Homework 1

Salvador Medina
ITAM
Design of Intelligent Information Systems

A type system was created in order to be able to solve the problem proposed. The problem consists of determining from a set which sentences answer a question, by a score to each sentence and ranking the sentences according to the score. Finally, the performance of the system would be measured according to the top N sentences coming from a ranked list.

The type system was created based on the requirements and the given processing pipeline specification, which is as follows:

> The information processing pipeline will consist of the following steps:
>
> 1. **Test Element Annotation**: The system will read in the input file as a UIMA CAS and annotate the question and answer spans. Each answer annotation will also record whether or not the answer is correct.
> 2. **Token Annotation**: The system will annotate each token span in each question and answer (break on whitespace and punctuation).
> 3. **NGram Annotation**: The system will annotate 1-, 2- and 3-grams of consecutive tokens.
> 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.
> 5. **Evaluation**: The system will sort the answers according to their scores, and calculate precision at N (where N is the total number of correct answers).

Within UIMA an *Annotation* is of a feature structure composed of a type with attributes. Therefore, analyzing the highlighted text from the pipeline requirements, the remarkable nouns corresponding to annotations are as follows:

- Input
- Question
- Answer
- Token
- N-Grams
- Precision

These are the annotation which can be inferred to be implemented within the required system. However, due to the architecture within UIMA, the system requires more than one

annotator. Therefore, our system will be defined by an Aggregate Analysis Engine. For these reason the system requires at least three Annotators:

- A parser annotator which decomposes the input text into questions and answers.
- An annotator which creates a ranked list according to the n-grams.
- An annotator which brings out the results with the corresponding precision.

In such way that the system has a flow as described in the following Figure:

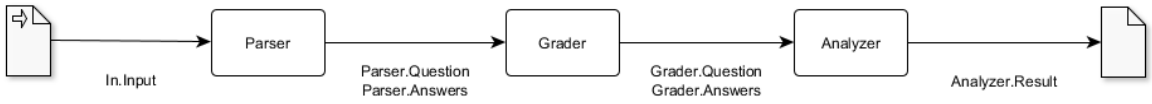

Figure 1 – System flowchart

In the Parser annotator the input file of the system will be accepted as input, as well as a configuration file to determine how the system should work. The Parser annotator will parse the sentences as defined in the specification and bring out as a result two outputs: a question and a list of answers. This output will be fed to the Grader annotator, where an analysis of comparison will be done through N-grams analysis. This annotator will give a score to the questions and feed the Analyzer annotator with the results. Finally, the Analyzer annotator will bring out a result containing the precision of the result.

Therefore the types defined for our three annotator analysis engine and following the nomenclature rules established for the types are defined as follows:

- In.Input
- Parser.Question
- Parser.Answers
- Grader.Question
- Grader.Answers
- Analyzer.Result

The following diagram denotes the specifics of each type as a class diagram. The colors denote the phase to which each class belongs to from the first to the last one. As we can perceive from this class design, it is quite simple if we consider that the Grader annotator is in charge of decomposing each sentence into tokens and grading each answer accordingly. In this way it is not necessary to define the n-gram nor the token type as it would be done from a first approach.

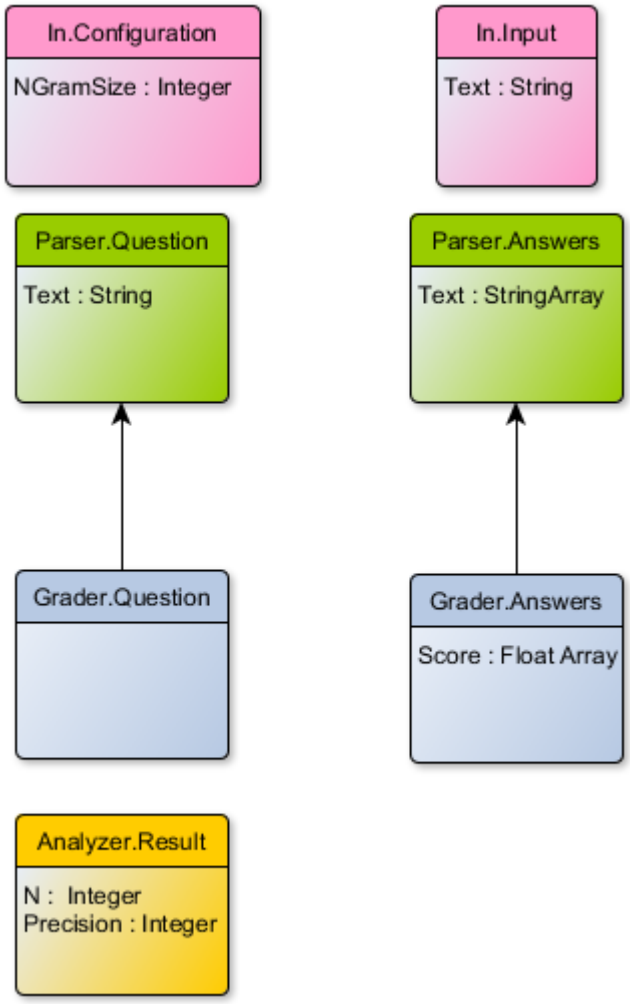

Figure 2 – Type system class diagram