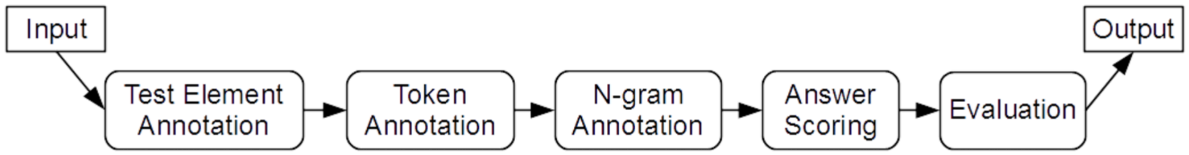
Yuqi Zhang
Andrew ID: yuqiz1
February 14, 2017

Project3 Report

The project is to design a logical data model to process question and answers information. The main idea of the project follows such process:



The input to the first pipeline is a text file containing one question followed by several potential answers. Each answer will have an id and a score which is either 1 or 0. 1 indicates that the answer is correct and 0 means that the answer is incorrect. Question and answers are separated by '\n'. The following is a sample input:

```
Q John loves Mary?
A1 1 John loves Mary with all his heart.
A2 1 Mary is dearly loved by John.
A3 0 Mary doesn't love John.
A4 0 John doesn't love Mary.
A5 1 John loves Mary.
```

The main idea of the design is to assign each answer a score based on its n-gram tokens overlap with the original questions. Finally, the system output the rank of the answers and also a precision value by comparing with the fact. The followings are description of each class in detail.

The first class named as *ElementAnnotation* will read in the name of the input file, and parse the documents. Each sentence will have an corresponding annotation, and the id is the name of the sentence such as "Q" or "A1". In each annotation, the original sentence will be recorded, and there will be a feature called *Type* to record whether the sentence is a question or a answer.

Meanwhile, for each answer there will be a feature called *Score* to record whether the answer is correct or in correct for the purpose of evaluation later. For example, the following is an annotation in the output view:

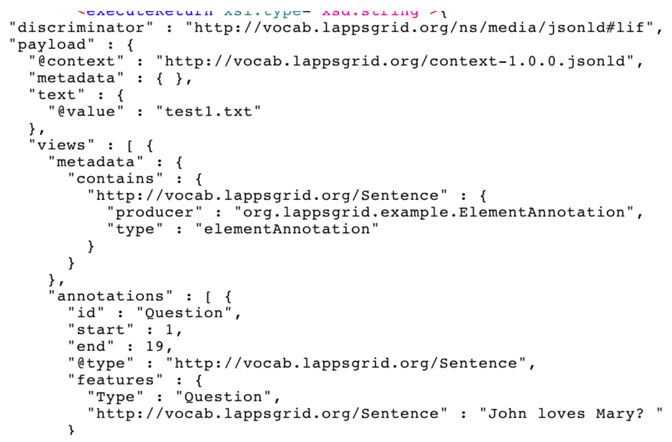
```
"id" : “A1",     \\name of the sentence
"start" : 4,      \\start point of the answer in the original sentence
"end" : 41,        \\end point of the answer in the original sentence
"@type" : “http://vocab.lappsgrid.org/Sentence", \\type of the annotation
"features" : {
   "Type" : “Answer",   \\record whether it is an answer or a question
   "http://vocab.lappsgrid.org/Sentence" : "John loves Mary with all his heart. ",
   "Score" : “1"   \\record whether it is correct answer
```

The input and output format of metadata are:

