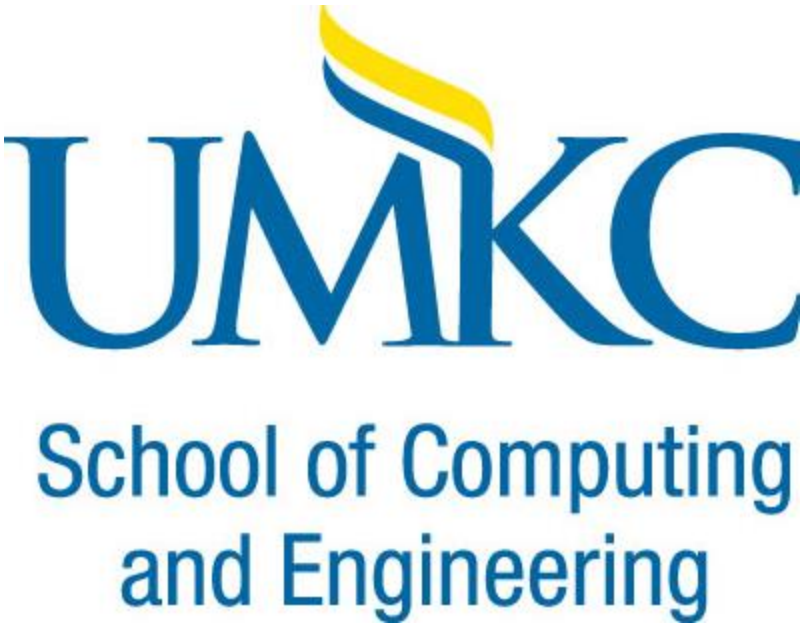


Dynamic Intelligent Q/A System

PROJECT INCREMENT 1 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 always in a search of Information. But Importantly there is a lot of difference in information & knowledge. Information intruding or a semantic google search is mature and we can retrieve relevant information at our finger tips that too sitting in home. “Question Answering” is a dynamic specialized form of Information Retrieval which learns knowledge. We are not only particular in getting the relevant pages but we are also particular in getting the specific answer to query we post. Question Answering is a itself intersection of the NLP Natural Language Processing, Information Retrieval, Machine Learning, Knowledge Representation, Logic, Inference and Sematic Search. It also provides a nice platform to deal into “almost” all Artificial Intelligence. Let’s assume If a statement is made that “Question Answering is the ultimate AI”, then the statement will be univocally accepted. A Question answering system in its being is an art of NLP and at the same instance it has a bit of science in its essence. Question Answering System is needed everywhere, let it be in medical science, an intelligent learning system for students, professional assistants etc. So, It is necessity in every aspect where we need some assistance from computers 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 complex models of information knowledge extraction, named Question/Answering (QA) model. If suppose any given question posed in natural language, QA systems are designed in such a way to extract the reality possible answer in the form of a semantic group or a pre-defined named-entity type, i.e. a person, an organization, a city, etc. Thus, we are relating query terms with existing entities in a given radius is very crucial in QA systems. Our main goal is to improve the performance of our Question Answering system by utilizing information from the natural groupings of words in documents, i.e. some topics, in relation to named entity types in their ranges.

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.

