11-791 Design and Engineering of Intelligent Information System
Assignment 2

Laleh Roosta Pour, AndrewID: lroostap

In this homework, I create analysis engines based on UIMA SDK and create Java annotators for the Sample Information Processing Task. To this end, I create four analysis engines.

1. **ElementAnnotator:** I create an ElementAnnotator descriptor with a java class ElementAnnotator. This annotator detects two types of element, `Question` and `Answer`. According to the pattern of question and answer in the input text file, these elements are found and saved as two system types (**Question**, **Answer**) in the JCas object.

2. **TokenAnnotator:** I create an TokenAnnotator descriptor with a java class TokenAnnotator. The JCas object is now passed to this annotator with a question and answers in it. For each element (question and answer) TokenAnnotator tokenizes the covered text of that element. It uses the Stanford nlp package for the tokenizer. In order to have no redundancy in the code, I created a new system type `Sentence` as a superclass of `Question` and `Answer` types. Then, the method tokenize is called with the Sentence type as an input. As a result in this annotator an array of Tokens will be assign to Sentence (which means to each Question and Answers).

3. **NGramAnnotator:** I create an NGramAnnotator descriptor with a java class NGramAnnotator. This Annotater retrieve the question and answers and the related tokens from the JCas object and generates Ngrams (unigram, bigram, and trigram) and add it to the JCas object. The N feature keeps the size of N in NGram (1,2,3), the `Owner` keeps which element is related to the current NGram. These NGrams are being used in the next step for calculating scores for each answer.

4. **AnswerScoringAnnotator:** I create an AnswerScoringAnnotator descriptor with a java class AnswerScoringAnnotator to calculate a score for each answer in the input. Now the Ngrams are in the JCas and passed to this Annotator. I used a very simple NGram checking algorithm to calculate a score for each answer. Whenever a unigram, bigram, or trigram in a question matches with a unigram, bigram, or trigram in an answer then a counter will be increased by 1, 2, 3. And at the end the numbers will be normalized and scores will be assigned to each answer. Adding the size of NGram to the counter for each match, could be considered as applying weights for the NGrams. We know that finding trigram give use a better confidence to detect a correct answer compare to the bigram and unigram. Note that this was a Naive technique to detect NGrams. There are many other algorithm and tricks to improve this matching. The score and the link to the element are kept in the AnswerScore type system. An the end the result will be printed in the console in the defined format.

In all these java classes I implemented the process method which gets a JCas object. Each of these classes modify the JCas object and pass it to the other class.

After implementing all these classes and the analysis engines, the aggregate analysis engine was implemented to aggregate all these engines and make the pipeline works properly. In this analysis engine the input and output of each engines will be specified with the proper order.