Homework 3 Report -- Xinyun (Victor) Zhao, Andrew ID: xinyunzh

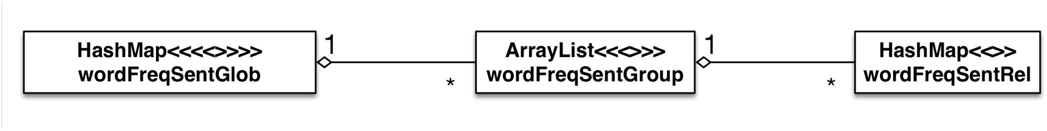
Task 1 -- Implementation of Basic Information Retreival Function

Detailed Implementation

The original data structure in RetrievalEvaluator

In this CASConsumer, the most challenging part is the design of the data structures that are used to store the information from CAS/Annotation. Since the UIMA framework doesn't preserve the results after each flow in the pipeline, several static nested Collection-based data structures have been created to store these data in memory. First, the consumer use a four-level HashMap to store all the information about each query and document, including term frequency, text of the words, Qid, relevance and so on. The reason of using a complex data structure to store the information is the in-memory characteristic, which is faster than retrieving from the file. Furthermore, by unfolding each level of nested data structure, the coherence within the same query has been enhanced because each document of same query locates in the same level of data structure. For instance, if a subroutine analyzes one query, unfolding the outside HashMap to certain level will achieve this goal instead of rebuild the data structure from serialized data.

On the other hand, in order to calculate the cosine similarity and other results from produced data, some private methods have been created to introduce a modular approach.



Task 2 -- Error Analysis and Refinement

Error Classification

After analyzing the first 5 query in the sample input documents and , there are three main errors that happen during tokenizing phases. Along with these different types of errors, more advanced tokenizer may be applied to reduce the possibility of encoutering them.

- Cognate Words

  A lot of words have similar spelling that may derive from a common root. Therefore, one solution is to use stemmer to deduct these word into same token.

- Tenses

  For example, "am", "was", and "be" are the same token for the process. Otherwise, different document may have the same link word but has been catagorized into different tokens. Using stemmer can also avoid this senario.

- Punctuation

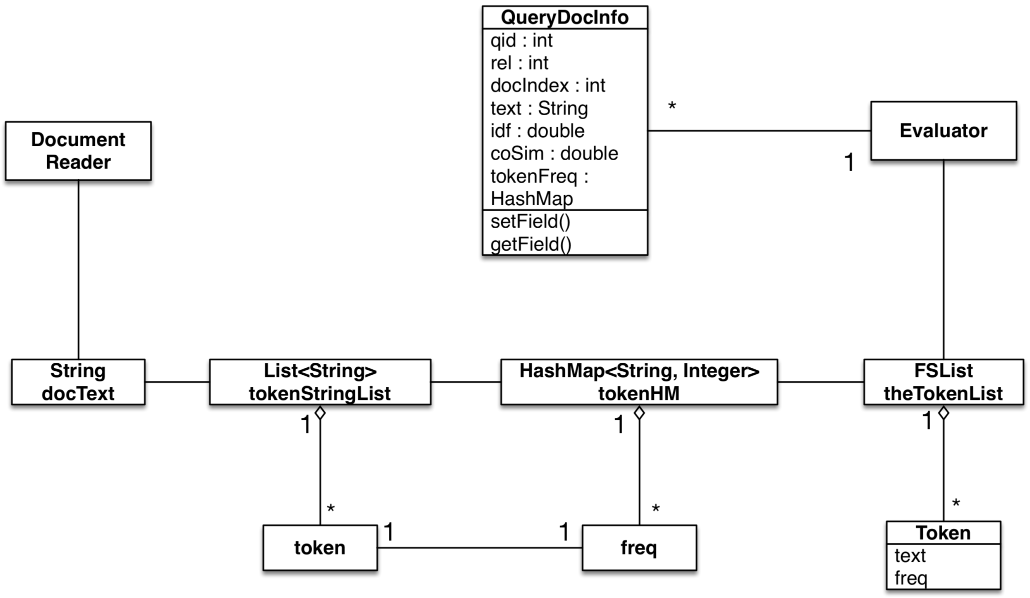
  Using regular expression of the built-in package of Java will remove this error, which is expressed as `"\\p{Punct}+"`.

