

# **Knowledge Discovery and Management**

## **Project Report**



**Tech Champs**

### **Team 1**

1. Jakkepalli, Rama Charan Pavan (8)
2. Puthana, Sujitha (24)
3. Yalamanchili, Sowmya (30)
4. Nandanamudi, Sreelakshmi (17)

# **1. Project Motivation, Objectives, and Significance**

## **1.1 Motivation:**

Data Science is a system to extract knowledge of data in various forms, either structured or unstructured from various domains, similar to Knowledge Discovery in Databases(KDD). Natural language processing is used for processing the text which is machine understandable and which will help for fast retrieval of data.

## **1.2 Specific Objectives:**

### **1.2.1 Easy search of information from huge amount of text.**

In present days, the amount of data is increasing and this is leading to the difficulties in handling the data. So we need the machine learning algorithms to handle these huge data. We are making use of artificial intelligence allogorithm for machine learning to handle data and search the data.

### **1.2.2 Helps in precise answer for customized questions.**

As we are using these AI algorithm for handling data, this helps in getting through different algothms available including TFIDF, NLP algorithms, word2vec algorithm, kmean algorithm, classification of data using all these process and analyzing the accuracy. This tremendous process leads to precise answer of the question.

### **1.2.3 Increase the knowledge management process.**

In the process of going through different AI algorithm to classify data and handle them. We could understand the importance of each algorithm with the specified uniqueness. By making using of all these algorithms simplify the management of data.

## **1.3 Specific Significance:**

This application helps in fetching the answer to particular questions by using NLP Process, word2vec, TF-IDF, N-gram. NLP , kmean, Classification of data, NLP algorithm is useful step for text processing and then we are extracting the relevant data.

However all the algorithm we are using in the project have its own significance.Comapring all these process to find the best process with respect to time, accuracy, cost to select the best process.

## **2. Domain and Q/A application**

We are taking News as our domain for our project and applying NLP operations on it and further applying question answering system for the dataset. For this question answering system, we are considering two datasets from News domain.

Question and answer application is build where a user can ask question like what, why, who, when related to the domain dataset. Then the algorithm is implemented to search the huge data. With very high processing speed and high accuracy, we will fetch the precise answer and display it to the user.

## **3. Related Work:**

In the present days, where the data is huge leading to data management issues. There are many algorithm already existing but the main problem in the existing algorithms are completeness and correctness. To solve this problem we need to consider all these algorithm and judge wisely which all are the algorithms that we can use to easily maintain data and give us the high accuracy. But a single algorithms or approach cannot solve this probem. Hence we should integrate multiple algorithm for high accuracy in designing the search engine.

Searching the huge amount of data is very difficult. Knowledge Graph represents the graphical representation of the entities and interrelated relationship. There are different knowledge graph available in the market but googles knowledge graph is the popular search engine algorithm. Best knowledge graph can be designed solving the completeness and correctness issue by integrating different approaches of knowledge graph available in the market.

Data souces that are available to us are limited. We can increase the accuracy to provide the best awswer to any question is by considering all the data souces that are available on the web. The solution for this approach is the knowdege vault that was made available to us by google that takes the data in RDD triplets i.e., subject, object, predicate. After collecting the data and finding the entities our next problem would be organizing the data. We Deep Dive approach helps in

resolving the problem of extraction of data and its integration to fetch accurate prediction making the training process easy.

After the data is represented in RDF triplets, the semantic relationship can be organized using the FehSen to merge the related information leading to more simplified data. It is known fact that structured data is easy to handle than unstructured data. Fonduer is the approach in focusing the construction of the structured data from the plain text. By using all these approaches helps in improving the handling of the data and solve the “completeness and correctness” problem.

## 4. Specific Datasets

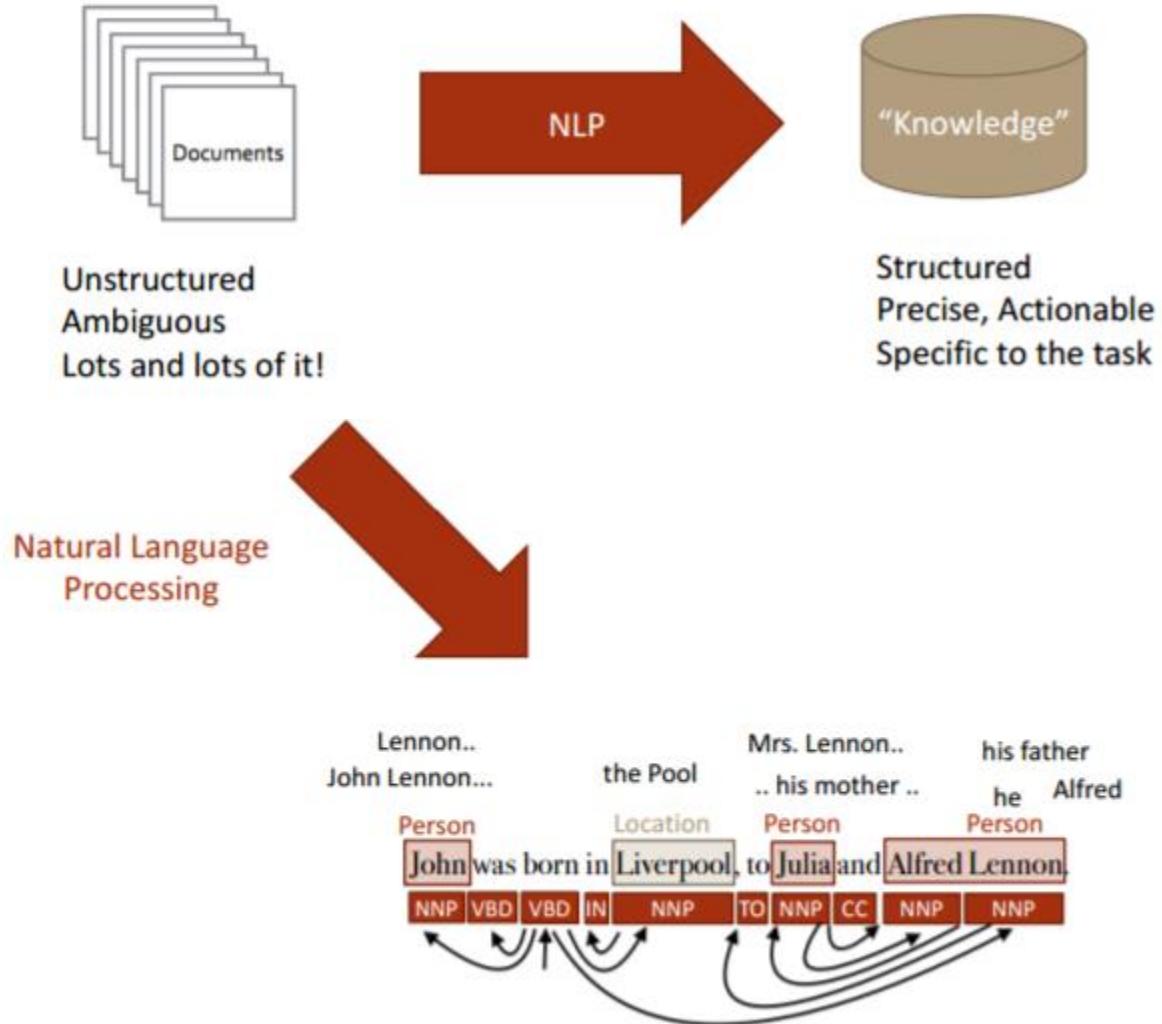
For our project implementation, we have considered two datasets as follows:

- WikiRef220
  - WikiRef220 is the collection of news articles taken from Wikipedia pages.
  - This dataset includes the information in the form of text data.
  - The articles included in this dataset are November 2015 Paris attack, Flight 370 Malaysian Airlines, Premier League, Michelle Obama, Samsung Galaxy.
- BBC News-In this especially we have selected politics area and sports.
  - This dataset includes the news articles collected from BBC.
  - This dataset was made available mainly for machine learning research. We are using this dataset for our process.
  - We mainly selected the political area and sports area of the BBC news dataset.

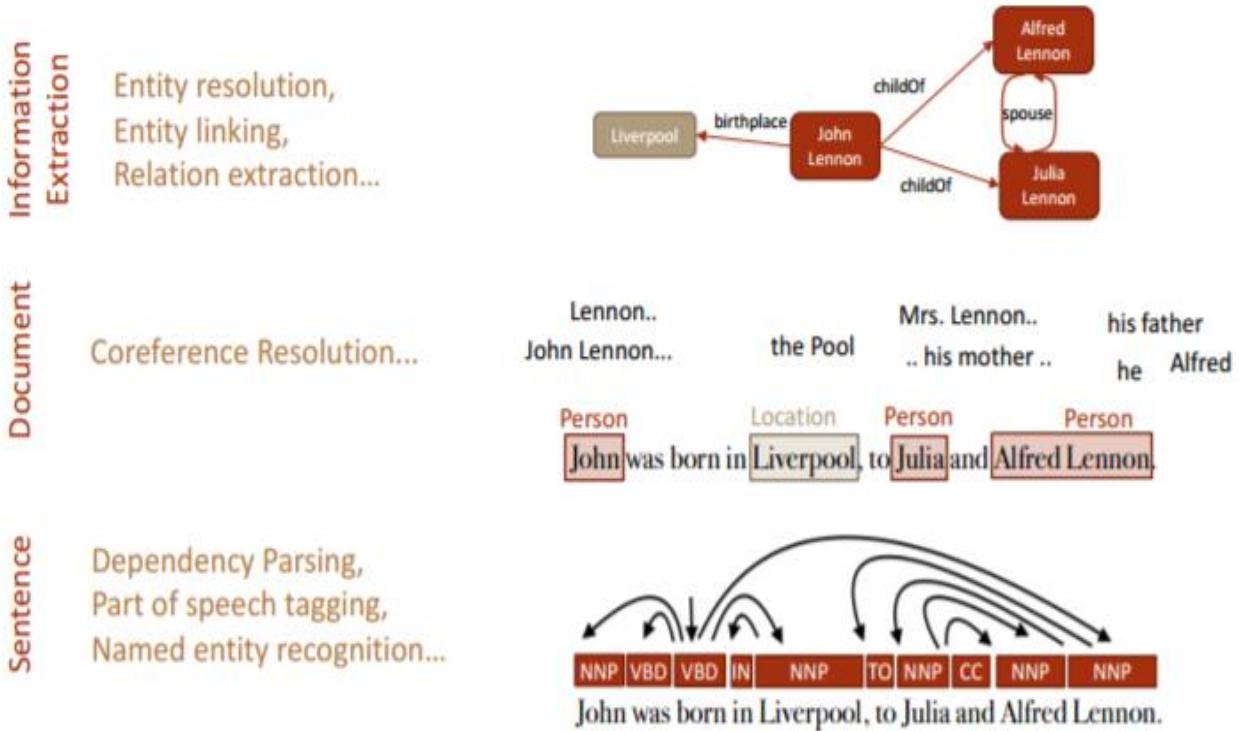
## 5. Design

### 5.1 Workflow

Step 1: Natural language processing – This process includes the identification of tokens, lemmatization, named entity recognition (NER), co-reference resolution.



Step 2: Information Retrieval – Retrieving the information from the text. We are including the identification of the NER i.e., PERSON, LOCATION, ORGANIZATION.



