11-791 Design and Engineering of Intelligent Information System
Engineering and Error Analysis with UIMA

Name: Chen Wang
AndrewID: chenw1

Requirement Analysis:
In this homework, we are asked to implement all necessary annotator and to implement a UIMA pipeline that uses consine similarity to measure how relavent each answer is, given a query. In order to implement this whole pipeline, we need a collection reader, annotators, and cas consumer. The collection reader and basic document annotator is already provided. What still needs to be implemented are token annotator and a cas consumer that carries the cosine similarity computation logic.

Design:
I tried to design the pipeline in a conventional way as previous homeworks. The original desgin is, having a token annotator that tokenizes each sentence based on the word boundary. And cas consumer retrieves information from cas object every time processCas(CAS) function is called, and performs cosine similarity computation whenever collectionProcessComplete() is called. I found out later that this conventional approach did not work well with the given VectorSpaceRetrieval.java, because the main function reads in one line of text at a time and resets the Cas object after a call of processCas(CAS). That means, in order to perform cosine similarity computation, cas consumer needs make a copy of all annotation in the Cas object. For the purpose of only computing cosine similarity, I found it easier to, instead of making a copy of all annotation, only store the string of queries in cas consumer, and have cas consumer do the annotation when collectionProcessComplete() is called. But for the sake of completeness, I still keep the token annotator in the pipeline, even if these annotations are never really used by cas consumer.

Error analysis:
Cosine similarity approach works perfectly for query 1, 2, and 3. For query 4 and 5, the error comes from the following facts:
1. Capitalization. Some words in query and answer are exactly the same but cosine similarity fails to recognize them because capitalization. For query 5, even though the query and the correct answer both have word “old” in them, the correct answer gets a score of 0 because the word “old” in answer is capitalized. The easiest way to get around this is to change all words to lower case.

2. Frequent use of words with little actual meaning. One observation I have in the process of error analysis is, not every word carries the same amout of information about whether the answer should be rated higher or lower. Take query 4 for example, the answer “Behind every girls smile is a best friend who put it there” has higher score than any other answer but the only words it has in common with the query is “smile”, “is” and “a”. Even if the answer has “is” and “a”

as common words, but those two words are not a good indicator of how similar this answer is to the query. One possible way around this is ignore all words that do not carry much meaning. A simple set of such words could be “the”, “a”, “an”, “and”, “is”, “was” and “were”.

3. Plural form of nouns. Some words are the same but program fails to recoganize because of plural form. This could be solved with existing NLP package.

After applying the above modification, MRR increases from 0.767 to 0.8.

Bonus:

	Jaccard Coefficient	Dice Coefficient	Cosine similarity
MRR	0.533	0.867	0.8 (modified)