Logical Data Model for a simple Information Processor

1 Problem Statement

The task in hand is to design a logical data model for a simple Information Processor (IP). A text file is fed as input to the IP. The text file consists of Question and several lines that can be answers to the question specified. The IP analyses the sentence and assigns a score to each answer. The IP has to

1. Assign a score to each sentence [0..1];

2. Rank the sentences according to score;

3. Select the top N sentences where N = the number of correct answers;

4. Measure performance by Precision@N (how many of the top N are correct).

The logic to do the above steps are split into several sub tasks like reading the input and assigning the question and answer spans, split the sentences into tokens delimited by whitespaces or punctuation; then annotate them using NGramAnnotator; assign scores to the answers based on the correctness; evaluate the correctness of the answer after sorting the answer based on scores and using the Precision @ N formula.

2 Designing the Solution

2.1 UML Use Case Model

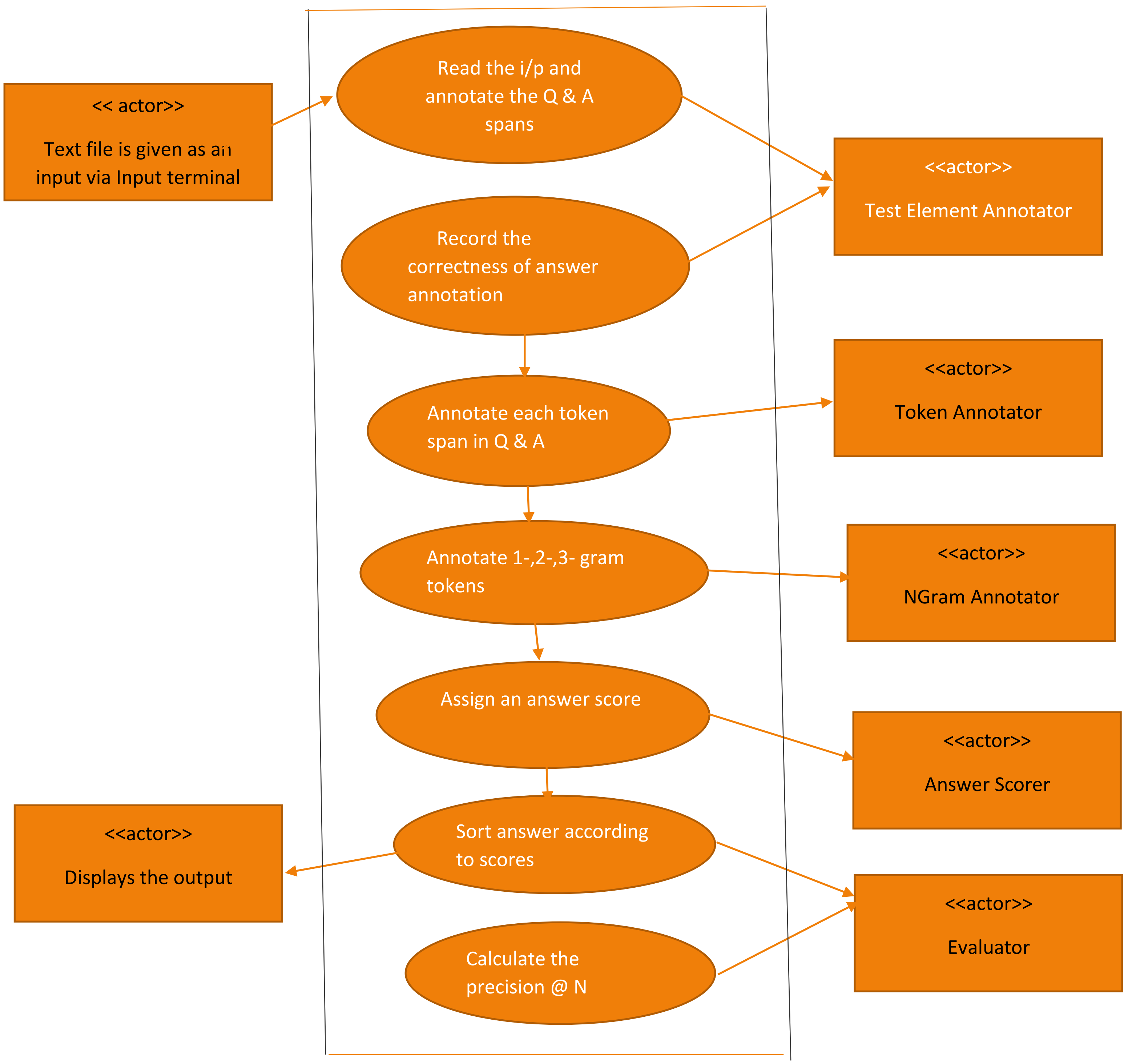
The use cases for IP are

1. Read the i/p and annotate the Q & A spans
2. Record the correctness of answer annotation
3. Annotate each token span in Q & A
4. Annotate 1-,2-,3- gram tokens
5. Assign an answer score
6. Sort answer according to scores
7. Calculate the precision @ N

Actors and Secondary Actors are:

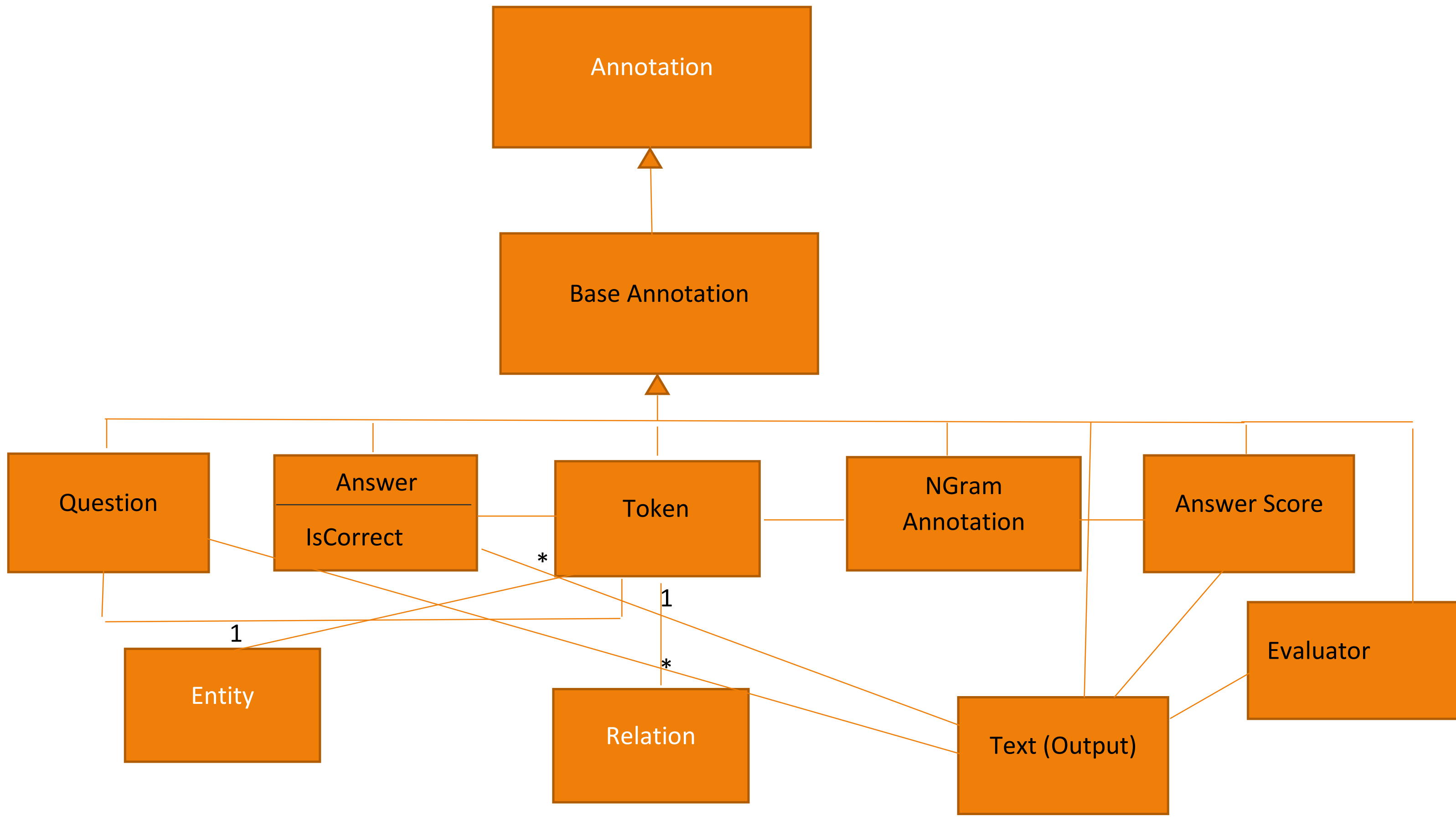
1. Unstructured data given as input in the form of document
2. Output displayed in the UI terminal
3. Test Element annotator
4. Token annotator