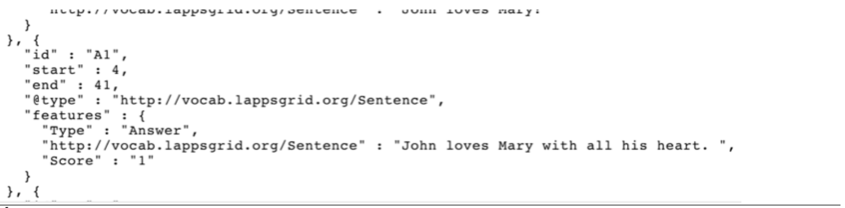
```
IOSpecification requires = new IOSpecification();
requires.addFormat(Uri.TEXT);           // Plain text (form)
requires.addFormat(Uri.LIF);            // LIF (form)
requires.addLanguage("en");             // Source language
requires.setEncoding("UTF-8");

// JSON for output information
IOSpecification produces = new IOSpecification();
produces.addFormat(Uri.LAPPS);          // LIF (form) synonymous to LIF
produces.addAnnotation(Uri.SENTENCE);      // SENTENCE (contents)
requires.addLanguage("en");             // Target language
produces.setEncoding(“UTF-8");
```

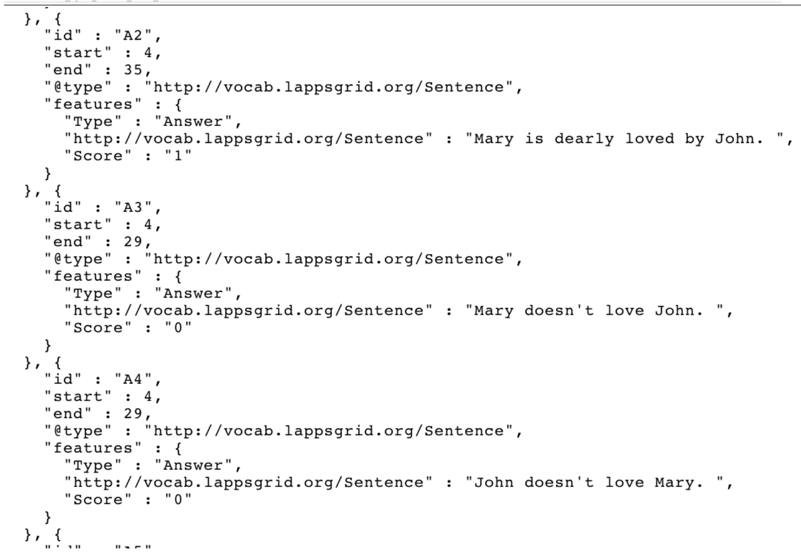
A sample output of the given class is shown below:



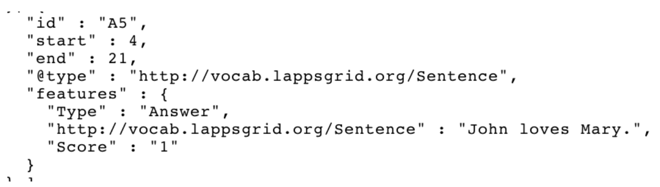
```
      <executeReturn xsi:type="xsd:string">{
"discriminator" : "http://vocab.lappsgrid.org/ns/media/jsonld#lif",
"payload" : {
  "@context" : "http://vocab.lappsgrid.org/context-1.0.0.jsonld",
  "metadata" : { },
  "text" : {
    "@value" : "test1.txt"
  },
  "views" : [ {
    "metadata" : {
      "contains" : {
        "http://vocab.lappsgrid.org/Sentence" : {
          "producer" : "org.lappsgrid.example.ElementAnnotation",
          "type" : "elementAnnotation"
        }
      }
    },
    "annotations" : [ {
      "id" : "Question",
      "start" : 1,
      "end" : 19,
      "@type" : "http://vocab.lappsgrid.org/Sentence",
      "features" : {
        "Type" : "Question",
        "http://vocab.lappsgrid.org/Sentence" : "John loves Mary? "
      }
```



```
        "http://vocab.lappsgrid.org/Sentence" : "John loves Mary?"
      }
    }, {
      "id" : "A1",
      "start" : 4,
      "end" : 41,
      "@type" : "http://vocab.lappsgrid.org/Sentence",
      "features" : {
        "Type" : "Answer",
        "http://vocab.lappsgrid.org/Sentence" : "John loves Mary with all his heart. ",
        "Score" : "1"
      }
    }, {
```



```
}, {
  "id" : "A2",
  "start" : 4,
  "end" : 35,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "Type" : "Answer",
    "http://vocab.lappsgrid.org/Sentence" : "Mary is dearly loved by John. ",
    "Score" : "1"
  }
}, {
  "id" : "A3",
  "start" : 4,
  "end" : 29,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "Type" : "Answer",
    "http://vocab.lappsgrid.org/Sentence" : "Mary doesn't love John. ",
    "Score" : "0"
  }
}, {
  "id" : "A4",
  "start" : 4,
  "end" : 29,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "Type" : "Answer",
    "http://vocab.lappsgrid.org/Sentence" : "John doesn't love Mary. ",
    "Score" : "0"
  }
}, {
```



```
  "id" : "A5",
  "start" : 4,
  "end" : 21,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "Type" : "Answer",
    "http://vocab.lappsgrid.org/Sentence" : "John loves Mary.",
    "Score" : "1"
  }
```