- Stop Word

  Using the dictionary provided may be effective in this case. In terms of implementation, a HashSet that stores all these stop words can be used to look up if there is an existing stop word.

Design Refinement

Overview Architecture

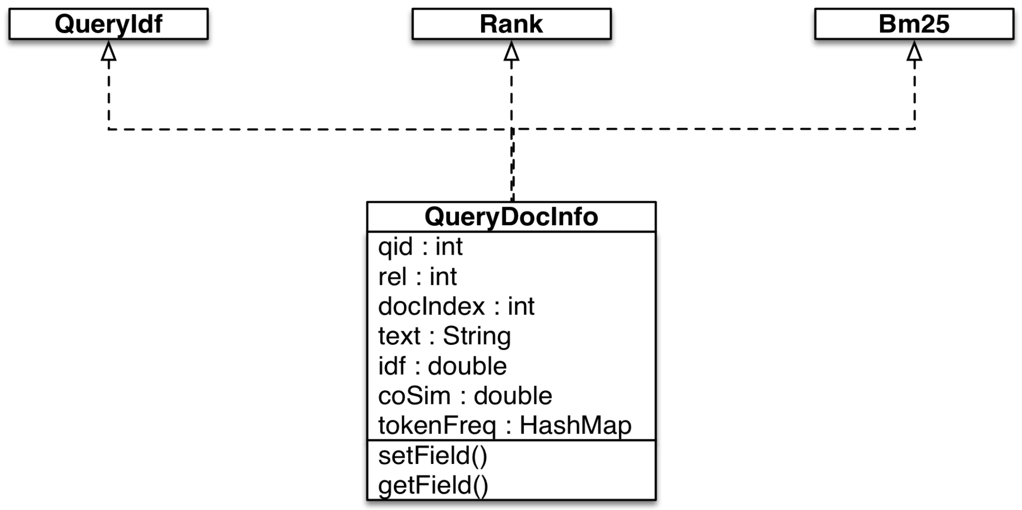


In task 2, the storage data structure has been changed from nested HashMap to a self defined class. Some necessary interfaces that are used to extensify the functionalities for other scoring machanism.

Global Storage

In order to achieve the flexibility of multiple types of information storage, the storage data structure has been redesigned to fulfill this requirement. In task 2, a new independent class has been implement to store all of the data that related to the whole process, which has also been implemented in the final upload version of task 1. In this diagram, the QueryDocInfo has serveral fields, which preserve all of the relevant information from JCas during the whole pipeline.

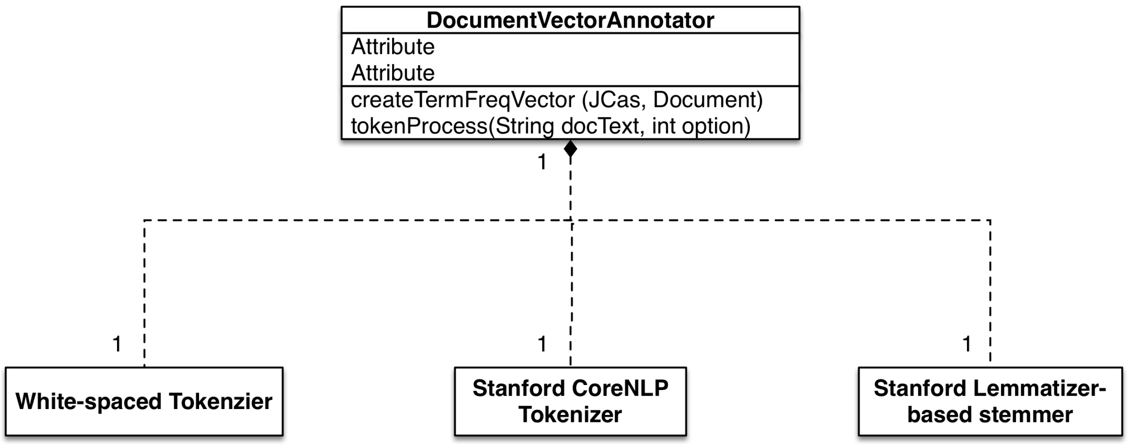
However, in terms of the system performance during processing documents, the class-based implementation is running a little slower than HashMap-based implementation, which is nearly 0.2s.



