Question Answer System for Twitter

Project Report 1

Submission Date:6/24/2016

by

Ganseh Taduri(#40)

Swatvik Gunamaneni(#10)

Venkata Sasidhar kanumuri(#13)

Tejo Kumar Manchu(#19)

Motivation: There are many advantages in extracting the information from the slew of sources available on the internet. Not a decade ago, we realized that we can process this large amount of data and extract meaningful information from it. There is a growing importance on the development of information technologies that are capable of accessing and analyzing the data available via social media. Twitter is one such source of large amount of raw data is which contains worldwide news, public opinion on several events which can be processed and relevant information can be extracted. On average, every second around 6000 tweets are tweeted on twitter. This data can be used to a build a question answer system.

Objectives:

The objective of the project is to learn machine learning algorithms, natural language processing through implementation of available services and APIs like Natural language classifier, Speech to text, speech to text etc. We plan to tap the twitter data and develop a question answering system which takes a text question as input and answers it in accordance with the data available on twitter.

Sample Input and output: Our system would understand the input question related to any topic/event in twitter and would answer the question.

Sample input:

Who do you think will win US elections 2016?

Sample Output:

Donald Trump: 50% confidence

Hillary Clinton: 50 % confidence

Domain: Question answering system

Class diagram:

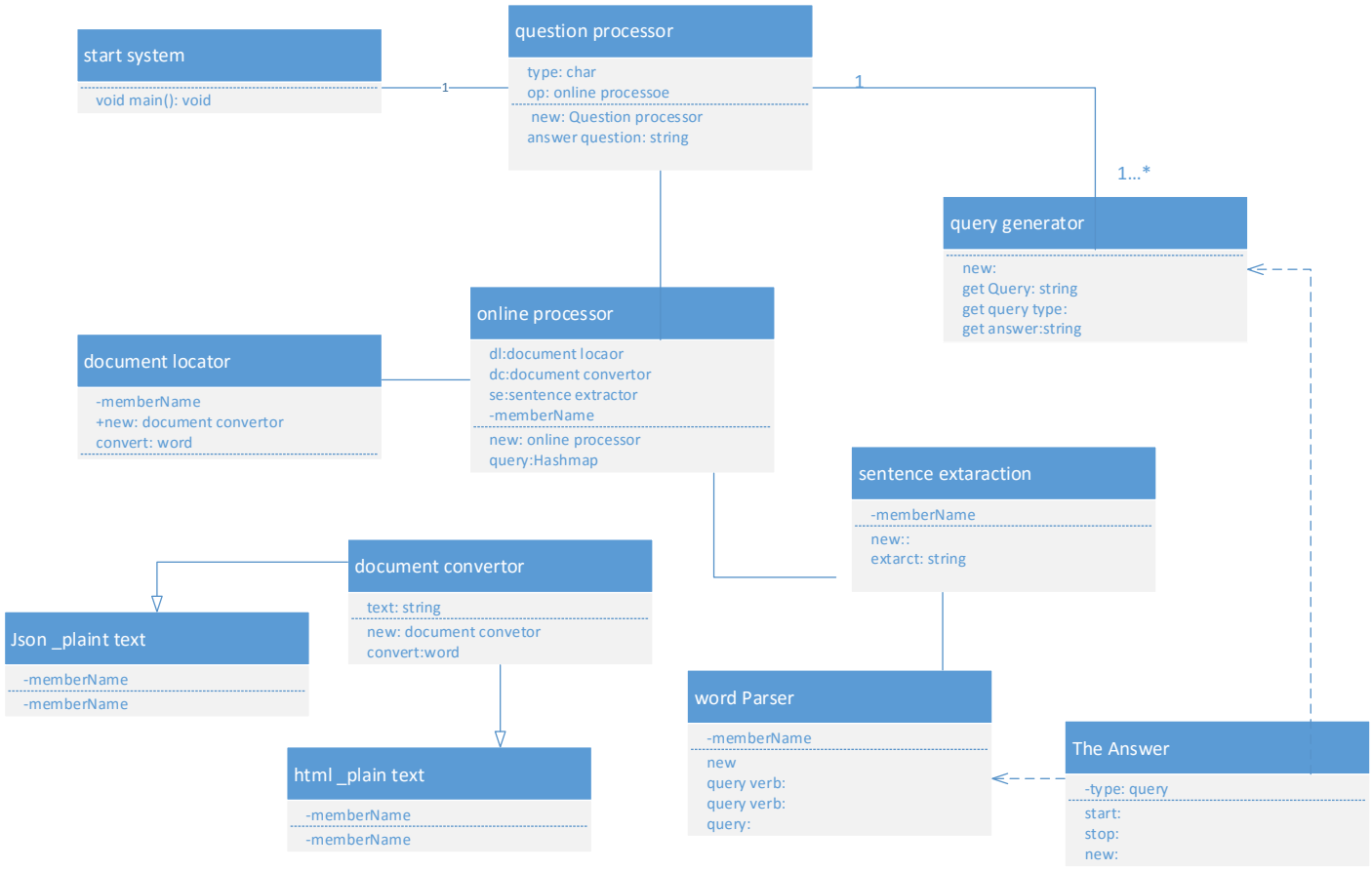
Class diagrams shows the classes in the system and their interrelationships between them

