11791 Design and Engineering of Intelligent Information Systems

Fall 2013 hw2

Name: Bo Ma, Andrew id: bom

Email: bom@andrew.cmu.edu

1. Type system

I have add some type into the original type system. In the following figure 1 , you will see the hierarchy of my type system. I will give the detailed information of each system.

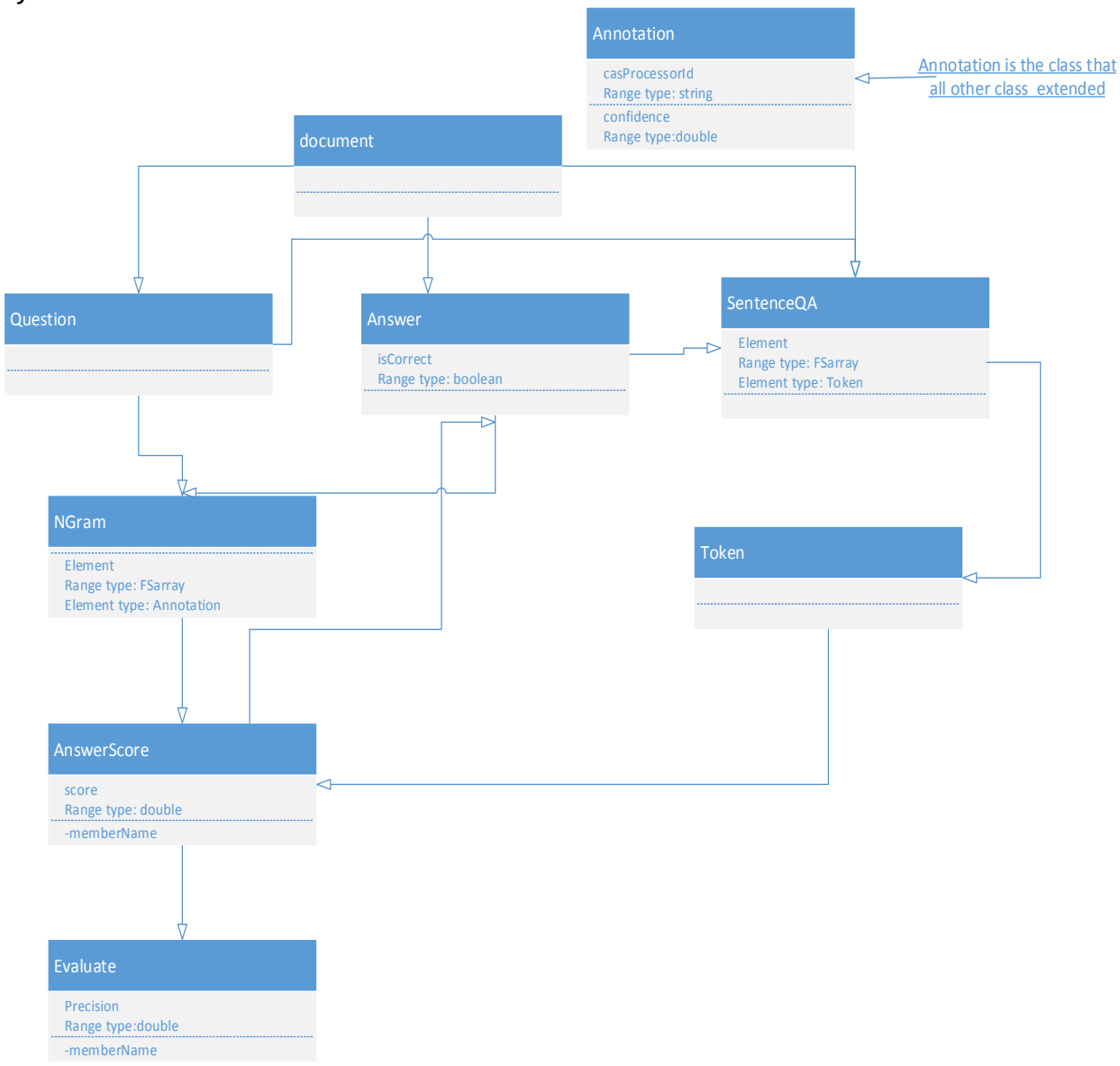

Figure 1: the The Diagram of Type hierarchy

- **Annotation:**

This kind of type is a base type in the system. The two features this type has are casProcessorId and confidence. The other type will be the extended class of this one. Each annotation need a confidence score, so this will provide confidence score for each annotation. casProcessorId is a string that we can get where the annotator is produced and provide information for annotator.

