Execution Architecture with CPE and Deployment Architecture with UIMA-AS

Name: Qichen Pan AndrewID: pqichen

Introduction

In this homework, I am required to execute the former aggregate analysis engine with a UIMA CPE pipeline and be familiar with UIMA AS client and server. I firstly rewrite some codes in Evaluator annotator to serve as the CasConsumer in CPE and perform the whole pipeline with the CollectionFileReader, Aggregate Analysis Engine and CasConsumer. Then I implement a UIMA AS client to take advantage of StanfordNLP server. I retrieve NameEntity from that server to enhance my AnswerScoring method. I perform another CPE pipeline with an additional analysis engine StanfordClient. The last thing I do is deploying my analysis engine on my local machine and use a new CPE pipeline to test it.

Task 1 Creating and Running your CPE

In this task I simply use my hw2-pqichen-aae as the analysis engine, and shift the Evaluator to the CasConsumer part. So the whole CPE pipeline has the structure shown below.

	Compoents
Collection Reader	FileSystemCollectionReader
Analysis Engine	hw3-pqichen-aae
CAS Consumer	CasConsumer(Evaluator.java)
CPE Descriptor	**hw3-pqichen-CPE**

Form 1.1 CPE structure in Task 1

It has the same function as the hw2-pqichen-aae. It takes text files as input, calculates precisions of answer scoring method and print out in the Eclipse console. The only thing not trivial in this task is to implement CAS Consumer as an inherited class from CasConsumer_ImplBase. It takes CAS as input instead of JCas.

Task 2.1 Creating an UIMA-AS client

In this task I create a client of StanfordNLP server called NameEntity.xml. I firstly create an Aggregate Analysis Engine Descriptor StanfordNLP_Client.xml to generator the actual client file NameEntity.xml. This file will send input to remote server and retrieve output from that server. In this case, I decide to use the NameEntityMention annotator to enhance the performance of my analysis engine. (In fact it does not contribute any valid improvement to my former engine.)

I also create a new CPE pipeline to test the performance of this UIMA AS Client. The structure of the CPE is shown below.

	Compoents
Collection Reader	FileSystemCollectionReader
Analysis Engine	StanfordNLP_Client -> NamedEntity
	hw3-pqichen-aae-withNamedEntity
CAS Consumer	CasConsumer_withNamedEntity(Evaluator_withNamedEntity.java)
CPE Descriptor	**hw3-pqichen-CPE-withNamedEntity**

Form 2.1 CPE structure in Task 2.1

The flow of CPE pipeline is also shown below.

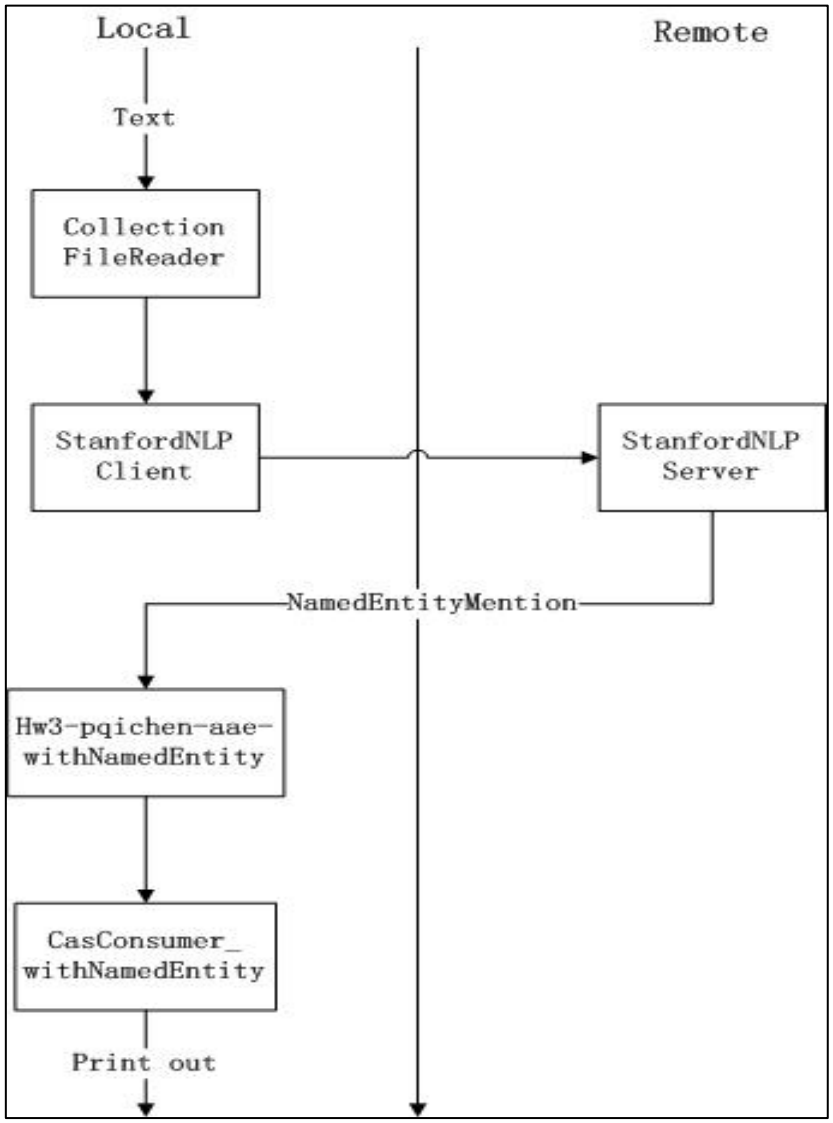

Figure 2.1 Pipeline flow of CPE with StanfordNLP

In this task, the system retrieves NamedEntityMention annotator from StanfordNLP and uses it to come up with a new method to score answers. The details of implementation store in NamedEntityAnswerScorer.java. System pipeline gets all name entities in question and answers and calculate the overlap rate of them to assign relative scores to each question. The overlap of name entities provides little (if any) improvement to our previous procedures. Because “PERSON” which as a name entity appears in both question and answer makes little sense on the

correctness.

