COMP3009J Information Retrieval
Learning Journal
Student Name:
Student Number:

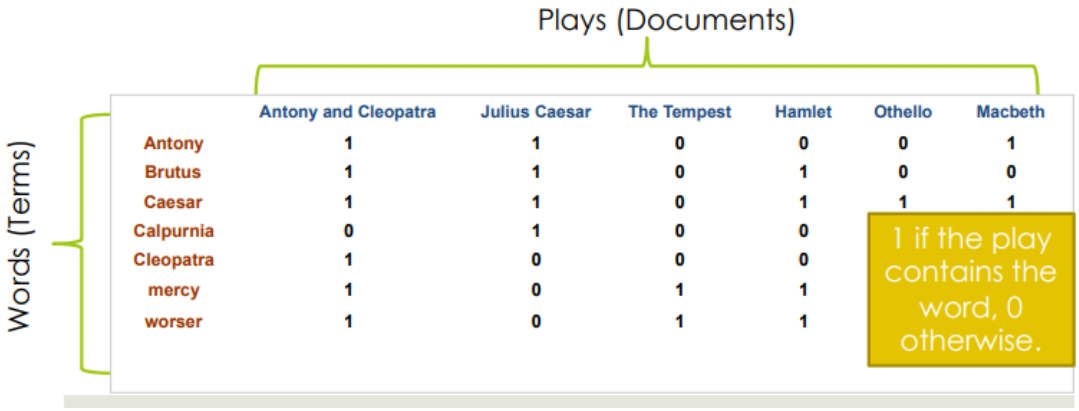
Week 2（beginning March 8th 2021)

Lecture Topic: Example and Indexing

My Learning:

The course has 3 parts in this week, IR example, indexing and query optimization.

As for the first part, it is meanly about Boolean queries which is a famous information retrieval example. The first step of Boolean queries is creating term-document incidence matrix by counting whether there is a corresponding term in a specific document. If the document contains the term, it is recorded as 1. If not, it is recorded as 0. In the following, it is the example of term-document incidence matrix.



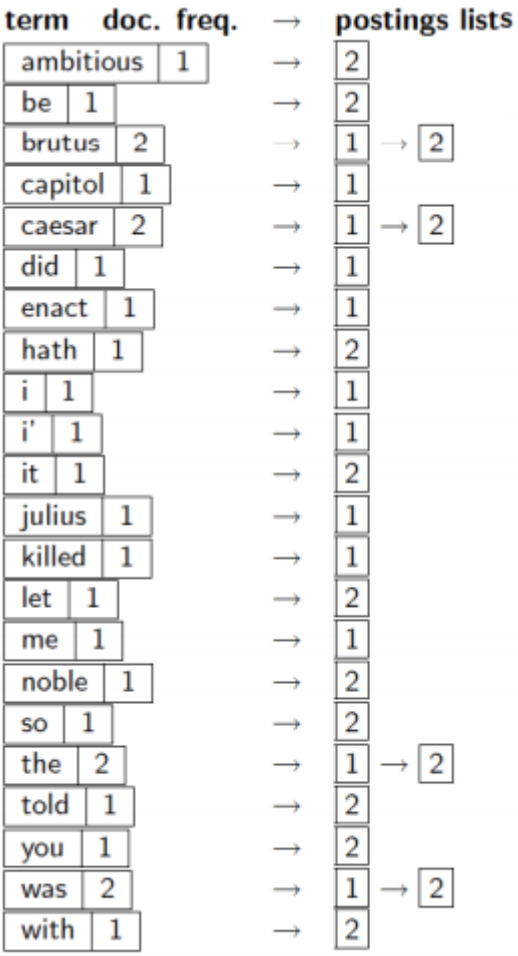
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0		
Cleopatra	1	0	0	0		
mercy	1	0	1	1		
worser	1	0	1	1		

The second step is forming the incidence vectors of each term, which is a string of numbers made up of ones and zeros. In addition, these numbers have the same meaning as the numbers in the matrix. In the previous example, the incidence vector of Antony is 110001. The final step is the Boolean style query operators. In this part, the researcher can get the combination of the incidence vector or the complement of the incidence vector according which terms they want to query. Then get the query answer by referring back to the term-document incidence matrix. In the previous example, we can get the combination vector of Antony and Caesar is 110000, and the answer is *Antony and Cleopatra* and *Julius Caesar*.

In the second part, some definitions about indexing were introduced. Indexing is the process of creating a data structure which was used to be a reference of query. In addition, Inverted index is a type of data structure where each term is mapped to a list of documents that contain this term. Inverted index needs two preliminary steps, splitting the text into tokens and processing the tokens. The first step of inverted index is token sequence which is creating a sequence list of pairs of token and document ID. The

second step is sorting the sequence by the terms (maybe alphabetically) and then by the document id. The final step is dictionary and posting. The entries which appear more than once in a document are merged, and the document frequency of each team also need be recorded. In the following, it is an example.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



The list also needs to be split into a dictionary (alphabetical order of the terms) and postings (ordered list of documents each term appears in). There are two steps about how do using the inverted index to process a query, locating the terms in the dictionary and retrieve its posting and merging the two postings of the document sets. The definitions of AND, OR and NOT are also mentioned. AND is used to narrow a search by ensuring that all the search terms should appear in the results. OR is used to broaden a search by retrieving any, some or all of the keywords used in the search statement. NOT is used to specifically exclude a term from your search.