- **Answer:**

This kind of type is to show a instance of one answer. Also, it answer whether this kind of answer is correct answer or just a wrong answer.

- **AnswerScore:**

This kind of type is give the calculating score of each answer. I use the three way to calculate the score, there are gold pipeline, token overlap pipeline, Ngram overlap pipeline.

- **Question:**

This kind of type is annotating the question in document so that we can get the question from the document.

- **Token:**

This kind of type is the smallest type in the document. The system will annotate each token span in each question and answer (break on whitespace and punctuation).

- **NGram:**

This is a kind of type that use N token to form a Gram to calculate the score of the answer.

- **SentenceQA:**

This type contain the sentence in the question and answer, and it is used to make it easy to tokenize the question and answer.

2. Analysis Engine Pipeline

In this homework, my main goal is to implement the analysis engine pipeline to create analysis engines and annotators to annotate the sample input files. I have implement the three model.(gold, token overlap, Ngram). If you want to test the three model, you should the flow and order in the Aggregate Analyse Engine.

The followings are my idea of solving this kind problem.

a. First, what we receive from the input is a document, and we should recognize the question and answer. In the question parse, we just need to find the sentence of the question that is enough, but when we come to the answer, we need to parse its ground truth(whether this is the right answer or the wrong answer) and sentence as well. I define a new type structure “sentence”, because it is it combine the question sentence and answer sentence together and easy for me to tokenize the sentence string into token array.

b. Next, what I need to do is to use the token that we get from the last step to build our language model. I have built three language model. The first is the Gold Pipeline, the second is token overlap and the third is Ngram overlap. The token structure is very useful to implement these pipeline and give the score to each answer.

c. Finally what I have done, is to use the score that I calculated in the last step to rank the answer and use the ground truth that we have parsed in the first step to evaluate my model and compare the different evaluation also.

In the following, I will give the structure of my each annotator.

QuestionAnswerTokenize Annotator

Input: Document
Output: Question, Answer, Token, sentence.
Description: This annotator will parse the Question and Answer first. I use the java package Matcher, Pattern. For the question I use the “"Q\\s(.*)\\?" to match the question sentence from the input string. And use the "A\\s([01])\\s(.*)\\." To match the answer sentence. This initial word of each sentence really helps my to recognize the question and answer.
Code: QuestionAnswerTokenizeAnnotator.java
Descriptor: QuestionAnswerTokenizeDescritor.xml

NGram Annotator

Input: Question, Answer, Token, sentenceDocument
Output: Ngram.
Description: This annotator will use the Ngram language model to get the unigram, bigram, trigram. Then the next step I can use the Ngram to get the score of each answer.
Code: NgramAnnotator.java
Descriptor: NgramDescriptor.xml

AnswerScore Annotator

Input: Question, Answer, Ngram
Output: AnswerScore
Description: This annotator will use the Ngram results. And calculate the score of each answer. It will find the Ngram that in the question appears also in the answer and use

this formula “questionN-Grams found / total ansewr N-Grams (for 1-, 2- and 3-grams)” to give the score to each answer
Code: AnswerScoreAnnotator.java
Descriptor: AnswerScoreDescriptor.xml

TokenOverlapAnswerScore Annotator

Input: Question, Answer, Ngram
Output: TokenOverlapAnswerScore
Description: This annotator will use the the Tokenoverlap method to give the score to each answer sentence. This calculating formula is “question tokens found / total answer tokens“
Code: TokenOverlapAnswerScoreAnnotator.java
Descriptor: TokenOverlapAnswerScoreDescriptor.xml

