Redbooks
ibm.com/redbooks

Building Cognitive Applications with IBM Watson Services: Volume 7 Natural Language Understanding

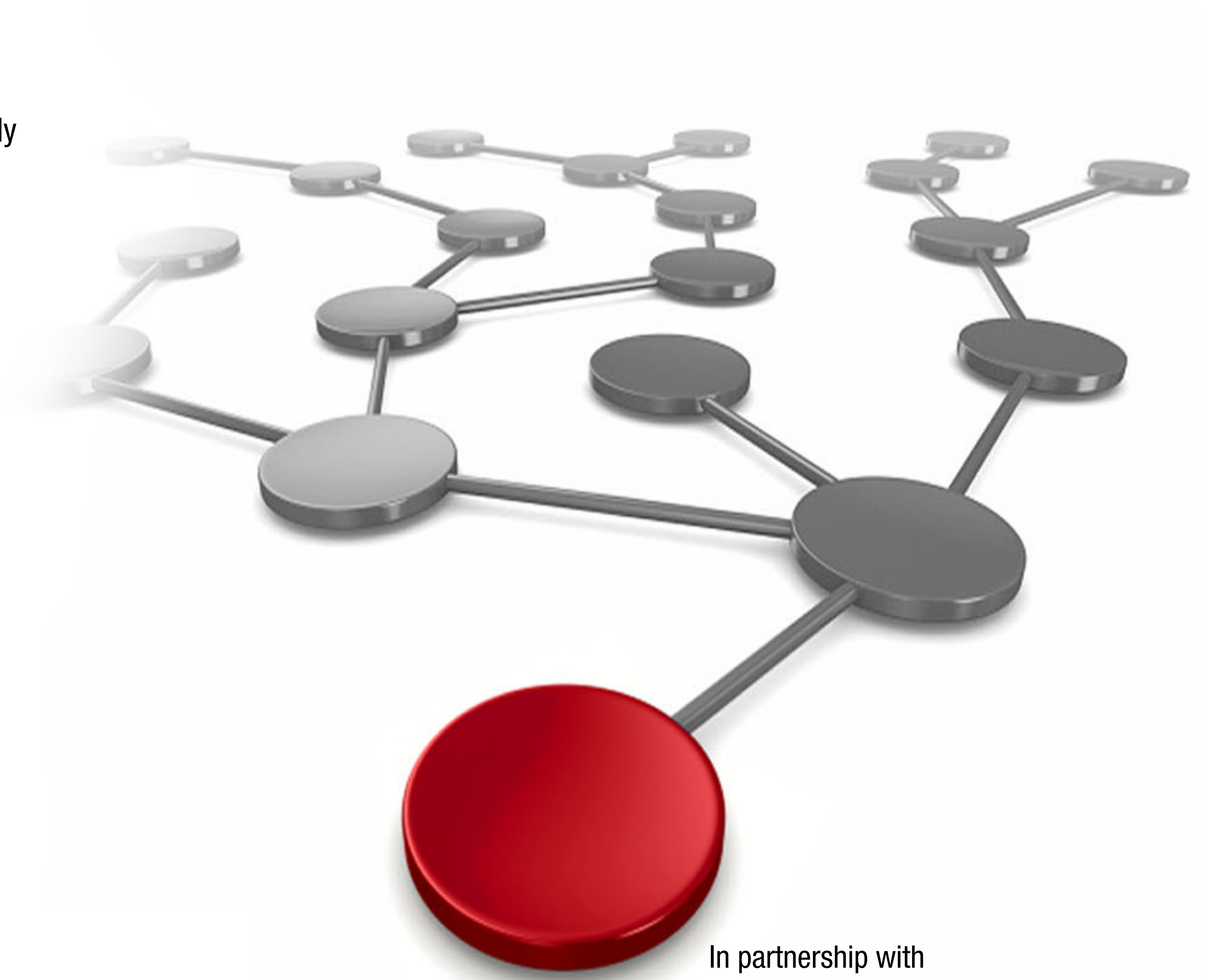
Sebastian Vergara

Mohamed El-Khouly

Mariam El Tantawi

Shireesh Marla

Lak Sri



In partnership with
IBM **Skills Academy Program**

IBM

Redbooks

International Technical Support Organization

Building Cognitive Applications with IBM Watson Services: Volume 7 Natural Language Understanding

June 2017

SG24-8398-00

First Edition (June 2017)

This edition applies to IBM Watson services in IBM Bluemix.

Contents

Notices

Preface

The *Building Cognitive Applications with IBM Watson Services* series is a seven-volume collection that introduces IBM® Watson™ cognitive computing services. The series includes an overview of specific IBM Watson® services with their associated architectures and simple code examples. Each volume describes how you can use and implement these services in your applications through practical use cases.

The series includes the following volumes:

- *Volume 1 Getting Started*, SG24-8387
- *Volume 2 Conversation*, SG24-8394
- *Volume 3 Visual Recognition*, SG24-8393
- *Volume 4 Natural Language Classifier*, SG24-8391
- *Volume 5 Language Translator*, SG24-8392
- *Volume 6 Speech to Text and Text to Speech*, SG24-8388
- *Volume 7 Natural Language Understanding*, SG24-8398

Whether you are a beginner or an experienced developer, this collection provides the information you need to start your research on Watson services. If your goal is to become more familiar with Watson in relation to your current environment, or if you are evaluating cognitive computing, this collection can serve as a powerful learning tool.

This IBM Redbooks® publication, Volume 7, introduces the Watson Natural Language Understanding service. This service is a collection of text analysis functions that derive semantic information from your content. This book includes a basic description of several of the Natural Language Understanding service features and provides sample code snippets to demonstrate their use. This book includes an example of an application that integrates the Watson Natural Language Understanding service with the Watson Personality Insights and Insights for Twitter services to create a simple application to analyze Tweets from a Twitter handle. You can develop and deploy the sample applications by following along in a step-by-step approach and using provided code snippets. Alternatively, you can download an existing Git project to more quickly deploy the application.

Authors

This book was produced by a team of specialists from around the world, working in collaboration with the IBM International Technical Support Organization.

Sebastian Vergara is an Expert Certified Architect in IBM Sales & Distribution, IBM Uruguay. His areas of expertise include cloud computing, DevOps, Design Thinking, and cognitive computing. He has over 8 years of experience in the IT industry. Sebastian led several projects to design and build cognitive solutions, such as the development of a transactional virtual assistant for an international bank and a cognitive chatbot for a major pharmaceutical company in Latin America that uses Watson Natural Language Classifier, Text to Speech, Natural Language Understanding, Visual Recognition, and other Watson technologies. Sebastian teaches at the Engineering College in the Universidad de la República Uruguay (UdelaR) where he introduces students to architecture and design, integration, cloud computing, and trending technologies.

Mohamed El-Khouly is a Certified IT Specialist in IBM Cloud Services, IBM Egypt. Mohamed has 15 years of experience in various roles in IT Services, including Software Development, Project Management, Testing, and Services Delivery. Mohamed currently focuses on cloud and analytics services, which include IBM Bluemix®, IBM SPSS® Modeler, IBM SPSS Statistics, and IBM Cognitive services.

Mariam El Tantawi is a Certified IT Specialist in Actualizing IT Solutions, IBM Egypt. Mariam is a Senior Software Developer; her areas of expertise include cloud computing, predictive analytics, text analytics, and cognitive computing. She has over 4 years of experience in the IT industry. Mariam participated in and led several projects to design and build cognitive solutions, such as an application that helps graduates to find and explore career paths and career-related hiring entities and jobs, depending on their field of study. Mariam was part of the team that developed a question answering system for Dubai Road and Transport Authority (RTA). The solution enables users to ask questions related to the services provided by RTA in their native language. It integrates several Watson services, such as Conversation, Text to Speech, and Speech to Text. Mariam is a frequent speaker at IBM conferences and she teaches university courses sponsored by IBM Skills Academy Programs in the areas of predictive analytics and business analytics. Mariam holds two IBM patents.

Shireesh Marla is a Mobile Solution Architect in IBM Global Business Services®, India. His areas of expertise include IBM MobileFirst™, Android, IBM Bluemix cloud development platform, IBM Watson Analytics, and Cognitive Computing. He has over 13 years of experience in the IT industry and has been involved in mobile application technologies and cloud computing technologies. Shireesh has been part of the team in building a bot for a digital satellite network entertainment company in South Africa. It uses IBM Watson Natural Language Classifier, Speech to Text, and Text to Speech APIs.

Lak Sri currently serves as a Program Director in IBM developerWorks®, part of the IBM Digital Business Group organization. Lak leads innovation in the developer activation space. He was the Technical Leader for the *Building Cognitive Applications with IBM Watson Services* Redbooks series. Lak led the development of the IBM Cloud Application Developer Certification program and the associated course. Earlier he worked as a Solution Architect for Enterprise Solutions in Fortune 500 companies using IBM Tivoli® products. He also built strategic partnerships in education and IBM Watson IoT™. Lak is an advocate and a mentor in several technology areas, and he volunteers in planning and supporting local community programs.

The project that produced this publication was managed by **Marcela Adan**, IBM Redbooks Project Leader, ITSO.

Thanks to the following people for their contributions to this project:

Swin Voon Cheok
Ecosystem Development (EcoD) Strategic Initiative, IBM Systems

Juan Pablo Napoli
Skills Academy Worldwide Leader, Global University Programs

Teja Tummalapalli
IBM Digital Business Group

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Basics of Watson Natural Language Understanding service

This chapter introduces the IBM Watson Natural Language Understanding service. The Natural Language Understanding service is a collection of text analysis functions that derive semantic information from your content. You can input text, HTML, or a public URL, and leverage sophisticated natural language processing techniques to get a quick high-level understanding of your content and obtain detailed insights.

This chapter provides basic information about how to use the service features, provides usage examples, and includes simple code examples in the following technologies:

- Node.js
- Java
- Node-RED

Note: A runtime environment that is specific to each technology is required in order to run the snippets provided in this chapter. The purpose of the snippets is to serve as code examples for future reference. Therefore, documentation about the installation and configuration of the technology-specific runtime environment is not included.

The following topics are covered in this chapter:

- Natural Language Understanding overview
- Service features
- Migrating from AlchemyLanguage to Natural Language Understanding
- References

1.1 Natural Language Understanding overview

With Natural Language Understanding, developers can analyze semantic features of input text and extract metadata from content, such as categories, concepts, emotion, entities, keywords, metadata, relations, semantic roles, and sentiment. With custom annotation models developed using IBM Watson Knowledge Studio, you can further customize the service to identify domain-specific entities and relations in your content.

Natural Language Understanding can be useful in many scenarios that demand rapid analysis of unstructured text without requiring in-depth natural language processing expertise. For example, you can monitor sentiment and emotion in customer support chat transcripts, or you can quickly categorize blog posts and sort them based on general concepts, keywords, and entities.

1.1.1 How it works

Figure 1-1 shows a high-level flow of the Natural Language Understanding service.

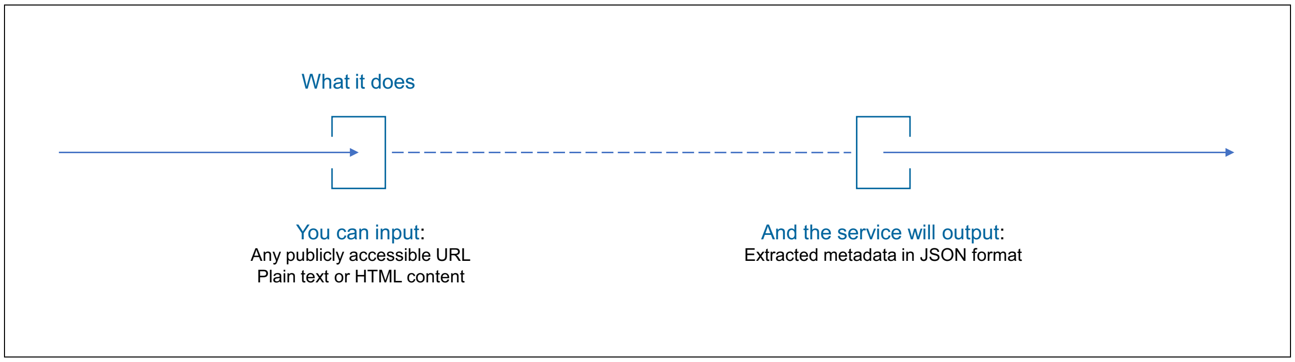

What it does
You can input:
Any publicly accessible URL
Plain text or HTML content
And the service will output:
Extracted metadata in JSON format

Figure 1-1 Natural Language Understanding high-level flow

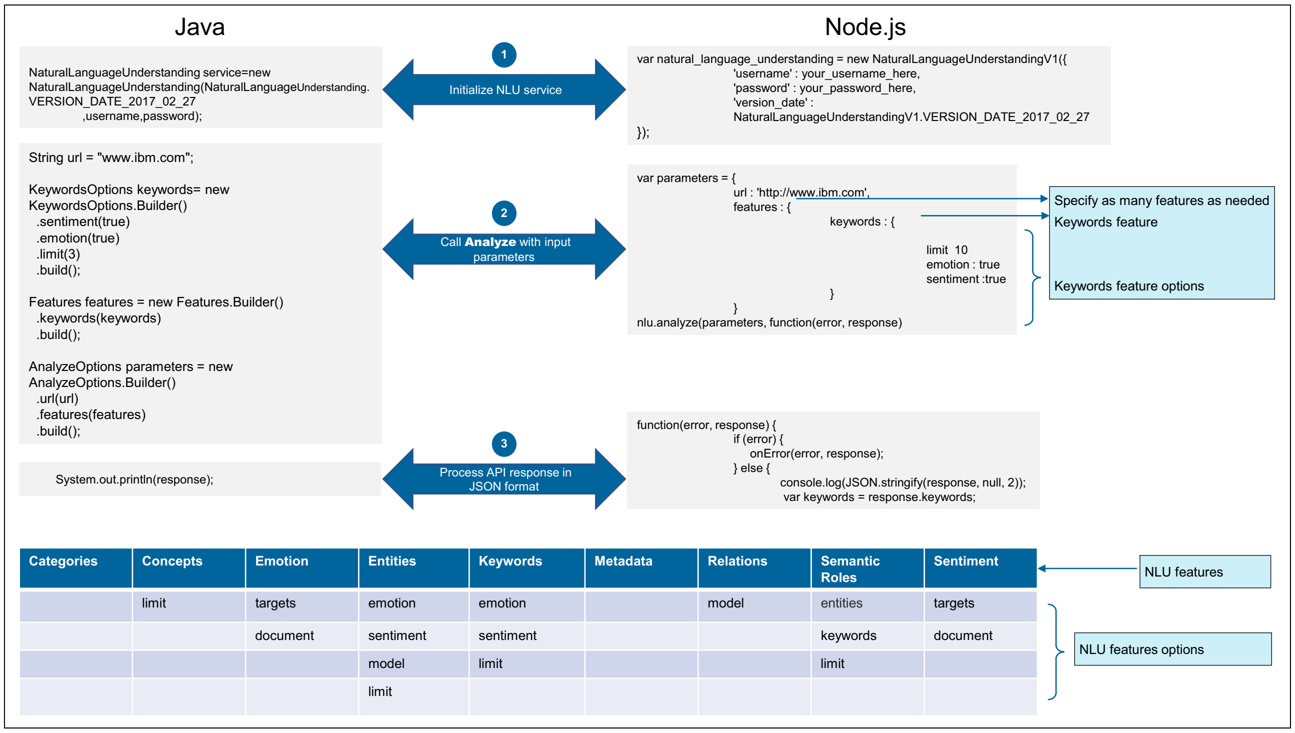
The flow is as follows:

1. You input the following types of text to be analyzed:
   - Any publicly accessible URL
   - Plain text or HTML content
2. The service will output this information:
   - Extracted metadata in JSON format

Figure 1-2 shows an overview of code snippets, in Java and Node.js, to call the API and process the response in these steps:

1. Initializes the Natural Language Understanding service instance, passing the credentials (`username` and `password`) and version.
2. Calls the `/Analyze` endpoint with the text, HTML, or a public URL to be analyzed; the Natural Language Understanding service features indicate the text analysis functions that the API should perform (the `Keywords` feature is shown in Figure 1-2 as an example). You can also specify the options for each feature.
3. Processes the API response returned in JSON format.

The table at the bottom of Figure 1-2 shows the Natural Language Understanding (NLU) service features and the options for each feature.



```
NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.
VERSION_DATE_2017_02_27
            ,username,password);
```

```
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
                    'username' : your_username_here,
                    'password' : your_password_here,
                    'version_date' :
                    NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
```

```
String url = "www.ibm.com";

KeywordsOptions keywords= new
KeywordsOptions.Builder()
  .sentiment(true)
  .emotion(true)
  .limit(3)
  .build();

Features features = new Features.Builder()
  .keywords(keywords)
  .build();

AnalyzeOptions parameters = new
AnalyzeOptions.Builder()
  .url(url)
  .features(features)
  .build();
```

```
var parameters = {
                    url : 'http://www.ibm.com',
                    features : {
                                        keywords : {

                                                            limit  10
                                                            emotion : true
                                                            sentiment :true
                                        }
                    }
nlu.analyze(parameters, function(error, response)
```

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

```
function(error, response) {
                    if (error) {
                        onError(error, response);
                    } else {
                            console.log(JSON.stringify(response, null, 2));
                         var keywords = response.keywords;
```

Categories	Concepts	Emotion	Entities	Keywords	Metadata	Relations	Semantic Roles	Sentiment
	limit	targets	emotion	emotion		model	entities	targets
		document	sentiment	sentiment			keywords	document
			model	limit			limit	
			limit					

Figure 1-2 Calling the NLU service and processing response

1.1.2 Supported languages

Table 1-1 shows the features that are supported by each language. For updated information, see the Supported languages web page.

Table 1-1 Features supported by each language

	Sentiment	Semantic roles	Relations[a]	Metadata	Keywords	Entities[a]	Emotion	Concepts	Categories
Arabic	X		X	X					X[c]
English	X	X	X	X	X	X	X	X	X
French	X		X[b]	X	X	X			X[c]
German	X		X[b]	X	X	X			
Italian	X		X[b]	X	X	X			X[c]
Japanese			X						
Portuguese	X		X[b]	X	X	X			X[c]
Russian	X			X	X	X			
Spanish	X	X	X	X	X	X		X	X[c]
Swedish				X	X	X			

a. You can build Watson Knowledge Studio custom models for entities and relations in English, French, German, Italian, Portuguese, and Spanish. You can use some of these languages in Natural Language Understanding or you can customize the models.
b. These languages are supported only through custom models in IBM Watson Knowledge Studio.
c. These languages are supported in the public service, but not in Bluemix Dedicated.

You can indicate the language to use for analysis with the ISO 639-1 code. This code overrides automatic language detection performed by the service. Valid codes are as follows:

ar	Arabic
en	English
fr	French
de	German
it	Italian
ja	Japanese
pt	Portuguese
ru	Russian
es	Spanish
sv	Swedish

1.1.3 Authentication

You authenticate to the Natural Language Understanding service with Basic Authentication in each request. To get the `username` and `password`, you must create a service instance and retrieve the credentials. For information, see Chapter 2, “Creating a Natural Language Understanding service in Bluemix” on page 59.

1.2 Service features

To use the Natural Language Understanding (NLU) service, send API requests to the `Analyze` endpoint with the input text, HTML, or a public URL, specify one or more of the supported service features, and specify the options for the features or accept the default options.

This section includes a basic description of several NLU service features and provides sample code snippets to demonstrate their use.

1.2.1 NLU Concepts

The NLU Concepts feature identifies high-level concepts that might not be directly referenced in the input text. Concept-related API functions understand how concepts relate. Concepts that are detected typically have an associated link to a *DBpedia* resource. See the following input and response examples.

Input

Text:

```
Machine learning is the science of how computers make sense of data using
algorithms and analytic models.
```

Response

Concepts tags:

- Computer
- Machine learning
- Artificial intelligence
- Computer science
- Alan Turing
- Scientific method
- Psychology
- Learning

Use case example: Clustering articles

Concepts tagging allows you to perform high-level analysis of the content. This feature can help you to cluster news articles based on concepts, and study or analyze articles associated with specific concepts. A use case might be the extraction of concepts from an online article by using, for example, the following URL as input to the API:

`http://www.bbc.com/news/technology-38595480`

NLU Concepts flow

Figure 1-3 on page 6 shows the basic flow:

1. Input (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`), for authentication, and URL to the news article to be analyzed.
2. Processing (analyze the input text with the `Concepts` feature).
3. Response (returns a response in JSON format):
   - `Text`: Name of the concept.
   - `Relevance`: Score for the concept in the range of 0 - 1. A score of 1 means the concept is highly relevant; 0 means it is not relevant.
   - `dbpedia_resource`: Link to the DBpedia resource that is associated with the concept.

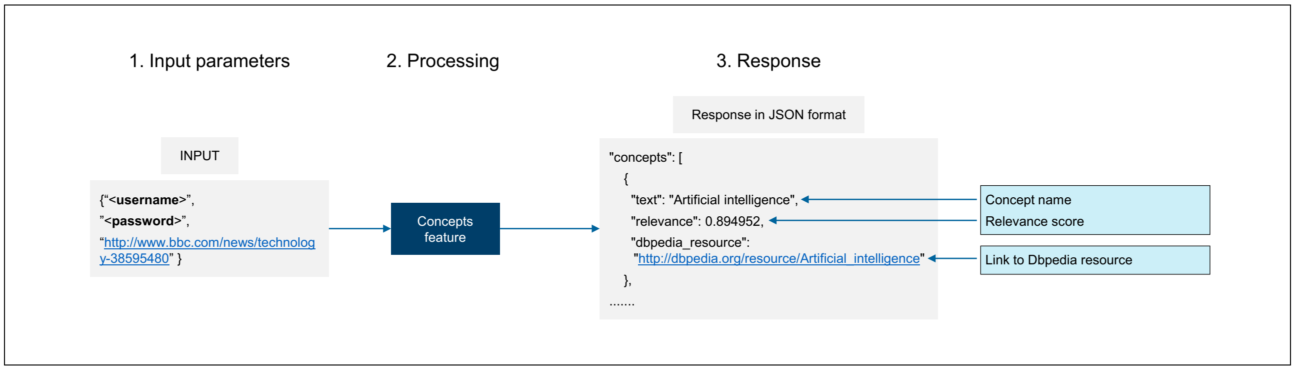


Figure 1-3 Concepts flow

NLU Concepts snippets

This section includes sample snippets to illustrate the use of the NLU Concepts feature in Node.js, Java, and Node-RED.

This example shows how to extract concepts from the following BBC article:

`http://www.bbc.com/news/technology-38595480`

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-1 shows the `nlu-concepts.js` snippet.

Example 1-1 Snippet: nlu-concepts.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url : 'http://www.bbc.com/news/technology-38595480',
   features : {
      concepts : {
         'limit' : 50
      }
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var concepts = response.concepts;
      // process the array of concepts
   }
});
```

Java snippet

Example 1-2 shows the `nluconcepts.java` snippet.

Example 1-2 Snippet: nluconcepts.java

```
/**
 * Created on 29-04-2017.
 */


import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.Features;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalyzeOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalysisResults;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.ConceptsOptions;



public class nluconcepts {

    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The URL to be analyzed for concepts
        String url = "http://www.bbc.com/news/technology-38595480";

        //The concepts objects, which will collect 50 concepts from the text
        ConceptsOptions concepts = new ConceptsOptions.Builder().limit(50).build();


        //Features object to hold analysis features
        Features features = new Features.Builder().concepts(concepts).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().url(url).features(features).build();

        //the top 50 concepts found in the analysis
        AnalysisResults response = service.analyze(parameters).execute();

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

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4).

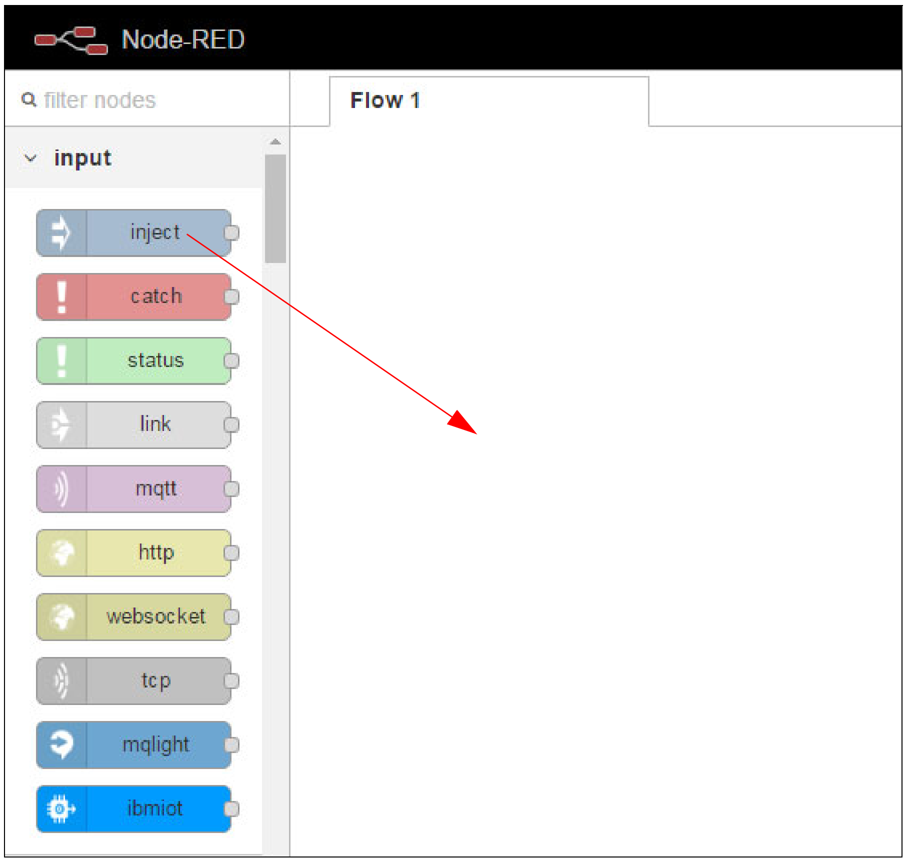
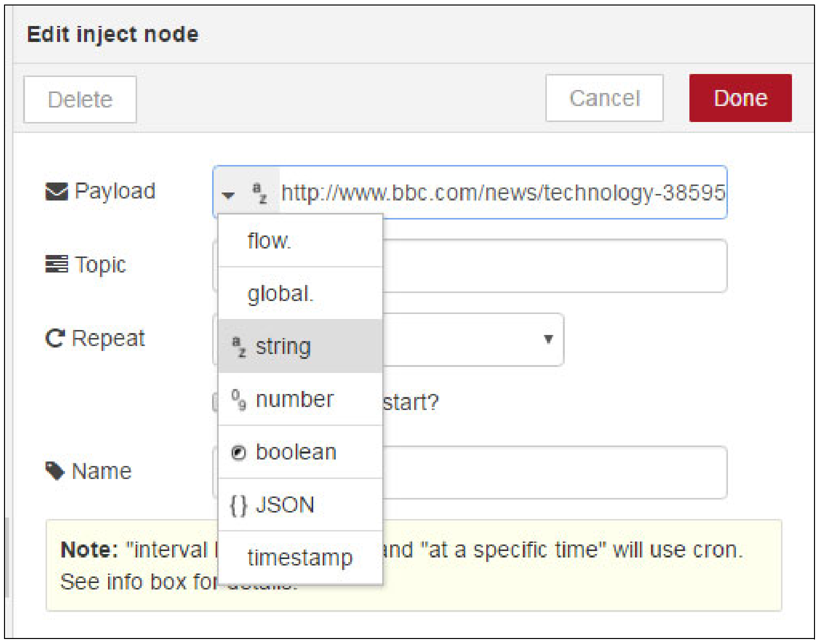


Figure 1-4 Drag inject node to canvas

2. Edit the `inject` node (Figure 1-5). Select **string** from the Payload pull-down and enter the URL to the document you want to analyze. In this example, the URL is `http://www.bbc.com/news/technology-38595480`.



Note: "interval ... and "at a specific time" will use cron. See info box for ...

Figure 1-5 Edit inject node dialog

3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6).

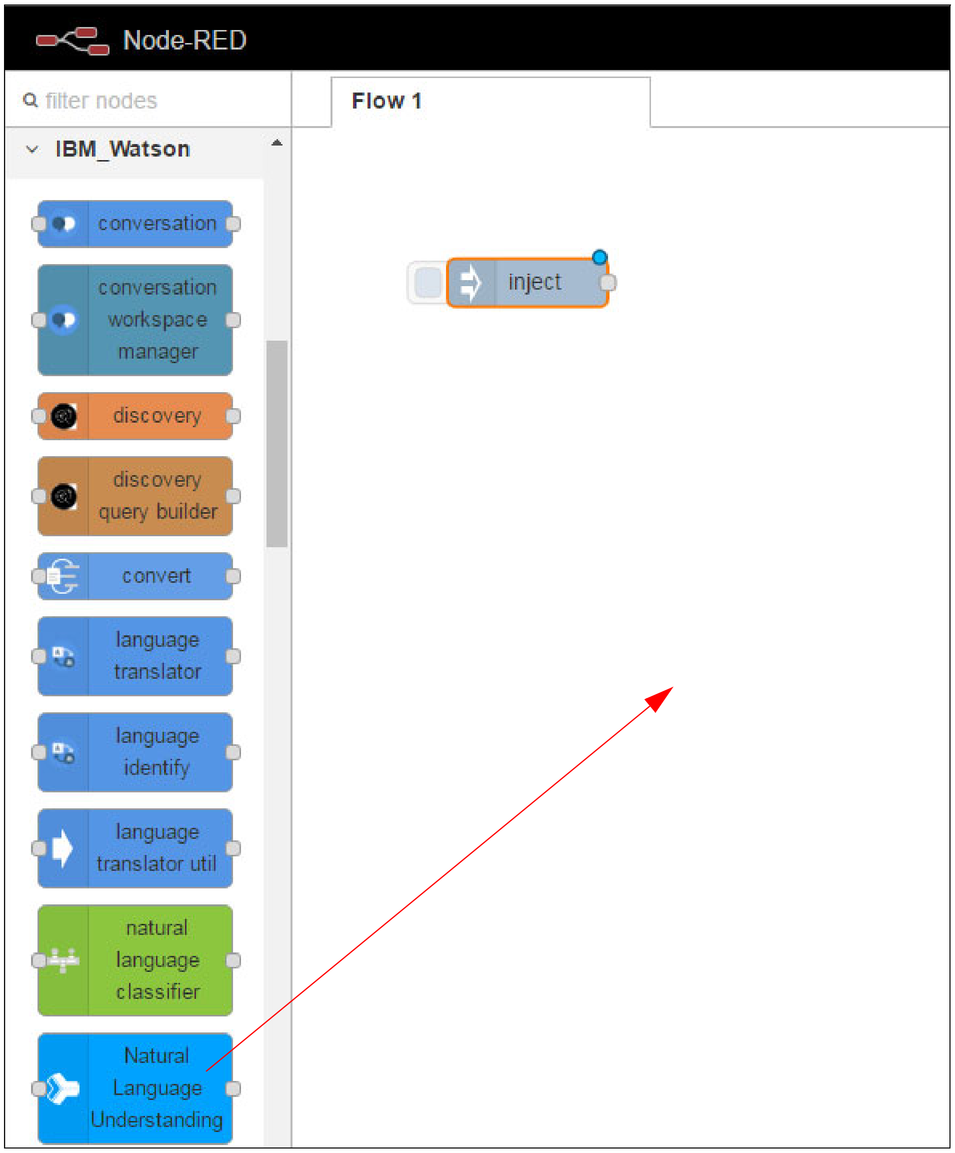
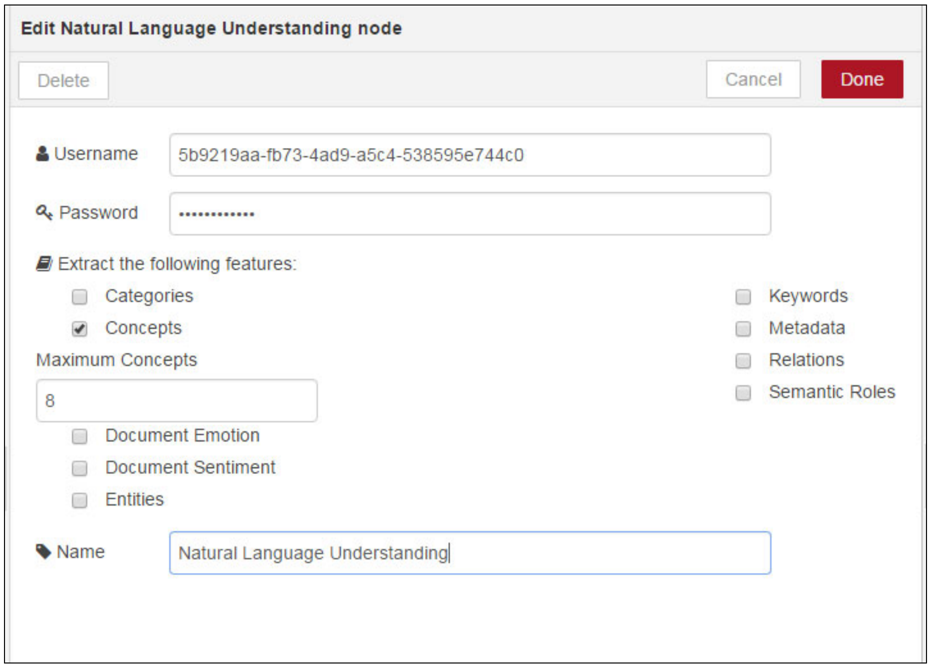


Figure 1-6 Drag Natural Language Understanding node to canvas

4. Edit the `Natural Language Understanding` node (Figure 1-7). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select the **Concepts** feature.


