HW #1 Report: Q&A System

Version: 0.1

Fall 2013 11-791

Jeffrey Gee

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the UIMA type system and data flow for a simple QA system.

1.2 Requirements

The following requirements were specified by the HW#1 assignment document.

1.2.1 System Input

The input types for this system will be text documents containing a single question statement and a collection of candidate answer statements. All answers will be labeled as to whether they are correct ("1") or not ("0"). An example is shown below.

```
Q John loves Mary?
A 1 John loves Mary with all his heart.
A 1 Mary is dearly loved by John.
A 0 Mary doesn't love John.
A 0 John doesn't love Mary.
A 1 John loves Mary.
```

1.2.2 Tasks

The following tasks will be accomplished by this system.

1. Assign scores to each answer in regard to the question.
2. Rank the answers according to score.
3. Select the top *N* sentences.
4. Measure performance by how many of the selected sentences are correct.

1.2.3 System Assumptions

The design described in this document is based on some additional assumptions below.

1. Each input document will contain ONLY ONE question.
2. Each answer is relevant to a SINGLE question. For example, answers will not be scored against future questions.
3. Each answer can be scored any number of times. For example, multiple scoring mechanisms can be used to score the same answer against the same question.
4. Tokens and Unigrams are different in that tokens can be transformed (such as stemmed) by a Unigram annotator without modifying the token itself (and therefore not affecting other N-Gram annotators).

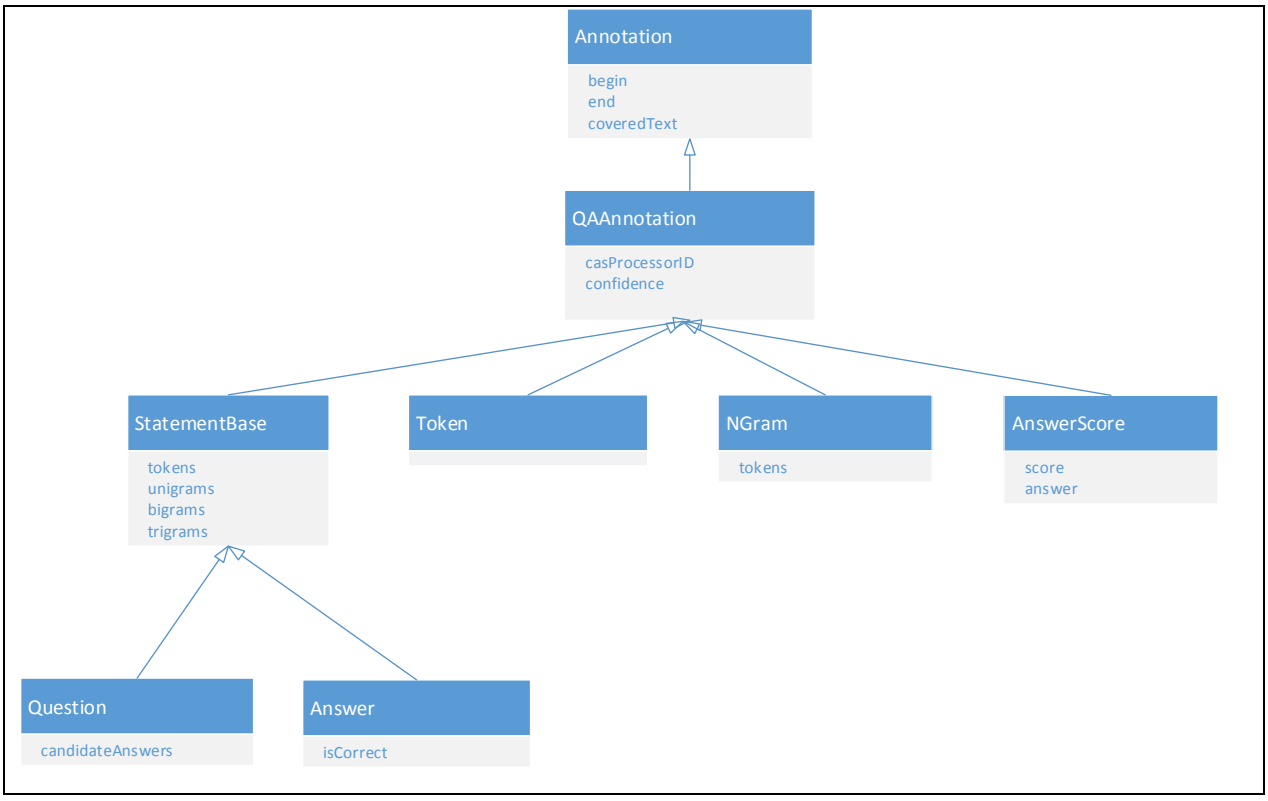
2 Type System

2.1 Annotation Types

The following table gives a high-level description of the annotations:

