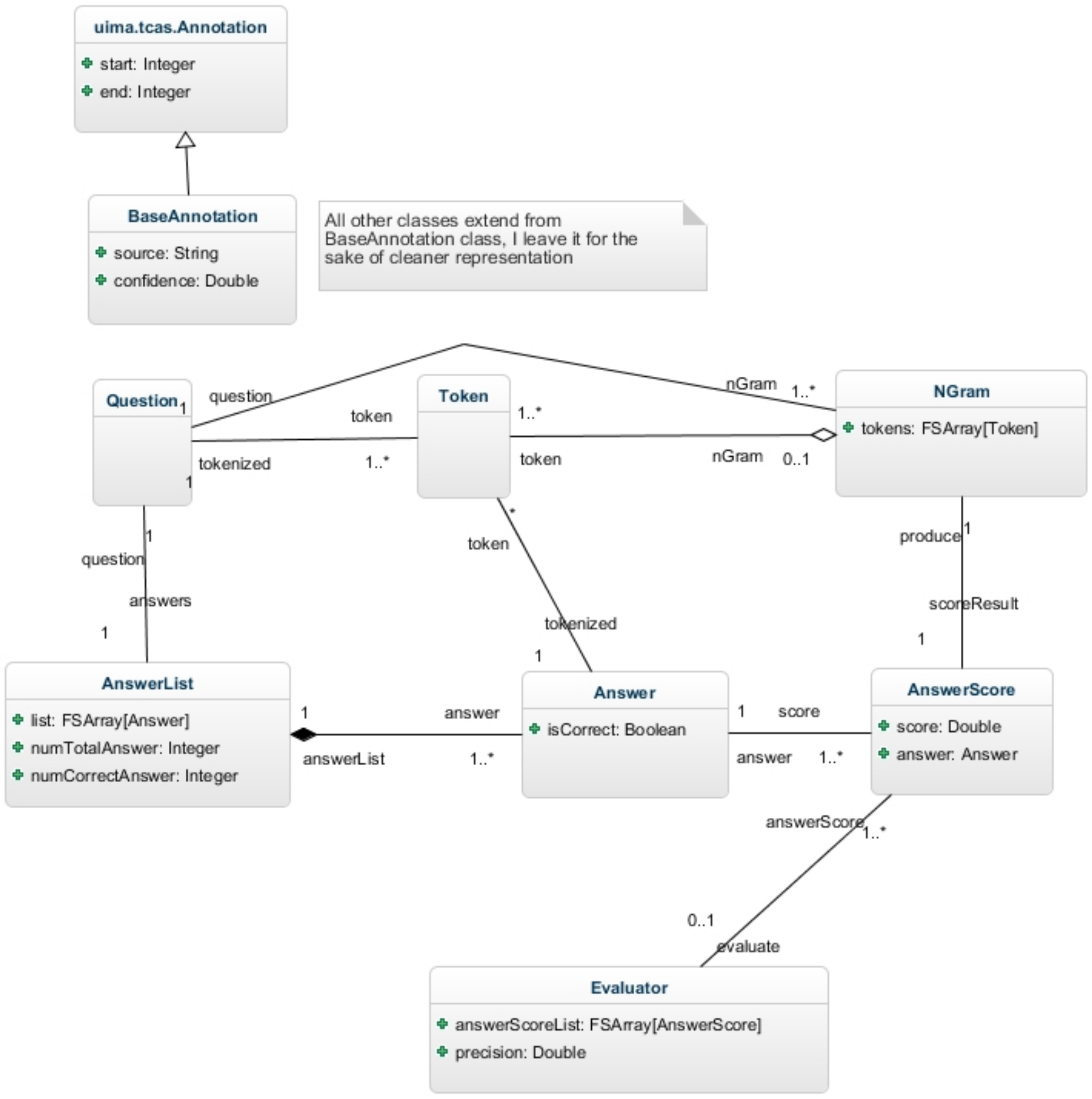
HW1 Report: Napat Luevisadpaibul
ID: nluevisa

Overview Design (using equivalent UML Class Diagram)



This System assume valid input (ie the format of both Question and Answers line are correct, valid file, etc)

Design Explanation by System Type

1. BaseAnnotation

This is my base class for all other class types. It has two features namely source(String) to record where this annotation was made by and confidence(Double) to indicate how confidence this annotation is. This class also extends from **uima.tcas.Annotation** so all other classes in my system inheriting from **BaseAnnotation** will have start, end, source and confidence feature.

2. Question

The question type will be generated from the input file in the first step of processing pipeline. It’s a question that we want to answer

3. Answer

This type is a possible answer to the question. The answer type will be generated from the input file in the first step of processing pipeline. Each line of answer will create one instance of answer so from one input file we can have one question instance and multiple answer instances. It have an additional feature called isCorrect (Boolean) to indicate whether this answer is correct or not. We need it to calculate precision.

4. AnswerList

This is a collection of Answer. It holds array of Answer so we can access the Answer from this class. It also had numTotalAnswer and numCorrectAnswer which are useful information in the evaluation pipeline process

5. Token

We need to tokenize our text from both Question and Answers into tokens of word so that we can do scoring by number of matching token.

6. NGram

This type holds an array of tokens. We form Unigram, Bigram, and Trigram (Unigram, Bigram and Trigram are of NGram type with different size of tokens) from tokens in Question. For example, given Question “Mary loves John”, Unigram tokens could be "love". Bigram tokens could be "John,loves" or "loves,Mary". Trigram tokens could be "John, loves, Mary"

7. AnswerScore

The AnswerScore contains score (Double) that evaluated from an answer using analysis engine and answer (Answer) that contain a reference to Answer that produce this score. Each answer may be referenced by many Instances of AnswerScore associated with method to calculate score(ie Unigram, Bigram, Token overlap) we can use feature “source” to determine the method we use for this scoring

8. Evaluator

This type contains array of AnswerScore named answerScoreList. We can use information from answerScoreList to calculate precision. Actually I have 2 ways to do this task

1. We can get number of correct answer and number of total answer from **AnswerList.** Then get N top score answer from answerScoreList where N = number of correct answer and calculate the precision.
2. We can also get number of correct answer from answerScoreList itself because each AnswerScore contains Answer which has feature "isCorrect". We can examine all AnswerScore in answerScoreList to get number of correct answer and number of total answer. The rest is the same as the first method.

We store the precision(Double) and answerScoreList so we can generate the report just by looking at this type

Design Explanation by processing pipeline

1. Test Element Annotation: The system will read in the input file as a UIMA CAS and annotate the question and answer spans. This will produce **Answer, AnswerList and Question**.
2. Token Annotation: The system will annotate each token span in each question and answer (break on whitespace and punctuation). This will create **Token** from **Question** and **Answer**

3. NGram Annotation: The system will annotate 1-, 2- and 3-grams of consecutive tokens. From **Token**, generated from **Question**, we produce **NGram**

4. Answer Scoring: The system will incorporate a component that will assign an answer score annotation to each answer. The answer score annotation will record the score assigned to the answer. This step create **AnswerScore**

5. Evaluation: The system will sort the answers according to their scores, and calculate precision at N. We use and record the result in **Evaluator**