Classes involved in this system are

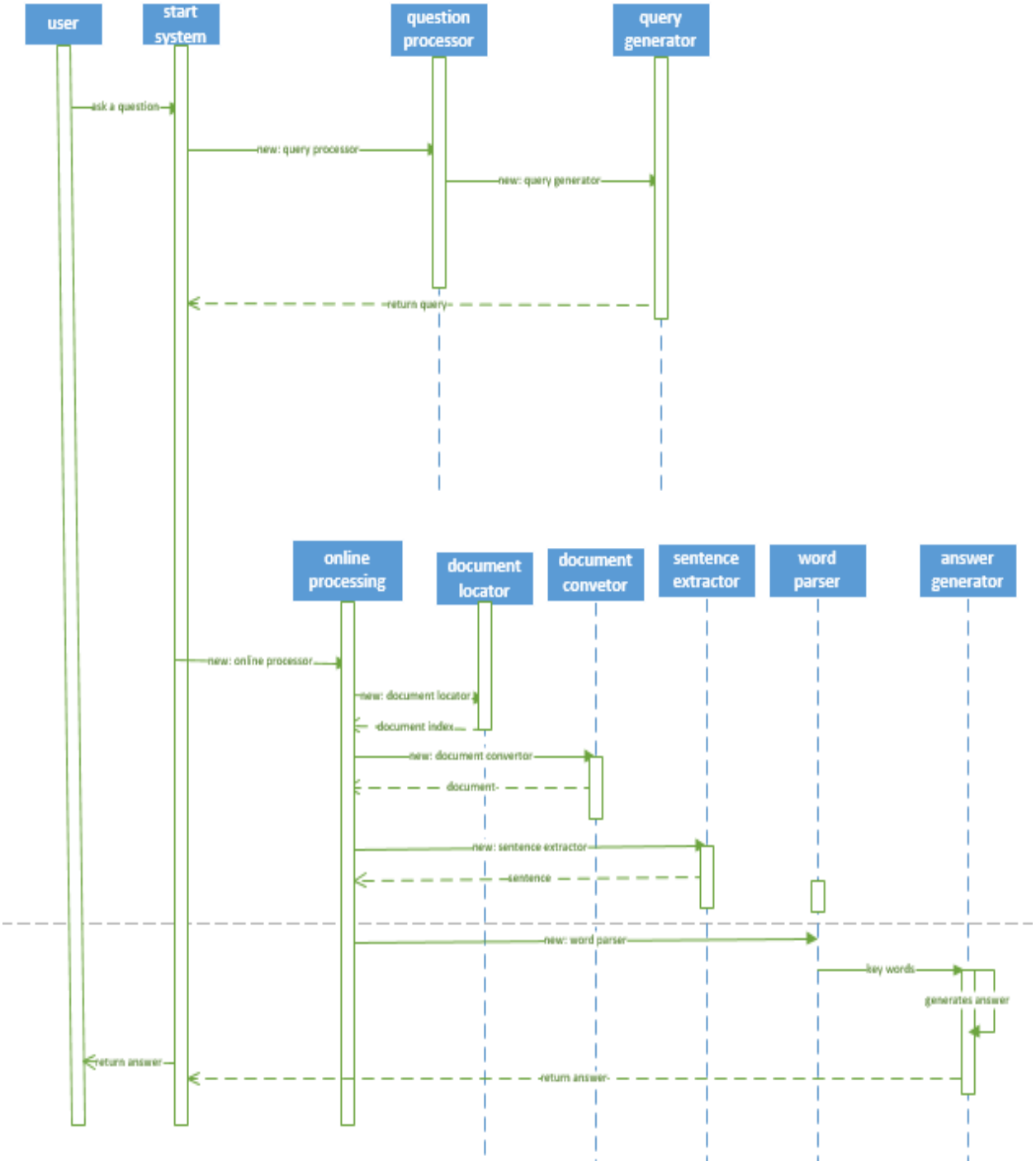
- Start system
- Online processor
- Query generator
- Document locator
- Question processor
- Document convertor
- Word parser
- Json_plain text
- Html_plain text
- Answer generator