Annotation	Comment
QAAnnotation	QAAnnotation is the base annotation for this system. It extends UIMA's built-in Annotation class with additional fields, namely *casProcessorID* and *confidence*.
StatementBase	StatementBase is the annotation used to describe a line of text. Statements contain a finite number of tokens.
Question	Question is the annotation used to describe a StatementBase that is a question.
Answer	Answer is the annotation used to describe a StatementBase that is an answer to a question. In our training data, answers have been manually labeled for correctness.
Token	A Token is a single piece of text that makes up a statement.
NGram	N-grams are N-consecutive token combinations contained within statements.
AnswerScore	An AnswerScore is an annotation given by some scoring mechanism.

The diagram below is a high-level UML for the Annotation Types:



2.1.1 QAAnnotation

The QAAnnotation is the base annotation type for the QA system. It extends UIMA's Annotation type (which contains *begin*, *end*, *covered_text* properties, among others) and adds two essential properties: **CAS ProcessorID** and **confidence**. The CAS processor ID stores the class name for the annotator for a given annotation. This becomes exceptionally important when evaluating answer scoring mechanisms, as remembering which scoring annotator resulted in which answer scores is pivotal. Confidence is a measure between 0 and 1 that evaluates how confident an annotator is in its annotation.

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.QAAnnotation	org.apache.uima.jcas.tcas.Annotation

Feature	Type	Comment
casProcessorID	*String*	The class name of the annotator that created this annotation.
confidence	*double*	Confidence of the annotator, between 0 and 1 (1 being the highest) that this annotation is correct.

2.1.2 StatementBase

StatementBase is the base annotation for all annotations that correspond to a single line/sentence of text. StatementBase extends the default QAAnnotation type with additional features, such as **lists for tokens and n-grams.**

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.StatementBase	edu.cmu.lti.deiis.qa.types.QAAnnotation

Feature	Type	Comment
tokens	*List<Token>*	A list of tokens for this statement
unigrams	*List<NGram>*	A list of unigrams for this statement
bigrams	*List<NGram>*	A list of bigrams for this statement
trigrams	*List<NGram>*	A list of trigrams for this statement

2.1.3 Question

The Question annotation extends the StatementBase annotation with an additional feature, **candidate answers**. In the case of our basic QA system, outlined by the requirements of HW#1, the actual use of this feature is **not required**. This is because we make the assumption that each input document into our aggregate analysis system contains ONLY one question, and additionally that all answers in that document correspond to the same question. This means that all Answer annotations for an input document can be assumed to correspond to the single Question annotation in the document. However, in a case that a single input document contains multiple sets of Question and Answer annotations, it may be convenient to store pointers to corresponding answers for a given question.

Class Name	SuperType

edu.cmu.lti.deiis.qa.types.Question	edu.cmu.lti.deiis.qa.types.StatementBase

Feature	Type	Comment
candidateAnswers (not required)	*List<Answer>*	A list of candidate Answers for this statement

2.1.4 Answer

The Answer notation extends the StatementBase annotation with an additional feature, **correctness**. For training data, it is important to store whether an answers is really correct or not. This can be used to train the various scoring mechanisms.

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.Answer	edu.cmu.lti.deiis.qa.types.StatementBase

Feature	Type	Comment
isCorrect	*boolean*	For trained data sets, determines whether the answer has been labeled correct (true), or incorrect (false).

2.1.5 Token

The Token annotation extends the QAAnnotation. Tokens are what comprise NGrams as well as Question and Answer Statements. Tokens need no additional features, as their primary feature, the string contents of the Token itself, are already accessible by calling the inherited *getCoveredText()* method from Annotation.

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.Token	edu.cmu.lti.deiis.qa.types.QAAnnotation

2.1.6 NGram

NGram annotations extend QAAnnotation's and are used to represent 1-, 2-, and 3-token groups contained within sentences.

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.NGram	edu.cmu.lti.deiis.qa.types.QAAnnotation

Feature	Type	Comment
tokens	*Token[]*	The tokens that comprise this n-gram.

2.1.7 AnswerScore

AnswerScore notations extend QAAnnotation and are used to store the scores of answers against questions. Similar as to the case of Question, since our basic QASystem makes the assumption that each

Answer is being scored against exactly one question, it is unnecessary to have a pointer from an AnswerScore to a specific question (as there is only one Question annotation stored for each document).

Each **score** is on a scale of 0 to 1, with scoring mechanisms assigning values closer to 1 to Answers they believe more likely to be true. An **answer** pointer is used to store which Answer an AnswerScore corresponds to.

