Homework 3

Salvador Medina
139323
DEIIS, ITAM

April 5, 2014

1 CPE

For the first part of the homework it was required to create a CPE based on the AAE created for homework 2. As the *consumer reader* we could use UIMA's *FileSystem-CollectionReader* and create its description file.

1.1 Implementation

In this particular case, we created the CPE as shown in Figure 1 in the descriptor file *hw3-ID-aae-as-CPE.xml*. This CPE takes as input the set of files given in homework 2 through the *FileSystemCollectionReader*. Each of the files then are fed to the aggreagated analysis engine developed for the previous homework. Finally the result are saved into an output file per input file under the same name but in a different folder.

For the final task we had to develop a consumer named *printConsumer* and update the type system.

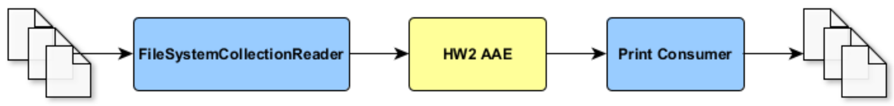


Figure 1: Resulting CPE

1.2 Print Consumer

For our collection consumer we had to add a type into our type system which was able to annotate the results obtained from calculting the Precision@n. Where N denotes the top results obtained from our analysis engine. Therefore, a *Result* type was added. This type holds two attributes as seen in Figure 2:

N: Number of top elements to consider

Precision: Calculated considering N

With this in mind, in the *PrintConsumer.java* all the *Result* annotations are taken from the CAS and printed into a file in the output directory that was established for the consumer.

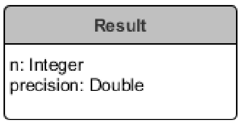


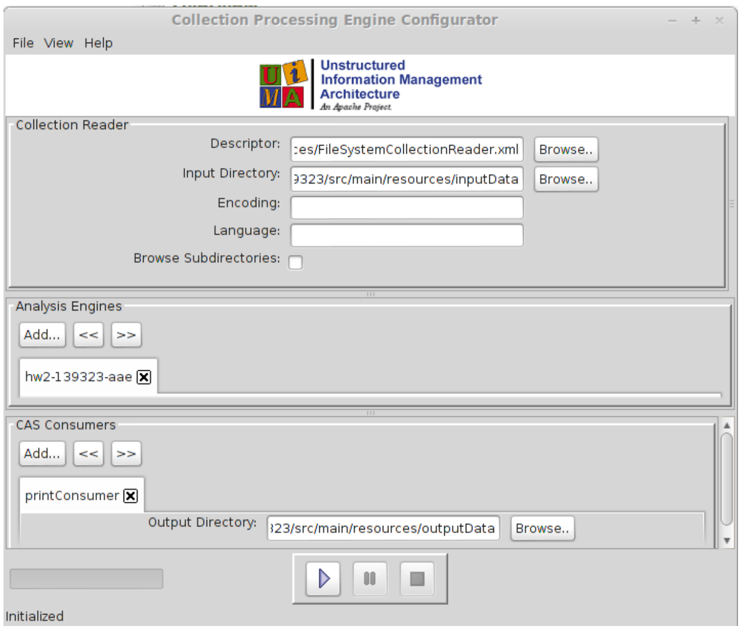
Figure 2: Result Type

1.3 Execution

The CPE GUI tool was executed from Eclipse by running the *UIMA CPE GUI* running configuration which was already established in the project. From there we fed the dialog shown in Figure 3 with the *FileSystemCollectionReader* descriptor, the aggregate analysis engine descriptor and the newly developed *PrintConsumer* descriptor. We set the input and output paths and clicke on the Play button.

From there the CPE executed showing the results in a dialog shown in Figure 4. Besides the shown results, we could also check that the expected output files were created.

This tool was used to create the CPE descriptor file *hw3-139323-cpe.xml* and imported into the Eclipse project.



