Homework 4 - Report

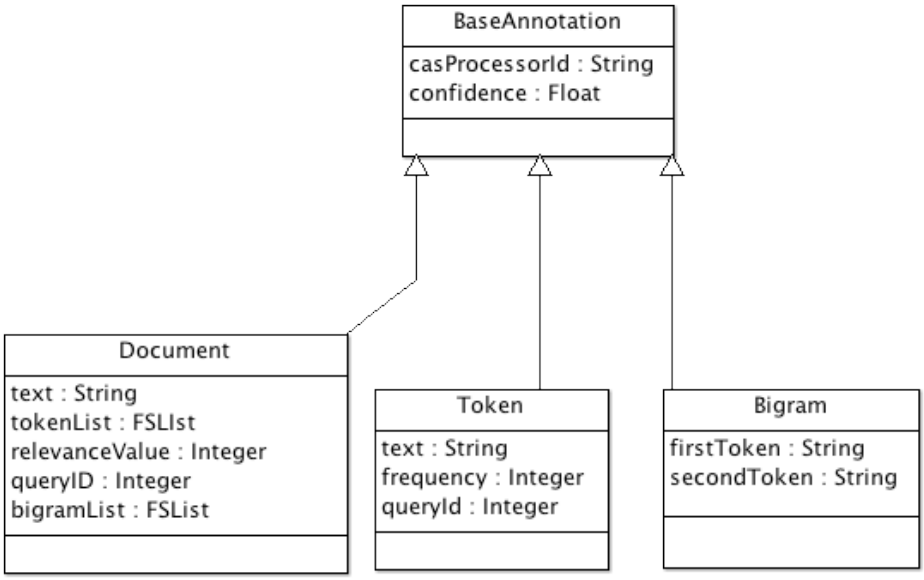
Vinay V Vemuri (Andrew ID: vvv)

Requirements

1. **Computing a Cosine Similarity Score for all answers -** Consider each query and its corresponding answers and for each answer compute a cosine similarity score. Rank each document answer using the cosine similarity score. Using the ranked document answers for all queries, compute a mean reciprocal rank for each query.
2. **Error Analysis -** Using the computed cosine similarity scores, perform error analysis describing why the cosine similarity measure fails to assign the highest to the correct answer. Design a better retrieval system to improve the MRR value.
3. **Description of Design -** Describe the system design and other architectural and algorithmic aspects of the design.
4. **(Bonus) Other Similarity measures** - Consider other similarity measures other than the cosine similarity and repeat steps 1 to 3 for the new similarity measure.

Type-system

The UML Class diagram shown below provides a visualization of the various Java type classes that are produced when the proposed type system is compiled into Java type classes.



Class diagram constructed after compiling proposed type system into Java type classes.

- **BaseAnnotation**: The BaseAnnotation type is the supertype of all the types in the proposed type system. The BaseAnnotation type inherits from the uima.tcas.Annotation type. Its purpose is to ensure that all subtypes have a CasProcessorId (to indicate which class made the annotation) and a confidence value (to indicate how confident the source of the annotation was).
- **Token:** The Token type captures information about tokens in a Sentence (delimited by space). The 'text' feature captures information about the text contained in the token. The 'frequency' feature represents the number of times the token occurs in a sentence. The 'queryId' feature keeps track of the query with which the feature is associated. This type is primarily used in the computation of the cosine similarity measure.
- **Bigram**: The Bigram type captures information about the bigrams contained in each document. The 'firstToken' feature represents the first word in the bigram whereas the 'secondToken' feature represents the second word of the Bigram. This type is primarily used in the computation of the Dice coefficient measure.

Similarity Measures Used & Results

Cosine Similarity

Using Cosine Similarity:

Score: 0.45226702 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.30618622 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.50709255 rank=1 rel=1 qid=3 The best mirror is an old friend
Score: 0.17213259 rank=3 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.15811388 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.8666666686534882

Dice Coefficient

Score: 0.18181818 rank=2 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.11111111 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.00000000 rank=3 rel=1 qid=3 The best mirror is an old friend
Score: 0.10000000 rank=3 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.00000000 rank=2 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.5333333373069763

Error Analysis

There are several issues with the current implementation of the cosine similarity and dice coefficient measures. The following table points out these issues and proposes solutions which if implemented will improve the MRR value.

Issue	Proposed Solution
During tokenization, punctuation is not being used as a delimiter. As a result the word ‘coin’ that occurs at the end of a sentence just before a period for example will be tokenized as ‘**coin.**’ and will not match the word ‘**coin**’. Similarly the word ‘smile’ just before a comma will be tokenized as the word ‘**smile,**’ and will not match the word ‘**smile**’.	Use punctuation as a delimiter and remove all punctuation during tokenization with the possible exception of apostrophe.
The tokenizer used in this project does not consider words like ‘categorize’ and ‘category’ to be the same even though they have same stem.	Perform stemming (Using the Porter Stemmer for example) on all words in the query and answer and compute cosine similarity on the stemmed tokens.
Determining what a ‘word’ is is not trivial. Consider for example ‘New York’ and ‘Rock 'n' roll’. Many would consider both of these to be a single word as opposed to multiple words. But a tokenizer that uses only whitespace for tokenization will treat these as two and three tokens respectively. When we treat ‘New York‘ as two words as opposed to one word, the words ‘New York’ and ‘New Zealand’ will have one word that matches but when we treat them as a single word, they will have no matches at all. The latter is more ideal since ‘New York’ and ‘New Zealand’ are actually not similar at all.	Change the tokenizer to treat words like ‘New York’, ‘Rock n’ roll’ etc to be a single word instead of multiple words.