Part 1. UML Design

1.1 Domain diagram for IntelligentInformationSystem

In this part, we were asked to draw a UML Domain Diagram to represent the domain concepts, associations (with multiplicities) and attributes for an Intelligent Information System. I designed a domain model for a license processing intelligent information system as shown in the file <hw1-jli3-uml.pdf> under hw1-jli3/src/main/resources/docs. This domain diagram contains mainly three phases: License Provider, License applicant, and License processing service. Each of these three phases includes several options. For example, the license processing service phase is implemented by four options: LicenseService, LicenseRequest, LicenseRepository and UnderReviewDecison. Each of them will be implemented by a Java class when it is actually implemented and each has any number of configuration parameters depending on the requirements.

The working scenario is described as follows: every time an applicant applies for a license, he/she needs to provide license processing service with his id, name, type, license id and his/her contact information as parameters, each with specific input type (id and license id can be long; name, type and contact information can be strings). Then this applicant output all this information as a package to the license service, which obtains license from the license phase. With the information from license phase and applicant phase, the license service can request a review to see whether the license can be granted or not.

Several relationships exist in the diagram:

1. Composition: An association representing a whole-part relationship. The relationships between license and license service, license request and license service, under review decision and license service are all in this type.
2. Import: A relationship between packages, indicating that one package includes all the definitions of another. The license, license request and applicant have an import relationship with their repositories.
3. Association: A relationship between the members of two classifiers. The relationship between applicant and license service is in this type

1.2 Domain diagram for AnalysisEngine

This domain model is similar with the above one. With the features of analysis engine in mind, I designed a domain model for a language translator as example of analysis engine here, shown in the same file as above. This analysis engine is composed of several translators (from English to Chinese, Spanish and French, respectively) and their paired annotators.

The working scenario is: the Chinese translator use English as input and it outputs Chinese annotations together with the raw data. And the finally, all annotations will be combined with the raw data. Every time this analysis engine input English as data, the translators and annotators get input and the size of the input. And finally, the result will be the integration of both raw data and out annotations.

Several relationships exist in this diagram:

1. Composition: An association representing a whole-part relationship. The relationships between the annotations and the result are all in this type.
2. Import: A relationship between packages, indicating that one package includes all the definitions of another. The relationship between data and Chinese Translator/Spanish Translator/French Translator are in this type.
3. Association: A relationship between the members of two classifiers. The relationships between translators and annotations in different languages fall in this type.

Part 2. Implementing a Named Entity Recognizer with UIMA SDK

After following the instruction of Task 3.1 and 3.2 provided by the homework instruction, I installed UIMA SDK and also created a Maven project from the provide archetype (the newest version, 0.0.9). Then the designed system for annotation was described as below.

2.1 Overview - Annotation

Since all annotations must share two common features, a super annotation class that describes the confidence and the annotator that generates this annotation (that is, casProcessorId) is important to lay a foundation for my system. In this system, the class of Annotation is designed for this purpose and all the other annotations can inherit from this super class Annotation to inherit these two features.

Other common features can also be added into Annotation if they are found later, and in this way, the consistence of system is promised. Begin and end position are also needed but both have already been built within the super class of org.apache.uima.jcas.tcas.Annotation.

2.2 Sentence Annotation

Basically, the system gets the input of a UIMA CAS, which is transformed from a text file given by the file path and name. Then the whole input is broken down to two levels: the sentence level and gene level. The sentence level gives its corresponding annotation, the SentenceAnnotation.

2.3 Gene Annotation

Except the sentence level, for the gene level, there are two types of annotations, Gene True Answer Annotation and Tagged Gene Annotation. Since they own the common features, Gene Annotation was designed as the parent annotation of the above two types of specific annotations.

2.4 Gene True Answer Annotation

In Gene True Answer Annotation, the gold standard answer was extracted from the provided file to generate “true answer” for our evaluable, which is a task-based standard to compare with the system output and give us the preliminary evaluation of the overall performance of the system.

2.5 Tagged Gene Annotation

In Tagged Gene Annotation, the annotation extracted from the provided input file was compared with the golden standard answer to get the recall and precision based on textual similarity.

