Requirements

1. Annotate question from input text: User provides an input file with question, proposed answers and a boolean flag indicating whether each answer is correct. System reads input file and annotates questions.

2. Annotate answers from input text: User provides an input file with question, proposed answers and a boolean flag indicating whether each answer is correct. System reads input file and annotates all proposed answers. System also records whether each proposed answer is correct.

3. Annotate tokens in each question and answer: The system takes in a sentence (question or an answer) as input and annotates all tokens in the sentence.

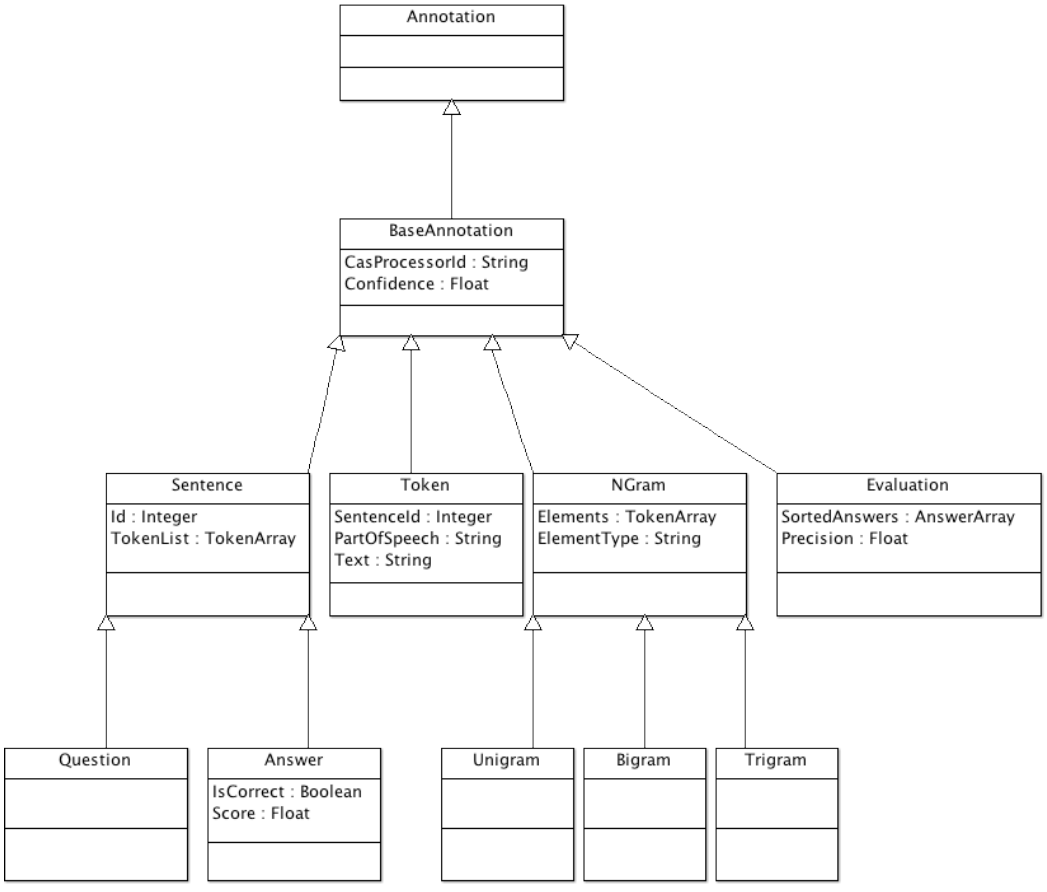
4. Annotate N-grams in each question and answer: The system takes in a sentence (question or an answer) as input and annotates all 1-grams, 2-grams and 3-grams in the sentence.

5. Assign a score to each answer: The system takes in an answer as input and assigns a score to the answer.

6. Sort answers according to scores and calculate precision: The system will take in all answers and the score assigned to each answer as input and sort the answers according to their scores. Uses the total number of correct answers and the number of predicted correct answers to compute precision.

Type-system

The UML Class diagram shown below provides a visualization of the various Java type classes that are produced when the proposed type system is compiled into Java type classes.



Class diagram constructed after compiling proposed type system into Java type classes.

- **BaseAnnotation**: The BaseAnnotation type is the supertype of all the types in the proposed type system. The BaseAnnotation type inherits from the uima.tcas.Annotation type.

