visualRecognition.createClassifier(custom_classifier, function(error, results) {
  console.log(JSON.stringify(results,null,2));
});
```

Delete a custom classifier

```
'use strict';

let watson = require('watson-developer-cloud');
let fs = require('fs');

var visualRecognition = watson.visual_recognition({
  version: 'v3',
  api_key: process.env.API_KEY,
  version_date:'2016-05-19'
});

let classifier_id_to_delete = 'TheNameofMyClassifier_485506080';

visualRecognition.deleteClassifier({classifier_id: classifier_id_to_delete}, function(error,
results) {
  console.log(JSON.stringify(results,null,2));
});
```

Classify an Image

Prerequisites

First, you have to install the `watson-developer-cloud` SDK.

```
$ npm install watson-developer-cloud
```

Classify an image URL

We'll use an image of Captain America from Wikipedia.

```
'use strict';

let watson = require('watson-developer-cloud');

var visualRecognition = watson.visual_recognition({
  version: 'v3',
  api_key: "<YOUR API KEY GOES HERE>",
  version_date:'2016-05-19'
});


let url =
"https://upload.wikimedia.org/wikipedia/commons/1/1c/Chris_Evans_filming_Captain_America_in_DC_cropped


visualRecognition.classify({url: url}, function(error, results) {
  console.log(JSON.stringify(results,null,2));
});
```

Read Visual Recognition online: https://riptutorial.com/ibm-watson-cognitive/topic/718/visual-recognition

Credits

S. No	Chapters	Contributors
1	Getting started with ibm-watson-cognitive	Community, Garrett M, German Attanasio
2	AlchemyLanguage	Garrett M
3	Retrieve and Rank	German Attanasio
4	Speech to Text	Garrett M, German Attanasio, seh, WvH
5	Visual Recognition	German Attanasio, Joshua Smith, seh