Step 3: Topic Discovery – Topic discovery helps identification of the topics from the context question.

Step 4: Knowledge Graph construction – Construction of the knowledge graph from generated NER.

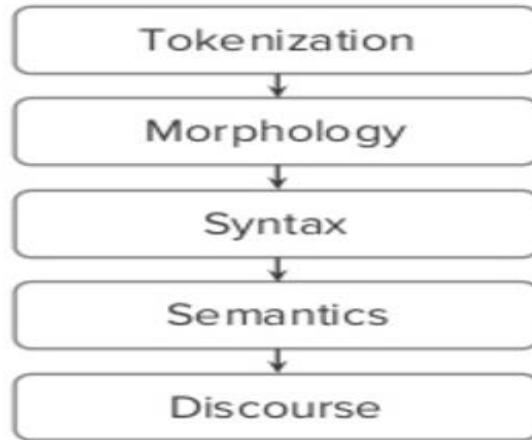
## 5.2 NLP



Natural Language Processing is the process that makes the computer understand, analyze and extract meaning from human understandable language in a useful and smart way. NLP algorithms help the organizing and to structure data in order to perform automatic summarization, named entity recognition, translation, relationship extraction, speech recognition, sentiment analysis, topic segmentation.

Steps in NLP designing:

- Tokenization – Break the text data into sentence, words.
- Lemmatization – Recognizing the base form of word.
- Morphology – Includes Part of Speech recognition , stemming i.e., excluding the postfix words to get the base root word, Named entity recognition.
- Syntax – Parsing Constituency or dependency
- Semantic – Coreference resolution i.e., finding the context that belongs to same entity.



## 5.3 Information Retrieval

Information retrieval is the process of tracing through the stored data and recovering specific information from huge amount of stored data. It is very difficult to find the specific data from such a huge amount of data. So we are using the below approaches to simplify the information retrieval process.

### 5.3.1 Term Frequency Inverse Document Frequency(TFIDF)

TFIDF is the numerical weight of the tokenized word that demonstrate the importance of the word in the huge document. The weight of the word increases with the repetition of word in the document. TFIDF is can be represented as TF\*IDF i.e., product of term frequency i.e., occurrence of word in a particular document and Inverse document frequency i.e., log value number of documents the word exists divided by the total number of documents.

$$w_{i,j} = tf_{i,j} \times \log \left( \frac{N}{df_i} \right)$$

$tf_{i,j}$  = number of occurrences of  $i$  in  $j$

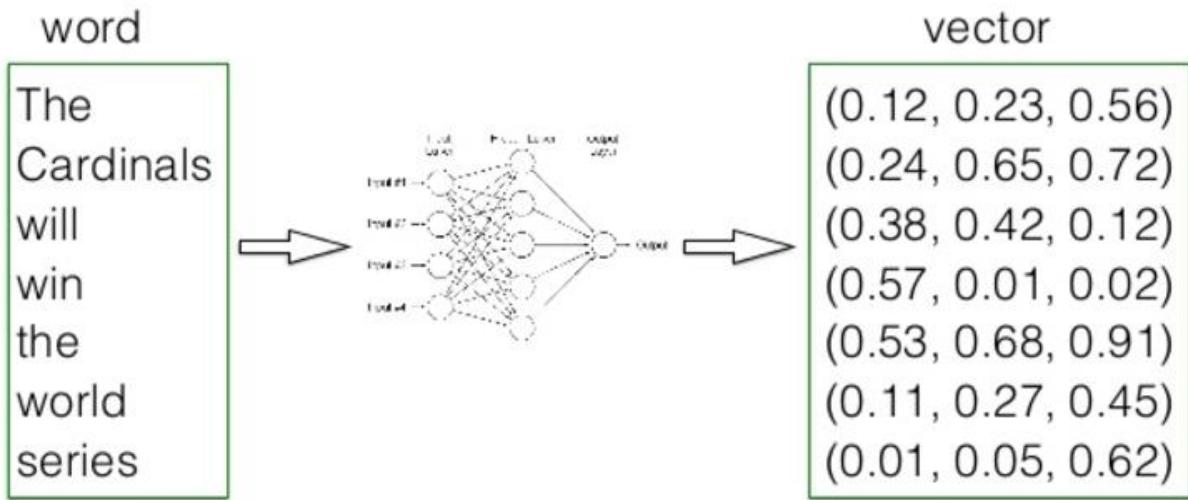
$df_i$  = number of documents containing  $i$

$N$  = total number of documents

### 5.3.2 Word2Vector

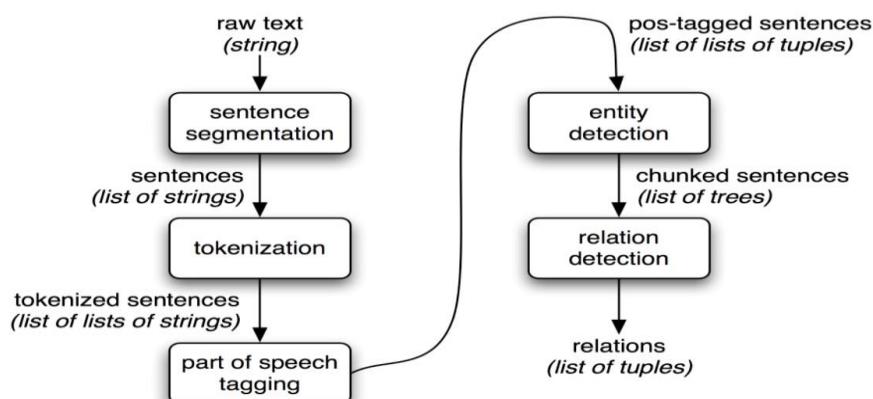
Word2Vec is the process of construction of the vector from the huge text document. All the word vectors are marked in the vector space where the closely meaning words are very close to each other. Thus mean that they are the same grouped words.

This model leads to the other distributed representation model i.e., Continuos bag of words, Skip gram. Bag of words mean predicting the words from context and the skip gram is predicting the context from words.



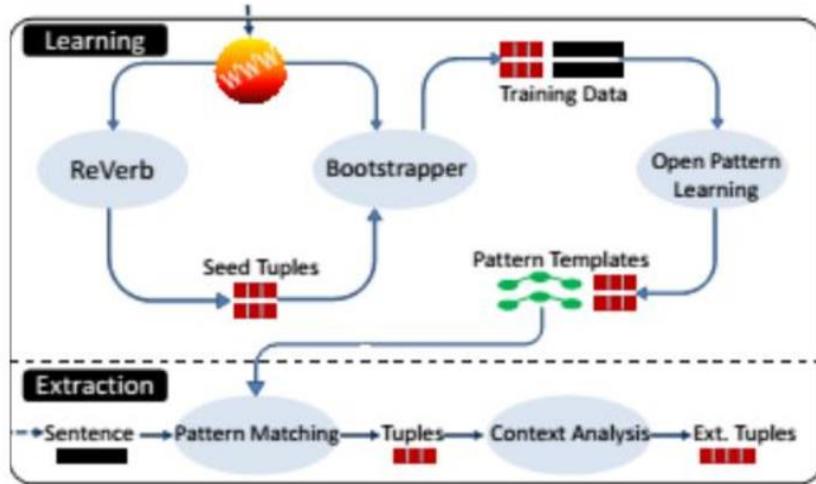
### 5.4 Information Extraction

Information extraction involves the process of extracting the information from the unstructured or the semi structured data i.e., normal text document. Information extraction utilizes the NLP process to extract the relationship between the entities.



### 5.4.1. OpenIE

Open information extraction is the process of extracting the RDF triplets. RDF triplets are subject, object, predicate.



Steps in OpenIE Triplets Extraction:

- Input the data to the system.
- Matching the pattern from already predefined algorithm.
- Extracting the tuples.
- Analyze the context.
- Extracting RDF triplets.

### 5.4.2 WordNet

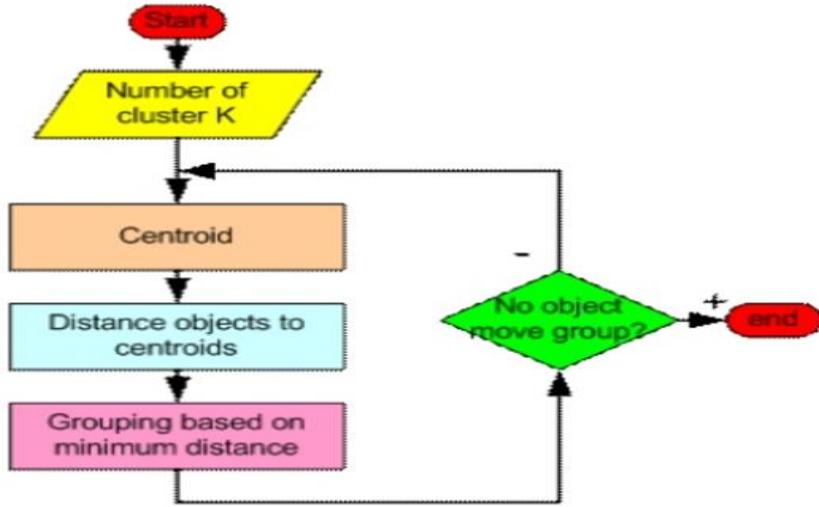
WordNet involves the generation of the synonym for a particular token of word. WordNet algorithm analyze the data to extract the correct information though we use the synonym of the word. WordNet generate the synsets, which is the group of words with similar meaning.

## 5.5 Machine Learning

Machine learning involves the process of automatic analyzation of data using the advanced artificial intelligence algorithm. This process simplifies the prediction from the existing huge data. Machine learning algorithms are very efficient.

### 5.5.1 Clustering

Clusters represent the group of similar kind. In data analysis we use clustering process to group together similar words using vector.

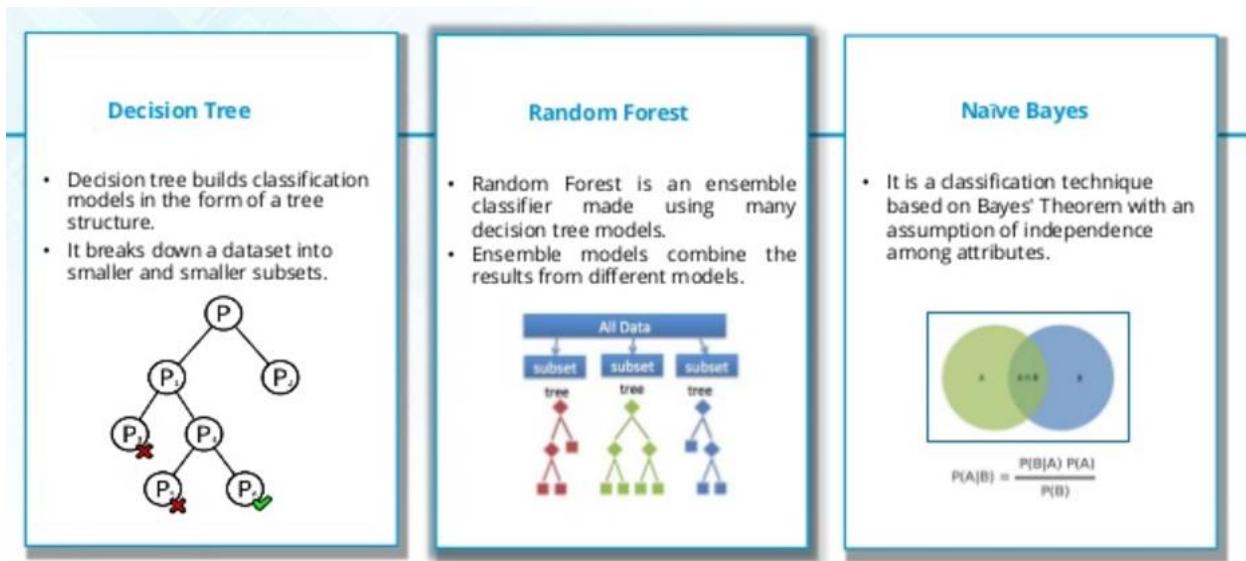


Steps involved in k-mean clustering:

- Input the dataset.
- Tokenize the input data.
- Implement the lemmatization i.e., generating the dictionary word.
- Remove the stop words.
- Generae the TFIDF.
- Determine the Kmeans.

