11-791 Design and Engineering of Intelligent Information System

Homework 3 Report

Execution Architecture with CPE and

Deployment Architecture with UIMA-AS

Name : Da Teng

AndrewID: dateng

Task 1. Execution Architecture with CPE

1.1 Creating your CPE

To run a CPE, we need to define a collection reader, an analysis engine and a CAS consumer. I defined these necessary components as follows:

- **Collection Reader**

Java File: org.apache.uima.tools.components.FileSystemCollectionReader
Decriptor: src/main/resources/hw3-dateng-CollectionReader.xml

The collection reader is responsible for reading documents in the input folder and passes document content to analysis enginee.

- **Analysis Engine**

Java File: src/main/java/edu.cmu.deiis.annotators.* :
src/main/java/edu.cmu.deiis.types.*
Decriptor: src/main/resources/hw2-dateng-aae.xml

The analysis engine is responsible for annotating the document passed in by collection reader and generating analysis result. For this task, I use the aggregate analysis engine from HW2, which calculates the score for each answer sentence.

- **Cas Consumer**

Java File: src/main/java/edu.cmu.deiis.cpe.CasConsumer.java
Decriptor: src/main/resources/hw3-dateng-CasConsumer.xml

The Cas consumer is responsible for exporting the result of analysis engines. In this homework, we simply use the Cas consumer to print the result of analysis engines to the console.

- CPE descriptor

Decriptor: src/main/resources/hw3-dateng-CPE.xml

This descriptor is an xml file telling the CPE GUI how to run a certain CPE task. In this descriptor, the three parts stated above are combined to run a CPR task.

1.2 Result of the CPE

```
+Booth shot Lincoln.: 1.0
-Lincoln shot Booth.: 0.5
+Booth assassinated Lincoln.: 0.3333333333333333
-Lincoln assassinated Booth.: 0.3333333333333333
+Lincoln was shot by Booth.: 0.25
-Booth was shot by Lincoln.: 0.25
+Lincoln was assassinated by Booth.: 0.16666666666666666
-Booth was assassinated by Lincoln.: 0.16666666666666666
Precision at 4: 0.5


+John loves Mary.: 1.0
+John loves Mary with all his heart.: 0.3333333333333333
-Mary doesn't love John.: 0.2222222222222222
-John doesn't love Mary.: 0.2222222222222222
+Mary is dearly loved by John.: 0.13333333333333333
Precision at 3: 0.6666666666666666


Average Precision: 0.5833333333333333
```

The figure above shows result of the CPE we defined in 1.1. The program prints the precision for each document and prints an average precision for all documents when CPE finish processing them. This information is output by the CasConsumer. To print "Precision at N" for each document, I use the processCas(CAS aCAS) method in CasConsumer. To print "Average Precision" for all documents, I use the collectionProcessComplete method in the CasConsumer.

Task 2. Deployment Architecture with UIMA-AS

2.1 Creating an UIMA-AS client

In this task, we need to use the service provided by StanfordCoreNLPAnnotator.

2.1.1 Add Maven dependency

To achieve this, we need to first add dependency to our maven project, so that we can use the types of the remote service in our local classes. We can use the codes in handout to import dependencies of cleartk-stanford- corenlp and uimaj-as-activemq to our maven project.

2.1.2 Create SCNLP client

Java File: src/main/java/edu.cmu.deiis.as.scnlpClient.java
Decriptor: src/main/resources/scnlp-dateng-client.xml

A UIMA AS client sents Cas to the remote service and receive annotations in a returned Cas. One thing worth mentioning is that I uses sendAndReceiveCAS() menthod in my SCNLP client, so that the program will wait until a result is returned from the AS service. In this way, I can make sure that the program has the result from SCNLP service and can use the result to help calculating answer scores.

2.1.3 CPE result after combining result from SCNLP service

In my aggregate engine for HW3(hw3-dateng-aae.xml), I use following method to combine result from CPE in answer scoring:

If all the PERSON NAMEs shown in the question sentence appear in an answer sentence, add 0.1 to the score of that answer sentence.

