11-693 Software Methods for Biotechnology

Name : Carol Cheng
Andrew ID : carolche

Homework 3
Report

Error Analysis

Ranking of Relevant Documents	Query ID	Number
Rank 1	10	1
Rank 2	1, 2, 4, 6, 8, 9, 14, 18, 20	9
Rank 3	3, 5, 7, 12, 13, 15, 16, 17, 19	9
Rank 4	11	1

Mean Reciprocal Rank(MRR) = 0.4375

In 20 queries, only one relevant document was ranked as the first, whose Query ID is 10. The error analysis on other 19 queries is as followed:

Error Type	Query ID	Number
Vocabulary mismatch / Question words	1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19	15
Sentence length / Redundant information	1, 2, 15, 17, 18, 19	6
Similar words or phrases / Frequency sensitive	1, 2, 3, 5, 6, 9, 11, 12, 14, 15, 16, 17, 19, 20	14
Stemming/Lemmatisation	3, 4, 5, 7, 8, 11, 13, 16, 18, 19	10
Tokenization	1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 17, 19, 20	15
Case sensitive	5, 9, 15	3

• Vocabulary mismatch / Question words

Vocabulary mismatch happens when the vocabulary or expressions used by the query are totally different from the relevant documents, even if conceptually they may convey the same or similar meaning. This kind of errors is difficult to exclude. On the other hand, question words like which, when, where, and what will never appear in the relevant documents, but these question words decide what would be the answer. In this kind of situation, cosine similarity will not perform well since queries and relevant documents might not have the same words.

qid	Examples	qid	Examples
1	Q: destroyed A: burying	11	Q: Where A: can be found
2	Q: the largest crowd A: 62,046 fans	12	Q: the height of A: stands 267 1/2 feet tall

3	Q: which year A: 1867	13	Q: deep A: 1,932 feet, depth
4	Q: what year A: 1962	16	Q: when A: in 1981
6	Q: who A: Roger Bannister Q: run the mile in less than four minutes A: break the four-minute mile barrier	17	Q: which U.S. state A: Iowa Q: the leading corn producer A: the nation's largest producer
7	Q: what year A: 1959	18	Q: where A: San Bernardino, Calif. Q: the first McDonald's built A: From a single hamburger stand
8	Q: when A: on June 28, 1997	19	Q: took place A: burned and crashed
9	Q: on the moon A: to reach the surface of the Moon. Q: spaceship A: spacecraft		

• Sentence length / Redundant information

If the relevant document provides a lot of irrelevant information, it might make the con sine similarity lower by making the magnitude of the term vector too large. The case is particularly obvious when the length of the relevant documents is much longer than the length of the queries. For example, the following relevant documents are some examples which contains about half of redundant information. Redundant information means that the information in the documents is not the queries target for and the used words cannot be found in queries.

qid	Examples
1	In A.D. 79, long-dormant Mount Vesuvius erupted, burying in volcanic ash the Roman city of Pompeii; an estimated 20,000 people died.
2	When Michael Jordan--one of the greatest basketball player of all time--made what was expected to be his last trip to play in Atlanta last March, an NBA record 62,046 fans turned out to see him and the Bulls.
15	A new look at NASA satellite data revealed that Earth set a new record for coldest temperature recorded in East Antarctica.
17	Corn futures found support from forecasts for above-normal temperatures in major growing areas, including Iowa, which often is the nation's largest producer.
18	From a single hamburger stand in San Bernardino, Calif., in 1948, the systematized approach that the McDonald brothers developed to offer customers reasonably priced food at a rapid pace formed the cornerstone of the fast-food business.
19	On May 6, 1937, the hydrogen-filled German dirigible Hindenburg burned and crashed in Lakehurst, N.J., killing 35 of the 97 people on board and a Navy crewman on the ground.

• Similar words or phrases

This type of errors happens when irrelevant documents have similar words or phrases that result in higher cosine similarity than the relevant documents. The effect of similar words is especially obvious when the queries are short or when the irrelevant documents have tokens of high frequency. For example, stop words like “the”, “of”, “a” which are more likely to have high frequency in the bag of words will affect the cosine similarity hugely, especially when the frequency is higher than 2 or 3 in both queries and documents.