Another advantage of implementing interfaces of other scoring/weighting machanism, which are TF-IDF, BM25 and so on, is that the extensibility has been acheived through this architechture. Some of the methods that are used to calculate scores for document within one query are encapsulated into class **QueryDocInfo**.

The Synthesis of multiple tokenizer

Through an intuitive way of thinking, the most error-prone part of this system lies in the part of Document Vector Annoator. Apparently, using a white-spaced tokenizer may influence the most for the entire pipeline since it only separate different token based on the white space rather than other criteiras in syntax level. Therefore, the other two tokenizer packages have been included in the Document Vector Annotator, which is illustrated in the following diagram.


DocumentVectorAnnotator
Attribute
Attribute
createTermFreqVector (JCas, Document)
tokenProcess(String docText, int option)
1
1
1
1
White-spaced Tokenzier
Stanford CoreNLP Tokenizer
Stanford Lemmatizer-based stemmer

This new annotator consists of three ways of tokenization. In the real environment implementation, a parameter **option** has been passed to the method to let the **tokenProcess** decide which one need to be choosed.

```
/** The parameter for different choices of tokenizer **/
private static int NLPTOK = 1;
private static int STALEM = 2;
private static int WHISPA = 0;

private List<String> tokenProcess(String docText, int options) {
// using specific tokenizer to process the text.
}
```

The results of tokenization through different tokenizer

In terms of the first query, there are three different token serials by using different tokenizer.

Using white-spaced tokenizer

```
[Give, us, the, name, of, the, volcano, that, destroyed, the, ancient, city, of,
Pompeii]
[In, A.D., 79,, long-dormant, Mount, Vesuvius, erupted,, burying, in, volcanic,
ash, the, Roman, city, of, Pompeii;, an, estimated, 20,000, people, died.]
[You, can, see, Vesuvius, in, the, background,, near, ruins, of, Pompei;, its,
last, eruption, was, in, 1944.]
[Vesuvius, is, located, near, the, ruins, of, the, destroyed, city, of,
Pompeii.]
[In, 79, A.D.,, this, ancient, city, was, buried, in, an, avalanche, of, hot,
ash, from,    Mount, Vesuvius.]
(MRR) Mean Reciprocal Rank ::0.4375
```

Using Stanford CoreNLP tokenizer

```
[Give, us, the, name, of, the, volcano, that, destroyed, the, ancient, city, of,
Pompeii]
[In, A.D., 79, ,, long-dormant, Mount, Vesuvius, erupted, ,, burying, in,
volcanic, ash, the, Roman, city, of, Pompeii, ;, an, estimated, 20,000, people,
died, .]
[You, can, see, Vesuvius, in, the, background, ,, near, ruins, of, Pompei, ;,
its, last, eruption, was, in, 1944, .]
[Vesuvius, is, located, near, the, ruins, of, the, destroyed, city, of, Pompeii,
.]
[In, 79, A.D., ,, this, ancient, city, was, buried, in, an, avalanche, of, hot,
ash, from, Mount, Vesuvius, .]
 (MRR) Mean Reciprocal Rank ::0.5125
```

