11-791: Design and Engineering of Intelligent Information Systems

Project Individual 2: UIMA Type System

Keith Maki

September 14, 2015

1 Introduction

In this report, I present my design and implementation for the logical data model pursuant to the PI2 sample information processing task outlined in figure 1. The report is structured as follows: section 2 describes the overall type system hierarchy and data class design; section 3 provides discussion of important design considerations and limitations to the selected approach; section 4 presents the UIMA type system descriptor and other relevant project files; and section 5 concludes the report.

2 Type System Design Process

In designing my first type system, I found it useful to focus initially on top-down organization. I began by describing the types of annotations produced for each test element by each of the analysis engine steps shown in Figure 1. Next, I identified primary data elements which would need to be represented in the type system (e.g. Question, Answer, Tokenization, Score). I then built a preliminary type system using the Eclipse interface. This type system centered around a generic datatype for each test element, which would be populated with additional annotations as the analysis pipeline continued to process the associated test element.

This solution likely could have been instantiated in an analysis framework, but it suffered from two major flaws. First, it resulted in many design choices which limited the ability of the type hierarchy to generalize to additional domains. My top-down approach

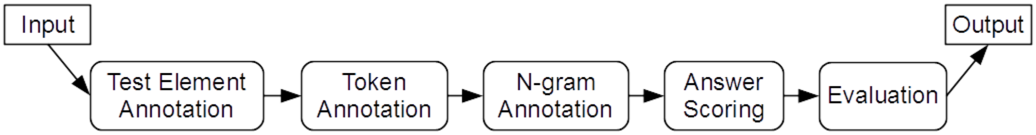

Figure 1: High-level architecture of the system scaffolded by this project.

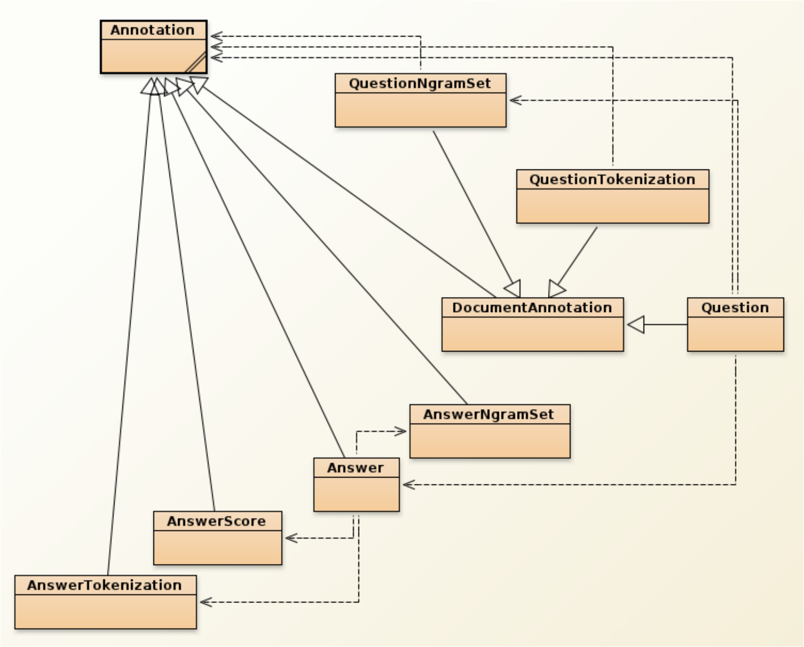

Figure 2: BlueJ Class Diagram for Preliminary Type System

inherently began with the sample task outline and refined the description from there, leaving it unable to easily capture broader notions. Second, it was difficult to take advantage of substructure present in the task, in that classes could not easily inherit from other classes, although many of my types began to look more similar as I continued to add detail to their descriptions.

An inheritance diagram of my preliminary type system is shown in Figure 2. Although I ultimately decided to scrap this version of my type system hierarchy, it was helpful to give myself an initial view of what was possible within the framework in order to inform a more intelligent design. It is important to note the weak notions of inheritance that plagued this design, with a heavy reliance on prebuilt types and an unfounded compulsion to separate Question and Answer-related data types. Notice also the unused QuestionTokenization class (which I believe was an error in this commit); it is easy to overlook missing dependencies like this in a poorly designed hierarchy.

Once I had gotten a feel for the kinds of elements which would be useful to complete the top-level hierarchy, I rebuilt my type system using a bottom-up perspective. This allowed me to ensure my design included all necessary subtypes before I extended or included them as fields in other types. During this step, I also completed the documentation strings for each field and type. In taking a broad view of the types of data that would be useful to annotate in general systems involving the analysis engine tasks outlined by Figure 1, I feel I was better able to conceptualize not only the aspects of each data element which were useful to extend and inherit within my type system design, but also those aspects which would enable them to be extensible and useful in more general analysis frameworks.