qid	Examples	qid	Examples
1	Q: the, destroyed, city, of 1: the, of, the, destroyed, city, of	12	Q: is, the, the, tallest 1: is, the, tallest
2	Q: Michael, Jordan, the, largest, crowd 1: Michael, Jordan, the, largest, crowd	14	Q: the, the, Commodores 1: Commodores, the, the
3	Q: a, purchase, of, Alaska 1: a, purchase, of, Alaska	15	Q: is, the, coldest, place 1: is, the, coldest, place 2: is, the, coldest, place
5	Q: river, is, called 1: river, is, called 2: is, river	16	Q: did, Bob, Marley 1: Bob, Marley, did 2: die, Bob, Marley
6	Q: to, run, the, mile, in, than, four, minutes 1: to, run, the, mile, than, in, four, minutes	17	Q: U.S., state, is, the, leading, corn 1: is, the, leading, corn 2: U.S., state, leading, corn
9	Q: was, the, first, the, moon 1: was, the first, the, the	19	Q: The, Hindenburg, disaster, took, place, in, 1937, in, which 1: The, Hindenburg, disaster, took, place 2: which, disaster, in, 1937
11	Q: is, Devil’s, Tower 1: Devil’s, Tower, is 2: is, Devil’s, Tower 3: Devil’s, Tower	20	Q: is, the, Keystone 1: Keystone, is, the, Keystone

*1:, 2: and 3: denote the irrelevant documents ranked as 1, 2 and 3 by cosine similarity.

• Stemming / Lemmatisation

There are some cases in which the same verbs with different tense or ‘China’ and ‘China’s’ were consider as different words and then influenced the calculation of cosine similarity. By using a stemmer or lemmatizer, we can reduce inflected words to their word stem. For example, ‘purchase’ and ‘purchased’ will be seen as the same word after processing by stemmer or lemmatizer.

qid	Examples
3	Q: purchase A: purchased
4	Q: score A: scored
5	Q: called A: call Q: China’s A: China

7	Q: become A: became
8	Q: bite A: bit
11	Q: Devil’s A: Devils
13	Q: deep A: depth
16	Q: die A: died
18	Q: McDonald’s A: McDonald
19	Q: New Jersey A: N.J.

• Tokenization

The simple tokenizer provided by the baseline system only tokenizes sentences by white-spaces. This results in a serious problem that punctuations were retained with the tokenized words and then it causes the wrong term frequency and biased cosine similarity. The case appear in 3/4 of the queries and documents and can be easily fixed by using a better tokenizer which can tokenize words by punctuations.

qid	Examples
1	erupted, / Pompeii; / Pompeii.
2	Jordan--one
4	points?
5	Sorrow? / China’s / “sorrow / China” / River,
6	four-minute
7	state? / state.
8	ear? / ear.
9	moon / Moon.
12	redwood? / redwood
13	Lake?
14	Commodores.
15	earth?
17	producer?
19	1937,
20	State? / State,

• Case sensitive

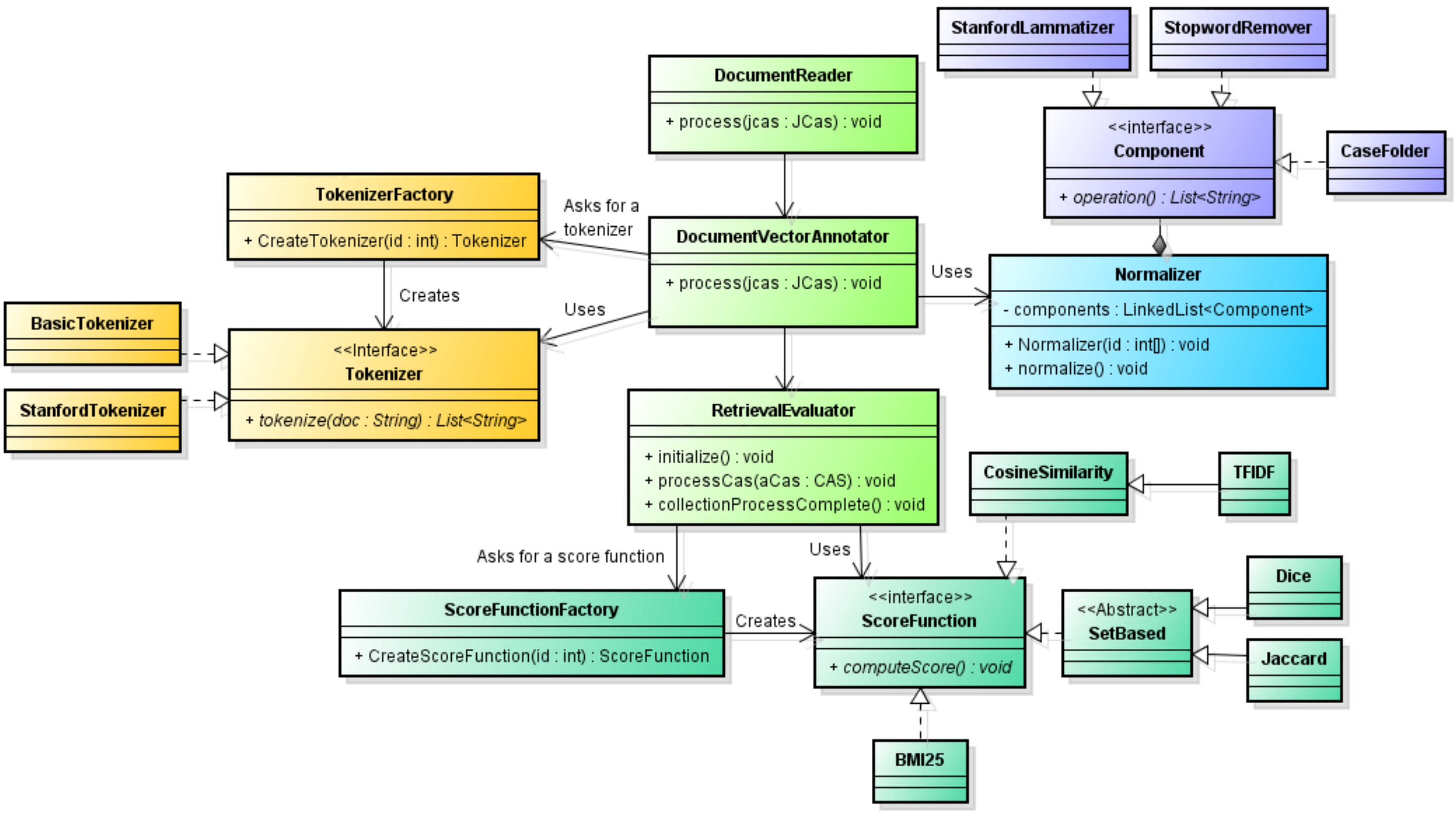
When calculating term frequency and cosine similarity, the bag of words are case sensitive, therefore, the same words with difference of case will be considered different. The type of errors did not appear quite often and can be solved by changing all of words from upper case to lower case.

