IIS — hw1

Chu–Cheng Lin `chuchenl`

September 2014

1 UML Design

See Fig. 1 and Fig. 2.

2 System Description

At its core, the system uses a statistical named entity recognizer provided by Lingpipe. The recognizer is HMM based and pre–trained.

The system is rather simple. There is a collection reader called `MyCollectionReader`, an annotator called `GeneAnnotator`, and a consumer called `MyConsumer`. There is also a wrapper class for the Lingpipe package we used, called `Chunker`. Following is the description of the components.

`MyCollectionReader` reads the input, which is specified in the descriptor by the parameter name `InputFileName`. The specified file is opened and read in `initialize`. There is a design decision of whether the whole file should be loaded in advance or each line should be extracted from the stream on the fly. I choose the former approach since `MyCollectionReader` needs to implement the abstract method `getProgress` of the convenient base class `CollectionReader_ImplBase`. If the content is read from the stream, it would be hard to know the current location in the file. However, loading the whole file can be costly; so if loading a huge corpus is required, something more sophisticated should be implemented. For now the whole file is loaded into a `List`, and `MyCollectionReader#getNext` gets the content of the next line in the file through an iterator on that `List`. We pass on the line as a `Sentence` annotation, which has two self–explanatory properties `id` and `text`.

`GeneAnnotator` has an `initialize` method where it initializes the wrapper `Chunker c` with the model file path specified in the configuration parameter `ModelPath`. The constructor of `Chunker` in turn loads the specified model file using the API provided by Lingpipe. The `process` method retrieves the `Sentence`s of the input annotation. The `text` property of every `Sentence` is fed into the named entity recognizer through the wrapper metho `Chunker#chunk`, which returns a `Set` of chunked named entites. For each recognized named

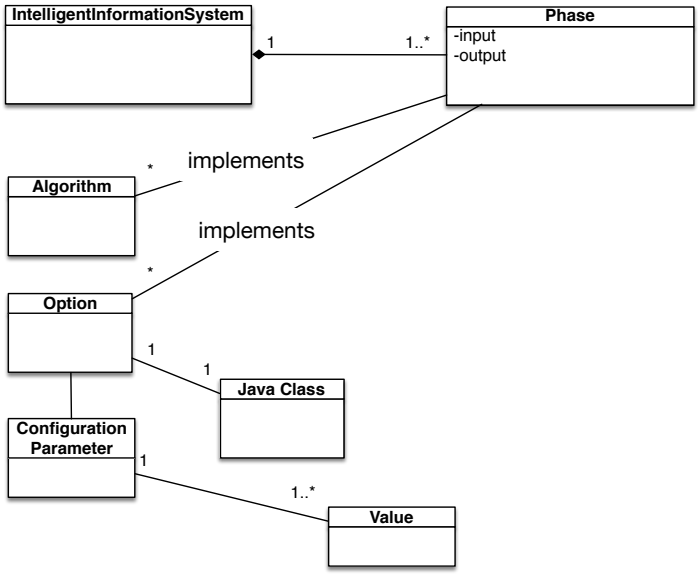

Figure 1: Task 1.1

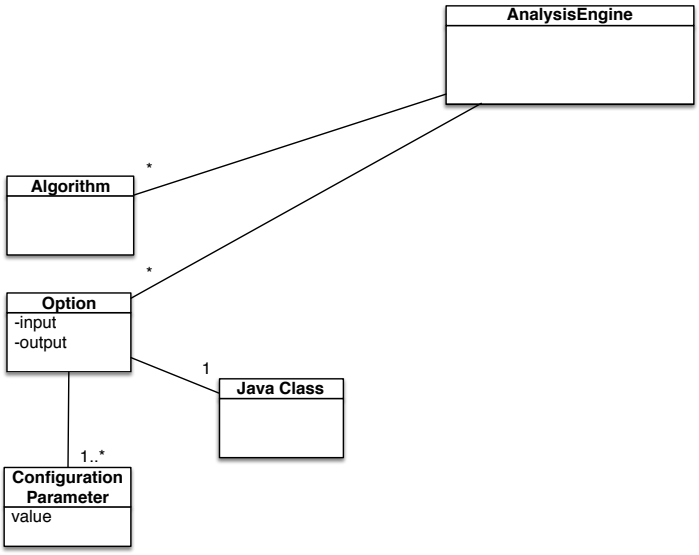

Figure 2: Task 1.2

entity, `GeneAnnotator` creates a new annotation `GeneAnnotation`, which contains the properties `id`. We use the inherited properties `begin` and `end` to store offsets, which we also compute in `GeneAnnotator`. While the homework description asks to compute the offset in the CAS consumer, I find that would introduce unnecessary dependency on `MyConsumer` — to compute the offset requires both the sentence and the chunked named entity. By computing the offset in `GeneAnnotator`, `MyConsumer` does not need to know about `Sentence` annotations, which may facilitate further maintenance.

`MyConsumer` gets the specified output filename from the configuration parameter `OutputPath` in the method `initialize`. It retrieves the `GeneAnnotation`s associated with an annotation, and and writes into the output file in the specified format.