Edit Natural Language Understanding node

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0

Password: ••••••••••••

Extract the following features:

- [ ] Categories
- [✓] Concepts

Maximum Concepts: 8

- [ ] Document Emotion
- [ ] Document Sentiment
- [ ] Entities
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name: Natural Language Understanding

Figure 1-7 Edit Natural Language Understanding node dialog

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8).

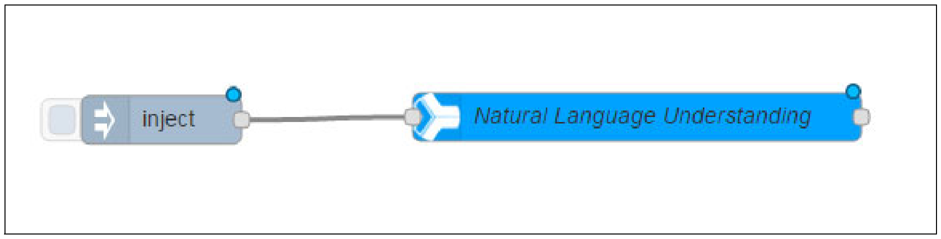


Figure 1-8 Connect the inject and Natural Language Understanding nodes

6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9).

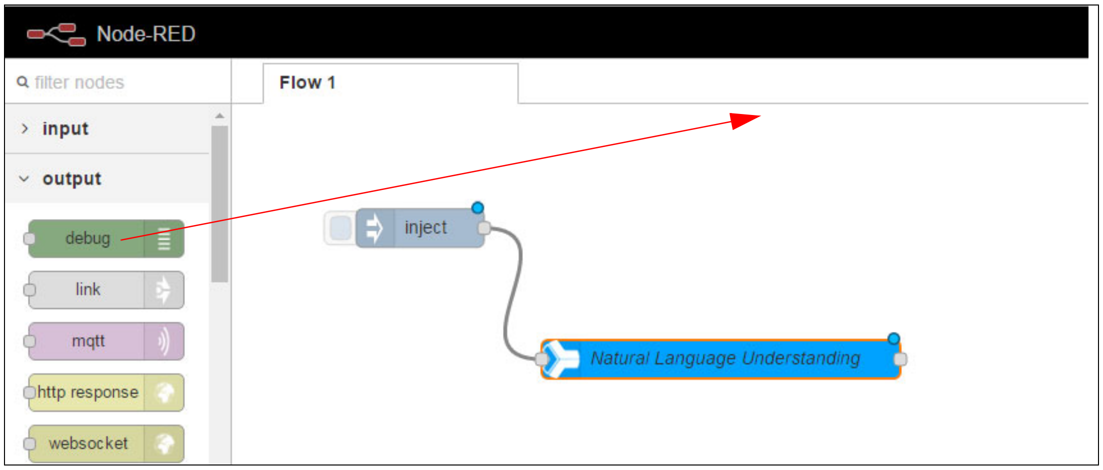
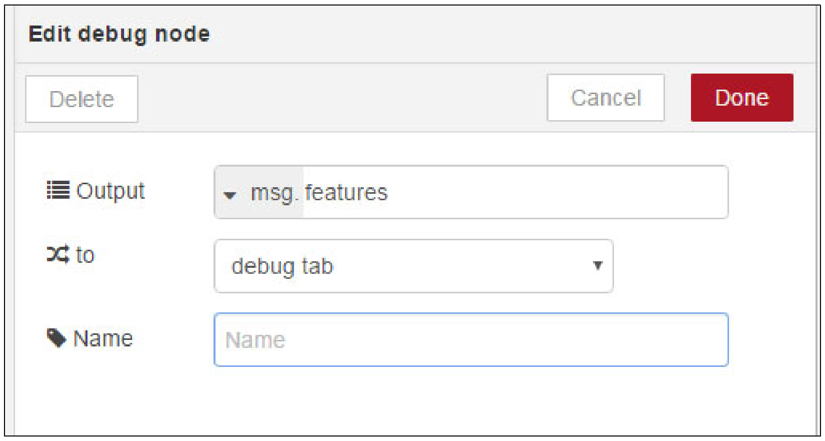


Figure 1-9 Drag the debug node to the canvas

7. Edit the `debug` node to match the configuration shown in Figure 1-10.



Output: msg. features
to: debug tab
Name: Name

Figure 1-10 Edit debug node dialog

8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11).

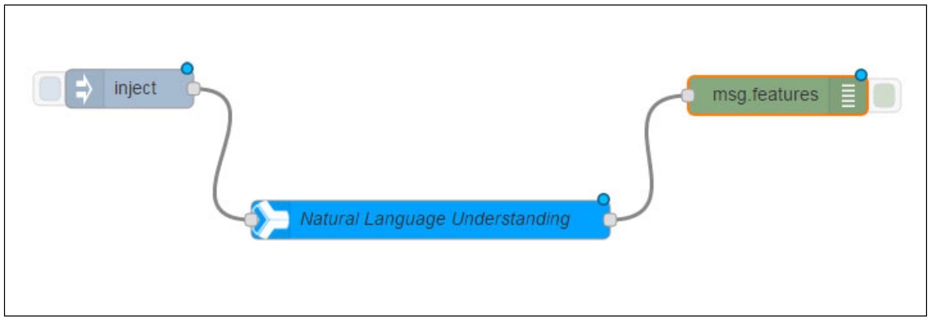


Figure 1-11 Connect the Natural Language Understanding and debug nodes

9. Click **Deploy** (Figure 1-12).

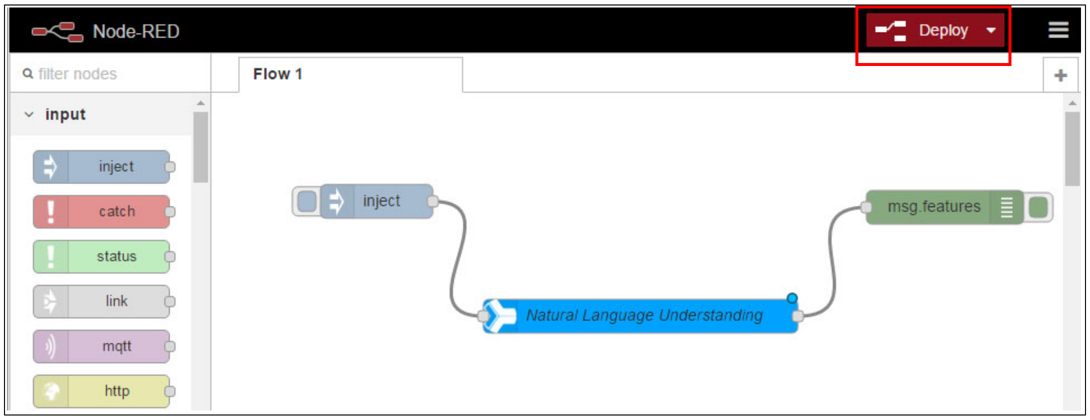


Figure 1-12 Deploy Node-RED flow

10. Click the button at the left side of the `inject` node to inject, into the flow, the document that is to be analyzed (Figure 1-13).

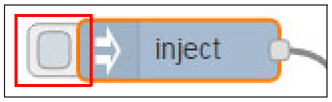
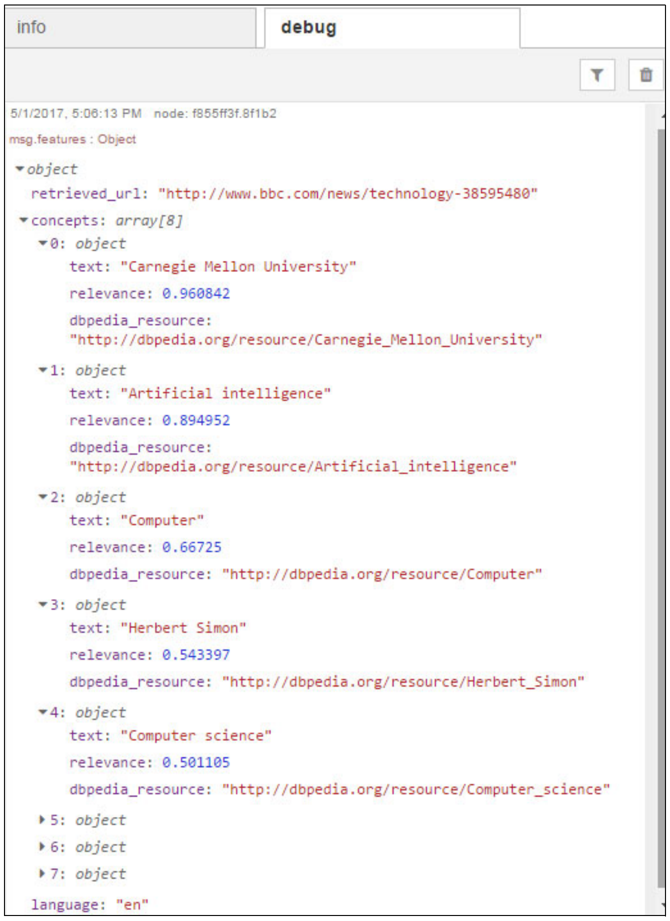


Figure 1-13 Button on the inject input node

11. Watch the debug tab. The output should be similar to Figure 1-14.



```
5/1/2017, 5:06:13 PM   node: f855ff3f.8f1b2
msg.features : Object
▾object
  retrieved_url: "http://www.bbc.com/news/technology-38595480"
 ▾concepts: array[8]
   ▾0: object
      text: "Carnegie Mellon University"
      relevance: 0.960842
      dbpedia_resource:
      "http://dbpedia.org/resource/Carnegie_Mellon_University"
   ▾1: object
      text: "Artificial intelligence"
      relevance: 0.894952
      dbpedia_resource:
      "http://dbpedia.org/resource/Artificial_intelligence"
   ▾2: object
      text: "Computer"
      relevance: 0.66725
      dbpedia_resource: "http://dbpedia.org/resource/Computer"
   ▾3: object
      text: "Herbert Simon"
      relevance: 0.543397
      dbpedia_resource: "http://dbpedia.org/resource/Herbert_Simon"
   ▾4: object
      text: "Computer science"
      relevance: 0.501105
      dbpedia_resource: "http://dbpedia.org/resource/Computer_science"
   ▸5: object
   ▸6: object
   ▸7: object
  language: "en"
```

Figure 1-14   NLU Concepts feature response on the debug tab

Example 1-3 shows the nodes and connections flow, exported in JSON format.

Example 1-3   NLU Concepts: Exported flow

```
[{"id":"a31695e2.13285","type":"tab","label":"Flow 
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language 
Un-derstanding","categories":false,"concepts":true,"maxconcepts":"8","doc-emotion"
:false,"doc-emotion-target":"","doc-sentiment":false,"doc-sentiment-target":"","en
tity":false,"entity-emotion":false,"entity-sentiment":false,"maxentities":"50","ke
yword":false,"keyword-emotion":false,"keyword-senti-ment":false,"maxkeywords":"50"
,"metadata":false,"relation":false,"semantic":false,"semantic-entities":false,"sem
antic-key-words":false,"maxsemantics":"50","x":454.9000244140625,"y":145.800003051
7578,"wires":[["587c390d.0c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a3
1695e2.13285","name":"","topic":"","payload":"http://www.bbc.com/news/technology-3
8595480","payloadType":"str","repeat":"","crontab":"","once":false,"x":244.9000091
5527344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debug"
,"z":"a31695e2.13285","name":"","active":true,"console":"false","complete":"featur
es","x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow from the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15).

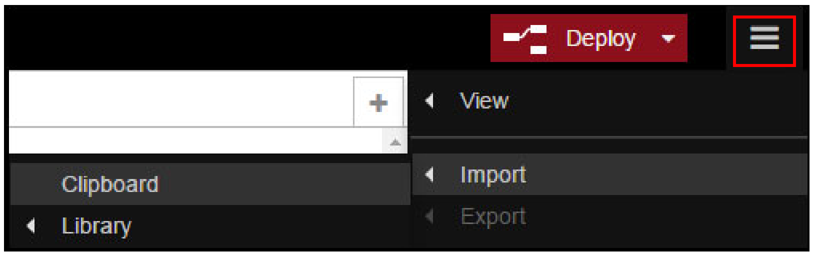


Figure 1-15 Importing Node-RED flow from the clipboard

1.2.2 NLU Emotion

The NLU Emotion feature detects anger, disgust, fear, joy, and sadness implied in text. It can analyze the overall emotional tone of the content or it can analyze emotion conveyed by specific target phrases. You can also enable emotion analysis for entities and keywords that are automatically detected by the service. See the following input and response examples.

Input

A document that includes customer reviews on a new smart device just released to the market.

Response

Emotion keys and score values (0.0 - 1.0), such as these:

- Anger score: 0.639028
- Disgust score: 0.009711
- Fear score: 0.037295
- Joy score: 0.00902
- Sadness score: 0.002552.

Use case example: Client emotion analysis

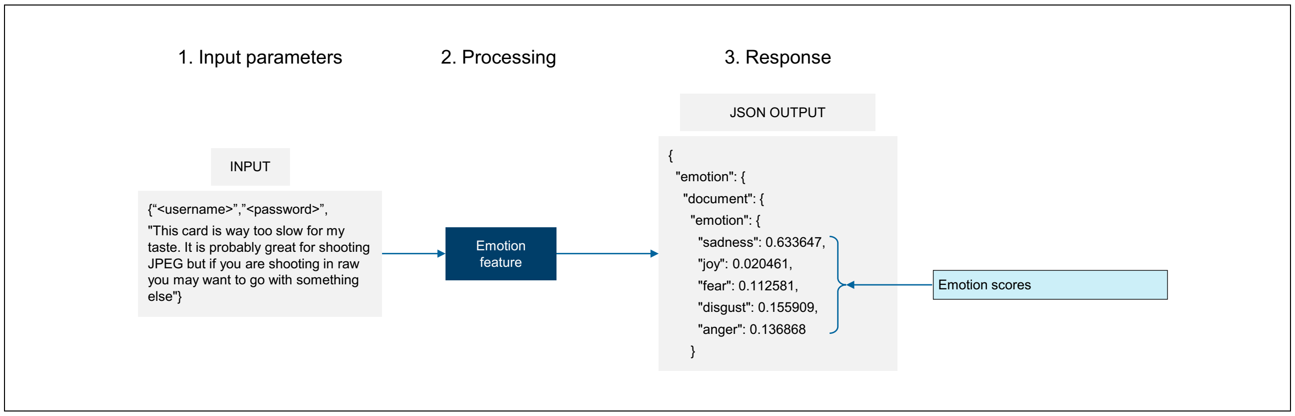
Emotion analysis can help call centers analyze the caller’s emotions from the caller’s reviews and feedback and then use the information to improve the services.

Emotion analysis can be used by an online company to analyze customer reviews and to understand customer feelings.

NLU Emotion flow

Figure 1-16 on page 16 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`), for authentication, and the text that will be analyzed for emotion.
2. Processing: The service analyzes emotion in the input text.
3. Response (returns response in JSON format):
   - `Document`: Object containing emotion analysis results for the entire document.
   - `Targets`: Array of objects containing emotion results for the targets.
   - `Emotion`: Emotion scores in the range of 0 - 1 for `sadness`, `joy`, `fear`, `disgust`, and `anger`. A score of 0 means the text does not convey the emotion; 1 means the text definitely carries the emotion.



{"<username>","<password>",
"This card is way too slow for my taste. It is probably great for shooting JPEG but if you are shooting in raw you may want to go with something else"}

```
{
  "emotion": {
    "document": {
      "emotion": {
        "sadness": 0.633647,
        "joy": 0.020461,
        "fear": 0.112581,
        "disgust": 0.155909,
        "anger": 0.136868
      }
```

Figure 1-16 Emotion flow

NLU Emotion snippets

This section includes sample snippets to illustrate the use of the NLU Emotion feature in Node.js, Java, and Node-RED.

This example shows how to analyze emotion in the following sample text:

```
This card is way too slow for my taste. It's probably great shooting JPEG but if
you are shooting Raw you may want to go with something else.
```

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-4 shows the `nlu-emotion_1.js` snippet.

Example 1-4 Snippet: nlu-emotion_1.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   text : "This card is way too slow for my taste. It's probably great shooting JPEG but if your
shooting Raw you may want to go with something else",
   features : {
      emotion : {}
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var docEmotions = response.emotion.document.emotion;
      // process object 'docEmotions' that contains properties 'anger', 'disgust', 'fear',
'joy', 'sadness'
   }
});
```

Java snippet

Example 1-5 shows the `nluemotion1.java` snippet.

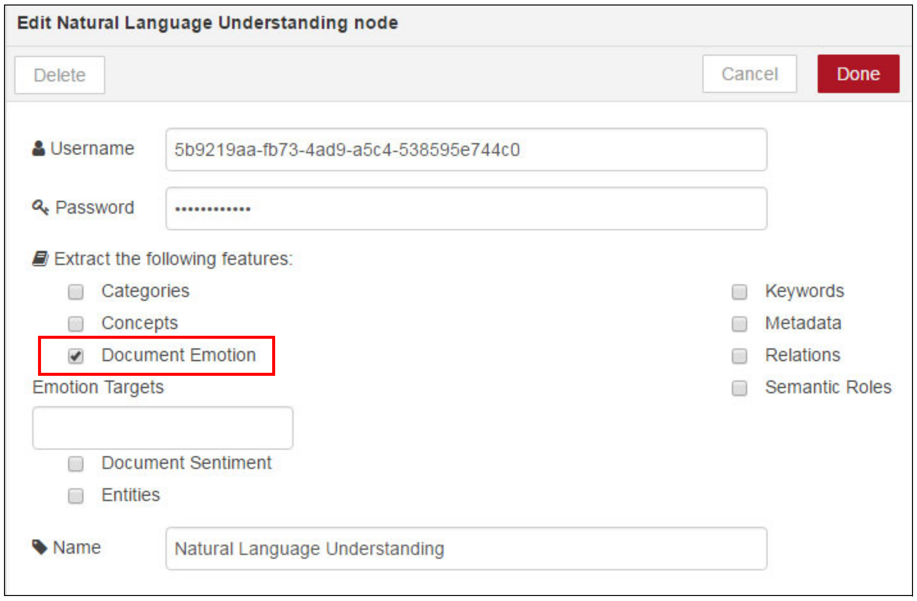
Example 1-5 Snippet: nluemotion1.java

```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.EmotionOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.Features;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalyzeOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalysisResults;


public class nluemotion1 {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The text to be analyzed
        String text = "This card is way too slow for my taste. It's probably great shooting JPEG
but if your shooting Raw you may want to go with something else";

        //The Emotions object
        EmotionOptions emotions = new EmotionOptions.Builder().build();


        //Features object to hold analysis features
        Features features = new Features.Builder().emotion(emotions).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().text(text).features(features).build();

        //The emotions returned from the analysis with their relative weight
        AnalysisResults response = service.analyze(parameters).execute();

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

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the text that you want to analyze: `This card is way too slow for my taste. It's probably great shooting JPEG but if you are shooting Raw you may want to go with something else.`
3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-17). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Document Emotion**.


Edit Natural Language Understanding node
Delete
Cancel
Done
Extract the following features:
Emotion Targets

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0
Password: ••••••••••••

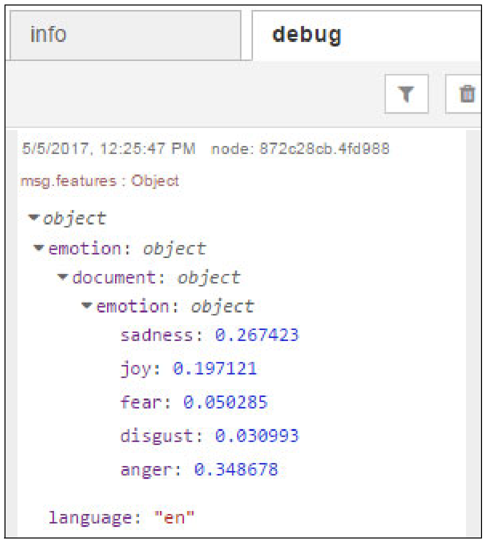
- [ ] Categories
- [ ] Concepts
- [x] Document Emotion
- [ ] Document Sentiment
- [ ] Entities
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name: Natural Language Understanding

Figure 1-17 Edit Natural Language Understanding node dialog

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the debug node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-18.



```
5/5/2017, 12:25:47 PM   node: 872c28cb.4fd988
msg.features : Object
▾object
 ▾emotion: object
   ▾document: object
     ▾emotion: object
         sadness: 0.267423
         joy: 0.197121
         fear: 0.050285
         disgust: 0.030993
         anger: 0.348678
  language: "en"
```

Figure 1-18   NLU Emotion feature response: The debug tab

Example 1-6 shows the nodes and connections flow, exported in JSON format.

Example 1-6   NLU Emotion: Exported flow

```
The Nodes and connections flow exported in JSON format.
[{"id":"a31695e2.13285","type":"tab","label":"Flow
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language
Un-derstanding","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion
":true,"doc-emotion-target":"","doc-sentiment":false,"doc-sentiment-target":"","en
tity":false,"entity-emotion":false,"entity-sentiment":false,"maxentities":"50","ke
yword":false,"keyword-emotion":false,"keyword-senti-ment":false,"maxkeywords":"50"
,"metadata":false,"relation":false,"semantic":false,"semantic-entities":false,"sem
antic-key-words":false,"maxsemantics":"50","x":454.9000244140625,"y":145.800003051
7578,"wires":[["587c390d.0c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a3
1695e2.13285","name":"","topic":"","payload":"I purchased this card from Best Buy
for around $69 to use in my new camcorder. It's perfect. The read/write speed is
exactly what I needed to record HD video and the storage capacity is enough for
several hours of video. I wish it had been a little cheaper when I bought it. I
see it's on sale now so get it while you can before the price goes back
up!","payloadType":"str","repeat":"","crontab":"","once":false,"x":244.90000915527
344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debug","z"
:"a31695e2.13285","name":"","active":true,"console":"false","complete":"features",
"x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.3 NLU Emotion (targets option)

The NLU Emotion feature with the targets option can detect emotions that are associated with targeted phrases, entities, or keywords. See the following input and response examples.

Input

A document that includes customer reviews on a new smart device was just released to the market with this target phrase:

```
Benefits, such as more personalized recommendations of things to watch.
```

Response

Emotion keys and score values (0.0 - 1.0) such as these:

- Anger score: 0.139028
- Disgust score: 0.009711
- Fear score: 0.037295
- Joy score: 0.60902
- Sadness score: 0.020552

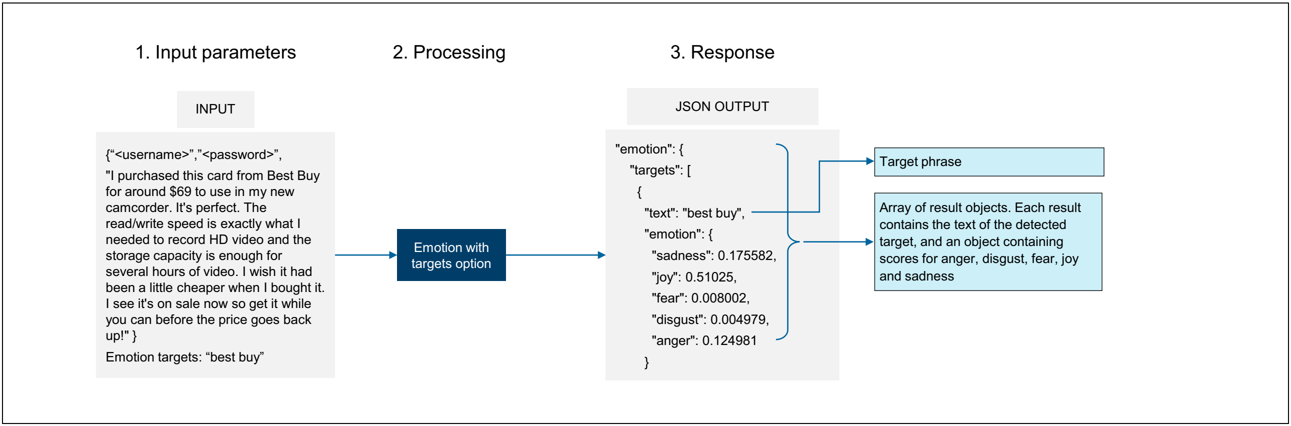
Use case example: Client emotion toward a brand

Targeted emotion can aid in recognizing the emotions toward a certain product or brand. For example, an online store can analyze reviews to objectively understand customer feelings toward the store's brand or the brand of the products that the store sells.

NLU Emotion with targets option flow

Figure 1-19 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication, the text to analyze, and the targets string (the service analyzes emotion for each target string found in the text).
2. Processing: The service analyzes emotion in the input text for the targets string.
3. Response (returns response in JSON format):
   - `Document`: Object containing emotion analysis results for the entire document.
   - `Targets`: Array of objects containing emotion results for the targets.
   - `Emotion`: Emotion scores in the range of 0 - 1 for `sadness`, `joy`, `fear`, `disgust`, and `anger`. A score of 0 means the text does not convey the emotion; 1 means the text definitely carries the emotion.


1. Input parameters
2. Processing
3. Response
INPUT
Emotion with targets option
JSON OUTPUT
Target phrase
Array of result objects. Each result contains the text of the detected target, and an object containing scores for anger, disgust, fear, joy and sadness

```
{“<username>”,”<password>”,
"I purchased this card from Best Buy for around $69 to use in my new camcorder. It's perfect. The read/write speed is exactly what I needed to record HD video and the storage capacity is enough for several hours of video. I wish it had been a little cheaper when I bought it. I see it's on sale now so get it while you can before the price goes back up!" }
Emotion targets: “best buy”
```

```
"emotion": {
  "targets": [
   {
    "text": "best buy",
    "emotion": {
     "sadness": 0.175582,
     "joy": 0.51025,
     "fear": 0.008002,
     "disgust": 0.004979,
     "anger": 0.124981
    }
```

Figure 1-19 Emotion with targets option flow

NLU Emotion with targets option snippets

This section includes sample snippets to illustrate the use of the NLU Emotion feature with the targets option in Node.js, Java, and Node-RED.

This example shows how to analyze emotion in the following sample text with targets phrase of `Best Buy`:

```
I purchased this card from Best Buy for around $69 to use in my new camcorder.
It's perfect. The read/write speed is exactly what I needed to record HD video and
the storage capacity is enough for several hours of video. I wish it had been a
little cheaper when I bought it. I see it's on sale now so get it while you can
before the price goes back up.
```

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-7 shows the `nlu-targetedEmotion.js` snippet.

Example 1-7 Snippet: nlu-targetedEmotion.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   text : "I purchased this card from Best Buy for around $69 to use in my new camcorder. It's
perfect. The read/write speed is exactly what I needed to record HD video and the storage
capacity is enough for several hours of video. I wish it had been a little cheaper when I bought
it. I see it's on sale now so get it while you can before the price goes back up!",
   features : {
      emotion : {
         targets : [ 'Best Buy' ]
      }
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error || response.status === "ERROR") {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      // as there's only 1 target, will select element 0 of the array of results
      var targetedEmotion = response.emotion.targets[0].emotion;
      // process object 'targetedEmotion' that contains properties 'anger', 'disgust', 'fear',
'joy', 'sadness'
   }
});
```

Java snippet

Example 1-8 shows the `nlutargetedEmotion.java` snippet.

Example 1-8 Snippet: nlutargetedEmotion.java

```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalysisResults;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalyzeOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.EmotionOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.Features;

import java.util.ArrayList;
import java.util.List;


public class nlutargetedEmotion {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The text to be analyzed for
        String text = "I purchased this card from Best Buy for around $69 to use in my new
camcorder. It's perfect. The read/write speed is exactly what I needed to record HD video and
the storage capacity is enough for several hours of video. I wish it had been a little cheaper
when I bought it. I see it's on sale now so get it while you can before the price goes back
up!";


        //The keywords to do emotion analysis on
        List<String> targets = new ArrayList<String>();
        targets.add("Best Buy");

        //The Emotions object
        EmotionOptions emotions = new EmotionOptions.Builder().targets(targets).build();

        //Features object to hold Analysis features
        Features features = new Features.Builder().emotion(emotions).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().text(text).features(features).build();

        //The emotions returned from the analysis with their relative weight
        AnalysisResults response = service.analyze(parameters).execute();

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

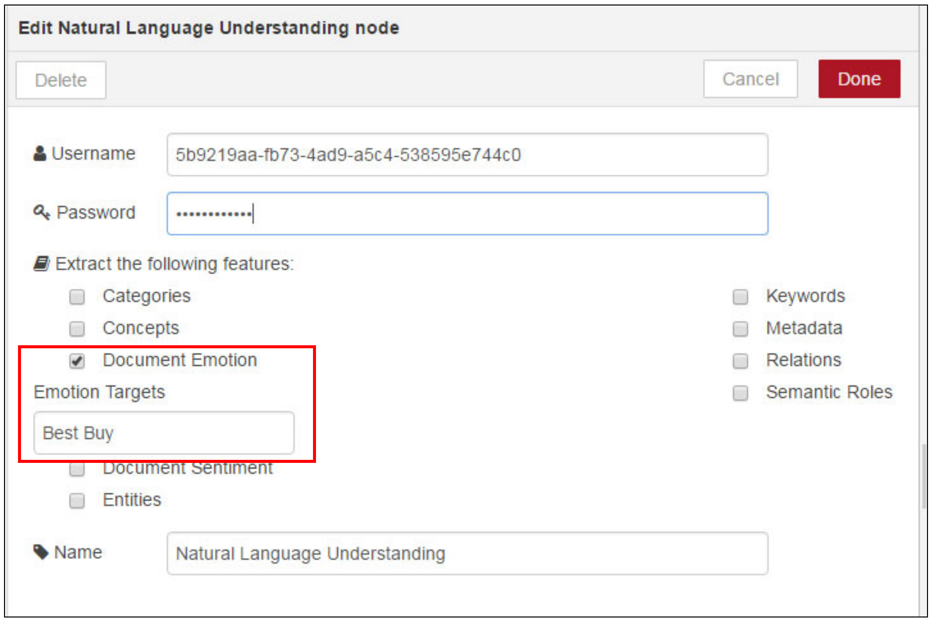
Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the text that you want to analyze:

```
I purchased this card from Best Buy for around $69 to use in my new camcorder.
It's perfect. The read/write speed is exactly what I needed to record HD video
and the storage capacity is enough for several hours of video. I wish it had
been a little cheaper when I bought it. I see it's on sale now so get it while
you can before the price goes back up.
```

3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-20). Enter your Natural Language Understanding service instance credentials (`username` and `password`), select **Document Emotion**, and enter `Best Buy` in the **Emotion Targets** field.