  Its purpose is to ensure that all subtypes have a CasProcessorId (to indicate which class made the annotation) and a confidence value (to indicate how confident the source of the annotation was).

- **Sentence**: The Sentence type is the supertype of the Question and Answer types. It captures features common to both the Question and Answer types.

  The 'Id' feature is a unique identifier assigned to each sentence in a given input file. The purpose of adding this feature was to allow each 'Token' to easily determine which sentence the Token is part of.

  The TokenList feature of the sentence type is an array of all the Tokens in the sentence. The actual sentence text is captured as a feature of each Token in the TokenList.

- **Question & Answer:** The Question type inherits from the Sentence type. For the purposes of this project, the Question type does not have any more features other than the ones inherited from Sentence. However, more fields can be added as the need arises.

  The Answer type also inherits from the Sentence type. However, unlike Question, the Answer has two other features in addition to the features inherited from Sentence. The 'IsCorrect' feature indicates whether the particular answer correctly answers the question. The 'score' feature records the score assigned to the answer to allow answers to be sorted according to score during evaluation.

- **Token:** The Token type captures information about tokens in a Sentence (delimited by space and punctuation). It has three features namely SentenceId, PartOfSpeech and Text. As mentioned previously, the 'SentenceId' is a unique identifier assigned to each sentence (and kept track of by each Token) to allow a Token to determine which sentence it is part of. The 'PartOfSpeech' feature represents the linguistic category of token whereas the 'Text' feature captures the text contained in a token.

- **NGram:** The NGram is the supertype of Unigram, Bigram and Trigram and captures information about continuous sequences of tokens in a Sentence (Question or Answer). It consists of two features namely a list of Tokens named 'elements' of length N contained in the NGram and the type of element being stored in the elements array.

- **Unigram, Bigram & Trigram**: Unigram, Bigram and Trigram inherit from the NGram type. For the purposes of this project, these types don't have any additional fields to the ones they inherit from the NGram type. However, more fields can be added as the need arises.

- **Evaluation:** The Evaluation type captures information necessary to measure the performance of the entire system. This type consists of an array of Answers sorted by score to allow users to see which answers from input file best answer the question. This type also consists of a precision value to allow users to see the ratio of the actual number of correct answers to the predicted number of correct answers.

Package layout

The types in this system can be organized into the following three categories:

- **Base**: BaseAnnotation
- **TestElement**: Sentence, Question, Answer
- **Processed**: Token, NGram, Evaluation, Unigram, Bigram, Trigram

Using this categorization, the types have been organized into the following three packages:

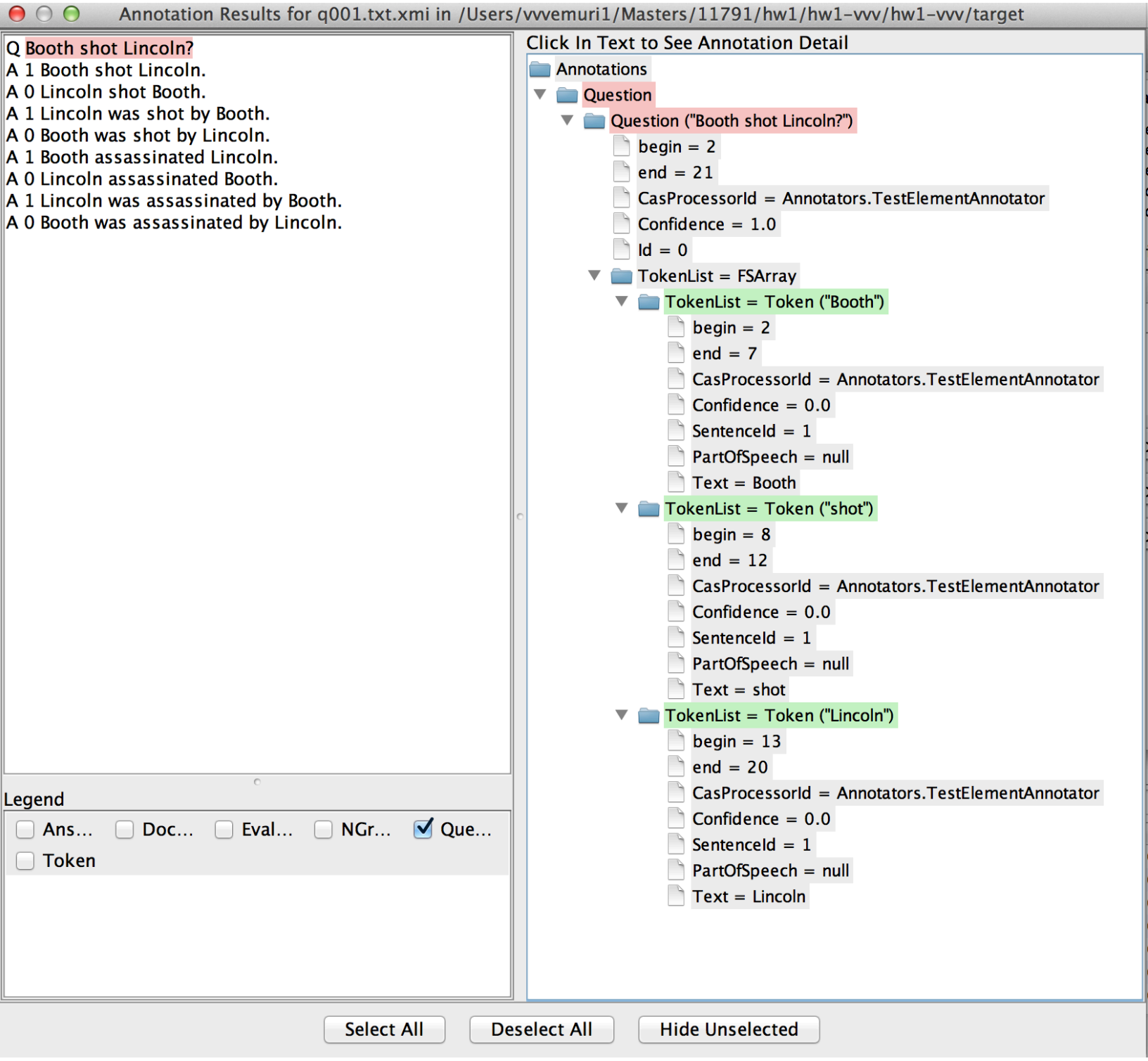
- **edu.cmu.lti.types.base**: BaseAnnotation
- **edu.cmu.lti.types.testElement**: Sentence, Question, Answer
- **edu.cmu.lti.types.processed**: Token, NGram, Evaluation, Unigram, Bigram, Trigram

Testing Type System

