

Chapter 11: Finding Answers: Watson Discovery

[Return to Table of Contents](#)

This chapter is an update to the original chapter, where we used Retrieve and Rank to create a collection, process queries against that collection as both a straight retrieve and then later including the ranking capability as part of that service.

Retrieve and Rank has been replaced with Watson Discovery. This means that we have better exploration tools available to us and, as well, can do things like create the necessary configuration file interactively in the browser rather than only on our laptops. We can still do what we did in the past, we just have more options, now.

Concept Changes

When using Retrieve and Rank, we are much more aware of the underlying technology and work with converted documents and do significant manual work to train the Ranking system. A lot of code is written to handle document upload, document conversion, and training. All of that is now handled by the Web interface for Watson Discovery. We can still write code to perform all those steps, and this code is included in the tutorial. However, it is no longer necessary to do that work.

Training can take longer, because we need 50 queries in the system to train Discovery and the documents returned for each query need to have some of them rated as relevant and some as not relevant. This is, in practical terms, the work of a subject matter expert. For us to do that, we need to actually read each of the 12 documents we're going to use in this small example, so that we can apply the 'relevant/not-relevant' responses.

Process Changes

All of the work can be done through the Watson Discovery Web Interface. We will still write a query interface, which draws very heavily from the Watson Discovery News work done in Chapter 8.

Code Changes

Only the query interface needs to be written for both the server side and the browser side


Server Side

We need a "find" service

Client (Browser)

We need a "find" interface

Executing the Lab

- We are going to use the IBM Institute for Business Value documents located at this URL: <https://www-935.ibm.com/services/us/gbs/thoughtleadership/>
- There are 12 documents at this location. We want all 12 of them downloaded to our system and stored in the Chapter11/HTML/Documents/Source folder.
- Log into IBM Cloud (formerly IBM Bluemix) at
 - <https://console.ng.bluemix.net> (Americas)
 - <https://console.eu-gb.bluemix.net> (Great Britain)
 - <https://console.eu-de.bluemix.net> (Europe)
 - <https://console.au-syd.bluemix.net> (Asia-Pacific)
 - 
- Create, if you haven't already done so, a Watson Discovery Service

(this is the same service you would have used in Chapter 8)

- Select "Watson" from the left hand menu and then select Discovery from the set of displayed services.



- Click on Service Credentials and then copy your service credentials into your env.json file. In the following file, you'll see environmentID, configurationID, and collectionID. Those are retrieved from the Manage Data page for your collection by clicking on the Use Collection In API link. (this is explained later)
 - (env.json example)





```
"discovery": {
  "url":
    "https://gateway.watsonplatform.net/discovery/api",
  "password": "password-from-credentials",
  "username": "user-name-from-credentials",
  "version_date": "2017-11-07",
  "documentList": "documents.json",
  "source_path": "HTML/Documents/Source",
  "environmentID": "your-environment-id",
  "configurationID": "your-configuration-id",
  "collectionID": "your-collection-id"
```






- Then click on Launch Tool and Create a new Data Collection by clicking on the Plus sign.



- This will create a private data collection. You see in my example that a private data collection has already been created titled Z2C_Collection
- On the following page,
 - First, Edit the configuration and add in all 'enhancements'. This gives you sentiment, keyword, concepts, etc.

- 2nd, upload all of your documents from Chapter11/HTML/Document/Source
 - 
- We now need to Train Watson Discovery. What's that mean? It means that we need to execute a minimum of 50 different queries against our collection and rank the results from each query. We rank results by taking one of 3 actions:
 - Click on "Relevant" if the returned document is relevant to the query
 - Click on "Not Relevant" if the returned document is particularly not relevant to the query.
 - Not all documents need to be marked, but all queries should be ranked.
 - There is a sample list of 50 queries at the end of this readme file.
- Click on the query icon on the left hand navigator
 - 
- And then click on Train Watson in the top right hand corner
 - 
- Which will bring you to the Training Query page. On this page, you'll type or paste in queries and also rank those queries
 - 
 - In the middle of the page, you can see three boxes, all of which have a line through them. As you add more queries and provide more ranking and more ranking diversity, these boxes will go from a bold text to what you see here – greyed out with a line through them. when all three look like this, you have completed the minimum required amount of ranking. Watson will now start training. Wait an hour or so after training starts before trying any queries.
 - At the bottom of the page, you can see the query 'What financial benefits do industry leaders get from cloud and AI' and a button marked "Rank" to the right.