The third part is query optimization. This part introduced the best order for the query processing with postings lists. If all of the interaction of terms are and, the smallest set should be calculated first, which is also the least frequency document. If OR appears in the increasing list, estimating the size of each OR operation is needed to be implement before AND operation by getting the sum of the frequencies of the terms, then it can be recognized as an increasing list. There is also a more optimized data structure which is named Skip Pointers. This method needs not iterating through every element, because of

the pointer / reference to the next node of a node. This is called skip element, and the smaller skip element needs be implemented first. This method can ignore the distribution of query terms and it is easy for the static index.

Lab/assignment work:

There is no lab or assignment this work. However, there are some exercises in the slides.

The answer of the exercise in the second slides is a little difficult. In my opinion, *Brutus* AND NOT *Caesar* can adapt the merge. In this situation, NOT Caesar is equal to the complement of Caesar, and then it is the same as AND operation. In other situations, it cannot adapt the merge straightly. Because A or B is not equal to A, B and A&B, they have overlapping part. So, the answer requires an extra step to get rid of the duplicates.

The answer of the exercise in the third slides is that *kaleidoscope* OR *eyes*, because the sum of the frequencies of these terms is the smallest. By this exercise, I master the basic optimization of query processing.

Reflections:

In this week, All of I learned are based on the Boolean queries. It is a very interesting method, so I think it is very useful. By reading the literature, I found it can be used as a source of phrases for a statistical retrieval model [1]. In the following, it is an example of basic Inference network. It is very similar to what we learned this week.

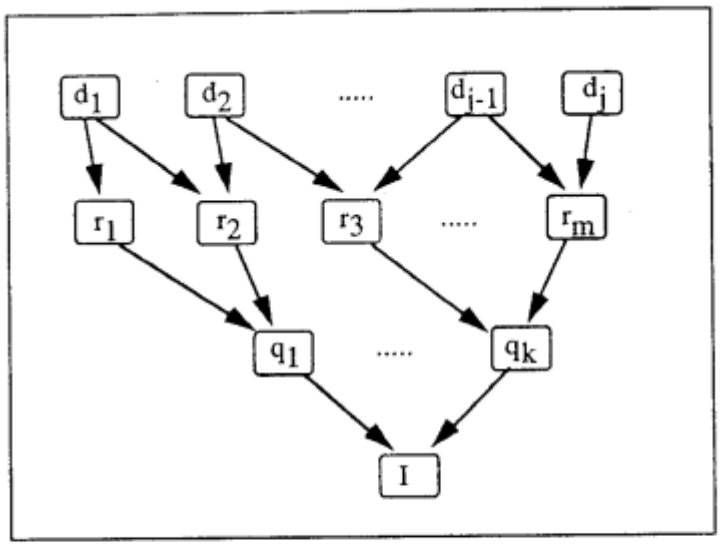
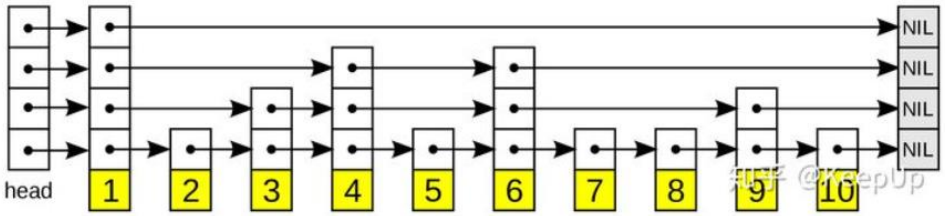


Figure 1: Basic Inference Network: d_i's are document nodes, r_i's are concept nodes, q_i's are query nodes, and I represents the user's information need.

However, it is not perfect, Turtle mentioned natural language is better than Boolean queries evaluation when dealing with the large documents in 1994 [2], since Natural languages are more scalable than Boolean queries.

As for the indexing, I was confused when I learned this part at the first time. Because this section is more about inverse indexing. I don not know the relationship between the them. Cui mentioned that the essence of the index is the characteristics of the information in his blog [3]. Both the indexing and inverse indexing are the structure which shows the relationship between information (document) and characteristics (terms). Indexing is the relationship between each information and some characteristics about it. In contrast, Inverted indexing shows the relationship between each characteristic and some information about it. Both of them are the tools for query, but inverted indexing is more useful, because the characteristics is the thing what we want to find. So, the most of the lecture is about the inverted indexing.

As for the third part, it mentioned an optimized data structure, skip pointers. In the lesson, a lot of advantages of skip pointers was mentioned, simple, easy and useful. At first, I did not know how the node knows its next node. Now I know it is one attribute of a node by the discussions on BrightSpace. Skip pointers is included by skip list which is an alternative to balanced trees [4]. Below is a schematic diagram of the jump meter which is more complex than what we learned. The lookup proceeds from the most sparse subsequence at the top until the element to be looked for is between two adjacent elements in the layer.



All in all, I think this week is the basic course. So, it is very important to understand the content of the course so that we can master the knowledge that I will learn in the future. This week' lesson is all based on Boolean queries, which is only one part of IR. Natural language processing is also a vital part of IR, it and Boolean queries have different advantages. We should choose the right method in the right situation.

Week 3（beginning March 15th 2021)

Lecture Topic: Preprocessing

My Learning:

This course has 2 parts, the problem and the solution of Tokenisation. Tokenisation is the first step of preprocessing of indexing which is process of turning what the user inputs into something that the computer can search for directly.