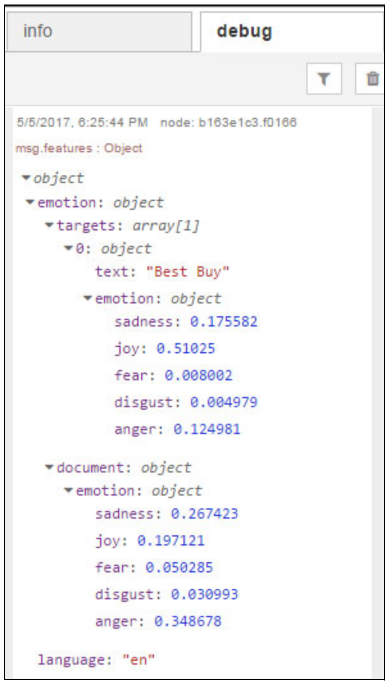
Username 5b9219aa-fb73-4ad9-a5c4-538595e744c0
Password ••••••••••••
Extract the following features:
- [ ] Categories
- [ ] Concepts
- [✓] Document Emotion
Emotion Targets: Best Buy
- [ ] Document Sentiment
- [ ] Entities
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name Natural Language Understanding

Figure 1-20 Edit Natural Language Understanding node dialog: Document Emotion and targets

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11.Watch the **debug** tab. The output should be similar to Figure 1-21.



```
5/5/2017, 6:25:44 PM   node: b163e1c3.f0166
msg.features : Object
▾object
 ▾emotion: object
   ▾targets: array[1]
     ▾0: object
        text: "Best Buy"
       ▾emotion: object
          sadness: 0.175582
          joy: 0.51025
          fear: 0.008002
          disgust: 0.004979
          anger: 0.124981
   ▾document: object
     ▾emotion: object
        sadness: 0.267423
        joy: 0.197121
        fear: 0.050285
        disgust: 0.030993
        anger: 0.348678
  language: "en"
```

Figure 1-21   NLU Emotion with targets option response: The debug tab

Example 1-9 shows the nodes and connections flow, exported in JSON format.

Example 1-9   NLU Emotion with targets option: Exported flow

```
The Nodes and connections flow exported in JSON format.
[{"id":"a31695e2.13285","type":"tab","label":"Flow
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language
Understand-ing","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion
":true,"doc-emotion-target":"Best
Buy","doc-sentiment":false,"doc-sentiment-target":"","entity":false,"entity-emotio
n":false,"entity-sentiment":false,"maxentities":"50","keyword":false,"keyword-emot
ion":false,"keyword-senti-ment":false,"maxkeywords":"50","metadata":false,"relatio
n":false,"semantic":false,"semantic-entities":false,"semantic-key-words":false,"ma
xsemantics":"50","x":454.9000244140625,"y":145.8000030517578,"wires":[["587c390d.0
c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a31695e2.13285","name":"","t
opic":"","payload":"I purchased this card from Best Buy for around $69 to use in
my new camcorder. It's perfect. The read/write speed is exactly what I needed to
record HD video and the storage capacity is enough for several hours of video. I
wish it had been a little cheaper when I bought it. I see it's on sale now so get
it while you can before the price goes back
up!","payloadType":"str","repeat":"","crontab":"","once":false,"x":244.90000915527
344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debug","z"
:"a31695e2.13285","name":"","active":true,"console":"false","complete":"features",
"x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.4 NLU Entities

The NLU Entities feature helps you Identify people, cities, organizations, and many other types of entities in your text. It returns items such as persons, places, and organizations that are present in the input text. Entity extraction adds semantic knowledge to content to help understand the subject and context of the text that is being analyzed. See the following input and response examples.

Input

Text:

```
IBM is an American multinational technology company headquartered in Armonk, New
York, United States, with operations in over 170 countries.
```

Response

Entities:

- IBM: Company
- Armonk: Location
- New York: Location
- United States: Location

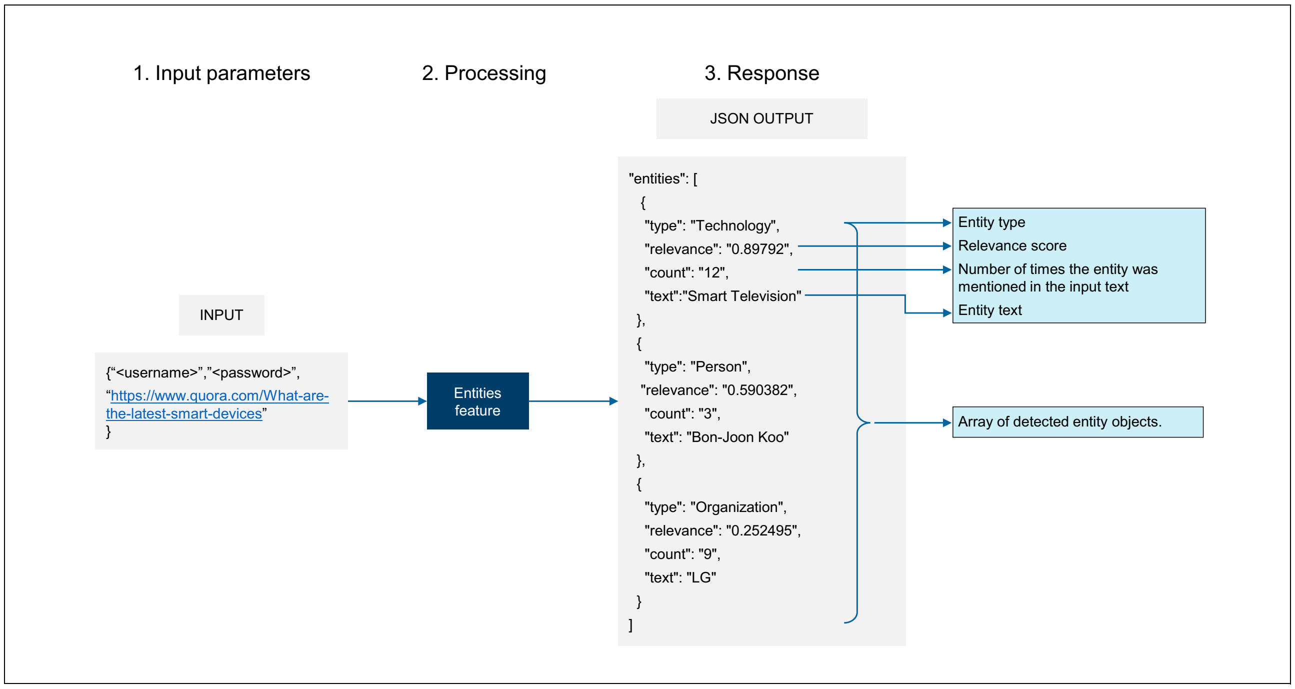
Use case example: Identify products and people in technology article

Analyze a news article on the latest smart devices and identify entities such as technologies, people, and organizations.

NLU Entities flow

Figure 1-22 on page 26 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication and the URL of the article to be analyzed.
2. Processing: The service analyzes the text and identifies entities.
3. Response (returns response in JSON format):
   - `Type`: Entity type.
   - `Text`: Entity text.
   - `Relevance`: Entity relevance score in the range of 0 - 1. A score of 0 means it is not relevant; 1 means it is highly relevant.
   - `Count`: Number of times the entity is mentioned in the text.


1. Input parameters
2. Processing
3. Response
JSON OUTPUT
INPUT
{"<username>","<password>",
"https://www.quora.com/What-are-the-latest-smart-devices"
}
Entities feature
Entity type
Relevance score
Number of times the entity was mentioned in the input text
Entity text
Array of detected entity objects.

```
"entities": [
  {
   "type": "Technology",
   "relevance": "0.89792",
   "count": "12",
   "text":"Smart Television"
  },
  {
   "type": "Person",
   "relevance": "0.590382",
   "count": "3",
   "text": "Bon-Joon Koo"
  },
  {
   "type": "Organization",
   "relevance": "0.252495",
   "count": "9",
   "text": "LG"
  }
]
```

Figure 1-22 Entities flow

NLU Entities snippets

This section includes sample snippets to illustrate the use of the NLU Entities feature in Node.js, Java, and Node-RED.

This example shows how to identify entities in the following sample news article about the latest smart phones:

http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-10 shows the `nlu-entities.js` snippet.

Example 1-10 Snippet: nlu-entities.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url :
'http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440',
   features : {
      entities : {
         limit: 250
      }
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var entities = response.entities;
      // process the array of entities
   }
});
```

Java snippet

Example 1-11 shows the `nluentities.java` snippet.

Example 1-11 Snippet: nluentities.java

```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.*;


public class nluentities {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The url to analyze its content
        String url =
"http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-t
he-world-today-645440";

        //The Entities object, to extract people, cities, organizations from our text
        EntitiesOptions entities = new EntitiesOptions.Builder().build();


        //Features object to hold Analysis features
        Features features = new Features.Builder().entities(entities).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().url(url).features(features).build();

        //The entities returned from the analysis with some information about their type,
relevance and count
        AnalysisResults response = service.analyze(parameters).execute();

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

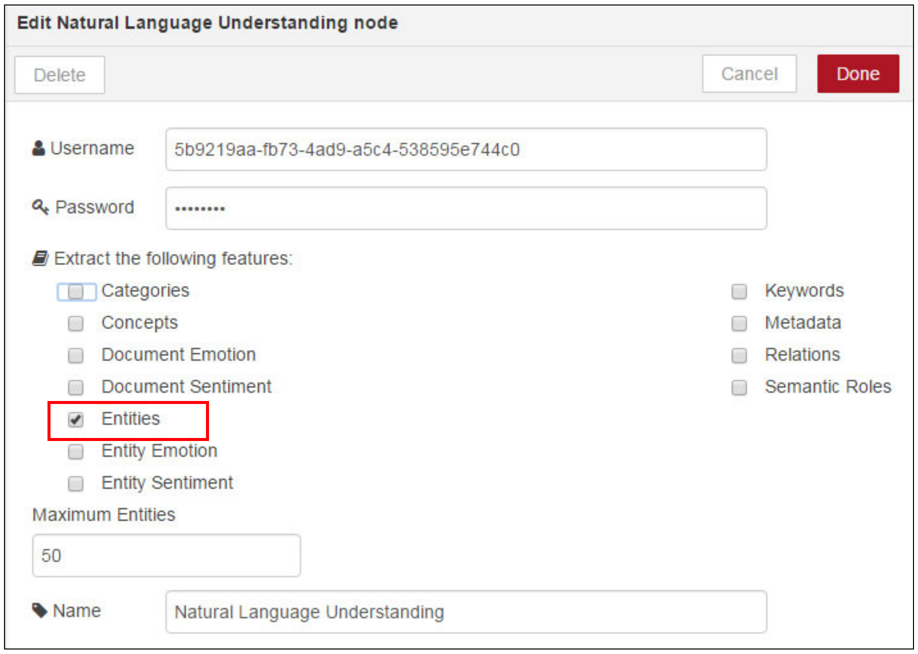
Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the URL of the article that you want to analyze:

   `http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440`

3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-23). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Entities**.


Edit Natural Language Understanding node

Username	5b9219aa-fb73-4ad9-a5c4-538595e744c0
Password	••••••••

Extract the following features:

[ ] Categories	[ ] Keywords
[ ] Concepts	[ ] Metadata
[ ] Document Emotion	[ ] Relations
[ ] Document Sentiment	[ ] Semantic Roles
[✓] Entities	
[ ] Entity Emotion	
[ ] Entity Sentiment	

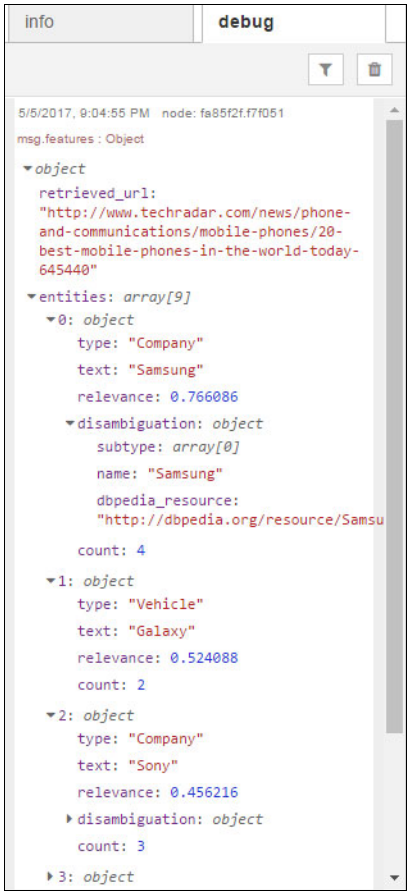
Maximum Entities: 50

Name: Natural Language Understanding

Figure 1-23 Edit Natural Language Understanding node dialog: Entities

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-24.



```
5/5/2017, 9:04:55 PM  node: fa85f2f.f7f051
msg.features : Object
▾object
  retrieved_url:
  "http://www.techradar.com/news/phone-
  and-communications/mobile-phones/20-
  best-mobile-phones-in-the-world-today-
  645440"
 ▾entities: array[9]
   ▾0: object
      type: "Company"
      text: "Samsung"
      relevance: 0.766086
     ▾disambiguation: object
        subtype: array[0]
        name: "Samsung"
        dbpedia_resource:
        "http://dbpedia.org/resource/Samsu
      count: 4
   ▾1: object
      type: "Vehicle"
      text: "Galaxy"
      relevance: 0.524088
      count: 2
   ▾2: object
      type: "Company"
      text: "Sony"
      relevance: 0.456216
     ▸disambiguation: object
      count: 3
   ▸3: object
```

Figure 1-24 NLU Entities feature response: The debug tab

Example 1-12 shows the nodes and connections flow exported in JSON format.

Example 1-12 NLU Entities: Exported flow

```
[{"id":"6157097b.7bf008","type":"tab","label":"Flow
8"},{"id":"4aeb2156.cd01","type":"natural-language-understanding","z":"6157097b.7bf008
","name":"Natural Language
Understanding","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion":fal
se,"doc-emotion-target":"","doc-sentiment":false,"doc-sentiment-target":"","entity":tr
ue,"entity-emotion":false,"entity-sentiment":false,"maxentities":"50","keyword":false,
"keyword-emotion":false,"keyword-sentiment":false,"maxkeywords":"50","metadata":false,
"relation":false,"semantic":false,"semantic-entities":false,"semantic-keywords":false,
"maxsemantics":"50","x":500,"y":232,"wires":[["6424a91f.ec0688"]]},{"id":"7e7eb298.be3
dac","type":"inject","z":"6157097b.7bf008","name":"","topic":"","payload":"http://www.
techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the
-world-today-645440","payloadType":"str","repeat":"","crontab":"","once":false,"x":107
,"y":251,"wires":[["4aeb2156.cd01"]]},{"id":"6424a91f.ec0688","type":"debug","z":"6157
097b.7bf008","name":"","active":true,"console":"false","complete":"features","x":858,"
y":287,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.5 NLU Keywords

The NLU Keywords feature identifies the important keywords in your content. Important topics in your content that are typically used when indexing data, generating tag clouds, or when searching are identified. The service automatically identifies supported languages in your input content, and then identifies and ranks keywords in that content. The supported languages are: English, French, German, Italian, Portuguese, Russian, Spanish, Swedish. See the following input and response examples.

Input

A document about the American Civil War:

`http://www.historynet.com/civil-war`

Response

- Civil war
- Union
- Battle
- Confederate

Use case example: Identify keywords to build a glossary

Keyword extraction allows for the automatic identification of terms (keywords) that best describe the subject of a document. This feature can be used by organizations to build a glossary based on the most common terms used in their documentation and the relevance of the term.

NLU Keywords flow

Figure 1-25 on page 31 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication and the URL of the article to be analyzed.
2. Processing: The service analyzes the text and extracts keywords.
3. Response (returns response in JSON format):
   - `Text`: Keyword text.
   - `Relevance`: Keyword relevance score in the range of 0 - 1. A score of 0 means it is not relevant; 1 means it is highly relevant.
   - `Count`: Number of times the entity is mentioned in the text.

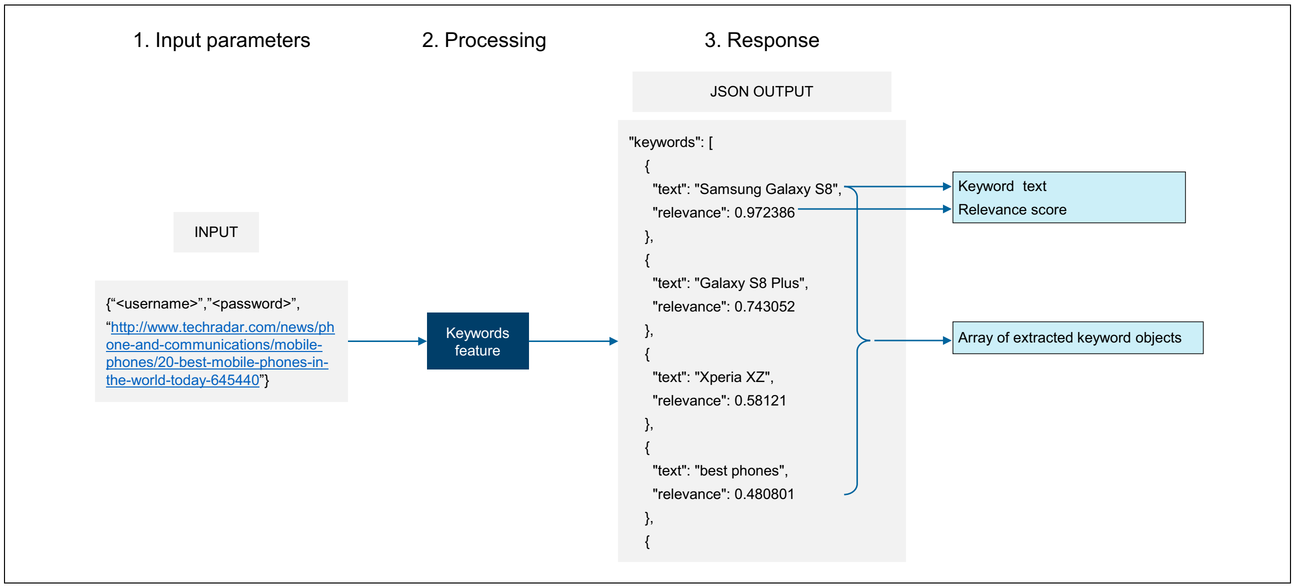


Figure 1-25 Keywords flow

NLU Keywords with emotion and sentiment options

The Keywords feature supports these options:

- `emotion`: Set this option to `true` to enable emotion analysis for detected keywords.
- `sentiment`: Set this option to `true` to enable sentiment analysis for detected keywords.

See the following input and response examples.

Input

The following article is input:

`http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440`

Response

After the article is analyzed, keywords are extracted and the corresponding sentiment and emotion for each keyword is computed as shown in Example 1-13.

Example 1-13 Response for NLU Keywords with emotion and sentiment options set to true

```
"retrieved_url":
"http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones
-in-the-world-today-645440",
  "keywords": [
    {
      "text": "Samsung Galaxy S8",
      "sentiment": {
        "score": 0.456834
      },
      "relevance": 0.972386,
      "emotion": {
        "sadness": 0.120647,
        "joy": 0.502089,
        "fear": 0.07519,
        "disgust": 0.010992,
        "anger": 0.127171
      }
    },
    {
      "text": "Galaxy S8 Plus",
      "sentiment": {
        "score": 0.409326
      },
      "relevance": 0.743052,
      "emotion": {
        "sadness": 0.129332,
        "joy": 0.459097,
        "fear": 0.09797,
        "disgust": 0.012402,
        "anger": 0.136951
      }
    },
    {
      "text": "Xperia XZ",
      "sentiment": {
        "score": 0.46716
      },
      "relevance": 0.58121,
      "emotion": {
        "sadness": 0.053549,
        "joy": 0.453336,
        "fear": 0.006229,
        "disgust": 0.008426,
        "anger": 0.316831
      }
```

NLU Keywords snippets

This section includes sample snippets to illustrate the use of the NLU Keywords feature in Node.js, Java, and Node-RED.

This example shows how to extract keywords and determine the corresponding relevance score in the following sample news article about the latest smart phones:

http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-14 shows the nlu-keywords.js snippet.

Example 1-14 Snippet: nlu-keywords.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url :
'http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones
-in-the-world-today-645440',
   features : {
      keywords : {
            limit : 250
        }
    }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var keywords = response.keywords;
      // process the array of keywords
   }
});
```

Java snippet

Example 1-15 shows the `nlu-keywords.java` snippet.

Example 1-15 Snippet: nlukeywords.java

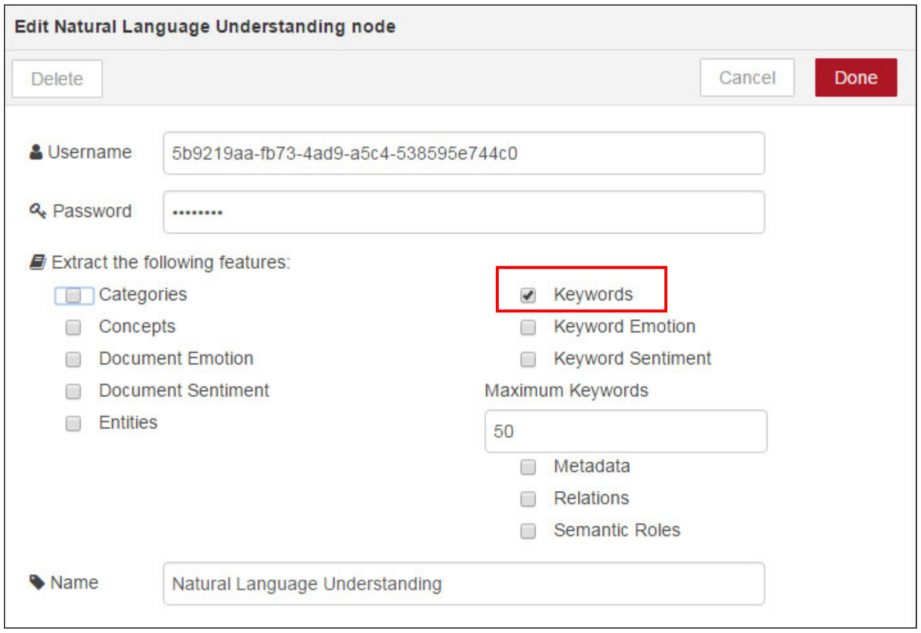
```
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.*;


public class nlukeywords {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The url to analyze its content
        String url =
"http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-to
day-645440";

        //The keywords object, to identify important keywords in the content
        KeywordsOptions keywords = new KeywordsOptions.Builder().build();


        //Features object to hold Analysis features
        Features features = new Features.Builder().keywords(keywords).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new AnalyzeOptions.Builder().url(url).features(features).build();

        //The keywords returned from the analysis, and their relevance.
        AnalysisResults response = service.analyze(parameters).execute();

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

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the URL of the article that you want to analyze:

   `http://www.techradar.com/news/phone-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440`
3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-26). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Keywords**.


Edit Natural Language Understanding node

Delete | Cancel | Done

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0

Password: ••••••••

Extract the following features:

- [ ] Categories
- [ ] Concepts
- [ ] Document Emotion
- [ ] Document Sentiment
- [ ] Entities
- [✓] Keywords
- [ ] Keyword Emotion
- [ ] Keyword Sentiment

Maximum Keywords: 50

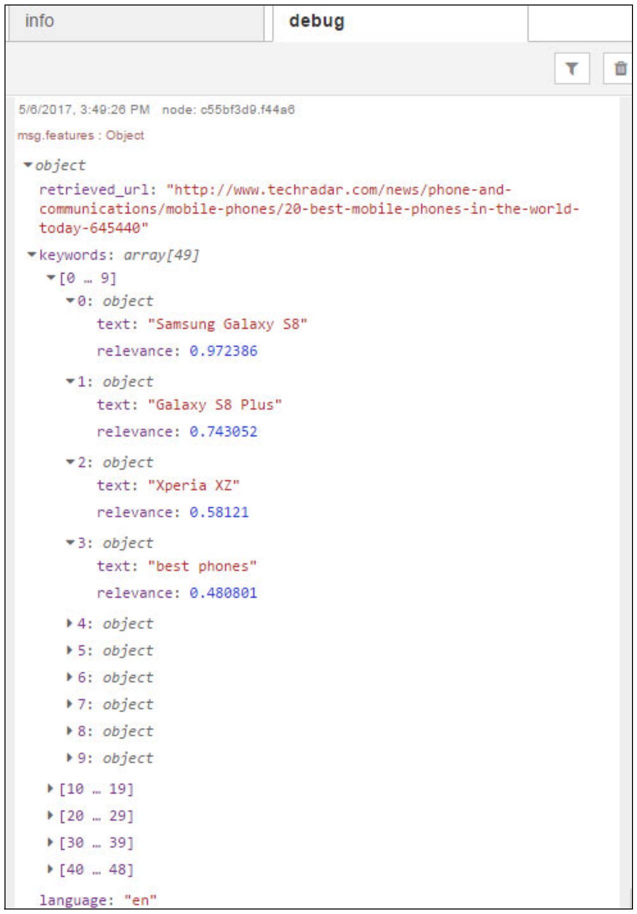
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name: Natural Language Understanding

Figure 1-26 Edit Natural Language Understanding node dialog: Keywords

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the debug node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-27.



```
5/6/2017, 3:49:26 PM   node: c55bf3d9.f44a6
msg.features : Object
▾object
  retrieved_url: "http://www.techradar.com/news/phone-and-
  communications/mobile-phones/20-best-mobile-phones-in-the-world-
  today-645440"
 ▾keywords: array[49]
   ▾[0 … 9]
     ▾0: object
        text: "Samsung Galaxy S8"
        relevance: 0.972386
     ▾1: object
        text: "Galaxy S8 Plus"
        relevance: 0.743052
     ▾2: object
        text: "Xperia XZ"
        relevance: 0.58121
     ▾3: object
        text: "best phones"
        relevance: 0.480801
     ▸4: object
     ▸5: object
     ▸6: object
     ▸7: object
     ▸8: object
     ▸9: object
   ▸[10 … 19]
   ▸[20 … 29]
   ▸[30 … 39]
   ▸[40 … 48]
  language: "en"
```

Figure 1-27   NLU Keywords feature response: The debug tab

Example 1-16 shows the nodes and connections flow exported in JSON format.

Example 1-16   NLU Keywords: Exported flow

```
[{"id":"512b4c36.e88b04","type":"tab","label":"Flow 
2"},{"id":"6a344cf4.a48dc4","type":"natural-language-understanding","z":"512b4c36.e88b04","
name":"Natural Language 
Understanding","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion":false,"d
oc-emotion-target":"Best 
Buy","doc-sentiment":false,"doc-sentiment-target":"","entity":false,"entity-emotion":false,
"entity-sentiment":false,"maxentities":"250","keyword":true,"keyword-emotion":false,"keywor
d-sentiment":false,"maxkeywords":"50","metadata":false,"relation":false,"semantic":false,"s
emantic-entities":false,"semantic-keywords":false,"maxsemantics":"50","x":454.9000244140625
,"y":145.8000030517578,"wires":[["c55bf3d9.f44a6"]]},{"id":"4719f358.a728cc","type":"inject
","z":"512b4c36.e88b04","name":"","topic":"","payload":"http://www.techradar.com/news/phone
-and-communications/mobile-phones/20-best-mobile-phones-in-the-world-today-645440","payload
Type":"str","repeat":"","crontab":"","once":false,"x":244.90000915527344,"y":84,"wires":[["
6a344cf4.a48dc4"]]},{"id":"c55bf3d9.f44a6","type":"debug","z":"512b4c36.e88b04","name":"","
active":true,"console":"false","complete":"features","x":672.9000244140625,"y":79.800003051
75781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.6 NLU Relations

The NLU Relations feature identifies subject, action, and object relations within sentences in the input content. After parsing sentences into subject, action, and object form, the Relations feature can use this information for subsequent processing by other Natural Language Understanding features such as entities, keywords, and so on.

Relation information can be used to automatically identify buying signals, key events, and other important actions. See the following input and response examples.

Input

Text:

```
Bob Dylan won the Nobel Prize in Literature in 2016. Bob Dylan was born in Duluth,
Minnesota.
```

Response

Output examples:

- "`affectedBy`" relation between “Bob Dylan” and “won”
- "`timeOf`" relation between “2016” and “won”
- "`awardedTo`" relation between "Nobel Prize" and "Bob Dylan"
- "`bornAt`" relation between "Bob Dylan" and "Duluth"
- "`locatedAt`" relation between "Duluth" and "Minnesota"

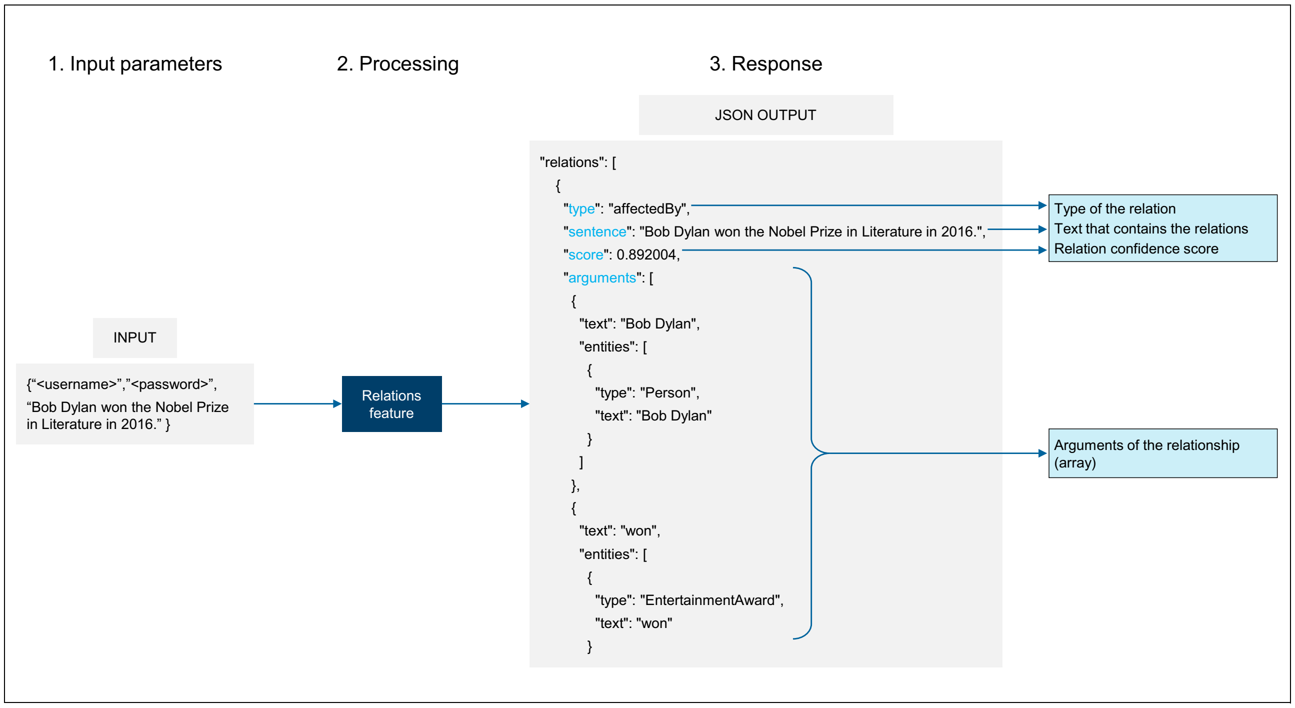
Use case example:

The Relations feature can be used to recognize when two entities are related and the type of relation between them. An organization can use this feature to better understand what customers are saying about them or to disambiguate a phrase.

NLU Relations flow

Figure 1-28 on page 37 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication and the text to be analyzed.
2. Processing: The service analyzes the text and identifies relations.
3. Response (returns response in JSON format):
   - `type`: Type of the relation.
   - `sentence`: Selection of text that contains the relation.
   - `score`: Relation confidence score in the range of 0 - 1. A score of 0 means it is not confident; 1 means it is highly confident.
   - `arguments`: The arguments of the relation. Each argument contains the argument text, and an entities object that details the type of entity involved in the relation.