We have the following relations for classes

- Dependency relation shows class is dependent on or uses other class
- Generalization which shows the inheritance of classes like which class is a subset of or what is the superclass of this subclass
- Generalization is placed from document convertor to Json_plain text and html_plain text where document convertor is the superclass of both classes
- Dependency is placed between word parser and sentence extraction as to parse the data we need text split into sentences and we get this from sentence extraction
- Other classes are linked with association relation that shows the connection of two classes



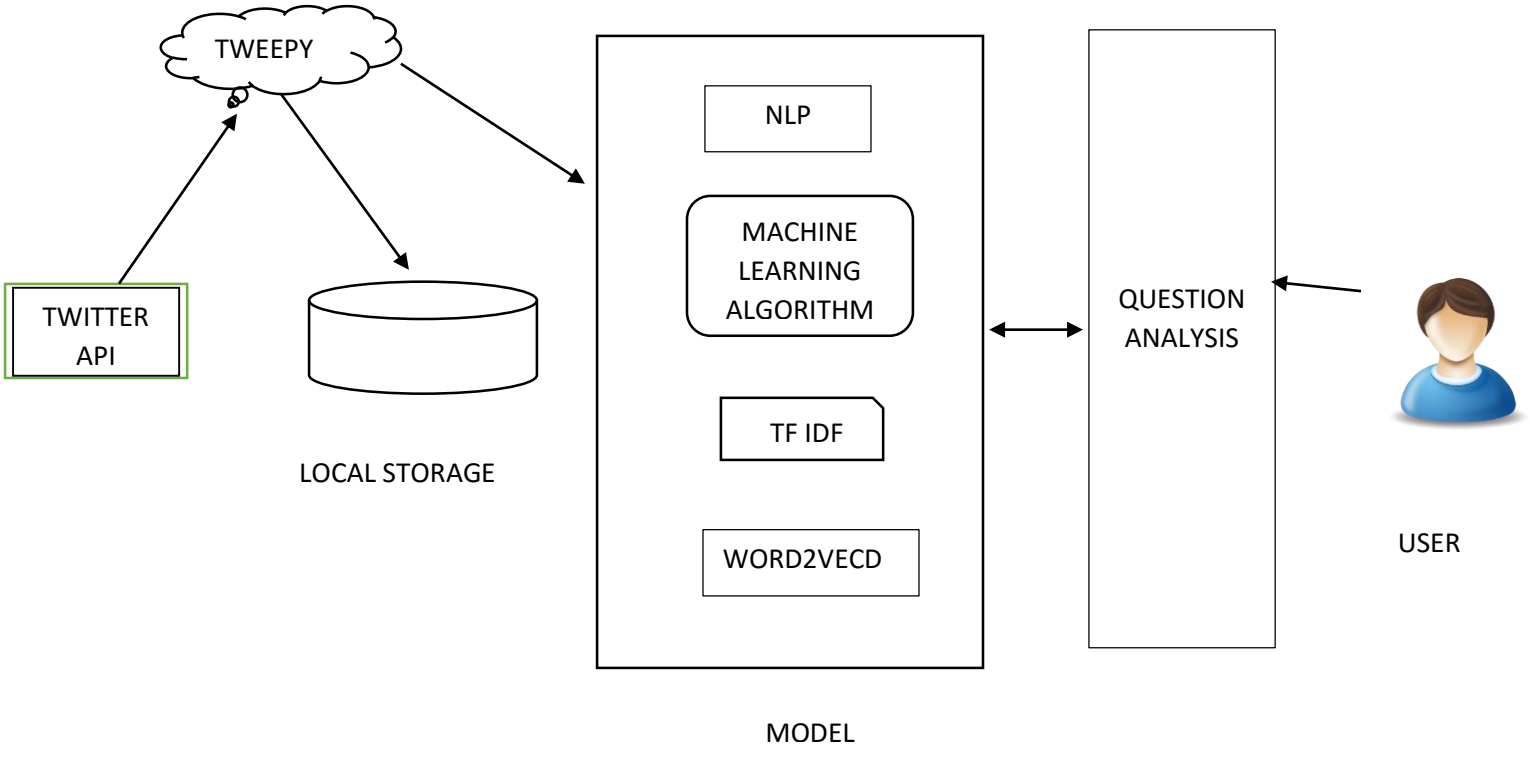
Sequence diagram: Sequence diagram gives the information about the interactions between the objects. The communication between the objects is done by passing messages from one object to another.