### 5.5.2 Classification

Classification is the extension of kmean clustering. There exists decision tree, naïve bayes, random forest approach for classification. Below are the different classification approaches available.



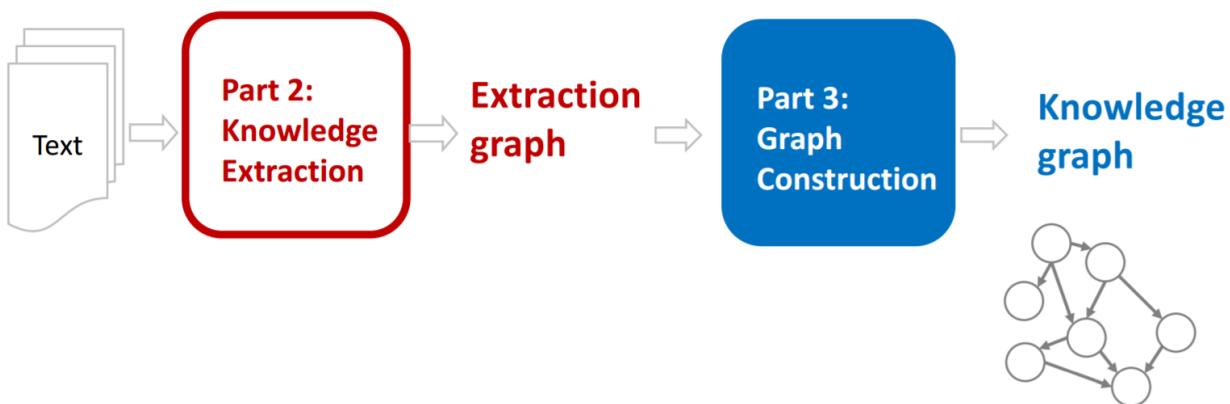
Steps involved in classification:

- Input the dataset.
- Tokenize the input data.
- Implement the lemmatization i.e., generating the dictionary word.
- Remove the stop words.
- Generae the TFIDF.
- Process one of the above classification approach.

## 5.6 Knowledge Graph

Knowlwdge Graph is used to simplify the search results. This graph represent the graphical representation of the flow of the text data. The main advatange of using this knowledge graph is simplified diagrammatical representation of the huge data, helps in easy knowledge transfer and documentaio easy.

### 5.6.1 Design workflow of knowledge Graph



Steps followed in designing this knowledge graph:

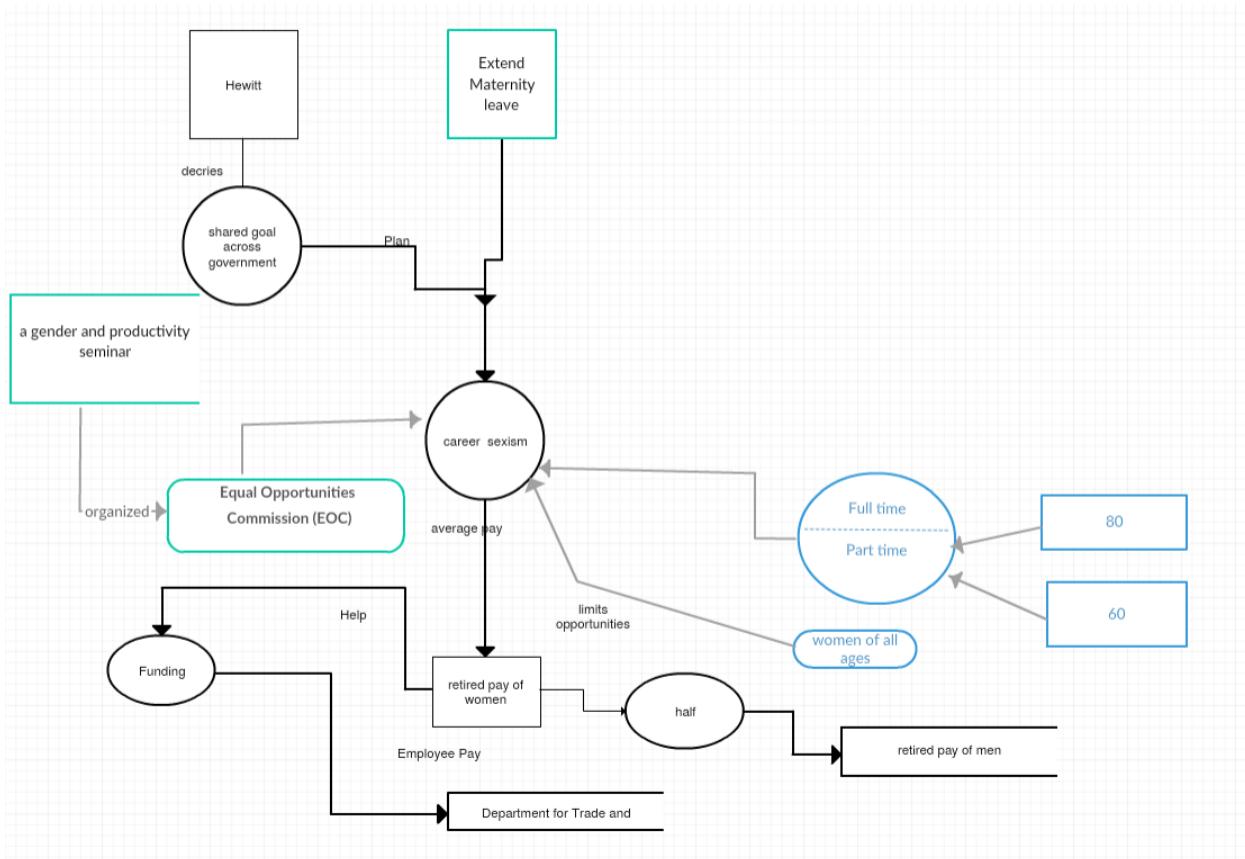
1. Recognizing the named entity reference including the people, organization, location, date etc.
2. Designing the data schema i.e., finding the relationship between these entities.
3. Representing them in diagrammatical graph

## 5.6.2 Knowledge Graph for our dataset:

We do not have any specified rules for designing this knowledge graph. Different companies have their own knowledge graph construction and follows their own rules.

We first recognized the entities in our dataset and designed the data schema to generate the relationships between the entities. Finalized the flow of data.

Below is the diagrammatical representation of the knowledge graph that is designed for our datasets.



## **5.7 A Question-Answer Set for our Dataset.**

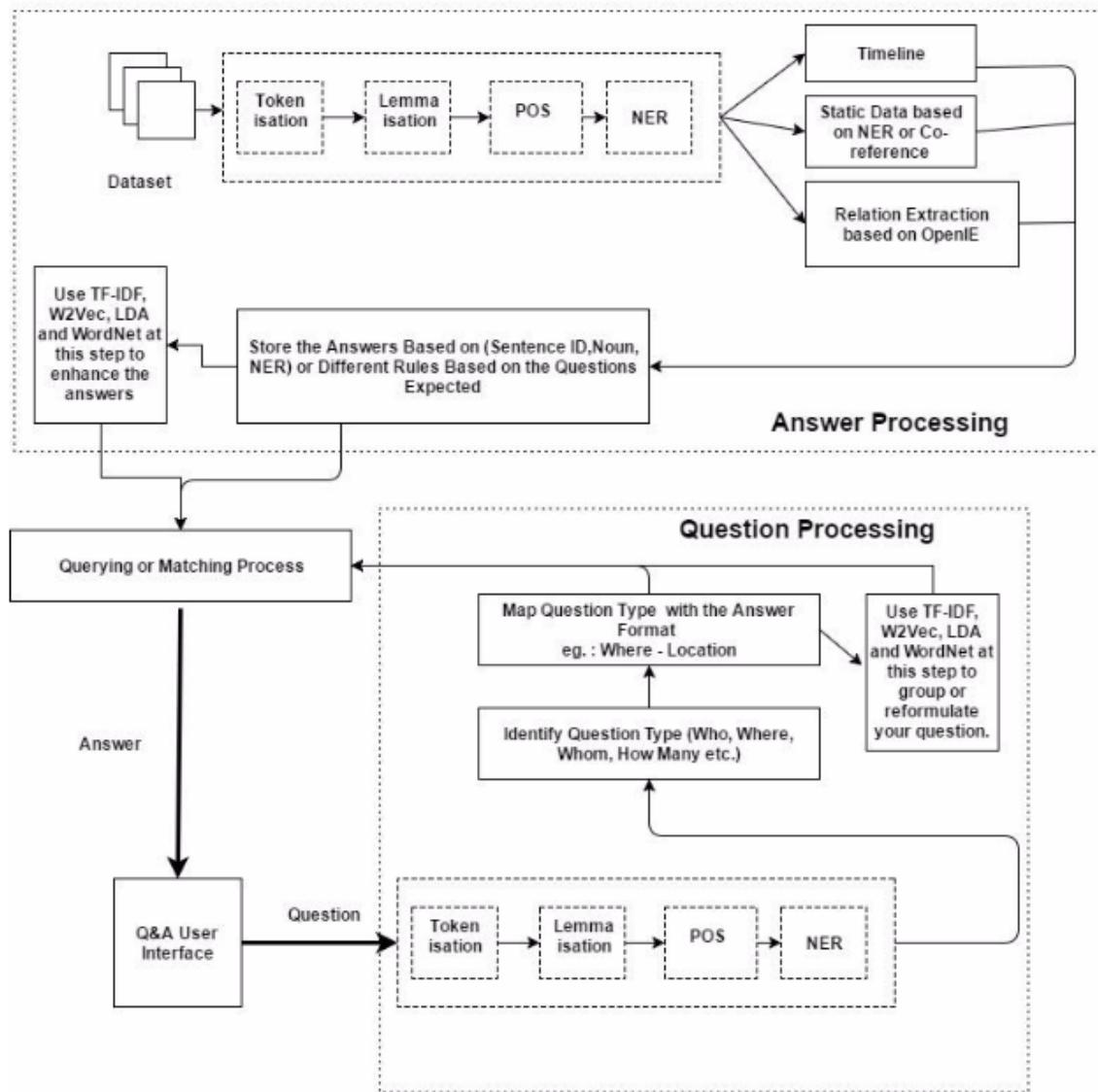
We are designing the questions from datasets considering mainly the PERSON, LOCATION, ORGANIZATION, NUMBER entity.