```
{"<username>","<password>",
"Bob Dylan won the Nobel Prize
in Literature in 2016." }
```

```
"relations": [
  {
   "type": "affectedBy",
   "sentence": "Bob Dylan won the Nobel Prize in Literature in 2016.",
   "score": 0.892004,
   "arguments": [
    {
     "text": "Bob Dylan",
     "entities": [
      {
       "type": "Person",
       "text": "Bob Dylan"
      }
     ]
    },
    {
     "text": "won",
     "entities": [
      {
       "type": "EntertainmentAward",
       "text": "won"
      }
```

Figure 1-28 Relations flow

NLU Relations snippets

This section includes sample snippets to illustrate the use of the NLU Relations feature in Node.js, Java, and Node-RED.

This example shows how to identify relations in the following sentence:

`Cutting Cash Would Be a Boon for the World's Poor, Rogoff Says.`

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-17 shows the `nlu-relations.js` snippet.

Example 1-17 Snippet: nlu-relations.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   text : 'Cutting Cash Would Be a Boon for the World's Poor, Rogoff Says',
   features : {
      relations : {}
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var relations = response.relations;
      // process the array of relations
   }
});
```

Java snippet

Example 1-18 shows the `nlurelations.java` snippet.

Example 1-18 Snippet: nlurelations.java

```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.*;


public class nlurelations {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The text to analyze its content
        String text = "Cutting Cash Would Be a Boon for the World’s Poor, Rogoff Says";

        //The Relations object, to identify important relations in the content
        RelationsOptions relations = new RelationsOptions.Builder().build();


        //Features object to hold Analysis features
        Features features = new Features.Builder().relations(relations).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().text(text).features(features).build();

        //The relations returned from the analysis, and their types.
        AnalysisResults response = service.analyze(parameters).execute();

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

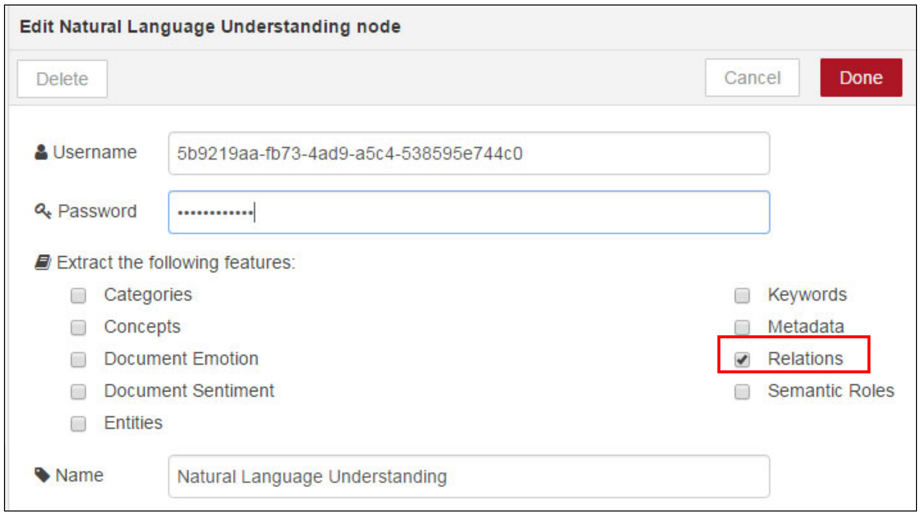
Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the text that you want to analyze:

   ```
   Cutting Cash Would Be a Boon for the World's Poor, Rogoff Says.
   ```

3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-29). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Relations**.

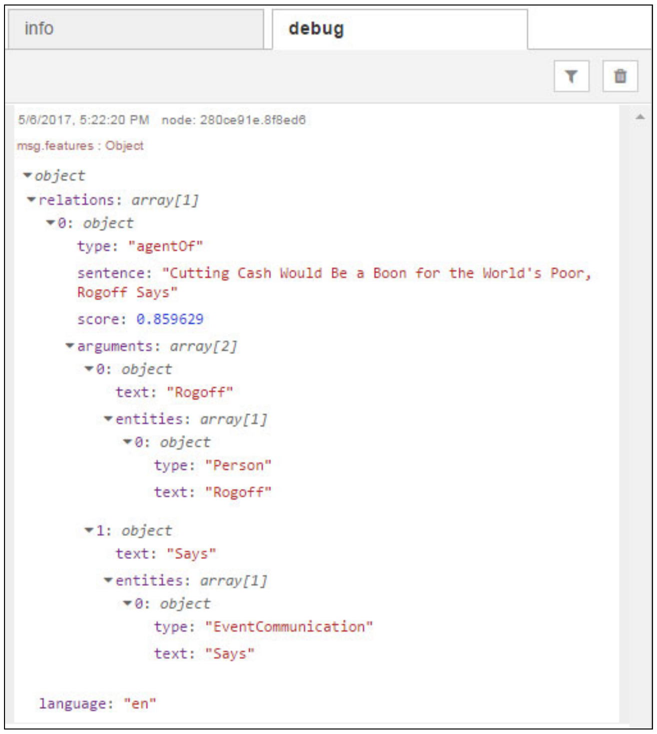

Edit Natural Language Understanding node

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0
Password: ....
Extract the following features:
[ ] Categories
[ ] Concepts
[ ] Document Emotion
[ ] Document Sentiment
[ ] Entities
[ ] Keywords
[ ] Metadata
[✓] Relations
[ ] Semantic Roles
Name: Natural Language Understanding

Figure 1-29 Edit Natural Language Understanding node dialog: Relations

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-30.



```
5/6/2017, 5:22:20 PM  node: 280ce91e.8f8ed6
msg.features : Object
▾object
▾relations: array[1]
  ▾0: object
     type: "agentOf"
     sentence: "Cutting Cash Would Be a Boon for the World's Poor,
     Rogoff Says"
     score: 0.859629
    ▾arguments: array[2]
      ▾0: object
         text: "Rogoff"
        ▾entities: array[1]
          ▾0: object
             type: "Person"
             text: "Rogoff"
      ▾1: object
         text: "Says"
        ▾entities: array[1]
          ▾0: object
             type: "EventCommunication"
             text: "Says"
 language: "en"
```

Figure 1-30 NLU Relations feature response: The debug tab

Example 1-19 shows the nodes and connections flow exported in JSON format.

Example 1-19 NLU Emotions: Exported flow

```
[{"id":"a31695e2.13285","type":"tab","label":"Flow
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language
Understand-ing","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion
":false,"doc-emotion-target":"Best
Buy","doc-sentiment":false,"doc-sentiment-target":"","entity":false,"entity-emotio
n":false,"entity-sentiment":false,"maxentities":"250","keyword":false,"keyword-emo
tion":false,"keyword-senti-ment":false,"maxkeywords":"50","metadata":false,"relati
on":true,"semantic":false,"semantic-entities":false,"semantic-key-words":false,"ma
xsemantics":"50","x":454.9000244140625,"y":145.8000030517578,"wires":[["587c390d.0
c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a31695e2.13285","name":"","t
opic":"","payload":"Cutting Cash Would Be a Boon for the World's Poor, Rogoff
Says","payloadType":"str","repeat":"","crontab":"","once":false,"x":244.9000091552
7344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debug","z
":"a31695e2.13285","name":"","active":true,"console":"false","complete":"features"
,"x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.7 NLU Sentiment

The NLU Sentiment feature identifies attitude, opinions, or feelings in the content that is being analyzed. You can use this feature to analyze the sentiment toward specific target phrases or simply analyze the sentiment toward the document as a whole. You can also get sentiment information for detected entities and keywords by enabling the sentiment option for those features. The supported languages are Arabic, English, French, German, Italian, Portuguese, Russian, Spanish. See the following input and response examples.

Input

Text:

```
I'm very upset about the quality of this product.
```

Response

Negative sentiment (score: -0.890748)

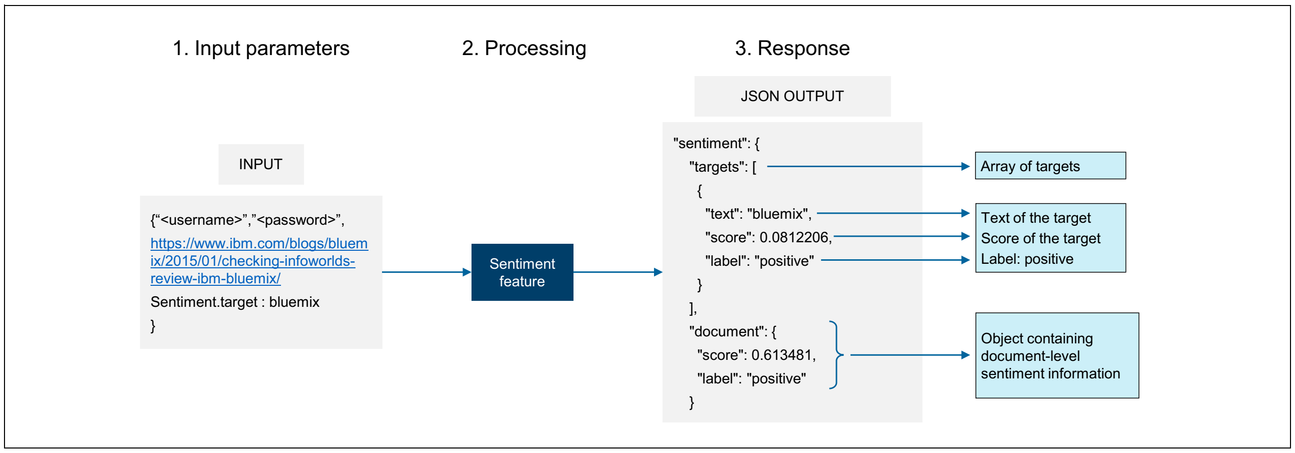
Use case example: Social media sentiment analysis

Analyze what the public is saying in social media channels about your products, services, and brand. Organizations can collect Twitter data with a Twitter crawler and then use the Sentiment feature to analyze what people are saying about the products and services.

NLU Sentiment flow

Figure 1-31 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication, the URL of the text to be analyzed, and the target option (optional).
2. Processing: The service analyzes sentiment for each target string found in the text and for the document as a whole.
3. Response (returns response in JSON format):
   - `document`: Document-level sentiment analysis results.
   - `targets`: Array of target analysis results. Each object contains the text of the target, sentiment score, and a label.
   - `score`: Sentiment score ranges from -1 (negative sentiment) to 1 (positive sentiment).
   - `label`: Indicates whether the sentiment is positive, neutral, or negative.


1. Input parameters
2. Processing
3. Response
INPUT
{"<username>","<password>",
https://www.ibm.com/blogs/bluemix/2015/01/checking-infoworlds-review-ibm-bluemix/
Sentiment.target : bluemix
}
Sentiment feature
JSON OUTPUT
Array of targets
Text of the target
Score of the target
Label: positive
Object containing document-level sentiment information

```
"sentiment": {
  "targets": [
   {
    "text": "bluemix",
    "score": 0.0812206,
    "label": "positive"
   }
  ],
  "document": {
   "score": 0.613481,
   "label": "positive"
  }
```

Figure 1-31 Sentiment with target option flow

NLU Sentiment snippets

This section includes sample snippets to illustrate the use of the NLU Sentiments feature in Node.js, Java, and Node-RED.

Example 1: Identify sentiment to document

The sample code performs these steps:

1. Analyzes an article at the following web page:

   `http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review`

2. Identifies the sentiment toward the entire document.
3. Prints the response to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-20 shows the nlu-sentiment.js `nlu-sentiment.js` snippet.

Example 1-20 Snippet: nlu-sentiment.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url :
'http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review',
   features : {
      sentiment : {}
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var docSentiment = response.sentiment.document;
      // process sentiment at 'docSentiment' object containing 'score' and 'type'
   }
});
```

Java snippet

Example 1-21 shows the `nlusentiment.java` snippet.

Example 1-21 Snippet: nlusentiment.java

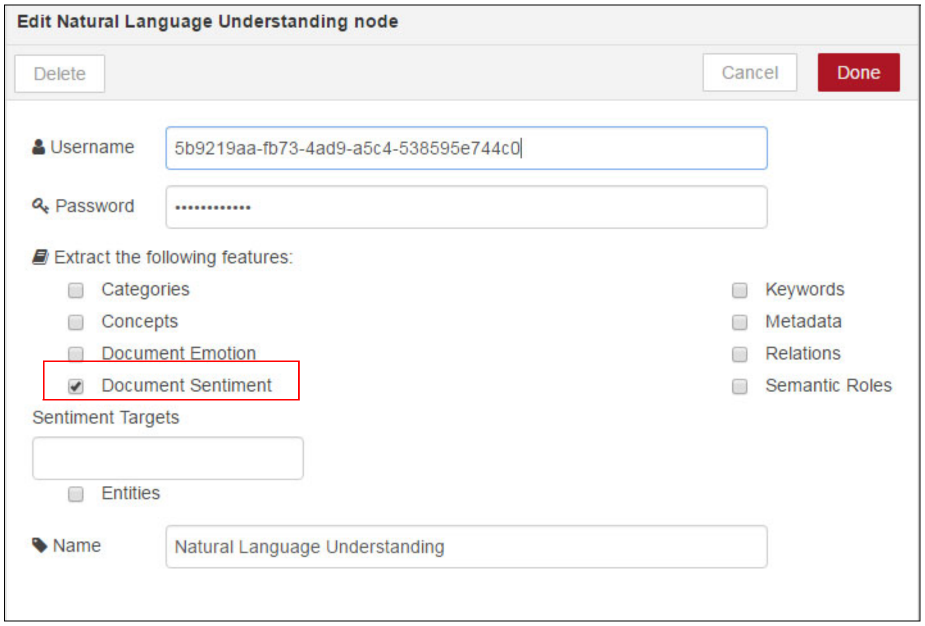
```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.*;


public class nlusentiment {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new 
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The text to analyze its content
        String url = "http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review";

        //The sentiment object, to get the sentiment measure in the content
        SentimentOptions sentiment = new SentimentOptions.Builder().build();


        //Features object to hold Analysis features
        Features features = new Features.Builder().sentiment(sentiment).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new 
AnalyzeOptions.Builder().url(url).features(features).build();

        //The sentiment score returned from the analysis, in the range of 0 to 1.
        AnalysisResults response = service.analyze(parameters).execute();

        System.out.println(response);
    }


}
```

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the URL to the article that you want to analyze:

   `http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review`
3. From the node palette, drag the a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-32 on page 44). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Document Sentiment**.



Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0

Password: ••••••••••••

Extract the following features:

- [ ] Categories
- [ ] Concepts
- [ ] Document Emotion
- [x] Document Sentiment
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

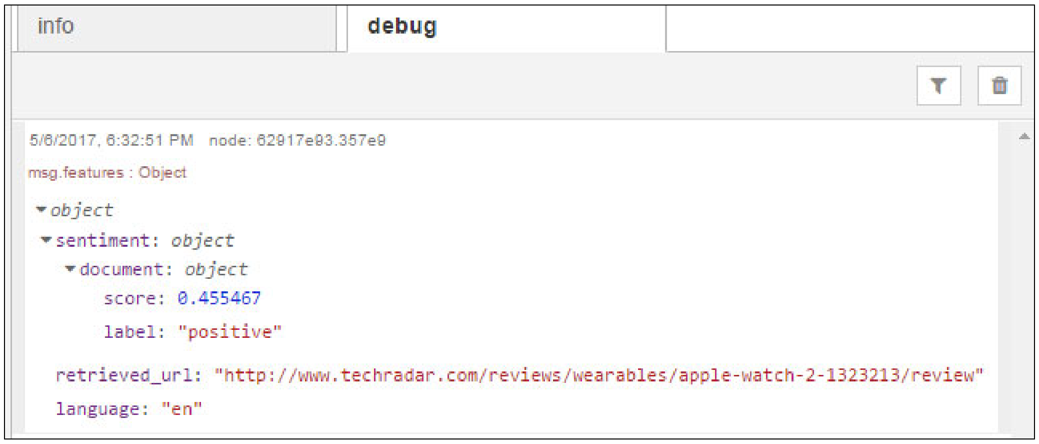
Sentiment Targets

- [ ] Entities

Name: Natural Language Understanding

Figure 1-32 Edit Natural Language Understanding node dialog: Document Sentiment

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).
11. Watch the **debug** tab. The output should be similar to Figure 1-33.



```
5/6/2017, 6:32:51 PM   node: 62917e93.357e9
msg.features : Object
▾object
 ▾sentiment: object
   ▾document: object
      score: 0.455467
      label: "positive"
 retrieved_url: "http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review"
 language: "en"
```

Figure 1-33 NLU Sentiment feature response: The debug tab

Example 1-22 shows the nodes and connections flow, exported in JSON format.

Example 1-22 NLU Sentiment: Exported flow

```
[{"id":"a31695e2.13285","type":"tab","label":"Flow
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language
Understand-ing","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion
":false,"doc-emotion-target":"Best
Buy","doc-sentiment":true,"doc-sentiment-target":"","entity":false,"entity-emotion
":false,"entity-sentiment":false,"maxentities":"250","keyword":false,"keyword-emot
ion":false,"keyword-senti-ment":false,"maxkeywords":"50","metadata":false,"relatio
n":false,"semantic":false,"semantic-entities":false,"semantic-key-words":false,"ma
xsemantics":"50","x":454.9000244140625,"y":145.8000030517578,"wires":[["587c390d.0
c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a31695e2.13285","name":"","t
opic":"","payload":"http://www.techradar.com/reviews/wearables/apple-watch-2-13232
13/review","payloadType":"str","repeat":"","crontab":"","once":false,"x":244.90000
915527344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debu
g","z":"a31695e2.13285","name":"","active":true,"console":"false","complete":"feat
ures","x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

Example 2: Identify sentiment to document and target text

The sample code does these steps:

1. Analyzes an article at the following web page:

   `http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review`

2. Identifies the sentiment toward the entire document and target text `fitness`.
3. Prints the response to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-23 shows the `nlu-targetedSentiment.js` snippet.

Example 1-23 Snippet: nlutargetedSentiment.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url :
'http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review',
   features : {
      sentiment : {
         targets : [ 'fitness' ]
      }
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error || response.status === "ERROR") {
```

```
            onError(error, response); // function to be defined by you
        } else {
            console.log(JSON.stringify(response, null, 2));
            // as there's only 1 target, will select element 0 of the array of results
            var sentiment = response.sentiment.targets[0];
            // process sentiment at 'sentiment' object containing 'score' and 'label'
        }
    });
```

Java snippet

Example 1-24 shows the `nlutargetedSentiment.java` snippet.

Example 1-24 Snippet: nlutargetedSentiment.java

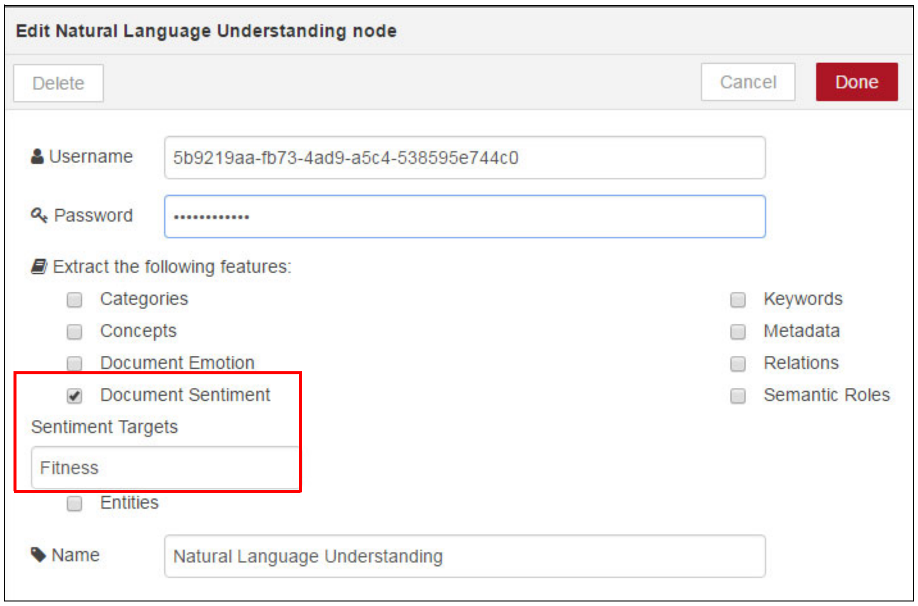
```
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalysisResults;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalyzeOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.Features;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.SentimentOptions;

import java.util.ArrayList;
import java.util.List;


public class nlutargetedSentiment {
    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
                ,username,password);
        //The text to analyze its content
        String url = "http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review";

        //The keywords to do sentiment analysis on
        List<String> targets = new ArrayList<String>();
        targets.add("fitness");

        //The sentiment object, to get the sentiment analysis in the content
        SentimentOptions sentiment = new SentimentOptions.Builder().targets(targets).build();


        //Features object to hold Analysis features
        Features features = new Features.Builder().sentiment(sentiment).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new AnalyzeOptions.Builder().url(url).features(features).build();

        //The sentiment score returned from the analysis, in the range of 0 to 1.
        AnalysisResults response = service.analyze(parameters).execute();

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

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the URL of the article that you want to analyze:

   `http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review`
3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-34). Enter your Natural Language Understanding service instance credentials (`username` and `password`), select **Document Sentiment,** and enter **Fitness** in the Sentiment Targets field.



Edit Natural Language Understanding node

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0

Password: ••••••••••••

Extract the following features:

- [ ] Categories
- [ ] Concepts
- [ ] Document Emotion
- [x] Document Sentiment

Sentiment Targets: Fitness

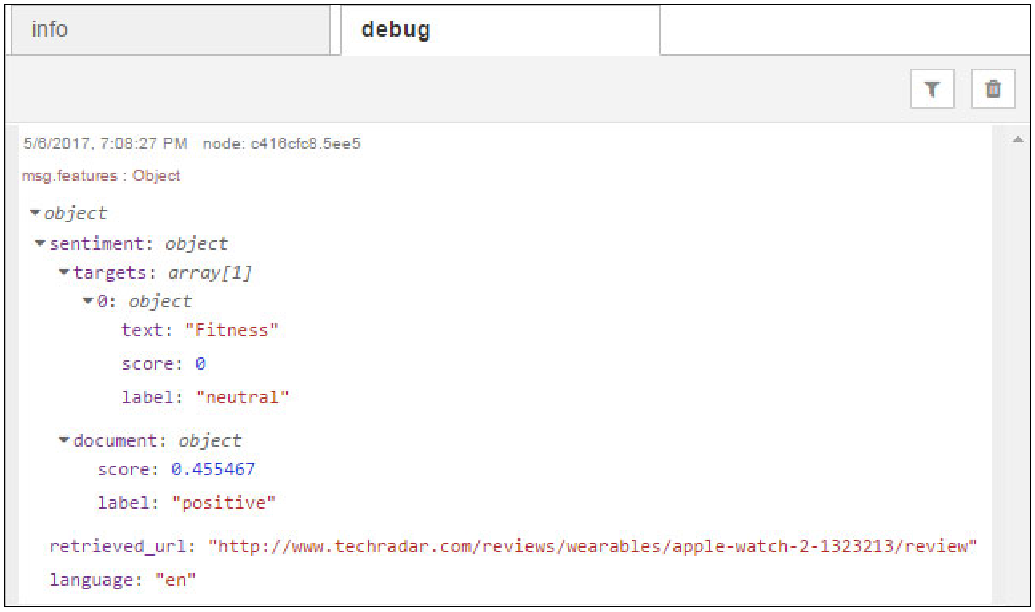
- [ ] Entities
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name: Natural Language Understanding

Figure 1-34 Edit Natural Language Understanding node dialog: Document Sentiment and Fitness

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-35.



```
5/6/2017, 7:08:27 PM   node: c416cfc8.5ee5
msg.features : Object
▾object
 ▾sentiment: object
   ▾targets: array[1]
     ▾0: object
        text: "Fitness"
        score: 0
        label: "neutral"
   ▾document: object
      score: 0.455467
      label: "positive"
 retrieved_url: "http://www.techradar.com/reviews/wearables/apple-watch-2-1323213/review"
 language: "en"
```

Figure 1-35 NLU Sentiment with targets "fitness" response: The debug tab

Example 1-25 shows the nodes and connections flow exported in JSON format.

Example 1-25 NLU Sentiment with target option: Exported flow

```
[{"id":"a31695e2.13285","type":"tab","label":"Flow 
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language 
Understand-ing","categories":false,"concepts":false,"maxconcepts":"8","doc-emotion
":false,"doc-emotion-target":"Best 
Buy","doc-sentiment":true,"doc-sentiment-target":"Fitness","entity":false,"entity-
emotion":false,"entity-sentiment":false,"maxentities":"250","keyword":false,"keywo
rd-emotion":false,"keyword-senti-ment":false,"maxkeywords":"50","metadata":false,"
relation":false,"semantic":false,"semantic-entities":false,"semantic-key-words":fa
lse,"maxsemantics":"50","x":454.9000244140625,"y":145.8000030517578,"wires":[["587
c390d.0c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a31695e2.13285","name
":"","topic":"","payload":"http://www.techradar.com/reviews/wearables/apple-watch-
2-1323213/review","payloadType":"str","repeat":"","crontab":"","once":false,"x":24
4.90000915527344,"y":84,"wires":[["7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type
":"debug","z":"a31695e2.13285","name":"","active":true,"console":"false","complete
":"features","x":672.9000244140625,"y":79.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.8 NLU Categories

The NLU Categories feature categorizes input text, HTML, or web-based content into a hierarchical taxonomy using a five-level classification hierarchy. The top three categories are returned. See the following input and response examples.

Input

Text:

```
Machine learning is the science of how computers make sense of data using
algorithms and analytic models.
```

Response

These are the responses:

- `/science/computer science/artificial intelligence`; score: 0.398614
- `/science/`; score: 0.386026
- `/science/mathematics/geometry`; score: 0.229613

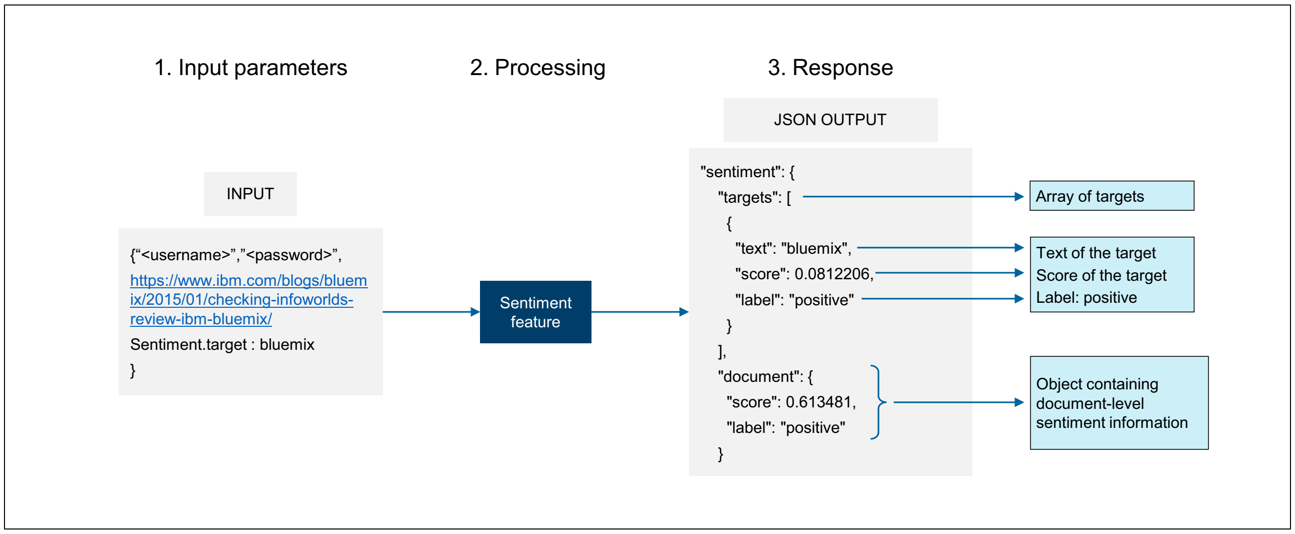
Use case example: Document classification

Analyze and classify an organization's documentation to organize articles and documents into directories and subdirectories. Use the Category feature in conjunction with the Concepts feature to create a well organized set of documentation, making access to the content easier.

NLU Categories flow

Figure 1-36 shows the basic flow:

1. Input parameters (call the API with input parameters): Pass the NLU service instance credentials (`username` and `password`) for authentication and the URL of the text to be analyzed.
2. Processing: The service analyzes the text and identifies categories and subcategories.
3. Response (returns response in JSON format):
   - `score`: Categorization score in the range of 0 - 1. A score of 0 means it is not confident in the categorization; 1 means it is highly confident.
   - `label`: Category label. Forward slashes separate category hierarchy levels.


1. Input parameters
2. Processing
3. Response
INPUT
{"<username>","<password>",
https://www.ibm.com/blogs/bluemix/2015/01/checking-infoworlds-review-ibm-bluemix/
Sentiment.target : bluemix
}
Sentiment feature
JSON OUTPUT
Array of targets
Text of the target
Score of the target
Label: positive
Object containing document-level sentiment information

```
"sentiment": {
  "targets": [
    {
      "text": "bluemix",
      "score": 0.0812206,
      "label": "positive"
    }
  ],
  "document": {
    "score": 0.613481,
    "label": "positive"
  }
```

Figure 1-36 Categories flow

NLU Categories snippets

This section includes sample snippets to illustrate the use of the NLU Categories feature in Node.js, Java, and Node-RED.

This example shows how to identify categories in the following article:

http://www.espn.com/tennis/story/_/id/18436908/australian-open-2017-tournament-news-schedule-live-scores-tv-coverage

The response is printed to the console or to the debug log (Node-RED).

Node.js snippet

Example 1-26 shows the `nlu-categories.js` snippet.

Example 1-26 Snippet: nlu-categories.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url :
'http://www.espn.com/tennis/story/_/id/18436908/australian-open-2017-tournament-ne
ws-schedule-live-scores-tv-coverage',
   features : {
      categories : {}
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var categories = response.categories;
      // process categories array
   }
});
```

Java snippet

Example 1-27 shows the `nlucategories.java` snippet.