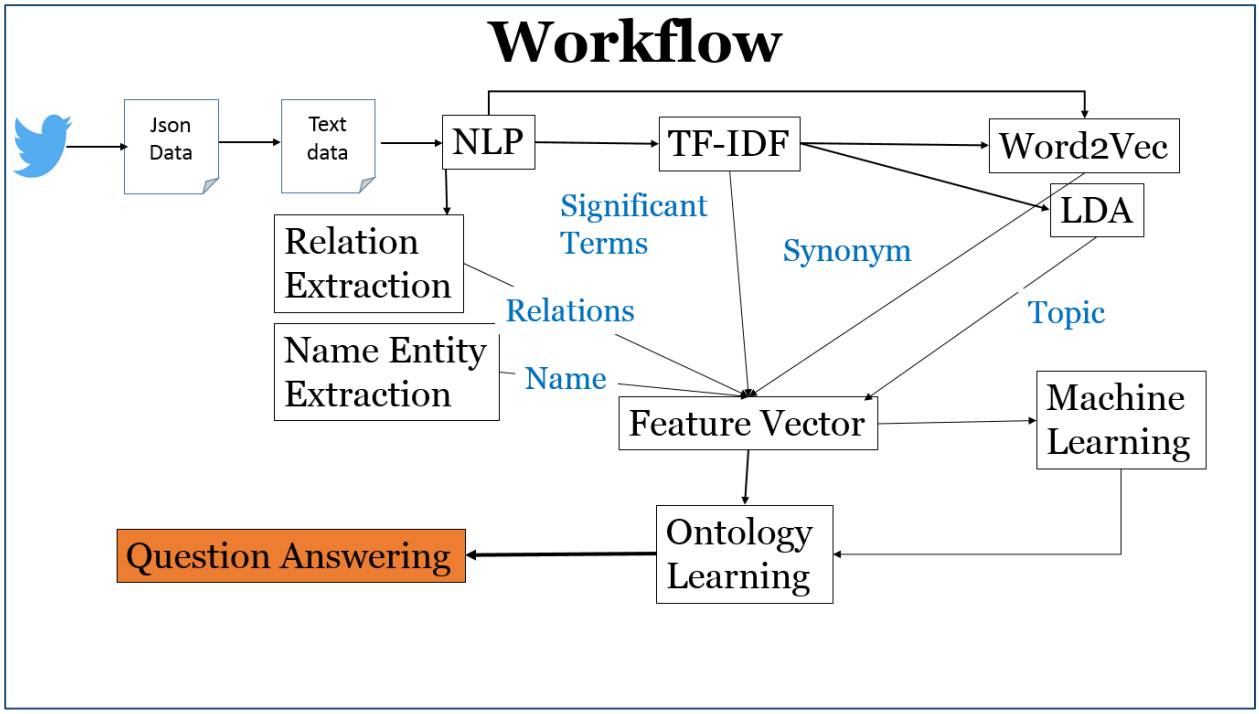


The objects involved in this model are:

- User: who interacts with the system by giving a question
- Start system: this takes the input from the user and interacts with our model
- Question processor: processes the question and classifies the question based on what, when, and other classifications
- Query generator: generates a machine readable query of the question from the user
- Online processing: streaming for dynamic data that comes from the twitter API
- Document locater: locates the document like this gives the index of the document in the local storage
- Document convertor: converts the html file or Json file
- Sentence extractor: gives the sentences from the paragraphs in the document
- Word parser: generates the tokens from the sentences
- Answer generator: generates the answers based on machine learning algorithms provided

System Architecture





Services used:

Natural language Classifier (NLC): this service uses the deep learning techniques and by using these techniques we make predictions about the predefined classes for sentences that we give as input for this service. This makes use of some machine learning algorithms to make predictions

Example of his service could be IBM Watson

MongoDB: Mongo DB is an open source database which has the convenience for the developers in terms of scalability and ease of use. Data is stored in the form of a Json file in mongo DB

Alchemy API: we make use of alchemy API like named entity extraction and other keyword extraction

- This twitter API forms as an interface for the developer to use data coming from twitter, here data is in the form of tweets. Data is collected in Json format.
- We will also use speech to text API to convert the speech that we give as input from the device into text format

Natural classifier demo: We plan to use Natural language classifier to determine the question type. In this section, the demo of the same classifier is shown. In this demo we train the classifier with twenty-five male names and twenty-five female names. Then we give a test name which the classifier takes as input and gives the percentage confidence of male and female classification.