2.6 Evaluation

This version of design doesn't cover much of the evaluation module. The next step is to call a function that transmits those documents into UIMA CAS shared by all annotators. The other main function of evaluation module is to get the scoring result of all answer candidates.

Two things need to emphasize here:

1. In order to run my system appropriately, you need to add the Program argument **< ./src/main/resources/hw1-jli3-cpe.xml>** in Run Configurations – Arguments.
2. Javadoc was stored under **/11791/hw1-jli3/src/main/resources/docs/javadoc**

The performance I got from this preliminary experiment is:

Precision = (true ∩ tagged) / tagged = 77%

Recall = (true ∩ tagged) / true = 85%

And the **full report of the performance** is as follows:

```
System Performance Evaluation:
Precision: 0.7700397117223121
Recall: 0.8508857467901837

Completed 2 documents; 3142338 characters
Total Time Elapsed: 10928 ms
Initialization Time: 4781 ms

Processing Time: 6147 ms
```

2.7 Answers to the Gene Mention task submission guideline

1. Please identify/describe any machine learning techniques used: HMM was used.

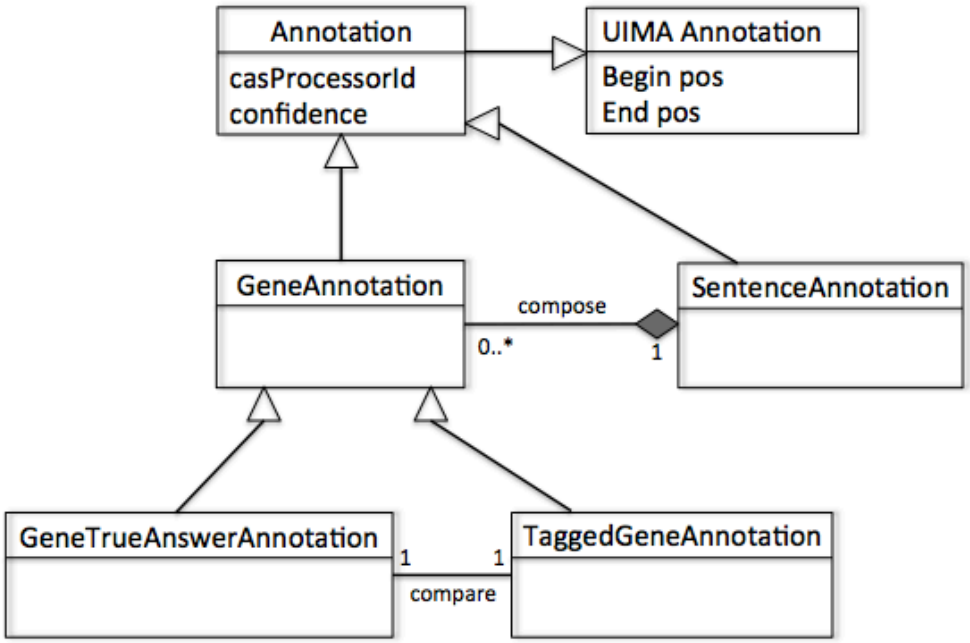
2. For NLP techniques/components, external (marked up text) training data, external lexical resources (terminology lists), rule sets: I didn't use these in this homework, but for the future improvement, I'll consider to include them.

3. If your system interacts with or uses data from any biological database(s), please describe: I use the trained data provided by the system, without external database(s).

4. Please identify/describe any other relevant resources used to train/develop your system: lingpipe HMM API was used to help improve the precision.

5. Please describe the general data flow in your system: the data in my system was transferred in this way: document -> sentences -> genes -> score

Part 3. Class Diagram of the system



Relationships included in this diagram are:

1. Inheritance: the relationship between Annotation and UIMA Annotation or between GeneAnnotation and Annotation are both examples of inheritance.
2. Association: the GeneTrueAnswerAnnotation and TaggedGeneAnnotation are in an association relationship
3. Composition: GeneAnnotation and SentenceAnnotation are in composition relationship.