Example 1-27 Snippet: nlucategories.java

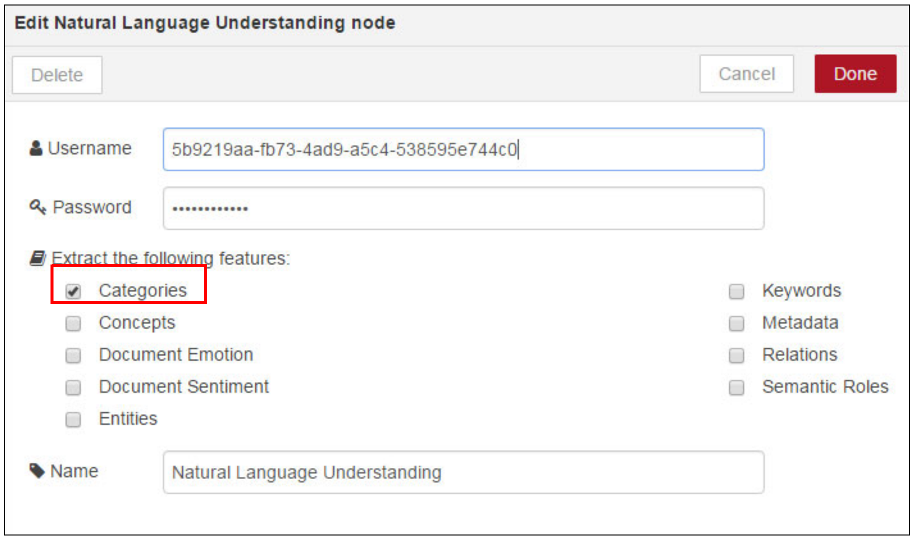
```
import
com.ibm.watson.developer_cloud.natural_language_understanding.v1.NaturalLanguageUnderstanding;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.Features;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalyzeOptions;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.AnalysisResults;
import com.ibm.watson.developer_cloud.natural_language_understanding.v1.model.CategoriesOptions;

public class nlucategories {

    private static final String username="87fa88fb-a638-463c-9bc3-225aa1c6f01e";
    private static final String password="VDlBnikeRTFS";
    public static void main(String[] arg) {
        //Construct NLU service instance
        NaturalLanguageUnderstanding service=new
NaturalLanguageUnderstanding(NaturalLanguageUnderstanding.VERSION_DATE_2017_02_27
        ,username,password);
        //The URL to be analyzed for categories
        String url =
"http://www.espn.com/tennis/story/_/id/18436908/australian-open-2017-tournament-news-schedule-li
ve-scores-tv-coverage";

        //The Categories options, 5-level taxonomy the content is to be categorized into
        CategoriesOptions categories = new CategoriesOptions();

        //Features object to hold Analysis features
        Features features = new Features.Builder().categories(categories).build();


        //Options to hold the options for our analysis , e.g., url or text
        AnalyzeOptions parameters = new
AnalyzeOptions.Builder().url(url).features(features).build();

        //the top three levels of categories found in the analysis
        AnalysisResults response = service.analyze(parameters).execute();

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

Node-RED flow

Follow these steps to create the flow:

1. From the node palette, drag an **inject** node (under the input palette) to the flow canvas (Figure 1-4 on page 8).
2. Edit the `inject` node (Figure 1-5 on page 9). Select **string** from the Payload pull-down and enter the URL to the article that you want to analyze:

   `http://www.espn.com/tennis/story/_/id/18436908/australian-open-2017-tournament-news-schedule-live-scores-tv-coverage`
3. From the node palette, drag a **Natural Language Understanding** node (under the IBM Watson palette) to the flow canvas (Figure 1-6 on page 10).
4. Edit the `Natural Language Understanding` node (Figure 1-37). Enter your Natural Language Understanding service instance credentials (`username` and `password`) and select **Categories**.


Edit Natural Language Understanding node

Username: 5b9219aa-fb73-4ad9-a5c4-538595e744c0

Password: ••••••••••••

Extract the following features:

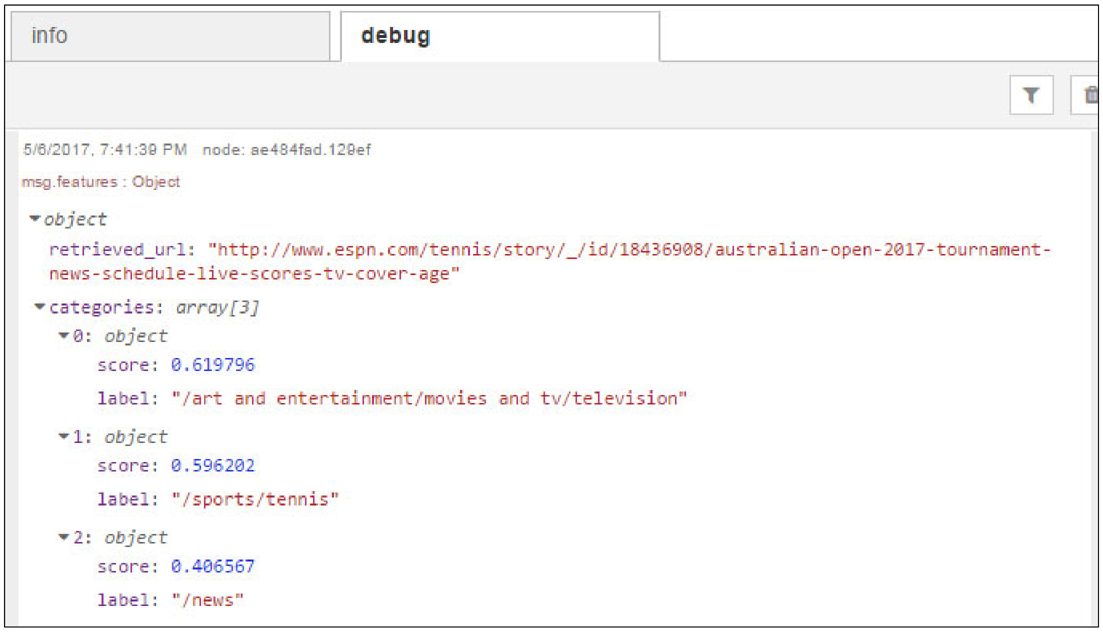
- [✓] Categories
- [ ] Concepts
- [ ] Document Emotion
- [ ] Document Sentiment
- [ ] Entities
- [ ] Keywords
- [ ] Metadata
- [ ] Relations
- [ ] Semantic Roles

Name: Natural Language Understanding

Figure 1-37 Edit Natural Language Understanding node dialog: Categories

5. Connect the **inject** and **Natural Language Understanding** nodes (Figure 1-8 on page 11).
6. From the node palette, drag a **debug** node (under the output palette) to the flow canvas (Figure 1-9 on page 12).
7. Edit the `debug` node (Figure 1-10 on page 12). Enter `msg.feature` in the Output field.
8. Connect the **Natural Language Understanding** node to the **debug** node (Figure 1-11 on page 12).
9. Click **Deploy** (Figure 1-12 on page 13).
10. Click the button at the left side of the `inject` node to inject into the flow the document that is to be analyzed (Figure 1-13 on page 13).

11. Watch the **debug** tab. The output should be similar to Figure 1-38.



```
5/6/2017, 7:41:39 PM  node: ae484fad.129ef
msg.features : Object
▾object
  retrieved_url: "http://www.espn.com/tennis/story/_/id/18436908/australian-open-2017-tournament-
  news-schedule-live-scores-tv-cover-age"
 ▾categories: array[3]
   ▾0: object
      score: 0.619796
      label: "/art and entertainment/movies and tv/television"
   ▾1: object
      score: 0.596202
      label: "/sports/tennis"
   ▾2: object
      score: 0.406567
      label: "/news"
```

Figure 1-38   NLU Categories feature response: The debug tab

Example 1-28 shows the nodes and connections flow, exported in JSON format.

Example 1-28   NLU Categories: Exported flow

```
[{"id":"a31695e2.13285","type":"tab","label":"Flow 
2"},{"id":"7cc616b1.7a5f8","type":"natural-language-understanding","z":"a31695e2.1
3285","name":"Natural Language 
Understand-ing","categories":true,"concepts":false,"maxconcepts":"8","doc-emotion"
:false,"doc-emotion-target":"Best 
Buy","doc-sentiment":false,"doc-sentiment-target":"Fitness","entity":false,"entity
-emotion":false,"entity-sentiment":false,"maxentities":"250","keyword":false,"keyw
ord-emotion":false,"keyword-senti-ment":false,"maxkeywords":"50","metadata":false,
"relation":false,"semantic":false,"semantic-entities":false,"semantic-key-words":f
alse,"maxsemantics":"50","x":454.9000244140625,"y":145.8000030517578,"wires":[["58
7c390d.0c7e2"]]},{"id":"e8fd9ee0.6bb058","type":"inject","z":"a31695e2.13285","nam
e":"","topic":"","payload":"http://www.espn.com/tennis/story/_/id/18436908/austral
ian-open-2017-tournament-news-schedule-live-scores-tv-cover-age","payloadType":"st
r","repeat":"","crontab":"","once":false,"x":244.90000915527344,"y":84,"wires":[["
7cc616b1.7a5f8"]]},{"id":"587c390d.0c7e2","type":"debug","z":"a31695e2.13285","nam
e":"","active":true,"console":"false","complete":"features","x":650.9000244140625,
"y":68.80000305175781,"wires":[]}]
```

To import the flow in the Node-RED flow editor, copy the flow to your clipboard, and then from the menu icon, select **Import** → **Clipboard** (Figure 1-15 on page 15).

1.2.9 NLU Language detection

Basic language detection is included in every request with the Natural Language Understanding service; it is *not* a separate feature. It detects the natural language in which input text, HTML, or web-based content is written. Language identification functions can identify English, German, French, Italian, Portuguese, Russian, Spanish, and Swedish. These functions enable applications to categorize or filter content based on the language in which it was written.

> **Note:** The Language Detection API that was available as a separate API for the AlchemyLanguage service is now embedded in the `/analyze` endpoint.

See the following input and response examples.

Input

Input is from a web page or text, such as the following page, in a supported language and any Natural Language Understanding feature, for example, Concepts:

`https://www.ibm.com/ar-es/`

Response

Here is the response:

```
{
  "retrieved_url": "https://www.ibm.com/ar-es/",
  "concepts": [
    {
      "text": "IBM",
      "relevance": 0.965628,
      "dbpedia_resource": "http://es.dbpedia.org/resource/IBM"
    }
  ],
  "language": "es"
```

Use case example: Document language detection

Automatically identify the natural language that documents are written in and cluster documents based on their language.

NLU Language detection snippets

This section includes sample snippets to illustrate the use of language detection in Node.js.

Example 1: Detect the language of a web page

This first example shows how to detect the language of this web page:

`http://www.elpais.com.uy/`

The response is printed to the console or to the debug log (Node-RED) in ISO-639-1 format.

Node.js snippet 1

Example 1-29 shows the `nlu-languageDetection_1.js` snippet.

Example 1-29 Snippet: nlu-languageDetection_1.js

```
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   url : 'http://www.elpais.com.uy/',
   features : {
      concepts : {
         limit : 1
      }
   }
};
natural_language_understanding.analyze(parameters, function(error, response) {
   if (error) {
      onError(error, response); // function to be defined by you
   } else {
      console.log(JSON.stringify(response, null, 2));
      var language = response.language;
      // process the language text in 'language'
   }
});
```

Example 2: Detect the language of multiple web pages

This second example shows how to detect the language of multiple web pages:

- `http://www.elpais.com.uy/`
- `https://www.ibm.com/blockchain/what-is-blockchain.html`

The response is printed to the console.

Node.js snippet 2

Example 1-30 on page 56 shows the `nlu-languageDetection_2.js` snippet.

The pseudocode for this sample snippet is as follows:

1. Get a reference to the Watson Developer Cloud module.
2. Get a reference to the async module.
3. Get a reference to the NLU module passing your `username` and `password` in the options parameter.

4. By using the async module, parallelize the following API calls:
   a. Call the `/analyze` endpoint selecting any Natural Language Understanding feature. This example uses the `concepts` feature. As an input parameter, pass this URL:

      `https://www.ibm.com/blockchain/what-is-blockchain.html`
   b. Call the `/analyze` endpoint selecting any Natural Language Understanding feature. this example uses the `concepts` feature. As an input parameter pass this URL:

      `http://www.elpais.com.uy/`
5. After the response from both calls is received, print the results to the console.

Example 1-30 Snippet: nlu-languageDetection_2.js

```
var async = require('async');
var NaturalLanguageUnderstandingV1 =
require('watson-developer-cloud/natural-language-understanding/v1.js');
var natural_language_understanding = new NaturalLanguageUnderstandingV1({
   'username' : your_username_here,
   'password' : your_password_here,
   'version_date' : NaturalLanguageUnderstandingV1.VERSION_DATE_2017_02_27
});
var parameters = {
   features : {
      concepts : {
         limit : 1
      }
   }
};
function callLanguageDetection(callback) {
   natural_language_understanding.analyze(parameters, function(error, response) {
      if (error) { // do some error handling
         onError(error, null);
         callback(error, null);
      } else {
         callback(null, response);
      }
   });
}
async.parallel({
   "call1" : function(callback) {
      parameters.url = 'https://www.ibm.com/blockchain/what-is-blockchain.html';
      callLanguageDetection(callback);
   },
   "call2" : function(callback) {
      parameters.url = 'http://www.elpais.com.uy/';
      callLanguageDetection(callback);
   }
}, function(err, results) {
   if (err) {
      onError(err);
   } else {
      console.log(JSON.stringify(results, null, 2));
      // process the languages of asynchronous calls
      var lang1 = results.call1.language;
      var lang2 = results.call2.language;
   }
});
```

1.3 Migrating from AlchemyLanguage to Natural Language Understanding

This information applies only to users of the AlchemyLanguage API. For detailed information, see the Migrating from AlchemyLanguage topic in Watson Developer Cloud.

Starting on April 7, 2017, creating a new instance of IBM AlchemyAPI® on Bluemix is no longer possible. Existing service instances will be supported until March 7, 2018. To continue using AlchemyLanguage features, you must migrate your code to Natural Language Understanding. Natural Language Understanding offers a more economical pricing model and a consolidated API that is much easier to use.

These are the major changes:

- New API request syntax: send requests to the `/analyze` endpoint.
- New response structure (code that parses AlchemyLanguage output does not work for Natural Language Understanding output).
- JSON is the only output format.
- Text extraction is enabled by setting the `return_analyzed_text` parameter to `true`.
- Microformats are not supported.
- Date extraction is not supported (Publication Date is still supported in the Metadata feature).
- The `"quotations"` options for entities are not supported.
- The `"knowledgeGraph"` is not supported for concepts, keywords, or entities.
- TypedRelations from AlchemyLanguage is Relations in Natural Language Understanding.
- Relations from AlchemyLanguage is Semantic Roles in Natural Language Understanding.
- Entity types have changed.

1.4 References

See the following resources:

- Natural Language Understanding documentation:

  https://www.ibm.com/watson/developercloud/doc/natural-language-understanding/
- Natural Language Understanding demonstration:

  https://natural-language-understanding-demo.mybluemix.net/
- Natural Language Understanding in Watson API Explorer:

  https://watson-api-explorer.mybluemix.net/apis/natural-language-understanding-v1
- Migrating from AlchemyLanguage:

  https://www.ibm.com/watson/developercloud/doc/natural-language-understanding/migrating.html

2

Creating a Natural Language Understanding service in Bluemix

IBM Watson Developer Cloud offers a variety of services for developing cognitive applications. One of these services, which is the focus of this book, is the Watson Natural Language Understanding (NLU) service.

This chapter explains how to create an instance of the NLU service in Bluemix. That Natural Language Understanding service instance is required for the use cases described in this book.

The following topics are covered in this chapter:

- Requirements
- Creating the NLU service instance

2.1 Requirements

To create service and perform the use cases in this book, you must have a Bluemix account. You can register to create an account and log in at IBM Bluemix. When you log in, you are prompted to authenticate with your email or IBMid and password.

2.2 Creating the NLU service instance

The two ways to create the NLU service instance are as follows:

- Creating the NLU service instance from the Bluemix website
- Creating the NLU service instance by using Cloud Foundry commands

2.2.1 Creating the NLU service instance from the Bluemix website

To create the service instance from Bluemix, follow these steps:

1. Log in to the IBM Bluemix website.
2. When the home page opens, click **Catalog**.
3. On the IBM Bluemix Catalog page, scroll to the Services section, select **Watson**, and then click **Natural Language Understanding**. Another way to find the service is to use the search filter (Figure 2-1).

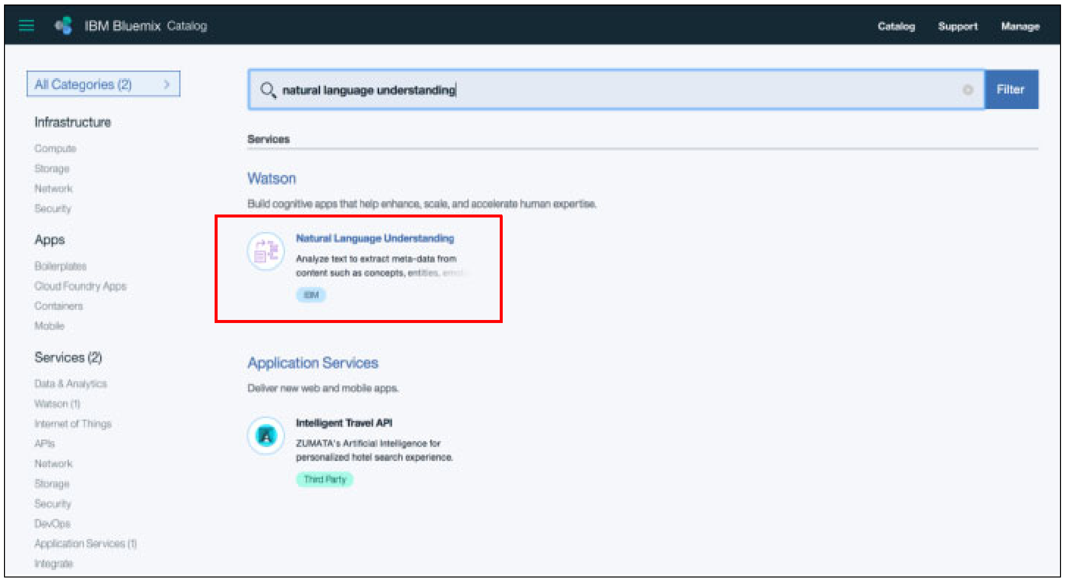
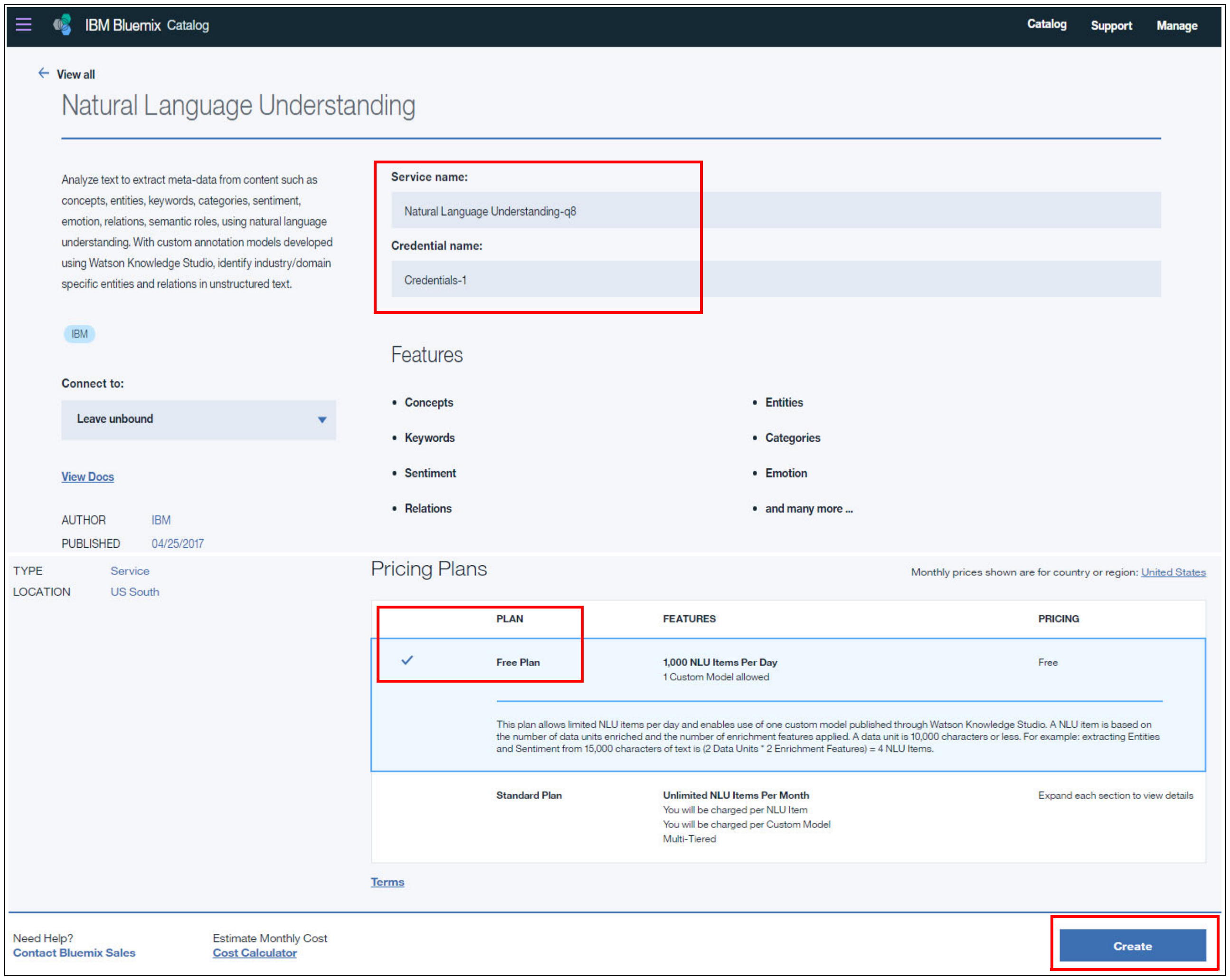


Figure 2-1 Natural Language Understanding in the Bluemix catalog

4. On the Natural Language Understanding page (Figure 2-2), create the service instance:
   a. You can *either* change the Service name and Credentials name fields by using your personal choices *or* keep the default values.
   b. Select the plan. In this example, the **Free Plan** tier is selected.
   c. Click **Create**.



Natural Language Understanding

Analyze text to extract meta-data from content such as concepts, entities, keywords, categories, sentiment, emotion, relations, semantic roles, using natural language understanding. With custom annotation models developed using Watson Knowledge Studio, identify industry/domain specific entities and relations in unstructured text.

IBM

Connect to: Leave unbound

View Docs

AUTHOR	IBM
PUBLISHED	04/25/2017
TYPE	Service
LOCATION	US South

Service name: Natural Language Understanding-q8

Credential name: Credentials-1

Features

- Concepts
- Keywords
- Sentiment
- Relations
- Entities
- Categories
- Emotion
- and many more ...

Pricing Plans

Monthly prices shown are for country or region: United States

	PLAN	FEATURES	PRICING
✓	Free Plan	1,000 NLU Items Per Day 1 Custom Model allowed	Free
	Standard Plan	Unlimited NLU Items Per Month You will be charged per NLU Item You will be charged per Custom Model Multi-Tiered	Expand each section to view details

This plan allows limited NLU items per day and enables use of one custom model published through Watson Knowledge Studio. A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units * 2 Enrichment Features) = 4 NLU Items.

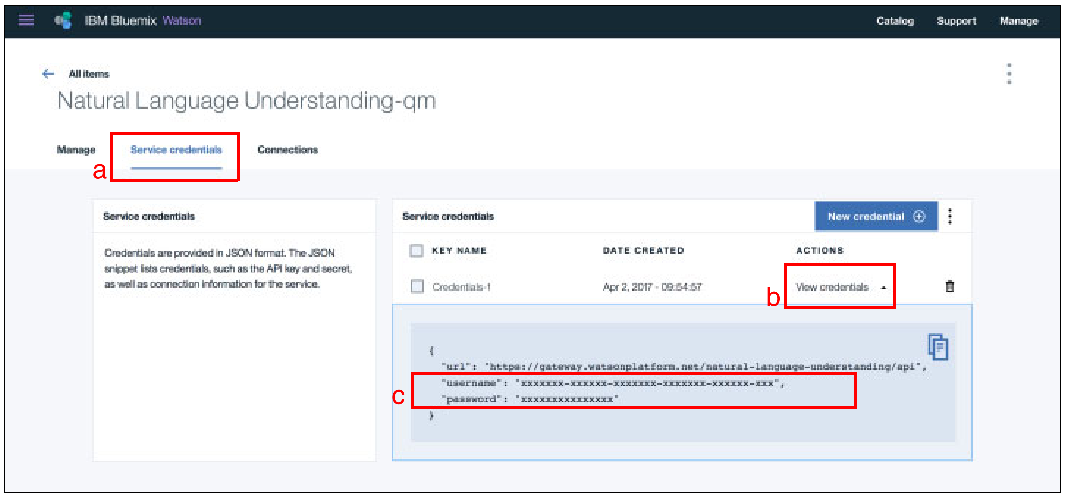
Terms

Need Help? Contact Bluemix Sales

Estimate Monthly Cost Cost Calculator

Figure 2-2 Creating Natural Language Understanding service instance

5. Get the username and password credentials (Figure 2-3) from the service instance you created in your Bluemix space:
   a. Click the **Service credentials** tab.
   b. Click **View credentials** to show the NLU service credentials.
   c. Store the username and password values in a separate file or note for future reference.



Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

```
{
  "url": "https://gateway.watsonplatform.net/natural-language-understanding/api",
  "username": "xxxxxxx-xxxxxx-xxxxxxx-xxxxxxx-xxxxxx-xxx",
  "password": "xxxxxxxxxxxxxxx"
}
```

Figure 2-3 Get username and password from Natural Language Understanding service instance

2.2.2 Creating the NLU service instance by using Cloud Foundry commands

To create the service instance with Cloud Foundry commands, follow these steps:

1. Download the Cloud Foundry software and install it on your computer.
2. Open a command prompt.
3. Run the **`cf login`** command and insert the email and password for your Bluemix account in the sequence that is shown in Example 2-1.

Example 2-1 Run login and provide email and password for the Bluemix account

```
cf login
      API endpoint: https://api.ng.bluemix.net

Email> <PUT_YOUR_BLUEMIX_EMAIL_ACCOUNT>

Password> <PUT_YOUR_PASSWORD_ACCOUNT>

Authenticating...
OK
```

4. If you have multiple organizations, select an organization and a Bluemix space to host the service (Example 2-2).

Example 2-2 Select an organization and Bluemix space

```
Select an org (or press enter to skip):
1. <YOUR_BLUEMIX_EMAIL_ACCOUNT>
2. Organization_2
3. Organization_3
```

```
Org> 1
Targeted org <YOUR_ORGANIZATION>

Select a space (or press enter to skip):
1. dev
2. itso-ed6000
3. space_3

Space> 1
Targeted space dev


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           <YOUR_BLUEMIX_EMAIL_ACCOUNT>
Org:            <YOUR_ORGANIZATION>
Space:          <YOUR_SPACE>
```

5. Run the following command to create an instance of the service:

   `cf create-service <service> <service_plan> <ser-vice_instance>`

   About the command:

`cf create-service`	The Cloud Foundry command to create the service instance.
`<service>`	The name of the service you want to create an instance of.
`<service_plan>`	The pricing plan.
`<service_instance>`	The service instance name you want to use.

   Example 2-3 shows the command.

Example 2-3 The cf create-service command

```
cf create-service natural-language-understanding free "NLU-ITSO-ED6000"
Creating service instance NLU-ITSO-ED6000 in org <YOUR_ORGANIZATION>/ space
<YOUR_SPACE> as <YOUR_BLUEMIX_EMAIL_ACCOUNT>......
OK
```

6. List the service information by using the **`cf service <service_name>`** command to confirm that it was successfully created (Example 2-4).

Example 2-4 The cf service command

```
cf service NLU-ITSO-ED6000

Service instance: NLU-ITSO-ED6000
Service: natural-language-understanding
Bound apps:
Tags:
Plan: free
Description: Analyze text to extract meta-data from content such as concepts,
entities, emotion, relations, senti-ment and more.
Documentation url:
https://www.ibm.com/watson/developercloud/doc/natural-language-understanding/
Dashboard:
https://www.ibm.com/watson/developercloud/dashboard/en/natural-language-underst
anding-dashboard.html
```

```
Last Operation
Status: create succeeded
Message:
Started: 2017-04-28T18:52:54Z
Updated:
```

7. Create user and password credentials to access the service by using this command:

   ```
   cf create-service-key <service_instance> <service_key>
   ```

   About the command:

cf create-service-key	The Cloud Foundry command to create the service key with user and password.
<service_instance>	The name of the NLU service instance.
<service_key>	The name of the service key you want to create.

   Example 2-5 shows this command.

   Example 2-5 The cf create-service-key command

   ```
   cf create-service-key "NLU-ITSO-ED6000" nlu-keys
   Creating service key nlu-keys for service instance NLU-ITSO-ED6000 as
   <YOUR_BLUEMIX_EMAIL_ACCOUNT>...
   OK
   ```

8. Get the username and password in order to access the service later by running the following command:

   ```
   cf service-key <service_instance> <service_key>
   ```

   About the command:

cf service-key	The Cloud Foundry command to view the username and password in the service key.
<service_instance>	The name of the service instance.
<service_key>	The name of the service key.

   Example 2-6 shows this command.

   Example 2-6 Use cf service-key to get username and password

   ```
   cf service-key "NLU-ITSO-ED6000" nlu-keys
   Getting key nlu-keys for service instance NLU-ITSO-ED6000 as
   <YOUR_BLUEMIX_EMAIL_ACCOUNT>...