Screen shot of giving training data:

```
C:\>curl -i -u "f6de402c-0f6e-4c26-bbdd-b5e314767f72":"agBMTwd7DPnF" \ -F training_data=@C:/Users/tmcx4/Desktop/names.csv
//gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers"
curl: (6) Could not resolve host: \
curl: (6) Could not resolve host: \
curl: (6) Could not resolve host: \
HTTP/1.1 100 Continue
X-Note: Gateway Ack

HTTP/1.1 200 OK
X-Backside-Transport: OK OK
Connection: Keep-Alive
Transfer-Encoding: chunked
Date: Sat, 25 Jun 2016 00:06:41 GMT
Content-Type: application/json
```

Classifier response:

```
{
  "classifier_id" : "2373f5x67-nlc-7103",
  "name" : "TutorialClassifier",
  "language" : "en",
  "created" : "2016-06-25T00:06:40.763Z",
  "url" : "https://gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers/2373f5x67-nlc-7103",
  "status" : "Training",
  "status_description" : "The classifier instance is in its training phase, not yet ready to accept classify requests"
}
C:\>
```

Classifier Status:

```
C:\>
C:\>curl -u "f6de402c-0f6e-4c26-bbdd-b5e314767f72":"agBMTwd7DPnF" \ "https://gateway.watsonplatform.net/natural-lang
curl: (6) Could not resolve host: \
{
  "classifier_id" : "2373f5x67-nlc-7103",
  "name" : "TutorialClassifier",
  "language" : "en",
  "created" : "2016-06-25T00:06:40.763Z",
  "url" : "https://gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers/2373f5x67-nlc-7103",
  "status" : "Available",
  "status_description" : "The classifier instance is now available and is ready to take classifier requests."
}
C:\>
```

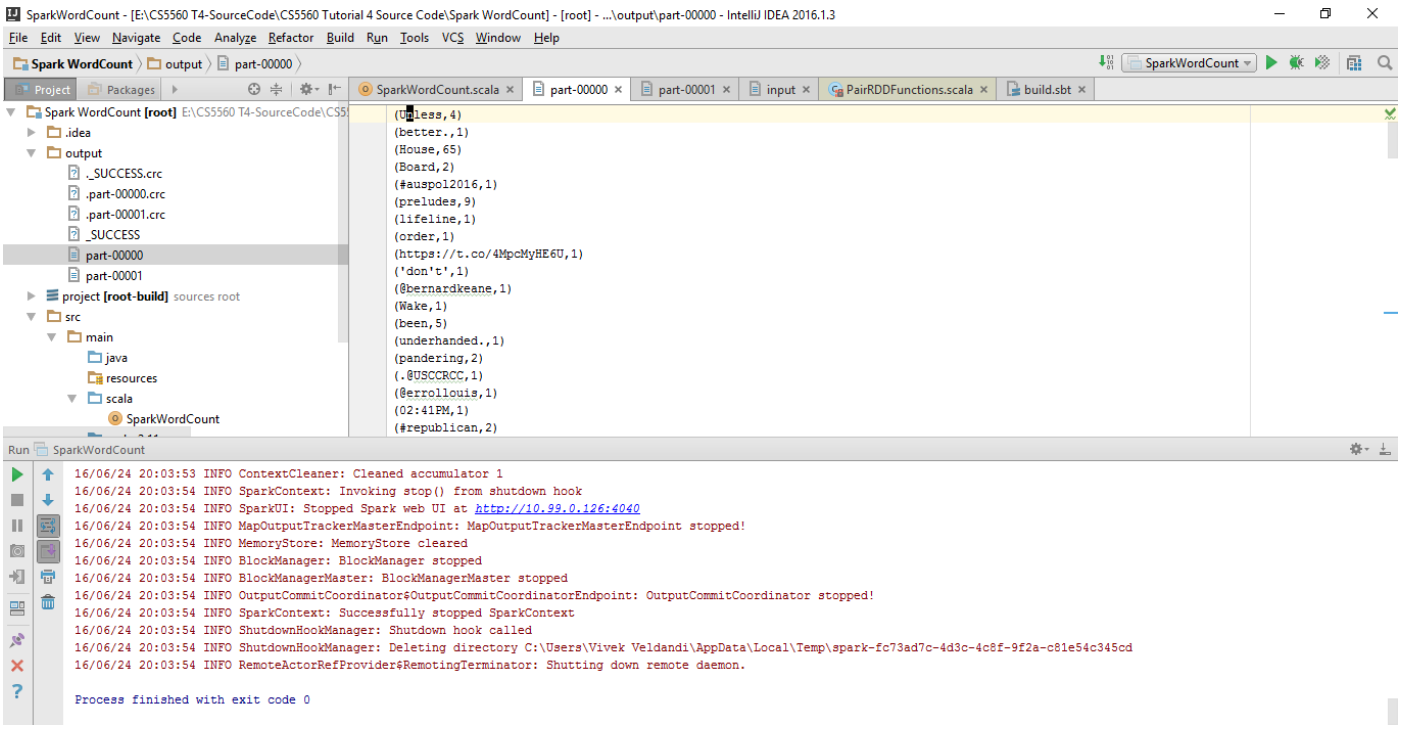
Classifier Output for test data: The name “Suresh” is classified as male with 85% confidence.