Task 2.2 Deploy your own UIMA-AS service

In this task, I am supposed to deploy my aggregate analysis engine as a remote UIMA AS server, and create a UIMA AS client to test it.

The test method is through CPE pipeline. The structure of this pipeline is shown below.

	Compoents
Collection Reader	FileSystemCollectionReader
Analysis Engine	StanfordNLP_Client -> NamedEntity
	hw3-pqichen-aae-clientDescriptor -> hw3-pqichen-aae-client
CAS Consumer	CasConsumer_withNamedEntity (Evaluator_withNamedEntity.java)
CPE Descriptor	**hw3-pqichen-aae-as-CPE**

Form 2.2 CPE structure in Task 2.2

The flow of CPE pipeline is also shown below.

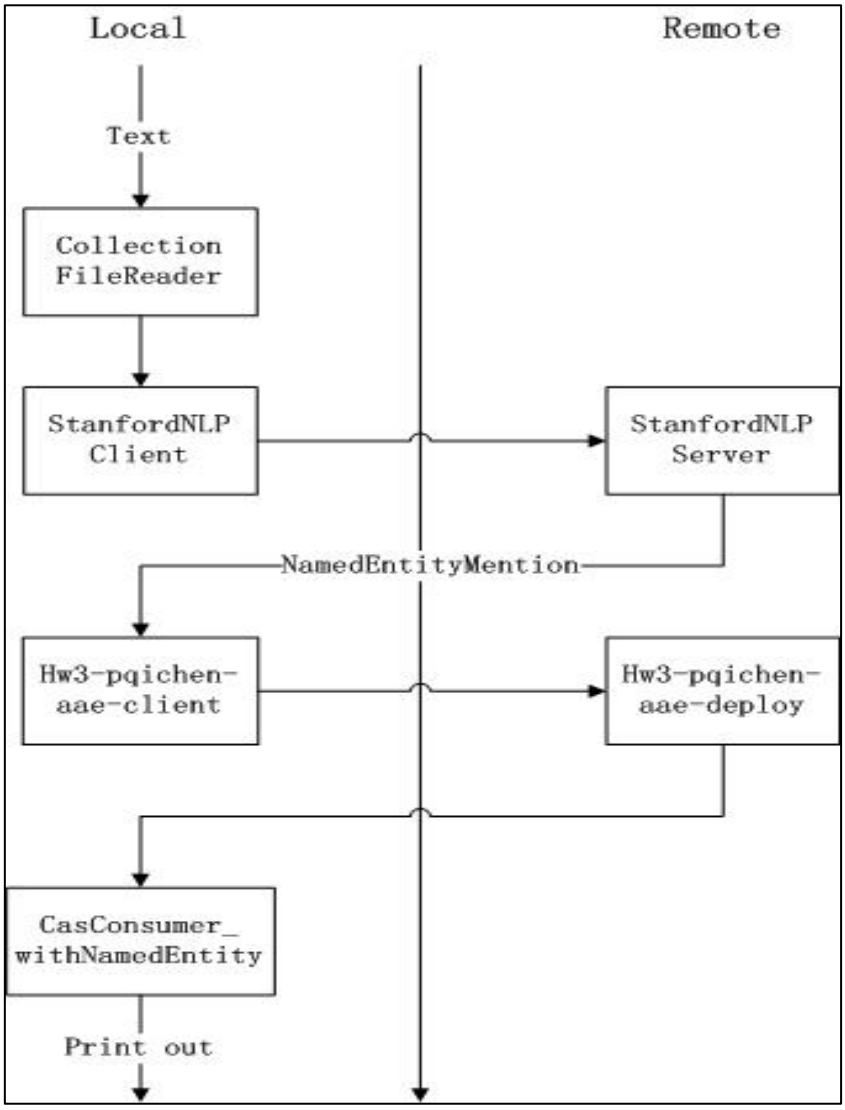

Figure 2.2 Pipeline flow of CPE with UIMA AS server

First of all, I create a UIMA AS deploy descriptor called ‘hw3-pqichen-aae-deploy’ to maintain the server information including endpoint and brokerURL. Then I create an aggregate analysis engine named ‘hw3-pqichen-aae-clientDescriptor’ to link to a remote analysis engine. This descriptor generates the actual client XML file ‘hw3-pqichen-aae-client’. The way to test the whole process is to put all these components into a new CPE ‘hw3-pqichen-aae-as-CPE’. The output of this CPE is the precisions of 4 different answer scoring methods on the console.

Comparison of Different Design Methods

The performance of 4 different answer scoring methods is shown respectively below.

Precision	John loves Mary	Booth shot Lincoln
GoldAnswerScorer	1.000	1.000
TokenOverlapAnswerScorer	0.333	0.500
NGramOverlapAnswerScorer	0.667	0.500
NamedEntityAnswerScorer	0.667	0.250

Form 3 Comparison of precisions from different methods

From the data we can obtain the observation that the NamedEntityAnswerScorer method can hardly do anything helpful to improve the precision of our prediction. The reason is that only the type of entity has little valuable information. If I can combine it with some parse tree or semantic analysis tools, there may be reasonable benefits of our whole system, for example, some tools that can predict passive relations.