   {
    "password": "xxxxxxxxxxxx",
    "url":
   "https://gateway.watsonplatform.net/natural-language-understanding/api",
    "username": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
   }
   ```

3

Sentiment and personality analysis

Natural Language Understanding (NLU) is a family of natural language processing (NLP) APIs that can be combined with other Watson services to perform social media sentiment analysis. These services can be combined to create practical applications, for example, to gain insights about the personality of your customers and their sentiment to your brand or products.

This chapter describes steps to create a simple application to analyze Tweets from a Twitter handle by using the `sentiment` feature of the Natural Language Understanding service, combined with the Personality Insights and Insights for Twitter services.

The following topics are covered in this chapter:

- Getting started
- Architecture
- Two ways to deploy the application: Step-by-step and quick deploy
- Step-by-step implementation
- Quick deployment of application
- References

3.1 Getting started

To start, read through the objectives, prerequisites, and expected results of this use case.

3.1.1 Objectives

By the end of this chapter, you should be able to accomplish these objectives:

- Create a Sentiment Analysis application in Node.js.
- Run the Sentiment Analysis application in IBM Bluemix.
- Feed the application with Tweets from one Twitter handle by using the Insights for Twitter service.
- Use the Personality Insights service to analyze personality of the Twitter handle owner.
- Use the Bluemix web user interface to create and manage services.

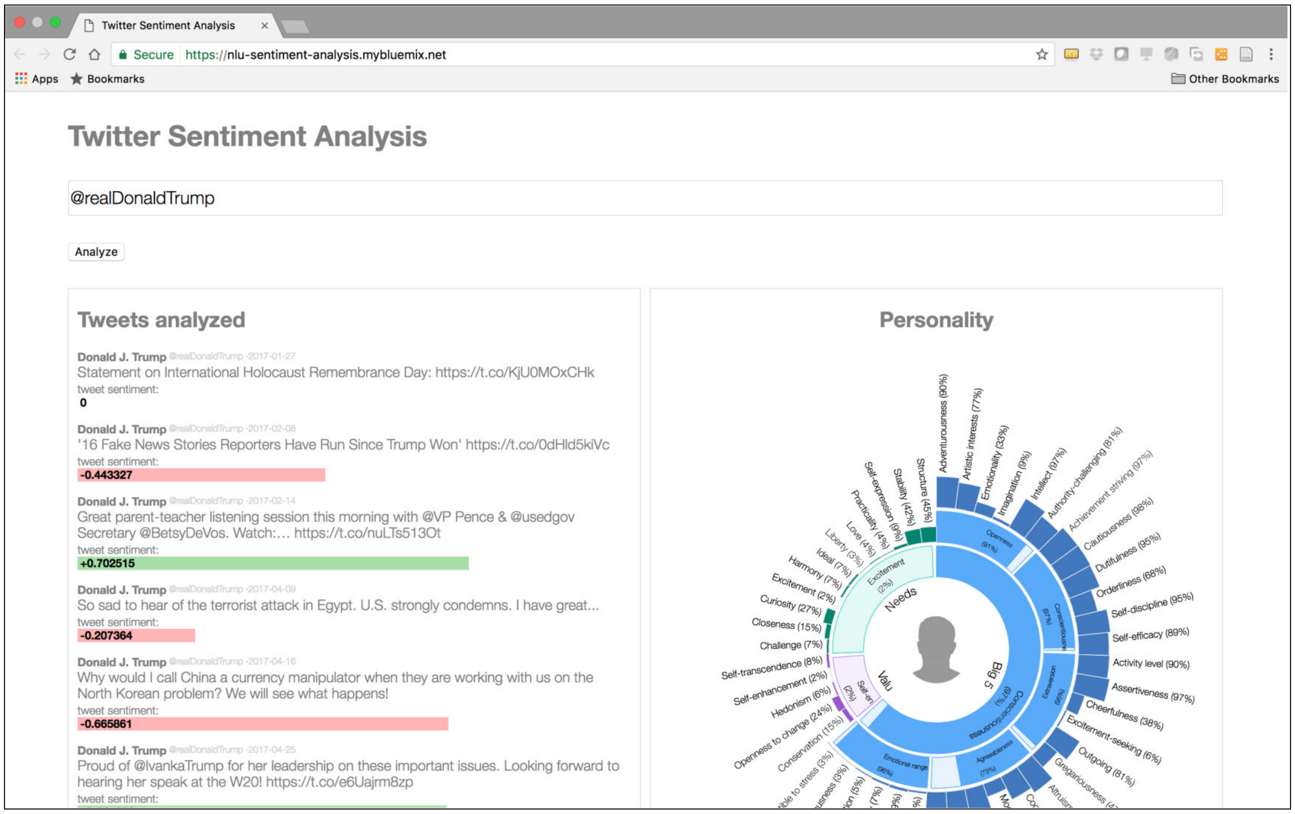
3.1.2 Prerequisites

You must have these prerequisites:

- Bluemix account.
- Review Chapter 1, “Basics of Watson Natural Language Understanding service” on page 1 and Chapter 2, “Creating a Natural Language Understanding service in Bluemix” on page 59.
- A web browser (Chrome, Safari, Firefox, Internet Explorer).
- An installed CloudFoundry CLI client.
- Basic JavaScript, HTML, and CSS skills.
- Git client:
  - Git download and installation
  - Cloud Foundry download and installation

3.1.3 Expected results

Figure 3-1 shows the expected result after running the application you develop in this chapter. You are asked for a Twitter handle and after you provide it, Tweets will be retrieved, analyzed for sentiment, and the personality of the user will be depicted.



Twitter Sentiment Analysis

@realDonaldTrump

Tweets analyzed

Donald J. Trump @realDonaldTrump ·2017-01-27
Statement on International Holocaust Remembrance Day: https://t.co/KjU0MOxCHk
tweet sentiment:
0

Donald J. Trump @realDonaldTrump ·2017-02-08
'16 Fake News Stories Reporters Have Run Since Trump Won' https://t.co/0dHld5kiVc
tweet sentiment:
-0.443327

Donald J. Trump @realDonaldTrump ·2017-02-14
Great parent-teacher listening session this morning with @VP Pence & @usedgov Secretary @BetsyDeVos. Watch:… https://t.co/nuLTs513Ot
tweet sentiment:
+0.702515

Donald J. Trump @realDonaldTrump ·2017-04-09
So sad to hear of the terrorist attack in Egypt. U.S. strongly condemns. I have great...
tweet sentiment:
-0.207364

Donald J. Trump @realDonaldTrump ·2017-04-16
Why would I call China a currency manipulator when they are working with us on the North Korean problem? We will see what happens!
tweet sentiment:
-0.665861

Donald J. Trump @realDonaldTrump ·2017-04-25
Proud of @IvankaTrump for her leadership on these important issues. Looking forward to hearing her speak at the W20! https://t.co/e6Uajrm8zp
tweet sentiment:

Figure 3-1 Sentiment and Personality Analysis expected result

3.2 Architecture

Figure 3-2 shows an overview of the application use case flow.

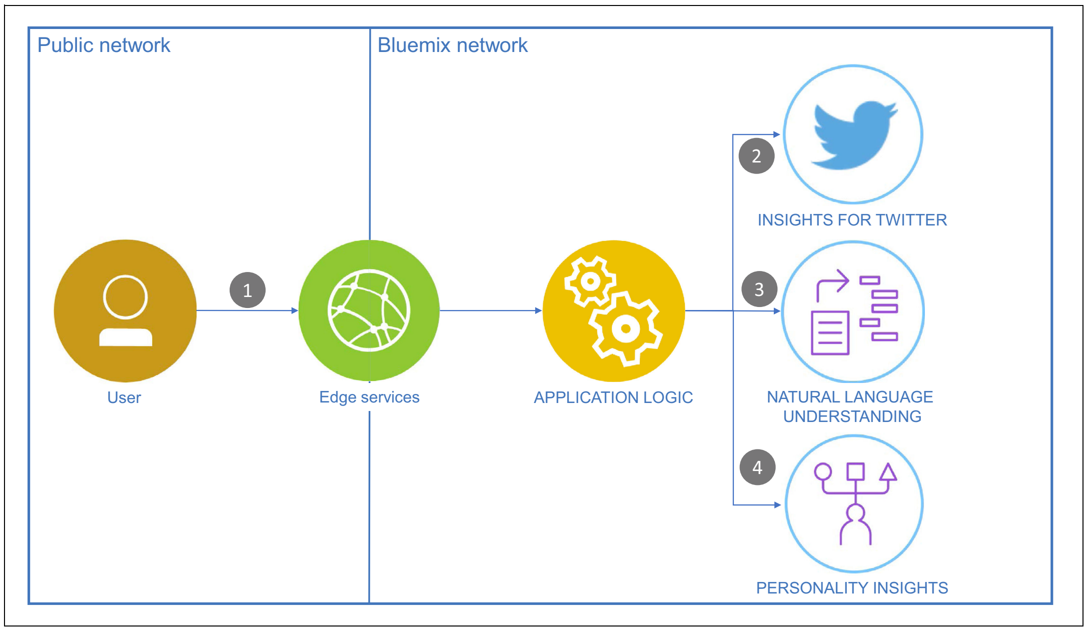


Figure 3-2 Architecture overview

The architecture has the following flow:

1. User enters a Twitter handle.
2. The Tweets are retrieved from the handle by using the Insights for Twitter service.
3. The text is passed to the Natural Language Understanding service (Sentiment feature) to analyze the sentiment of each Tweet.
4. All Tweets are passed to the Personality Insights service.

Figure 3-3 shows the use case design flow.

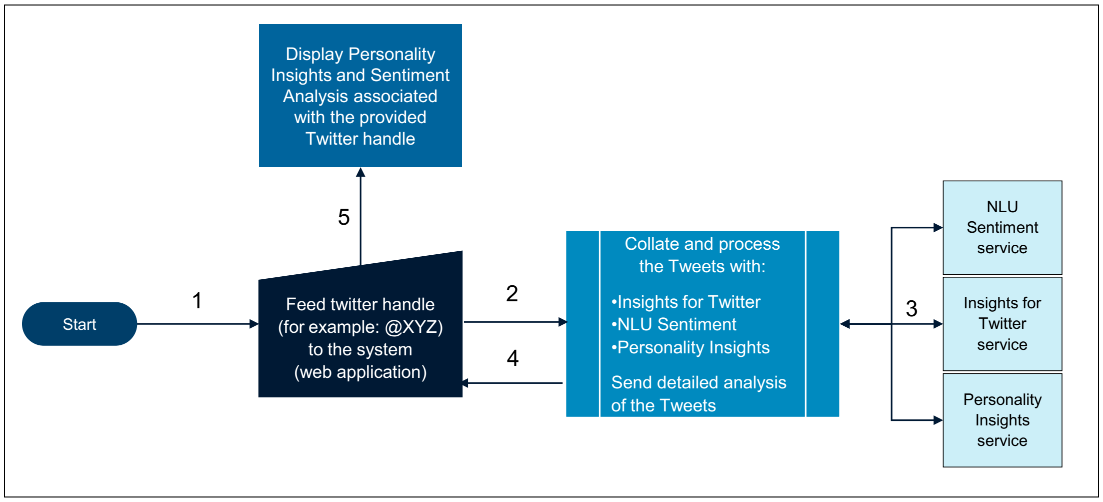

Display Personality Insights and Sentiment Analysis associated with the provided Twitter handle
5
Start
1
Feed twitter handle (for example: @XYZ) to the system (web application)
2
4
Collate and process the Tweets with:
•Insights for Twitter
•NLU Sentiment
•Personality Insights
Send detailed analysis of the Tweets
3
NLU Sentiment service
Insights for Twitter service
Personality Insights service

Figure 3-3 Use case flow design

The use case has the following flow:

1. The Twitter handle (for example, `@XYZ`) is input to the web application.
2. The Twitter handle is then fed to the back-end processing system.
3. Integration occurs with NLU Sentiment, Insights for Twitter, and Personality Insights services.
4. The back-end system passes the results of the Tweets analysis with Personality Insights and Sentiment Analysis associated with Twitter handle to the front end.
5. The Personality Insights and Sentiment Analysis that are associated with the Twitter handle are displayed to the user.

3.3 Two ways to deploy the application: Step-by-step and quick deploy

Two Git repositories are provided for this use case:

- Step-by-step implementation (incomplete) version of the application.

  This repository contains an incomplete version of the application and is used in all sections of 3.4, “Step-by-step implementation” on page 69. This version takes you through the key steps to integrate the IBM Watson APIs with the application logic.

- Quick-deploy (complete) version of the application

  This repository contains the final version of the application. If you want to bypass the implementation steps and instead run the application as a demonstration, download this full version. Downloading and running this full version demonstration is explained in 3.5, “Quick deployment of application” on page 87.

3.4 Step-by-step implementation

Implementation of the application involves the following steps:

1. Creating the Natural Language Understanding service instance
2. Creating the Insights for Twitter service instance
3. Creating the Personality Insights service instance
4. Coding the Node.js application
5. Pushing the application into Bluemix
6. Binding all the services to the Bluemix application (alternative option).
7. Testing the application

3.4.1 Creating the Natural Language Understanding service instance

The steps for creating the Natural Language Understanding (NLU) service instance and retrieving the service credentials are described in Chapter 2, “Creating a Natural Language Understanding service in Bluemix” on page 59.

3.4.2 Creating the Insights for Twitter service instance

Complete the following steps:

1. Log in to your IBM Bluemix account.
2. When the home page opens, click **Catalog**.
3. On the IBM Bluemix Catalog page, scroll to the Services section, select **Data & Analytics**, and then click **Insights for Twitter**. Another way to find the Insights for Twitter service is to use the search filter to search for `twitter` (Figure 3-4).

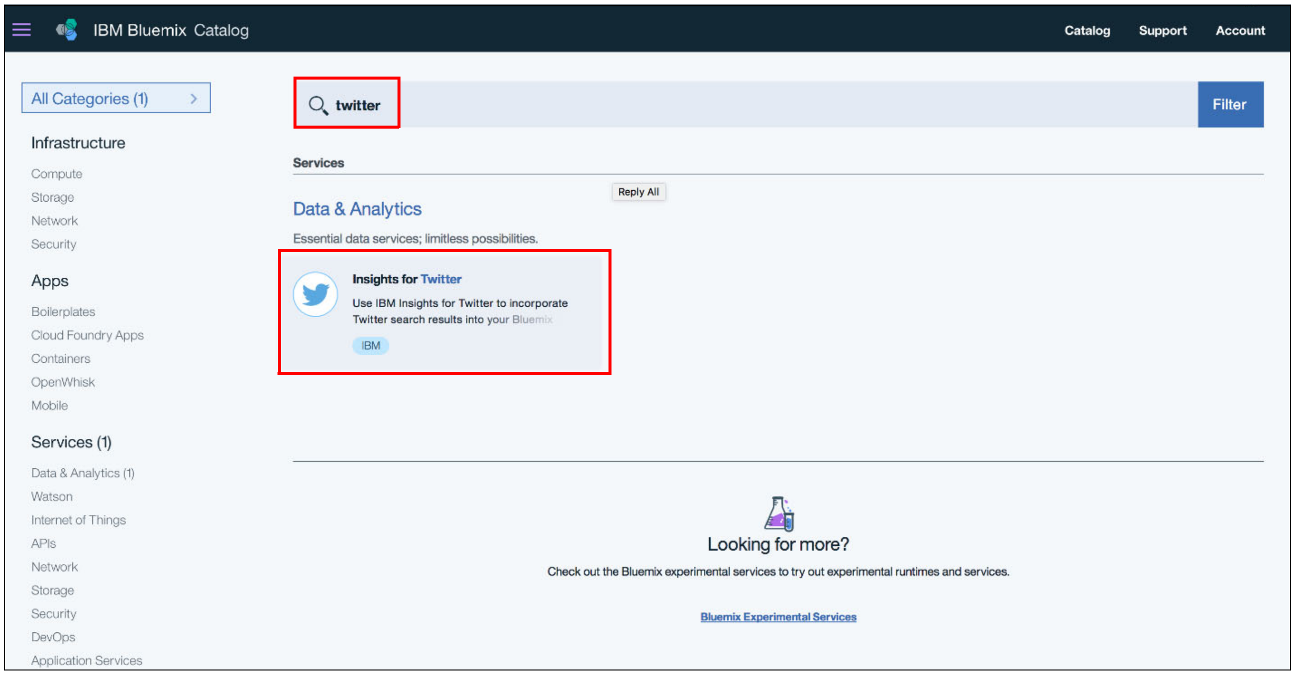
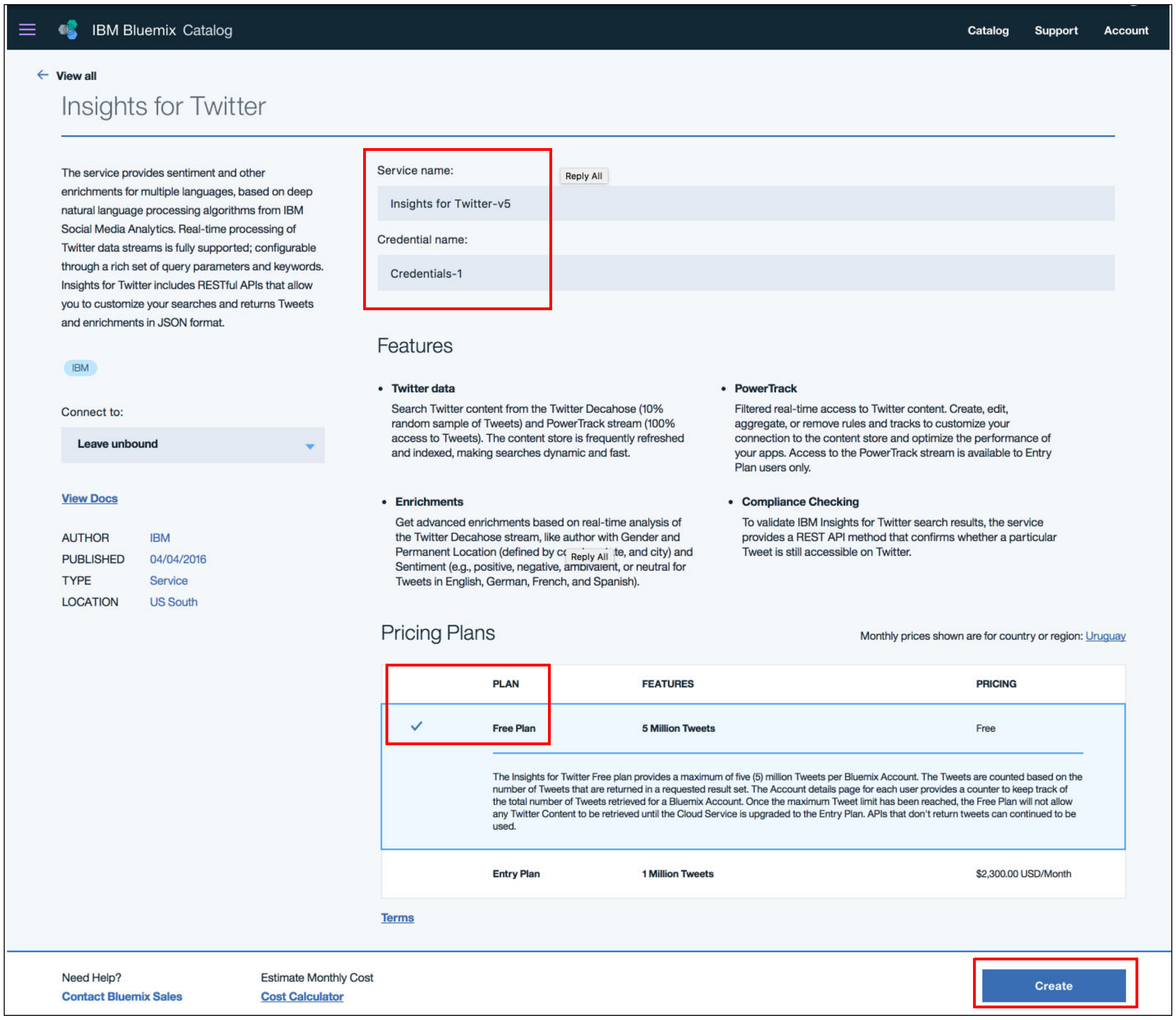


Figure 3-4 Insights for Twitter service in the Bluemix catalog

4. On the Insights for Twitter page (Figure 3-5), create the service:
   a. You can *either* change the Service name and Credentials name fields by using your personal choices *or* keep the default values.
   b. Select the plan. In this example, the **Free Plan** tier is selected.
   c. Click **Create**.



Insights for Twitter

The service provides sentiment and other enrichments for multiple languages, based on deep natural language processing algorithms from IBM Social Media Analytics. Real-time processing of Twitter data streams is fully supported; configurable through a rich set of query parameters and keywords. Insights for Twitter includes RESTful APIs that allow you to customize your searches and returns Tweets and enrichments in JSON format.

Connect to: Leave unbound

AUTHOR	IBM
PUBLISHED	04/04/2016
TYPE	Service
LOCATION	US South

Service name: Insights for Twitter-v5

Credential name: Credentials-1

Features

- **Twitter data**
  Search Twitter content from the Twitter Decahose (10% random sample of Tweets) and PowerTrack stream (100% access to Tweets). The content store is frequently refreshed and indexed, making searches dynamic and fast.
- **PowerTrack**
  Filtered real-time access to Twitter content. Create, edit, aggregate, or remove rules and tracks to customize your connection to the content store and optimize the performance of your apps. Access to the PowerTrack stream is available to Entry Plan users only.
- **Enrichments**
  Get advanced enrichments based on real-time analysis of the Twitter Decahose stream, like author with Gender and Permanent Location (defined by co[illegible]te, and city) and Sentiment (e.g., positive, negative, ambivalent, or neutral for Tweets in English, German, French, and Spanish).
- **Compliance Checking**
  To validate IBM Insights for Twitter search results, the service provides a REST API method that confirms whether a particular Tweet is still accessible on Twitter.

Pricing Plans

Monthly prices shown are for country or region: Uruguay

	PLAN	FEATURES	PRICING
✓	Free Plan	5 Million Tweets	Free
	Entry Plan	1 Million Tweets	$2,300.00 USD/Month

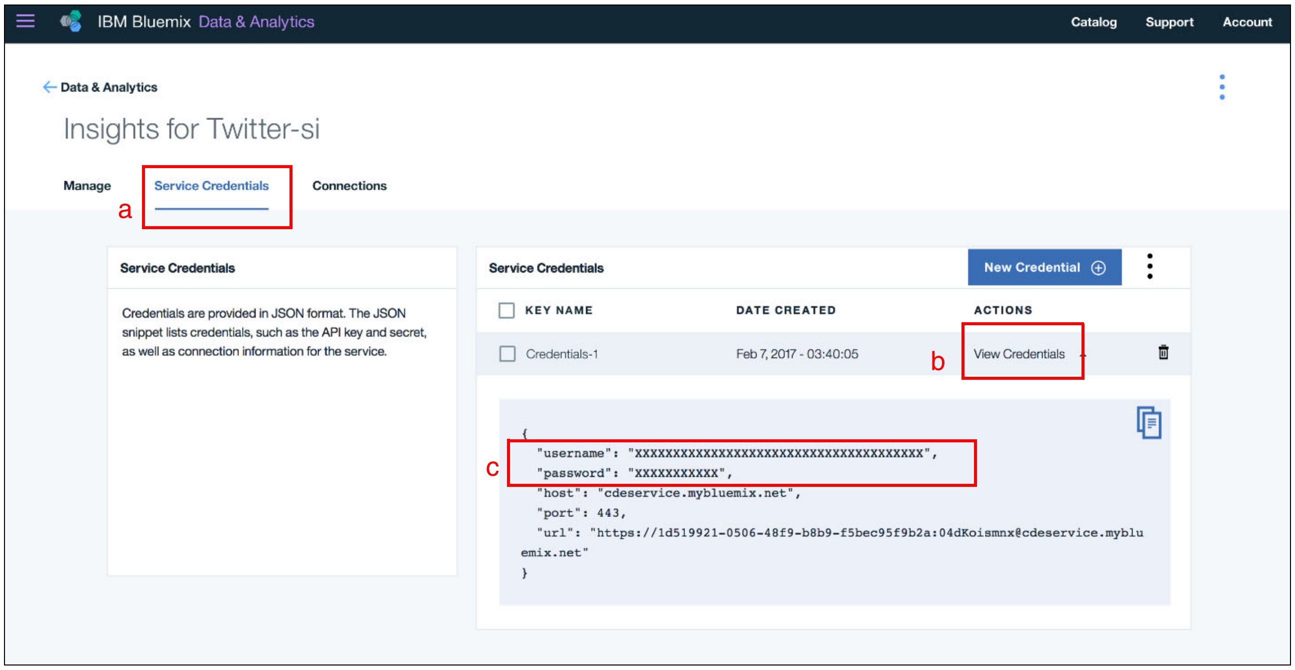
The Insights for Twitter Free plan provides a maximum of five (5) million Tweets per Bluemix Account. The Tweets are counted based on the number of Tweets that are returned in a requested result set. The Account details page for each user provides a counter to keep track of the total number of Tweets retrieved for a Bluemix Account. Once the maximum Tweet limit has been reached, the Free Plan will not allow any Twitter Content to be retrieved until the Cloud Service is upgraded to the Entry Plan. APIs that don't return tweets can continued to be used.

Need Help? Contact Bluemix Sales

Estimate Monthly Cost Cost Calculator

Figure 3-5 Insights for Twitter service creation

5. After the service instance is created in your Bluemix space, get the username and password credentials from the service instance (Figure 3-6):
   a. Click the **Service credentials** tab.
   b. Click **View credentials** to show the service credentials.
   c. Store the username and password values in a separate file or note for future reference.



Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

KEY NAME	DATE CREATED	ACTIONS
Credentials-1	Feb 7, 2017 - 03:40:05	View Credentials

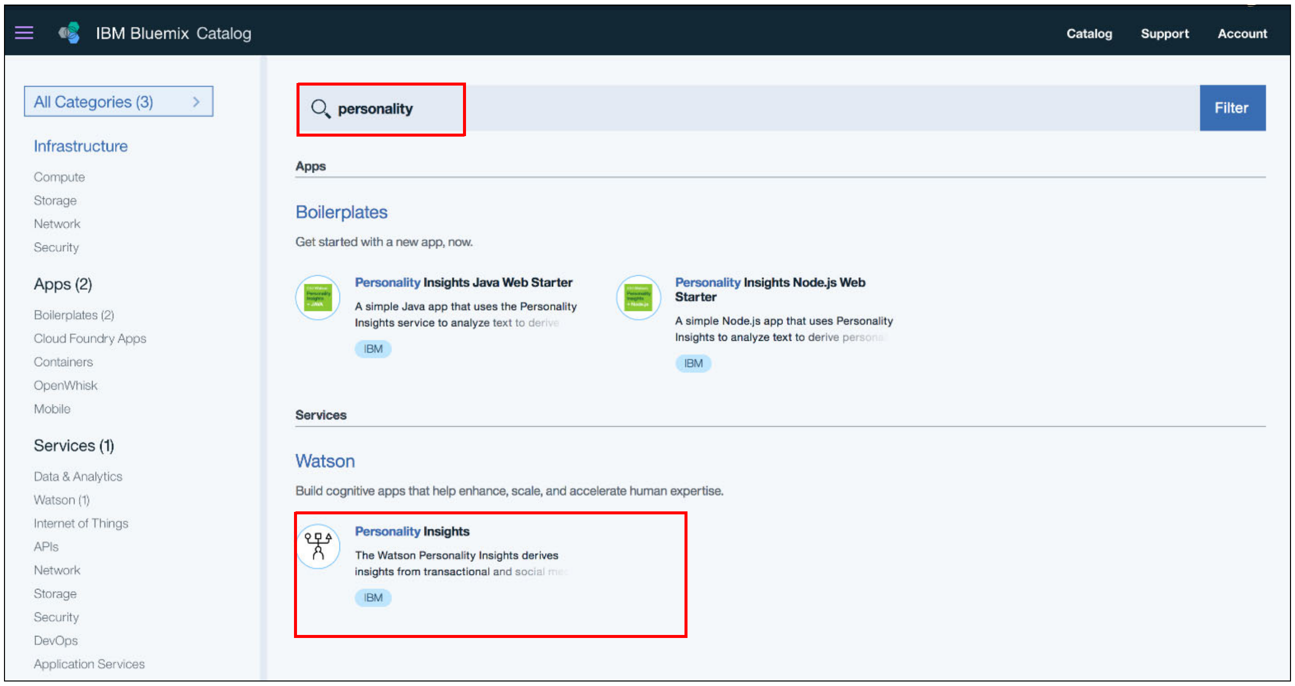
```
{
  "username": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "password": "XXXXXXXXXXX",
  "host": "cdeservice.mybluemix.net",
  "port": 443,
  "url": "https://1d519921-0506-48f9-b8b9-f5bec95f9b2a:04dKoismnx@cdeservice.myblu
emix.net"
}
```

Figure 3-6 Get username and password from Insights for Twitter service instance

3.4.3 Creating the Personality Insights service instance

Complete the following steps:

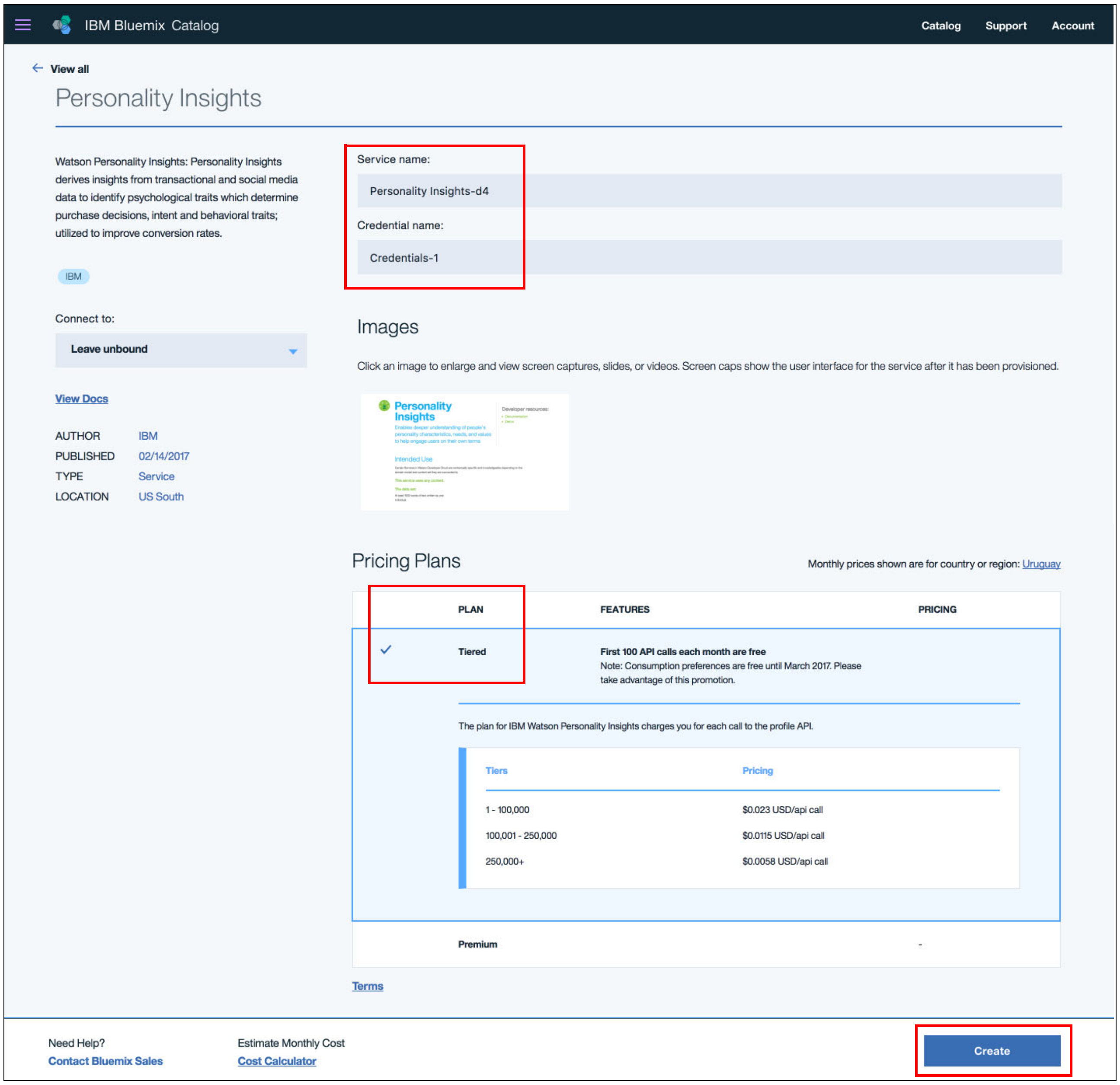
1. Log in to your IBM Bluemix account.
2. When the home page opens, click **Catalog**.
3. On the IBM Bluemix Catalog page (Figure 3-7), type `personality` in the search filter. Several results are display. Click **Personality Insights**.



IBM Bluemix Catalog
Catalog
Support
Account
All Categories (3)
Infrastructure
Compute
Storage
Network
Security
Apps (2)
Boilerplates (2)
Cloud Foundry Apps
Containers
OpenWhisk
Mobile
Services (1)
Data & Analytics
Watson (1)
Internet of Things
APIs
Network
Storage
Security
DevOps
Application Services
personality
Filter
Apps
Boilerplates
Get started with a new app, now.
Personality Insights Java Web Starter
A simple Java app that uses the Personality Insights service to analyze text to derive
IBM
Personality Insights Node.js Web Starter
A simple Node.js app that uses Personality Insights to analyze text to derive persona
IBM
Services
Watson
Build cognitive apps that help enhance, scale, and accelerate human expertise.
Personality Insights
The Watson Personality Insights derives insights from transactional and social me
IBM

Figure 3-7 Search for and select Personality Insights

4. On the Personality Insights page (Figure 3-8), create the service:
   a. You can *either* change the Service name and Credentials name fields by using your personal choices *or* keep the default values.
   b. Select the plan. In this example, the **Tiered Plan** tier is selected.
   c. Click **Create**.



Personality Insights

Watson Personality Insights: Personality Insights derives insights from transactional and social media data to identify psychological traits which determine purchase decisions, intent and behavioral traits; utilized to improve conversion rates.

IBM

Connect to: Leave unbound

View Docs

AUTHOR	IBM
PUBLISHED	02/14/2017
TYPE	Service
LOCATION	US South

Service name: Personality Insights-d4

Credential name: Credentials-1

Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the service after it has been provisioned.

Pricing Plans

Monthly prices shown are for country or region: Uruguay

PLAN	FEATURES	PRICING
✓ Tiered	**First 100 API calls each month are free** Note: Consumption preferences are free until March 2017. Please take advantage of this promotion.	
Premium		-

The plan for IBM Watson Personality Insights charges you for each call to the profile API.

Tiers	Pricing
1 - 100,000	$0.023 USD/api call
100,001 - 250,000	$0.0115 USD/api call
250,000+	$0.0058 USD/api call

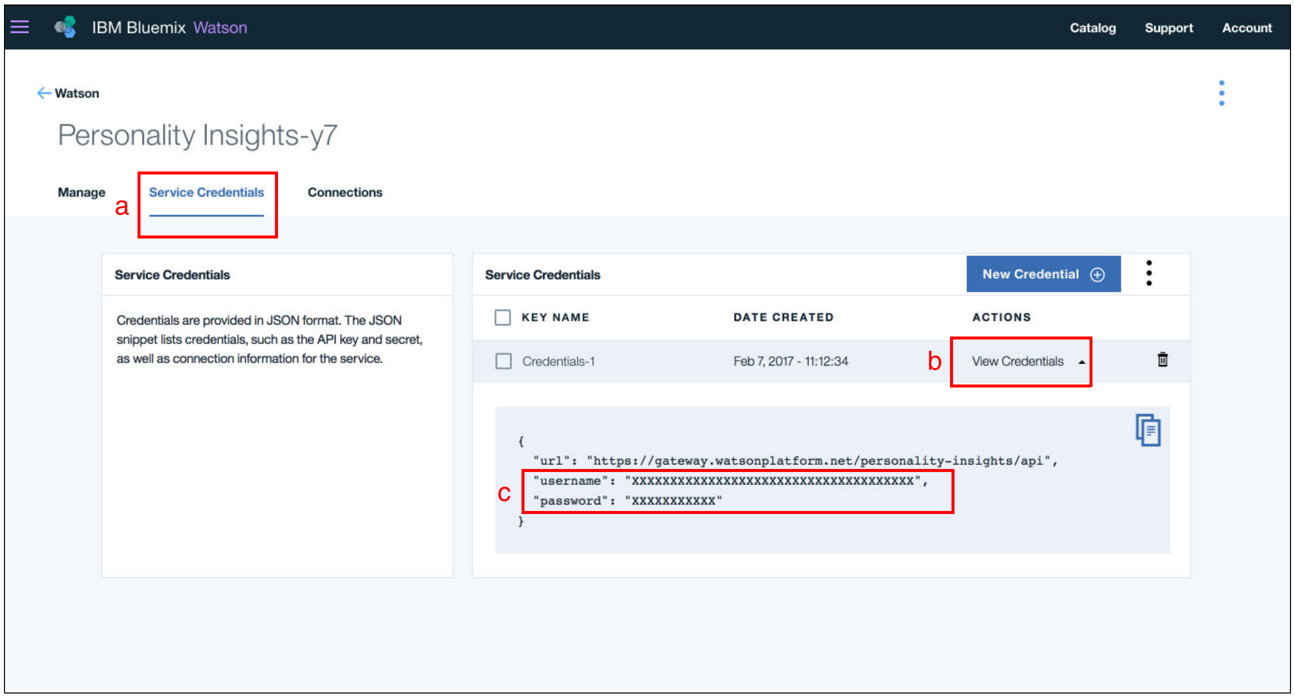
Terms

Need Help? Contact Bluemix Sales

Estimate Monthly Cost Cost Calculator

Figure 3-8 Personality Insights service creation

5. After the service instance is created in your Bluemix space, get the username and password credentials from the service instance (Figure 3-9):
   a. Click the **Service credentials** tab.
   b. Click **View credentials** to show the service credentials.
   c. Store the username and password values in a separate file or note for future reference.



Service Credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

Service Credentials

KEY NAME	DATE CREATED	ACTIONS
Credentials-1	Feb 7, 2017 - 11:12:34	View Credentials