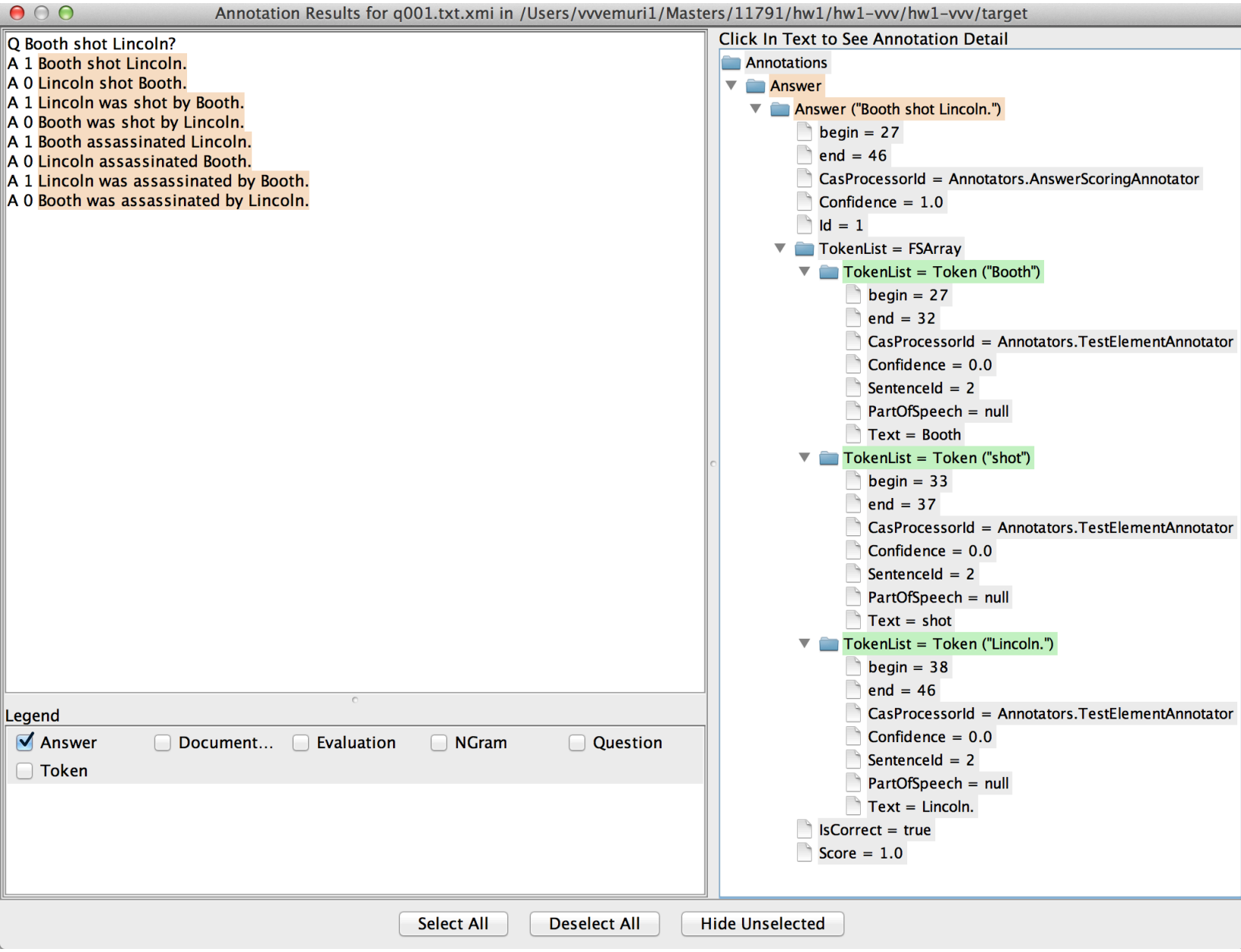
In order to test whether the type system sufficiently captures all of the information needed, I implemented some parts of the five annotators. Here are the results of the implementation.To perform the scoring, a Gold Standard scoring system was used.