The second task is to create token annotation. Like a whitespace tokenizer, after reading in the output of the first class, the class, *TokenAnnotation*, will tokenize each sentence by whitespace. Each token will have an annotation and a feature, *Group*, recording which sentence it comes from. A sample annotation looks like:

```
"id" : “Question-tok0”,    \\id of the token
"start" : 0,               \\start point of the token in the sentence
"end" : 4,                  \\end point of the token in the sentence
"@type" : “http://vocab.lappsgrid.org/Token",    \\type of the annotation
"features" : {
     "word" : “John",                \\original word
      "Group" : “Question"           \\which sentence it comes from
 }
```

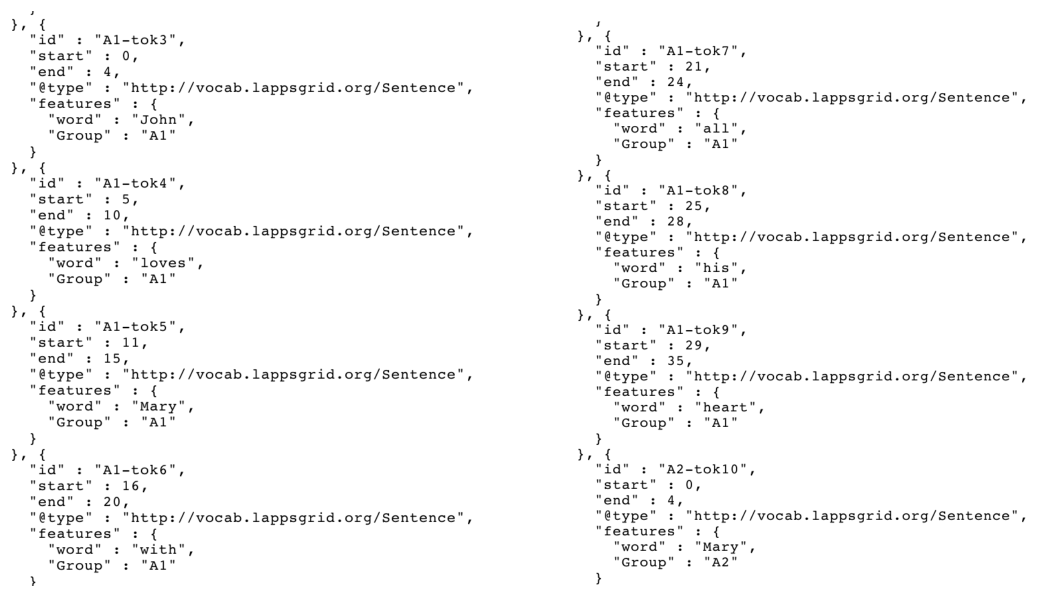
The input and output format of metadata are:

```
IOSpecification requires = new IOSpecification();
```

```
requires.addFormat(Uri.TEXT);           // Plain text (form)
requires.addFormat(Uri.LAPPS);            // LIF (form)
requires.addLanguage("en");             // Source language
requires.setEncoding("UTF-8");

// JSON for output information
IOSpecification produces = new IOSpecification();
produces.addFormat(Uri.LAPPS);          // LIF (form) synonymous to LIF
produces.addAnnotation(Uri.TOKEN);      // Tokens (contents)
requires.addLanguage("en");             // Target language
produces.setEncoding("UTF-8");
```

