



School of Computing and Engineering

CS5560 Knowledge Discovery and Management

Dynamic Intelligent Q/A System

PROJECT 3 REPORT

SUMMER 2017

Team – 3

Nageswara, Rao Nandigam – 18

Chakilam, Revanth – 2

Syed, Moin – 28

Sankar, Pentyala – 22

Motivation:

We the Team Innovators 2.0 are in a search of data and knowledge. But Importantly we have a lot of difference between information, data, statistics and knowledge. When we speak about Information intruding and a semantic google search is more improvised and we can retrieve relevant information sitting in home. "Question Answering" is a dynamic way of retrieving Information, which learns knowledge. The main focus is particular in obtaining the respective documents but also particular in obtaining the respective response to query we post. Question and Answer system is capable of performing the NLP Processing, Information gathering, topic discovery, Machine Learning and Semantic Search. Question and answer system itself is the beauty of NLP and same instance have a bit of science in its essence. Question and Answer System is required every aspect, let be in the field of health and sciences, an intelligent learned system for children at schools, professional assistant etc. So, this is necessary in each case when we require some help from computer as well. It goes without saying that it is worth exploring the exciting field of question answering.

Objective:

Our Project critically deals with the building of typical model of information knowledge retrieval, named Question & Answering model. If suppose any given query asked in natural human language, Our question and answer system is implemented in such a way to extract the reality possible answer in the form of a pre-defined named-entity type, that is a human person, or may be an organization, a location, etc. Thus, we are connecting the question objects with live entities in a given radius is important in question and answer system. The projects main motto is to enhance the performance of the Question Answering system by using the knowledge and data from the natural grouping of word in the document files.

Significance:

We are constructing a knowledge graph such a way to build the question and answering system to deliver answers very effectively. To give better the results from the system designed we are applying different techniques such as NLP operations, Information retrieval, topic discovery and knowledge discovery.

Q/A System:

For this project, we have taken the Data set from BBC sports concentrating on the sport Cricket. From this Data set we try to construct knowledge graph and making system dynamic to answer all possible questions on sports questions.

Related Work:

Knowledge graph refinement:

Firstly, Knowledge graph is the one of the advanced and trending concepts today. By making use of it, we can build dynamic systems which can precisely answer our questions contrary to the existing search engines which gives us information and a set of related links which makes the user to spend more time on the web for the obtaining the information one was looking for.

Almost as every technology with the usage it needs refinement the same applies to the knowledge graph too. In this paper, the technologies used to evaluate knowledge graph are Partial gold standard, Silver standard, Retrospective evaluation. Each of this technology is a trade-off between reliability and cost. Completeness aims at increasing the coverage of the knowledge graph & various models are used to evaluate completeness and correctness which cannot be both achieved at the same time. This paper is based on the survey results.

Knowledge vault:

The knowledge Vault is a database of what google would call the facts that has been scraped from across the entire web. Unlike the traditional ranking system which relies on the incoming links and number of those links to determine the quality of a source, the knowledge vault allows google to develop this system whereby they count the number of true or false facts, where a true or false is determined by how often that appears on the rest of the web. So, its kind of like querying our collective consciousness to ask us what's true or not.

Controversial facts are going to trip up algorithms like this and create even more controversy about what should be surfaced at the top and what should be lowered down depending on sort of objective opinions. The knowledge vault is mainly of extracting triplets and giving the score to them and it has advantages of to automatically crawl, index and organize information across the web.

Knowledge base Construction:

We are at that point of life where Data is growing day by day. There is a lot of unstructured data in various organizations and medical fields. A need for dark data extraction & KBC - Knowledge base construction is increasing every moment. So Deep Dive is a solution to establish a SQL database with data from various unknown sources like images, emails, static web pages etc. In every Q/A system its been a long standing problem and Deep dive provides a better solution to it. Currently in this project, Deep dive gives a choice to domain experts to design each ones own KBC systems. Gibbs sampling is used in the processing of data which has high accuracy. The output is the probability of the words which are needed to be present in the data set.

Semantic data integration for knowledge graph construction:

The introduction of Web documents in to a Webservices and data has definitely reulted in the increment availability of data from each type of domain networks. A new Semantic-Data-Integration approach called "FUHSEN" has been invented which this system can exploit use the key words and structured strengths of webdata sources and can generate quality know;edge graphs by amalgaming the data collected from

various data sources. This system FUHSEN depends on two things. i) RESOURCE DESCRIPTION FRAMEWORK (RDF) : for semantically describing the collection of entities

ii) Semantic similarity measures among the collected entities and establishing the relation between them.

The results of the Fuhsen integration system are evaluated from DBpedia knowledge base. So in the current project, using the Fuhsen data integration technique, we can accurately integrate the various similar data entities semantically and transform them into quality knowledge graphs.

Knowledge Base Construction from Richly Formatted Data:

In this framework, the entities, relation between them and the attributes are related via tabular, textual, structured and visualized expression. This FONDUER introduces a unique model for KNOWLEDGE BASE CONSTRUCTION built on a unified data representation. This is a new KBC system for the RFIE - richly formatted information data extraction and it also uses human loop algorithm in order to train machine learning systems. In our project, we can use FONDUER to ease the burden of traditional approaches. This model in addition with data programming allows the end users to supervise and help to implement Knowledge Base Construction process over the richly formatted data.

Clustering Approach :

The blooming of World Wide Web 2.0 Community Question Answering such as Yahoo! Answers, WikiAnswer etc., have emerged as other approaches for knowledge and information extraction and acquisition. Over a period of time, a huge number of question and answers with good quality has been devoted by human intelligence have been achieved as a knowledge base. Besides the search engines, which return long queries, finding in the CQA services can achieve proper answers to the questions by automatically finding similar questions that has been answered by different users. It improves the efficiency of information retrieval. However, if given a question, finding the similar answered query is a difficult task. The main task is the word mismatch between question query and candidate query for retrieval query.

Knowledge Base Completion:

In recent years, huge amounts of world knowledge have been in publicly available knowledge bases, such as Freebase and YAGO. Having huge database they look incomplete. Here we propose a way to leverage old Web based search question answering technology to fulfil the gaps in knowledge bases. In general, for every entity attribute learn the best set of queries to ask, from that the answer snippets returned will contain the correct value for the attribute. The system learns and add disambiguating terms, in order to make it more likely that the search results contain correct mentioning. The system will also learn how many queries to ask for each attribute, since in many situations, asking many can reduce accuracy, predictions for possible values for each attribute. Lastly we evaluate our system and show that it is able to extract a large number of facts with high accuracy.

Predictive Analysis :

An expert in domain can analyze huge data to make meaningful predictions from the data. For example, by scanning through the research papers and patent records, a domain expert can predict the geographic location and time , where and when the technology will become useful. However, this is an tedious task. This paper presents an system that acquaints the heterogeneous data into a knowledge graph in the RDF (Resource Description Framework) format using ontology method. Hence, the user can easily query the knowledge graph to get the required data from the predictive analysis tools. The system extracts the detailed data from public sources including research papers and patent records. Next, the system an ontology-based data deiled method to generate knowledge in the RDF format to enable quickly between machine learning for predictive analytic tasks.

ConceptNet 5.5:

Machine learning language can be improved by providing with knowledge and resources of external information. ConceptNet is particularly suited to be used with modern NLP techniques such as word embeddings. It is a knowledge graph that mainly connects words and phrases of natural language with labeled edges. Its knowledge is collected from many sources that include expert created resources and games with a purpose. It is designed to represent the basic knowledge involved in processing the natural languag by improving its applications by allowing the application to understand the synonyms behind the words that we use. When this approach is combined with word embeddings which is acquired from semantics such as word2vec, it provides application that they would not require from semantics alone, nor from the resources such as DBPedia and WordNet.

Datasets:

BBC Sports domain:

We have taken BBC sports dataset as one of our dataset for the design of Question and Answering system. This dataset consists of all kind of sports which includes athletic, cricket, football ,rugby and tennis.

