HW2-Report

Kartik Goyal
kartikgo@cs.cmu.edu

September 23, 2013

1 Type System

The type system is largely similar to the one provided to us. The only changes made are the ones to 'Token', where two features are added.

- POS: The part of speech tag associated with the token.
- lemma: The lematization of token

The above two features will help in creating better score matches. They tend to generalize the tokens. For eg. John, Mary etc $\rightarrow$ NNP
eat, ate $\rightarrow$ eat:lemma etc.

2 Annotator Engines

Following annotator Engnes have been designed:

- TestElementAnnotator: This annotates the questions and answers, which are identified by regular expressions whose patterns are in the configuration part of xml. These spans help in matching tokens and NGrams later on. Answers 'isCorrect' feature is also set
- TokenAnnotator: The tokens, tokenized by stanford corenlp toolkt, are annotated by this method. Every token belonging to a span(question/answer) has an identifier in casProcessorId of the form 'stanford:question' or 'stanford:answer-no.'. This identifier makes it very easy to find NGrams for the relevant text. Their begin and end are marked accordingly.This annotator takes questions and answers as input.
- NGram Annotators: These take tokens as input and output NGrams for relevant spans of text. Their begin and end are set according to the span they cover.Following annotators are NGram annotators:
  - UniGramAnnotator
  - BiGramAnnotator
  - TriGramAnnotator

- ScoreAnnotator: This annotator takes input as questions,answers, tokens and NGrams, and computes token,part of speech, lemma, ngram, ngram POS, ngram lemma overlaps of the relevant spans of questions and answers. Then it combines these overlap scores in a weighted linear manner. More importance has been given to surface form tokens and n-grams, then lemma and least importance has been given to part of speech tags as they tend to over-generalize and yield high overlaps for text spans. The output are 'AnswerScore' objects.
- Evaluator: This engine takes input as Questions and AnswerScore objects. It sort the answers according to the predicted score. It also calculates precision for each instance. Using AnswerScore it is very easy to calculate precision as 'isCorrect' feature will help us find the number of correct answers in the example and sorted out put then can be used to calculate the precision. Finally, using initialize and destroy functions, it calculates average precision of the system.
- hw2-kartikgo-aae: This is the final engine, which aggregates all other engines and runs them in the following order:

  - TestElementAnnotator
  - TokenAnnotator
  - BiGramAnnotator
  - UniGramAnnotator
  - TriGramAnnotator
  - ScoreAnnotator
  - Evaluator

3 Output

The output is in form of standard out which is printed by Evaluator.java. Also, the annotations can be found in the output of Document Analyser.