The following is part of the sample output view of *TokenAnnotation:*



```
  ,
}, {
  "id" : "A1-tok3",
  "start" : 0,
  "end" : 4,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "John",
    "Group" : "A1"
  }
}, {
  "id" : "A1-tok4",
  "start" : 5,
  "end" : 10,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "loves",
    "Group" : "A1"
  }
}, {
  "id" : "A1-tok5",
  "start" : 11,
  "end" : 15,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "Mary",
    "Group" : "A1"
  }
}, {
  "id" : "A1-tok6",
  "start" : 16,
  "end" : 20,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "with",
    "Group" : "A1"
  }
```

```
  ,
}, {
  "id" : "A1-tok7",
  "start" : 21,
  "end" : 24,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "all",
    "Group" : "A1"
  }
}, {
  "id" : "A1-tok8",
  "start" : 25,
  "end" : 28,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "his",
    "Group" : "A1"
  }
}, {
  "id" : "A1-tok9",
  "start" : 29,
  "end" : 35,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "heart",
    "Group" : "A1"
  }
}, {
  "id" : "A2-tok10",
  "start" : 0,
  "end" : 4,
  "@type" : "http://vocab.lappsgrid.org/Sentence",
  "features" : {
    "word" : "Mary",
    "Group" : "A2"
  }
```

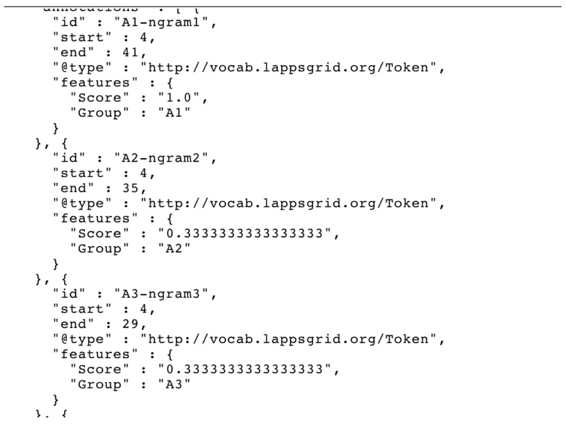
Next step is the N-gram Annotation, and the corresponding class is called *NgramAnnotation*. The purpose of this class is to annotate the 1-, 2- and 3-grams of consecutive answers. In this output view, each sentence in the input file will have an annotation. And each annotation will have three features which are 1-, 2- and 3-grams. In each feature, a map containing the n-gram and the corresponding occurrences will be recorded. For example:

```
"id" : "A2-ngram2",
        "start" : 4,
        "end" : 35,
        "@type" : "http://vocab.lappsgrid.org/Sentence",
        "features" : {
```

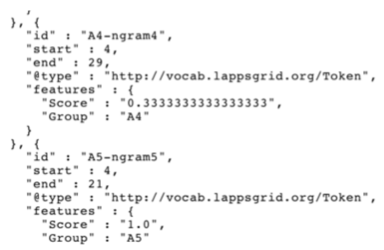
```
    "1-Gram" : {
      "loved" : 1,
      "by" : 1,
      "John" : 1,
      "is" : 1,
      "dearly" : 1,
      "Mary" : 1
    },
    "2-Gram" : {
      "is dearly" : 1,
      "by John" : 1,
      "Mary is" : 1,
      "loved by" : 1,
      "dearly loved" : 1
    },
    "3-Gram" : {
      "loved by John" : 1,
      "dearly loved by" : 1,
      "is dearly loved" : 1,
      "Mary is dearly" : 1
    },
    "Group" : "A2"
  }
```

The input and output format for metadata is the same as above.

