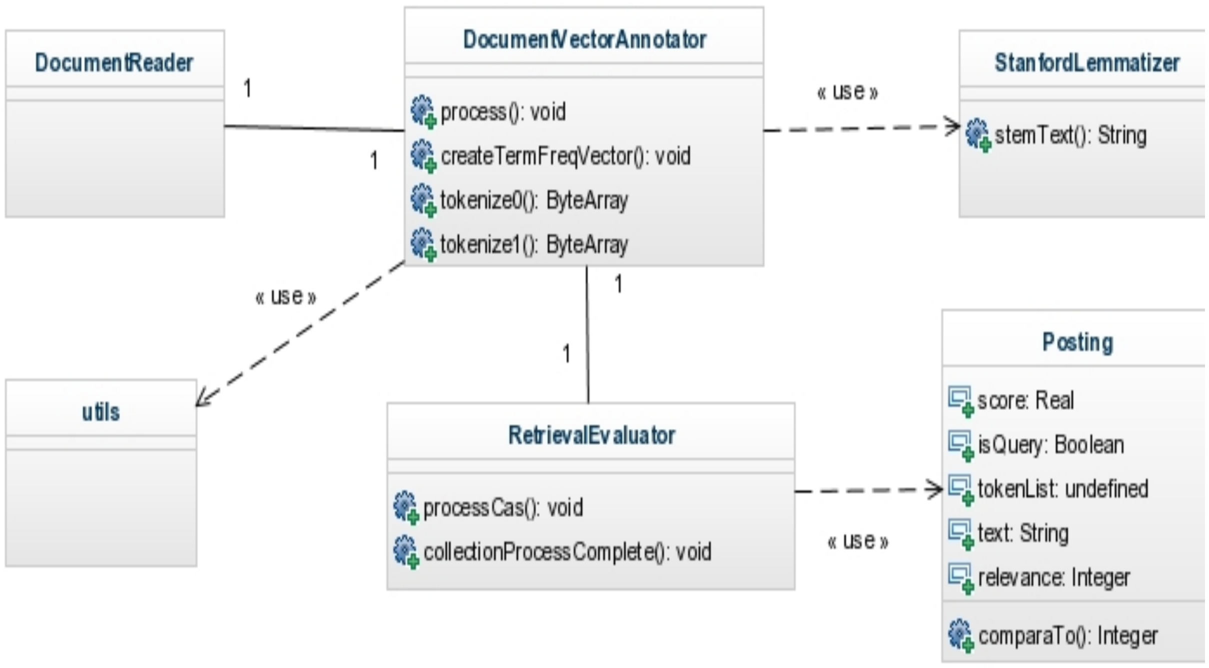
Homework 3 Report

Rui Wang (ruiw1)

1. System Design

1.1 Class Model



1.2 Type System

1.2.1 Document

Document is used to save query and document information, like query id, relevance. The most importance of Document is to save term frequency vector.

1.2.2 Token

Token is basic unit of term frequency vector, containing feature like term and term frequency. It is the basic part of token list in vector.

1.2.3 Query (not used)

This type was designed to save query, in order to distinguish with document themselves. However, during implementation, I found it is not necessary to use two kinds of types because it increases complexity of system. However, it is still a good practice to save query independently.

1.3 Annotator

There is only one annotator this time, named DocumentVectorAnnotator. This annotator is used to build term frequency vector.

2. Error Analysis

2.1 Error Type

2.1.1 Vocabulary Mismatch

This error mainly caused by lack or inappropriate stemming. For instance, in most context, feelings and feeling have the same meaning. However, without stemming, VSM will not consider this term's weight. Stemming could find the connection between Noun and Adjective, Singular and Plural, etc. in different vectors.

Here is some example in sample corpus:

a) Query: In which year did a purchase of Alaska happen?
   Document: Alaska was purchased from Russia in year 1867.

In this query, "purchase" is a noun, but in document, we get "purchased". So Stemming can eliminate this mismatch error.

b) Query: What year did Wilt Chamberlain score 100 points?
   Document: On March 2, 1962, Wilt Chamberlain scored a record 100 points in a game against the New York Knicks.

Without stemming, "score" and "scored" will mismatch.

Another possible reason may result in mismatch problem is about tokenization. Simple tokenization on white spaces cannot handle other symbols, e.g. comma, question mark. Stemming will fail on cases like "China's ". Stemming considers this is a Plural and return "China' ". Similar problem happens like "happen? ", "Pompeii; ". So more rules should be applied to tokenize.

2.1.2 Semantic Error

Due to variety of natural language, one concept can be expressed by different ways and different phrases and words. For example, in query "Who was the first person to run the mile in less than four minutes" uses "four minutes" to describe speed of this person. However, in relevant document, "four-minute" is used as a effective expression, but result in a query mismatch.

2.1.3 Length biases

Cosine Similarity can measure two vector's similarity, but the drawback of this method is it has length biases.

