Creating a Custom Classifier

1. Introduction to the Watson Visual Recognition API 2

2. Introduction to Creating Classifiers 3

3. Acquiring the API Key 4

4. Getting Started with the Lab Material 5

5. Introduction to the Node.js Code 6

6. Introduction to the Code 7

7. The /classifyimage Route 8

8. Appendix 9

9. First Time with Node.js? 9

10. JSON 9

# Introduction to the Watson Visual Recognition API

The IBM Watson™ Visual Recognition service uses deep learning algorithms to identify scenes, objects, and celebrity faces in images you upload to the service. You can create and train a custom classifier to identify subjects that suit your needs.

Source: [https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/ - introduction](https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#introduction)

Additional Information: <https://www.ibm.com/watson/developercloud/doc/visual-recognition/>

# Introduction to Creating Classifiers

Train a new multi-faceted classifier on the uploaded image data. A new custom classifier can be trained by several compressed (.zip) files, including files containing positive or negative images (.jpg, or .png). You must supply at least two compressed files, either two positive example files or one positive and one negative example file.

Compressed files containing positive examples are used to create “classes” that define what the new classifier is. The prefix that you specify for each positive example parameter is used as the class name within the new classifier. The "\_positive\_examples" suffix is required. There is no limit on the number of positive example files you can upload in a single call.

The compressed file containing negative examples is not used to create a class within the created classifier, but does define what the new classifier is not. Negative example files should contain images that do not depict the subject of any of the positive examples. You can only specify one negative example file in a single call. For more information, see [Structure of the training data](http://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.shtml#structure), and [Guidelines for good training](http://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.shtml#goodtraining).

Source: <https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#classifiers>

# Acquiring the API Key

1. You will first need to have an IBM Bluemix ID to be able to access the Watson APIs. Sign-up and configure your IBM Bluemix ID here  
     
   <https://console.ng.bluemix.net/>
2. Once you have completed that process, login and create a new application
3. Search for “Visual Recognition” and select that service under the Watson section on the Bluemix Console.
4. Confirm or change the service name and credential name. Make sure you start with the free plan and then click create.
5. Select your service on the console in the “All Services” section.
6. Identify and copy your API key from the service to be used through the rest of this lab

# Getting Started with the Lab Material

* Acquire an API key before you start
* Install node + npm <https://nodejs.org/en/download/>
* Download and save the lab code to an empty folder
* From the command line run "node --version"
* From the command line run "npm --version"
* From the command line run the following commands to install Node components to the global (-g) environment needed for the lab

npm install -g express

npm install -g nodemon

npm install -g body-parser

npm install -g url-exists

npm install -g request

npm install -g fs

npm install -g ejs

npm install -g watson-developer-cloud

* From the command line change to the directory where you downloaded the lab code and run the following commands

run "npm init" in the directory where the package.json file is to pull in the necessary Node.js component to the project folder

run "npm start" to launch the application

# Introduction to the Node.js Code

The /customclassify route takes a URL from a form POST method and calls the Watson Visual Recognition API. The data returned from the API is a JSON object.

[Find details about the JSON object here under the RESPONSE section](https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#classify_an_image)

The JSON object is nested with the meta data at the root and the class information found here

custclass: response.classes

The call to the Watson API is done here where params contains either pointers to local files.

var params = {};

params['name'] = classifier

params[classifier.toLowerCase() + '\_positive\_examples'] = fs.createReadStream(positives);

params['negative\_examples'] = fs.createReadStream(negatives);

# Introduction to the Code

The URL is being past in from a form (index.ejs) with the POST method and we extract the value from the request body

var classifier = req.body.classifier;

var positives = req.body.positive\_file;

var negatives = req.body.negative\_file;

The files containing the positives and negatives are passed in as paramaters to the API call

var params = {};

params['name'] = classifier

params[classifier.toLowerCase() + '\_positive\_examples'] = fs.createReadStream(positives);

params['negative\_examples'] = fs.createReadStream(negatives);

All data required for the API call have now been gathered and the request can be executed

visual\_recognition.createClassifier(params, function(err, response) {

if (err) {

if ( debug ) { console.log( "There was a problem calling the Watson API" ) }

console.log(err);

} else {

// The Watson API call was successful, process the content

if ( debug ) {

console.log( "The call to the Watson API was successful, let's render the data" )

console.log( JSON.stringify(response, null, 2) );

console.log( response.classes[0].class);

};

res.render('pages/customclassify',{

positives: positives,

classifierid: response.classifier\_id,

name: response.name,

owner: response.owner,

status: response.status,

created: response.created,

custclass: response.classes[0].class

});

}

});

# The /classifyimage Route

The full Node.js code below passes the JSON object to a view (classifyimage.ejs) eventually rendering the content returned by the Watson API.