We have mainly focused on the cricket dataset for designing and extracting information using Natural language processing, word2vec, N-gram and TF-IDf techniques

<http://mlg.ucd.ie/datasets/bbc.html>

BBC News:

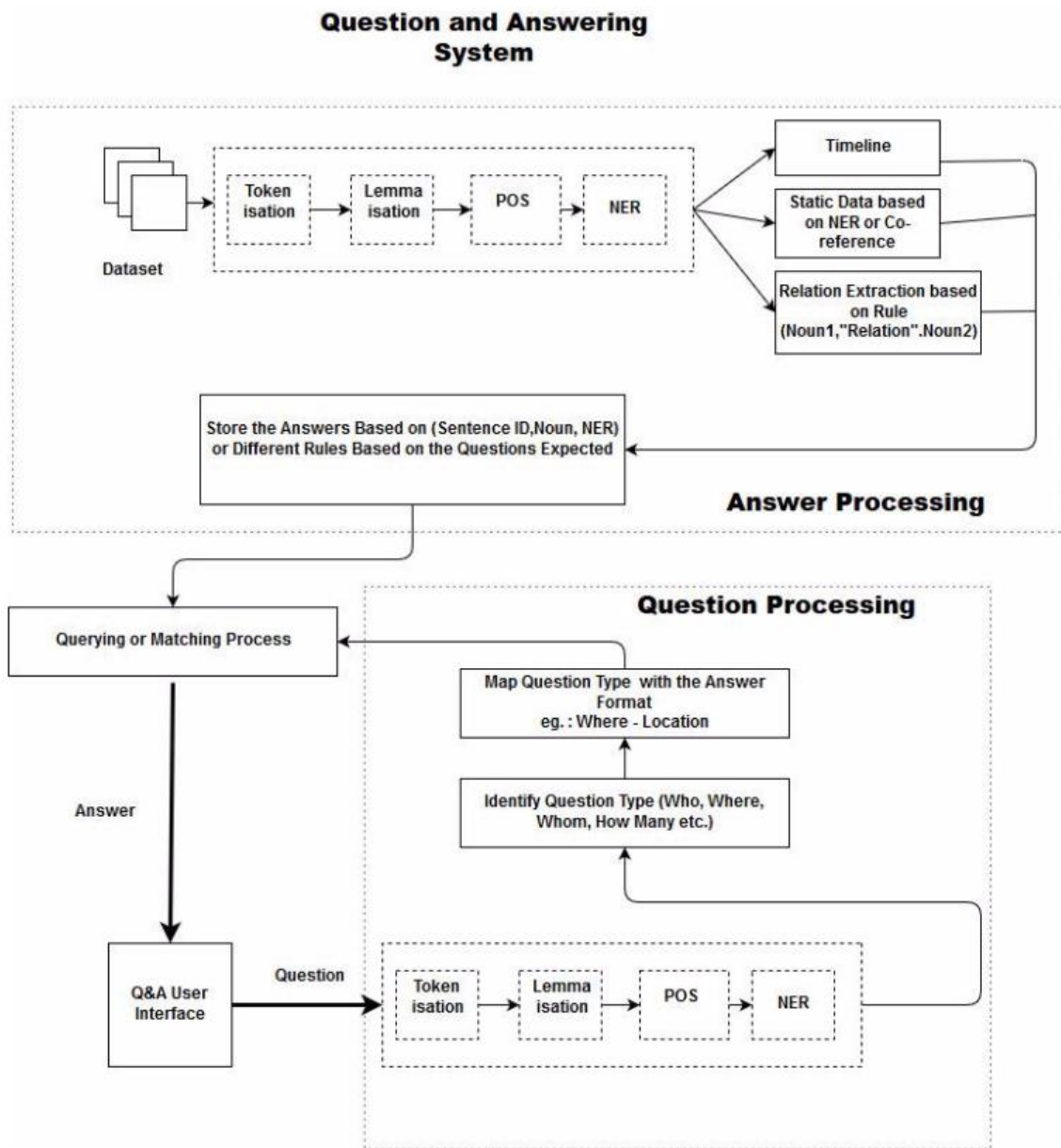
We have taken BBC news dataset as one of second dataset for the design of Question and Answering system. This dataset consists of all kind of news which includes business, entertainment, politics, sports and technology.

We have mainly focused ont the sports dataset for designing and extracting information using Natural language processing, word2vec, N-gram and TF-IDf techniques.

<http://mlg.ucd.ie/datasets/bbc.html>

Design:

Workflow:



NLP:

Open NLP is a toolkit available in variety of programming languages which supports most of the Natural Language Processing tasks like tokenization, pos tagging, chunking, name recognition, sentence segmentation. These tasks are generally required to produce services that require more advanced text processing.



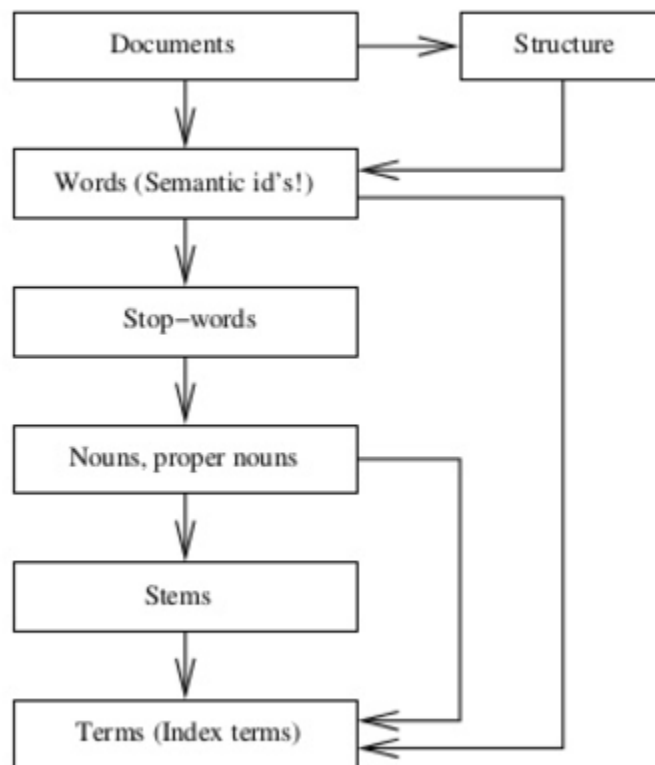
Information Retrieval:

Skip-gram negative sampling (or Word2Vec) is an algorithm based on a shallow neural network which aims to learn a word embedding. It is highly efficient, as it avoids dense matrix multiplication and does not require the full term co-occurrence matrix. Given some target word w_t , the intermediate goal is to train the neural network to predict the words in the c -neighborhood of w_t : $w_t - c, \dots, w_t - 1, w_t + 1, \dots, w_t + c$. First, the word is directly associated to its respective vector, which is used as input for a (multinomial) logistic regression to predict the words in the c -neighborhood. Then, the weights for the logistic regression are adjusted, as well as the vector itself (by back-propagation). The Word2Vec

algorithm employs negative sampling: additional k noise words which do not appear in the c -neighborhood are introduced as possible outputs, for which the desired output is known to be false. Thus, the model does not reduce the weights to all other vocabulary words but only to those sampled k noise words. When these noise words appear in a similar context as w_t , the model gets more and more fine-grained over the training epoch.

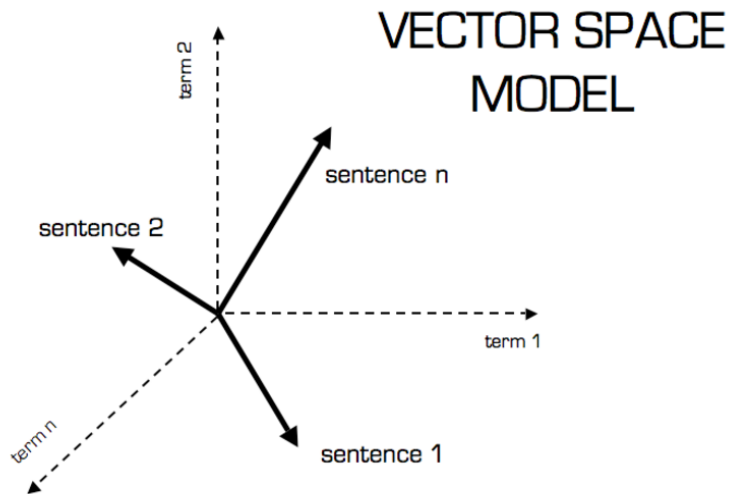
➤ TF – IDF

Term Frequency Inverse document frequency produces an output with the word and term weight. It gives how importance a word is to a document collection/corpus.



➤ W2V

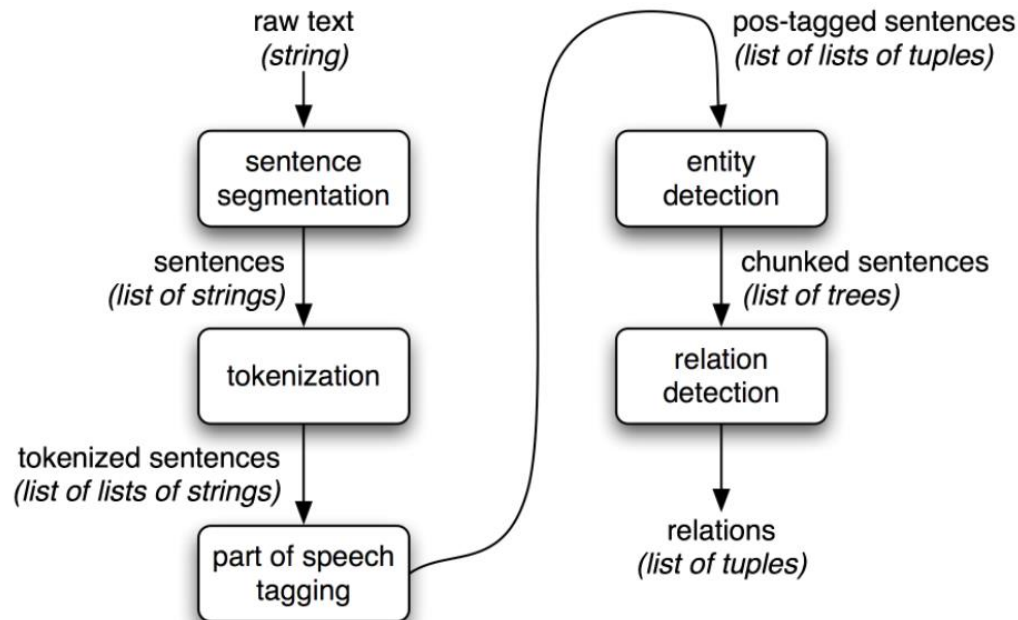
The input to this model is a text corpus. It is a 2-layer neural network that processes the text data. The output is a series of vectors which are feature vectors for the words in the input corpus.