5. NGram Annotator
6. Answer Scorer
7. Evaluator


<< actor>>
Text file is given as an input via Input terminal
Read the i/p and annotate the Q & A spans
<<actor>>
Test Element Annotator
Record the correctness of answer annotation
Annotate each token span in Q & A
<<actor>>
Token Annotator
Annotate 1-,2-,3- gram tokens
<<actor>>
NGram Annotator
Assign an answer score
<<actor>>
Answer Scorer
<<actor>>
Displays the output
Sort answer according to scores
<<actor>>
Evaluator
Calculate the precision @ N

2.2 UML Domain Model

Based on the use case designed above, it becomes clear about the classes we would be needing to design IP.



2.3 DESIGN Explanation and implementation

My understanding is to get the input, break it down to simple words (tokens) and using these tokens , our NGram annotator has to come up with an analysis result which will be scored and the high score wins and gets displayed.

The Question and the Answer boxes are my input types. I have defined them as Annotation.Input.Question and Annotation.Input.Answer. These classes help in reading the text and deciphering the Q and A and comes up with the list of correct answers

The Token Annotator initiates the processing. It splits the sentences into small tokens delimited by spaces. E.g.: I'm a new-bee here. The token annotator splits this into {I, m, a, new, bee, here} based on the spaces or based on the punctions.

Once the tokens are obtained the analysis part begins by first identifying the tokens as Entities (Person/Organization/Location name, etc) and the verbs/predicates as the Relation (e.g. loves, shot at). Multiple tokens can be mapped to the same person. So, the multiplicity is Many to one mapping. Same can be applied for Relation as well. The word shoot can refer to a movie shooting as well as an action made using a weapon. So, It is one to many mapping. The token annotation type system is named as Annotation.Pipeline.Token.

The output of above step will be a set of tokens which can be analysed by permuting the tokens in the set of sizes 1/2/3 and the annotator gives the analysis based on that. The type system corresponding to this is named as Annotation.Pipeline.NGramAnnotator

The output of this will give meaningful insight about the sentences and we need to score the answer based on the relevancy to the question and/or how exact the answer is to the question. The type system is named as Annotation.Pipeline.AnswerScorer

So, now we have the necessary information about the input. The IP has to just scan through it and find the best answer based on the maximum score. It also finds the Precision and Recall of the results obtained. This is done by the Evaluator and is named as Annotation.Pipeline.Evaluator.

The output from this systems should be a sorted Input based on the Q followed by A and among A, the answers should be listed based on the score and correctness. The name of the type system for this is created as Annotation.Output.Text. This consolidates the Input, Answer Score and the Evaluator to come up with the Output type.

Apart from this, I have created a Base Annotation type that will have the features namely Source and Confidence. This is inherited by all the other types.

I have created a class for Entity and Relation. The tokens are classified based on this and this will be helpful in analyzing the context of the given problem.