qid	Examples
5	Sorrow / sorrow river / River
9	moon / Moon
15	earth / Earth

System Design

Based on the error analysis above, here are some possible solutions to solve the problems:

1) **Use a better tokenizer**
   The errors caused by the baseline tokenizer are quite obvious and can be easily fixed by using a better tokenizer that can do tokenization by white-spaces and punctuations like StanfordCoreNLP tokenizer.
2) **Add a stemmer or lemmatizer**
   The wrapper of StanfordCoreNLP lemmatizer has been provided in package util.StanfordLemmatizer.
3) **Remove stop words**
   This method can reduce the influence of stop words, like “the”, “of”, “a”. The archetype already provides a list of stop words.
4) **Do case-folding**
   Case-folding can be achieved by reducing all letters to lower case.
5) **Use better or different similarity measures**
   Some other similarity measures are like Dice coefficient, Jaccard coefficient, TF-IDF and BM25.



Given the solutions mentioned, I design a new architecture of the system. The new system can be presented by the UML diagram above.

1) Type System

In order to calculate BMI25, we need the length of the documents in words, therefore, I will add a new feature called **`length`** under Document type.

2) DocumentVectorAnnotator

The new system uses a creational design pattern — **simple factory pattern**. In order to keep the basic tokenizer and allow others using this system to add other tokenizers as they want, the DocumentVectorAnnotator will call **`TokenizerFactory`** with a integer argument, **`id`**, to create the specified tokenizer. The id will be specified by configuration parameter in the annotator descriptor. For example, **`StanfordTokenizer`**'s id is 1, then I can simply create a StanfordTokenizer by configure the parameter to 1 and then the TokenizerFactory will create StanfordTokenizer without changing code manually.

In addition, the new system will add some new components like stop words remover, case folder and include some external components like tokenizers, stemmers, lammatizers. I hope the system can always be configured whether to use these components or not. As a result, I add a **`Normalizer`** class to manage the components. Normalizer will be created by given an array of string as an argument — the argument is from configuration parameters. Given the argument, the constructor of Normalizer will add the specified components into a list. Then the method **`normalize()`** will run each components in order. For example, if you don't configure the parameter or configure it as 0, then Normalizer will not be used. If you configure it as "1 2 3", then **`StanfordLemmatier`**, **`StopwordRemover`**, and **`CaseFolder`** will process the list of tokens in order, which means you can also set up the order of text normalization. You can also add any component you want if you implement **`Component`** interface and specified the id for the component. Here I use **template method pattern**. For brevity, the template methods are not shown in this UML diagram.

3) RetrievalEvaluator

The ideal system should equipped with several functions to test the effect of different similarity coefficients or score functions, therefore, here I use the same design pattern, simple factory pattern. All of score functions included will implement **`ScoreFunction`** interface. RetrievalEvaluator can call the method **`CreateSorreFunction()`** with an **`id`** argument then the specified ScoreFunction object will be created. By giving a configuration parameter like 1 for cosine similarity and 2 for TFIDF weights cosine similarity and simply using the method **`computeScore()`** of all ScoreFunction object, you can use different score functions without changing a line of code. For reuse code, I also make **`TFIDF`** class extend from **`CosineSimilarity`** class and make **`Dice`** class and **`Jaccard`** class extend an abstract class, **`SetBased`**, since the way of calculation has something in common.