11-791 HW3: Execution Architecture with CPE and Deployment Architecture with UIMA-AS

Xiaohua Yan

Language Technologies Institute

In this homework, we are required to execute my pipeline from Homework 2 with a Collection Processing Engine (CPE) and deploy it as a service with UIMA-AS.

Task 1

Task 1 is about learning the basics of CPE and how to use it. Basically, CPE has 3 main components: Collection Readers, CAS Initializers and CAS consumers. The collection reader for this task is already provided for the input folder located on the file system, and the Collection Reader descriptor (named `CollectionReader.xml`) can be created directly based on that.

Concerning the CAS consumer, I have 2 CAS consumers named `DocAnnotationCasConsumer` and `XmiWriterCasConsumer` respectively doing different jobs to improve the cohesion of the information system. To be specific, the `DocAnnotationCasConsumer` is built based on the Evaluator Component of my Homework 2, which prints the scores of each answer for each question in the input files and the final precision evaluated at the total number of correct answers. The output of this CPE is as follows:

```
Question: Booth shot Lincoln?
+ 1.00 Booth shot Lincoln.
- 0.80 Lincoln shot Booth.
- 0.62 Booth was shot by Lincoln.
+ 0.62 Lincoln was shot by Booth.
- 0.53 Lincoln assassinated Booth.
+ 0.53 Booth assassinated Lincoln.
- 0.41 Booth was assassinated by Lincoln.
+ 0.41 Lincoln was assassinated by Booth.
Precision at 4: 0.50

Question: John loves Mary?
+ 1.00 John loves Mary.
+ 0.63 John loves Mary with all his heart.
- 0.46 John doesn't love Mary.
- 0.46 Mary doesn't love John.
+ 0.38 Mary is dearly loved by John.
Precision at 3: 0.67
Average precision: 0.58
```

Using the CPE Configurator, the CPE descriptor for my CPE pipeline is created (see the file named `hw3-xiaohuay-CPE.xml`).

Task 2

In this task, we are required to integrate a remote UIMA-AS service (Stanford CoreNLP and its Named Entity Recognizer) into our CPE pipelines. Based on the scoring method (cosine similarity) in my Homework 2 source files, I modified my `ScoreAnnotator` by adding a named entity overlap score into the total matching score between a (question, answer) pair. In particular, the scoring annotator generates a score for each answer annotation, based on token overlap, n-gram overlap, and named entities overlap scores measured by cosine similarity. The weights for token overlap, n-gram overlap and NE overlap are 0.7, 0.2 and 0.1 respectively. Finally the output of the CPE using the scnlp-client I created is displayed as follows (the elapsed time is 1.237 seconds):

```
Question: Booth shot Lincoln?
+ 1.00 Booth shot Lincoln.
- 0.80 Lincoln shot Booth.
- 0.64 Booth was shot by Lincoln.
+ 0.64 Lincoln was shot by Booth.
- 0.57 Lincoln assassinated Booth.
+ 0.57 Booth assassinated Lincoln.
+ 0.46 Lincoln was assassinated by Booth.
- 0.43 Booth was assassinated by Lincoln.
Precision at 4: 0.50

Question: John loves Mary?
+ 1.00 John loves Mary.
+ 0.63 John loves Mary with all his heart.
- 0.50 John doesn't love Mary.
- 0.50 Mary doesn't love John.
+ 0.43 Mary is dearly loved by John.
Precision at 3: 0.67

Average precision: 0.58
```

As can be observed from the output above, the average precision did not change while the score for each answer has a minor change, which did not affect the overall precision because the mention types of all the recognized named entities are "PERSON" and the same named entities appear in both the question and the answer.

In the next sub-task, I deployed my aggregate analysis engine on my own machine using the deployment descriptor (`hw2-xiaohuay-aae-deploy.xml`). A client descriptor (`hw2-xiaohuay-aae-client.xml`) is also created to test the my service (I used `maven plungins dependency:copy-dependencies` to copy my dependencies into the `target` folder).

Bonus

1. I ran a Stanford CoreNLP that annotates named entities and then ran the same aggregate analysis engine to compare the speed with the CPE that calls the remote service. The result is the runtime for the local CPE is 11.891 seconds while the runtime of the remote one is 1.237 seconds.
2. I incorporated other annotations from Stanford CoreNLP using the remote service. The output is as follows (the weights for token overlap, n-gram overlap, POS overlap, lemma overlap, named entity over are set to be 0.5, 0.1, 0.1, 0.1 and 0.1 respectively).

```
+ 0.95 Booth shot Lincoln.
- 0.75 Lincoln shot Booth.
- 0.65 Booth was shot by Lincoln.
+ 0.65 Lincoln was shot by Booth.
+ 0.60 Booth assassinated Lincoln.
- 0.58 Lincoln assassinated Booth.
+ 0.49 Lincoln was assassinated by Booth.
- 0.47 Booth was assassinated by Lincoln.
Precision at 4: 0.50

Question: John loves Mary?
+ 1.00 John loves Mary.
+ 0.64 John loves Mary with all his heart.
- 0.52 John doesn't love Mary.
- 0.52 Mary doesn't love John.
+ 0.49 Mary is dearly loved by John.
Precision at 3: 0.67

Average precision: 0.58
```

3. I successfully used a remote service deployed by Zhengzhong (Hecotr) Liu using the client descriptor named `dbpediaService-client.xml`, which annotates the input documents with features available in DBpedia. The broker URL of this service is `tcp://128.237.200.33:61616` and the endpoint is `dbpediaQueue`.