11-791: Design and Engineering of Intelligent Information Systems

Project Individual 6: Error Analysis

Keith Maki (andrewID: kmaki)

October 12, 2015

1 Introduction

In this report, I present my design and implementation for the error analysis model pursuant to the PI6 sample information processing system evaluation task of measuring the performance of a passage ranking information retrieval system. The report is structured as follows: section 2 describes the overall error 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 data; and section 5 concludes the report.

2 Error analysis for Passage Ranking System Design Process

I present an analysis framework for evaluating a passage ranker using several metrics. I apply this evaluation framework to the passage ranker from the previous Project Individual, which took an ngram-based approach to computing the ranking, but this could be exchanged for an arbitrary passage ranker which interfaces with the metrics analysis engine using the proposed framework. 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 the metrics evaluations, which is then incorporated into an aggregate analysis engine. This framework is described in detail below. The analysis engine from the previous PI was then chained with this framework to produce a single aggregate analysis engine which instantiated the analysis component engine flow followed by the evaluation component engine flow. The steps in the flow incrementally process 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. Aggregate Analysis Engine

   2.1. Analysis AAE

      i. 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.

      ii. 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.

      iii. 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.

      iv. 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.

2.2. Evaluation AAE

   i. Error Analysis Annotator This annotator is similar to the performance annotator from last week. See the documentation for more details.

   ii. Output Annotator In this final analysis stage, the evaluations for each question are gathered and compiled into an output string matching the format of the task.

3. 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.

One major limitation in the current evaluation engine framework is that the evaluation metrics are all implemented within a single primitive analysis engine. It would be fairly straightforward to implement these metrics as separate ComponentAnnotations, and populate a Performance type with each metric during a sequential aggregate analysis engine. I worked to improve the flexibility of my Performance type this week, but I ran out of time and did not implement separate analysis engine components.

I still find it very challenging to divide my efforts among several modules while working under a time crunch, and this week found me again unable to complete the PI within the allotted time. Although I can only budget so much time to spend on the assignments for this course, I have begun to build up a repertoire of analysis engines and type system components which are flexible enough to be applied in fairly general projects. Ultimately, however, development of these systems is still time consuming, and I spent a significant amount of time fighting with runtime errors resulting from improper configuration settings in my eclipse workspace. While I have a fully usable system this week, which can be installed and executed through Maven, I had to focus exclusively on implementing the evaluation framework for this assignment, meaning I continue to ignore more interesting extensions. For example, in addition to short-sightedly implementing a simplified design as outlined above, I was unable to implement a more sophisticated ranking Strategy which would examine more than a single ngram annotator’s outputs.

Question ID	TP	FN	FP	Precision	Recall	F1
1927	2	1	3	0.667	0.667	0.667
1995	1	0	4	1.000	1.000	1.000
2043	0	0	5	0.000	0.000	0.000
2067	0	0	5	0.000	0.000	0.000
2109	1	0	4	1.000	1.000	1.000
2153	1	2	4	0.333	0.333	0.333
2186	1	0	4	1.000	1.000	1.000
2275	0	0	5	0.000	0.000	0.000
2341	1	0	4	1.000	1.000	1.000
2352	3	3	2	0.500	0.500	0.500

Figure 2: The TP, FN, and FP counts for each question in the random subset.

4 Error Analysis UIMA CPE Implementation and Results

The full pi6-kmaki project repository including this report be obtained at https://github.com/kortemaki/pi6-kmaki.

The TPs, FNs, and FPs for each question of the random subset are displayed in Figure 2. The pie chart in Figure 3 shows the relative shares of TP, FN, and FP in the random subset, and the pie chart in Figre 4 shows the relative shares for the entire dataset. The table in Figure 5 displays information about the f1-statistic for the corpus and random subset. Although I could not complete a detailed error analysis, it is straightforward to understand the relatively poor performance of my system at this time. The current passage scoring metric relies entirely on ngram overlap, and specifically trigram overlap. This metric results in a large number of passages achieving the same similarity score with the question (0.0). Because of this, the rankings produced by the system are essentially random. In the coming project individuals, I will look forward to implementing more sophisticated ranking metrics and comparing the performance of the system across multiple configurations. For now, I estimate that the necessary effort to implement additional ranking components will be on the order of 20 hours. The estimated effort to implement a combined ngram approach is 5 hours. It is unclear what the impact of these changes will be, but it can be easily computed by the system if multiple configurations are run side-by-side. I will explore this in future weeks.

