Homework3

Name: Qiankun Zhuang
Andrew ID: qzhuang

Task 1

System Design

Basically I followed the pipeline of the vector retrieval process.

The Document Reader identifies the query id and relevant scores. DocumentVectorAnnotator identifies the word and word frequency in each sentence. RetrievalEvaluator computes the similarity and performance metric of the collection.

For Document Reader I didn't add much code because it's already done. For DocumentVectorAnnotator, I used white-space based tokenize0 to split every sentence to construct a word vector and update the tokenList in the Document Type.

For the RetrievalEvaluator, first I used a HashMap to store the word vector and its corresponding frequencies. Then I used another HashMap to store the map between query ID and the HashMap that stores the query's word vector. For documents the way to store them is similar, the only difference is that for each query ID it is mapped to a linked list of HashMap, because there are multiple documents for each query. With these data structure, I can easily calculate similarity between sentences from different CASes. For the convenience of ranking, I used a class Triple to store the id, relevance and similarity of each document under a specific query. The id records the offset of the document under the query. Relevance records whether it is labelled as relevant. Similarity records the similarity with respect to a specific measurement between the document and the query. The Triple class implements the Comparable interface and I override the compare method so that we can use Collections.sort() in the way defined by myself. Finally I used another HashMap to store query ID and the rank of relevant document. With this I can calculate the MRR across all of the queries.

Task 2

Error Types Classification:

Error Types	Query ID	Total Number
Vocabulary Mismatch	5, 6, 7, 11, 12, 16, 19	7
Structure Mismatch	1, 13, 17, 18, 20	5
Scope Mismatch	2, 3, 4, 8, 9, 14, 15	7

Analysis and Improvement

Vocabulary mismatch is caused by when people use different forms of words to describe the same thing. To deal with the Vocabulary Mismatch problems, we can use better stemming algorithm.

Here I tried Stanford Lemmatizer from StanfordNLP package. This Leammatizer can change the plural form, past tense and many other variations of words to the original form. For example, is, are , am all means be but have different forms in different situations. If you compare them directly, they will be considered as different words. However, for most of the time they are actually the same thing. So when calculating the similarity, the Vocabulary Mismatch caused by different forms of the same word will be reduced.

I created a class in my project which is responsible for calling the lemmatizer from Stanford NLP library. With lemmatization, the MRR is improved to 0.5792 from 0.4333.
Besides, I change all the word into lowercase to decrease the Vocabulary Mismatch. This improves the MRR to 0.6458 from 0.6250.

Structure Mismatch happens when we compute the similarity between vectors, we never consider the order of the words in the vector, and that sometimes could lead to some mismatches. Different structures may be used to describe the same thing. To deal with the Structure Mismatch problem, we can try to remove stop words and apply better tokenization algorithm.

I store all the common stop words in a HashSet, and wrote a wrapper for queries on the existence of a given string. There's a drawback with the Stanford Lemmatizer that it will take punctuations as a single token. So I added some common punctuation to the stop words to remove them. This modification improves the MRR to 0.6000 from 0.5792.