As for the problem of Tokenisation, there are 4 kinds of them. The first kind is about superfluous and meaningless content, and they often appear in the middle of searches. Therefore, this may result in failure to find the correct word when identifying, such as special symbols and Stopword. Special symbols are the symbols which also appears in one word, such as '-' and '.' . A stopword is a commonly occurring term that appears in so many documents that it does not add to the meaning of the document, and appear in almost texts, such as 'the' and 'of'. The second part is about how to identity the words which means the computer cannot extract form a sentence what we want to query easily because of the language issue or the phrase. Some languages are written backwards such as Arabic, some languages do not have distinct lexical separators such as Chinese, and some languages has multiple alphabets intermingled in a sentence such as Japanese. The third problem is about synonyms and homonyms. The homonyms may cause redundant results in one search, the synonyms can cause that the right result cannot be found. A synonym may be two completely unrelated words or words with the same root and different parts of speech. The final problem is spelling mistakes.

This lecture mentioned four kinds of solution about these questions. The first solution is normalization. This method can solve the problem about the special symbols. Because some symbols can be removed. For instance, 'U.S.A' can be changed to 'usa'. It also can solve some language issues. For example, all letters in English can be changed to lowercase, all kanji and katakana can be changed to hiragana for Japanese. The second method is Thesauri and soundex. Thesauri is useful to deal with the synonyms. When the document contains one word, index it under all of its synonyms. Another way is to expand the content of query. When the query contains a word, look under its synonyms as well. Soundex can solve the spelling mistakes. It forms equivalence classes of words based on phonetic heuristics. That means it indexes terms using their sounds rather than their spelling. The third way is Stopword Removal which is removing the frequent and useless content in the text, such as 'of'. Zipf's law can help people to identity this kind of content. The method is simple but limited. These words may be meaningful in some situation. For instance, if the words are in the title, it can not be removed. So, it is important to distinguish when the content make sense and it cannot be removed. The last kind of solution is Stemming and Lemmatisation. Both of these methods are used to solve the problem of incomplete search for words with the same root. Stemming is very useful, but it cannot deal with the problems which are about homographs and similar but not related

words. Because it just turns all the words that have the same root into the common root directly by some algorithms such as Porter's Stemming Algorithm. Lemmatisation is more effective and accurate, but it is slower. Because it is a Natural language processing technique for converting words into lemmas which requites more text analysis than stemming. The appropriate method can be selected according to the different requirements. When dealing with the long and simple text, Stemming is better. Otherwise, Lemmatisation is better.

Lab/assignment work:

I am not familiar with Python. To finish this assignment, I learned a lot of knowledge on Runoob website [5] to help me read the file and so on. I only finished Q1, Q2 and Q3 of the assignment by myself. Because I try to use array and map to solve Q4 and Q5, it is complex and it is not the same as Java. By learning the solution of these questions and the information on Runoob website. Following is what I have learned about Python in this week.

1. Using with open ('A ',' B') as f can help me open the file without closing. 'A' is the name of the file, 'B' is the type of reading and writing. The following is what the specific type means.

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open for updating (reading and writing)

2. Strip method is use to remove characters (default space or newline) or sequences specified at the beginning and end of a string. In this assignment, it can make the output more concise and beautiful.
3. The dictionary can used to store some pairs of key and value. The form of it is d= {key1 : value1 , key2 : value 2}. It is very useful, because it is easy to add, search and sort the value by the key. There are two characters of the key. One is that the key can be repeated. In this situation, the last key will be the right key. The other is that the key cannot be changed, so the list cannot be the key. I think the dictionary is very similar to the map in JAVA.
4. The difference of sort method and sorted method is that sort is the method of list, whereas sorted can work with any iterable object, such as the dictionary.

The lab in this week helps me implement some operations in inverted indexing, such as

normalization and get the frequency. This assignment is mainly to let me experience how to use Python to implement information retrieval and master the use of dictionary.

Reflections:

As for the lesson in this week, almost problem can be solved by the method which this lesson mentioned. The question about phrase may be solved in next week. But how to deal with the information in which there is no distinct lexical separators may not be mentioned. I posted this question on BrightSpace, and read some information which you recommended. I learned a very useful way to address this problem. Jieba is a very useful segmentation to identity the words in a sentence [6]. It builds a directed acyclic graph for all possible word combinations [7]. Dynamic programming is used to find the most probable combination based on the word frequency. In addition, an HMM-based model can deal with the unknown words. Jieba can help people to identity the Chinese words by these algorithms. In my opinion, Tokenization is about linguistics. Information is presented differently depending on the language, which caused that we need to use different ways to parse the information so that that computer can understand it.

I also learned more information about Python. My Python development experience is very limited. I develop some project by Java. So, I was wondering why do I use Python for information retrieval and not Java. Almost successful search engines in the world are implemented through the web. Both of the Python and Java are good way to implement the web. However, there are more libraries of algorithms in Python, because python is a scripting language. What's more, I came into contact with Flask last semester. These may be the reasons why we use python to implement information retrieval.

All in all, the lecture is very useful this week. I master how to do Tokenisation in almost situation and how to write code using Python.

Week 4（beginning March 22nd 2021)

Lecture Topic: Phrase Queries and Vector Space (part 1)

My Learning:

The content of the course has three parts in this week, phrase queries, the introduction of vector space model and the cosine similarity of vector space model.