1. When was Obama born?  
Born on Aug. 4, 1961.
2. Where did Obama did his schooling?  
Punahou School.
3. Who is father of Obama?  
Barack Hussein Obama.
4. Whom did Obama compete in primary race?  
Hillary Rodham Clinton.
5. What is the minimum duration for maternity leave?  
6 months.
6. What is the topic about?  
career sexism.
7. Who is the speaker?  
Ms. Hewitt.
8. What is the average pay for full-time women.  
80p
9. What is the average pay for part-time women.  
60p.
10. What is the average pay for retired women compared to men?  
Half.

## **6. Implementation**

### **6.a Workflow diagram for our dataset**

As the diagram mentions we are taking the question and the dataset and we are applying the NLP operations on tha dataset and then we are storing the result of the NLP and on top of it applying the other shown approeches for the better performance.



## Output of NLP operations for our dataset

We have performed the NLP operations on the dataset which we have chosen and the result of each operation is shown in the below mentioned screenshots.

Tokenization:

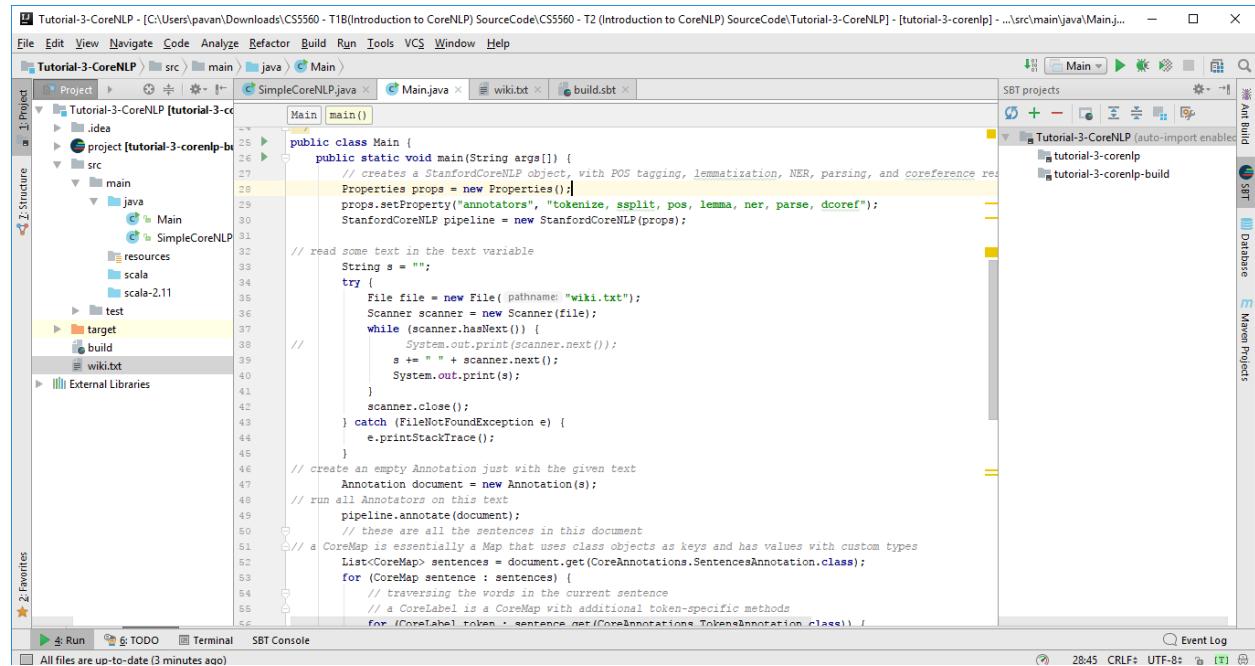
Lemmatization:

POS Tagging:

NER:

Coreference Resolution:

Below is the code for all the operations of the NLP performed on our dataset.



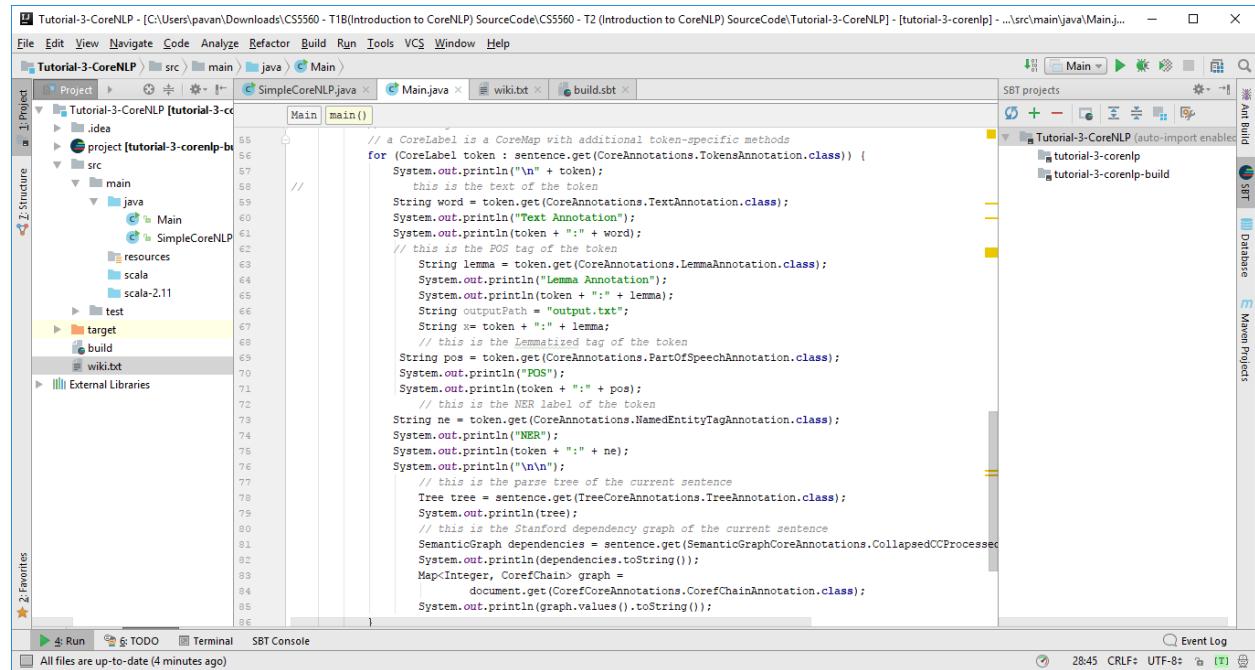
The screenshot shows the IntelliJ IDEA interface with the project 'Tutorial-3-CoreNLP' open. The code editor displays the file 'Main.java' with the following content:

```
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 s = "";
        try {
            File file = new File (pathname: "wiki.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext()) {
                System.out.print(scanner.next());
                s += " " + scanner.next();
                System.out.print(s);
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        // create an empty Annotation just with the given text
        Annotation document = new Annotation(s);
        // run all Annotators on this text
        pipeline.annotate(document);
        // these are all the sentences in this document
        List<CoreMap> sentences = document.get(CoreAnnotations.SentencesAnnotation.class);
        for (CoreMap sentence : sentences) {
            // traversing the words in the current sentence
            // a CoreLabel is a CoreMap with additional token-specific methods
            for (CoreLabel token : sentence.get(CoreAnnotations.TokensAnnotation.class)) {

```



The screenshot shows the IntelliJ IDEA interface with the project 'Tutorial-3-CoreNLP' open. The code editor displays the file 'Main.java' with the following content:

```
// a CoreLabel is a CoreMap with additional token-specific methods
for (CoreLabel token : sentence.get(CoreAnnotations.TokensAnnotation.class)) {
    System.out.println("\n" + token);
    this is the text of the token
    String word = token.get(CoreAnnotations.TextAnnotation.class);
    System.out.println("Text Annotation");
    System.out.println(token + ":" + word);
    // this is the POS tag of the token
    String lemma = token.get(CoreAnnotations.LemmaAnnotation.class);
    System.out.println("Lemma Annotation");
    System.out.println(token + ":" + lemma);
    String outputPath = "output.txt";
    String x = token + ":" + lemma;
    // this is the Lemmatized tag of the token
    String pos = token.get(CoreAnnotations.PartOfSpeechAnnotation.class);
    System.out.println("POS");
    System.out.println(token + ":" + pos);
    // this is the NER label of the token
    String ne = token.get(CoreAnnotations.NamedEntityTagAnnotation.class);
    System.out.println("NER");
    System.out.println(token + ":" + ne);
    System.out.println("\n\n");

    // this is the parse tree of the current sentence
    Tree tree = sentence.get(TreeCoreAnnotations.TreeAnnotation.class);
    System.out.println(tree);
    // this is the Stanford dependency graph of the current sentence
    SemanticGraph dependencies = sentence.get(SemanticGraphCoreAnnotations.CollapsedCCProcessedDependenciesAnnotation.class);
    System.out.println(dependencies.toString());
    Map<Integer, CorefChain> graph =
        document.get(CoreAnnotations.CorefChainAnnotation.class);
    System.out.println(graph.values().toString());
}

```

Below are the outputs of the operations after applying the NLP.

The image shows two screenshots of a Gmail inbox. Both screenshots have a redacted subject line and show the message content in a code-like format representing the NLP output.

**Message 1 (Top Screenshot):**

```

<root> (S (ADVP (RB Once)) (.,) (PP (IN during) (NP (NP (DT the) (NN beat)) (PP (IN of) (NP (NP (DT the) (JJ primary) (NN race)) (PP (IN between) (NP (NP (NNP Obama)) (CC and) (NP (NNP Hillary) (NNP Rodham) (NNP Clinton))))))) (. ,) (NP (DT a) (NN claims) (VP (VBD came) (PP (IN from) (NP (NP (NNP Bill) (NNP Clinton)) (PP (IN that) (NP (NP (PRP he))))))) (.,))) (.,))
    -> Once/RB (modad)
    -> ./, (punct)
    -> heat/NN (modduring)
    -> during/IN (case)
    -> ., (punct)

```

**Message 2 (Bottom Screenshot):**

```

<root> (S (ADVP (RB Once)) (.,) (PP (IN during) (NP (NP (DT the) (NN beat)) (PP (IN of) (NP (NP (DT the) (JJ primary) (NN race)) (PP (IN between) (NP (NP (NNP Obama)) (CC and) (NP (NNP understand) (.,))) (NP (NP (NNP Clinton))))))) (. ,) (NP (DT a) (NN claims) (VP (VBD came) (PP (IN from) (NP (NP (NNP Bill) (NNP Clinton)) (PP (IN that) (NP (NP (PRP he))))))) (.,))) (.,))
    -> Once/RB (modad)
    -> ./, (punct)
    -> heat/NN (modduring)
    -> during/IN (case)
    -> ., (punct)

```

The NLP output consists of tokens and their annotations, such as part-of-speech (POS), dependency relations (head, dependents), and semantic roles (like `heat`, `beat`, `understand`, etc.). The annotations provide detailed information about the sentence structure and meaning.

