11-791: Design and Engineering of Intelligent Information Systems

Project Individual 4: Component and Type Design Patterns

Keith Maki (andrewID: kmaki)

September 28, 2015

1 Introduction

In this report, I present my design and implementation for the analysis model pursuant to the PI4 sample information processing task of passage ranking for information retrieval. The report is structured as follows: section 2 describes the overall analysis engine pipeline and design; section 3 provides discussion of important design considerations and limitations to the selected approach; section 4 presents the relevant project files; and section 5 concludes the report.

2 Passage Ranking System CPE Design Process

I present an analysis framework for ranking passages based on relatedness to a question query. I build roughly on the framework from the previous Project Individual, taking an ngram based approach to computing the ranking. 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. The steps in the flow incrementally add annotations to a CAS as follows:

```
1894 QUESTION How far is it from Earth to Mars?
1894 APW19980923.1395 1 hours, JPL said. After five more months of aerobraking each orbit should take less than
    two hours.  Mars is currently 213 million miles (343 million kilometers) from Earth.
1894 APW19990429.0168 -1 ‘‘The data suggests that Mars was once magnetic and was far more similar to Earth’s
    global magnetic field than had been previously assumed,’’ Ness said.
1894 APW19990429.0249 -1 ‘‘The data suggests that Mars was once magnetic and was far more similar to Earth’s
    global magnetic field than had previously assumed,’’ Ness said.
1894 NYT19981211.0308 -1 c.1998 N.Y. Times News Service <TEXT> <P> CAPE CANAVERAL, Fla. _ The first of two NASA
    science probes embarked Friday on a 416-million-mile voyage to Mars, catapulted from Earth by a Boeing Delta
     2 rocket. </P> <P> The spacecraft, called Mar
1894 NYT19990923.0365 1 its farthest point in orbit, it is 249 million miles from Earth. And, so far as anyone
    knows, there isn’t a McDonalds restaurant on the place. </P> <P> And yet we keep trying to get there. Thirty
     times in the past 40 years, man has sent a spacecra
1894 APW19990429.0168 -1 LINE> <TEXT> <P>  WASHINGTON (AP) -- Early in its history, the geology of Mars may have
     been much like that of Earth, with molten rock gurgling to the surface from a hot interior and with crustal
     plates drifting on the surface, according to new s
1894 NYT19990923.0363 -1 so see Mars-sider) By MARK CARREAU c.1999 Houston Chronicle <TEXT> <P> After a trouble-
    free, 91/2-month voyage from Earth, NASA’s Mars Climate Orbiter disappeared early Thursday as it was
    maneuvering into orbit around the Red Planet. </P> <
1894 NYT20000324.0337 -1 For more than two years, planetary geologist Ken Edgett has been among the first
    experts to examine the steady stream of photography dispatched to Earth by NASA’s Mars Global Surveyor, a
    small camera-equipped probe that has been circling the Red Plan
1894 NYT20000324.0337 -1 geologist Ken Edgett has been among the first experts to examine the steady stream of
    photography dispatched to Earth by NASA’s Mars Global Surveyor, a small camera-equipped probe that has been
    circling the Red Planet since September 1997. </P> <P>
1894 NYT19990313.0104 -1 far. </P> <P> The rain of debris from Mars was hardest, experts agree, in the Earth’s
    early days. And the reverse trip was far less likely because the Sun pulls Earth debris away from Mars and
    toward itself. </P> <P> Burns of Cornell and his
```

Figure 1: Example input question with ten passage choices.

1. Document Reader This preliminary stage manages the input to the system, reading in questions and passages of the form presented in Figure 1. The questions and passages are not assumed to be in any order, and no assumptions are made about the number of passages or questions present in the input document. The document is assumed to contain at most one question per question id, but the question ids need not be sequential (or even integral). The use of an initialization method in my implementation of this class allows for the input document to be sorted into separate question-passage entries. These entries are then used to populate the CAS when the document reader's getNext() method is called.

2. Test Element Annotator In this stage, the document is parsed according to the major sections visible in the example input in figure 1. The question(s) and corresponding passage(s) present in the input document are identified and instantiated as Questions or Passages using the defined UIMA Type System. Additionally, appropriate annotational meta data is identified and recorded (e.g. for a passage, this includes the source document id, the label identifying whether the passage correctly answers the question, the isolated text of the passage, and the indices into the text which identify the raw input which is being annotated.). Copies of the raw spans for the questions and passages are also created which are then maintained for use in the downstream annotators and CAS Consumers.

3. 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. A pointer to the raw input of each tokenization is also preserved, for the benefit of any downstream tasks.

4. N-gram Annotator In this stage, each of the tokenized spans from the previous annotator are annotated with ngrams of fixed (parameterizable) length. The parameter is read from the UimaContext when the annotator is initialized. Again, pointers to the raw input spans of each ngram set are preserved for the benefit of downstream tasks.

5. Score Annotator In this stage, each passage is assigned a similarity score based on its ngrams and the ngrams of its respective question. To ensure the scoring metric is easy to interpret, a score of 1 is given for identical texts and 0 for completely dissimilar texts. For this assignment, the Ngram annotations of the two spans are used to compute a percentage overlap between the two texts.

6. Output Annotator In this final analysis stage, the scored passages for each question are gathered and compiled into an output string along with their respective scores.

7. CAS Consumer The final stage of the pipeline manages the system output, writing the output annotations to a single output file.

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). 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.

This was my first full UIMA pipeline project that resulted in a workable output system. I still find 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 implement a more sophisticated ranking Strategy which would examine more than a single ngram annotator's outputs. The Composite design pattern would work well to combine such multiple ranking Strategy designs into a single ranking Strategy.

It is not easy to construct a system of this size from the ground up in a few short days when working alone, but now that I have become familiar with the eclipse framework for UIMA component design, I am looking forward to exploring more powerful design frameworks.

4 Passage Ranker UIMA CPE Implementation

The full pi4-kmaki project repository including this report be obtained at https://github.com/kortemaki/pi4-kmaki.

5 Conclusion

In this report, I presented my UIMA Passage Ranker in pursuit of the requirements of the PI4 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.