Using Stemmer

```
[give, us, the, name, of, the, volcano, that, destroy, the, ancient, city, of,
pompeius]
[in, a.d., 79,, long-dormant, mount, vesuvius, erupted,, bury, in, volcanic,
ash, the, roman, city, of, pompeii;, an, estimate, 20,000, people, died.]
[you, can, see, vesuvius, in, the, background,, near, ruin, of, pompei;, its,
last, eruption, be, in, 1944.]
[vesuvius, be, locate, near, the, ruin, of, the, destroy, city, of, pompeii.]
[in, 79, a.d.,, this, ancient, city, be, bury, in, an, avalanche, of, hot, ash,
from, mount, vesuviu.]
(MRR) Mean Reciprocal Rank ::0.5500
```

As it can be seen from the results above, comparing to the first two tokenizer, Stemmer remove most of the irrelavant information from single token, such as tense, reduction of plural form and so on, which increase the term frequency among one document. Since some of the scoring mechanism highly relie on this metric, such as normal cosine similarity or TF-IDF, this denoisying process influence a lot on following steps.

The implementation of TF-IDF scoring machanism

The main reason to apply TF-IDF on information retreival system is to reduce the effects of some common tokens that exist in multiple documents in order to make other words that may reside in one or few documents relatively unique. Here is the formula to calculate IDF:

$$idf(t, D) = \log \frac{N}{|d \in D : t \in d|}$$

- N is total number of documents in the corpus
- The denominator stands for number of documents where the term t appears

According to the characteristics of this formula, if the number of documents where the term t appears becomes larger, the value of idf decrease. Therefore, the common keywords that most of documents have will play insignificant roles among these documents.

The definition of TF-IDF is that

$$tfidf(t, d, D) = tf(t. d) * idf(t, D)$$

- tf(t, d) means the original term frequency of a specific token among its document

For example, if an article like "the, a, an" appears a lot across one set of queries, there is no reason to take the frequency of these particular words into consideration.

Here is the code of implementation of calculating IDF.

```
/**
* This static method reads the documents within one Query, and calculate the
* IDF value for each token.
*/
public static void calculateIdf(ArrayList<QueryDocInfo> groupQDI) {
    int d = groupQDI.size() - 1;
    ArrayList<QueryDocInfo> groupDocQDI = new ArrayList<QueryDocInfo>(groupQDI);
    for (QueryDocInfo qDI : groupDocQDI) {
        double idf = 0.0;
        Set<String> tokenSet = qDI.tokenFreq.keySet();
        Iterator<String> tokItr = tokenSet.iterator();
        while (tokItr.hasNext()) {
            int num = 0;
            String token = tokItr.next();
            for (QueryDocInfo anoQDI : groupDocQDI) {
                Set<String> anoTokenSet = anoQDI.tokenFreq.keySet();
                if (anoTokenSet.contains(token)) {
                    num++;
                }
            };
            idf = Math.log((double) d / (double) num);
            if (qDI.getRel() != 99) {
                qDI.tokenIdf.put(token, idf);
            }
        }
    }
}
```

Here is the outcome of the combination of TF-IDF scoring machanism and eliminating stopwords based on the given data set.

```
consine=0.0884  rank=3  qid=1   rel=1   In A.D. 79, long-dormant Mount
consine=0.1612  rank=3  qid=2   rel=1   When Michael Jordan--one of the
consine=0.6708  rank=1  qid=3   rel=1   Alaska was purchased from Russia in
consine=0.4529  rank=2  qid=4   rel=1   On March 2, 1962, Wilt Chamberlain
(MRR) Mean Reciprocal Rank ::0.5083
```

As it can be seen from the result above, the cosine similarity produced has been reduced a lot comparing to the outcome given by Term Frequency based results, which means that a lot of bad keywords that cannot be used to distinguish relevant and non-relevant documents and terms.

Since both using TF-IDF machanism and eliminating stopwords may incur overlapping performance, the MRR is not as good as expected. Some of the stopwords may increase the performance of calculating the cosine similarity, for example, if "a" and "the" are the common token among a set of query, the corresponding TF value indicate high correlation between these two document. Nevertheless, the real meaning between these two may far away from each other.

Reference:

Tf–idf. (2014, August 31). In Wikipedia, The Free Encyclopedia. Retrieved 02:10, October 23, 2014, from http://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=623618416