File Edit Format View Help

Chicago" in sentence 18, "Chicago" in sentence 19, "it" in sentence 19, "Chicago" in sentence 21], CHAIN158-[ "Chicago , the antipode of remote Honolulu , deep in the fold of the mainland" : , CHAIN185-[ "Hawaii and Chicago" in sentence 16, "the two main threads weaving through the cloth of Barack Obama 's life" in sentence 16], CHAIN187-[ "the cloth of Barack Obama 's life" in : inner conflicts" in sentence 19], CHAIN211-[ "the subtle , coolly ambitious persona" in sentence 19], CHAIN212-[ "the presidential election" in sentence 19], CHAIN214-[ "first" in sentence 20], sentence 24, "It" in sentence 25, "It" in sentence 26, "It" in sentence 27, "that" in sentence 28], CHAIN232-[ "community work" in sentence 24], CHAIN233-[ "lives of public service" in sentence 24], HATN255-[ "his grandparents , Madelyn and Stan Dunham , Toot and Gramps , the white couple with whom he lived for most of his teenage years , she practical and determined , he impulsive , he ve , hokey , well-intentioned and , by his grandson 's account , burdened with the desperate lost hopes of a Willy Loman-style salesman" in sentence 26], CHAIN261-[ "the white couple with whom he lived for most of his teenage years , she practical and determined , he impulsive , he ve , hokey , well-intentioned and , by his grandson 's account , burdened with the desperate lost hopes of a Willy Loman-style salesman" in sentence 26], CHAIN261-[ "the white couple with whom he lived for most of his teenage years , she practical and determined , he impulsive , he ve , hokey , well-intentioned and , by his grandson 's account , burdened with the desperate lost hopes of a Willy Loman-style salesman" in sentence 26], CHAIN284-[ "the west Coast to Hawaii" in sentence 27], CHAIN286-[ "52" in sentence 28], CHAIN291-[ "their daughter , who followed the Pacific farther to Indonesia" in sentence 28], CHAIN292-[ "in sentence 29], CHAIN310-[ "1995" in sentence 29], CHAIN313-[ "his debut" in sentence 29], CHAIN315-[ "the national stage" in sentence 29], CHAIN316-[ "a book about himself that searched for , a politician , without his mother 's sensibility , naïve or adventurous or both" in sentence 30, "They" in sentence 32, "they" in sentence 33, "they" in sentence 33], CHAIN336-[ "a politici- re him , and perhaps most like Bill Clinton" in sentence 34], CHAIN362-[ "many presidential aspirants before him" in sentence 34], CHAIN364-[ "most like Bill Clinton" in sentence 34], CHAIN364-[ "most like Bill Clinton" in sentence 36], CHAIN379-[ "their backgrounds and families" in sentence 36]]

heat-5  
Text Annotation  
heat-5:heat  
Lemma Annotation  
heat-5:heat  
POS  
heat-5:NW  
NER  
heat-5:O

(ROOT (S (ADV (RB Once) (, ,) (PP (IN during) (NP (NP (DT the) (NN heat)) (PP (IN of) (NP (NP (DT the) (JJ primary) (NN race)) (PP (IN between) (NP (NP (NNP Obama)) (CC and) (NP (NNP Hill)

-> came/VBD (root)  
-> Once/RB (advmod)  
-> ,/ (punct)  
-> heat/NN (nmod:during)  
-> during/IN (case)  
-> the/DT (det)  
-> race/NN (nmod:of)  
-> of/IN (case)  
-> the/DT (det)  
-> primary/JJ (amod)  
-> Obama/NNP (nmod:between)  
-> between/IN (case)  
-> and/CC (cc)  
-> clinton/NNP (conj:and)  
-> Hillary/NNP (compound)  
-> Rodham/NNP (compound)  
-> Clinton/NNP (nmod:between)  
-> ,/ (punct)

## 6.b TF-IDF for our dataset

Generating the term frequency for the words in the dataset.

The screenshot shows an IDE interface with the following details:

- Project:** Spark\_TFIDF\_W2V
- File:** TF\_IDF.scala
- Code:** The code implements a TF-IDF calculation. It reads a file named "bbcdataset(politics).txt", processes it into a HashMap of word counts, and then calculates the TF-IDF values for each word.
- Output:** The output window displays the raw text from the input file, which is a political speech by Barack Obama.
- IDE Features:** The interface includes a Project tree, a Structure view, and various toolbars for build, SBT, and Maven integration.

Generated Output.

The screenshot shows an IDE interface with the following details:

- Project:** Spark\_TFIDF\_W2V
- File:** TF\_IDF.scala
- Output File:** part-00000
- Content:** The file contains a list of words and their TF-IDF scores. The first few lines are:

```
1 [job, 6.695905734286686]
2 (Career, 6.338324625039507)
3 (cost, 6.238324625039507)
4 (the, 5.54517744479562)
```
- IDE Features:** The interface includes a Project tree, a Structure view, and various toolbars for build, SBT, and Maven integration.

## Generation N-gram for the dataset

```

NGRAM main(args: Array[String])
def main(args: Array[String]): Unit = {
  val sparkConf = new SparkConf().setAppName("TermFrequency - IDF").setMaster("local[*]")
  val sc = new SparkContext(sparkConf)
  val a = sc.textFile("ProjectData/wiki(obama).txt", 2)
  a.foreach(f=>println(f.mkString(" ")))
}

17/06/23 15:30:29 INFO deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
Hawaii is about the forces that shaped him, and Chicago is about how he reshape
17/06/23 15:30:29 INFO deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
On weekday mornings as a teenager, Barry Obama left his grandparents' apartment
Hawaii involves the struggles of a teenage hapa at Punahoa School who wanted no
It is about his mysterious father, Barack Hussein Obama, an imperious if alluri
An adolescent life told in five Honolulu blocks, confined and compact, but far,
Those who come from islands are inevitably shaped by the experience. For Obama,
And that was not far enough for their daughter, who followed the Pacific farthe
As the son of a white woman and a black man, he grew up as a multiracial kid, a
The simple fact is that he would not exist as a human being, let alone as a pol

```

## Generating the TF\_IDF for N-gram output.

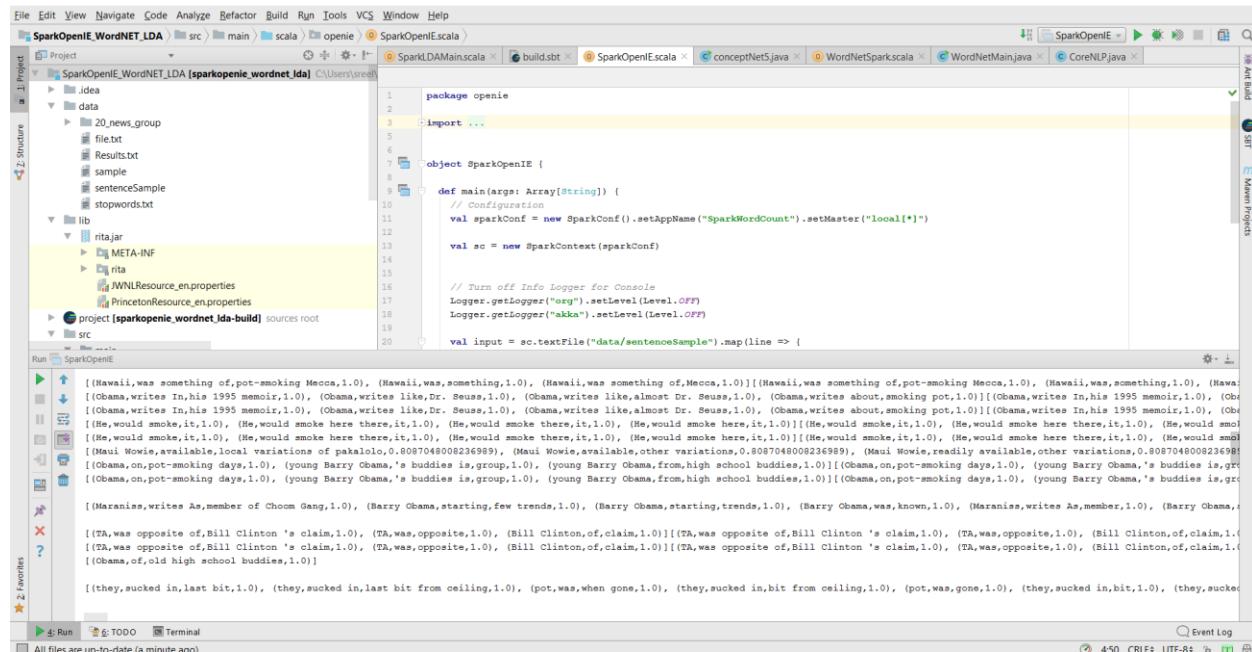
```

1 [e,32.88103504155912]
2 (a,21.087024039092262)
3 (t,28.098339035514158)
4 (i,23.913480030224818)
5 (h,22.717806028713575)
6 (o,22.119969027957954)
7 (n,17.935110022668614)
8 (l,15.543762019646131)
9 (r,11.358903014356788)
10 (c,10.163229012845546)
11 (d,8.967555011334307)
12 (f,8.369718010578685)
13 (be,7.7718810098230655)
14 (s,7.7718810098230655)
15 (g,6.931471805599453)
16 (m,6.576207008311824)
17 (p,5.9783700075562045)
18 (-,5.249110622493388)
19 (6,4.605170185988092)
20 (w,4.184859005289343)
21 (,,4.184859005289343)
22 (0,3.7942399697717626)
23 ('',3.7942399697717626)
24 (S,3.7942399697717626)
25 (b,3.5870220045337224)
26

```

## 6.c Information Extraction

### OpenIE for our dataset:

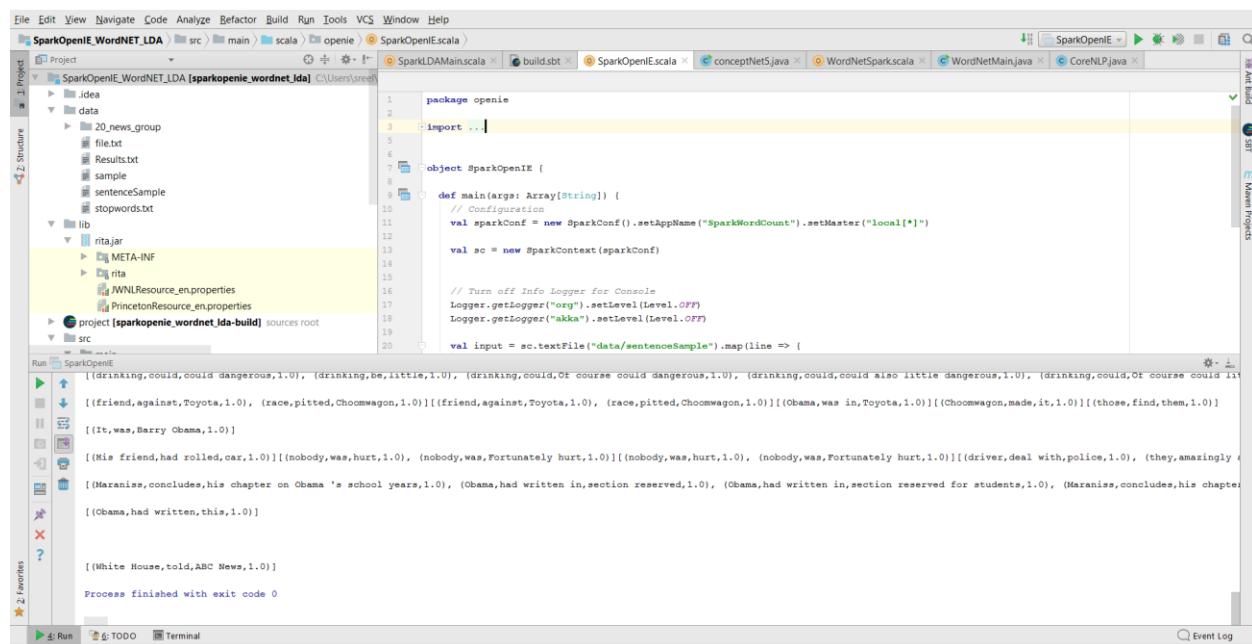


```
package openie
import ...
object SparkOpenIE {
  def main(args: Array[String]) {
    // Configuration
    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)

    // Turn off Info Logger for Console
    Logger.getLogger("org").setLevel(Level.OFF)
    Logger.getLogger("akka").setLevel(Level.OFF)

    val input = sc.textFile("data/sentenceSample").map(line => {
```