Class Name	SuperType
edu.cmu.lti.deiis.qa.types.AnswerScore	edu.cmu.lti.deiis.qa.types.QAAnnotation

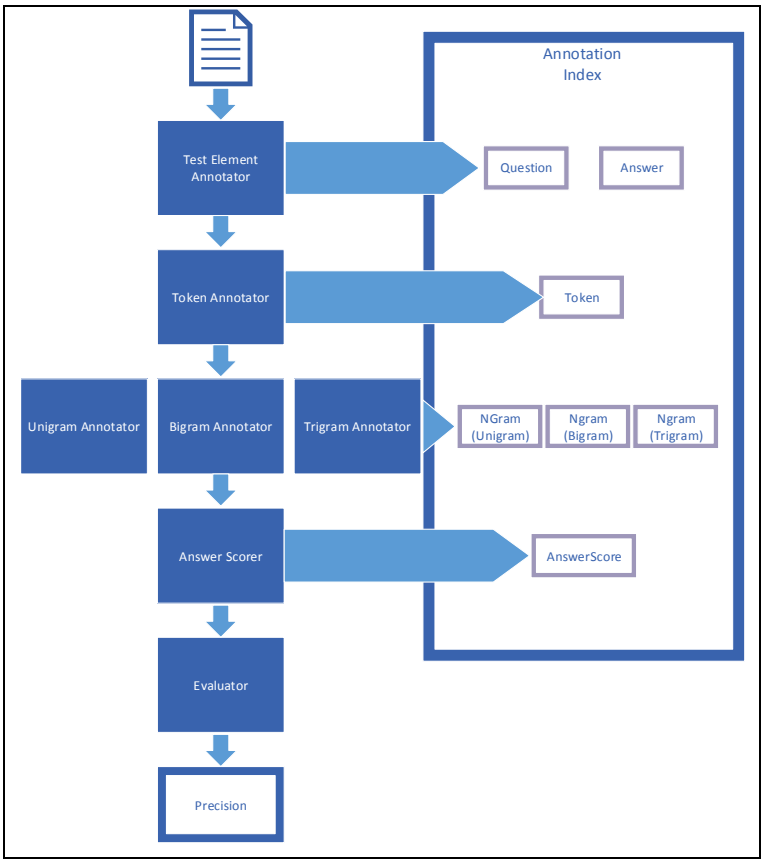
Feature	Type	Comment
score	*double*	The score value (between 0 and 1) for a question and answer pair.
answer	*Answer*	The answer to which this answer score is assigned to.
question (not required)	*Question*	The question to which this answer score is assigned to.

3 Data Pipeline

3.1 Simple QA System Pipeline

As per the assignment description, the simple QA System while consist of the aggregated processing pipeline below:

Annotator	Comment
Test Element Annotator	Read in the simple input file and annotate question and answer spans.
Token Annotator	Annotate each token span in each question and answer.
Unigram Annotator	Annotate each unigram in each question and answer.
Bigram Annotator	Annotate each bigram in each question and answer.
Trigram Annotator	Annotate each trigram in each question and answer.
Answer Scorer	Score each candidate answer against the question
Evaluator	Evaluate the precision for a set of AnswerScores produced by a given Answer Scorer.



3.1.1 Test Element Annotator

Input	Output
As input, the Test Element takes the CAS representation of the input document.	The Test Element Annotator produces two sets of annotations. Question annotations are made on sections of the text that are question statements. Answer annotations are made on sections of the text that are answer statements.

3.1.2 Token Annotator

Input	Output
As input, the Token Annotator takes all Question and Answer annotations that belong to the input CAS object.	The Token Annotator produces Token annotations for each of the tokens in each Question and Answer text span.

3.1.3 NGram Annotators

Input	Output
As input, NGram Annotators take in Token annotations belonging to each Question and Answer annotation.	The NGram Annotators output 1-, 2-, and 3-gram, NGram annotations belonging to each Question and Answer.

3.1.4 Answer Scorer

Input	Output
Based on the type of scoring mechanism used, the Answer Scorer will take any of the previously generated annotations.	The Answer Scorer will output Answer Score annotations for each of the Answer notations against the Question notation.

3.1.5 Evaluator

Input	Output
The Evaluator will read in all of the Answer Score annotations belonging to the Question in the document.	The Evaluator will rank the Answer Score annotations based on score values, and then predict a precision value based on the top *N* values.

4 Summary

This document outlined the UIMA type-system and processing pipeline for a simple QA system.

Section 1 introduced the QA system and the requirements for designing the system. We described the input format for the system as well as assumptions that were made when designing the type system.

Section 2 proposed UIMA types and annotations for solving the problems presented by the QA system. The UIMA type-system utilizes inheritance to efficiently address the requirements made by the system.

Section 3 presented a high-level overview of the data pipeline for the QA system. A diagram is used to describe which annotations are being contributed by which annotators in the pipeline.