As for the phrase queries, there are two ways. The first one is Biword indexes, which means that the computers index every consecutive pair of terms in the text as a phrase. This way is not suitable for the longer phrase queries. It usually is a part of a compound strategy. The second way is positional indexes, which means the postings also need store for each term the positions in which it appears in the documents. This way is very large, but it is very useful. Both of these ways may be used together.

As for the second part, it introduces what vector space model is. In this model， both documents and queries are in an N-dimensional space. It can be used for Machine learning and clustering. It also can solve the problem that Boolean model has no ranking occurs. In this model, each term is represented by a dimension in the vector space, if a document contains this term, it was represented by a number which is known as a term weight, or it was represented by 0. In addition, a similarity score is used to rank the list which is calculated based on the proximity of two vectors in the N-dimensional space.

As for the formulae of cosine similarity, it was be shown in the following picture.

Cosine Similarity Formulae

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

- $\vec{d_j}$ is a vector representing a document d_j
- $\vec{q}$ is a vector representing the query q
- $\vec{d_j} \cdot \vec{q}$ is the dot product of $\vec{d_j}$ and $\vec{q}$
- $|\vec{d_j}|$ is the length of $\vec{d_j}$
- $|\vec{q}|$ is the length of $\vec{q}$

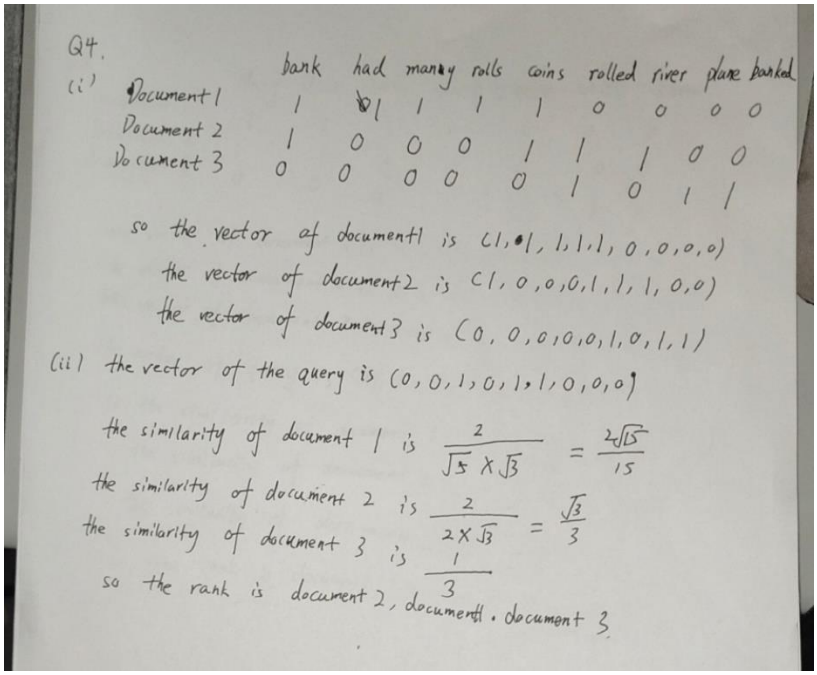
In addition, these two formulae can be useful to calculate the cosine similarity.

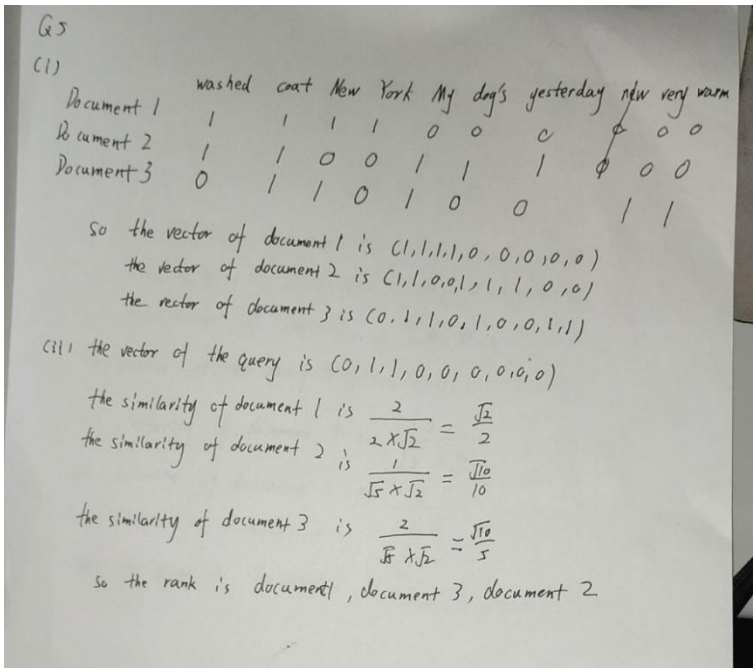
- $\vec{a}.\vec{b} = \sum_{i=1}^{T}(w_{i,a} \times w_{i,b})$
- $|\vec{a}| = \sqrt{\sum_{i=1}^{T} w_{i,a}^2}$

