# 11-791 Homework 1 Report

Yuanchi Ning
**Andrew ID:** yuanchin **Date:** Sep. 2013

# 1. Design

## 1.1 Goal

Given the goal that designing logical data model for information processing of specific input files, here we use UIMA Annotator to annotate the artifacts (which are text files here) in order to facilitate the subsequent processing and analysis.

To annotate the text files, the primary goal of the assignment is to define CAS types that annotator will use. And these definitions are described in the form of "Type System Descriptor", which defines the type system using a XML format.

## 1.2 System Design

To build a complete and irredundant type system, here we define **six** kinds of types.

As the tutorial suggested and based on the requirement that "*All annotations must record the name of the component that produce the annotation, and the confidence score assigned to the annotation by the component*", we create a base annotation type "AnnotationType", which contains the features "begin", "end", "confidence" and "casProcessorId".

| **0. AnnotationType: Base Annotation Type** | |
|---|---|
| **begin: Integer** | Records the beginning of span of annotation |
| **end: Integer** | Records the ending of span of annotation |
| **confidence: Double** | Records the confidence score assigned |
| **casProcessorId: String** | Records the name of the component producing the annotation |

The other 5 types inherit from the "AnnotationType":

| **1. Question: Annotate the question in the sentences** | |
|---|---|
| **begin (inherited) : Integer** | Records the beginning of span of the question in sentence |
| **end (inherited) : Integer** | Records the ending of span of the question in sentence |
| **confidence (inherited) : Double** | Records the confidence score assigned |
| **casProcessorId (inherited): String** | Records the name of the component producing the annotation |

| **2. Answer: Annotate the answer in the sentences** | |
|---|---|
| **begin (inherited) : Integer** | Records the beginning of span of the answer in sentence |
| **end (inherited) : Integer** | Records the ending of span of the answer in sentence |
| **confidence (inherited) : Double** | Records the confidence score assigned |
| **casProcessorId (inherited): String** | Records the name of the component producing the annotation |
| **isCorrect: Boolean** | Records whether the answer is correct |

| **3. AnswerScore: Annotate the answer score of each *Answer* in the sentences** | |
|---|---|
| **begin (inherited) : Integer** | Records the beginning of span of the answer in sentence |
| **end (inherited) : Integer** | Records the ending of span of the answer in sentence |
| **confidence (inherited) : Double** | Records the confidence score assigned |
| **casProcessorId (inherited): String** | Records the name of the component producing the annotation |
| **score: Double** | Records the score assigned to the answer |
| **answer: Answer** | Indicates which answer the score is assigned to |

*This type is quite tricky to define. Since the answer score can also be defined as a feature of the type "Answer", this type is not absolutely necessary. However, for more explicit and convenient subsequent analysis, here we still define a new type. And the "begin", "end" attributes corresponding to the *Answer* type annotation the *Answer Score* associated to.

| **4. Token: Annotate each token span in each *Question* and *Answer* (break on whitespace and punctuation)** | |
|---|---|
| **begin (inherited) : Integer** | Records the beginning of span of the token in sentence |
| **end (inherited) : Integer** | Records the ending of span of the token in sentence |
| **confidence (inherited) : Double** | Records the confidence score assigned |
| **casProcessorId (inherited): String** | Records the name of the component producing the annotation |

| **5. NGram: Annotate 1-, 2- and 3-grams of consecutive tokens** | |
|---|---|
| **begin (inherited) : Integer** | Records the beginning of span of the consecutive tokens in sentence |
| **end (inherited) : Integer** | Records the ending of span of the consecutive tokens in sentence |
| **confidence (inherited) : Double** | Records the confidence score assigned |

| | |
|---|---|
| **casProcessorId (inherited): String** | Records the name of the component producing the annotation |
| **elements: FSArray of type *Token*** | Record consecutive tokens in NGram |
| **elementType: String** | Indicates the element type in NGram |

# 2. Implementation

As described above, we implement the types and features assigned through the XML format file “Type System Descriptor”.

Use UIMA to create the CAS types shown as below:

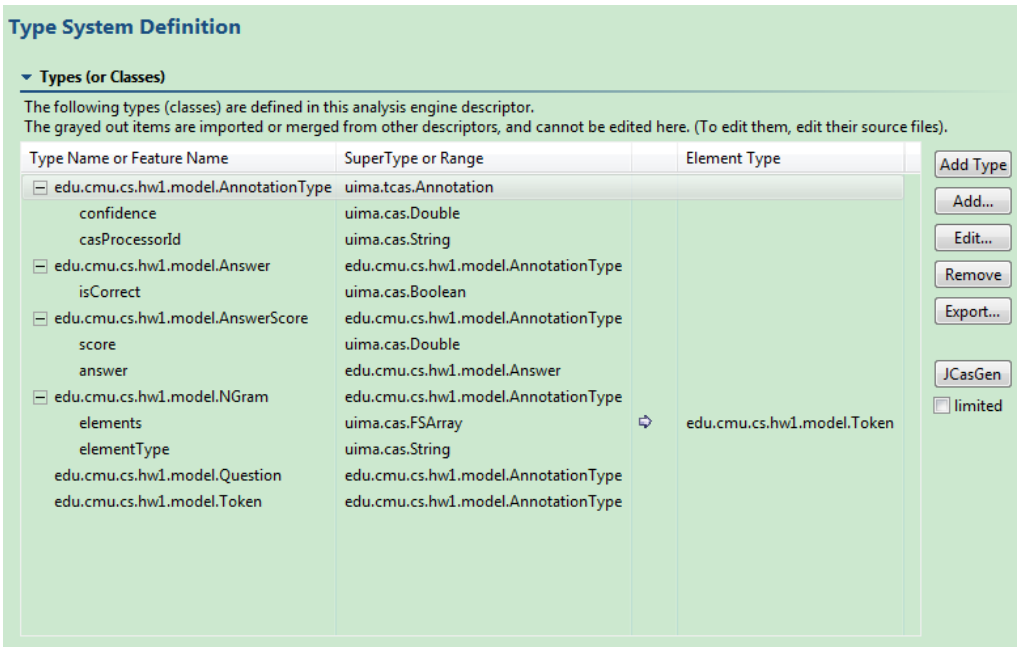


Fig 1. Type Definitions in Implementation

Then automatically generate the Java classes for these types.

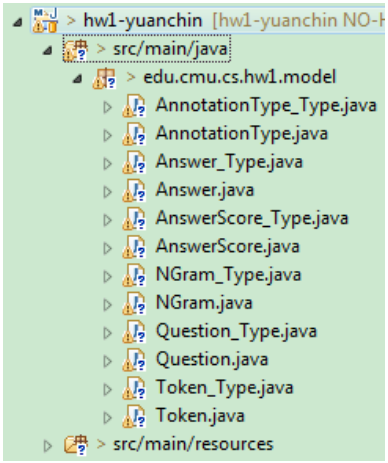


Fig 2. Java classes generated of the types

The UML Class Diagrams are shown as below:

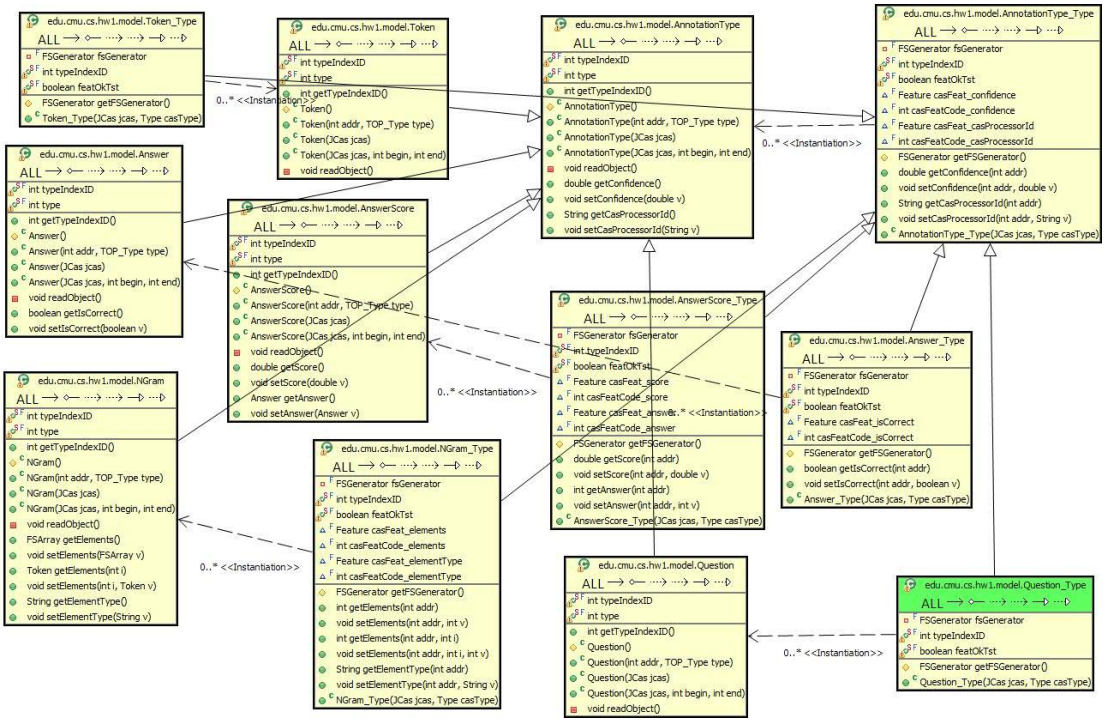


Fig 3. UML Class Diagrams (Generated by *Green UML*, clearer version could be seen at **/src/main/resources/docs/hw1-yuanchin-UML.jpg**)

# Summary

This assignment forced me to figure out what the concepts like *Analysis Engine, Annotator, Annotation, Type System, Feature Structure*, *CAS, etc.* exactly mean. And help me understand more about preparations of upper level information processing applications like NLP and machine learning, even though I have just defined the types of the system and not yet write the actual annotator Java code and the Analysis Engine Descriptor.

What disappointed me a little bit is that I didn't see many roots of algorithmic aspects as the tutorial suggested that we should consider of when designing the types of annotations. Thus it seems that I need further understanding of the role that logical data model played in subsequent analysis.