// Detect faces in an image

app.post('/detectfaces', function(req, res) {

if ( debug ) { console.log( "POST request was submitted for detecting faces of an image" ) }

var imageurl = req.body.imgurl;

urlExists(imageurl, function(err, exists) {

if ( exists ) {

if ( debug ) {

console.log( "Entering the detect faces code" );

console.log( "Does the URL actually exist => " + exists );

}

var params = {

// Usage images\_file to pass in a file from the local file system

//images\_file: fs.createReadStream('images\_animals/bws\_5d\_turkey\_061.jpg')

//Use url to pass in an image from the Internet

url: imageurl

};

if ( debug ) { console.log( "Calling the Watson API" ) }

visual\_recognition.detectFaces(params, function(err, response) {

if (err) {

if ( debug ) { console.log( "There was a problem calling the Watson API" ) }

console.log(err);

} else {

// The Watson API call was successful, process the content

if ( debug ) {

console.log( "The call to the Watson API was successful, let's render the data" )

console.log( response.images[0].faces );

};

res.render('pages/detectfaces',{

mytext: imageurl,

images: response.images[0].faces

});

}

});

}

});

});

# Appendix

# First Time with Node.js?

Look here for getting started material

<https://expressjs.com/en/starter/hello-world.html>

<https://www.youtube.com/watch?v=pU9Q6oiQNd0>

<https://www.youtube.com/watch?v=-u-j7uqU7sI&list=PL6gx4Cwl9DGBMdkKFn3HasZnnAqVjzHn_>

<https://nodejs.org/en/docs/>

# JSON

## The JSON Challenge

Use these web sites to look at the returned JSON

[www.jsonlint.org](http://www.jsonlint.org/)

<http://jsonmate.com/>

Use this Link to understand how to access the various bits inside the JSON object

<http://www.w3schools.com/js/js_json_arrays.asp>

// app.get('/', function(req, res) {

//

//     var myObj = {

//      "custom\_classes": 0,

//      "images": [{

//           "classifiers": [{

//                "classes": [{

//                     "class": "Manx cat",

//                     "score": 0.832,

//                     "type\_hierarchy": "/animal/mammal/carnivore/feline/domestic cat/Manx cat"

//                }, {

//                     "class": "domestic cat",

//                     "score": 0.948

//                }, {

//                     "class": "animal",

//                     "score": 0.968

//                }, {

//                     "class": "tomcat",

//                     "score": 0.702,

//                     "type\_hierarchy": "/animal/mammal/carnivore/feline/domestic cat/tomcat"

//                }, {

//                     "class": "tiger cat",

//                     "score": 0.554,

//                     "type\_hierarchy": "/animal/mammal/carnivore/feline/domestic cat/tiger cat"

//                }, {

//                     "class": "tan color",

//                     "score": 0.557

//                }],

//                "classifier\_id": "default",

//                "name": "default"

//           }],

//           "image": "bws\_5d\_turkey\_061.jpg"

//      }],

//      "images\_processed": 1

// }

//

// console.log('Custom Classes: ' + myObj.custom\_classes);

// console.log('Images Processed Classes: ' + myObj.images\_processed);

// console.log('=========');

// console.log(myObj.images[0].classifiers[0]);

// console.log('=========');

// console.log(myObj.images[0].classifiers[0].classes[0]);

// console.log('=========');

// console.log(myObj.images[0].classifiers[0].classes[1]);

// console.log('=========');

// console.log(typeof(myObj.images[0].classifiers[0].classes[4]));

// console.log('=========');

//

// });