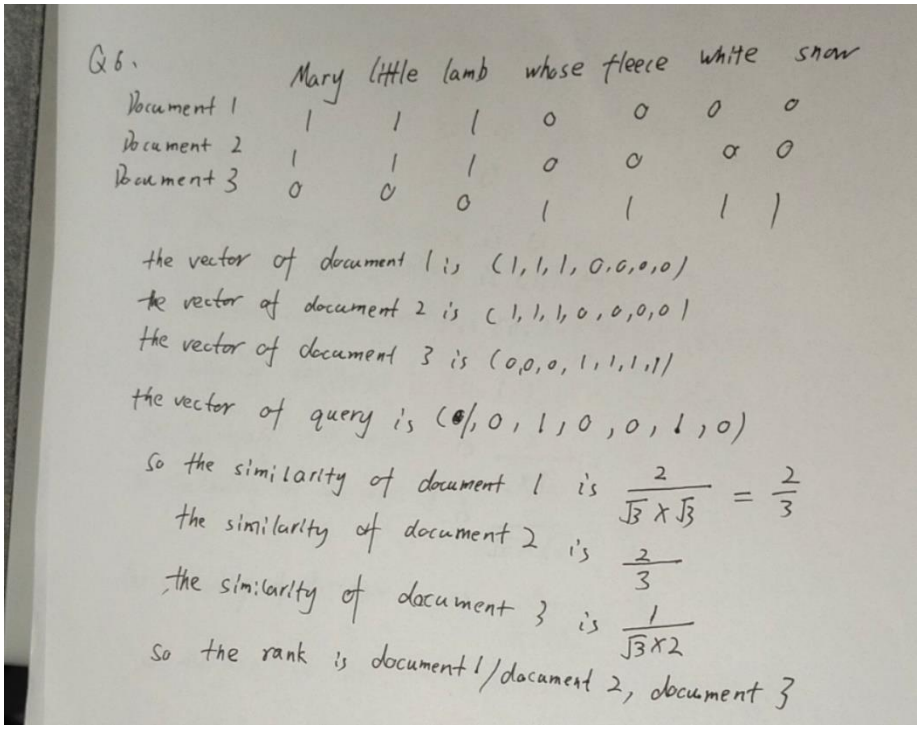
Lab/assignment work:

This week has an exercise and a lab.

As for the exercise, it is divided into 2 parts, phrase queries and vector space model. The answers of the first part are d1 and d2 \ d1 and d4\ d1 and d4. The solution of this kind of question has 2 steps. The first 1 is calculating the value of the subtraction between terms, and then find the document whose next positions also can subtract to get that value. The second step is observing the position which is near that pair of positions that was found in step 1. If that position is not the term in the phrase, that document can not the answer. In the following, it is the answers of the part 2.







As for the solutions of this kind of questions, there are three steps. The first step is listing the terms, the second step is getting the vector of each document, and the third step is calculating the similarity of each document and get the ranked list. This kind of questions is very simple, but it may be not accurate, because the weighting scheme is too simple.

As for the lab of this week, it is more complex than last week, and it is about the question which was mentioned in week1. By writing code to implement the task of this week, I found I was wrong in the first week, because I was just thinking about it theoretically on the first week, but now I am thinking about it at the code level and I realize that the view of the first week is not true. The data structure I want to choose is the dictionary, because the key can be the term, I can search it by terms. And I use list to store the docID of each term, because I can find the term according to the location of it, and it also can save the rank by the index of array. During writing the code, I find the order of the list is really important. Because if the id in one list is likely to be equal to another id in other list, it must appear before the ID changed from the smaller to the larger. Thanks to this point, I do not have to worry about removing duplicates when implementing mergeOR, because I can avoid duplicates when I add them. I learned the following knowledge about Python due to this lab.

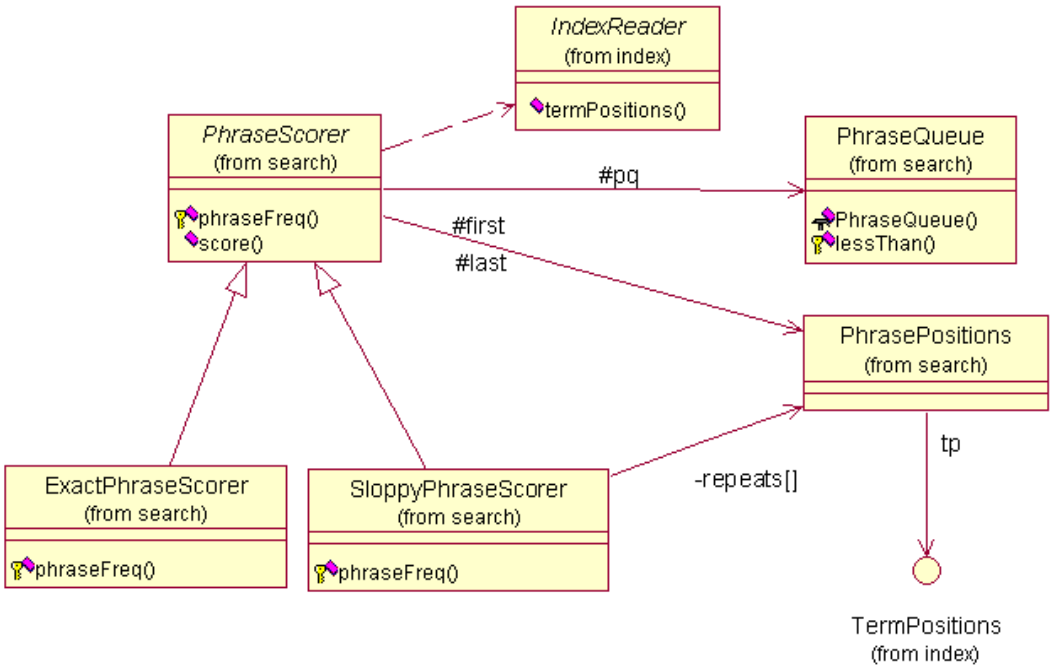
1. The int() function is used to convert a string or number to an integer. It is easier than Java.
2. In python 3.x, The input(prompt) function takes a standard input data and returns it as a string. Prompt is the same as what you print out to let the user know what they should input. It is a very convenient way to get the input of the user.
3. List is a date structure which is very similar to the array in Java. They also have a lot of differences, such as list do not have a specific number of elements at the beginning. The append（） method is used to add thing to the list.

