11-791 Design and Engineering of Intelligent Information System & 11-693 Software Methods for Biotechnology Homework 2

Logical Architecture and UIMA Analysis Engines Design & Implementation

Important dates

- **Hand out: February 14.**[a]
- **Turn in: February 28.** In this homework, you will implement the logical architecture for the sample information processing task you have been working on since last homework. Based on a standard type system we give you in the new archetype, you will need to create analysis engines and annotators to annotate the sample input files. You should organize your project in the same hierarchy as shown below:

```
hw2-ID
|- pom.xml
`- src
   `- main
      |- java
      |  `- **/*.java      /* Java classes generated by JCasGen
      |                        and your UIMA annotators         */
      `- resources
         |- hw2-ID-aae.xml /* your aggregate analysis engine */
         |- **/*.*                /* analysis engine and other resources */
         `- docs
            `- hw2-ID-report.pdf  /* your report for design */
```

[a] This version was built on February 14, 2014

Several notes about organizing your Maven project and other additional information:

1. **Submission:** The same way as you did for Homework 0 and 1 (set up GitHub repo, create Maven project, write your code, submit to Maven repo), except that the name has changed to hw2-ID.

2. **Your report for design:** We expect to see how you design the logical architecture for the sample information system in your report. We will pull out your documents from your jar files. Remember to include your ID as part of file names, and put your name and ID in your document. Please submit in PDF format only.

3. **Javadocs:** Please remember to give an appropriate description for each annotator you create.

4. Please post your questions regarding Homework 2 on Piazza `https://piazza.com/itam.mx/spring2014/com35702`. For other issues or concenrs, you can also send us mails to, Elmer Garduno (`elmer.garduno@itam.mx`).

Task 1

Implementing A Simple Information Processing Task with UIMA SDK

In this task, you need to create analysis engines based on UIMA SDK and create your own Java annotators. Again, you will find *Apache UIMA Manuals and Guides* (`http://uima.apache.org/documentation.html#manuals_and_guides`) will be helpful for you to finish this task. The *Tutorials and Users' Guides* gives you a step-by-step reference to create analysis engines.

Task 1.1 Creating Maven project from the archetype

For this task, we have prepared another archetype to help you quickly build your project. The tutorial for Homework 1 might help you create a Maven project from an archetype.

For Homework 2, you need to add the following Catalog URL

http://oaqa.github.io/DEIIS-hw2-archetype/repository/archetype-catalog.xml

The archetype for Homework is

hw2-archetype

Also remember that the **Group Id** and **Artifact Id** for Homework 2 are

edu.cmu.lti.11791.f13.hw2

and

hw2-ID

with ID being your Andrew Id.

Similarly, you need to edit the `pom.xml` file to type in the SCM information of your GitHub repository. You can see that different from Homework 1, we have also included the type system file with Java classes generated from `JCasGen` that your annotators might need. Your implementation starts here.

Task 1.2 Adding analysis engines for your pipeline

You are required to use the type system we included in the archetype to accomplish this homework. However, if your implementation needs additional data models (or UIMA types) to store intermediate data, you can also extend the type system on your own. You need to implement an aggregated analysis engine (hw2-ID-aae.xml) with several analysis engines that annotate the inputs and evaluate the performance of the aggregate analysis engine by comparing the system outputs with the gold-standard outputs.

We will evaluate your implementation by feeding in unseen inputs and execute your *aggregated analysis engine (hw2-ID-aae.xml)*.

You can test your aggregated analysis engine by running the UIMA Document Analyzer `http://uima.apache.org/d/uimaj-2.4.0/tools.html#ugr.tools.doc_analyzer`. We provided an Eclipse run configuration in hw2's archetype run_configurationUIMA Document Analyzer.launch.

Task 1.3 The Processing Pipeline

The information processing pipeline will consist of the following steps:

1. **Test Element Annotation**: The system will read in the input file as a UIMA CAS and annotate the question and answer spans. Each answer annotation will also record whether or not the answer is correct.