- To Rank a query, click on the Rank button, which will take you to a page similar to the following:
 - 
 - Here, you'll tag documents as "Relevant" or "Not Relevant". Note while not all documents are tagged, all queries should be ranked.
- You can test your collection by going to the Query Page and submitting natural language queries.
 - click on the query icon on the left side and select your collection
 - 
 - click on "Search for documents" and type in a natural language query
 - Click on the "Run Query" button at the bottom of the page
 - 
 - explore your results

Creating a Web interface:

- we'll reuse some of the UX work we did in Chapter 8
- we have to create a web page to hold the query and it's results
- we have to create a javascript file to manage the user experience
- We have to create a single service to execute the query

##Creating the web service:

- The discovery_complete.js file has the following routine in it (there are others which you can explore which support creating an administrative interface on the web for Watson Discovery)

```
/**
 * find queries a specific collection in a
 * specific environment for the currently identified
```


```

for the userid and password stored in the
env.json file
*
* @param {object} req - nodejs object with
the request information
* req.body.queryString has the query details
* @param {object} res - nodejs response
object
* @param {object} next - nodejs next object
- used if this routine does not provide a
response
*/
exports.find = function(req, res, next)
{
  let _method = 'find';
  discovery.query({ environment_id:
config.discovery.environmentID, collection_id:
config.discovery.collectionID,
natural_language_query:
encodeURIComponent(req.body.queryString),
passages: true },
    function (err, response) {
      if (err)
      {console.log('error:', err);
      res.send({'result': 'error', 'message':
err.message});}
      else
      {res.send({'result': 'success', 'data':
response});}
    });
}



```

- The environment and collection id are gathered programatically from the Watson Discovery Service. If you don't want to get the data programatically, you can copy it from the Watson Discovery Service web interface by going to the displayed page and clicking on "Use this collection in API" and then storing the displayed

information someplace like your env.json file.

- 
- The web page is very simple. We have a query input file locked to the bottom of the page (we did this in Chapter 10) and will display the results above the input field.

```
<div class="container">
  <div class="title">
    <h1>Chapter 11: Watson
Discovery</h1>
  </div>
  <div class="scrollingPane">
    <div class="col-lg-12" id="discovery">
    </div>
  </div>
  <input id="textInput" class="input
showfocus wide" width="100%",
        placeholder="Type a query here"
type="text", onkeypress="detectKey(event,
'doQuery')">
  </div>
```

- You may remember the 'detectKey' routine from Chapter 10. We've extended it (z2c_utilities.js) by adding a new case statement for this chapter. This enables a person to type a query and execute it simply by pressing the enter key on their keyboard.
 - the initial page looks like this:
 - 
 - Pressing the enter key returns the following scored results with most likely results first
 - 
 - Clicking on a result line expands the accordian (you'll

remember this from Chapter 8) and shows you the text that Watson Discovery stored and, if you kept the documents in your HTML/Documents/Source then clicking on "View Original Document" will display the original PDF in a new window.

- 

#Browser Javascript

There are only three routines we create for the browser:

- initiateDiscovery
- doQuery
- display_find
- *initiateDiscovery*
- This routine simply sets the query source and query result target html objects for later use

```
let q_source;
let q_target;

// initialize the page
function initiateDiscovery()
{
    let _method = 'initiateDiscovery';
    q_source = $("#textInput");
    q_target = $("#discovery");
}
```

- *doQuery*
- This routine sets up and executes the asynchronous call the the previously described find service on the server and displays the

'loading' gif we first introduced in chapter 9 (Visual Recognition)

```
function doQuery()
{
    let _method = 'doQuery';
    q_target.empty(); q_target.append("<center><img src='icons/loading.gif' /></center>")
    let _options = {};
    _options.queryString=q_source.val();
    q_source[0].value = "";
    console.log(_method, _options);
    postIt('/discovery/find', _method,
display_find, q_target, _options);

}
```

- *display_find*
- this routine extracts part of the returned data, sorts it and then formats the results.
- The opening of the routine creates a temporary array and extracts portions of the returned result set into that array. At the end of the for loop, the array is sorted, based on the document score. The sort is significantly faster with the smaller amount of data to move around memory.

```
let tmpArr = new Array();
for (each in _res.data.results)
{(function(_idx, _arr)
{
    let tmpObj = {};
    tmpObj.extracted_metadata =
_arr[_idx].extracted_metadata;
    tmpObj.id = _arr[_idx].id;
```

```

        tmpObj.sentiment =
_arr[_idx].enriched_text.sentiment.document;
        tmpObj.emotion =
_arr[_idx].enriched_text.emotion.document.emotion;
        tmpObj.result_metadata =
_arr[_idx].result_metadata;
        tmpObj.html = _arr[_idx].html;
        tmpArr.push(tmpObj);
    })(each, _res.data.results);
}

tmpArr.sort(function(a,b){return
(b.result_metadata.score > a.result_metadata.score) ?
1 : -1;});

