11-791 HW1 Report

Zhiyu Li

Sept. 8th, 2013

Name: Zhiyu Li
Andrew ID: zhiyul

1 Summarize

This report includes the architecture design and type system design for the sample information system specified in the homework 1 handout.

2 Architecture Design

According to the assignment's handout, Task 1.3 "The Processing Pipeline", this system's pipeline consists five components: **Test Element Annotator, Token Annotator, NGram Annotator, Answer Scorer and Evaluator.** Each component contains some inputs and outputs. In my design, I divide NGram Annotator into three sub-Annotators, namely UniGram Annotator, BiGram Annotator and TriGram Annotator. The input and output for different components are listed as below:

1. **Test Element Annotator:**
   Input: Text File
   Output: Question+, Answer+, N (total number of correct answers in the Text File)
2. **Token Annotator:**
   Input: Question+, Answer+
   Output: Token+
3. **UniGram Annotator:**
   Input: Question+, Answer+
   Output: NGram+
4. **BiGram Annotator:**
   Input: Question+, Answer+
   Output: NGram+

5. **TriGram Annotator:**

   Input: Question+, Answer+

   Output: NGram+

6. **Answer Scorer:**

   Input: Question+, Answer+

   Output: AnswerScore+

7. **Evaluator:**

   Input: AnswerScore+, N

   Output: Precision at N (print to the screen so do not need to create a type for this)

This order above is also the work flow of the processing pipeline. In each step, the corresponding annotator will assign its annotations the source and the confidence. Different algorithms can be used in these components, for example, we can use the NLP tools from Stanford CoreNLP to tokenize the sentences in components 2, 3, 4 and 5.

The reason why all the three NGram sub-Annotators all produce the NGram-type instances instead of more granular NGram-type instances will be discussed in the *UIMA Type System Design* section.

For the Answer Scorer, depending on the scoring algorithms we are using, the inputs can be different (e.g. The N-Gram Overlap Scoring Algorithm will require the NGram+ from both Question+ and Answer+ as input, while the Token Overlap Scoring Algorithm will require Token+ from both Question+ and Answer+ as input). Since it is not specified in this homework what scoring algorithms we are going to use, I generalize the input as Question+ and Answer+, defer the specific input types design until the scoring algorithms are decided (kind of similar to the Factory Method Pattern).

3 UIMA Type System Design

The UML diagram of the class design is shown in Figure 1.

According to the handout, every annotation must record the name of the component that created it, as well as the confidence score, so I defined a base annotation where all other annotations can inherit from. It has two features: source and confidence.

Question and Answer both extend BaseAnnotation. Notice we did not create another base class called Sentence for them because *the Sentence* will only be extended by these two. Also, we will not use the Sentence annotation in the whole pipeline. Thus it is not worth it.

As mentioned in previous section, we did not define more granular types for NGram. This is for the convenience of the scoring algorithms we are going to use later. For example, the N-Gram Overlap Scoring Algorithm will consider UniGram, BiGram and TriGram as the same entity—NGram—and assign score accordingly, so it is not necessary to divided the NGram type into three.

In the AnswerScore, we have a feature "answer" so after *the Evaluator* sort the scores, we can find the corresponding answer by looking at this feature.

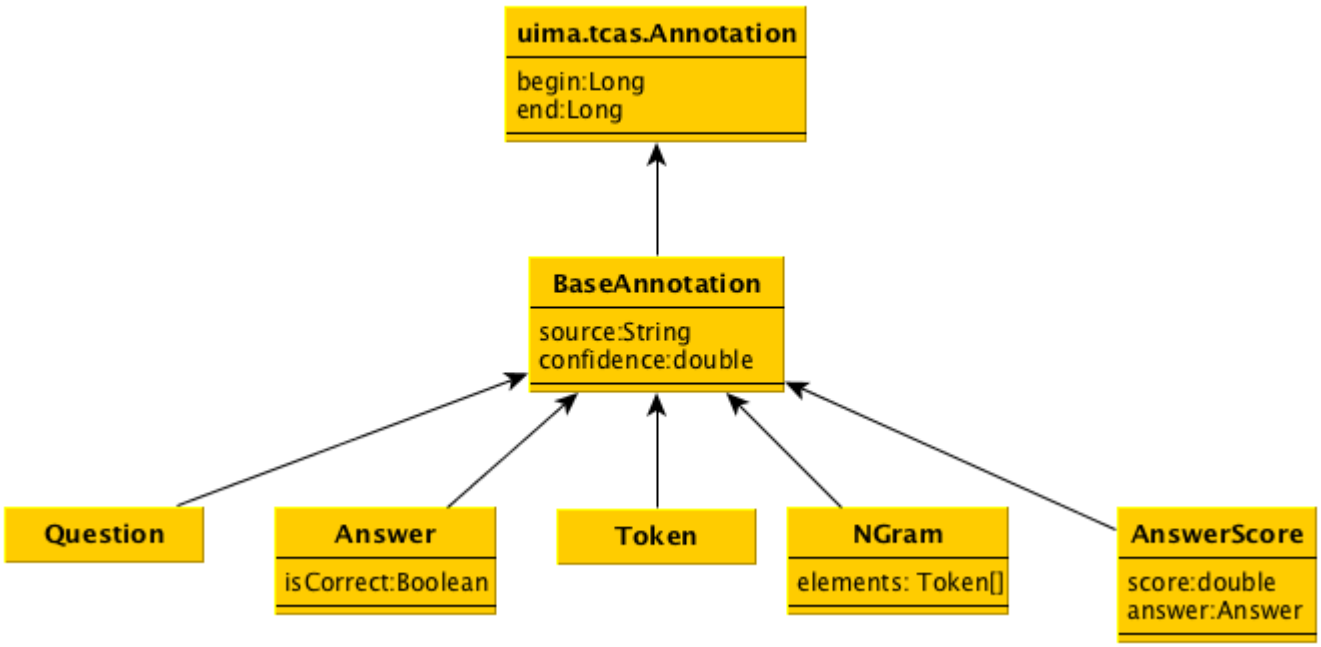

Figure 1: The UML for homework 1.