Andrew Wilkinson
DEIIS Fall 2015

"Project Individual" 2 Report: UIMA Type System

The following is a précis of the types I have created for my version of the question answering system of this week's homework assignment. I will describe the system roughly from the top down and provide concise reasoning for my choices.

There are two distinct hierarchical structures, as can be seen in aewilkin-pi2-diagram.pdf. The first has the built-in "Annotation" type at the top and the other has "TOP" at the top. First, the hierarchy inheriting from "Annotation"…

AnnoPlus: a subtype of Annotation that has the features "Confidence" (float) and "ComponentName" (String). These features are there because the homework instructed us that "…every annotation must record the following: (1) the name of a component that produces the annotation, and (2) the component's confidence score assigned to the annotation." So, all other types in this hierarchy will inherit these features.

DocumentAnno: a subtype of AnnoPlus that will contain all the text for one whole input .txt document. I don't know whether this level of representation is actually necessary—one of many mysteries of UIMA thus far—but in case it is not redundant, here it is.

LineAnno: a subtype of DocumentAnno that contains one individual line from an input document, separated at the newline character. These lines may contain either a question or an answer. This type does not have a feature for line number, because I couldn't think of a need for that information in this application. But it could have that.

AnswerAnno: a subtype of LineAnno that contains one individual line with an answer on it, separated at the newline character. So, this type is coterminous with a LineAnno but is specifically for answer lines. It has features that question lines don't have, since there is different information given in a line depending on whether it is a question or

an answer. Specifically, it has features "AnswerNumber" (int) to be extracted from the question number markers "A1", "A2", etc.; and "IsCorrectAnswer" (boolean) to be derived from the "0" or "1" indicators.

AnswerActual: a subtype of AnswerAnno that contains the natural language part of an answer line, without the "A*X*" or "0"/"1" designations included. E.g., "Booth was assassinated by Lincoln." This will be what the tokenizer uses as input.

AnswerAnnoTokenized: a subtype of AnswerAnno that contains and implements a tokenizer that uses the (string versions of) AnswerActual annotations as input and outputs those tokenized sentences in the form of a StringArray feature. I.e., each token in the sentence is one string in the array.

AnswerOneGrams: a subtype of AnswerAnnoTokenized that uses the StringArray feature of the parent AnswerAnnoTokenized type as input and outputs a StringArray of 1-grams as a feature. (In actuality, this StringArray of 1-grams will look identical to the StringArray feature of tokens used as input.)

AnswerTwoGrams: a subtype of AnswerAnnoTokenized that uses the StringArray feature of the parent AnswerAnnoTokenized type as input and outputs a StringArray of 2-grams as a feature. E.g., ["Booth was", "was assassinated", "assassinated by", "by Lincoln", "Lincoln ."]

AnswerThreeGrams: like AnswerTwoGrams but with 3-grams.

AnswerScorer: a subtype of AnswerAnno that implements code to score the answer contained in its annotation. It will compare AnswerOneGrams to QuestionOneGrams, AnswerTwoGrams to QuestionTwoGrams, and AnswerThreeGrams to QuestionThreeGrams to determine similarity measures between those three n-gram levels of representation, and use those measures to calculate an overall similarity score that will be used in classifying the sentence as either a correct or an incorrect answer later on in the Sorter type when it is ranked with the other sentences. This type has the feature "Score" (float).

QuestionAnno: a subtype of LineAnno that contains one individual line with a question on it, separated at the newline character. So, this type is coterminous with a LineAnno but is specifically for question lines. It does not have certain features that answer lines have, since there is different information given in a line depending on whether it is a question or an answer. It contains the initial character "Q", which could be used to differentiate it from answer lines (although other methods could be used to do that, so I didn't make that into a feature).

QuestionActual: a subtype of QuestionAnno that contains the natural language part of a question line, without the "Q" designation included. E.g., "John loves Mary?" This will be what the tokenizer uses as input.

QuestionAnnoTokenized: a subtype of QuestionAnno that contains and implements a tokenizer that uses the (string versions of) QuestionActual annotations as input and outputs those tokenized sentences in the form of a StringArray feature. I.e., each token in the question is one string in the array.

QuestionOneGrams: a subtype of QuestionAnnoTokenized that uses the StringArray feature of the parent QuestionAnnoTokenized type as input and outputs a StringArray of 1-grams as a feature. (In actuality, this StringArray of 1-grams will look identical to the StringArray feature of tokens used as input.)

QuestionTwoGrams: a subtype of QuestionAnnoTokenized that uses the StringArray feature of the parent QuestionAnnoTokenized type as input and outputs a StringArray of 2-grams as a feature. E.g., ["Booth shot", "shot Lincoln", "Lincoln ?"]

QuestionThreeGrams: like QuestionTwoGrams but with 3-grams.

The other hierarchy inherits from "TOP." I chose this because this structure contains the type that sorts answers according to their scores and the type that calculates and returns a precision score, and neither of those elements seemed like they needed to be of the type "Annotation" like the rest of the types. I chose "TOP" because I didn't know

what else to choose as a supertype from which to inherit. In my imagining, this is where the "Main" method is found (maybe? this is another of UIMA's mysteries so far).

Sorter: a subtype of TOP that takes as input all the outputs from the AnswerScorer type run over all the answer sentences in a .txt document and ranks the sentences according to their score. The feature "SortedAnswers" is a StringList reflecting this ranking. The other features, "Confidence" (float) and "ComponentName" (String) are there because I was feeling like, "Better safe than sorry; I should include these because that seems to be what they're asking for on this assignment, even if this type is not an 'Annotation.'"

PrecisionCalculator: a subtype of Sorter that takes as input the SortedAnswers feature of Sorter and adds up the number of IsCorrectAnswer boolean features from AnswerAnno and uses those values to calculate the precision of this piece of software over the inputs. It returns that value as the feature "Precision" (float); it also includes "Confidence" and "ComponentName" (String).

How about an Interesting Discovery? I discovered (with the help of Vivian R.) that if I'm spending a bunch of time adding the same features to every type in a hierarchy (as I had initially done with "Confidence" and "ComponentName," I should stop wasting my time and add a type higher up that has those features, that they may be inherited by all subtypes.