After applying this rule, the CPE result is as follows,

```
******UIMA AS client send request to service********
*****UIMA AS client receive HW2 AEE Annotations*****
+Booth shot Lincoln.: 1.1
-Lincoln shot Booth.: 0.6
+Booth assassinated Lincoln.: 0.43333333333333335
-Lincoln assassinated Booth.: 0.43333333333333335
+Lincoln was shot by Booth.: 0.35
-Booth was shot by Lincoln.: 0.35
+Lincoln was assassinated by Booth.: 0.26666666666666666
-Booth was assassinated by Lincoln.: 0.26666666666666666
Precision at 4: 0.5


******UIMA AS client send request to service********
*****UIMA AS client receive HW2 AEE Annotations*****
+John loves Mary.: 1.1
+John loves Mary with all his heart.: 0.43333333333333335
-Mary doesn't love John.: 0.3222222222222222
-John doesn't love Mary.: 0.3222222222222222
+Mary is dearly loved by John.: 0.23333333333333334
Precision at 3: 0.6666666666666666


Average Precision: 0.5833333333333333
```

After applying the rule stated above, the precisions are the same with those in HW2. But since we call SCNLP service before running the aggregate analysis engine, speed is slower than that in HW2.

2.2 Deploying your own UIMA-AS service

In this task, I use the aggregate analysis engine from HW2, which dose not use information from the SCNLP service to facilitate answer sentence scoring.

2.2.1 Create deployment descriptor

Decriptor: src/main/resources/hw2-dateng-aae-deploy.xml

This descriptor defines that we are going to deploy the aggregate enginee hw2-dateng-aae.xml as a service.

2.2.2 Setup environment and start UIMA AS broker

In order to run the scripts for UIMA AS service, we need to setup JAVA_HOME and UIMA_HOME variables. In my mac computer, the commands are:

```
export  JAVA_HOME=$(/usr/libexec/java_home)
export  UIMA_HOME=~/Download/apache-uima-as-2.4.0
```

Then we can go to $UIMA_HOME/bin and run *startBroker.sh* to start broker.

2.2.3 Add maven dependency to UIMA classpath

First, use maven assembly plugin to generate a jar containing all maven dependency files. Add following content to pom.xml and run *mvn clean compile assembly:single* to get the jar containing all dependencies.

```
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <archive>
        <manifest>
          <mainClass>fully.qualified.MainClass</mainClass>
        </manifest>
      </archive>
      <descriptorRefs>
        <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
</plugin>
```

Then add the generated jar to UIMA_CLASSPATH. In my computer, the command is:

export UIMA_CLASSPATH =/User/tengda/workspace/hw3-dateng/target/

2.2.4 Deploy UIMA service

Go to $UIMA_HOME/bin and run deployAsyncService.sh to deploy a UIMA AS service. We need to give a deployment descriptor and a broker url as parameters. After deploying, the result is like following:

```
tengdatekiMacBook-Air:bin tengda$ ./deployAsyncService.sh /Users/tengda/Documents/workspace/hw3-dateng/src/main/resources/hw2-dateng-aae-deploy.xml -brokerURL tcp://tengdatekiMacBook-Air.local:61616
>>> Setting defaultBrokerURL to:tcp://tengdatekiMacBook-Air.local:61616
Service:hw2-dateng-aae Initialized. Ready To Process Messages From Queue: hw2AAEqueue
Press 'q'+'Enter' to quiesce and stop the service or 's'+'Enter' to stop it now.
Note: selected option is not echoed on the console.
```

2.2.5 Test of the deployed service

We can use the runRemoteAsyncAE.sh script to check our serive by pass a broker url, service endpoint and a collection reader as parameters. The result in my computer is shown below,

```
tengdatekiMacBook-Air:bin tengda$ ./runRemoteAsyncAE.sh tcp://tengdatekiMacBook-Air.local:61616 hw2AAEqueue -c /Users/tengda/Documents/workspace/hw3-dateng/src/main/resources/hw3-dateng-CollectionReader.xml
UIMA AS Service Initialization Complete
..Completed 2 documents; 459 characters
Time Elapsed : 1884 ms
```