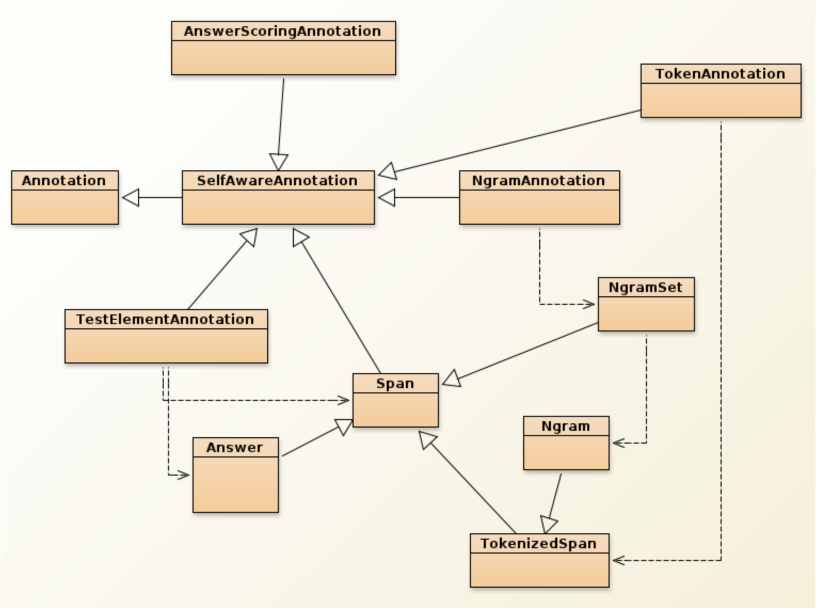

Figure 3: BlueJ Class Diagram for Final Type System

An inheritance diagram of my final type system is shown in Figure 3. This design has much deeper motivation behind its inheritance structure and avoids the field declaration redundancies that were so prevalent in the earlier design.

3 Design Considerations

In developing my final type system, I aimed to ensure that the machinery required to produce the related types was as generalizable as possible. Each of the analysis engines should not depend heavily on the structure of the system overall, and should only rely on behavior from upstream engines that can reasonably be expected to be part of such an upstream engine. For example, the scorer engine should not need to reproduce Answer information (e.g. the "correct" field from the gold data) even though it might be easier for downstream engines like the evaluator if these data were in the same place as the annotations produced by those engines. Tweaking an analysis engine's designs is only justified when it becomes clear that engine does not meet the specifications drafted for its *own* requirements.

I also tried to encourage flexibility in my type system design where possible with respect to configurability on the part of the analysis engines. For example, I provide for arbitrary numbers of ngram annotations, but I require each ngram annotation keep track of its order of n so that a scorer which should only consider certain ngrams (e.g. no ngrams of length > 5) can easily filter the annotations to suit its needs.

Of particular note is the affordance in my Span annotations of a String text field aside from the inherited begin and end Integer indices. This allows for more general annotation engines can be easily implemented using the same underlying type system. For instance, a many-to-one tokenizer could be implemented, which maps alternate spellings onto the same tokenization (e.g. "don't" → "do not", "do not" → "do not"). Similarly, tokenizers which perform disambiguation using an ontology (e.g. "Booth shot Lincoln" → "John_Wilkes_Booth shot Abraham_Lincoln") can be easily implemented by annotating the appropriate text field in each Span.

4 UIMA Type System Descriptor

Figure 4 provides the complete UIMA type system descriptor for the described logical data model. The concise documentation provided in the form of class and field descriptions works alongside the UIMA JCasGen automatic code generation scripts to produce functional and easy-to-work-with Java class files from this xml. The full pi2-kmaki project repository including this report be obtained at https://github.com/kortemaki/pi2-kmaki.

5 Conclusion

In this report, I presented my UIMA Type system in pursuit of the requirements of the PI2 assignment. The techniques employed in the design of this type system will be foundational in 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.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<typeSystemDescription xmlns="http://uima.apache.org/resourceSpecifier">
    <name>pi2-kmaki-typesystem</name>
    <description/>
    <version>1.0</version>
    <vendor/>
  <types>
    <typeDescription>
      <name>SelfAwareAnnotation</name>
      <description>Annotation subclass which is aware of the annotator that produced it.

Fields inherited from Annotation:  begin, end</description>
      <supertypeName>uima.tcas.Annotation</supertypeName>
      <features>
        <featureDescription>
          <name>annotator</name>
          <description>The name of the annotator that produced this annotation.</description>
          <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>Span</name>
      <description>Subclass of SelfAwareAnnotation.  Annotates a span of text with a String that may encode information about the span of text.

Fields inherited from SelfAwareAnnotation:  begin, end, annotator</description>
      <supertypeName>SelfAwareAnnotation</supertypeName>
      <features>
        <featureDescription>
          <name>text</name>
          <description>The text annotated by the span.

**Note that this text may differ from the portion of the annotated TestElement as indexed between begin and end!!</description>
          <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>TokenizedSpan</name>
      <description>Subclass of Span which annotates a tokenization for its span.