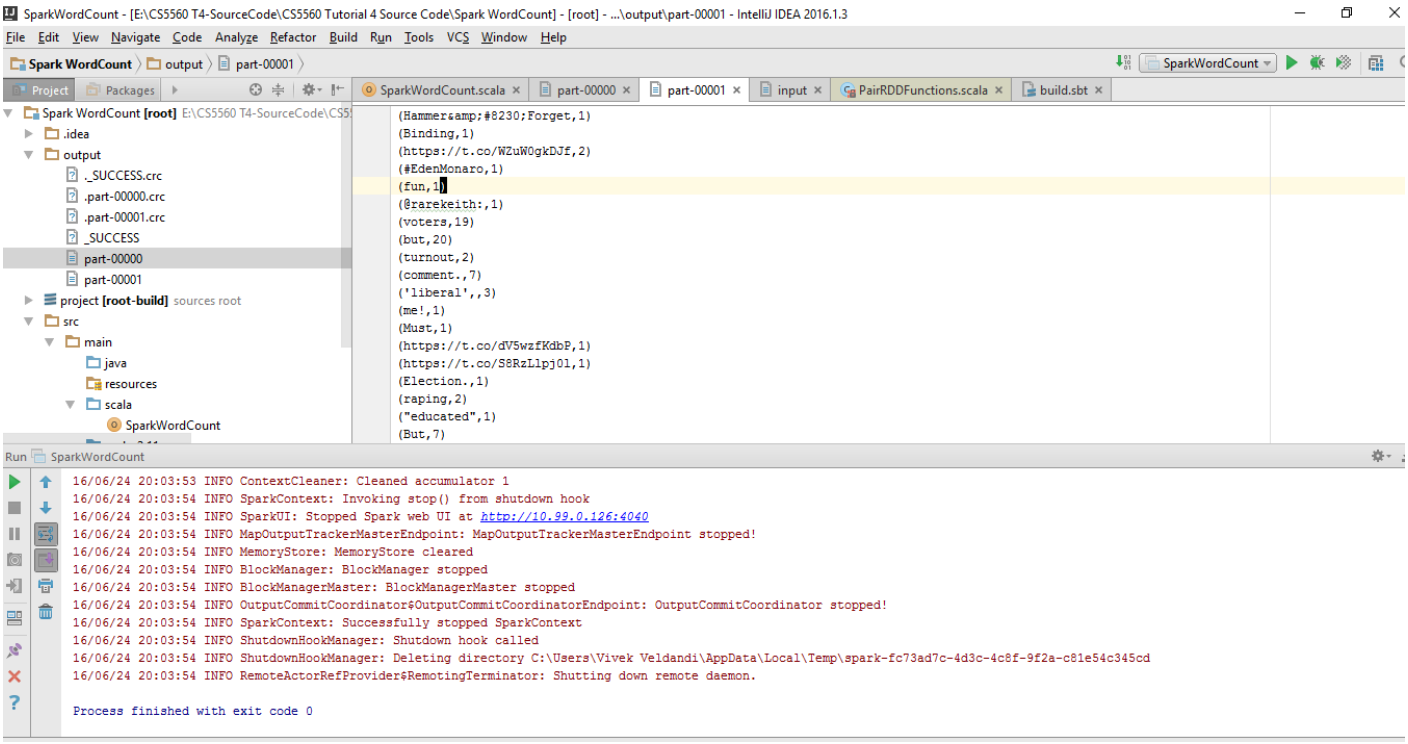
```
C:\>curl -G -u "f6de402c-0f6e-4c26-bbdd-b5e314767f72":"agBMTwd7DPnF" \ "https://gateway.watsonplatform.net/natural
resh"
curl: (6) Could not resolve host: \
{
  "classifier_id" : "2373f5x67-nlc-7103",
  "url" : "https://gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers/2373f5x67-nlc-7103",
  "text" : "suresh",
  "top_class" : "male",
  "classes" : [ {
    "class_name" : "male",
    "confidence" : 0.8489180884346
  }, {
    "class_name" : "female",
    "confidence" : 0.15108191156540002
  } ]
}curl: (6) Could not resolve host: \
```