Information Extraction:

Word net is a software package developed at Princeton University providing a lexical data base of English words meaning and categorization. It provides the means to determine the categorization of a word which is used in answer ranking phase.

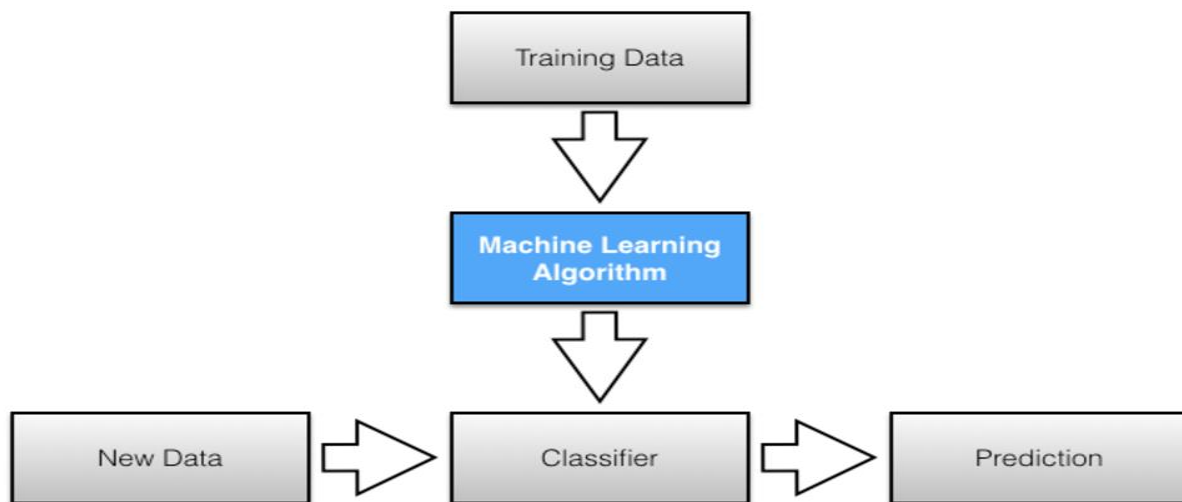
IE is a process of extracting a structured data from an unstructured data through a means of Natural Language Processing.



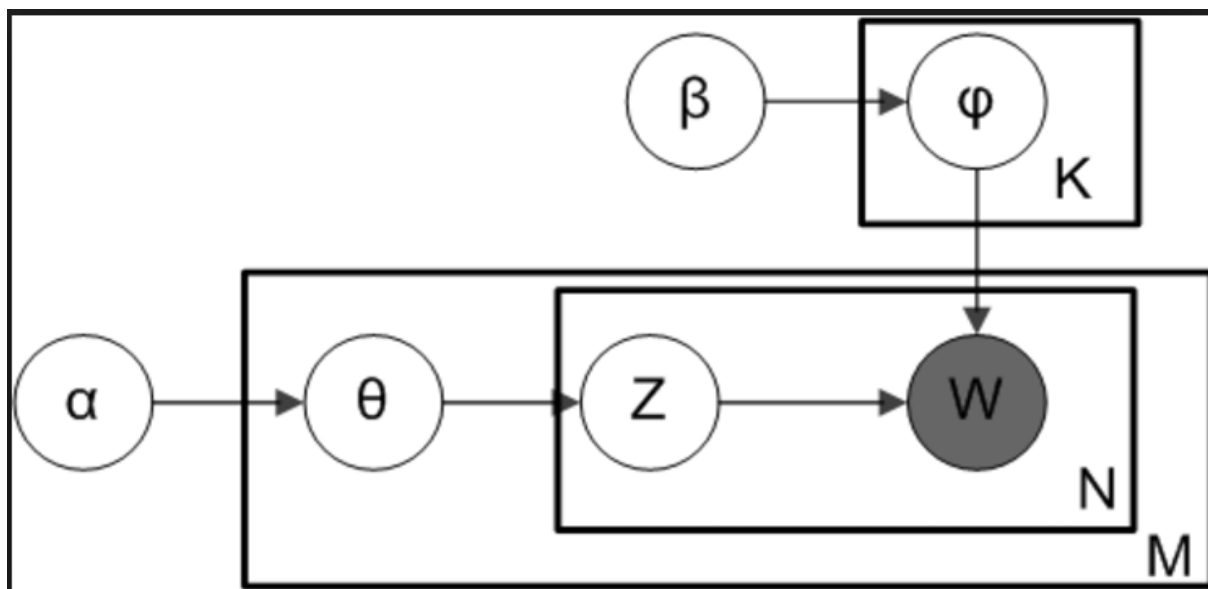
Machine Learning:

Classification is the result of supervised learning which means that there is a known label that you want the system to generate. For example, if you built a fruit classifier, it would say “this is an orange, this is an apple”, based on you showing it examples of apples and oranges.

Clustering is the result of unsupervised learning which means that you’ve seen lots of examples, but don’t have labels. In this case, the clustering might return with “fruits with soft skin and lots of dimples”, “fruits with shiny hard skin” and “elongated yellow fruits” based not simply showing lots of fruit to the system, but not identifying the names of different types of fruit.

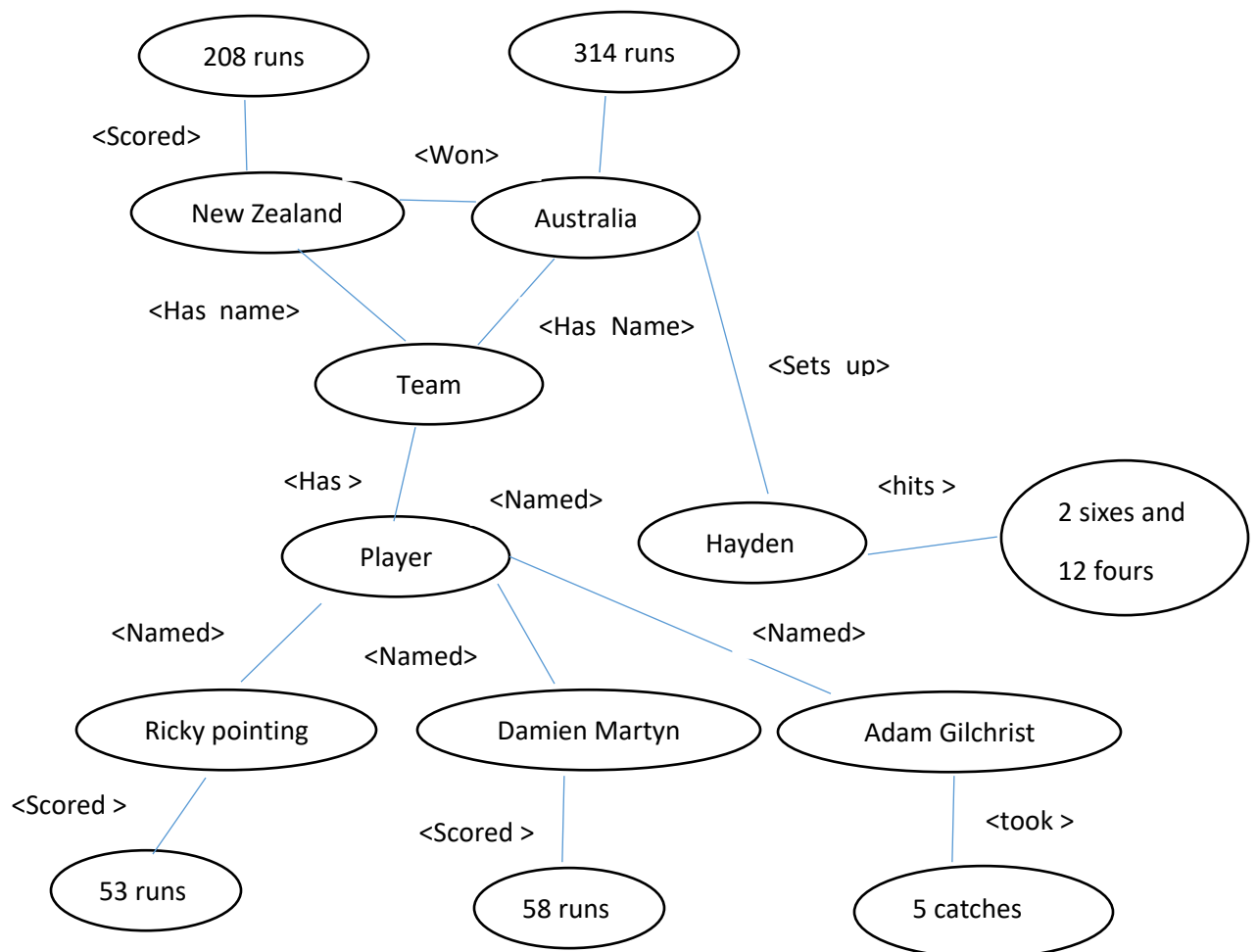


LDA:



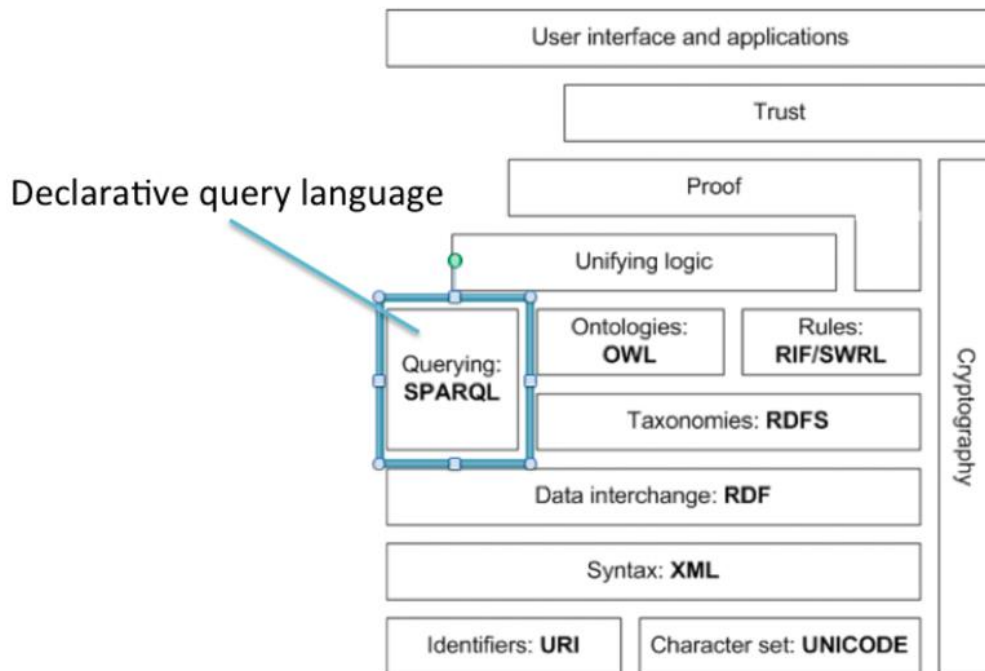
Knowledge Graph:

A knowledge graph is mainly organized as a graph, which is not always true of knowledge bases. The primary benefits of a graph are that relationships in the data are first-class citizens, you can easily connect new data items as they are injected into the data pool, and, finally, you can easily traverse links to discover how remote parts of a domain relate to each other there's a huge value in linking information. A graph is one of the most flexible formal data structures, so you can easily map other data formats to graphs using generic tools and pipelines.



SPARQL:

It is the extension of spark query language for operating not only on Resource Descriptive Framework but also on the documents/corpus data on arbitrary format.



Questions and Answers:

1) Who all played between nez vs aus

Answer:

S P Fleming, N J Astle, M S Sinclair, J Wilson, C D McMillan, H J H Marshall, C L Cairns, B B McCullum, K D Mills, D L Vettori, DTuffey

M L Hayden, A C Gilchrist, R T Ponting, D R Martyn, A Symonds, M J Clarke, M E K Hussey, G B Hogg, B Lee, J Gillespie, G D McGrath

2) When nez vs Aus match happened

Answer:

March 1993

3) Where nez vs aus match was held

Answer:

Jade Stadium

Implementation, Results and Evaluation

NLP Process:

- 1) Doing NLP processing of tokenizing, lemmatization and extracting named entity relations and storing it in HashMap.

```
public String lemm(String data) {  
  
    Properties prop = new Properties();  
  
    StringBuilder res = new StringBuilder();  
    prop.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");  
    StanfordCoreNLP pipeline = new StanfordCoreNLP(prop);  
    Annotation doc = new Annotation(data);  
  
    pipeline.annotate(doc);  
  
    List<CoreMap> sents = doc.get(CoreAnnotations.SentencesAnnotation.class);  
    for (CoreMap sentence : sents) {  
        for (CoreLabel token1 : sentence.get(CoreAnnotations.TokensAnnotation.class)) {  
            String lemma = token1.get(CoreAnnotations.LemmaAnnotation.class);  
            res.append(lemma + " ");  
        }  
    }  
    return res.toString();  
}
```

```

public void nerealtions(String d)
{
    Properties p=new Properties();
    p.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");
    StanfordCoreNLP pipeline = new StanfordCoreNLP(p);
    Annotation a = new Annotation(d);
    pipeline.annotate(a);
    List<CoreMap> lines=a.get(CoreAnnotations.SentencesAnnotation.class);
    for(CoreMap line:lines){
        for(CoreLabel t:line.get(CoreAnnotations.TokensAnnotation.class))
        {
            String ne=t.get(CoreAnnotations.NamedEntityTagAnnotation.class);
            String w=t.get(CoreAnnotations.TextAnnotation.class);
            h.put(ne, w);
        }
    }
}

public String ret(String data,String ans)
{
    nerealtions(data);
    Collection<String> c=h.get(ans);

    StringBuilder b=new StringBuilder();
    for(String ele:c)
    {
        b.append(ele+" ");
    }

    return b.toString();
}

```

Information Retrieval:

TF-IDF:

```
//Reading the Text File
val documents = sc.textFile("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcspot\\cricket\\001.txt")

//Getting the Lemmatised form of the words in TextFile
val documentseq = documents.map(f => {
  val lemmatised = CoreNLP.returnLemma(f)
  val splitString = lemmatised.split(" ")
  splitString.toSeq
  // val x=NGRAM.getNGrams(f,2).map(f=>{f.mkString(" ")})
  // x.toSeq
})

// val lemmatised = CoreNLP.returnLemma(f)
//val splitString = lemmatised.split(" ")
//val x=NGRAM.getNGrams(f,2).map(f=>{f.mkString(" ")})
//Creating an object of HashingTF Class
val hashingTF = new HashingTF()

//Creating Term Frequency of the document
val tf = hashingTF.transform(documentseq)
tf.cache()

val idf = new IDF().fit(tf)

//Creating Inverse Document Frequency
val tfidf = idf.transform(tf)
```

```

val tfidfindex = tfidf.flatMap(f => {
  val ff: Array[String] = f.toString.replace("[", ";").split(";")
  val indices = ff(1).replace("]", "").replace(")", "").split(",")
  indices
})

tfidf.foreach(f => println(f))

val tfidfData = tfidfindex.zip(tfidfvalues)

var hm = new HashMap[String, Double]

tfidfData.collect().foreach(f => {
  hm += f._1 -> f._2.toDouble
})

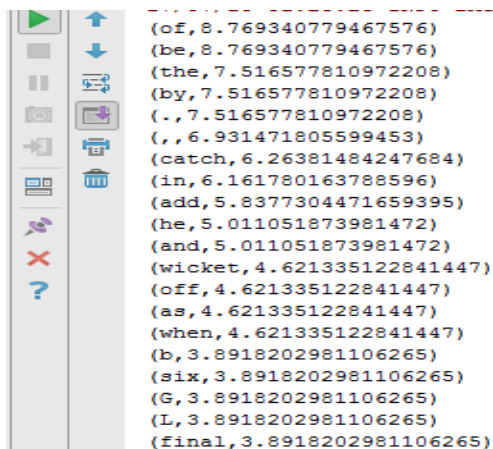
val mapp = sc.broadcast(hm)

val documentData = documentsseq.flatMap(_.toList)
val dd = documentData.map(f => {
  val i = hashingTF.indexOf(f)
  val h = mapp.value
  (f, h(i.toString))
})

val dd1 = dd.distinct().sortBy(_._2, false)
dd1.take(20).foreach(f => {
  println(f)
})

```

Output:



```

(off,8.769340779467576)
(be,8.769340779467576)
(the,7.516577810972208)
(by,7.516577810972208)
(.,7.516577810972208)
(,,6.931471805599453)
(catch,6.26381484247684)
(in,6.161780163788596)
(add,5.8377304471659395)
(he,5.011051873981472)
(and,5.011051873981472)
(wicket,4.621335122841447)
(off,4.621335122841447)
(as,4.621335122841447)
(when,4.621335122841447)
(b,3.8918202981106265)
(six,3.8918202981106265)
(G,3.8918202981106265)
(L,3.8918202981106265)
(final,3.8918202981106265)

