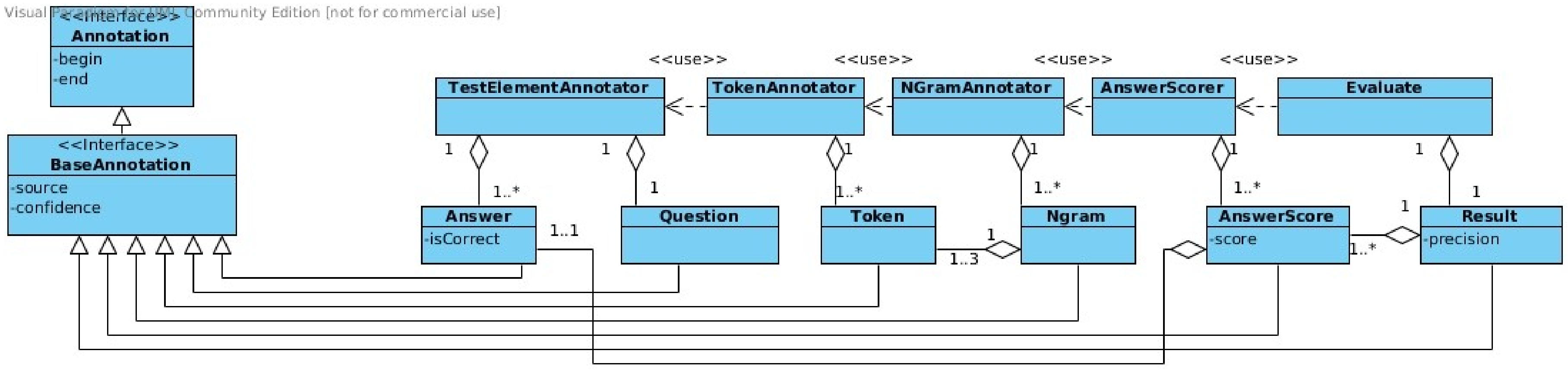
Jonathan Barker
11791
Hw #1 – Report

Type system diagram:



Above is a UML class diagram representing the type system and relations between components. This diagram doesn't contain methods or attributes for Annotators and Analysis Engines.

The types I have defined are: **Question**, **Answer**, **Token**, **Ngram**, **AnswerScore**, **Result** (and **BaseAnnotation**). These are all subclasses of the **BaseAnnotation** class, which is itself a subclass of **Annotation**. The **BaseAnnotation** class adds a string feature called "source" (to document which class created it) and a numeric feature called "confidence" (to record the confidence of the component which created the annotation) to the "begin" and "end" integer features which record the indices of the annotation's span of text.

The **Answer** type adds the boolean feature "isCorrect" to record whether a candidate answer answers the question or not.

The **NGram** type contains a list of the **Token** instances which comprise it. For this information processing task the number of tokens per **Ngram** is one, two, or three.

AnswerScore adds the numeric value "score" which represents how well the answer answers the question according to the scoring model.

Result is the end result of analysis containing the precision of the ranking evaluated at *n* (where *n* is the number of correct answers). This also contains the n-best list of **AnswerScore** instances.

Notes:
This type system contains all necessary input and output types for annotators and analysis engines. The initial input document is the “subject of analysis” and is stored in the CAS. As such it doesn't need it's own type in the system.
The final result is stored in the **Result** type. The point of representing annotations and scores in classes is so that the objects may be serialized and seamlessly integrated for other tasks. Displaying output to the screen can be done via a pretty printing function, instead of preassembling string outputs and wasting disk memory. To print these results one would need a simple function to read out the question and n-best list with scores from the **Question**, and **Result** annotations, thus a separate type to contain a string of output is not necessary.

One possible variation of the **Result** type would be to add a **Question** feature for storing the question. This way all necessary information can be assembled for output by just reading this one object's attributes.

The UML diagram shows each annotator using the previous one to gain access to the necessary type instances for proceeding. The annotations are linked to the CAS however and need not be aggregated by the classes themselves. An alternate diagram would have <<use>> and “creates” links leading from the annotators to the annotations.