11-791 Design and Engineering of Intelligent Information System

Homework 1 Report

Logical Data Model and UIMA Type System Design & Implementation

Name : Da Teng

AndrewID: dateng

Date: 09/11/2013

Part 1. Type System Design

1.1 Logical Data Model

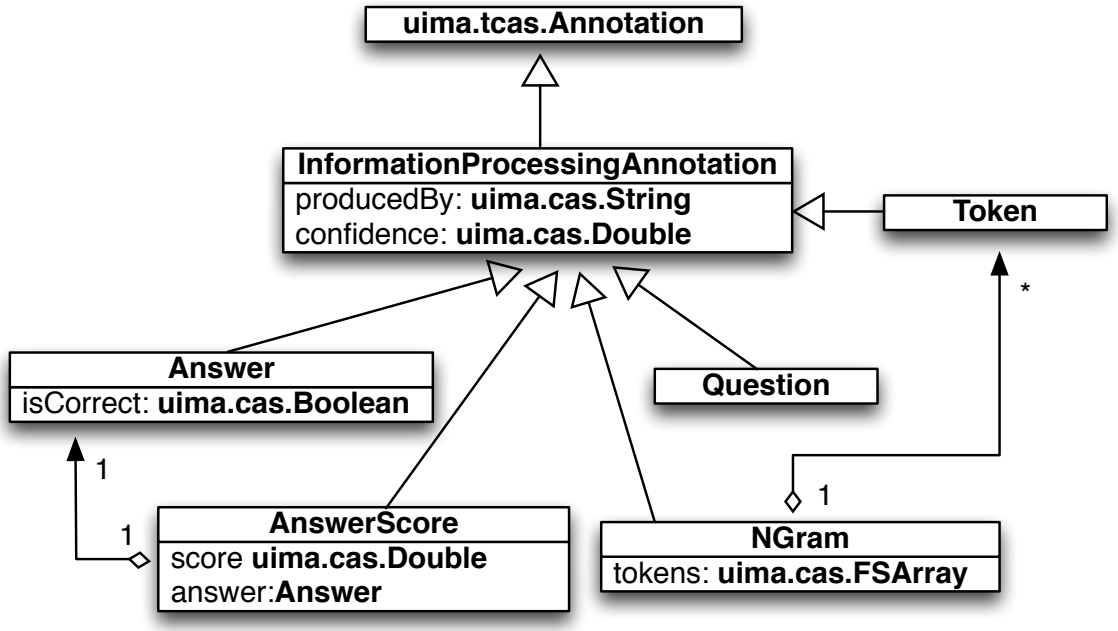

Figure 1.1: System logical data model

According to the requirement of the sample information processing task, five types is designed to complete the task, namely ***InformationProcessingAnnotation***, ***Answer***, ***Question***, ***Token***, ***AnswerScore*** and ***NGram***. The logical data model of these types are shown in Figure 1.1.

InformationProcessingAnnotation: is a base annotation type for the system. It inherit from the built-in uima.tcas.Annotation type to keep track of the span of text to which an annotation refers. It declares two features. The producedBy feature records the name of the component that produced the annotation. The confidence feature records the confidence score assigned to the annotation by the component. All other types in the system inherit from this base annotation type, so that they all have the features stated above.

- **Question**: is a type for recording the input question sentences.

- **Answer**: is a type for recording the input answer sentences. The isCorrect feature of it is used to record whether the sentence is a correct answer for the question. Since an answer can only have values 0 and 1(either correct or incorrect), the isCorrect feature is defined as a Boolean type.

- **Token**: is a type for representing a token in a sentence. When an answer

sentence is entered, it will be parsed into tokens. Then these tokens will be used in NGram analyzing to get an estimated score for whether or not this sentence is a correct answer.

- **NGram**: is a type for representing N contiguous words in a sentence, including unigram, bigram and trigram. It has a feature tokens of FSArray type, which is used to store the N contiguous words. N-grams found in an answer sentence will be used to match with the N-Grams in the Question and generate a score for the answer.

- **AnswerScore**: is a type for recording the score for a certain input answer sentence. The score feature is a score from the N-Gram matching method, ranging from 0 to 1. The answer feature is the corresponding answer sentence of the score.

1.2 Sentence scoring method

Given an answer sentence, system is going to use following N-Gram matching method in generating a score for it.

$$\text{answer score} = \frac{\text{number of N-Grams found in question sentence}}{\text{total number of N-Grams in the answer sentence}}$$

1.3 Information Processing Pipeline

The system will go through following steps for an information processing task:

1. **Test Element Annotation**: System reads question and answer sentences from an input file and generate corresponding Question and Answer annotation. The isCorrect feature of an Answer annotation records whether or not the answer is correct.

2. **Token Annotation**: System creates a Token annotation for each token in question and answer sentences.

3. **NGram Annotation**: System creates N-Gram annotations using contiguous tokens.

4. **Answer Scoring**: System uses N-Gram matching method to generate a score for each answer sentence and creates an AnswerScore annotation for the answer-score pair.

5. **Evaluation**: System sorts answer sentences according to their scores and calculate how many of the top N sentences are correct.

Part 2. Implementing Logical Data Model with UIMA SDK

Type Name or Feature Name	SuperType or Range	Element Type
Answer	InformationProcessingAnnotation	
isCorrect	uima.cas.Boolean	
AnswerScore	InformationProcessingAnnotation	
score	uima.cas.Double	
answer	Answer	
InformationProcessingAnnotation	uima.tcas.Annotation	
producedBy	uima.cas.String	
confidence	uima.cas.Double	
NGram	InformationProcessingAnnotation	
tokens	uima.cas.FSArray	Token
Question	InformationProcessingAnnotation	
Token	InformationProcessingAnnotation	

Figure 2.1 Logical model implementation

The logical model designed in part 1 can be implemented as in Figure 2. After defining system types in the xml file, we can use JCasGen to generate JAVA classes, result shown in Figure 2.2.

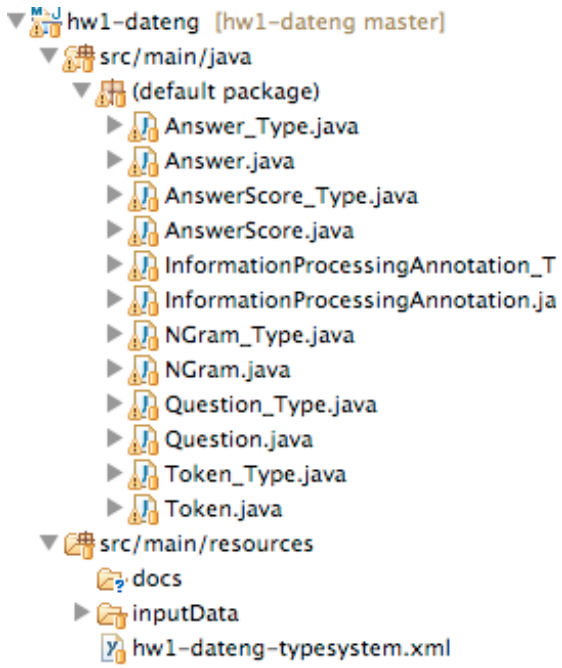

Figure 2.2 JAVA classes generated by JCasGen