4. The join() method is used to add new string to the original string, it is very easy to use.
5. In python, () is the tuple data structure, [] is the list, { } is the dictionary.

Reflections:

As for the Phrase Queries, I think the positional indexes is a very good way to research the phrase, but it is too big.

There is an open source full-text search engine toolkit in the Java development environment called Lucene. Lucene is a high-performance, scalable information search library. It adds indexing and search capabilities to your application, which is a full-fledged open source project implemented in Java, part of the famous Apache Jakarta family, and under the Apache Software License. Also, Lucene is a very popular, free Java Information Search library. In Lucene, phrase query was implemented. In the following, it is a UML diagram about how to implement it [8].



It is similar to the positional indexes, and it can limit the distance between two terms in the result. It is a good example to implement phrase queries.

Boolean queries have no ranking, so it is difficult to identity which document is more important. This affects the efficiency of our retrieval. But the vector space model can solve this problem [9]. By the exercise, I found if the weighting scheme is too simple, which just contains 0 and 1, the order of the document is not very accurate. In Q6 of the exercise, the document1 and document2 are different, but their ranks are the same one. But the times of the term appear in these documents are different. So, there may be more complex scheme to calculate the weighting.

Week 5（beginning March 29th 2021)

Lecture Topic: Vector Space Model (part 2)

My Learning:

The course has two parts in this week. The first part is about the term weighting and the example, the second part is the conclusion of vector space model.

As for the term weighting, the course introduces a very useful metric, TF-IDF. It can be used to sort the list more scientifically. TF means Term Frequency. It assumes that a term that occurs many times in a document is more important in describing that document than one that occurs rarely. Each term in term document has a value of TF. TF is equal to that the number of the times the term occurs in a document divided by the maximum number of times any term occurs in that document. IDF means document frequency, which thinks that only occur in very few documents are more important than those that are found in many documents. IDF is calculated using the following formula.

- $idf_i = log_2\left(\frac{N}{n_i}\right)$
- N is the total number of documents in the collection
- n_i is the number of documents in the collection that contain term i.

The weight for a term I in a document j is equal to TF i,j * IDF i. There is one example about how to calculate the cosine similarity with TF-IDF. The solution will be concluded in Lab/assignment work part.

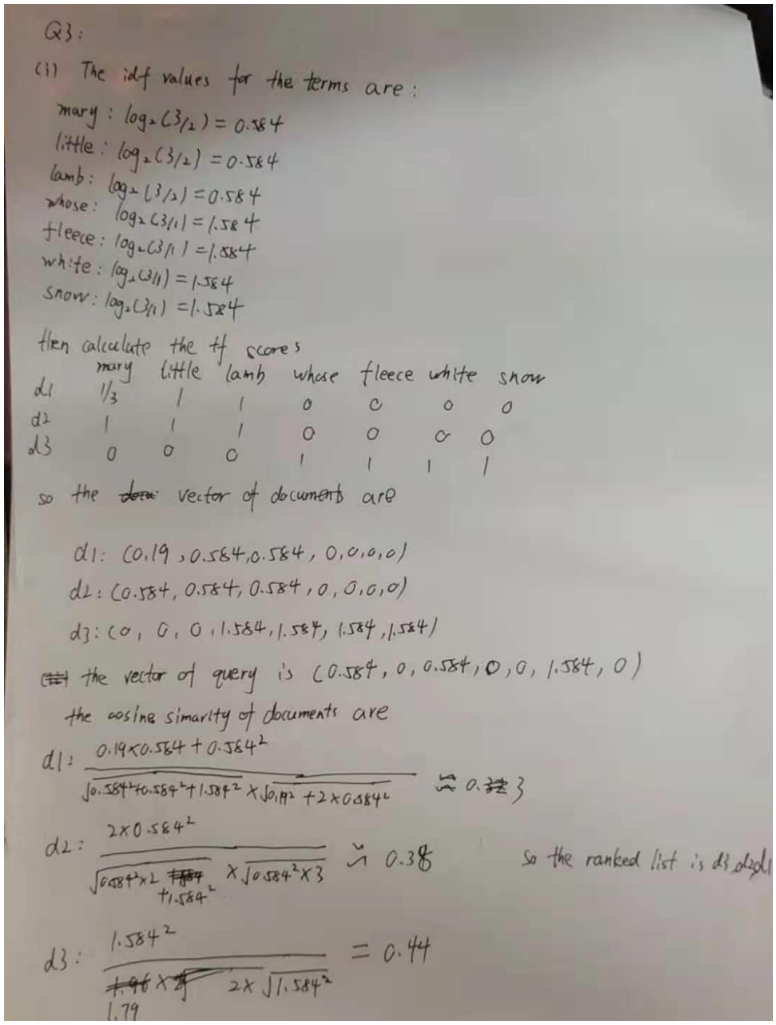
As for the conclusion part, it is mainly about the advantages and disadvantages of vector space model. There are three main advantages of the vector space model. As for the first one, the term weighting schemes can improve retrieval performance. The second one is that partial matching strategy allows for retrieval of documents that match part of the query. Lastly, the cosine similarity can be used to return a ranked list of documents. However, the terms are assumed to be independent in this model, which is bad for the performance.