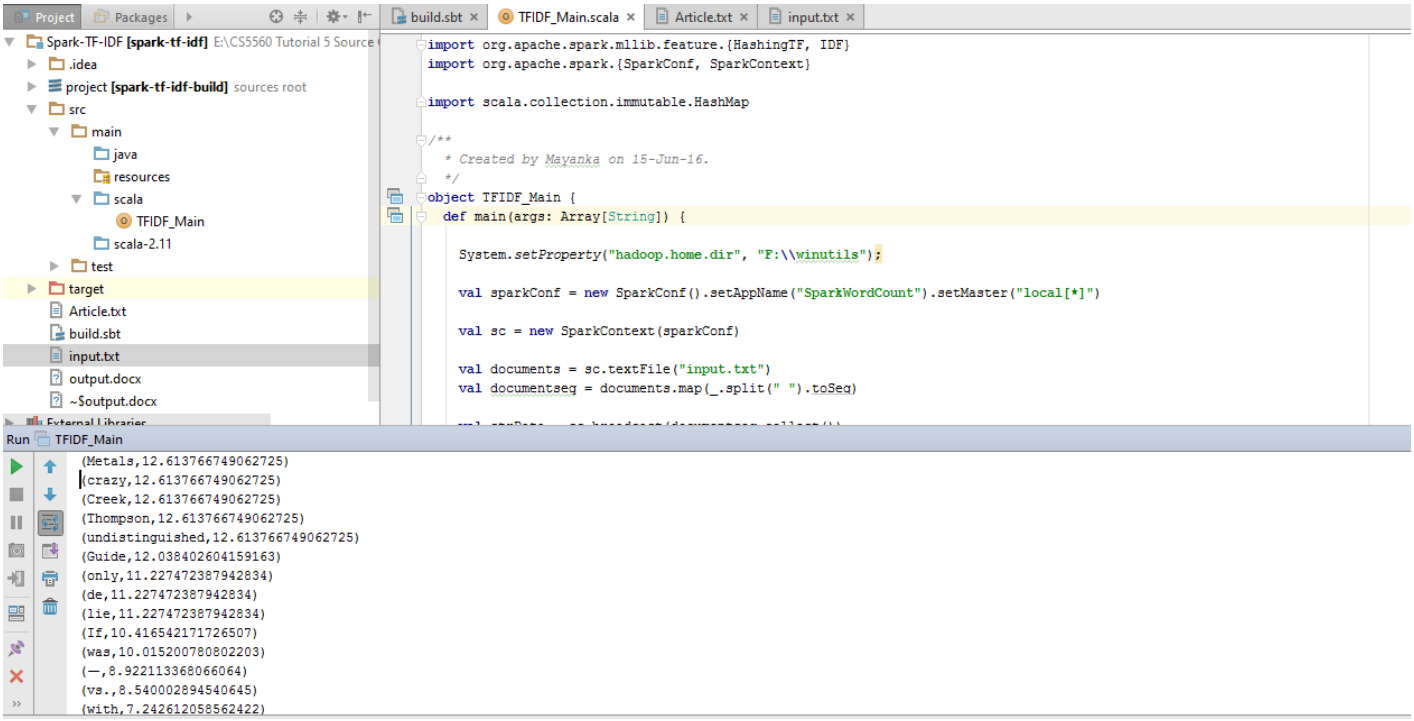
The name “Jessica” is classified as female with 95% confidence.

```
C:\>curl -G -u "f6de402c-0f6e-4c26-bbdd-b5e314767f72":"agBMTwd7DPnF" \ "https://gateway.watsonplatform.net/natural-
ssica"
curl: (6) Could not resolve host: \
{
  "classifier_id" : "2373f5x67-nlc-7103",
  "url" : "https://gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers/2373f5x67-nlc-7103",
  "text" : "jessica",
  "top_class" : "female",
  "classes" : [ {
    "class_name" : "female",
    "confidence" : 0.9571645890651067
  }, {
    "class_name" : "male",
    "confidence" : 0.042835410934893264
  } ]
}curl: (6) Could not resolve host: \
```

