Jung In Lee
AndrewID: junginl

Architectural Aspects:

Information system is about producing information types and knowledge types from the given raw data. Appropriate design and engineering must be supported in order to build a good Information System. Before taking on any of these, the first necessary step is to identify the goals and requirements of the given task.

Requirements & Use Cases

Given a text data with a question and a set of answer candidates, the goal of this project was to identify which of the answer candidates actually answer the given question. In order to perform this, we assigned a score to each of the answer candidates, and for evaluation, we ranked the candidates with regards to their computed scores and then picked the top N candidates to compute the precision, which is our performance measure for the project.

UML Use Case Model would use diagrams to simplify the above paragraph of Use Cases.

Domain Modeling

Domain modeling is the process of visualizing abstract concepts. UML Domain Modeling would have diagrams of objects and the relevant associations and attributes. Yet, for this project, listing noun phrases was sufficient to help identify the conceptual classes.

List of Noun Phrases
Question
Answers
Tokens
N-grams
Scores
Precision
…

Type System Design

1.Base Annotation
- source: a string feature to keep track of where the annotation was created
- confidence: a double feature. a numeric measure computing the confidence of the correctness of the output
- begin: integer feature. marking the beginning position of the sentence or tokens
- end: integer feature. marking the ending position of the sentence of tokens

2. Test Element Annotation
Input: The input textile the system reads in. In type Sofa.

Output:
Question: stringList. The question sentence in the input file
AnswerList: stringList. The list of candidate answer sentences. Each answer candidate sentence can be extracted from this list.
IsCorrect: boolean. The 0/1 binary values associated with each candidate answers, with 1 indicating that it is correct and 0 otherwise.

3. Token Annotation
Input: Question and AnswerList
Output:
TokenList: StringList. A list tokens for each question and answer candidates
Word: String. Each string value from the TokenList (for the sake of convenience)

4. NGram Annotation
Input: TokenList
Output:
OneGram: StringList. A list of 1-grams for each question and answer candidates
TwoGram: StringList. A list of 2-grams for each question and answer candidates
ThreeGram: StringList. A list of 3-grams for each question and answer candidates
elements: FSList. A list with a pointer that iterates over the list.

5. Answer Scoring
Input: Question, AnswerList
Intermediate steps:
QuestionTokens: StringList. A list of n-grams for the question sentence to be reviewed for comparison with n-grams of each answer candidates
AnswerTokens: StringList. A list of n-grams for each answer candidates
Output:
Score: double. A score for each answer candidate. Number of question n-grams / number of answer candidate's n-grams

6. Evaluation
Input: Score, AnswerList, IsCorrect
Intermediate steps:
NumCorrect: IntegerArray. The number of the answer candidates with the score 1, i.e. sentences identified as correct by the system
NumPredicted: IntegerArray. The number of the answer candidates with the initial value 1, i.e. sentences indicated as correct in the first place

Algorithmic Aspects:Architectural Aspects

The main algorithm used in this project is the simple yet often powerful N-gram annotations. We first break down the sentences (both the question and answer

candidates) into a single token, two sequences of tokens or three sequences of tokens depending on which N-gram we are examine at hand. Then we compare the question tokens and each of the answer candidate's tokens to assign scores to each candidate. The score is computed by looking at how many N-grams overlap between the question and the answer candidate, i.e. question N-grams/ answer candidate's total N-grams.

Possible ways of improvement:

For now, we are just comparing the scores computed as above with the given gold standard, i.e. assigning 1 if correct and 0 otherwise. Yet, it would be much more interesting to compare this set of scores with other sets of scores computed using different algorithms such as the Bag of Words analysis. Our intuition leads to the conclusion that the N-gram method would perform better than BOW, but numerical analysis supporting this point would make our argument much stronger.

Implementing more knowledge-based algorithms would improve the performance. Incorporating lexical features would successfully identify that the two sentences "Booth shot Lincoln" and "Booth assassinated Lincoln" have the same meaning. Furthermore, adding semantic roles to the features would enable the system to recognize the passive tense, detecting the sentence like "Lincoln was shot by Booth" as correct.