Lab/assignment work:

This week has an exercise and a lab.

As for the exercise, it only has one kind of question, and it is similar to the example of the course. This kind of question has an easy way to deal with. The first step is calculating IDF values of all terms according to the formula. Afterwards, we need to calculate IF scores for all the terms in each document, the results can be tabulated. Thirdly, calculating IF * IDF, and get the vectors of all documents. Fourthly, get the vector of the query by the same

way. Finally, calculating the cosine similarity of each document, and get the order. In the following, it is the answer of Q3 in the exercise, the solution of other question is very similar, so it can be another example of solving this kind of question.



As for the lab, the coding is not very difficult. Understanding how skip list works is more difficult. Skip list can reduce the number of comparisons between two lists, thereby improving the efficiency of the merging. When the element a which is in list 1 is less than the element b in list 2, it needs to compare the skip of a and b. If the skip of a is bigger than b, b should compare to the next node of a. If the skip of a is smaller than b, b should compare to the skip of the skip of a. When the elements are equal, it can be added to the answer list. After measuring how long it takes skip list to merge the lists, compared to normal list. It saves a lot of times.

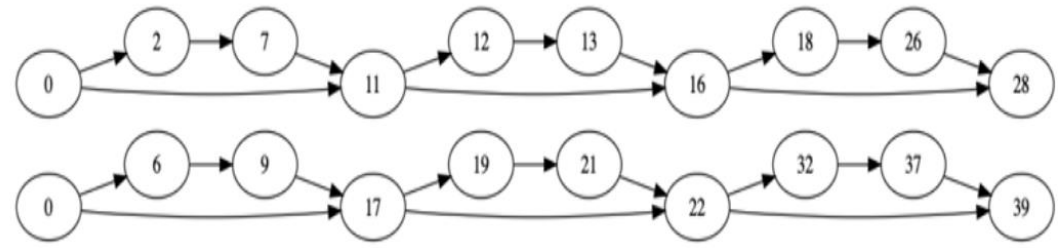
As for the advanced question, Q6 is very interesting. At the beginning, I thought the frequency is linearly related to time spent. The time taken by the merging fluctuates as I increase or decrease the frequency value. The time taken is lowest, when the frequency is equal to the square root of the number of elements in the list. At this situation, the time complexity is equal to the double square root of the number of elements in the list.

Supposing that the length of one list is M, the length of another list is N, the frequency is FRE. The time complexity is equal to (FRE+M/FRE)+(FRE+N/FRE), if M is bigger than N, the result is FRE+M/FRE, or the result is FRE+N/FRE. Because the list only needs to be iterated once, FRE+M/FRE is the time complexity of find the elements in one loop. The minimum value of FRE+M/FRE is equal to 2 √ FRE*M/FRE which is 2 √ M according to perfect square formula. Consequently, the minimum of time complexity is double square root of the number of elements in the longer list.

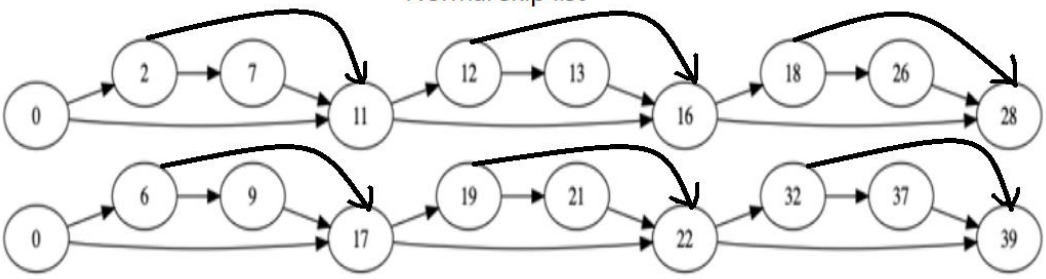
Reflections:

I think writing Lab can really deepen the understanding of knowledge.

As for the skip list, I think there may be another kind of skip list. In the normal skip list, there are a few nodes have the skip nodes. If the frequency is high, there are a lot of common nodes between two skip nodes. Once comparing these common nodes, the system must finish all common nodes because they do not have skip nodes. If the element in the second common node is more than the element in the other list, the rest of common nodes need not be compared to that element. So, I think each node has a link to the skip node can solve this problem, and the skip nodes are the same as the normal skip list. Once finishing the necessary comparison, it can skip to the skip nodes. In the following, there is a graph to compare these two kinds of skip list.



Normal skip list



New skip list

Although the new skip list may decrease time complexity, the room complexity must be increased definitely. So, the comprehensive performance of this kind of skip list is not very good after testing, so it is not promoted now.

