11-791: Design and Engineering of Intelligent Information Systems

Project Individual 3: UIMA Analysis Engines

Keith Maki (andrewID: kmaki)

September 21, 2015

1 Introduction

In this report, I present my design and implementation for the analysis model pursuant to the PI3 sample information processing task outlined in figure 1. The report is structured as follows: section 2 describes the overall analysis engine pipeline and design; section **??** provides discussion of important design considerations and limitations to the selected approach; section **??** presents the UIMA Aggregate Analysis Engine Descriptor and other relevant project files; and section **??** concludes the report.

2 Analysis Engine Design Process

I present an analysis framework following the high-level pipeline diagram shown in Figure 1. The UIMA framework makes it easy to sequentially chain arbitrarily many analysis engines to process documents in a modular way, building new annotations with the help of those from previous layers. I implemented a single primitive analysis engine for each of the five major analysis steps in the pipeline. These are described in detail below. The five analysis engines were then linked with a single aggregate analysis engine which instantiated the component engine flow shown in Figure 1. The steps in the flow incrementally add annotations to a CAS as follows:

1. Test Element Annotator In this stage, the document is parsed according to the major sections visible in the example input in figure 2. The question(s) and corresponding answer choice(s) present in the document are identified using a regex matching

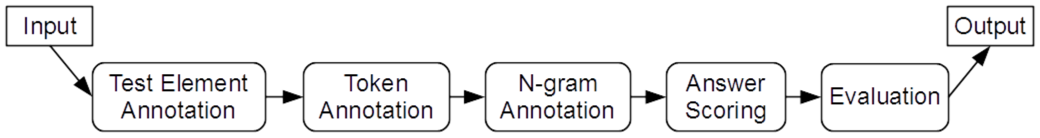

Figure 1: High-level architecture of the system implemented by this project.

```
Q Booth shot Lincoln?
A1 1 Booth shot Lincoln.
A2 0 Lincoln shot Booth.
A3 1 Lincoln was shot by Booth.
A4 0 Booth was shot by Lincoln.
A5 1 Booth assassinated Lincoln.
A6 0 Lincoln assassinated Booth.
A7 1 Lincoln was assassinated by Booth.
A8 0 Booth was assassinated by Lincoln.
```

Figure 2: Example input question with eight answer choices.

scheme that identifies the appropriate character sequences at the start of each line. These spans are instantiated as Questions or Answers using the defined UIMA Type System, and provided with the appropriate annotational meta data (e.g. for an answer, this includes the answer choice number, whether the answer was correct or not, the isolated text of the answer choice, and the indices into the text which identify the raw input which is being annotated.

2. Tokenization Annotator In this stage, each portion of the document (question and answers) is tokenized using Java's StringTokenizer utility class. Each token is given a separate annotation span, and the collections of tokens from each of the portions described above are given collective annotations. Each question in the document is additionally given a single, separate TokenAnnotation which groups its tokenized span and the tokenizations of each of its answer choices.

3. N-gram Annotator In this stage, each of the tokenized spans from before are annotated with ngrams of fixed (parameterizable) length. The parameter is read from the UimaContext when the annotator is initialized.

4. Score Annotator In this stage, each answer choice in the document is assigned a score based on its ngrams and the ngrams of its respective question. Due to time constraints, every such pair is assigned the placeholder score of 0, but it would be easy to implement a TF.IDF metric or another meaningful semantically motivated value using the Ngram annotations of these two spans.

5. Output Annotator In this final analysis stage, the Precision@N for each question is computed and compiled into an output string along with the answer choice rankings and associated scores.

3 Design Considerations

In order to maintain the extensibility of my analysis framework, I was careful to design each module to reference and produce annotations for each Test Element annotated in the document (i.e. a question with its set of answer choices). I make no assumption that there is a single question in a document, nor do I assume anything about the number of answers for each question other than the formatting information provided in the PI3 documentation. The modules do not directly pass inputs and outputs to eachother, and do not make excessive use of configuration parameter passing to circumvent this isolated set-up. Each analysis engine has been designed with a mind for modularity, so it would be easy to develop other modules which behave similarly and produce alternate annotations. These annotations could then be processed by downstream analysis engines and would maintain their "self awareness" in the automatic logging of the module that produced each annotation. This can improve the efficiency of error analysis and allow for ease of management of large multi-stage pipelines.

In developing my first UIMA Analysis Framework, I found it challenging to divide my efforts among the several modules while working under a time crunch. Ultimately I had to make several sacrifices which significantly reduce the usability of my final project. For example, I was unable to complete the top-level CPE Descriptor XML for my design, and I did not prioritize careful testing of my components, so I am unable to conclude much beyond the obvious uncompilability of my project at this time. It is not easy to construct a system of this size from the ground up in a few short days when working alone, but I am looking forward to the latter part of the semester when we will work in teams to better manage our productivity and ensure adequate coverage of the scope of our projects.

4 UIMA Analysis Engine

The full pi3-kmaki project repository including this report be obtained at `https://github.com/kortemaki/pi3-kmaki`.

5 Conclusion

In this report, I presented my UIMA Analysis Engine in pursuit of the requirements of the PI3 assignment. The techniques employed in the design of this system will be integral to designing and implementing more elaborate information systems in later units of this course and in the Team Project assignments in the latter half of the semester. Beyond this, the underlying design principles serve as important building blocks for simple yet effective software engineering which will continue to provide benefits long after this course has been completed.