GoldAnswerScore Annotator

Input: Question, Answer, Ngram
Output: goldAnswerScore
Description: This annotator will use the the gold pipeline method to give the score to each answer sentence. This calculating formula is inspect answer key, rank all correct answers 1.0, 0.0 otherwise.
Code: GoldScoreAnnotator.java
Descriptor: GoldDescriptor.xml

Evaluate Annotator

Input: Question, Answer, AnswerScore
Output: outputfile, precision , average precision
Description: As we get the score of each answer, we can use the sort method to rank the answer. After getting the sorted answer, what I am going to do is to judge evaluation of my model. I will calculate the “precistion at N” . N is the correct answer in the ground truth. And also the destroy method also let me calculate the average precision.
Code: EvaluateAnnotator.java
Descriptor: EvaluateDescriptor.xml

Aggregate Analyse Engine

Description: This engine is to manage how the annotator will run. The flow or order of the whole annotator. It will manage the input and output flow of the annotator.
Descriptor: hw2-bom-aee.xml

In the following , you can see the flow of each annotator.

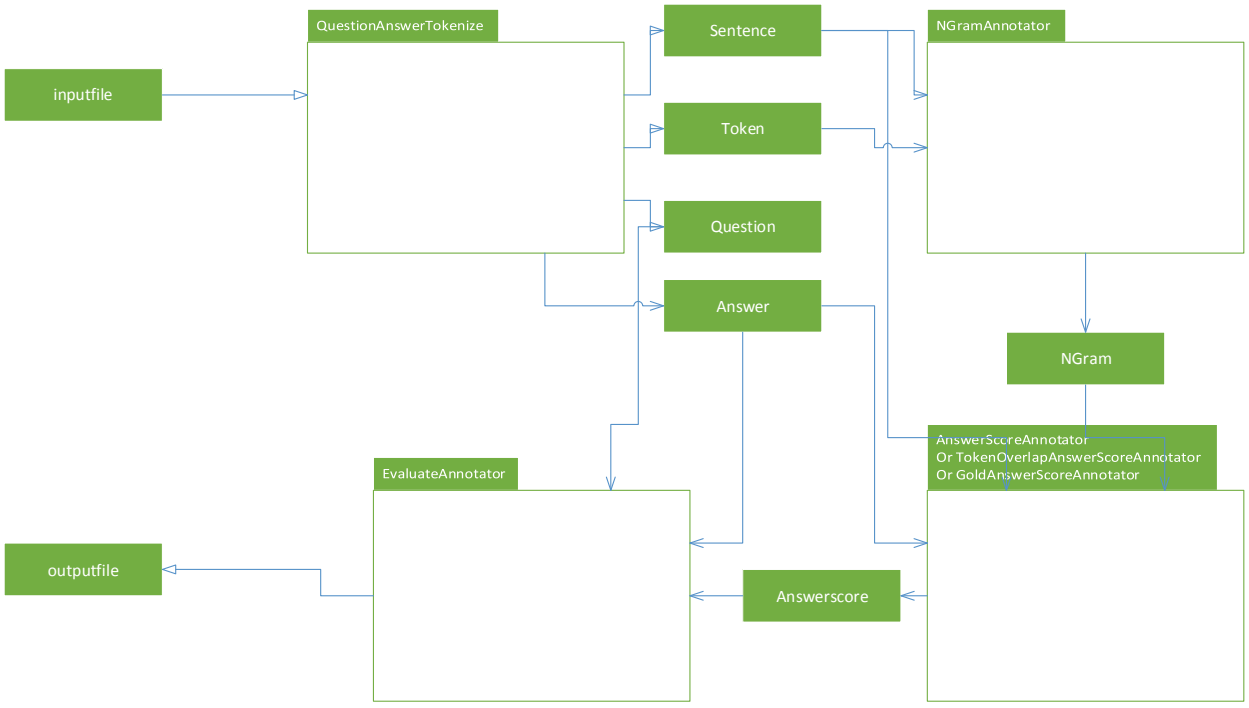

Figure 2: the order of the annotator

Here is also the show of the program:

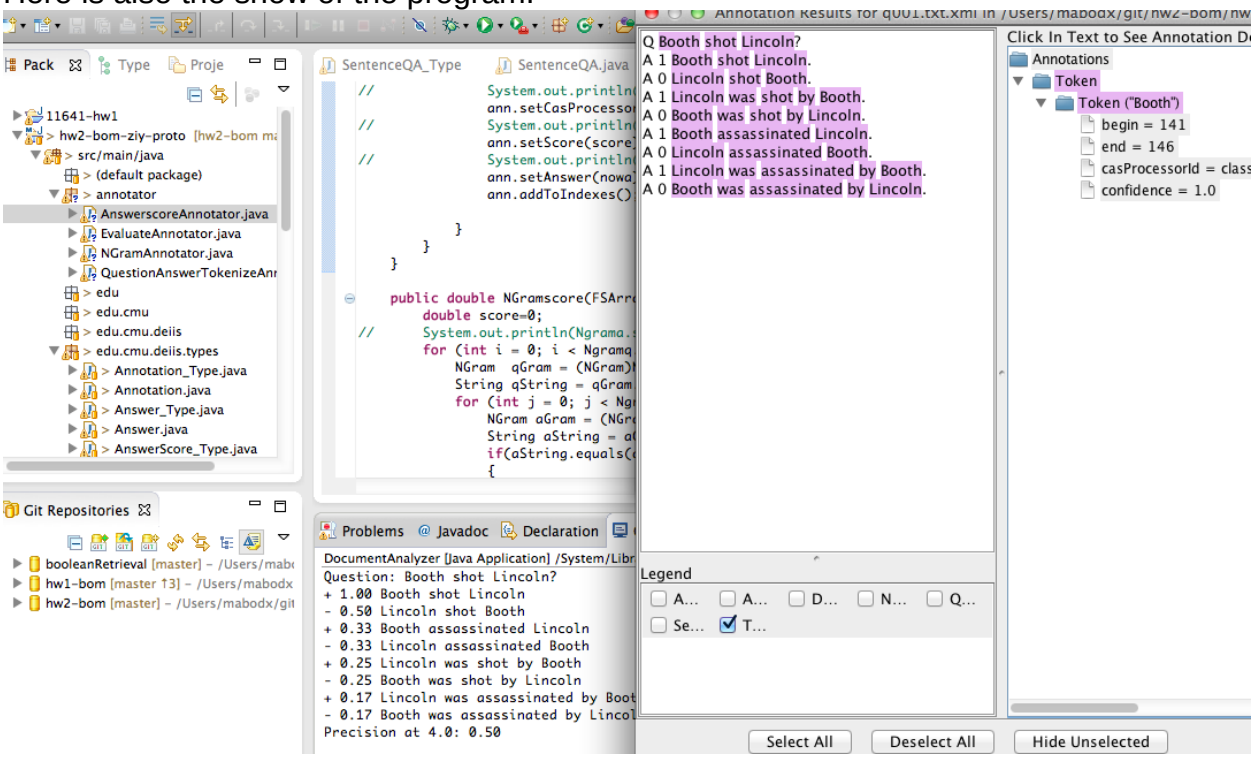

Figure 3: the show of the result of the program

3. Conclusion

I find a very interesting thing. I meet a problem when I want to calculate the “average precision”. Since, I don’t know how to sum all the question’s precision and divide the number of the precision. Then I find that I can implement this in the destroy method of the EvaluateAnnoator. Because the destroy method can only run once. So I can store the average precision in this method and output it. It really helps me the destroy , initial and process method of the AnalysisComponent.

I have implemented gold pipeline, token overlap pipeline, Ngram overlap pipeline. If you want to test the three model, you should the flow and order in the Aggregate Analyse Engine. The token overlap’s result is not as good as the ngram’s. This QA analysis system really helps me to understand the UIMA framework. I combine the Question Answer and tokenize annotator together, because I think it will help me save some code. And some result of the question and answer annotator can be used as to tokenize the sentence.

I use the regular expression method to parse the sentence, It is really convenient and easy to analyse the question and answer. Although most of the type system has already been implemented. I also create a sentence type. Because I think it will be help for to combine the question and answer to tokenize the sentence string together.