Name: Dishan Gupta

AndrewID: dishang

LOGICAL ARCHITECTURE DESIGN

The naming conventions that I have used for the annotators should make the design self-explanatory. For this reason, I am not providing a diagram (or figure) detailing the process flow.

Token - The Token type in the type system is implemented by TokenAnnotator.java and the Analysis Engine descriptor for the same is given by TokenAnnotator.xml. It annotates all the tokens present in the text document. You can test it by using UIMA Document Analyzer.

Question - The Question type in the type system is implemented by QuestionAnnotator.java and the Analysis Engine descriptor for the same is given by QuestionAnnotator.xml. It annotates the question (I have assumed there would only be one) present in the text document. You can test it by using UIMA Document Analyzer.

Answer - The Answer type in the type system is implemented by AnswerAnnotator.java and the Analysis Engine descriptor for the same is given by AnswerAnnotator.xml. It annotates all the sample answers (I have assumed they would immediately follow the question) present in the text document. It also stores their gold-standard rating of whether they are correct or not via the use of the 'isCorrect' feature in the Answer data type. You can test it by using UIMA Document Analyzer.

NGram - The NGram type in the type system is implemented by TokenUniGramAnnotator.java, TokenBiGramAnnotator.java and TokenTriGramAnnotator.java, and the Analysis Engine descriptors for the same are given by TokenUniGramAnnotator.xml, TokenBiGramAnnotator.xml and TokenTriGramAnnotator.xml, respectively. As the names suggest, they annotate unigram, bigram and trigram tokens in the text document respectively. I have not used the 'elementType' and 'elements' features since for each type (length) of token I have a separate Analysis Engine implementation.   You can test it by using UIMA Document Analyzer.

AnswerScore - The AnswerScore type in the type system is implemented by AnswerScoreAnnotator.java and the Analysis Engine descriptor for the same is given by AnswerScoreAnnotator.xml. This is where all the information from the NGram, the Question and the Answer annotations types is extracted and a score for each answer is computed. The higher the score, the more related the answer to the question. I have used weighted NGram overlap as the scoring function, where we calculate the number of unigram, bigram and trigram matches between the question and a particular answer. The weights are set to 1, 2 and 3 respectively, for the type of match. The scores for each type are normalized by the total number of that type of token present in the answer. The 'answer' feature stores the answer in current consideration and the 'score' feature stores the corresponding score. You can test it by using UIMA Document Analyzer.

Evaluate - I created the Evaluate type in the type system to store the final precision score for the system. There could have been other ways to store the precision score (like storing it in any of the other data types), but this approach seemed better in terms of design to me. The Evaluate type is implemented by Evaluator.java and the Analysis Engine descriptor for the same is given by Evaluator.xml. It essentially takes the AnswerScore type as input and iterates over all the AnswerScore annotations. Then it sorts the answers based on the scores computed through the scoring function (described in previous section) and checks the gold-standard ratings (correct or not) of the top N answers, where N is the total number of correct answers in the sample as determined by the gold-standard. The precision is basically the ratio of the total correct answers in top N and N. The precision score is stored in the 'precision' feature. You can test it by using UIMA Document Analyzer.

CONNECTING TO REMOTE SERVICE

This is accomplished by the scnlp-dishang-client.xml descriptor which connects to the remotely deployed Stanford CoreNLP service. The Stanford CoreNLP service essentially provides us with the NamedEntityMention annotations to be used in our Aggregate Analysis Engine processing pipeline.

AGGREGATE ANALYSIS ENGINE

The AAE combines the above mentioned AEs in the following order using Fixed Flow :

1. TokenAnnotator.xml - Input = Document Annotation ; Output = Token Annotations

2. QuestionAnnotator.xml - Input = Document Annotation ; Output = Question Annotation

3. AnswerAnnotator.xml - Input = Document Annotation ; Output = Answer Annotations

4. TokenUniGramAnnotator.xml - Input = Document Annotation ; Output = UniGram Annotations

5. TokenBiGramAnnotator.xml - Input = Document Annotation ; Output = BiGram Annotations

6. TokenTriGramAnnotator.xml - Input = Document Annotation ; Output = TriGram Annotations

7. scnlp-dishang-client.xml-Input = Document Annotation ; Output = NamedEntityMention Annotations

7. AnswerScoreAnnotator.xml - Input = Question, Answer, Unigram, Bigram and Trigram Annotations ; Output = AnswerScore Annotations

8. Evaluator.xml - Input = AnswerScore Annotations ; Output = Evaluate Annotations

COLLECTION PROCESSING ENGINE 1

The first Collection Processing Engine contains the following elements:

1. FileSystemCollectionReader.xml - Input = Input Directory; Output = Document Annotations

2. hw2-dishang-aae.xml - Input = Document Annotations; Output = Precision Score (with and without Named Entity)

3. EvaluatorCasConsumer.xml - Post-processes the CAS after it has been worked on by the HW2 Aggregate Analysis Engine. Basically, prints the output precision scores by extracting Evaluate annotations.

The descriptor for CPE 1 is given by hw3-dishang-CPE.xml.

DEPLOYMENT

The AAE for HW2 is deployed by first starting the broker locally and then deploying the hw2-dishang-aae-deploy.xml descriptor. This descriptor basically establishes the brokerURL (as determined by the broker) and the endpoint for the locally deployed service.

COLLECTION PROCESSING ENGINE 2

The second Collection Processing Engine contains the following elements:

1. FileSystemCollectionReader.xml - Input = Input Directory; Output = Document Annotations

2. hw2-dishang-aae-client.xml - Connects to the locally deployed analysis engine.

3. No CAS Consumer was required in this case. The precision output with and without the use of NamedEntityMention were print on the console.

The descriptor for CPE 2 is given by hw3-dishang-aae-as-CPE.xml.