```

- The second for loop formats the output. Document sentiment information is used in an identical manner to Chapter 9. We are using the same accordian structure and CSS as in Chapter 9 and, in fact, nearly identical code.

```

    for (each in tmpArr)
    {
        (function(_idx, _array)
        { console.log('_array['+_idx+'] id is: ',
_arr[_idx].id+' name:
'+_array[_idx].extracted_metadata.title+' score:
'+_array[_idx].result_metadata.score);
            var _hdr = "find_"+_idx+"_header";
            var _bdy = "find_"+_idx+"_content";
            var _sentiment_icon;
            if (_array[_idx].sentiment.label ==
"positive") {_sentiment_icon = '<td></td>';}
            if (_array[_idx].sentiment.label ==
"neutral") {_sentiment_icon = '<td></td>';}
            if (_array[_idx].sentiment.label ==

```

```

"negative") {_sentiment_icon = '<td></td>';}
        var _link = '<tr><td>Link: </td><td><a
href="Documents/Source/'+_array[_idx].extracted_metad
ata.filename+'"' target="_blank"><b>View Original
Document</b></a>';
        // since we have this information, let's
display the article summary text in the accordian
window.

        // we're using a table format for display
purposes, so use both columns to display the text
        var _text = '<tr><td
colspan="2">'+_array[_idx].html+'</td></tr>';
        var _hdr_html = '<div class="acc_header
off" id="'+_hdr+'"' target="'+_bdy+'"'
onClick="accToggle('\newsfeed\' ,
\'find_'+_idx+'\');"><table><tr>'+_sentiment_icon+"
<td>"+_array[_idx].extracted_metadata.title+'<br/>Sco
re: <b>'+_array[_idx].result_metadata.score+'</b>
</td></tr></table></div >';
        var _bdy_html = '<div class="acc_body
off" id="'+_bdy+'"'><table>'+_link+_text+'</table>
</div>';
        _target.append(_hdr_html+_bdy_html);
    })(each, tmpArr);
}

```

#50 Sample Queries

- What financial benefits do industry leaders get from cloud and AI
- what kinds of technologies or solutions are top industry performers running in the cloud
- what does it mean to be a high-performing cloud organization
- what's the relationship between cloud and cognitive
- how will these emergent technologies affect a Media company business model
- what should I do as a media company to leverage cognitive

- can you give me an example of how cognitive technology has been used by media industry
- who are the disrupters most affecting Media industry
- how does consumer device selection affect media companies
- What are the key drivers for change in Media industry
- can you give me a roadmap to cognitive?
- Can you give me examples of businesses using cognitive computing
- How can I determine my readiness to gain value from cognitive computing
- Is there a business case for cognitive computing
- What's the business case for cognitive computing
- how can cognitive computing influence sales
- How can cognitive computing influence marketing
- Is there any difference in how Gen Z engages between growth markets and mature markets
- What do companies need to do to influence Gen Z
- what distinguishes a Gen Z from others
- What do Gen Z expect from industry and government
- How can a non-profit get started with AI or Cognitive technologies
- How can non-profit demonstrate greater value to their investors
- What can government do to increase high-demand skills
- who's responsible for skill development
- What skills are government organizations trying to hire
- what are some of the key challenges faced by governments today?
- Is there a measurable difference in AI usage between market leaders and market followers
- how are different industries leveraging image processing and AI
- how is machine learning affecting supply chain logistics
- What technologies are influencing supply chains
- how important is iot to supply chain
- what defines the ability of a company to gain access to consumer information
- how can insurance increase their access to customer information

- how are banks getting started with Blockchain
- how are fintechs using blockchain
- how are financial institutions using blockchain
- Do non-profits use AI
- How do I target Gen Z?
- How is Gen Z different?
- How do I leverage AI
- how is cognitive computing relevant to government
- How is cognitive computing relevant to non-profits
- what are the top banking trends
- what are the top automotive industry trends
- What are the top retail industry trends
- can you show me how visual recognition solves business problems
- how do I use data science
- how is Cloud driving business transformation
- how is AI driving business transformation