```


Word2Vec:

```
val input = sc.textFile("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcspot\\cricket\\001.txt").map(line => line.split(" ").toSeq)

val modelFolder = new File("myModel")

if (modelFolder.exists()) {
  val sameModel = Word2VecModel.load(sc, "myModel")
  val synonyms = sameModel.findSynonyms("ball", 10)

  for ((synonym, cosineSimilarity) <- synonyms) {
    println(s"$synonym $cosineSimilarity")
  }
} else {
  val word2vec = new Word2Vec().setVectorSize(1000).setMinCount(1)

  val model = word2vec.fit(input)

  val synonyms = model.findSynonyms("ball", 10)

  for ((synonym, cosineSimilarity) <- synonyms) {
    println(s"$synonym $cosineSimilarity")
  }

  model.getVectors.foreach(f => println(f._1 + ":" + f._2.length))

  // Save and load model
  model.save(sc, "myModel")
}
```

Output:

```
turn-round 8.112398718752908E-4
fifty 7.927529137858012E-4
only 7.463412338428283E-4
pairing 7.437037947049131E-4
13 6.681809740532757E-4
22, 6.437686232189962E-4
Gillespie 6.24310004943662E-4
over, 6.023842918913169E-4
able 5.738480799238281E-4
Mike 5.419502444764308E-4
```

Information extraction:

OpenIE:

```
Document doc = new Document(sentence);
String lemma="";
for (Sentence sent : doc.sentences()) { // Will iterate over two sentences

    Collection<Quadruple<String, String, String, Double>> l=sent.openie();

    lemma+= l.toString();

    // System.out.println(lemma);
}

return lemma;
```

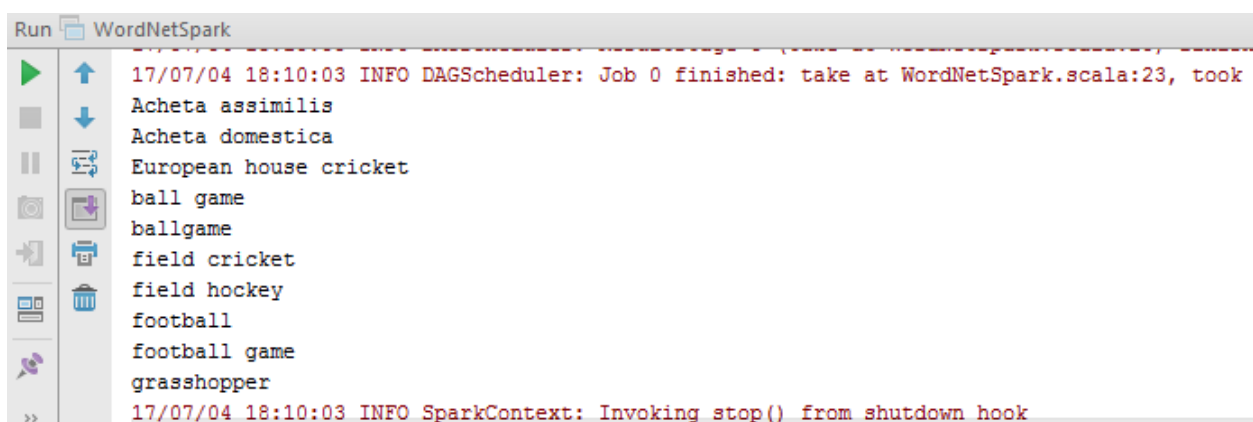
Output:

```
[(Ricky Ponting,provided,Damien Martyn,1.0), (Ricky Ponting,provided,main support,1.0), (Ricky Ponting,provided,support,1.0)][(Adam Gilchrist,taking,five catches,1.0), (New
[(Gilchrist,was caught from,ball,1.0), (Gilchrist,was,caught behind off Daryl Tuffey from ball,1.0), (Gilchrist,was,caught behind Daryl Tuffey from ball,1.0), (Gilchrist,wa
[(new ball pairing,made,short work of New Zealand 's top order,1.0), (ball pairing,made,short work,1.0), (new ball pairing,claiming,two wickets,1.0), (ball pairing,made,sho
```

Wordnet:

```
object WordNetSpark {  
  def main(args: Array[String]): Unit = {  
    System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")  
    val conf = new SparkConf().setAppName("WordNetSpark").setMaster("local[*]").set("spark.driver.memory", "4g")  
    val sc = new SparkContext(conf)  
  
    val data=sc.textFile("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcspot\\cricket\\001.txt")  
  
    val dd=data.map(f=>{  
      val wordnet = new RiWordNet("E:\\UMKC\\Sum_May\\KDM\\WordNet-3.0")  
      val farr=f.split(" ")  
      getSynonyms(wordnet,"cricket")  
    })  
    dd.take(1).foreach(f=>println(f.mkString("\n")))  
  }  
  def getSynonyms(wordnet:RiWordNet,word:String): Array[String] =(  
    println(word)  
    val pos=wordnet.getPos(word)  
    println(pos.mkString(" "))  
    val syn=wordnet.getAllSynonyms(word, pos(0), 10)  
    syn  
  )  
}
```

Output:



```
Run WordNetSpark  
17/07/04 18:10:03 INFO DAGScheduler: Job 0 finished: take at WordNetSpark.scala:23, took  
Acheta assimilis  
Acheta domestica  
European house cricket  
ball game  
ballgame  
field cricket  
field hockey  
football  
football game  
grasshopper  
17/07/04 18:10:03 INFO SparkContext: Invoking stop() from shutdown hook
```

Machine Learning:

Clustering:

```
object SparkKMeansMain {  
  def main(args: Array[String]): Unit = {  
    System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")  
    val conf = new SparkConf().setAppName(s"KMeansExample").setMaster("local[*]").set("spark.driver.memory",  
    val sc = new SparkContext(conf)  
  
    val inputPath=Seq("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcspot\\*")  
    Logger.getRootLogger.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(s"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()  
  }  
}
```

```

// Run KMeans.
val startTime = System.nanoTime()

val k= 5
val numIterations=20

val corpusKM=corpusVector.map(_._2)
val model = KMeans.train(corpusKM, k, numIterations)

val elapsed = (System.nanoTime() - startTime) / 1e9

println(s"Finished training KMeans model. Summary:")
println(s"\t Training time: $elapsed sec")

topic_output.println(s"Finished training KMeans model. Summary:")
topic_output.println(s"\t Training time: $elapsed sec")

val predictions = model.predict(corpusKM)

val error = model.computeCost(corpusKM)
val results = data.zip(predictions)
val resultsA = results.collect()
var hm = new HashMap[Int, Int]
resultsA.foreach(f => {
  topic_output.println(f._1._1 + ";" + f._2)
  if (hm.contains(f._2)) {
    var v = hm.get(f._2).get
    v = v + 1
    hm += f._2 -> v
  }
})

```

Output:

```
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/023.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/024.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/025.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/026.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/027.txt;3
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/028.txt;4
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/029.txt;4
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/030.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/031.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/032.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/033.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/034.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/035.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/036.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/037.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/038.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/039.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/040.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/041.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/042.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/043.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/044.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/045.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/046.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/047.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/048.txt;3
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/049.txt;3
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/050.txt;3
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/051.txt;3
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/052.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/053.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/054.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/055.txt;1
file:/E:/UMKC/Sum_May/KDM/week1/bbcsport/rugby/056.txt;3
```

Classification:

Decision tree:

```
private def run(params: Params) {
  System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")
  val conf = new SparkConf().setAppName(s"KMeansExample with $params").setMaster("local[*]").set("spark.driver.extraJavaOptions", "-Xmx1g")
  val sc = new SparkContext(conf)

  Logger.getRootLogger.setLevel(Level.WARN)

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

  val hm = new HashMap[String, Int]()
  val IMAGE_CATEGORIES = List("athletics", "cricket", "football", "rugby", "tennis")
  var index = 0
  IMAGE_CATEGORIES.foreach(f => {
    hm += IMAGE_CATEGORIES(index) -> index
    index += 1
  })

  val splits = featureVector.randomSplit(Array(0.6, 0.4), seed = 11L)
  val training = splits(0)
  val test = splits(1)
  val numClasses = IMAGE_CATEGORIES.length
  val categoricalFeaturesInfo = Map[Int, Int]()
  val impurity = "gini"
  val maxDepth = 5
  val maxBins = 32

  val model = DecisionTree.trainClassifier(training, numClasses, categoricalFeaturesInfo,
    impurity, maxDepth, maxBins)

  val predictionAndLabel = test.map(p => (model.predict(p.features), p.label))

  val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

  val metrics = new MulticlassMetrics(predictionAndLabel)

  // Confusion matrix
  topic_output.println("Confusion matrix:")
  topic_output.println(metrics.confusionMatrix)

  topic_output.println("Accuracy: " + accuracy)

  sc.stop()
}
```

Output:

1	Confusion matrix:
2	25.0 0.0 6.0 3.0 0.0
3	1.0 17.0 16.0 7.0 6.0
4	5.0 4.0 87.0 13.0 2.0
5	3.0 3.0 22.0 31.0 0.0
6	0.0 1.0 10.0 1.0 25.0
7	Accuracy: 0.6423611111111112
8	