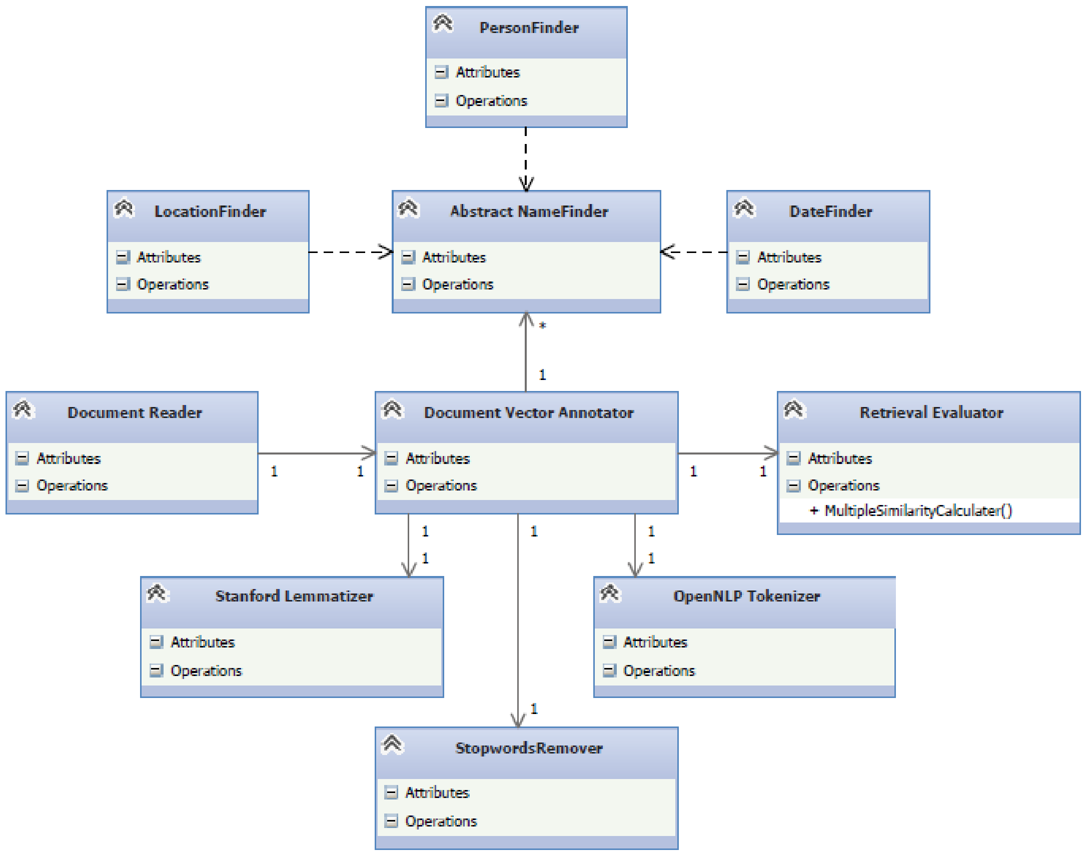
For a better tokenization performance, I used Apache OpenNLP Library. With a tokenizer model provided by OpenNLP, it has some improvement on the tokenization strategy. For example, "isn't" gets split into "is" and "n't", since it is a a brief format of "is not". This improve our MRR from 0.6000 to 0.6250.

Scope Mismatch happens when the query wants to ask about the location, a document containing date may be selected due many common words it shares with the query. However that's not what we should expect. To deal with the Scope Mismatch problems, we can try other similarity measures.

For similarity measures, besides cosine similarity, I tried Dice coefficient and Jaccard coefficient. However, their performances are worse than the cosine similarity, with MRR reduced from 0.6250 to 0.6075. The if-tdf measurement is much better than other methods, with a MRR = 0.8333. However, when I look into every query, there are lots of ties. It's because we always put relevant document at the higher rank when tie happens that the MRR gets so good. It won't be that good in practical use.

I come up with another idea to deal with the Scope Mismatch problems. For queries there are often some keywords like when, where, who, etc. I extracted these keywords from each query and records what aspects of information it's asking for. Then I used a Name Finder from OpenNLP to check each document to see if there are location names, person names or date names. The document containing the information that the query is asking for should be rewarded. For every match, I used sqrt(x) to improve the similarity between the document and the query. This method brings a very remarkable improvement on the performance. I only extracted when, where, who, and the MRR is increased to 0.7125. The accuracy for problems asking for when, where, who is very high. But I haven't thought of an idea to deal with what, which, how yet. But it should not be more difficult than this.

UML Diagram



PersonFinder
Attributes
Operations
LocationFinder
Attributes
Operations
Abstract NameFinder
Attributes
Operations
DateFinder
Attributes
Operations
*
1
Document Reader
Attributes
Operations
1
1
Document Vector Annotator
Attributes
Operations
1
1
Retrieval Evaluator
Attributes
Operations
+ MultipleSimilarityCalculater()
1
1
1
1
1
Stanford Lemmatizer
Attributes
Operations
OpenNLP Tokenizer
Attributes
Operations
1
StopwordsRemover
Attributes
Operations