Project Individual 4: Component and Type Design Patterns

Leah Nicolich-Henkin: lnicolic

September 28, 2015

Overview

This report outlines a UIMA architecture to rank different passages based on whether they provide answers to a given question. This task is based off the NIST TREC 2003 task and dataset. The system assumes an single input file that includes questions, labeled with a question ID, and passages, labeled with the same question ID, as well as a source document ID, and the ground truth of whether they contain the answer to the question. It processes these and writes the passages to an output file – `passageRanking.txt` – with the ground truth changed to the score of the given passage.

Type System

In the type system used for this project, every `Annotation` extends the type `ComponentAnnotation`, which includes *component ID* and *score* features. The main types of annotations are `Questions`, which include an *ID* and a *sentence*, and `Passages`, which include the same, and also a *source document ID* and a *label* indicating whether the `Passage` contains an answer to the question.

Where I changed the given type system is that the other `Annotation`, instead of the `InputDocument`, is a `QuestionSet`. A `QuestionSet` consists of a single `Question`, and an FSArray of `Passages`. Essentially, it's a way to associate questions and passages by *question id*, rather than figuring out their association separately in each annotator. By using this `QuestionSet` type, it is possible for an annotator to simply loop through the `QuestionSets` and have all the information necessary for calculating a *score* available and isolated at once.

If we were working with multiple input documents, an `InputDocument` type might contain an array of `QuestionSets`, but currently that is not the case.

Descriptor Design

The CPE points to three components: the collection reader, the aggregate analysis engine, and the consumer. In this case, the collection reader simply reads in the content of the input file and adds it to the CAS. The aggregate analysis engine consists of four annotator descriptors – described below – but no other AAEs. The consumer receives QuestionSets complete with scored Passages, and is merely responsible for iterating over the Passages and printing them to the output file.

Annotation Pipeline

Question Annotator This annotator simply goes through the input document and uses a regular expression to annotate the questions, including their question ID and sentence.

Passage Annotator This annotator uses a regular expressions to annotate the features of a Passage: question ID, source document ID, label, and sentence. In addition, when it annotates the sentence, it performs some basic pre-processing to remove html tags and unrecognized characters.

QuestionSet Annotator This annotator takes in both Questions and Passages, and combines them as appropriate into Question Sets. It does this by first storing them in separate HashMaps by question ID, and then retreiving them together.

Score Annotator This annotator takes in Question Sets, and calculates the score feature for each Passage.

Methods

The ranking method implemented in the Score Annotator is the MITRE method of word overlap, which is essentially a unigram comparison. The score is the percentage of words in the question that also occur in the passage. It is not a particularly powerful method, but serves as a good baseline for future work. Its main flaws are that it can't recognize different forms of the same word (ex. "die" vs. "died"), and it has no connection to word order. These will be addressed in future projects.