```
{
  "url": "https://gateway.watsonplatform.net/personality-insights/api",
  "username": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "password": "XXXXXXXXXXX"
}
```

Figure 3-9 Get username and password from Personality Insights service instance

3.4.4 Coding the Node.js application

Coding involves two basic steps:

1. Clone the base structure from Git.
2. Code the remainder of the application.

Clone the base structure from Git

Complete these steps:

1. Fork the base structure from the following Git repository:

   `https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs-student`

   a. Open a new command window.
   b. Change from the current working directory to the location where you want to clone the remote Git repository.
   c. Code the Git repository by using the **`git clone`** command:

   ```
   git clone
   https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs-student.git
   ```

2. Change to the newly created directory:

   `redbooks-nlu-201-sentiment-nodejs-student`

Example 3-1 shows an Express Node.js application that follows the naming conventions and structure, which all Express applications have.

Example 3-1 Express Node.js application structure

```
root directory
   bin
   helpers
   public
   routes
   views
   app.js
   package.json
   README.md
```

You code the remainder of the application, as described in the next section ("Code the remainder of the application" on page 76).

Code the remainder of the application

The application contains files to accelerate the development of the application and some blank lines (marked with XXX) and instructions so that you can complete the code. To code the remainder of the application, you complete the blank lines. Start in the back-end layer and then work in the presentation layer.

Back-end layer

Complete the back-end layer code:

1. At the beginning of the app.js file (lines 1 - 4), replace the value of the variables listed in Example 3-2 with their corresponding credentials values.

   Example 3-2 Entering the service instance credentials

   ```
   var NLU_USER = XXX; // REPLACE WITH YOUR NLU USER
   var NLU_PASSWORD = XXX; // REPLACE WITH YOUR NLU PASSWORD
   var TWITTER_INSIGHTS_USER = XXX; // REPLACE WITH YOUR TWITTER_INSIGHTS USER
   var TWITTER_INSIGHTS_PASSWORD = XXX; // REPLACE WITH YOUR TWITTER_INSIGHTS
   PASSWORD
   ```

2. Navigate to the app.js file and replace the last XXX (line 83) with a call to the Natural Language Understanding service with the sentiment feature. The sentiment feature is set in the parameters var (lines 77 - 80). In the callback, if an error occurs, load the nlu_sentiment property of the input Tweet with the error string (Example 3-3); otherwise, load the same property with the sentiment.document object returned from the call to the NLU service.

   Example 3-3 Calling NLU analyze endpoint with sentiment feature and handling of errors

   ```
   natural_language_understanding.analyze(parameters, function(error,
   sentimentResponse) {
       if (error) {
           tweet.nlu_sentiment = "ERROR"
       } else {
           tweet.nlu_sentiment = sentimentResponse.sentiment.document;
       }
       return resolve();
   });
   ```

3. The Insights for Twitter service requires you to specify the number of Tweets to return. In this example, this number is specified by setting the `size` parameter to the appropriate value (default value is 100, max is 500) in line 42 (Example 3-4).

   Also specify the query string that is used to search Tweets. This query string must be URL-encoded and must conform to support operators and terms (see the REST API documentation section at the Using the Insights for Twitter REST APIs web page).

   In this example, the query string is formed in the presentation layer and is passed in the query property of the body of the request. In line 43, set the query with this property (Example 3-4).

   Example 3-4 Set the size and query variables

   ```
   var size = 200 /* default value, max = 500 */;
   var query = req.body.query;
   ```

4. In `callNLUAPI`, set the value of the text parameter in the `parameters` object in line 78 (Example 3-5).

   Example 3-5 Set the value of text parameter

   ```
   var parameters = {
       text: tweet.message.body,
       features : {
           sentiment : {}
       }
   };
   ```

5. In the `helpers/personality.js` module, complete the `personalityInsights` object by replacing the `XXX` (lines 4 and 5) with the username and password you obtained from the Personality Insights service instantiation in Bluemix in step 5 on page 75.

6. Complete the `toContentItem` function, line 10, (Example 3-6) in the `personality.js` module. The objective is to map the Tweet received as a parameter to a `ContentItem` expected in the Personality Insights API.

   Example 3-6 Complete the toContentItem function

   ```
   function toContentItem(tweet) {
       return {
           id: tweet.message.id,
           language: tweet.message.twitter_lang,
           contenttype: 'text/plain',
           content: tweet.message.body.replace('[^(\\x20-\\x7F)]*', ''),
           created: Date.parse(tweet.message.postedTime),
       };
   }
   ```

7. Complete the `Promise` declaration in the `getProfile` function (line 16), in the `personality.js` module (Example 3-7).

   Example 3-7 Complete the Promise declaration in the getProfile function

   ```
   function getProfile(params) {
       return new Promise(function (resolve, reject) {
           personalityInsights.profile(params, function (error, response) {
               if (error) {
                   reject(error);
               } else {
   ```

```
                resolve(response);
            }
        })
    });
}
```

Presentation layer

You now work in the presentation layer.

In the `/public/javascripts/index.js` file, you will complete the `analyze` function. This function is responsible for calling the back-end modules in order to get the Tweets and their corresponding sentiment, and then it loads the personality insights.

Complete these steps (Example 3-8):

1. Review the message object (line 9). The `query` parameter specifies the query string that will be used to search for Tweets in the Insights for Twitter service.
2. After the search query is defined, a POST request is created.
3. In the response of the POST request, you complete the call to these functions:
   - The `loadTweetFeed` function modifies the DOM of the HTML document to list all the Tweets and their corresponding sentiment (line 18).
   - The `loadPersonality` function modifies the DOM of the HTML document to include the sunburst chart (line 19).

Example 3-8 Complete the analyze function in the index.js file

```
function analyze() {
    showProcessing(true);
    var message = {
        "query": 'from:'.concat($("#tw_search").val().replace("@",
"")).concat(" posted:2017-01-01")
    };
    var request = new XMLHttpRequest();
    request.open('POST', 'analyze', true);
    request.setRequestHeader("Content-type", "application/json");
    request.onload = function () {
        // process response
        var result = JSON.parse(request.response);
        loadTweetFeed(result.tweets);
        loadPersonality(result.personality, result.error);
        showProcessing(false);
    };
    request.onerror = function () {
        showProcessing(false);
    }
    request.send(JSON.stringify(message));
}
```

3.4.5 Pushing the application into Bluemix

Complete these steps:

1. Open a command prompt window and change to the directory where the project is downloaded. For example, use this command:

   ```
   cd redbooks-nlu-201-sentiment-nodejs-student
   ```

2. Log in with the CLI client **`cf login`** command:

   a. You are prompted for your user name and password.
   b. If you have multiple organizations, select an organization.
   c. If you have multiple spaces, select a space.

3. After you are logged in, enter the **`cf push XXX`** command, where **`XXX`** is the unique name of your application (in this example, it is `nlu-sentiment-analysis`).

   This step creates the Node.js application in Bluemix, uploading all your code.

   Upon success, you see results similar to Example 3-9.

Example 3-9 Success

```
Showing health and status for app nlu-sentiment-analysis in org <YOUR_ORG> /
space <YOUR_SPACE> as <YOUR_BLUEMIX_ID> ...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: nlu-sentiment-analysis.mybluemix.net
last uploaded: Sun May 14 05:15:16 UTC 2017
stack: cflinuxfs2
buildpack: SDK for Node.js(TM) (ibm-node.js-6.10.2,
buildpack-v3.12-20170505-0656)

     state     since                    cpu    memory    disk   details
#0   running   2017-05-14 02:16:42 AM   0.0%   59.5M of  1G     100.8M of 1G

Finished: SUCCESS
```

4. Navigate to your dashboard in Bluemix to see your newly created application. The application is now in a `Running` status, which is green (Figure 3-10).

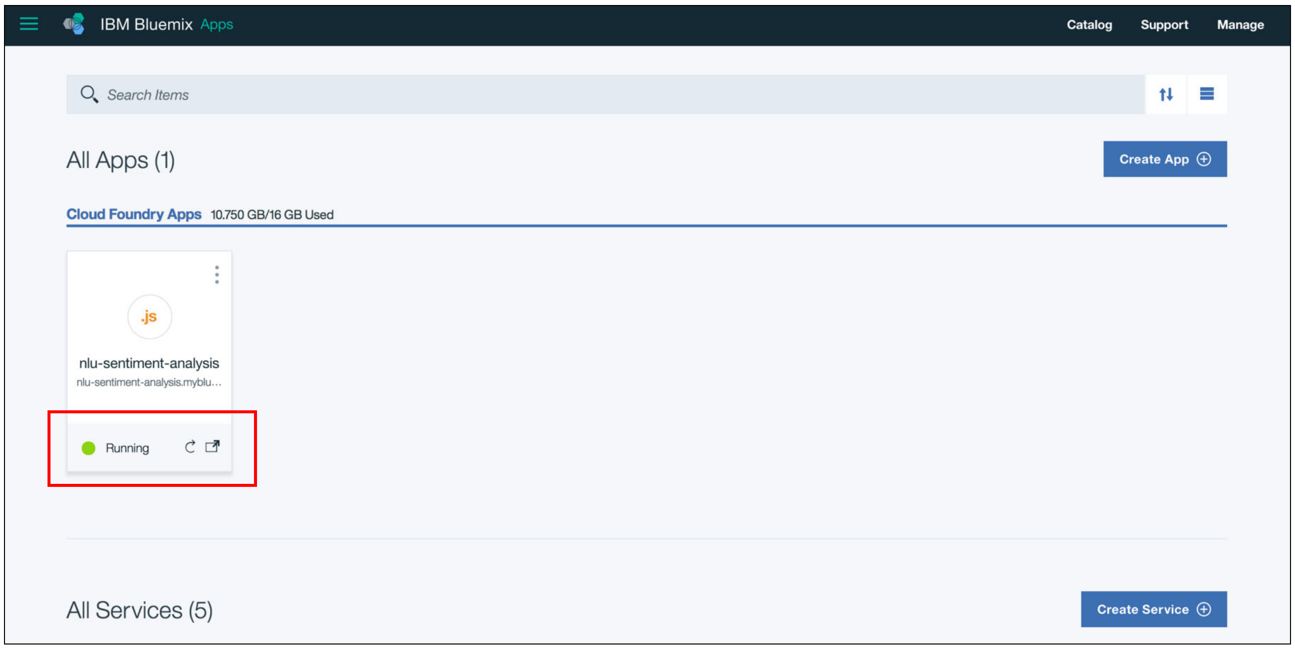


Figure 3-10 Dashboard with new application

3.4.6 Binding all the services to the Bluemix application (alternative option)

As you might have observed, up to now, the services and the application are somewhat independent. The application uses the services by targeting their credentials (URL, username, and password). This is correct and it is one way to connect applications and services.

Another approach is to bind the services to the application and let the application use the VCAP_SERVICES variable, as the following steps describe.

Note: In this use case, you entered the username and password for the services instances in the code. This section is optional for this example, but is included here to show the preferred approach to bind the services to the application.

Complete these steps:

1. From your dashboard, click the application you just created (Figure 3-11).

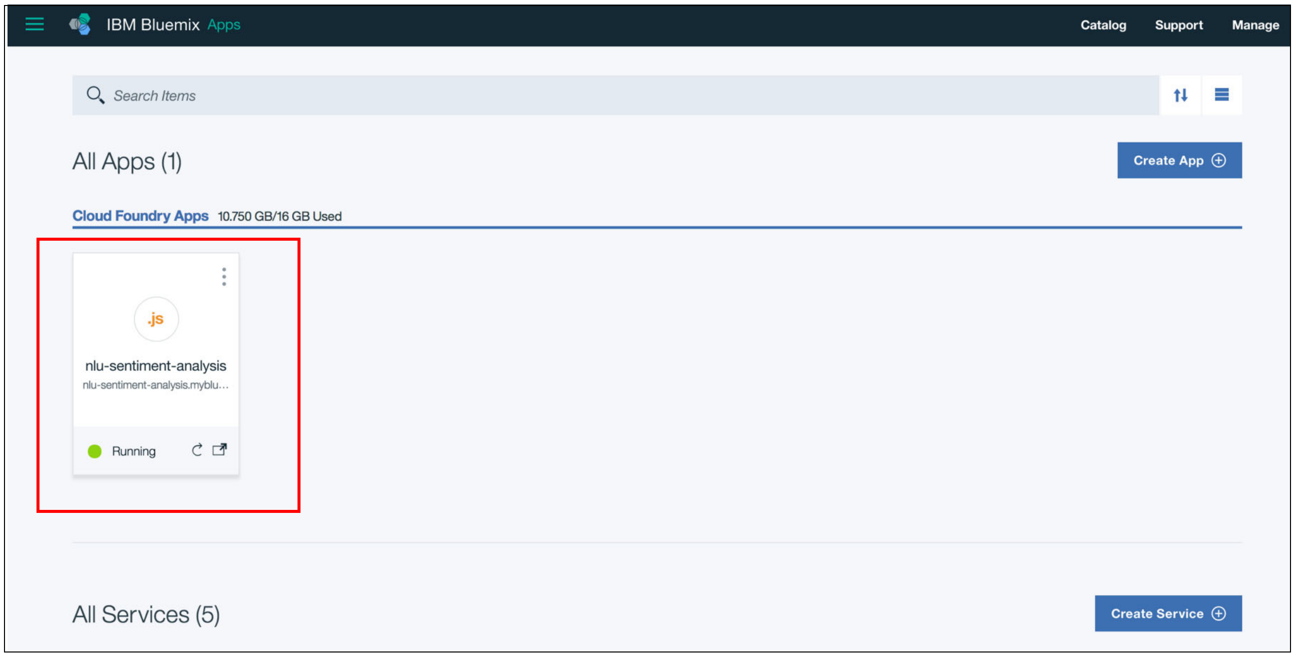


Figure 3-11 Click the application on the dashboard

2. Click **Connections** (Figure 3-12).

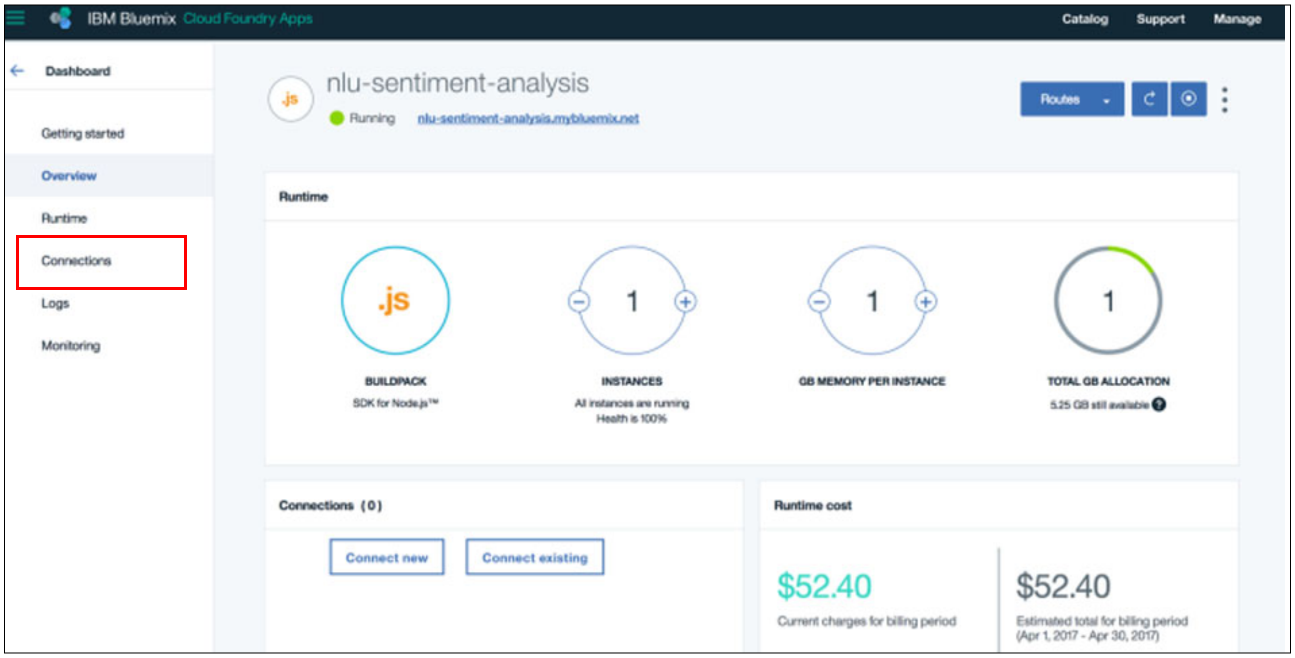


Figure 3-12 Click Connections

3. Click **Connect existing** (Figure 3-13).

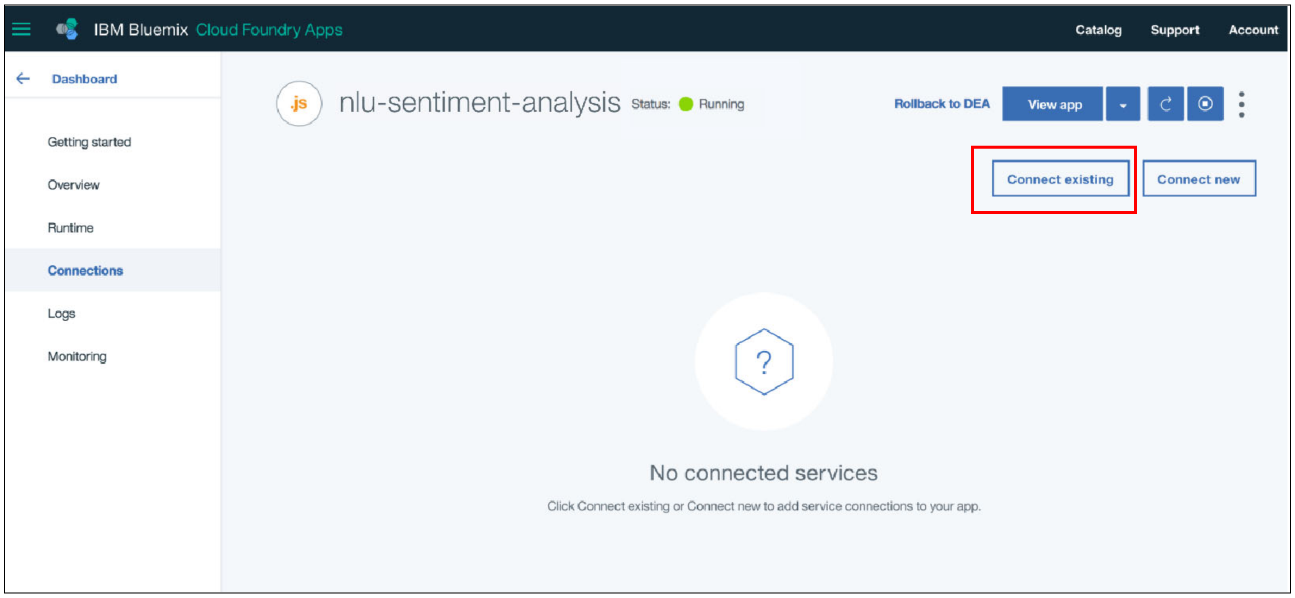


Figure 3-13 Connect existing services

4. Your existing services are listed (Figure 3-14). Select the **Natural Language Understanding** service and click **Connect**.

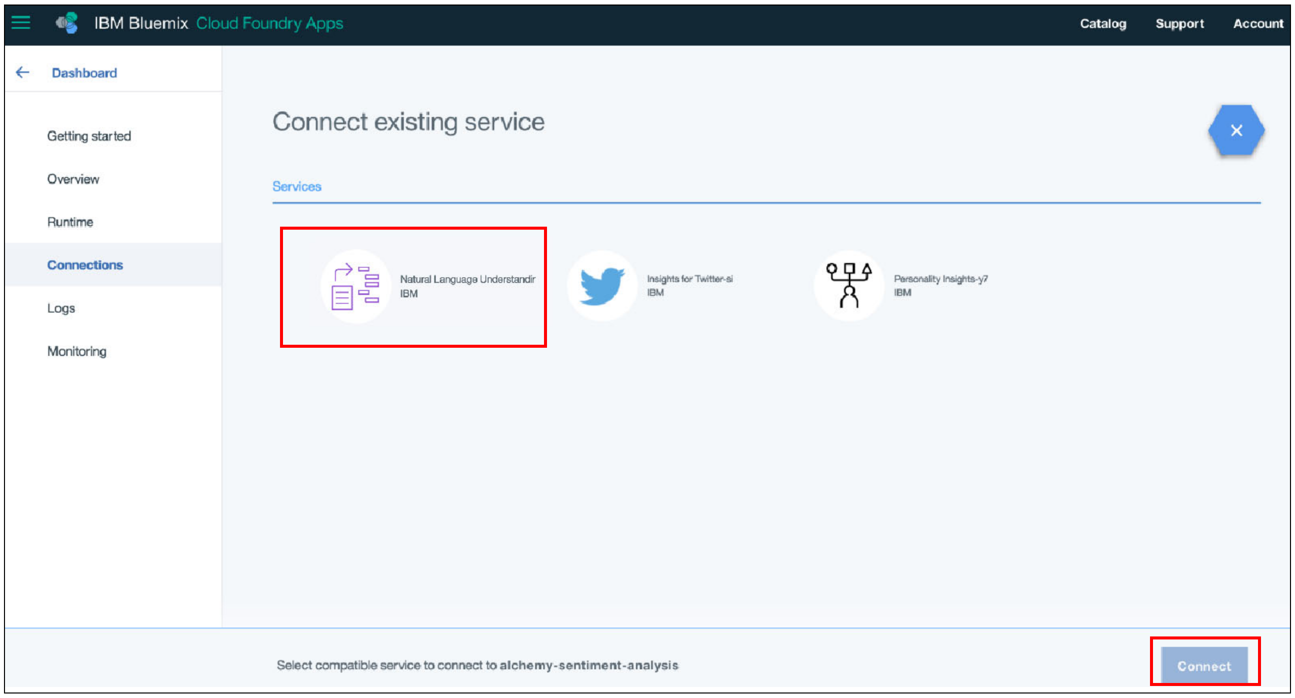


Figure 3-14 Select the service to connect

5. You must restage your application; at the prompt, click **Restage**.
6. Repeat step 3 through step 5 for both the Personality Insights service and for the Insights for Twitter service.

7. Click **Overview**. Your window will look similar to Figure 3-15.

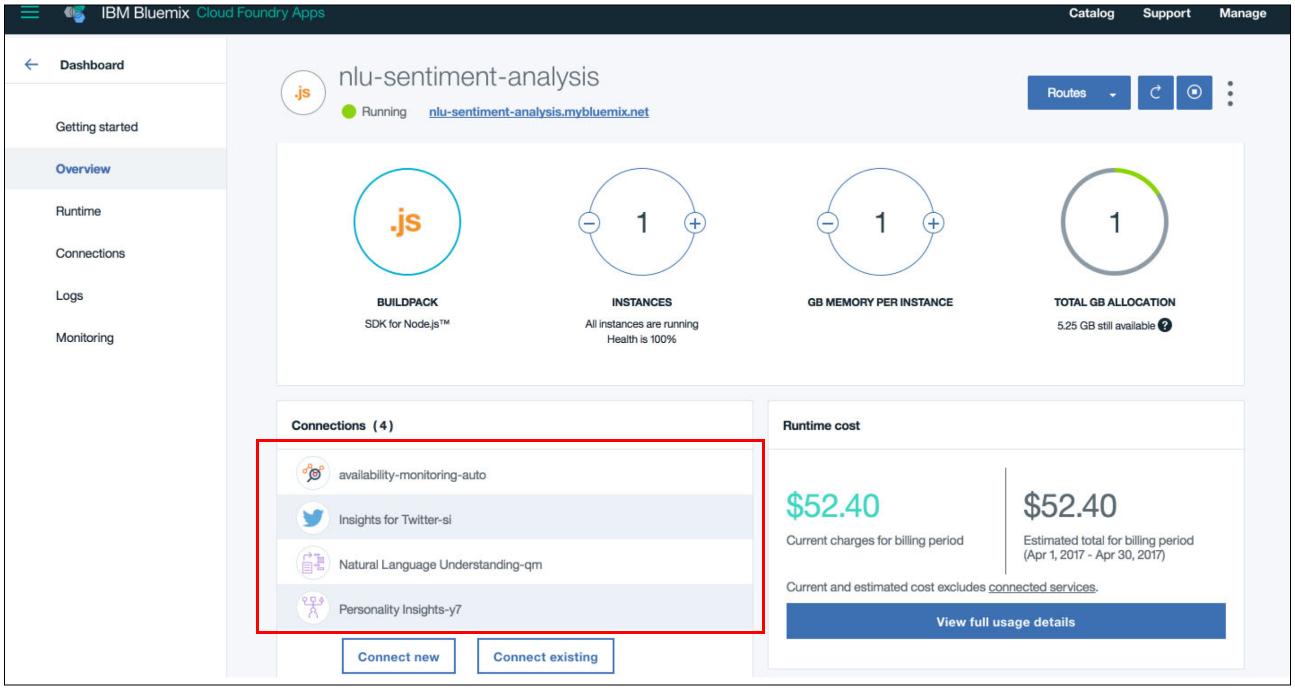


Figure 3-15 Application with services after binding

To take advantage of the bound services, the `app.js` code (lines 1 - 17) should be similar to Example 3-10. See the full `app.js` listing:

https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs/blob/master/app.js

Example 3-10 The app.js code

```
var NLU_USER = '46941c99-9f46-4fe0-980f-da4c5e492558'; // REPLACE WITH YOUR NLU USER
var NLU_PASSWORD = '1pyPqZAfdBpm'; // REPLACE WITH YOUR NLU PASSWORD
var TWITTER_INSIGHTS_USER = '1d519921-0506-48f9-b8b9-f5bec95f9b2b'; // REPLACE WITH YOUR
TWITTER_INSIGHTS USER
var TWITTER_INSIGHTS_PASSWORD = '04dKoismny'; // REPLACE WITH YOUR TWITTER_INSIGHTS PASSWORD

if (process.env.VCAP_SERVICES) {
    var services = JSON.parse(process.env.VCAP_SERVICES);
    for (var svcName in services) {
        if (svcName.match("natural-language-understanding")) {
            NLU_USER = services[svcName][0].credentials.username;
            NLU_PASSWORD = services[svcName][0].credentials.password;
        } else if (svcName.match("twitterinsights")) {
            TWITTER_INSIGHTS_USER = services[svcName][0].credentials.username;
            TWITTER_INSIGHTS_PASSWORD = services[svcName][0].credentials.password;
        }
    }
}

var express = require('express');
...................................
```

Likewise, the `personality.js` code (lines 1 - 22) should be similar to Example 3-11. See the full `personality.js` listing:

https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs/blob/master/helpers/personality.js

Example 3-11   The personality.js code