Sample screenshots of tutorial outputs on collected data:






Project
Packages
build.sbt
TFIDF_Main.scala
Article.txt
input.txt
Spark-TF-IDF [spark-tf-idf] E:\CS5560 Tutorial 5 Source
.idea
project [spark-tf-idf-build] sources root
src
main
java
resources
scala
TFIDF_Main
scala-2.11
test
target
Article.txt
build.sbt
input.txt
output.docx
~$output.docx
External Libraries
import org.apache.spark.mllib.feature.{HashingTF, IDF}
import org.apache.spark.{SparkConf, SparkContext}
import scala.collection.immutable.HashMap
/**
* Created by Mayanka on 15-Jun-16.
*/
object TFIDF_Main {
def main(args: Array[String]) {
System.setProperty("hadoop.home.dir", "F:\\winutils");
val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
val sc = new SparkContext(sparkConf)
val documents = sc.textFile("input.txt")
val documentseq = documents.map(_.split(" ").toSeq)
Run
TFIDF_Main
(Metals,12.613766749062725)
(crazy,12.613766749062725)
(Creek,12.613766749062725)
(Thompson,12.613766749062725)
(undistinguished,12.613766749062725)
(Guide,12.038402604159163)
(only,11.227472387942834)
(de,11.227472387942834)
(lie,11.227472387942834)
(If,10.416542171726507)
(was,10.015200780802203)
(—,8.922113368060064)
(vs.,8.540002894540645)
(with,7.242612058562422)


Tutorial-3-CoreNLP
src
main
java
Main
Project
Main.java
output.txt
SimpleCoreNLP.java
build.sbt
SBT projects
Tutorial-3-CoreNLP (auto-import ena
tutorial-3-corenlp
tutorial-3-corenlp-build
Tutorial-3-CoreNLP [tutorial-3-c
.idea
project [tutorial-3-corenlp-b
src
main
java
Main
SimpleCoreNL
resources
scala
scala-2.11
test
target
build.sbt
output.txt
External Libraries
/**
* Created by Mayanka on 13-Jun-16.
*/
public class Main {
public static void main(String args[]) {
// creates a StanfordCoreNLP object, with POS tagging, lemmatization, NER, parsing, and coreference resolution
Properties props = new Properties();
props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");
StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
// read some text in the text variable
String text = "what is the public opinion on Donald Trump?"; // Add your text here!
// create an empty Annotation just with the given text
Annotation document = new Annotation(text);
// run all Annotators on this text
pipeline.annotate(document);
Run
Main
(ROOT (SBARQ (WHNP (WP what)) (SQ (VBZ is) (NP (NP (DT the) (JJ public) (NN opinion)) (PP (IN on) (NP (NNP Donald) (NNP Trump))))) (. ?)))
-> what/WP (root)
-> is/VBZ (cop)
-> opinion/NN (nsubj)
-> the/DT (det)
-> public/JJ (amod)
-> Trump/NNP (nmod:on)
-> on/IN (case)
-> Donald/NNP (compound)
-> ?/. (punct)
[CHAIN1-["Donald Trump" in sentence 1], CHAIN2-["the public opinion on Donald Trump" in sentence 1]]

Contribution of Team members:

Ganesh Taduri (Team Leader) & Tejo Kumar: Handled the data collection part. The data is collected using Twitter API and Tweepy from Twitter. This data is in Json format. We then extracted the tweet text using Json parser. We tried using the natural language classifier for the sample data.

Data collection URL: www.twitter.com

Swatwik & Sasidhar: Conceived the project idea, designed the project architecture and necessary UML diagrams. Handled the documentation part. Implemented the tutorial concepts on the collected data.

Note: All the teammates are involved in all the aspects of project.

Project GitHub Link: https://github.com/ganeshtaduri/KDM

Concerns/Issues: Although we started with the idea of collected data from Twitter and LinkedIn, we had major glitches while collecting data. We had trouble in getting authorization keys.

Future Work: We plan to solve the data collection problem from LinkedIn and collect data from it. We then incorporate the additional linked in feature, which deals with the questions related job trends and related news.

References:

http://natural-language-classifier-demo.mybluemix.net/

http://adilmoujahid.com/posts/2014/07/twitter-analytics/

www.developers.twitter.com

www.stackoverflow.com