5 Conclusion

In this report, I presented my UIMA Metrics passage ranker error analysis framework in pursuit of the requirements of the PI5 assignment. The techniques employed in the design of this system will be integral to managing the evaluation and design of more elaborate information systems in later units of this course and in the Team Project assignments in the latter portion of the semester. Beyond this, the underlying design principles continue to serve as important building blocks for simple yet effective software engineering which

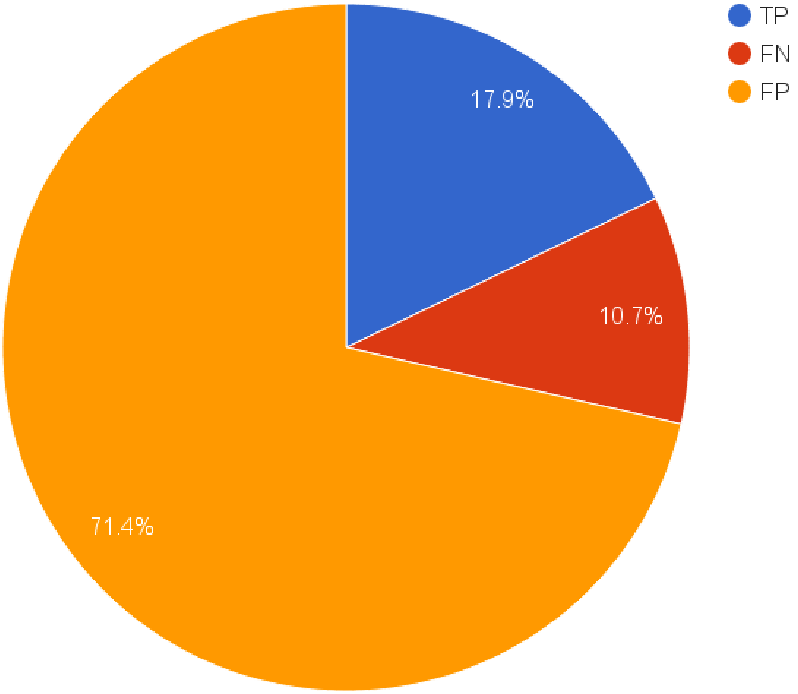

Figure 3: Graph showing TP, FN, and FP for the random subset.

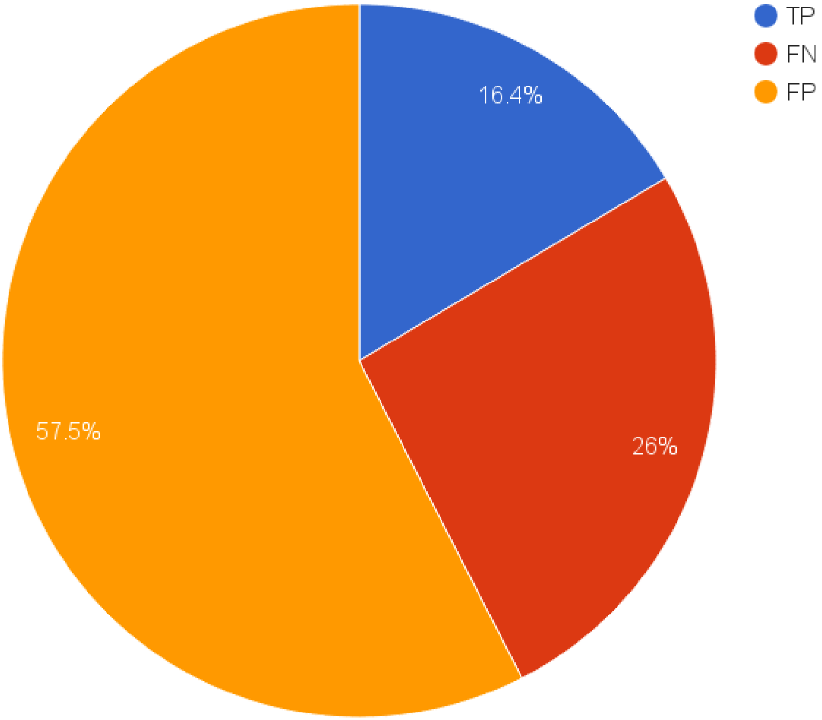

Figure 4: Graph showing TP, FN, and FP for the entire corpus.

	Micro	Macro
Corpus	0.283	0.381
Subset	0.303	0.55

Figure 5: The macro and micro averaged F1 statistic for the corpus.

will provide a conceptual foundation to build on long after this course has been completed.