Note: PartOfSpeech has not been implemented and hence shows up as null in the following images.

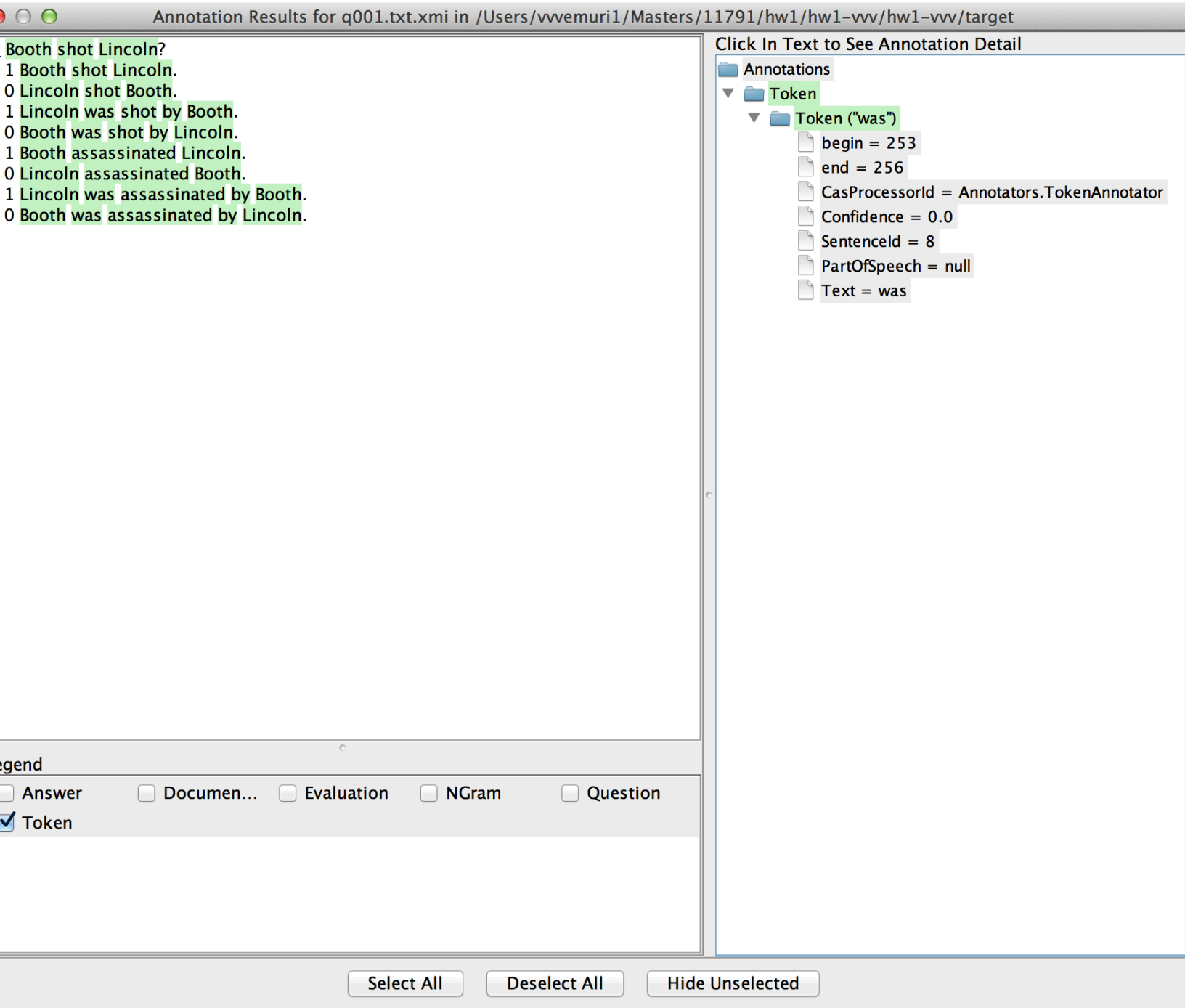
1.Question



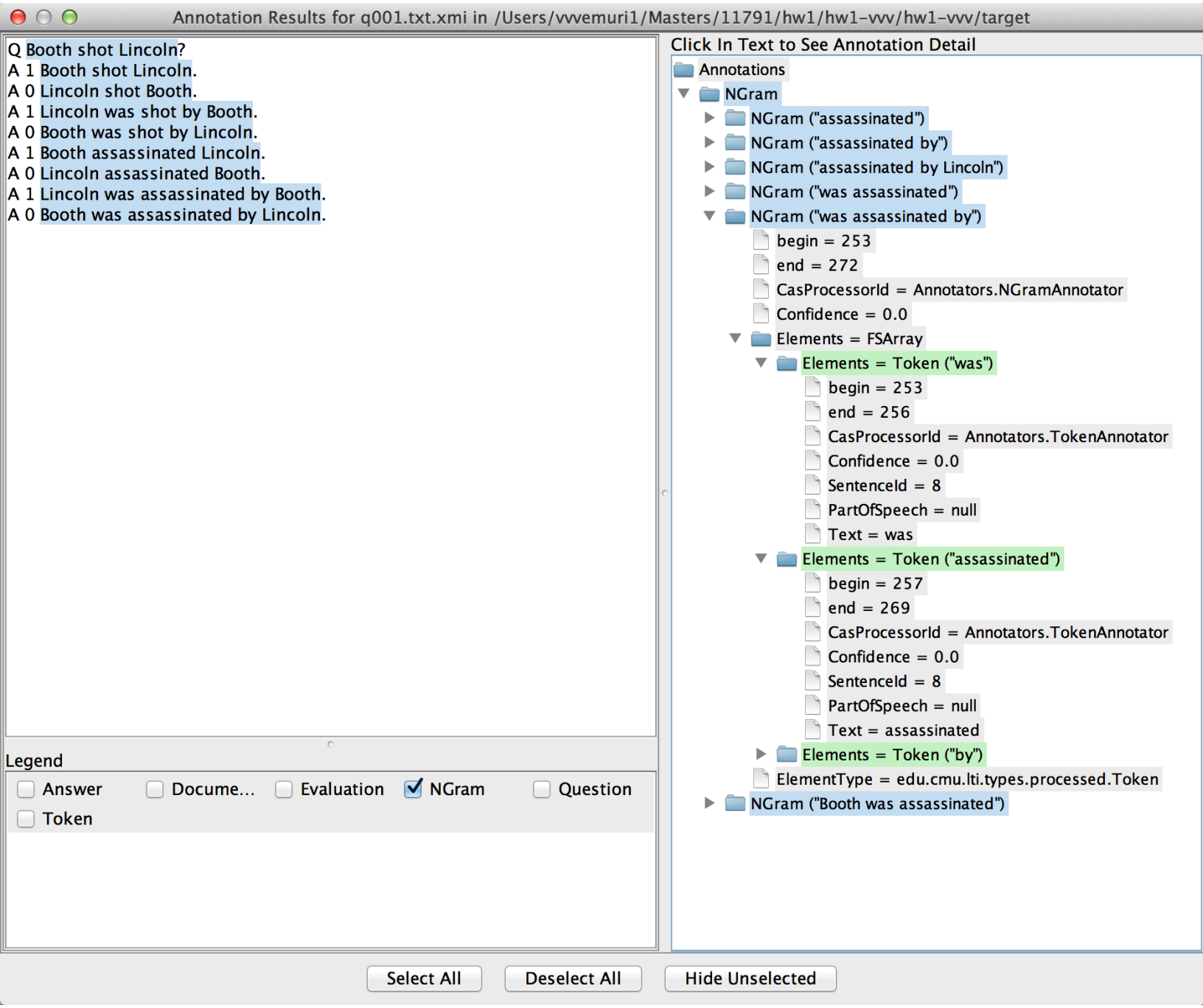
2. Answer


Annotation Results for q001.txt.xmi in /Users/vvvemuri1/Masters/11791/hw1/hw1-vvv/hw1-vvv/target
Q Booth shot Lincoln?
A 1 Booth shot Lincoln.
A 0 Lincoln shot Booth.
A 1 Lincoln was shot by Booth.
A 0 Booth was shot by Lincoln.
A 1 Booth assassinated Lincoln.
A 0 Lincoln assassinated Booth.
A 1 Lincoln was assassinated by Booth.
A 0 Booth was assassinated by Lincoln.
Legend
Answer
Document...
Evaluation
NGram
Question
Token
Click In Text to See Annotation Detail
Annotations
Answer
Answer ("Booth shot Lincoln.")
begin = 27
end = 46
CasProcessorId = Annotators.AnswerScoringAnnotator
Confidence = 1.0
Id = 1
TokenList = FSArray
TokenList = Token ("Booth")
begin = 27
end = 32
CasProcessorId = Annotators.TestElementAnnotator
Confidence = 0.0
SentenceId = 2
PartOfSpeech = null
Text = Booth
TokenList = Token ("shot")
begin = 33
end = 37
CasProcessorId = Annotators.TestElementAnnotator
Confidence = 0.0
SentenceId = 2
PartOfSpeech = null
Text = shot
TokenList = Token ("Lincoln.")
begin = 38
end = 46
CasProcessorId = Annotators.TestElementAnnotator
Confidence = 0.0
SentenceId = 2
PartOfSpeech = null
