11-791 Design and Engineering of Intelligent Information System
Assignment 1
Laleh Roosta Pour, AndrewID: lroostap

We want to design a logical data model for a sample information processing task. We have a question and a set of answers as an input and a pipeline including 5 steps to score the answers and evaluate the result. Now lets analyze the requirement and explain the data model according to the each step.

1. Before going through the pipeline steps, I created a type "BaseAnnotation"which has features `source` and `confidence` to use it whenever we want to use annotation with confidence .

2. In the first step the system reads an input and recognizes a question, answers and for each answer a value to show whether it is a correct answer or not. Later in the pipeline we want to process each answer and the question. I consider two type system "question"and "answer". answer is subtypes of "BaseAnnotation"(which is subclass of "Annotation"). Annotation as the superclass has two `begin` and `end` features which match our requirement for identifying question and answers in the input file in this scenario. These two elements (question and answer) are necessary since they are going to be referred to, and passed through the pipeline. In order to capture whether a specific answer is correct or not I add a feature `isCorrect` as a Boolean (True if the answer is correct and False otherwise). I also added a feature `question` to the answer type so that each answer has a pointer to the question. having source and confidence is very useful here. When a person is not sure whether an answer to a question is correct or not, he/she can represent it with these features and it has impacts on our evaluation later.

3. The second step is annotation of tokens.Tokens are the basic and most fundamental element in NLP. In most in not all NLP processing cases tokenization is one of the pre-processing steps. In this task also the input is tokenized the tokens will be used late. Therefore, I added a type "token"as a subclass of BaseAnnotation.

4. The third step is ngram annotation. As mentioned in the step explanation 1-, 2- and 3-grams of consecutive tokens will be annotated. Therefore, I created a type "ngram"which has a feature, `elements`, which is a FSList of the "token"type that we created. I also added a feature, `ngramOrder` as an integer which defines the order of ngram (in other words size of `n`; 1, 2, or 3).

   Note that the ngram annotation is on the consecutive tokens, therefore we can add the feature sofa for the ngram type and indicate the Subject of Analysis of it another document which is the sequence of tokens.

5. The forth step is answer scoring. What we want to capture in this phase is the score that the system calculated for each answer. I create a type “answerScore”with a feature, `answer`, which is an answer type that we created, then each answer score is assigned to an answer. The other feature `score` is to save the score that the system assign to the `answer`.

6. The last step is sorting the answers based on their scores and calculating the precision at N, which is the total number of correct answers. I created the “ranking”type which record the precision in the `precision` feature, and has a FSList of “answerScore”from which we can have the top N correct answers with their scores. That is because, ranking has pointers to each answerScores via its asnwerScore feature and each answerScore type points to an answer type via its own answer feature. The ranking type is a subtype of the general type “TOP”, which is the root of the type system, since the ranking nature is not similar to the previous types we had (the subtype of annotation). Since each answer has a pointer to the question, we have also the question via this type and we can capture and print the result with this class.

   Note that since we assumed that there is only 1 question in the whole pipeline then all the answers are pointed to one question and the is no difficulties in our system. However there are other options to have pointers between question and answer. we can easily add a feature as an instance of “question”type to the answer type.

Persistence:

So far the design assumption has been that the system does not require persistence, i.e., the program runs in the memory and the resulting sorted scores are generated. However, in case the system requires persistence, in which case these system types are stored in files or a database for later use, some changes are necessary. In such a case instead of using object pointers, the answer to a question has a pointer to the question, we need to maintain an object ID for the question so that the answer can refer to it. Essentially, the object pointers act as an object ID which has to be made explicit in case the records have to be stored in a file. As a result the following changes are necessary:

1. The token should have a feature `tokenID`. Then, in the ngram type, the feature `elements` should be a FSList of Integers (which are tokenID).

2. The answerScore should have a feature `answerScoreID`. Then in the ranking type, the feature `elements` should be a FSList of Integers (which are answerScoreID).

3. The answer should have a feature `answerID`. Then, in the answerScore, the feature `answer` should be an integer (which is an answerID).

4. The question should have a feature `questionID`. Then, in the answer, the feature `question` should be an integer (which is a questionID).

Note that I think that we can have confidence and source anywhere that we have annotation, in this example I put it for tokens and answer, but in general I think it could also be in other types of annotations, the cases that the system estimate something, it can show how confident the system was).