```
var PersonalityInsightsV3 = require('watson-developer-cloud/personality-insights/v3');

var PERSONALITY_USER;
var PERSONALITY_PASSWORD;

if (process.env.VCAP_SERVICES) {
    var services = JSON.parse(process.env.VCAP_SERVICES);
    for (var svcName in services) {
        if (svcName.match("personality_insights")) {
            PERSONALITY_USER = services[svcName][0].credentials.username;
            PERSONALITY_PASSWORD = services[svcName][0].credentials.password;
        }
    }
}

var personalityInsights = new PersonalityInsightsV3({
    username: PERSONALITY_USER,
    password: PERSONALITY_PASSWORD,
    version_date: '2016-10-19'
});

function toContentItem(tweet) {
...
```

Note: At this point, if you bound the services to the application as described in this section, you must push the application into Bluemix again. Follow the steps in 3.4.5, “Pushing the application into Bluemix” on page 79.

3.4.7 Testing the application

To test the application, run it as follows:

1. Click **Overview** and then click the application route, which is the link next to the `Running` status (Figure 3-16).

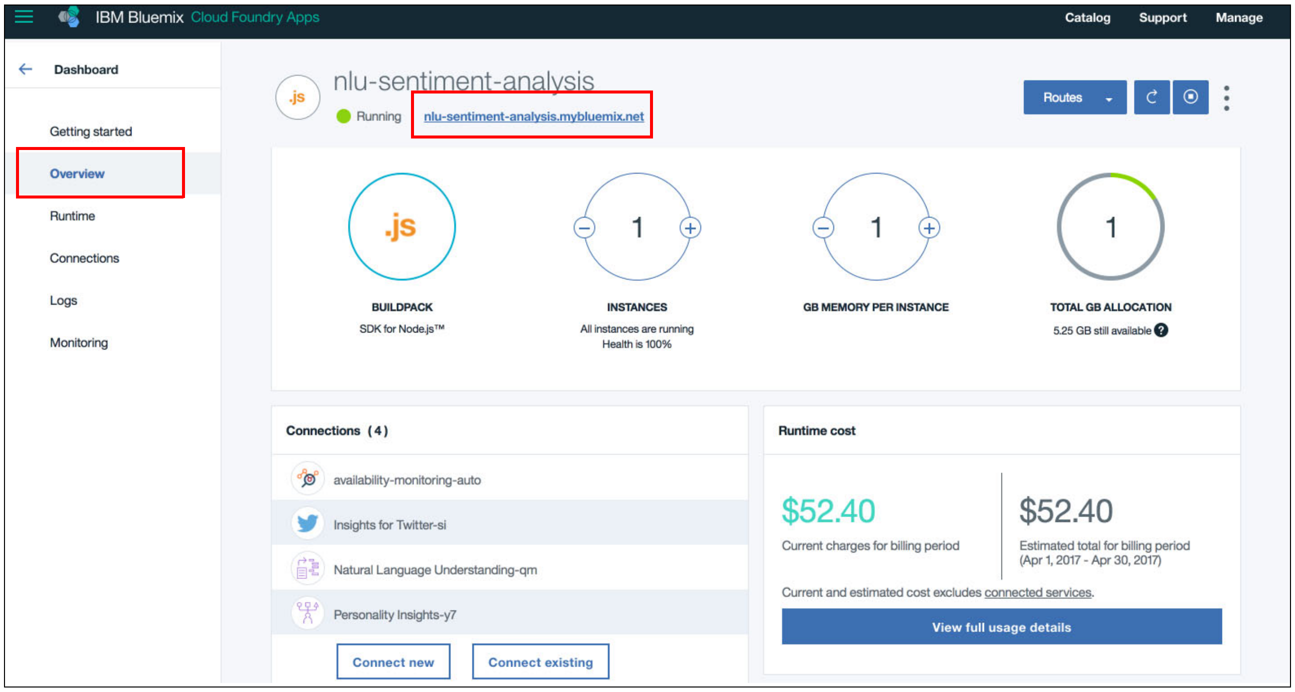
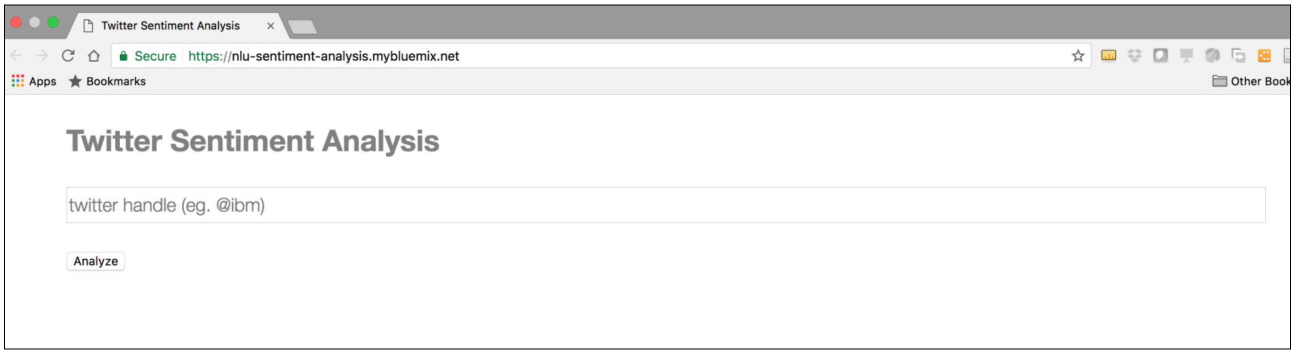


Figure 3-16 Overview tab

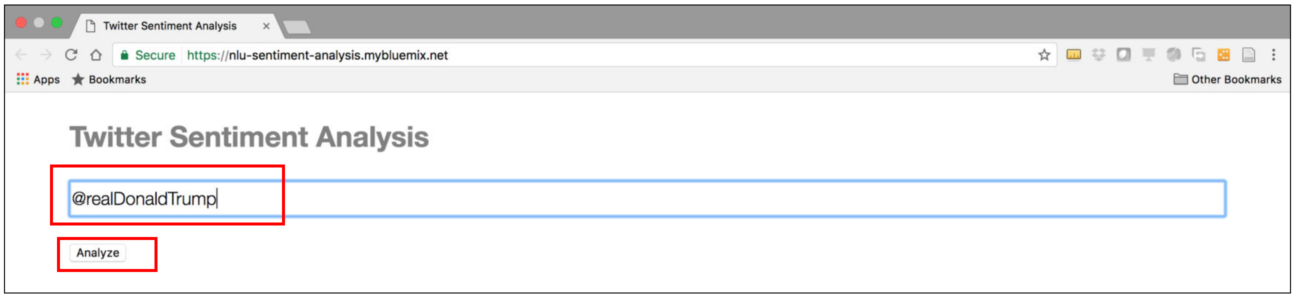
A new browser tab opens (Figure 3-17).



Twitter Sentiment Analysis
Secure https://nlu-sentiment-analysis.mybluemix.net
Apps
Bookmarks
Other Book
Twitter Sentiment Analysis
twitter handle (eg. @ibm)
Analyze

Figure 3-17 Your new application

2. Insert a Twitter handle in the input text field. This will be used to retrieve Tweets, and analyze sentiment and personality of the account owner (Figure 3-18). Click **Analyze**.


Twitter Sentiment Analysis
Secure https://nlu-sentiment-analysis.mybluemix.net
Apps
Bookmarks
Other Bookmarks

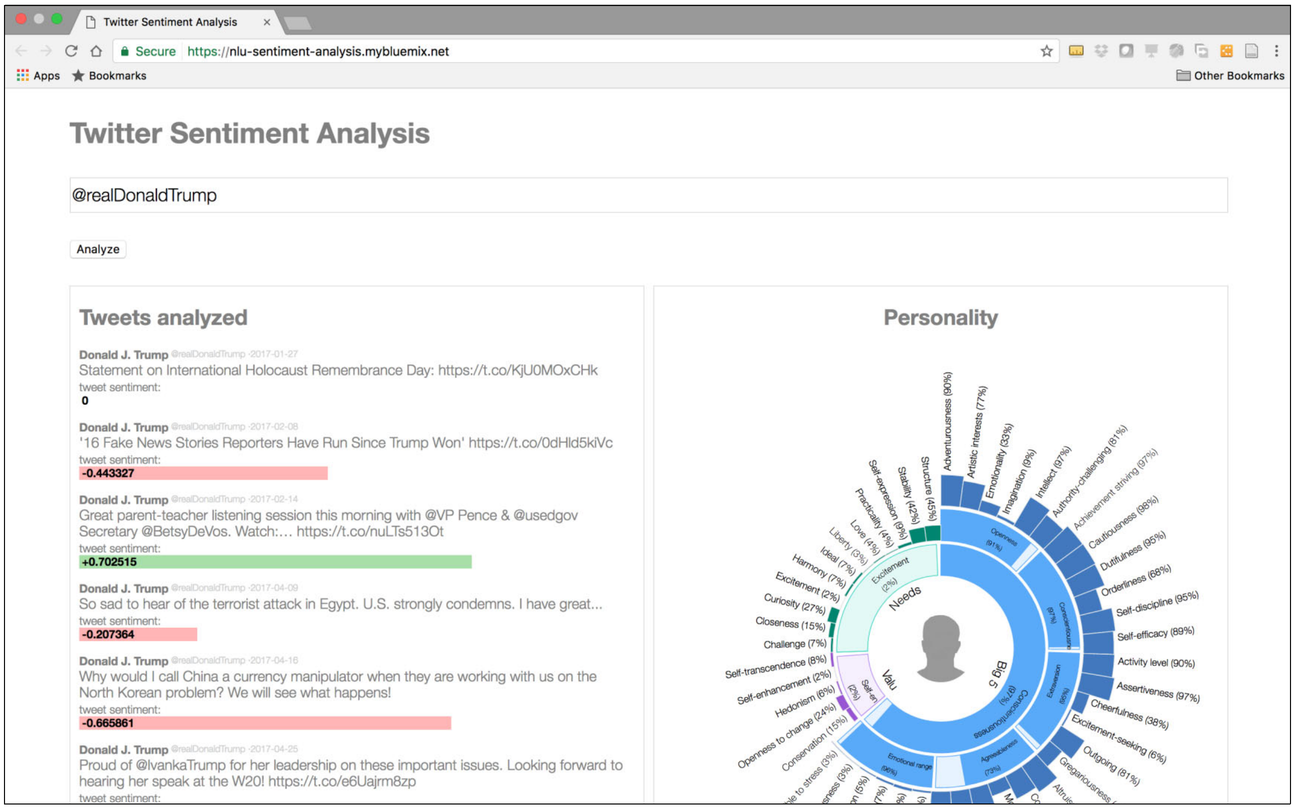
Twitter Sentiment Analysis

@realDonaldTrump

Analyze

Figure 3-18 Insert Twitter handle in input text field and click Analyze

3. Two new sections are displayed (Figure 3-19): one lists all the Tweets; the other shows a graphical depiction of the personality.



Twitter Sentiment Analysis

@realDonaldTrump

Analyze

Tweets analyzed

Donald J. Trump @realDonaldTrump ·2017-01-27
Statement on International Holocaust Remembrance Day: https://t.co/KjU0MOxCHk
tweet sentiment:
0

Donald J. Trump @realDonaldTrump ·2017-02-08
'16 Fake News Stories Reporters Have Run Since Trump Won' https://t.co/0dHld5kiVc
tweet sentiment:
-0.443327

Donald J. Trump @realDonaldTrump ·2017-02-14
Great parent-teacher listening session this morning with @VP Pence & @usedgov Secretary @BetsyDeVos. Watch:… https://t.co/nuLTs513Ot
tweet sentiment:
+0.702515

Donald J. Trump @realDonaldTrump ·2017-04-09
So sad to hear of the terrorist attack in Egypt. U.S. strongly condemns. I have great...
tweet sentiment:
-0.207364

Donald J. Trump @realDonaldTrump ·2017-04-16
Why would I call China a currency manipulator when they are working with us on the North Korean problem? We will see what happens!
tweet sentiment:
-0.665861

Donald J. Trump @realDonaldTrump ·2017-04-25
Proud of @IvankaTrump for her leadership on these important issues. Looking forward to hearing her speak at the W20! https://t.co/e6Uajrm8zp
tweet sentiment:

Figure 3-19 Analysis result

Examples of Twitter handle names you can use

The following examples are Twitter handle names that you can use to test the app:

- @realDonaldTrump
- @katyperry
- @TheEllenShow
- @Atleti
- @developerWorks
- @IBMBluemix
- @jimmyfallon

3.5 Quick deployment of application

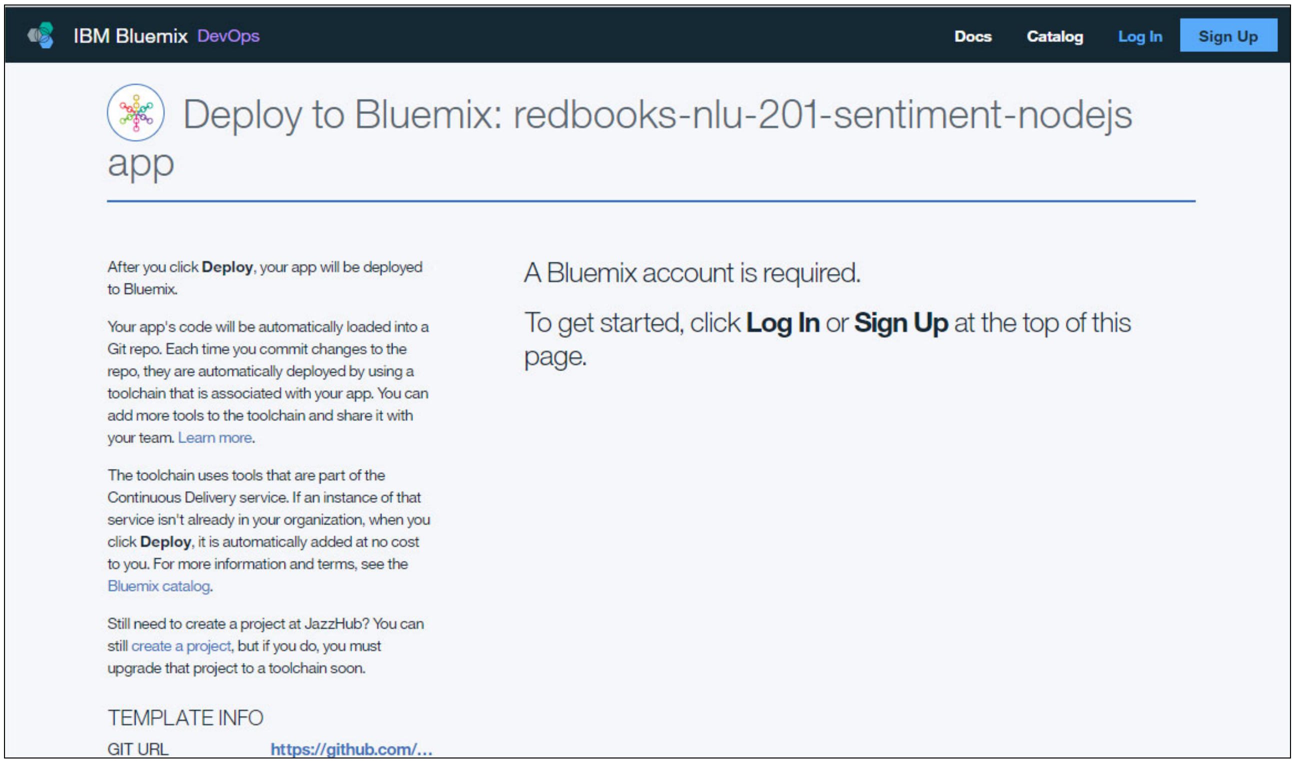
As described in 3.3, “Two ways to deploy the application: Step-by-step and quick deploy” on page 69, a Git repository containing the full application code is provided so that you can run the application with minimal steps.

To run the application more quickly, follow these steps:

1. Go to the following web page:

   `https://bluemix.net/deploy?repository=https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs`

2. The page shown in Figure 3-20 opens. Click **Log In**.



Deploy to Bluemix: redbooks-nlu-201-sentiment-nodejs app

After you click **Deploy**, your app will be deployed to Bluemix.

Your app's code will be automatically loaded into a Git repo. Each time you commit changes to the repo, they are automatically deployed by using a toolchain that is associated with your app. You can add more tools to the toolchain and share it with your team. Learn more.

The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your organization, when you click **Deploy**, it is automatically added at no cost to you. For more information and terms, see the Bluemix catalog.

Still need to create a project at JazzHub? You can still create a project, but if you do, you must upgrade that project to a toolchain soon.

TEMPLATE INFO

GIT URL https://github.com/...

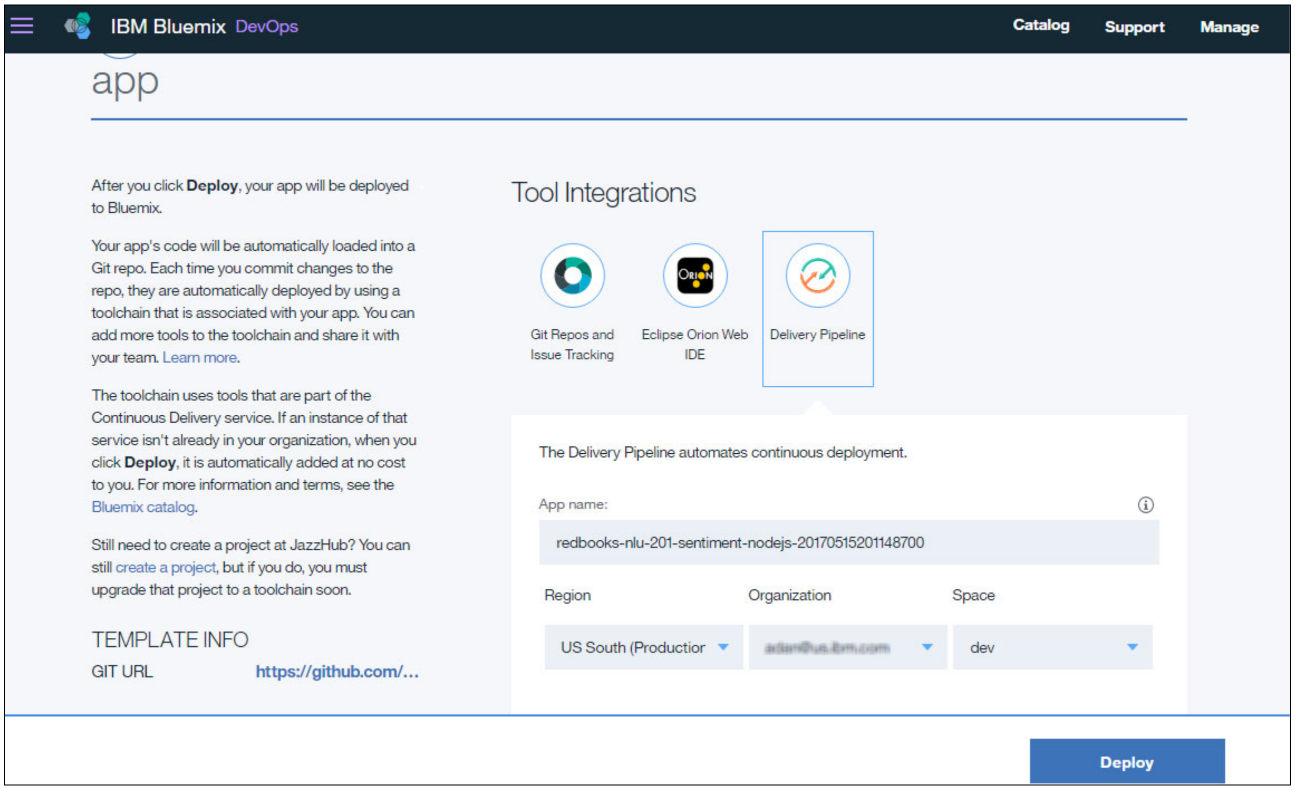
A Bluemix account is required.

To get started, click **Log In** or **Sign Up** at the top of this page.

Figure 3-20 Deploy the sample application to Bluemix: Log in

3. Enter your Bluemix ID and password.

4. On the next page (Figure 3-21), enter a unique name for your app and then click **Deploy**.



After you click **Deploy**, your app will be deployed to Bluemix.

Your app's code will be automatically loaded into a Git repo. Each time you commit changes to the repo, they are automatically deployed by using a toolchain that is associated with your app. You can add more tools to the toolchain and share it with your team. Learn more.

The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your organization, when you click **Deploy**, it is automatically added at no cost to you. For more information and terms, see the Bluemix catalog.

Still need to create a project at JazzHub? You can still create a project, but if you do, you must upgrade that project to a toolchain soon.

TEMPLATE INFO

GIT URL https://github.com/...

Tool Integrations

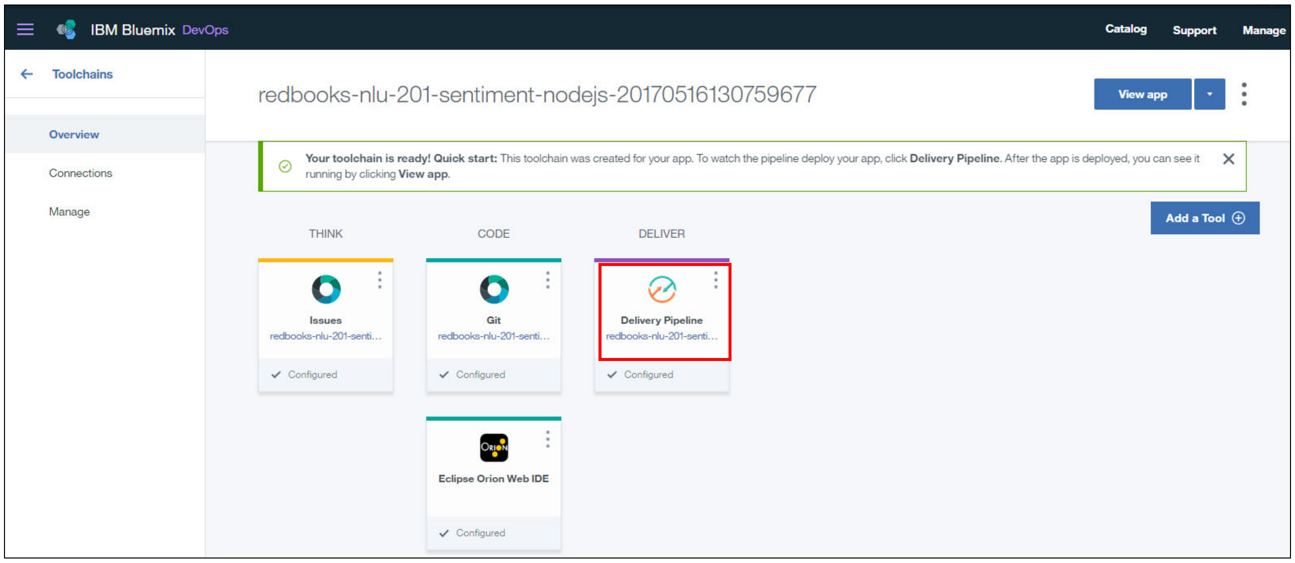
The Delivery Pipeline automates continuous deployment.

App name: redbooks-nlu-201-sentiment-nodejs-20170515201148700

Region	Organization	Space
US South (Productior	[redacted]	dev

Figure 3-21 Deploy the sample application to Bluemix: Deploy

5. Click **Delivery Pipeline** (Figure 3-22).

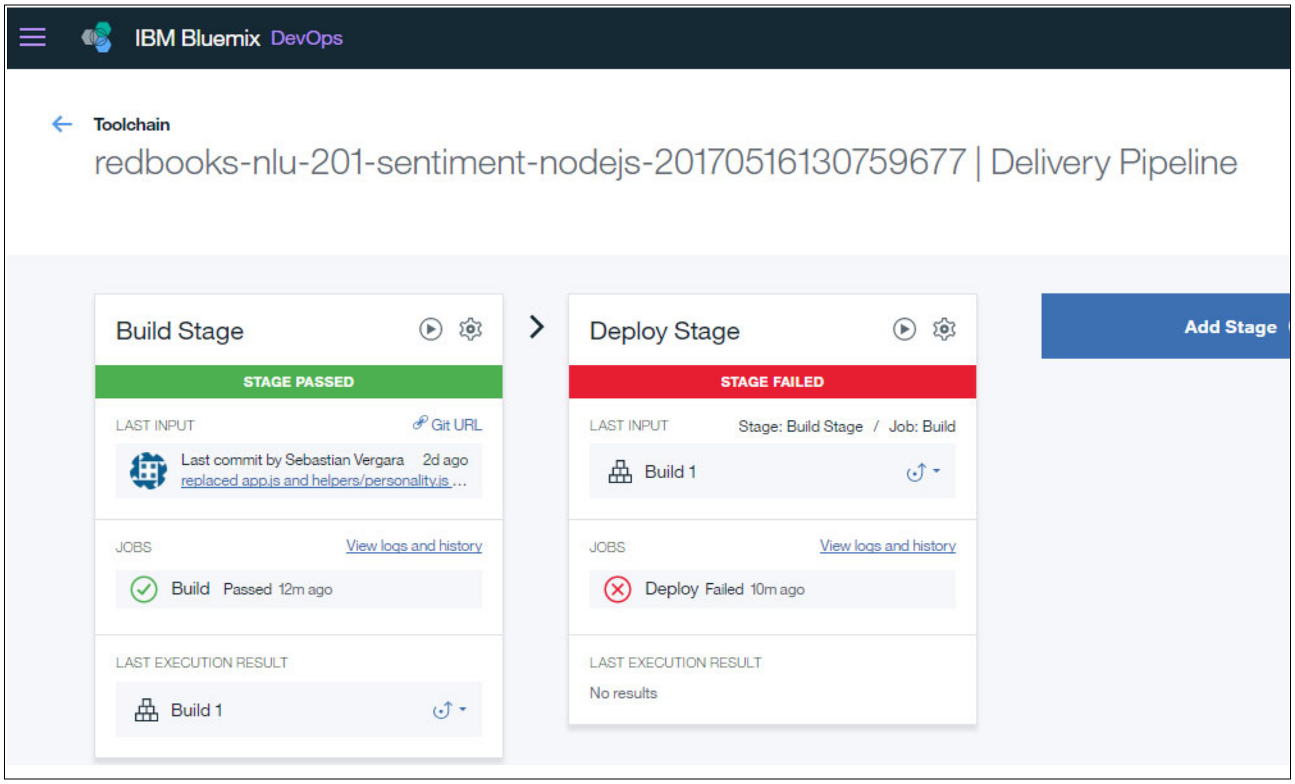


redbooks-nlu-201-sentiment-nodejs-20170516130759677

Your toolchain is ready! Quick start: This toolchain was created for your app. To watch the pipeline deploy your app, click **Delivery Pipeline**. After the app is deployed, you can see it running by clicking **View app**.

Figure 3-22 Deploy the sample application to Bluemix: Delivery Pipeline

6. Wait until the Deploy Stage completes. The deployment will fail (Figure 3-23) because it expects the Natural Language Understanding, Insights for Twitter, and Personality Insights services to be bound to the application.


IBM Bluemix DevOps

Toolchain

redbooks-nlu-201-sentiment-nodejs-20170516130759677 | Delivery Pipeline

Build Stage

STAGE PASSED

LAST INPUT Git URL

Last commit by Sebastian Vergara 2d ago
replaced app.js and helpers/personality.js ...

JOBS View logs and history

Build Passed 12m ago

LAST EXECUTION RESULT

Build 1

Deploy Stage

STAGE FAILED

LAST INPUT Stage: Build Stage / Job: Build

Build 1

JOBS View logs and history

Deploy Failed 10m ago

LAST EXECUTION RESULT

No results

Figure 3-23 Deploy the sample application to Bluemix: Deploy Stage Failed

7. Create those three services:
   a. Create the NLU service as described in 3.4.1, “Creating the Natural Language Understanding service instance” on page 69.
   b. Create the Insights for Twitter service as described in 3.4.2, “Creating the Insights for Twitter service instance” on page 70.
   c. Create the Personality Insights service as described in 3.4.3, “Creating the Personality Insights service instance” on page 73.
8. Bind the three services to the application as described in 3.4.6, “Binding all the services to the Bluemix application (alternative option)” on page 80.

Note: Do not push the application to Bluemix after binding the services.

9. After the three services are bound and the application is restaged, see that the application starts and is running (Figure 3-24).

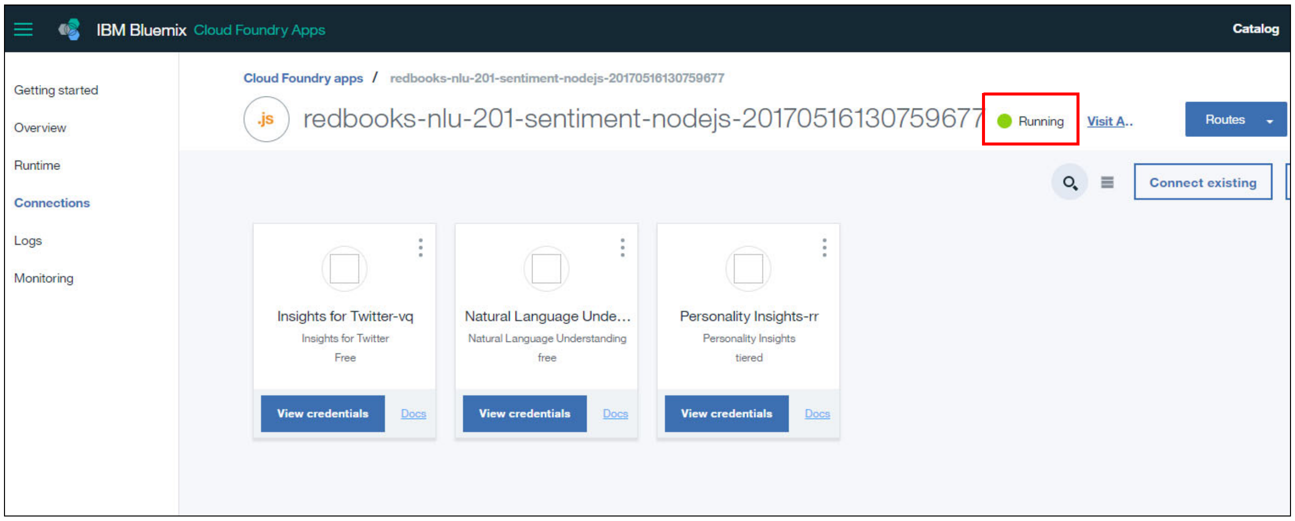


Figure 3-24 Application connected to the three services and status Running

10. Click **Overview**.

11. Click **Routes** and select the link to access the application (Figure 3-25).

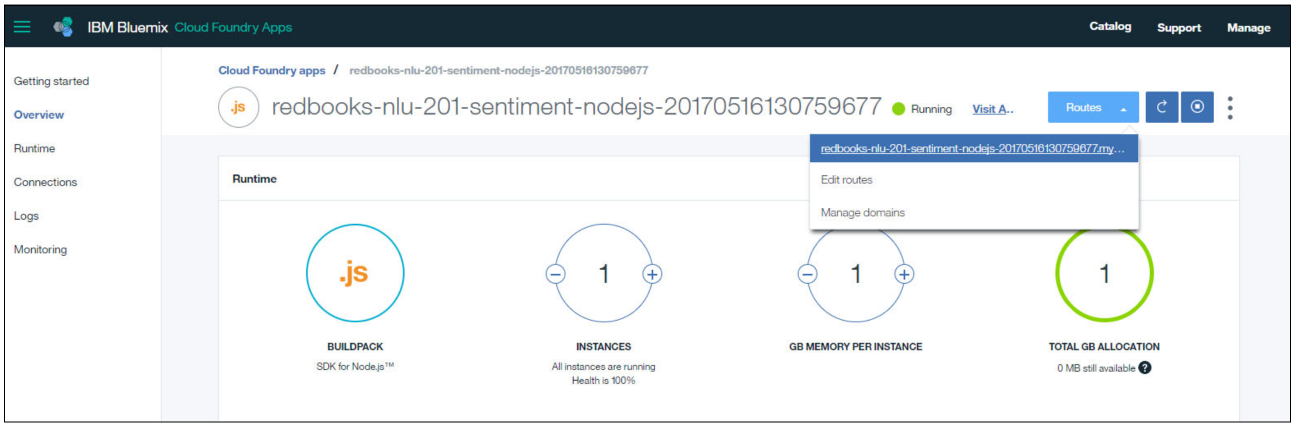


Figure 3-25 Opening the application

12. Test the application as described in 3.4.7, “Testing the application” on page 85.

3.6 References

See the following resources:

- Overview of the IBM Watson Natural Language Understanding service:

  https://www.ibm.com/watson/developercloud/doc/natural-language-understanding/index.html
- Personality Insights:

  https://www.ibm.com/watson/developercloud/personality-insights.html
- About Insights for Twitter:

  https://console.ng.bluemix.net/docs/services/Twitter/twitter_overview.html#about_twitter
- SIgn up for Bluemix account:

  https://console.ng.bluemix.net/registration/
- personality-sunburst-chart:

  https://www.npmjs.com/package/personality-sunburst-chart

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The following Git repositories are available to help you with the examples in these chapters:

- Chapter 1, “Basics of Watson Natural Language Understanding service” on page 1:
  - Node.js sample snippets:

    `https://github.com/snippet-java/redbooks-101/tree/master/nodejs`
  - Java sample snippets:

    `https://github.com/snippet-java/redbooks-101/tree/master/java`
- Chapter 3, “Sentiment and personality analysis” on page 65:
  - For the *incomplete* code (step-by-step implementation version):

    `https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs-student`
  - For the *complete* code (quick deployment version) that you can use for verification or as a code reference:

    `https://github.com/snippet-java/redbooks-nlu-201-sentiment-nodejs`

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

The volumes in the *Building Cognitive Applications with IBM Watson APIs* series:

- *Volume 1 Getting Started*, SG24-8387
- *Volume 2 Conversation*, SG24-8394
- *Volume 3 Visual Recognition*, SG24-8393
- *Volume 4 Natural Language Classifier*, SG24-8391
- *Volume 5 Language Translator*, SG24-8392
- *Volume 6 Speech to Text and Text to Speech*, SG24-8388
- *Volume 7 Natural Language Understanding*, SG24-8398

You can search for, view, download or order these documents and other IBM Redbooks, IBM Redpapers™, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM Watson Knowledge Studio:

  https://www.ibm.com/us-en/marketplace/supervised-machine-learning

- Supported languages:

  https://www.ibm.com/watson/developercloud/doc/natural-language-understanding/#supported-languages

- IBM Bluemix:

  https://console.ng.bluemix.net/

- Cloud Foundry software:

  https://github.com/cloudfoundry/cli/releasesDescription1

- Git downloads:

  https://git-scm.com/downloads

- Cloud Foundry download and installation:

  https://github.com/cloudfoundry/cli/releasesDescription2

- REST API documentation in the Using the Insights for Twitter REST APIs:

  https://console.ng.bluemix.net/docs/services/Twitter/twitter_rest_apis.html#rest_api

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Building Cognitive Applications with IBM Watson Services: Volume 7 Natural Language Understanding

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8398-00

ISBN 0738442623