RandomForest:

```
private def run(params: Params) {
  System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")
  val conf = new SparkConf().setAppName(s"RFExample with $params").setMaster("local[*]").set("spark.driver.maxResultSize", "1g")
  val sc = new SparkContext(conf)

  Logger.getRootLogger.setLevel(Level.WARN)

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

  var hm = new HashMap[String, Int]()
  val IMAGE_CATEGORIES = List("athletics", "cricket", "football", "rugby", "tennis")
  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)
val numClasses = IMAGE_CATEGORIES.length
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val featureSubSet = "auto"
val maxDepth = 5
val maxBins = 32
val numTrees = 10

val model = RandomForest.trainClassifier(training, numClasses, categoricalFeaturesInfo, numTrees,

val predictionAndLabel = test.map(p => (model.predict(p.features), p.label))

val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

val metrics = new MulticlassMetrics(predictionAndLabel)

// Confusion matrix
topic_output.println("Confusion matrix:")
topic_output.println(metrics.confusionMatrix)

topic_output.println("Accuracy: " + accuracy)

sc.stop()
}

```

Output:

1	Confusion matrix:				
2	14.0	2.0	16.0	2.0	0.0
3	0.0	19.0	27.0	1.0	0.0
4	1.0	0.0	110.0	0.0	0.0
5	0.0	1.0	45.0	13.0	0.0
6	0.0	1.0	31.0	1.0	4.0
7	Accuracy: 0.5555555555555556				
8					

Naïve Bayes:

```
private def run(params: Params) {
  System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")
  val conf = new SparkConf().setAppName(s"NBExample with $params").setMaster("local[*]").set("spark.driver.extraJavaOptions", "-Xmx1024m")
  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("athletics", "cricket", "football", "rugby", "tennis")
  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)
```

```

val model = NaiveBayes.train(training, lambda = 1.0, modelType = "multinomial")

val predictionAndLabel = test.map(p => (model.predict(p.features), p.label))

val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

val metrics = new MulticlassMetrics(predictionAndLabel)

// Confusion matrix
topic_output.println("Confusion matrix:")
topic_output.println(metrics.confusionMatrix)

topic_output.println("Accuracy: " + accuracy)

sc.stop()
}
//++

```

Output:

```

1 Confusion matrix:
2 34.0  0.0  0.0  0.0  0.0
3 0.0  47.0  0.0  0.0  0.0
4 0.0  0.0 108.0  3.0  0.0
5 1.0  1.0  3.0  54.0  0.0
6 0.0  0.0  1.0  1.0 35.0
7 Accuracy: 0.9652777777777778
8

```

Knowledge Graph (KG):

```
val hashSet1: HashSet[String] = HashSet(" ")
val openie = input.map(coreNLP.openIE(_)).map(f=>f.split(":"))
openie.collect().foreach(f=>
{
  for (x <- f) {
    if(!x.trim.isEmpty)
    {
      val openiearray = x.split(",")
      val subject = openiearray(0).trim.replace(" ", "_")
      val predicate = openiearray(1).trim.replace(" ", "_")
      val object1 = openiearray(2).trim.replace(" ", "_")

      val subner = coreNLP.NER(subject).mkString(" ")
      val objner = coreNLP.NER(object1).mkString(" ")
      if (!subner.isEmpty && !objner.isEmpty)
      {
        val x = predicate.toString + "," + subner.split(" ")(0) + "," + objner.split(" ")(0) + "," +
        objprop_outfile.println(x)
        val triplet = subject+","+predicate+","+object1+", "+"Obj"
        triplets_outfile.println(triplet)
        predlist_outfile.println(predicate)
      }
      if(!subner.isEmpty && objner.isEmpty)
      {
        if(!subner.isEmpty && objner.isEmpty)
        {
          val triplet = subject+","+predicate+","+object1+", "+"Data"
          triplets_outfile.println(triplet)
          predlist_outfile.println(predicate)
          val dataprop = predicate + "," + subner.split(" ")(0) + "," + "String"
          hashSet1 += dataprop
          //Dataprop_outfile.println(dataprop)
        }
      }
    }
  }
})
for(a <- hashSet1)
{
  Dataprop_outfile.println(a)
}
objprop_outfile.close()
classes_outfile.close()
Individuals_outfile.close()
triplets_outfile.close()

//Ontology Creation
val ONTOLOGYURI="http://www.semanticweb.org/Innovators/ontologies/2017/7/"
```

```

val ontology = manager.createOntology(IRI.create(ONTOLOGYURI, "Sports#"))
//Prefix for all the entities
val pm = new DefaultPrefixManager(null, null, ONTOLOGYURI+"Sports#")

// Declaration Axiom for creating Classes

val classes1=Source.fromFile("Data/classes.txt").getLines()

classes1.foreach(f=>{
    val cls = df.getOWLClass(f, pm)
    val declarationAxiomcls= df.getOWLDeclarationAxiom(cls)
    manager.addAxiom(ontology, declarationAxiomcls)
})

val objprop=Source.fromFile("Data/ObjProp.txt").getLines()
objprop.foreach(f=> {
    val farr=f.split(",")
    val domain = df.getOWLClass(farr(1), pm)
    val range = df.getOWLClass(farr(2), pm)
    //Creating Object property 'hasGender'
    val objpropaxiom = df.getOWLObjectProperty(farr(0), pm)

    val rangeAxiom = df.getOWLObjectPropertyRangeAxiom(objpropaxiom, range)

    val objpropaxiom = df.getOWLObjectProperty(farr(0), pm)

    val rangeAxiom = df.getOWLObjectPropertyRangeAxiom(objpropaxiom, range)
    val domainAxiom = df.getOWLObjectPropertyDomainAxiom(objpropaxiom, domain)

    //Adding Axioms to ontology
    manager.addAxiom(ontology, rangeAxiom)
    manager.addAxiom(ontology, domainAxiom)
    if(farr(3)=="Func")
        manager.addAxiom(ontology, df.getOWLFunctionalObjectPropertyAxiom(objpropaxiom))
    else if(farr(3).contains("InvOf"))
    {
        val inverse=farr(3).split(":")
        val inverseaxiom = df.getOWLObjectProperty(inverse(1), pm)

        val rangeAxiom = df.getOWLObjectPropertyRangeAxiom(inverseaxiom, domain)
        val domainAxiom = df.getOWLObjectPropertyDomainAxiom(inverseaxiom, range)

        //Adding Axioms to ontology
        manager.addAxiom(ontology, rangeAxiom)
        manager.addAxiom(ontology, domainAxiom)
        manager.addAxiom(ontology, df.getOWLInverseObjectPropertiesAxiom(objpropaxiom, inverseaxiom))
    }
})

val dataprop=Source.fromFile("Data/DataProp.txt").getLines()

```

```

dataprop.foreach(f=>{
    val farr=f.split(",")
    val domain=df.getOWLClass(farr(1),pm)
    // Creating Data Property 'fullName'
    val fullName = df.getOWLDataProperty(farr(0), pm)
    val domainAxiomfullName = df.getOWLDataPropertyDomainAxiom(fullName, domain)
    manager.addAxiom(ontology, domainAxiomfullName)
    if(farr(2)=="string") {
        //Defining String Datatype
        val stringDatatype = df.getStringOWLDatatype()
        val rangeAxiomfullName = df.getOWLDataPropertyRangeAxiom(fullName, stringDatatype)
        //Adding this Axiom to Ontology
        manager.addAxiom(ontology, rangeAxiomfullName)
    }
    else if(farr(2)=="int")
    {
        //Defining Integer Datatype
        val Datatype = df.getIntegerOWLDatatype()
        val rangeAxiomfullName = df.getOWLDataPropertyRangeAxiom(fullName, Datatype)
        //Adding this Axiom to Ontology
        manager.addAxiom(ontology, rangeAxiomfullName)
    }
})

val individuals=Source.fromFile("Data/Individuals.txt").getLines()

individuals.foreach(f=>{
    val farr=f.split(",")
    val cls=df.getOWLClass(farr(0), pm)
    val ind = df.getOWLNamedIndividual(farr(1), pm)
    val classAssertion = df.getOWLClassAssertionAxiom(cls, ind)
    manager.addAxiom(ontology, classAssertion)
})

val triplets=Source.fromFile("Data/triplets.txt").getLines()
triplets.foreach(f=>{
    val farr=f.split(",")
    val sub = df.getOWLNamedIndividual(farr(0), pm)

    if(farr(3)=="Obj")
    {
        val pred=df.getOWLObjectProperty(farr(1),pm)
        val obj=df.getOWLNamedIndividual(farr(2), pm)
        val objAsser = df.getOWLObjectPropertyAssertionAxiom(pred,sub, obj)
        manager.addAxiom(ontology, objAsser)
    }
    else if(farr(3)=="Data")
    {
        val pred=df.getOWLDataProperty(farr(1),pm)
        val dat=df.getOWLLiteral(farr(2))
        val datAsser = df.getOWLDataPropertyAssertionAxiom(pred,sub, dat)
        manager.addAxiom(ontology, datAsser)
    }
})

```

```

val os = new FileOutputStream("Data/Sports_ontology.owl")
val owlxmlFormat = new OWLXMLDocumentFormat
manager.saveOntology(ontology, owlxmlFormat, os)
System.out.println("Ontology Created")
os.close()