Text = Lincoln.
IsCorrect = true
Score = 1.0
Select All
Deselect All
Hide Unselected

3. Tokenization



4. NGram


Annotation Results for q001.txt.xmi in /Users/vvvemuri1/Masters/11791/hw1/hw1-vvv/hw1-vvv/target
Q Booth shot Lincoln?
A 1 Booth shot Lincoln.
A 0 Lincoln shot Booth.
A 1 Lincoln was shot by Booth.
A 0 Booth was shot by Lincoln.
A 1 Booth assassinated Lincoln.
A 0 Lincoln assassinated Booth.
A 1 Lincoln was assassinated by Booth.
A 0 Booth was assassinated by Lincoln.
Legend
Answer
Docume...
Evaluation
NGram
Question
Token
Click In Text to See Annotation Detail
Annotations
NGram
NGram ("assassinated")
NGram ("assassinated by")
NGram ("assassinated by Lincoln")
NGram ("was assassinated")
NGram ("was assassinated by")
begin = 253
end = 272
CasProcessorId = Annotators.NGramAnnotator
Confidence = 0.0
Elements = FSArray
Elements = Token ("was")
begin = 253
end = 256
CasProcessorId = Annotators.TokenAnnotator
Confidence = 0.0
SentenceId = 8
PartOfSpeech = null
Text = was
Elements = Token ("assassinated")
begin = 257
end = 269
CasProcessorId = Annotators.TokenAnnotator
Confidence = 0.0
SentenceId = 8
PartOfSpeech = null
Text = assassinated
Elements = Token ("by")
ElementType = edu.cmu.lti.types.processed.Token
NGram ("Booth was assassinated")
Select All
Deselect All
Hide Unselected

5. Evaluation