2. **Token Annotation**: The system will annotate each token span in each question and answer (break on whitespace and punctuation).

3. **NGram Annotation**: The system will annotate 1-, 2- and 3-grams of consecutive tokens.

4. **Answer Scoring**: The system will incorporate a component that will assign an answer score annotation to each answer. The answer score annotation will record the score assigned to the answer.

5. **Evaluation**: The system will sort the answers according to their scores, and calculate precision at N (where N is the total number of correct answers).

Here are some suggestions to consider.

1. If you want to employ a resource (e.g., a model you trained offline) in your annotator, you could consider UIMA's *resource manager* (refer to the official tutorial for details about this). Be sure to put your resource in `src/main/resources` so that your submission will also bundle those resources along with your codes.

2. **If you want to incorporate other NLP or machine learning tools, please try to avoid non-Java packages. If you want to dependent your artifact on other Java packages other than those provided by the archetype but can be found in the course repository, you can add a Maven dependency for this artifact, if you have the jar package on your machine, but it doesn't exist in the course repository, please let us know, we will try to deploy them in a 3rd party repository.**

3. Remember to add comments and Javadocs to your annotators, we will also evaluate the quality of your codes.

4. The most creative ideas will happen in the intermediate annotators. We have mentioned some possible solutions to do this. You can try and implement multiple approaches. All the annotations should be kept in the CAS until a final "merging" component takes all the annotations and make a final judgment.

   You can use the type's *casProcessorId* feature to identify, among all the approaches, which annotator is attributed this annotation, and *confidence* feature as a weight to vote for a final decision. This approach was also applied in IBM's Watson system.

Task 1.4 Writing up your report

We expect you to highlight the features of your design and your system. Finally, don't forget to put your name and Andrew ID at the top of the document, name the file as "hw2-ID-report.pdf" and put it under `src/main/resources/docs`.

Grading Guidelines

Designing and Implementing UIMA Analysis Engine (60 pts)

- We will evaluate your system for the necessary IIS steps and its overall structure whether designed and implemented in a proper way, by looking at your codes (Java classes and analysis descriptors) and report. Specifically, each of following items carry 50% design and 50% implementation points: Overall Structure, Test Element Annotation, Token Annotation, NGram Annotation, Answer Scoring, Evaluation and any other annotators you might have added.

- If the basic requirement (i.e., it works!) for each of the items is met, then the student can get upto 21-30 points, otherwise 0-20 points will be given.

- If the design and implementation of each annotator is considered more carefully (according to the suggestions given in the homework instruction or outside), then upto 31-50 pts points will be given. For example, the type system is implemented in an inherited way.
- If you meet the baseline performance you can earn you upto 60 pts in this part of the homework. You will get bonus 30 points if your system's performance is significanly better than baseline.

Documentations, comments, coding style (30 pts)

- We will first look into your report to see if it is complete, if you have given explanations (examples) and solutions to all our concerns, and then we will take a look at the Javadoc package for your project to see if it covers all the necessary information that potential users of your package need to know.
- Basic points are upto 15-20 for report, if some important part(s) is missing, the 0-15 points might be given.
- A detailed documentation can be given upto 30 pts, for example,

  1. good insights to the problem or architecture design covered in the report,
  2. apply design pattern in an appropriate way,
  3. detailed comparison of more than one methods in the report,
  4. interesting discovery,
  5. etc.

Submission, folder structure, and name convention (10 pts)

- We will first check the correctness of your submission (5 pts), and we will look into your report, code and descriptor to see whether they follow the normal Java name convention
- Specifically, we expect your homework is located at the right repository and your project folders/files are organized as required (5 pts). 0 to 4 points might be given if you submitted to wrong path and/or badly organized folder/files.
- The names of your types and files should follow the normal Java name convention (5 pts). Otherwise, 0 to 4 points will be given for bad naming.