Fields inherited from Span:  begin, end, annotator, text</description>
      <supertypeName>Span</supertypeName>
      <features>
```

Figure 4: The UIMA Type System Descriptor XML for the presented type system

```
        <featureDescription>
          <name>tokens</name>
          <description>The tokens identified in the tokenization of this Span.</description>
          <rangeTypeName>uima.cas.FSList</rangeTypeName>
          <elementType>Span</elementType>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>Ngram</name>
      <description>Subclass of TokenizedSpan which annotates a subset of a tokenization corresponding to an ngram of finite length n.

Fields inherited from TokenizedSpan:  begin, end, annotator, text, tokens</description>
      <supertypeName>TokenizedSpan</supertypeName>
      <features>
        <featureDescription>
          <name>n</name>
          <description>The cardinality of the ngram identified by this TokenizedSpan.</description>
          <rangeTypeName>uima.cas.Integer</rangeTypeName>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>NgramSet</name>
      <description>Subclass of Span annotating selected ngrams for the given span.

Fields inherited from Span: begin, end, annotator, text</description>
      <supertypeName>Span</supertypeName>
      <features>
        <featureDescription>
          <name>ngrams</name>
          <description>List of ngrams identified for this Span.</description>
          <rangeTypeName>uima.cas.FSList</rangeTypeName>
          <elementType>Ngram</elementType>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>Answer</name>
      <description>Subclass of span annotating an answer choice for a particular TestElement.  Is aware of whether it is a correct answer choice or not.

Fields inherited from Span:  begin, end, annotator, text</description>
```

Figure 5: Continued from Figure 4

```xml
      <supertypeName>Span</supertypeName>
      <features>
        <featureDescription>
          <name>correct</name>
          <description>Indicates correctness of this Span as an answer choice under ”gold” labels.</description>
          <rangeTypeName>uima.cas.Boolean</rangeTypeName>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>TestElementAnnotation</name>
      <description>Subclass of SelfAwareAnnotation holding question and Answer Span annotations for a TestElement.

Fields inherited from SelfAwareAnnotation:  begin, end, annotator</description>
      <supertypeName>SelfAwareAnnotation</supertypeName>
      <features>
        <featureDescription>
          <name>question</name>
          <description>Identifies the question for this TestElement.</description>
          <rangeTypeName>Span</rangeTypeName>
        </featureDescription>
        <featureDescription>
          <name>answers</name>
          <description>Array holding annotations identifying each answer choice for this TestElement.</description>
          <rangeTypeName>uima.cas.FSArray</rangeTypeName>
          <elementType>Answer</elementType>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>TokenAnnotation</name>
      <description>Subclass of SelfAwareAnnotation holding tokenization annotations for a TestElement.

Fields inherited from SelfAwareAnnotation:  begin, end, annotator</description>
      <supertypeName>SelfAwareAnnotation</supertypeName>
      <features>
        <featureDescription>
          <name>questionTokens</name>
          <description>Tokenization of the question for this TestElement.</description>
          <rangeTypeName>TokenizedSpan</rangeTypeName>
        </featureDescription>
        <featureDescription>
          <name>answersTokens</name>
```

Figure 6: Continued from Figure 5

```
          <description>Array holding a tokenization for each answer choice for this TestElement.</description>
          <rangeTypeName>uima.cas.FSArray</rangeTypeName>
          <elementType>TokenizedSpan</elementType>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>NgramAnnotation</name>
      <description>Subclass of SelfAwareAnnotation holding ngram annotations for a TestElement.

Fields inherited from SelfAwareAnnotation:  begin, end, annotator</description>
      <supertypeName>SelfAwareAnnotation</supertypeName>
      <features>
        <featureDescription>
          <name>questionNgrams</name>
          <description>The ngrams identified in the question for this TestElement.</description>
          <rangeTypeName>NgramSet</rangeTypeName>
        </featureDescription>
        <featureDescription>
          <name>answersNgrams</name>
          <description>Array of ngrams identified in each answer choice for this TestElement.</description>
          <rangeTypeName>uima.cas.FSArray</rangeTypeName>
          <elementType>NgramSet</elementType>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
    <typeDescription>
      <name>AnswerScoringAnnotation</name>
      <description>Subclass of SelfAwareAnnotation holding answer scoring annotations for a TestElement.

Fields inherited from SelfAwareAnnotation:  begin, end, annotator</description>
      <supertypeName>SelfAwareAnnotation</supertypeName>
      <features>
        <featureDescription>
          <name>scores</name>
          <description>Array of scores assigned to each answer by the analysis engine.</description>
          <rangeTypeName>uima.cas.FloatArray</rangeTypeName>
          <multipleReferencesAllowed>true</multipleReferencesAllowed>
        </featureDescription>
      </features>
    </typeDescription>
  </types>
</typeSystemDescription>
```

Figure 7: Continued from Figure 6