Collection Reader
Descriptor: :es/FileSystemCollectionReader.xml
Input Directory: ɜ323/src/main/resources/inputData
Encoding:
Language:
Browse Subdirectories: ☐

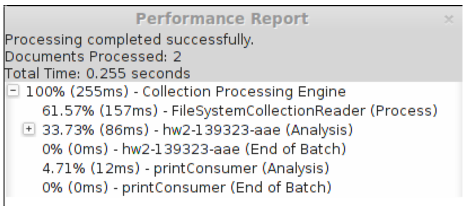
Analysis Engines
hw2-139323-aae

CAS Consumers
printConsumer
Output Directory: ɜ23/src/main/resources/outputData

Initialized

Figure 3: CPE GUI example of execution

The output files produced by the *PrintConsumer* output are stored in [PROJECT] /src/main/resources/outputData/ and look as shown here. In the result file the + denotes that the answer is correct and the number denotes the score given by our analysis engine. Below shows all the results calculated from the scores obtained from the aforementioned process.



```
Performance Report
Processing completed successfully.
Documents Processed: 2
Total Time: 0.255 seconds
⊟ 100% (255ms) - Collection Processing Engine
     61.57% (157ms) - FileSystemCollectionReader (Process)
   ⊞ 33.73% (86ms) - hw2-139323-aae (Analysis)
     0% (0ms) - hw2-139323-aae (End of Batch)
     4.71% (12ms) - printConsumer (Analysis)
     0% (0ms) - printConsumer (End of Batch)
```

Figure 4: CPE GUI execution results

2 UIMA AS

The UIMA AS part was troublesome due to installing the version 2.4.2 instead of the stable version 2.4.0. This brought troubles such as having multiple-binding problems with the SLF4J therefore not being able to watch the correct log. Nonetheless, the most crucial problema was not being able to deploy remotely the AAE to the broker, without having any possibility of trying the Stanford Queue.After overcoming the system installation, an AS could be deployed and executed remotely.

2.1 Deployment and Execution

For the execution, a set of script files were generated for each stage of the remote AAE AS execution, all of them are located in the root directory of the project. To start with, the broker was executed locally by calling *startBroker.sh* provided with the UIMA AS SDK. Once the broker was executing locally, the AAE was deployed as an AS by using the tool *deployAsyncService.sh* provided with the SDK:

deployAsyncService.sh ./src/main/resources/hw2-139323-aae-deploy.xml -brokerUrl tcp://localhost:61616

Where *hw2-139323-aae-deploy.xml* is the deployment descriptor file.

Afterwards, the service was executed by using *runRemoteAsyncAE.sh* as described in the script file *runAS.sh*. This returned an XMI file per document in the input which describes all the annotators generated by the remote AAE.

The client could also be deployed programmatically by following the UIMA AS tutorial having as a result 2 *java* files. The first one is *Pipeline.java* which is in charge of deploying and executing the remote AS. The other one contains the main program called *Application.java* which generates an instance of of the Pipeline class, through which the program deploys and executes our AAE from HW2.

2.2 Stanford Core NLP

The annotator which uses the Stanford Core NLP (SCNLP) is described by *scnlp-139323-client.xml* and implemented in edu.cmu.deiis.tools.StanfordCoreNlpAnnotator.java. It is based in the one mentioned in the handout, however adapted to our needs. Its main task is to extract the sentences from the document, tokenize them and extract for each token the POS, lemma and name entity. For this purpose a *Sentence* type was added into our type system as well as the fields *POS*, *lemma* and *ne* to the *Token* type. In this annotations all the info extracted from the SNCLP is stored.

Token	POS	Lemma	N.E.
Mary	NNP	Mary	Person
do	VBZ	do	0
not	RB	not	0
love	VB	love	0
John	NNP	John	Person

Table 1: NNP: Proper Noun, VBZ: Verb 3rd pers sing pres, RB: Adverb, VB: verb base form

AE	Time (ms)
HW2	3104
SCNLP	6005

Table 2: Execution Time

The annotator does not distinguish a Question from an Answer, it takes the whole document and extracts the annotations that it finds while processing the whole file.

After executing the AE it was a nice surprise to obtain such good results while analyzing the sentences, for example for a particular sentence: "**Mary doesn't love John**" from the *q002.txt* we obtained the results of Table 1.

This results were obtained from a manual analysis of the XMI obtained, no consumer was implemented to interpret this results.

2.3 Remote execution

The remote session using the service provided nor the service deployed into the server **mu.lti.cs.cmu.edu** could not be done. The server session expired when trying to run the service using a *FileSystemCollectionReader*.

When trying to deploy our annotators (hw2 and the SCNLP based) the AE's could de deployed but while trying to use the service, the sessions also expired.

However, the execution times obtained from executing the HW2 AAE and the AE as an AS locally are shown in Table 2