Based on the N-Gram Annotation, we can score each answer based on the overlap of the answer with the original question in *AnswerScoring*. We consider all the n-grams tokens of the question as a bag of words. Then for each answer, if a token occurs in the answer, we add one point for the answer. The n is determined by the user input. Finally, we normalized the scores for those answers by the total count of the bag of words in the question. In this output view, each answer will have an annotation recording its score. For example:



```
      "id" : "A1-ngram1",
      "start" : 4,
      "end" : 41,
      "@type" : "http://vocab.lappsgrid.org/Token",
      "features" : {
        "Score" : "1.0",
        "Group" : "A1"
      }
    }, {
      "id" : "A2-ngram2",
      "start" : 4,
      "end" : 35,
      "@type" : "http://vocab.lappsgrid.org/Token",
      "features" : {
        "Score" : "0.3333333333333333",
        "Group" : "A2"
      }
    }, {
      "id" : "A3-ngram3",
      "start" : 4,
      "end" : 29,
      "@type" : "http://vocab.lappsgrid.org/Token",
      "features" : {
        "Score" : "0.3333333333333333",
        "Group" : "A3"
      }
    }, {
```



```
  ,
}, {
  "id" : "A4-ngram4",
  "start" : 4,
  "end" : 29,
  "@type" : "http://vocab.lappsgrid.org/Token",
  "features" : {
    "Score" : "0.3333333333333333",
    "Group" : "A4"
  }
}, {
  "id" : "A5-ngram5",
  "start" : 4,
  "end" : 21,
  "@type" : "http://vocab.lappsgrid.org/Token",
  "features" : {
    "Score" : "1.0",
    "Group" : "A5"
```

The input and output format for metadata is the same as above.

The last task is *Evaluation.* In the part of evaluation, the answers will be sorted by the scores. Finally, we calculated precision at N and N is the total number of correct answers. For example, in this sample input file, A1, A2 and A5 are correct and therefore, N is 3 in this example. Based on the rank of the answers, the precision at 3 is 0.67:

```
"annotations" : [ {
        "id" : "Final Result",
        "start" : 0,
        "end" : 0,
        "@type" : "http://vocab.lappsgrid.org/Token",
        "features" : {
          "Scorelist" : [ "1 A1 1 A5 0.6667 A2 0.6667 A3 0.6667 A4" ],
          "Precision" : “0.66666667"
        }
      } ]
```

The function could run with commands similar to the following:

```
mvn exec:java -Dexec.mainClass="Main" -Dexec.args="1 src/main/resources/inputData path/to/your/outputData"
```

, where inputData and outputData are two directories. The input file name should start with ‘q’ for example “q001.txt”.

Difficulties:

The main difficulty of designing the system is to understand the inner structure and the characteristics of LAPPS. It took me some time to understand how to use the LAPPS format.

Moreover, the pipeline is difficult to debug since it is hard to find out the direct reason that causes the failure.

Package Structure:

```
Pi2-yuqiz1
  - src
        -main
              -java
                    -default package
                          -Pipeline
                          -QAPipeline
                    -org/lappsgrid/example
                          -AnswerScoring.java
                          -ElementAnnotation.java
                          -Evaluation.java
                          -NgramAnnotation.java
                          -TokenAnnotation.java
              -resources
                    -inputData
                          -q001.txt (sample input file)
                    -outputData
                    -docs
                          -pi3-yuqiz1-report.pdf
              -webapp
        -test
  - doc   (javadoc of the system)
  - pom.xml
```