The output window displays a long list of triples extracted by the OpenIE system. The triples are separated by newlines and follow a standard triple pattern: subject, predicate, object. Many of the subjects and objects are names of people, such as "Barry Obama", "Bill Clinton", and "Dr. Seuss". The predicates often involve actions like "smoke", "drink", or "be", along with descriptive adjectives like "dangerous" or "little". The list is very long, spanning multiple lines of the screenshot.



```
package openie
import ...
object SparkOpenIE {
  def main(args: Array[String]) {
    // Configuration
    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)

    // Turn off Info Logger for Console
    Logger.getLogger("org").setLevel(Level.OFF)
    Logger.getLogger("akka").setLevel(Level.OFF)

    val input = sc.textFile("data/sentenceSample").map(line => {
```

The output window displays a different set of triples compared to the first screenshot. These triples also follow the subject-predicate-object pattern. Some examples include "drinking,could,could dangerous,1.0", "drinking,be,little,1.0", and "friend,against,Toyota,1.0". The list is shorter than the one in the first screenshot.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar displays the project structure for "SparkOpenIE\_WordNET\_LDA". It includes a "data" folder containing "20\_news\_group", "file.txt", "Results.txt", "sample", "sentimentSample", and "stopwords.txt". A "lib" folder contains "rita.jar" and "META-INF" (which includes "WNLResource\_en.properties" and "PrincetonResource\_en.properties").
- Code Editor:** The main window shows the "SparkOpenIE.scala" file. The code implements a Spark application named "SparkOpenIE" that reads from "file.txt", processes it using CoreNLP, and outputs results.
- Run Tab:** The bottom left shows the "Run" tab with the configuration "SparkOpenIE" selected.
- Event Log:** The bottom right shows the "Event Log" tab with the message "2043 CRLF= UTF-8" and a green checkmark icon.

```
package openie

import ...

object SparkOpenIE {
    def main(args: Array[String]) {
        // Configuration
        val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
        val sc = new SparkContext(sparkConf)

        // Turn off Info Logger for Console
        Logger.getLogger("org").setLevel(Level.OFF)
        Logger.getLogger("akka").setLevel(Level.OFF)

        val input = sc.textFile("data/sentenceSample").map(line => {
            // Getting OpenIE Form of the word using Ida.CoreNLP
            ...
        })
    }
}
```

Screenshot of the IntelliJ IDEA IDE interface showing the SparkOpenIE\_WordNET\_LDA project. The code editor displays the `SparkLDA.scala` file with Scala code for a LDA model. The terminal window shows the output of a command-line run, indicating success with exit code 0. The status bar at the bottom right shows the compilation completed successfully in 3s 545ms (2 minutes ago).

```
package openie
import ...
object SparkOpenIE {
  def main(args: Array[String]) {
    // Configuration
    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)

    // Turn off Info Logger for Console
    Logger.getLogger("org").setLevel(Level.OFF)
    Logger.getLogger("akka").setLevel(Level.OFF)

    val input = sc.textFile("data/sentenceSample").map(line => {
      // Getting OpenIE Form of the word using lda.CoreNLP
      ...
    })
  }
}
```

Screenshot of the IntelliJ IDEA IDE interface showing the same project and code as the first screenshot. The terminal window shows the output of a command-line run, indicating success with exit code 0. The status bar at the bottom right shows the compilation completed successfully in 3s 545ms (5 minutes ago).

```
package openie
import ...
object SparkOpenIE {
  def main(args: Array[String]) {
    // Configuration
    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)

    // Turn off Info Logger for Console
    Logger.getLogger("org").setLevel(Level.OFF)
    Logger.getLogger("akka").setLevel(Level.OFF)

    val input = sc.textFile("data/sentenceSample").map(line => {
      // Getting OpenIE Form of the word using lda.CoreNLP
      ...
    })
  }
}
```

## Wordnet for our dataset

Screenshot of the IntelliJ IDEA IDE showing the WordNetMain.java code and its execution output.

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SparkOpenIE_WordNET_LDA src main scala wordnet WordNetMain
Project SparkLDA.scala build.sbt SparkOpenEscala conceptNet5.java WordNetSparkscala WordNetMain.java CoreNLP.java
  .idea
    20_news_group
      file.txt
      Results.txt
      sample
      sentenceSample
      stopwords.txt
  lib
    rita.jar
      META-INF
        rita
          AWNLResource_en.properties
          PrincetonResource_en.properties
  project [sparkopenie_wordnet_lda-build] sources root
  src
    main
      scala
        concept5
          conceptNet5
  Run WordNetMain
  C1
    did-12
    Text Annotation
    did-12:did
    Lemma Annotation
    did-12:do

    Finding parts of speech for do.
    v
    n

    Synonyms for did (pos: v)
    accompany
  Run TODO Terminal
  All files are up-to-date (a minute ago)
  
```

The code in WordNetMain.java reads a file named "sentenceSample" and processes it using the StanfordCoreNLP pipeline. It prints the parts of speech for the word "do". The output shows that "do" is a verb (v) and lists its synonyms, which include "accompany".

Screenshot of the IntelliJ IDEA IDE showing the WordNetMain.java code and its execution output.

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SparkOpenIE_WordNET_LDA src main scala wordnet WordNetMain
Project SparkLDA.scala build.sbt SparkOpenEscala conceptNet5.java WordNetSparkscala WordNetMain.java CoreNLP.java
  .idea
    20_news_group
      file.txt
      Results.txt
      sample
      sentenceSample
      stopwords.txt
  lib
    rita.jar
      META-INF
        rita
          AWNLResource_en.properties
          PrincetonResource_en.properties
  project [sparkopenie_wordnet_lda-build] sources root
  src
    main
      scala
        concept5
          conceptNet5
  Run WordNetMain
  criticise
  criticise
  deconstruct
  disc-jockey
  .-12
  Text Annotation
  .-12:.
  Lemma Annotation
  .-12:.

  Finding parts of speech for ..
  Process finished with exit code 0
  
```

The code in WordNetMain.java reads a file named "sentenceSample" and processes it using the StanfordCoreNLP pipeline. It prints the parts of speech for the word "..". The output shows that ".." is a punctuation mark (.) and lists its synonyms, which include "criticise", "deconstruct", and "disc-jockey".

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SparkOpenIE\_WordNET\_LDA [sparkopenie\_wordnet\_lda] C:\Users\sreel\src\main\scala\wordnet > WordNetMain

```

WordNetMain [main()]
  30 import java.io.IOException;
  31
  32 public class WordNetMain {
  33     public static void main(String args[]) {
  34
  35         RiWordNet wordnet = new RiWordNet( "C:\\\\Users\\\\sreel\\\\IdeaProjects\\\\WordNet-3.0" );
  36         Properties props = new Properties();
  37         props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");
  38         StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
  39         String line = null;
  40         String fileName = "data/sentenceSample";
  41
  42
  43         try {
  44             // FileReader reads text files in the default encoding.
  45             FileReader fileReader =
  46                 new FileReader(fileName);
  47
  48             // Always wrap FileReader in BufferedReader.
  49             BufferedReader bufferedReader =
  50                 new BufferedReader(fileReader);
  
```

Finding parts of speech for News.

- that-7
- Text Annotation
- that-7:that
- Lemma Annotation
- that-7:that

Finding parts of speech for that.

- it-8
- Text Annotation
- it-8:it
- Lemma Annotation
- it-8:it

Run WordNetMain

Event Log

All files are up-to-date (2 minutes ago)

50:22 CRLF: UTF-8

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SparkOpenIE\_WordNET\_LDA [sparkopenie\_wordnet\_lda] C:\Users\sreel\src\main\scala\wordnet > WordNetMain

```

WordNetMain [main()]
  30 import java.io.IOException;
  31
  32 public class WordNetMain {
  33     public static void main(String args[]) {
  34
  35         RiWordNet wordnet = new RiWordNet( "C:\\\\Users\\\\sreel\\\\IdeaProjects\\\\WordNet-3.0" );
  36         Properties props = new Properties();
  37         props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");
  38         StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
  39         String line = null;
  40         String fileName = "data/file.txt";
  41
  42
  43         try {
  44             // FileReader reads text files in the default encoding.
  45             FileReader fileReader =
  46                 new FileReader(fileName);
  47
  48             // Always wrap FileReader in BufferedReader.
  49             BufferedReader bufferedReader =
  50                 new BufferedReader(fileReader);
  
```

Finding parts of speech for Quinn.

- LRB--109
- Text Annotation
- LRB--109:-LRB-
- Lemma Annotation
- LRB--109:-LRB-

Finding parts of speech for -lrb-.

