11-791 Homework 1 Report

Chen Sun   chens1@andrew.cmu.edu

1. Design of Type System

1.1 Requirement Analysis

Based on the homework document and Eric's slides in classes, there are several methods covered in the information process pipeline.

For the initial input, it is essential to distinguish the Question and Answer spans. Thus, a Question type and an Answer type are needed for this test annotation step. For the Answer type, it should indicate whether the answer is true, thus an *isCorrect* field shall be added.

Generally considered, Question and Answers consist of word tokens. So a Question or Answer type should have multiple (implemented as an Array or a List) Tokens contained in its field. However, from the perspective of system analysis, there is no hierarchical dependencies between tokens and an answer or a question in text process, thus it is better to set Token as a separate type and not included in Answer or Question to reduce the complexity of the system processing.

N-Gram essentially contains multiple N-Grams (1-, 2-, 3-grams). Each N-Gram type should contain an array of tokens it uses to generate the N-Gram. So an N-Gram should contain an array of tokens as its features. This helps to build the N-Gram type from token level.

For an Answer or a Question, however, it may have or generate multiple N-Gram elements. So it is quite useful to build a type named N-GramList to embed several N-grams one sentence generates. In this case, one N-GramList instance corresponds to one Answer or Question.

So far, I have create the models for the input system. However, concerning the output of the text processing, there are still more types to be defined. Firstly, the AnswerScore type. It's a middle type that grants an answer a certain score using various scoring methods. The basic component of an AnswerScore type should have an Answer type as its feature. Besides, it should contain a double number feature to indicate the score the corresponding answer has.

Finally, for the final output, a new type containing the question, all the ranked AnswerScores and precision calculation should be built to meet the requirement. This help the system to build the formatted final output.

1.2 Structure Overview

In this section I will present the structure of the Type System.
Figure 1 is an UML showing the basic structure. All the type classes inherit the basic type system, which name is General. Since all the types are used as “meta” elements, methods are left blank. This diagram omits that “General inherits the default uima super type”. All the classes in the first row are classified as input and the other two are classified as output. This difference will be presented as different packages.
There are no inheritance except that all the classes inherited from the General class, which provides the universal feature. AnswerScore treat Answer as its feature rather than a super type because although AnswerScore contains all the fields Answer holds, an Answer is truly a stand-alone object in the AnswerScore. There is no modification or override of the inside of Answer. It simply adds a new, corresponding field to an Answer object. Thus, it is better to include it rather than inherit it. Working this way, any future modification of Answer type will have no effect on AnswerScore and AnswerScore could simple manage the score feature.
The same reason holds for Evaluator object. It combines the Question and Answers, as well as the number of correct answer, picked correct answer by different scoring methods, and finally a precision rate. It could be used by the Analysis Engine to form a final answer.

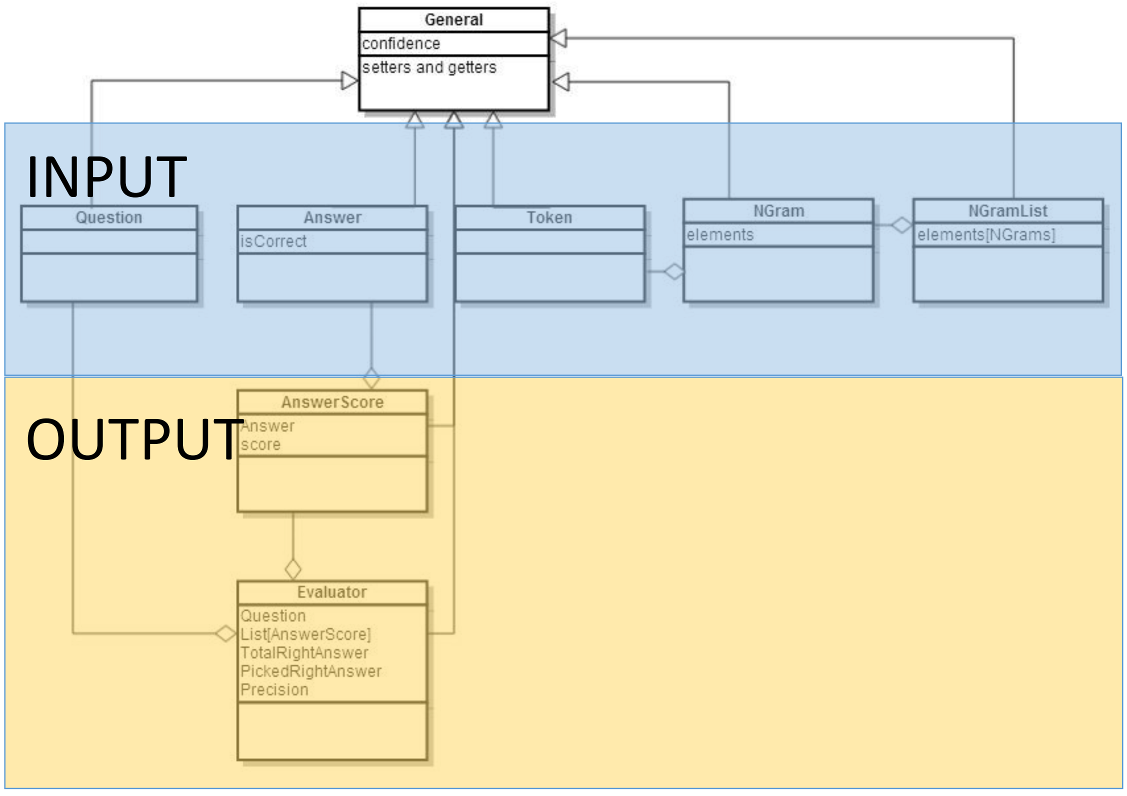

Figure 2 Overall Structure

2. Implementation

2.1 Follow the instruction of homework 1 and install all the required plugins. Then download the required jars to include in the building path. And modify the pom.xml file to make the maven project right.

2.2 Follow the instructions in UIMA Version 2.4.2 Tutorial, creating the corresponding Type System Descriptor .xml file. Add the desired type and feature(s).

2.3 With the coding template available at http://uima.apache.org/codeConventions.html, all the generated coding follow the Apache UIMA code conventions.

2.4 In the submission I have actually implemented two annotators, which were when I did not quite catch what the first homework intended to do. And the two annotators work for the Question/Answer capture and Token analyze. I have run it correctly with my test code (namely System.out.println(staff)) under AE.

2.5 Uima provides us with easy and simple way to generate, represent and use logical model. However, it was really complicated when you first get in touch with them. Several XML files have to be carefully recognized and analyzed. Otherwise you may made the mistakes I have made.

2.6 Since this homework only requires the generation of type system, it is of little use to analyze the type according to its features, such as NLP or machine learning, etc. These features might be helpful in future operations or implementations.

3. Reference

3.1 11-791 Homework 1 Documentation (In blackboard)

3.2 11-791 Homework 0 Documentation (for github use)

3.3 UIMA Tutorial and Developers' Guides(http://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html)