```

Output:

```

<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
  xmlns:base="http://www.semanticweb.org/Innovators/ontologies/2017/7/Sports#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  ontologyIRI="http://www.semanticweb.org/Innovators/ontologies/2017/7/Sports#">
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#" />
  <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
  <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace" />
  <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#" />
  <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#" />
  <Declaration>
    <Class IRI="DATE" />
  </Declaration>
  <Declaration>
    <Class IRI="DURATION" />
  </Declaration>
  <Declaration>
    <Class IRI="LOCATION" />
  </Declaration>
  <Declaration>
    <Class IRI="MISC" />
  </Declaration>
  <Declaration>
    <Class IRI="NUMBER" />
  </Declaration>
  <Declaration>
    <Class IRI="ORDINAL" />
  </Declaration>
  <Declaration>
    <Class IRI="ORGANIZATION" />
  </Declaration>
  <Declaration>
    <Class IRI="PERSON" />
  </Declaration>
  <Declaration>

```

```
<NamedIndividual IRI="Latif"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="Chris"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="Sean"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="African"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="Anil"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="Butt"/>
</Declaration>
<Declaration>
  <DataProperty IRI="whips_ball_from"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="Pakistan_&apos;s_Abdul_Razzaq"/>
</Declaration>
<Declaration>
  <DataProperty IRI="is_dropped_at"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="three-day_warm-up"/>
</Declaration>
```



```
<ClassAssertion>  
    <Class IRI="MISC"/>  
    <NamedIndividual IRI="Australian"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
<ClassAssertion>  
    <Class IRI="PERSON"/>  
    <NamedIndividual IRI="Azharuddin"/>  
</ClassAssertion>  
  
-----  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="serving"/>  
    <NamedIndividual IRI="Azharuddin"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">life_ban</Literal>  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="serving"/>  
    <NamedIndividual IRI="Azharuddin"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">life_ban_for_match-fixing</Literal>  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="told"/>  
    <NamedIndividual IRI="Azharuddin"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">BBC_Sport</Literal>  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="backed_at_time"/>  
    <NamedIndividual IRI="BCCI"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">last_September</Literal>  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="backed_out_of"/>  
    <NamedIndividual IRI="BCCI"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">deal</Literal>  
</DataPropertyAssertion>  
<DataPropertyAssertion>  
    <DataProperty IRI="backed_out_of"/>  
    <NamedIndividual IRI="BCCI"/>  
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">deal with Zee</Literal>
```

```

    <Class IRI="PERSON"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_deposited"/>
    <Class IRI="PERSON"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_deposited"/>
    <Class IRI="PERSON"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_dropped_at"/>
    <Class IRI="PERSON"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_three-day_warm-up_from"/>
    <Class IRI="LOCATION"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_three-day_warm-up_from"/>
    <Class IRI="LOCATION"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="is_warm-up_from"/>
    <Class IRI="LOCATION"/>
</DataPropertyDomain>

    <Class IRI="LOCATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="is_warm-up_against"/>
    <Class IRI="LOCATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="is_warm-up_against"/>
    <Class IRI="LOCATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="is_warm-up_in"/>
    <Class IRI="LOCATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="is_warm-up_in"/>
    <Class IRI="LOCATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="is_with"/>
    <Class IRI="DURATION"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="launches"/>
    <Class IRI="PERSON"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="made"/>
    <Class IRI="PERSON"/>

```

Generation of SPARQL/SWRL:

```
//reading predicate list
BufferedReader br1 = new BufferedReader(new FileReader( fileName: "E:\\UMKC\\Sum_May\\KDM\\week 7\\Q2Query\\Q2Query.txt"));
String s1;
ArrayList props = new ArrayList<String>();
while(( s1= br1.readLine()) != null)
{
    String s2[] = s1.split( regex: "," );
    for(String s3:s2)
        props.add(s3);
}

String a[] = classExpression.split( regex: " ");

for(String x : a)
{
    Iterator<String> i1 = props.iterator();
    while(i1.hasNext())
    {
        String s3 = i1.next();
        if(s3.contains(x)) {
            query = s3 + " value " + a[a.length-1];
        }
    }
}

static class DLQueryEngine {

    //Reasoner for validating owl
    //Parser for going through ontology file

    private final OWLReasoner reasoner;
    private final DLQueryParser parser;

    public DLQueryEngine(OWLReasoner reasoner, ShortFormProvider shortFormProvider) {...}

    //getting super classes
    public Set<OWLClass> getSuperClasses(String classExpressionString, boolean direct) {...}

    //getting equivalent classes
    public Set<OWLClass> getEquivalentClasses(String classExpressionString) {...}

    //getting subclasses
    public Set<OWLClass> getSubClasses(String classExpressionString, boolean direct) {
        if (classExpressionString.trim().length() == 0) {
            return Collections.emptySet();
        }
        OWLClassExpression classExpression = parser
            .parseClassExpression(classExpressionString);
        NodeSet<OWLClass> subClasses = reasoner.getSubClasses(classExpression, direct);
        return subClasses.getFlattened();
    }
}
```

```

static class DLQueryParser {
    //creates easily readable and querable format i.e Manchester OWL syntaxed
    private final OWLOntology rootOntology;
    private final BidirectionalShortFormProvider bidiShortFormProvider;

    public DLQueryParser(OWLOntology rootOntology, ShortFormProvider shortFormProvider) {
        this.rootOntology = rootOntology;
        OWLOntologyManager manager = rootOntology.getOWLOntologyManager();
        Set<OWLOntology> importsClosure = rootOntology.getImportsClosure();
        // Create a bidirectional short form provider to do the actual mapping.
        // It will generate names using the input
        // short form provider.
        bidiShortFormProvider = new BidirectionalShortFormProviderAdapter(manager,
            importsClosure, shortFormProvider);
    }

    public OWLClassExpression parseClassExpression(String classExpressionString) {
        OWLDataFactory dataFactory = rootOntology.getOWLOntologyManager()
            .getOWLDataFactory();
        ManchesterOWLSyntaxEditorParser parser = new ManchesterOWLSyntaxEditorParser(
            dataFactory, classExpressionString);
        parser.setDefaultOntology(rootOntology);
        OWLEntityChecker entityChecker = new ShortFormEntityChecker(bidiShortFormProvider);
        parser.setOWLEntityChecker(entityChecker);
        return parser.parseClassExpression();
    }
}

```

```

static class DLQueryPrinter {

    //Printing results for the query

    private final DLQueryEngine dlQueryEngine;
    private final ShortFormProvider shortFormProvider;

    public DLQueryPrinter(DLQueryEngine engine, ShortFormProvider shortFormProvider) {
        this.shortFormProvider = shortFormProvider;
        dlQueryEngine = engine;
    }

    public String askQuery(String classExpression) {
        if (classExpression.length() == 0) {
            // System.out.println("No class expression specified");
            return "No class expression specified";
        } else {
            StringBuilder sb = new StringBuilder();

            Set<OWLNamedIndividual> individuals = dlQueryEngine.getInstance(
                classExpression, direct: true);
            printEntities( name: "Instances", individuals, sb);
            // System.out.println(sb.toString());
            return sb.toString();
        }
    }
}

```

Question Answering using your KG and SPARQL/SWRL:

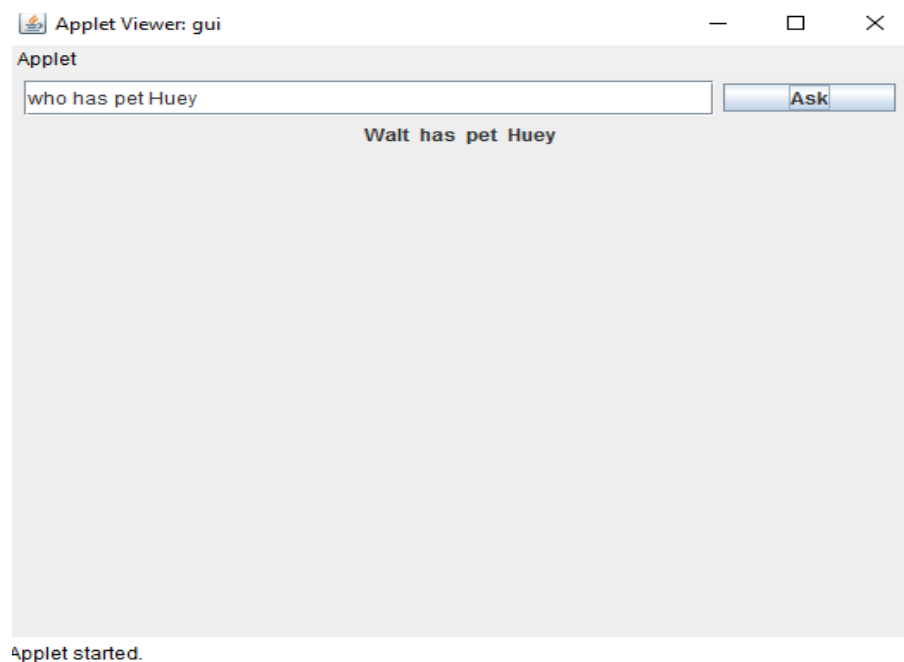
```

private JPanel getCustPanel() {
    JPanel panel = new JPanel();
    // panel.setLayout ((LayoutManager) new BoxLayout(panel, BoxLayout.Y_AXIS));
    text=new JTextField();
    text.setPreferredSize( new Dimension( width: 400, height: 25 ) );
    text.setToolTipText("Please enter your question");
    text.setAlignmentX(Component.CENTER_ALIGNMENT);
    button = new JButton( text: "Ask");
    button.setPreferredSize(new Dimension( width: 100, height: 20));
    button.setAlignmentX(Component.CENTER_ALIGNMENT);
    button.addActionListener( k: this);
    panel.add(text);
    panel.add(button);
    l=new JLabel();
    text.setPreferredSize( new Dimension( width: 400, height: 25 ) );
    l.setAlignmentX(Component.CENTER_ALIGNMENT);
    panel.add(l);
    return panel;
}

public void actionPerformed(ActionEvent e) {
    try {
        Question2Query q = new Question2Query();
        ans = q.call(text.getText());
        l.setText("\n"+ans);
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}

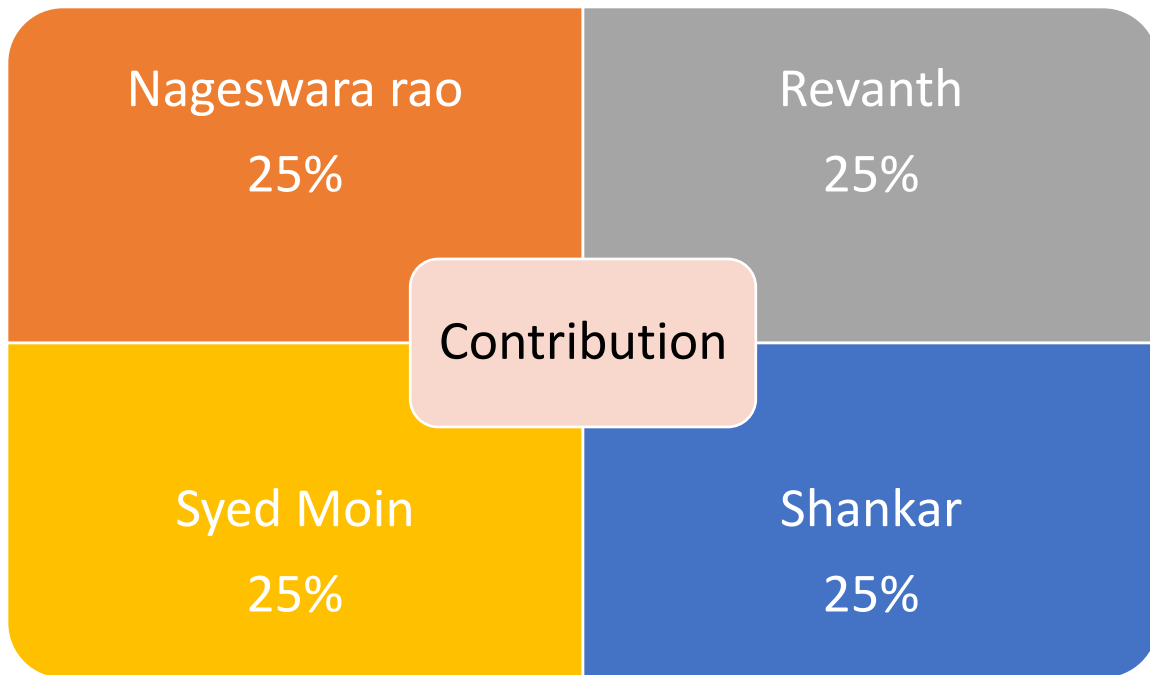
```

Output:



Project Management:

Contribution:



GitHub screens:

Board:

The screenshot displays the GitHub Boards interface, which is a Kanban-style board for managing issues. The top navigation bar includes links to Code, Issues (4), Pull requests (0), Boards (selected), Reports, Projects (0), Wiki, and Insights. Below the navigation bar, there are filters for Labels, Milestones, Assignees, Epics, and Releases, along with a search bar and a 'New Issue' button.

The board is organized into four columns, each representing a stage in the workflow:

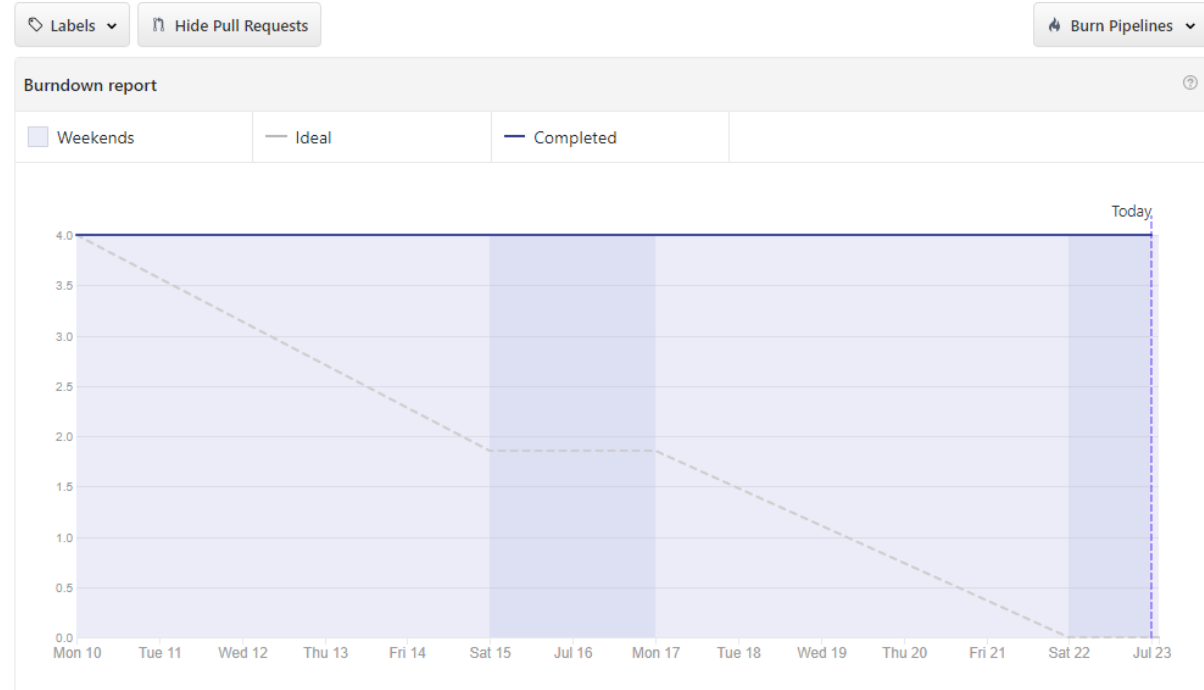
- Progress**: 0 issues, 0 Story Points.
- Review/QA**: 1 issue, 2 Story Points. It contains one issue: 'KDM_Project #9: SPARQL query not able to run' (Increment 3) with a 'question' label.
- Done**: 3 issues, 4 Story Points. It contains three issues: 'KDM_Project #6: Relation Extraction using OpenIE' (Increment 2) with a 'duplicate' label, 'KDM_Project #8: Identifying classes, sub classes and object properties' (Increment 3) with a 'help wanted' label, and 'KDM_Project #10: creation of ontology'.
- Closed**: 6+ issues, 6 Story Points. It contains four issues: 'KDM_Project #5: Term weight using TFIDF' (Increment 2) with a 'question' label, 'KDM_Project #4: Documemation' (Increment 1) with an 'enhancement' label, and 'KDM_Project #3: Function calling from spark' (Increment 1).

Each issue card includes a user avatar, the issue title, the increment value, and any associated labels. The 'Closed' column has a 'Closed' status icon. On the far right, there is a button to 'Add a Pipeline'.

Burndown report:

Increment 3

Start: Jul 10, 2017 [Change](#) Due: Jul 23, 2017 [Change](#)



Issues:

Filters

is:issue is:open

Labels

Milestones

New issue

4 Open

6 Closed

Author

Labels

Projects

Milestones

Assignee

Sort

creation of ontology

bug 1

#10 opened 25 seconds ago by Nagumkc

Increment 3

Done

SPARQL query not able to run

question 2

#9 opened a minute ago by Nagumkc

Increment 3

Review/QA

Identifying classes, sub classes and object properties

help wanted 1

#8 opened 3 minutes ago by Nagumkc

Increment 3

In Progress

Relation Extraction using OpenIE

duplicate 2

#6 opened 14 days ago by shankarpentyala07

Increment 2

Done

Future work:

Till now, we have created the ontology and the data statically from the Data set using the NLP techniques and have created question and answering system based on the extracted data.

The Future scope is to generate the Question and Answering based on the Knowledge graph dynamically by parsing data and finding out the entities and relationship between the entities. The main Entity extraction information is implicitly done using Natural language and explicitly done using structured data markup.