- Ireland-110
- Text Annotation
- Ireland-110:Ireland
- Lemma Annotation
- Ireland-110:Ireland

Run WordNetMain

Event Log

Compilation completed successfully in 3s 352ms (2 minutes ago)

40:41 CRLF: UTF-8

## 6.d Machine learning Techinques

## KMeans

```
Spark_MachineLearning - [C:\Users\pavan\Desktop\CS5560 - Tutorial 5 Source\Spark_MachineLearning] - [spark_machinelrning] - ...src\main\scala\kMeans\SparkKMeansMain.scala - IntelliJ IDEA 2017.1.4
```

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
```

```
Spark_MachineLearning src main scala kMeans SparkKMeansMain.scala
```

```
SparkKMeansMain
```

```
SparkKMeansMain [main(args: Array[String])]
```

```
def main(args: Array[String]): Unit = {  
    System.setProperty("hadoop.home.dir", "C:\\Users\\pavan\\Desktop\\hadoop-winutils-2.6.0")  
    val conf = new SparkConf().setAppName("kMeansExample").setMaster("local[*]").set("spark.driver.memory", "4g").set("spark.executor.memory", "4g")  
    val sc = new SparkContext(conf)  
    val inputPath=Seq("data/input")  
    Logger.getLogger().setLevel(Level.WARN)  
    val topic_output = new PrintStream("data/Results_KMeans.txt")  
    // Load documents, and prepare them for KMeans.  
    val preprocessStart = System.nanoTime()  
    val (corpusVector, data, vocabSize) = preprocess(sc, inputPath)  
    val actualCorpusSize = corpusVector.count()  
    val actualVocabSize = vocabSize  
    val preprocessElapsed = (System.nanoTime() - preprocessStart) / 1e9  
    println()  
    println("Corpus summary:")  
    println(s"\t Training set size: $actualCorpusSize documents")  
    println(s"\t Vocabulary size: $actualVocabSize terms")  
    println(s"\t Preprocessing time: $preprocessElapsed sec")  
    println()  
    topic_output.println()  
    topic_output.println("Corpus summary:")  
    topic_output.println(s"\t Training set size: $actualCorpusSize documents")  
    topic_output.println(s"\t Vocabulary size: $actualVocabSize terms")  
    topic_output.println(s"\t Preprocessing time: $preprocessElapsed sec")  
    topic_output.println()  
    // Run KMeans.  
    val startTime = System.nanoTime()  
    val k = 5  
    val numIterations=20  
    val corpusRM=corpusVector.map(_.r)  
    val model = KMeans.train(corpusRM, k, numIterations)  
    val elapsed = (System.nanoTime() - startTime) / 1e9
```

```
Compilation completed successfully in 23s 159ms (27 minutes ago)
```

```
29:36 CRLF+ UTF-8+ Event Log
```

```

17/07/09 23:31:15 INFO BlockManagerMaster: Registering BlockManagerBlockManagerId(driver, 192.168.1.170, 62442)
17/07/09 23:31:15 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.1.170:62442 with 1444.8 MB RAM, BlockManagerId(driver, 192.168.1.170, 62442)
17/07/09 23:31:15 INFO BlockManagerMaster: Registered BlockManagerBlockManagerId(driver, 192.168.1.170, 62442)
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger ... done [1.4 sec].

```

Corpus summary:

- Training set size: 2 documents
- Vocabulary size: 1097 terms
- Preprocessing time: 14.262089813 sec

17/07/09 23:31:30 WARN KMeans: The input data is not directly cached, which may hurt performance if its parent RDDs are also uncached.

17/07/09 23:31:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS

17/07/09 23:31:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

17/07/09 23:31:31 WARN LocalKMeans: KMeansPlusPlus initialization ran out of distinct points for centers. Using duplicate point for center k = 1.

17/07/09 23:31:31 WARN LocalKMeans: KMeansPlusPlus initialization ran out of distinct points for centers. Using duplicate point for center k = 2.

17/07/09 23:31:31 WARN LocalKMeans: KMeansPlusPlus initialization ran out of distinct points for centers. Using duplicate point for center k = 3.

17/07/09 23:31:31 WARN LocalKMeans: KMeansPlusPlus initialization ran out of distinct points for centers. Using duplicate point for center k = 4.

Finished training KMeans model. Summary:

- Training time: 1.36070016 sec
- 17/07/09 23:31:31 WARN KMeans: The input data was not directly cached, which may hurt performance if its parent RDDs are also uncached.

Process finished with exit code 0

## NaiveBias

```

private def run(params: Params) {
    System.setProperty("hadoop.home.dir", "C:\\\\Users\\\\pavan\\\\Desktop\\\\spark\\\\hadoop-winutils-2.6.0")
    val conf = new SparkConf().setAppName("NBExample with $params").setMaster("local[*]").set("spark.driver.memory", "4g").set("spark.executor.memory", "4g")
    val sc = new SparkContext(conf)

    Logger.getRootLogger.setLevel(Level.WARN)

    val topic_output = new PrintStream("data/NB_Results.txt")
    // Load documents, and prepare them for NB.
    val preprocessStart = System.nanoTime()
    val (inputVector, corpusData, vocabArrayCount) =
        preprocess(sc, params.input)

    var hm = new HashMap[String, Int]()
    val IMAGE_CATEGORIES = List("sci.crypt", "sci.electronics", "sci.med", "sci.space")
    var index = 0
    IMAGE_CATEGORIES.foreach(f => {
        hm += IMAGE_CATEGORIES(index) -> index
        index += 1
    })
    val mapping = sc.broadcast(hm)
    val data = corpusData.zip(inputVector)
    val featureVector = data.map(f => {
        val location_array = f._1._1.split("/")
        val class_name = location_array(location_array.length - 2)
        new LabeledPoint(hm.get(class_name).get.toDouble, f._2)
    })
    val splits = featureVector.randomSplit(Array(0.6, 0.4), seed = 11L)
    val training = splits(0)
    val test = splits(1)
}

```

## 6.e Question Answering for our dataset

After performing the NLP operations, we have taken the post processed dataset and we have separately stored the result of NER output like from the NER result we have the all person

related entities to one file and similarly we have done for every group and based on that we have generated answers for the questions we choose. The below screenshots will depict the same.

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor contains the following Scala code:

```

qa - [C:\Users\sree\IdeaProjects\Tutorial 4 Source Code\QA] - [week2] - ...src\main\scala\qa.scala - IntelliJ IDEA 2017.1.2
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
QA src main scala qa.scala
Project Structure
1 Project
src
  main
    java
      CoreNLP
      NLP
    scala
      qa
      SparkOpenIE
  target
  build.sbt
  sample.txt
External Libraries
Run: qa qa
he smoke in dorm room of brother
Maranisa writes As member of Choom Gang
Maranisa writes As member

he smoke in dorm room of brother
Maranisa writes As member of Choom Gang
Maranisa writes As member

he smoke in dorm room of brother
Maranisa writes As member of Choom Gang
Maranisa writes As member

he smoke in dorm room of brother
Maranisa writes As member of Choom Gang
Maranisa writes As member

he smoke in dorm room of brother
Maranisa writes As member of Choom Gang
Maranisa writes As member

```

The terminal window at the bottom shows the output of the Scala code, which prints several lines of text. The status bar at the bottom right indicates the date and time as 7/9/2017, 12:56 PM.

This screenshot is similar to the first one, showing the IntelliJ IDEA interface with the qa.scala code and its output. The code is identical to the previous screenshot. However, the terminal window now displays log messages from the StanfordCoreNLP library, indicating the loading of various annotators and models. The log output includes:

```

who is the member of choom gang
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator tokenize
17/07/09 12:54:29 INFO TokenizerAnnotator: TokenizerAnnotator: No tokenizer type provided. Defaulting to PTBTokenizer.
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator split
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator pos
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger ...
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator tokenize
done [0.5 sec].
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator split
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator lemma
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator pos
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator ner
17/07/09 12:54:29 INFO StanfordCoreNLP: Adding annotator lemma
Loading classifier from edu/stanford/nlp/models/ner/english.all.3class.distsim.crf.ser.gz ...

```

The status bar at the bottom right indicates the date and time as 7/9/2017, 12:54 PM.

QA - [C:\Users\sreel\IdeaProjects\Tutorial 4 Source Code(QA) - [week2] - \src\main\scala\qa.scala - IntelliJ] IDEA 2017.1.2

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

QA src main scala qa.scala

Project Structure

```

    project [week2-build] sources root
        src
            main
                java
                    CoreNLP
                    NLP
                scala
                    qa
                    SparkOpenIE
            target
                build.sbt
                sample.txt
        External Libraries

```

Run qa

Output

```

17/07/09 12:55:34 INFO StanfordCoreNLP: Adding annotator tokenize
Steve
Musi
Puna
Girl
Maraniss
Barry
Chomwagom
Bendix
Ray
Barack

```

Event Log

Compilation completed successfully in 25s 172ms (5 minutes ago)

757:1 LF+ UTF-8 7/9/2017 12:55 PM

QA - [C:\Users\sreel\IdeaProjects\Tutorial 4 Source Code(QA) - [week2] - \src\main\scala\qa.scala - IntelliJ] IDEA 2017.1.2

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

QA [week2] C:\Users\sreel\IdeaProjects\Tutorial 4 Source Code(QA) - [week2] - \src\main\scala\qa.scala

Project Structure

```

    project [week2-build] sources root
        src
            main
                java
                    CoreNLP
                    NLP
                scala
                    qa
                    SparkOpenIE
            target
                build.sbt
                sample.txt

```

Run qa

Output

```

who wrote about eating green eggs
17/07/09 12:44:21 INFO StanfordCoreNLP: Adding annotator tokenize
17/07/09 12:44:21 INFO TokenizerAnnotator: TokenizerAnnotator: No tokenizer type provided. Defaulting to PTBTokenizer.
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator split
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator pos
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger ... 17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator tokenize
done [0.8 sec].
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator asplit
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator lemma
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator pos
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator ner
17/07/09 12:44:22 INFO StanfordCoreNLP: Adding annotator lemma
Loading classifier from edu/stanford/nlp/models/ner/english.all.3class.distsim.crf.ser.gz ...

```

Event Log

All files are up-to-date (6 minutes ago)

561 LF+ UTF-8 7/9/2017 12:44 PM