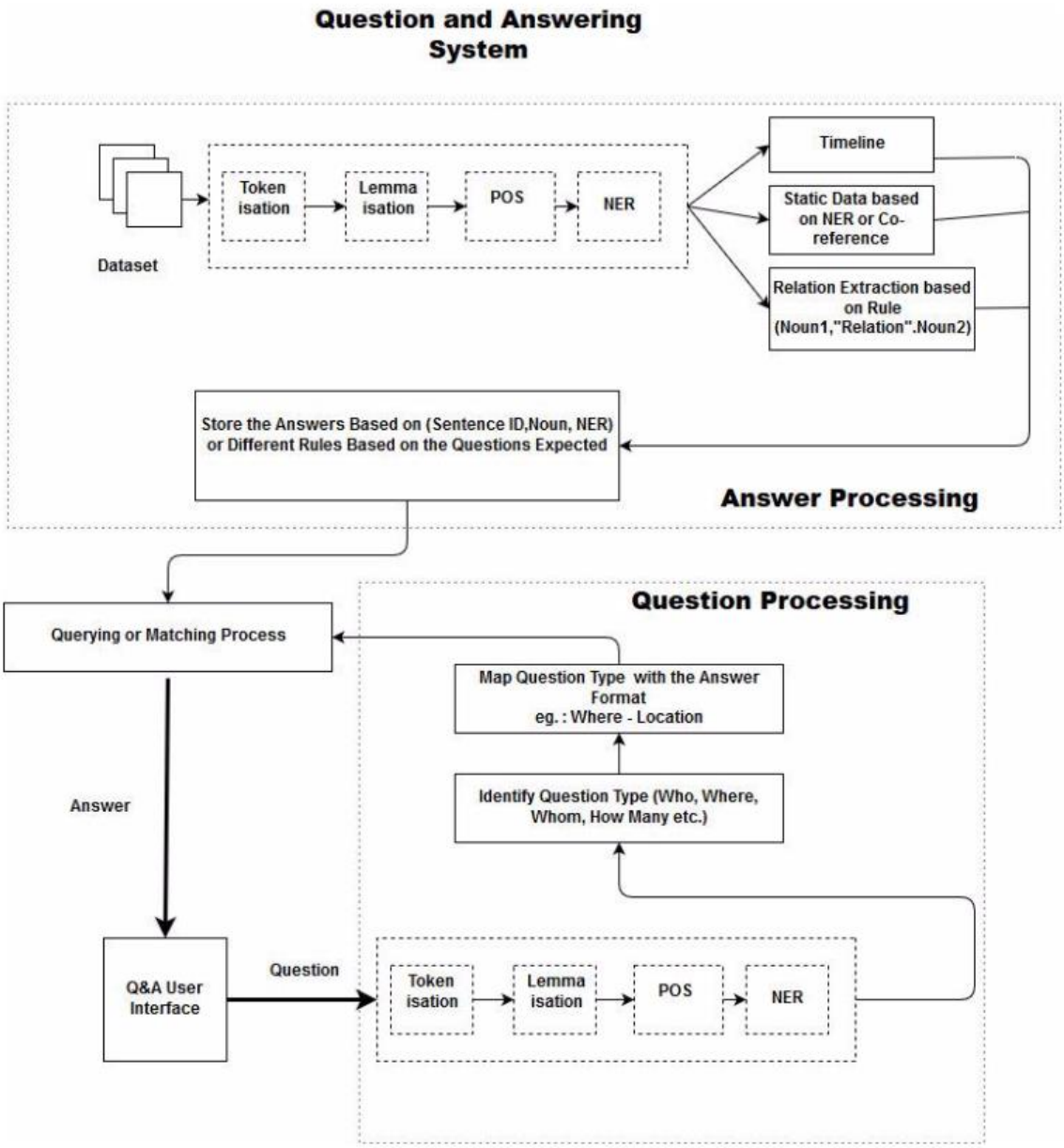
Datasets:

