11-791 HW 3 Report

Chenyan Xiong
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
cx@cs.cmu.edu

1 Task 1 Execution Architecture with CPE

In this task, I run my pipeline implemented in last homework within CPE framework. There are three components implemented in the very basic CPE framework: collection reader, analysis engine and cas consumer.

Collection Reader In this homework, as described in homework requirement, the requirement for collection reader is just to read a folder of files on local file system. And the org.apache.uima.tools.components.FileSystemCollectionReader satisfies our needs. Thus I directly use it as my collection reader for this homework.

Cas consumer Converting the evaluator in UIMA framework to Cas consumer is rather simple. There are only two parts to be modified: the inherited class changes to CasConsumer_ImplBase, and the parameter for processCAS() is CAS type. Fortunately, CAS type has a function to provide JCas result, and could be transferred easily.

Using this feature, we could re-use most of our evaluator codes in previous homework.

Running CPE Running CPE is very straightforward, as a nice GUI is provided to connect all components. We could easily add each steps' descriptors using the GUI and run them together. The XML file of CPE running configuration can be easily saved by the saving function in CPE's GUI.

2 Deployment Architecture with UIMA-AS

UIMA-AS is an architecture used to do server-client type UIMA processing. It contains components to deploy service and client part to call deploy service.

2.1 Creating an UIMA-AS client

The first task is to create an UIMA-AS client to call the UIMA-AS server provided. The UIMA-AS-client descriptor I make is modified from UIMA-AS example provided. The only modification required is listed in homework: changing brokerURL and endpoint.

2.2 Using Stanford Core NLP name entity annotator

My way of using Stanford core NLP name entity annotator is to treat NER as another level of annotation, similar to N-gram. The score of NER is calculated by the cosine similarity between question and answer, and is then merged with Ngram cosine similarity. Because only two training data is available, there is no good way to train the weights of combination. Thus I simply use uniform weights for NER similarity and cosine similarity. The performance of using NER is relatively the

same with not using. There is a slight gain, but as there is only two testing data, I believe it won't pass signification test. The speed is also almost the same.

2.3 Deploying UIMA-AS service

The deploying does not require any coding. And mostly works are about writing descriptors. Fortunately, most of them are available in the UIMA example project, and modifying them to local host is simple. The only tricky point is the class dependency in local broker need to be set correctly. We need add to UIMA_CLASSPATH the MVN target file and exportedDependencies of installed MVN dependencies. Then we could simply using the provided bash file startBroker.sh to start the Broker, and using the deployAsyncService.sh to deploy the deployment descriptor.