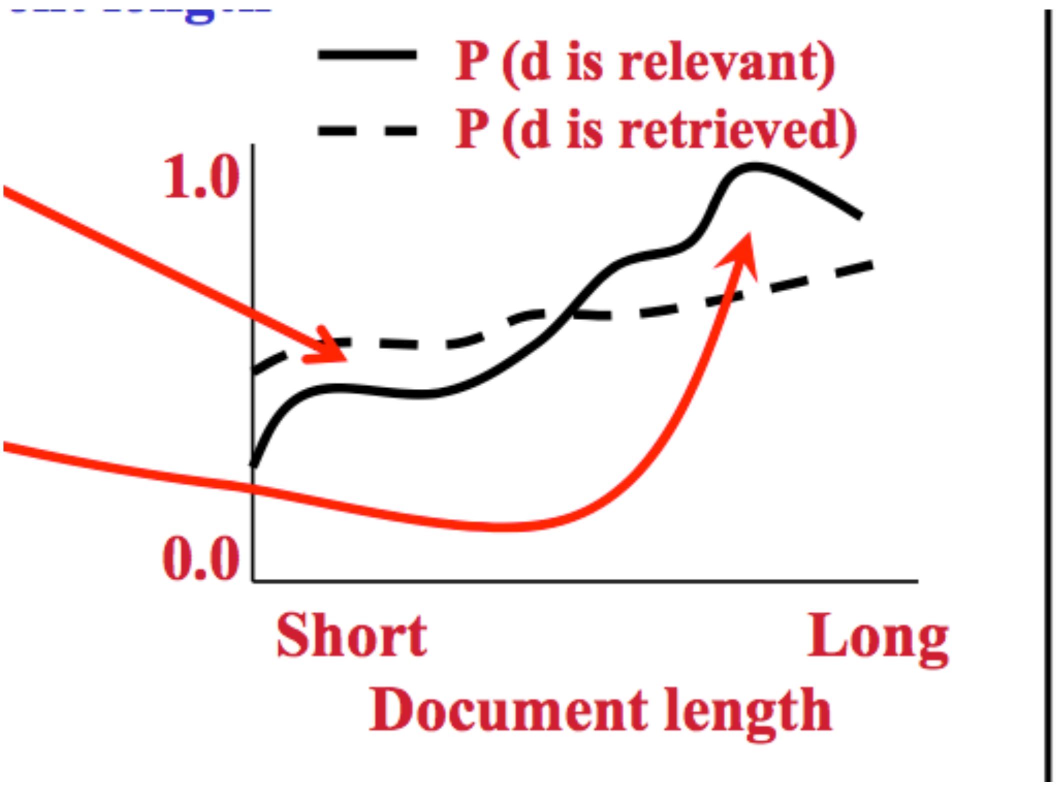


Figure 1: Length biases of cosine similairty
Source: VSM, page 10

Typically, it overestimate document with short length, and underestimate document with long length. So this may result in mismatch in some cases.

2.2 Solutions

2.2.1 Processing Punctuation
I write a new rule based tokenization algorithm, in order to remove unnecessary symbols. If a token is end with " ? ", " ; " , " ' ", regular expression I define will remove these symbols.

2.2.2 Stemming
To achieve stemming, I use Stanford Lemmatizer. It effectively transforms different forms of a word into its origin. The stemming package of Lucene is another choice.

2.2.3 Synonym
An approach to find synonym could be used to reduce semantic error. However, I didn't implement synonym algorithm because it is too complicated, and it is context dependent, which increases problem complexity.

2.2.4 Stopwords

A empirical rule is that the most frequent terms in vocabulary are terms like and, is, this, etc. If considering stopwords' importance in our retrieval model, maybe it would cost computational waste, because if every document contains more or less stopwords, it does not matter if I remove these words, and result will not be changed.

2.2.5 Reduce Length biases

To reduce negative effect of cosine similarity, formula of cosine similairty must be changed. For example, I can stop using common length normalization, but to use pivoted length normalization. For example, I can use (average # unique terms) / (# unique terms) as length normalization.

2.3 Performance Improvement

Solution	MRR Before utilization	MRR After utilization
Punctuation	0.4375	0.4708
Stemming	0.4375	0.5500
Punctuation&&Stemming	0.4375	0.5833
Stopwords	0.4375	0.4750
Punctuation&&Stopwords	0.4375	0.4750
Punctuation &&Stopwords &&Stemming	0.4375	0.5417

Combine different methods have different effect, but they do have positive effect on accuracy. Stemming greatly contributes to performance improvement. Punctuation processing and removing stopwords helps a little.

3. Similarity Measures

There are different kinds of similarity computation, so I implement two more measures, Dice coefficient and Jaccard coefficient.

3.1 Measurement

Measures	MRR
Cosine Similarity	0.5833
Dice coefficient	0.575
Jaccard coefficient	0.575

3.2