BBC Sports domain

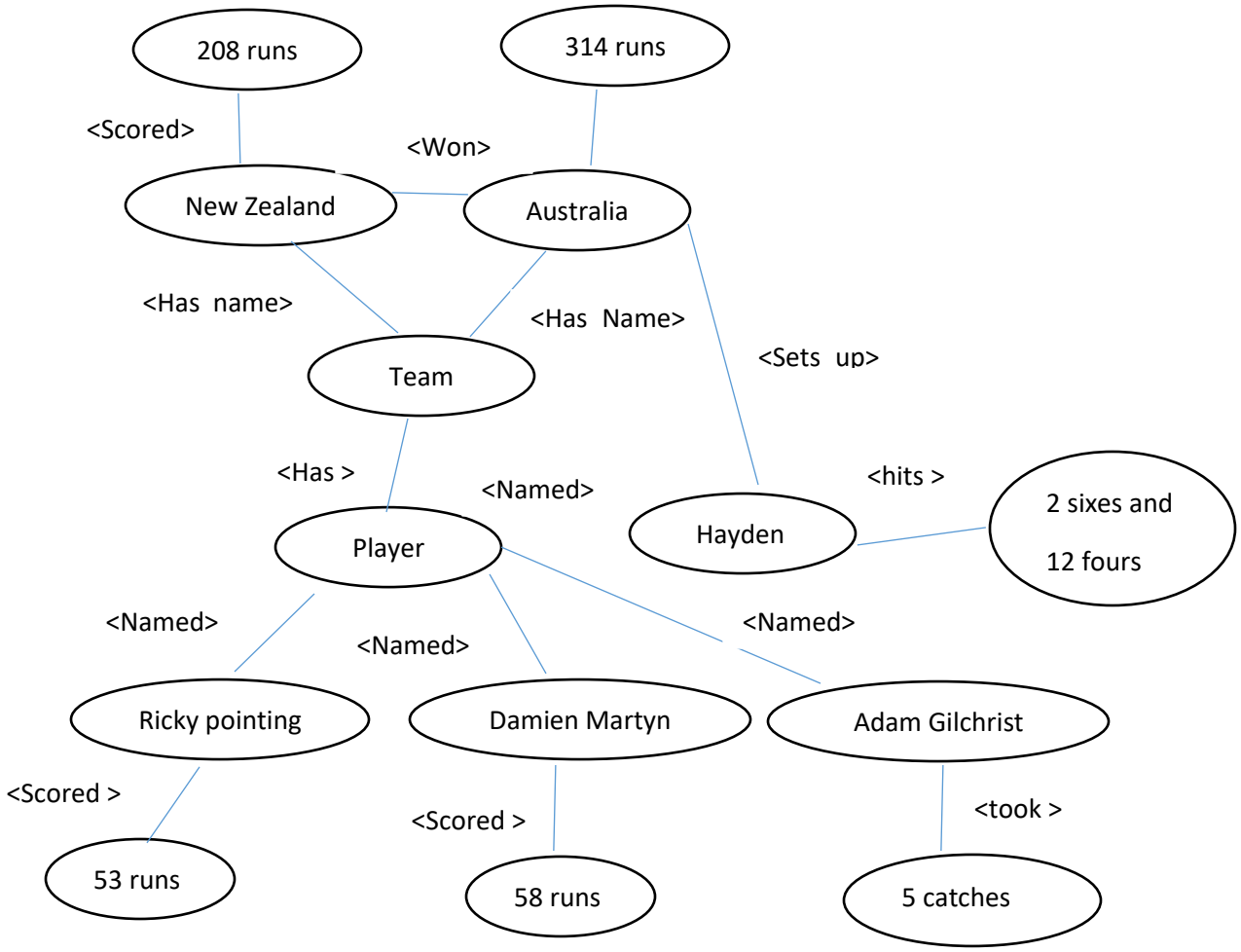
http://mlg.ucd.ie/datasets/bbc.html

Design:

Workflow:



Knowledge Graph:



Implementation:

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();
}
```

Asking questions and finding answer type based on “wh” words

```
System.setProperty("hadoop.home.dir", "E:\\UMKC\\Sum_May\\KDM\\winutils")

val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

val sc = new SparkContext(sparkConf)
val call: NLP = new NLP();
val i = 0

val text = sc.textFile("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcsport\\cricket\\001.txt");
for (a <- 0 to 2) {
  val input = scala.io.StdIn.readLine()
  if (input.contains("who")) {
    val r1 = text.map(line => {
      call.ret(line, "PERSON")
    })
    fun(r1,input)
  }
  if (input.contains("where")) {
    val r1 = text.map(line => {
      call.ret(line, "LOCATION")
    })
    fun(r1,input)
  }
  if (input.contains("when")) {
    val r1 = text.map(line => {
      call.ret(line, "DATE")
    })
    fun(r1,input)
  }
```

And based on answer type, we are calling particular document who is responsible to answer.

```
object sparkgrouplemm {
 def main(args:Array[String]): Unit ={

   System.setProperty("hadoop.home.dir","E:\\UMKC\\Sum_May\\KDM\\winutils")

   val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

   val sc=new SparkContext(sparkConf)
   val call:NLP=new NLP();
val text=sc.textFile("E:\\UMKC\\Sum_May\\KDM\\week1\\bbcsport\\cricket\\001.txt");
   val t1=text.map(l=>{call.lemm(l)})

   val t2=t1.flatMap(d=>{d.split(" ")}).filter(f=>(!(f.contains(",")|f.contains(".")|(f.isEr
   val t3=t2.groupBy(g=>{g.charAt(0)})
   t3.collect().foreach(println)
 }
}
```

Question and answers:

Question 1:

```
17/06/20 13:36:59 INFO SparkContext: Created broadcast 0 from textFile at qanda.scala:18
where the venue of newzland vs australia match
17/06/20 13:37:39 INFO FileInputFormat: Total input paths to process : 1
17/06/20 13:37:39 INFO SparkContext: Starting job: take at qanda.scala:49
```

Answer:

```
17/06/20 13:38:12 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 3) in 12 ms
17/06/20 13:38:12 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all com
Australia
New
Jade
Zealand
Stadium
```

Question 2:

```
when the newzland vs australia match happened
17/06/20 13:38:50 INFO SparkContext: Starting job: take at qanda.scala:49
```

Answer:

```
past
now
1993
March
once
```

Question 3:

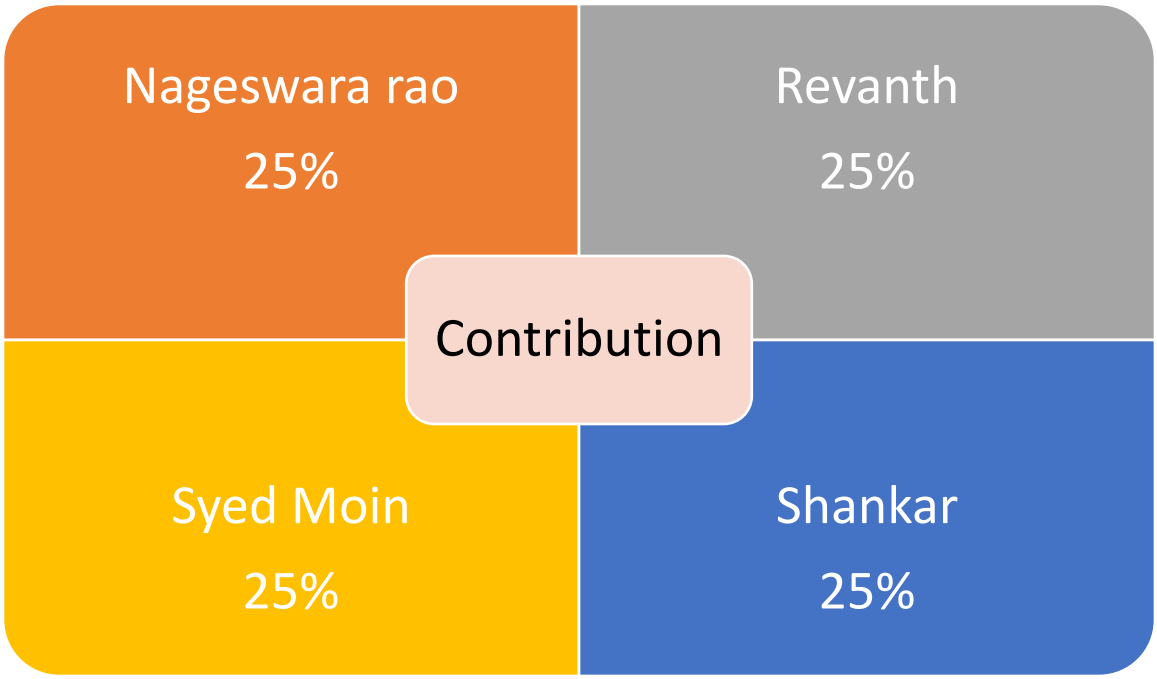
```
17/06/20 13:38:56 INFO DAGScheduler: Job 3 finished: take at qanda.scala:49, took 0.021529 s
who all are played between australia vs newzland match
17/06/20 13:39:27 INFO SparkContext: Starting job: take at qanda.scala:49
```

Answer:

```
Ricky
Hamish
Craig
Damien
Wilson
McGrath
Glenn
Cairns
Marshall
```

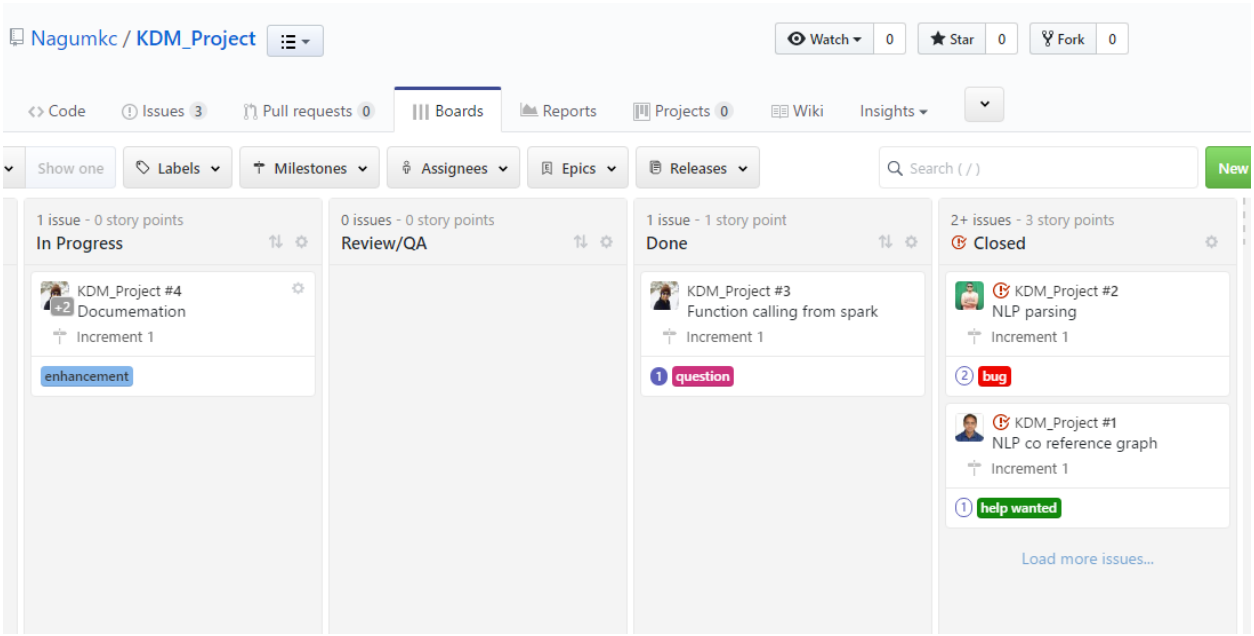
Project Management:

Contribution:



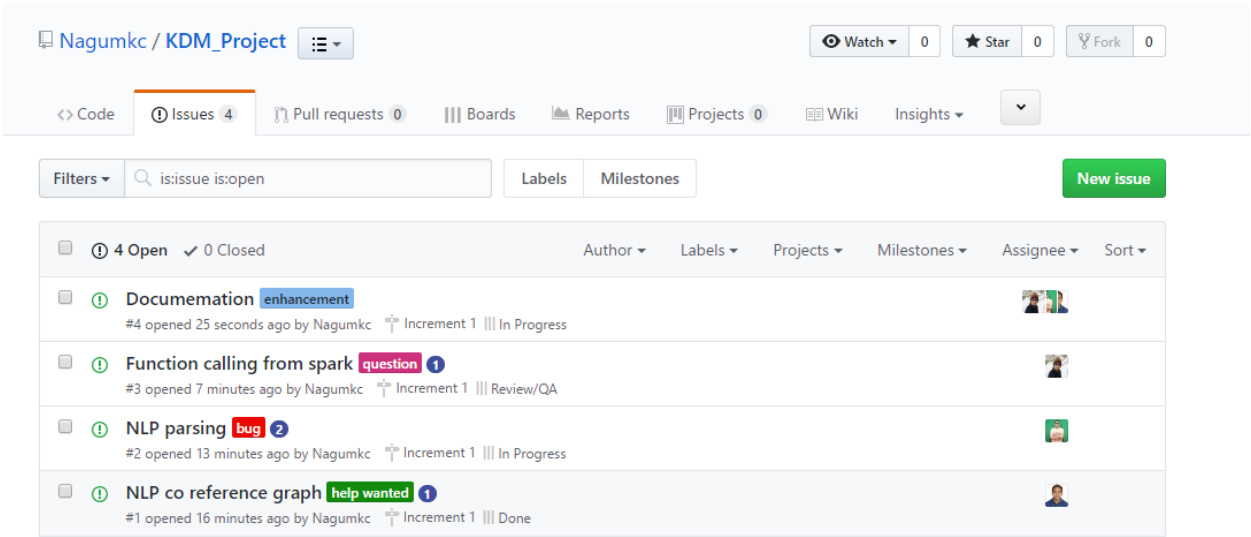
GitHub screens:

Board:



Burndown report:

Issues:



Future work:

Till now, we have extracted 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.