```

17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator split
17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator pos
17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator lemma
17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator ner
17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator parse
17/07/09 12:44:52 INFO StanfordCoreNLP: Adding annotator dcoref

```

## 7. Project Management

### Programming Language Used:

We have collaborated various languages in the development of the project and in building the application. Some of them are,

- Java
- Scala
- Spark

### IDE Used:

Integrated development environments, helps in easy development of software with the facility of comprehensive integrated environment.

- IntelliJ
- PyCharm

## 7.a Contributors

- Jakkepalli, Rama Charan Pavan - **25%**
- Puthana, Sujitha - **25%**
- Yalamanchili, Sowmya - **25%**
- Nandanamudi, Sreelakshmi - **25%**



Name	Implementation	Documentation
Pavan	<b>Increment-1:</b> Basic Question Answer System  <b>Increment-2:</b> K-means Question answer using TF-IDF	<b>Increment-1:</b> Domain, Specific Dataset, Future Work  <b>Increment-2:</b> Added more description to document, related work, machine learning
Sujitha	<b>Increment-1:</b> TF-IDF  <b>Increment-2:</b> Classification Algorithm TF-IDF question and answer	<b>Increment-1:</b> Design workflow, Question Answer, Knowledge Graph  <b>Increment-2</b>

		Added more description to document, related work, design of Information Extraction
Sowmya	<b>Increment-1:</b> Core NLP <b>Increment-2</b> Wordnet Question answer using openIE	<b>Increment-1:</b> Project Motivation, Objective, Significance. <b>Increment-2</b> Design of Information Retrieval
Sreelakshmi	<b>Increment-1:</b> Named Entity Recognition <b>Increment-2</b> OpenIE Question answer using openIE	<b>Increment-1:</b> Contribution, Milestone, issues creation, Work Completed <b>Increment-2</b> Design of Machine Learning

## 7.b Zen-Hub Screenshots

For the first increment, we had issues regarding the working of the questions and answers section and generating the NLP output for the dataset we have chosen as the size of the dataset is larger.

The screenshot shows the GitHub Issues page for the repository "sujithaPuthana / TechCharmProject". The page displays 6 open issues and 7 closed issues. The issues are listed in descending order of creation date. Each issue card includes a title, a brief description, the number of comments, the user who created it, the time since creation, the milestone it's associated with, its status (e.g., In Progress, New Issues), and a small icon representing the issue type (e.g., bug, enhancement). A search bar at the top allows filtering by issue status (is:issue is:open). A "New issue" button is located in the top right corner of the main content area.

Issue Title	Description	Comments	User	Created	Milestone	Status
Developing the question and answering system for the dataset using these techniques	#13 opened 17 hours ago by sowmya5c6	1	sowmya5c6	17 hours ago	Milestone-2(Dev...)	In Progress
developing classification and clustering methods for the dataset	#12 opened 17 hours ago by sowmya5c6	2	sowmya5c6	17 hours ago	Milestone-2(Dev...)	In Progress
developing the openIE for the dataset	#11 opened 17 hours ago by sowmya5c6	2	sowmya5c6	17 hours ago	Milestone-2(Dev...)	New Issues
Developing wordnet for the dataset	#10 opened 17 hours ago by sowmya5c6	2	sowmya5c6	17 hours ago	Milestone-1(Dev...)	Review/QA
developing N-GRAM code	#9 opened 17 hours ago by sowmya5c6	3	sowmya5c6	17 hours ago	Milestone-1(Dev...)	In Progress
Developing the TF-IDF for the dataset	#8 opened 18 hours ago by sowmya5c6	3	sowmya5c6	18 hours ago	Milestone-1(Dev...)	Icebox

## Project Timeline, Members, and Task Responsibility

The issues that are registered and current one's which we are working are updated and can be viewed in GitHub repository. The below screenshot will show you the issues and their respective categorization's i.e. New issues, Icebox, Backlog, In Progress.

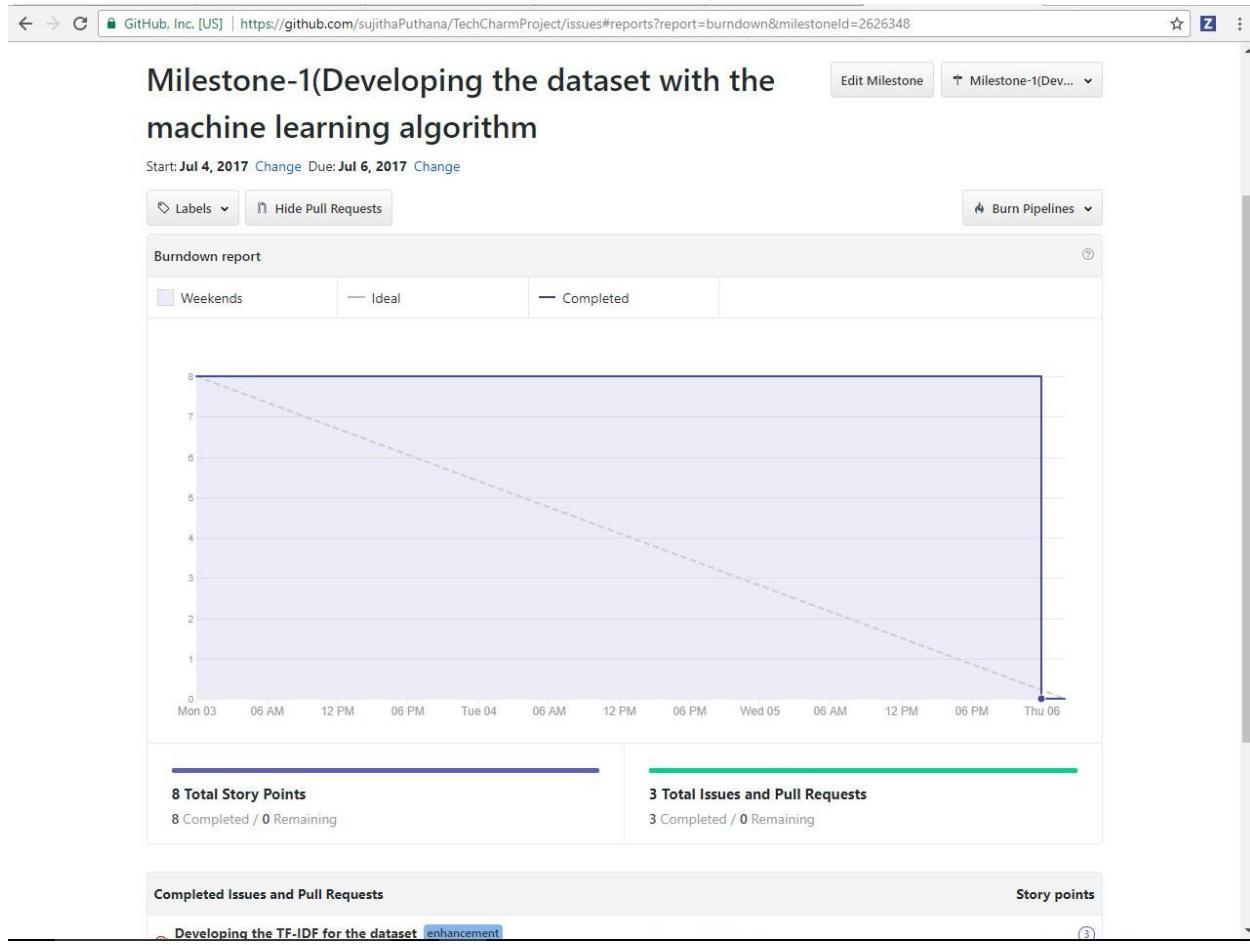
The screenshot shows the GitHub Boards interface for the repository 'sujithaPuthana / TechCharmProject'. The boards are organized into four columns:

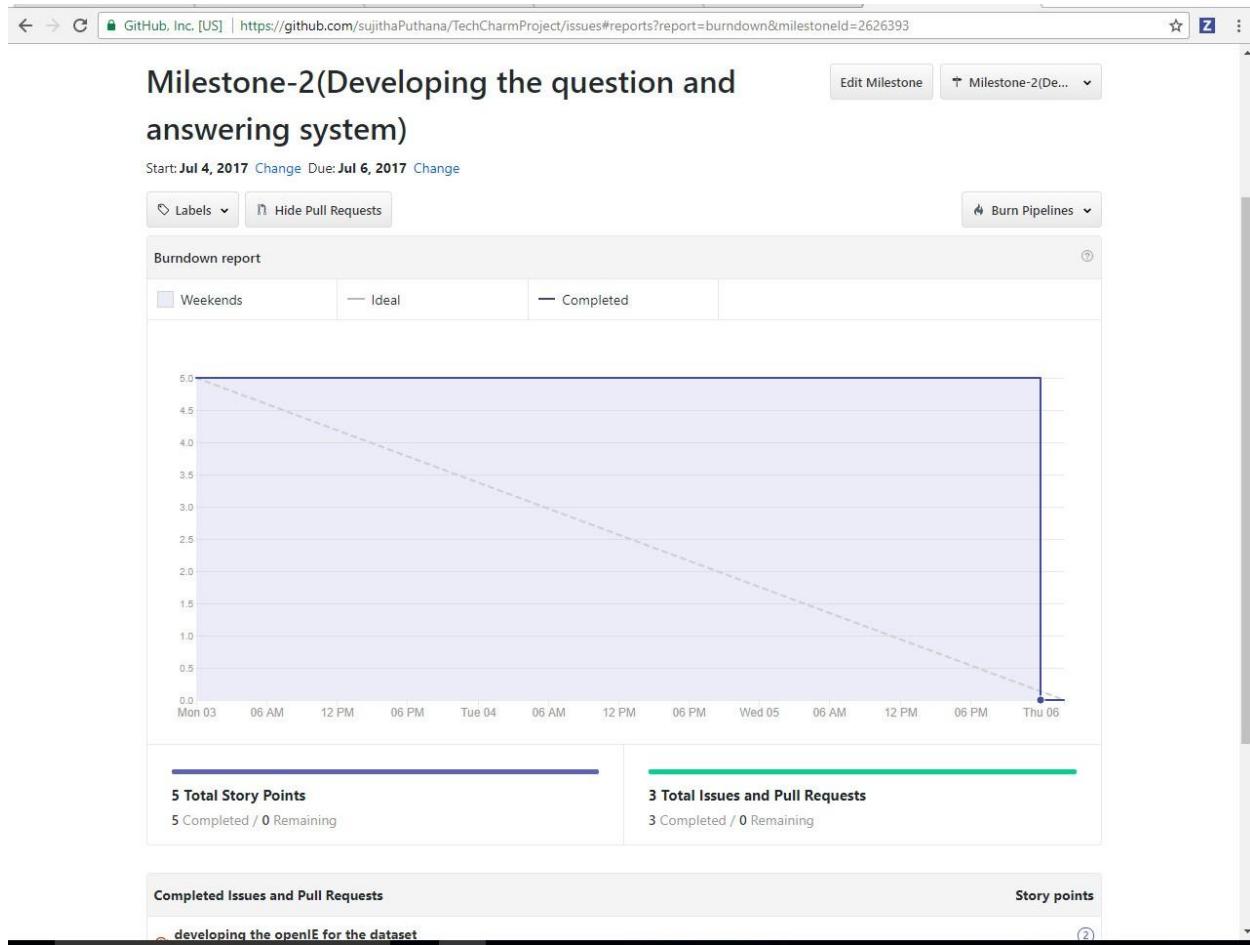
- New Issues**: 1 Issue - 2 Story Points. Contains one item: 'TechCharmProject #11 developing the openIE for the dataset'.
- Icebox**: 1 Issue - 3 Story Points. Contains two items: 'TechCharmProject #8 Developing the TF-IDF for the dataset' and 'TechCharmProject #10 Developing wordnet for t dataset'.
- Backlog**: 0 Issues - 0 Story Points. Empty.
- In Progress**: 3 Issues - 6 Story Points. Contains three items:
  - 'TechCharmProject #12 developing classification and clustering methods for the dataset' (status 2)
  - 'TechCharmProject #9 developing N-GRAM code' (status 3)
  - 'TechCharmProject #13 Developing the question and answering system for the dataset using these techniques' (status 1)
- Review/QA**: 1 Issue - 2 Story Points. Contains one item: 'TechCharmProject #10 Developing wordnet for t dataset'.

At the top, there are navigation tabs for Code, Issues (6), Pull requests (0), Boards (selected), Reports, Projects (0), Wiki, and Insights. Below the boards, there are filters for View, Repos (1/1), Labels, Milestones, Assignees, Epics, Releases, and a search bar. A 'New Issue' button is located in the top right corner of the board area.

## Burn-Down Chart:

Burn-Down chart is created for the above issues via Milestones in GitHub. Below is the screenshot for more information,





## GitHub Wiki Page

The GitHub wiki page URL for the screenshots and the process flow is updated in the following link

- <https://github.com/sujithaPuthana/TechcharmProject>

## Work Completed

The completed tasks in this increment are,

- Performed the TF-IDF and N-Gram approach for the dataset and embedded that in the question answering system.
- Implemented the OpenIE, wordnet, clustering and classification techniques for the dataset.
- Question and answer system using the OpenIE.

### **7.c Concerns**

- Faced an issue while implementing the k-means algorithm to the dataset.
- While integrating the OpenIE with the question answering system a bit issue we faced.

### **7.d Future Work**

- ❖ We need to implement the question and answer approach using the K-means integrated with the NLP operations.
- ❖ Need to integrate various techniques for better question and answering system.

### **Bibliography**

1. <https://blog.algorithmia.com/introduction-natural-language-processing-nlp/>
2. [https://en.wikipedia.org/wiki/Question\\_answering](https://en.wikipedia.org/wiki/Question_answering)
3. <https://nlp.stanford.edu/>