There are a lot of ways of term weighting besides TF-IDF. But TF-IDF is a very basic one, and it has many varieties. In some situation, TF means the number of times a term occurs in a document which is different from the TF I have learned [10]. To avoid the impact of long documents, the logarithm of IF (log (1+TF)) needs to be calculated. There is also a interesting factor named probabilistic IDF which was derived from the probabilistic model and has an IDF form. It can be used to calculate term weighting sometimes. So, both of if and idf have various variants. The variants of if are log(tf), log(tf+1) or log(tf) +1. The number 1 in the formula is used for smooth calculation. IDF factor usually is computed as log(N/ni)) with a number of variants, such as log(N/ni + 1), log(N/ni) + 1 and log(N/ni − 1) (i.e.idf prob).

Week 6（beginning April 5th 2021)

Lecture Topic: Probabilistic Model and BM25

My Learning:

The content of this course has two parts, probabilistic model and BM25.

Probabilistic model is one of classic model. It is not the same as Boolean queries or Vector Space models. This model needs the feedback of the user. At first, the machine will produce an initial guess and send the reasonable initial set to the user. Then the user interacts with the system and describe what the ideal answer set should be by marking which ones are relevant. Repeating this interaction over and over again, almost all relevant documents are marked. At that time, the ideal answer set (the set of documents that are relevant to the user's query) is the same as the set of documents that the user has verified as being relevant.

The probabilistic model also has a similarity score which is used to rank the list. The formula of this is complex, which is in the following.

- This gives the following formula:

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)$$

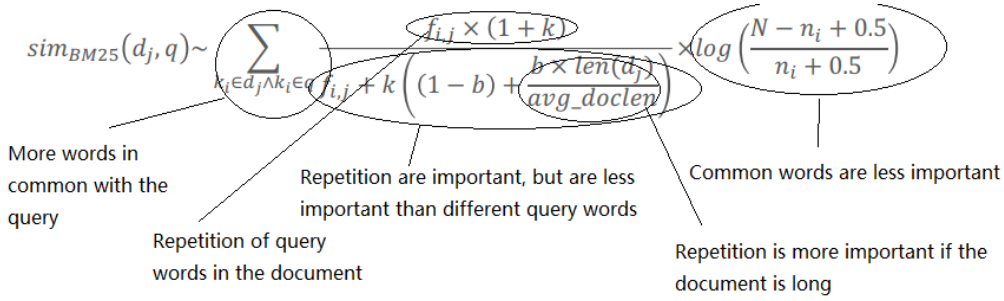
- To calculate this we need:
  - t: Total number of terms in the collection (very easy)
  - $w_{i,q}$: Weight of term k_i in query q (easy)
  - $w_{i,j}$: Weight of term k_i in document d_j (easy)
  - $P(k_i|R)$: Probability that a relevant document contains k_i (difficult)
  - $P(k_i|\bar{R})$: Probability that a non-relevant document contains k_i (difficult)

As for two probabilities, they need to be estimated. Assuming that all terms in the query have the same probability of being contained in relevant documents, which means probability that a relevant document contains ki can be estimated to 0.5. In general, the set of non-relevant documents will be very similar to the set of all documents. So, the probability that a non-relevant document is equal to the total number of documents divided by the total number of documents in the collection. After the interaction of user, the probabilities can be improved, and the formula of improving probabilities is in the following.

- $P(k_i|R) = \frac{V_i + \frac{n_i}{N}}{V+1}$
- $P(k_i|\bar{R}) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}$

Vi is the number of documents we know to be relevant that contains term ki. There is another kind of probabilistic model which is called automatic probabilistic retrieval. It avoid the interaction with user by assuming the top r documents are relevant.

BM25 is an advanced retrieval method that operates using the same principles as TF-IDF with better performance.



N is the total number of documents in the collection. ni is the total number of documents in the collection that contain term ki. fi,j is the frequency of ki in dj (i.e. the number of times the term appears in the document. k and b are constants that can be set to suit the document collection and the desired behaviour. For general collections. k=1 and b=0.75 have been found to work well. Len(dj) is the length of dj (i.e. the number of terms in it). avg_doclen is the average length of a document in the collection.

Lab/assignment work:

The lab of this week is about file parsing and term frequencies. The most difficult thing of this lab is the choice of data structure.

As for the structure of storing stopwords, it depends how the stopwords are use. Stopwords are used to judge if any term is in the list of stopwords. That means each stopword need to be visited in order. So, list and set can be chosen. What's more, the stopwords need not be changed, and initialization needs to be fast. Consequently, set is a suitable data structure.

Terms needs to be traversed sequentially, and it is created by reading files. So, list is a suitable data structure.

The structure of storing term frequency has better be dictionary. Both of the name of the term and the frequency of the term need to be stored, and we need to search the frequency according to the name of the term. So, the name of the terms can be used as keys. Using dictionary is a very suitable way to store the information.

As for the structure of storing the dictionary which was mentioned before and the

document id, both of list and dictionary is ok. Both of the document id and another dictionary need to be stored, and the user need to search the information according to the document id. So, the document id can be the key of dictionary. In addition, the document id is integer and in order. So, document id also can be the index of list. Consequently, both of the dictionary and list is suitable to store information. By testing, the process time using these two kinds of data structure is similar.

```
Time of dictionary is 0.65625 seconds
Time of list is 0.6875 seconds
```

By testing, calculating the frequency of each term is the operation which take the most time, because it is a multiple loop and stem method waste a lot of time. Whether it is necessary or not, the stem method will deal with it. So, the way to optimize programs is that avoid useless stemming over and over again. We can create a dictionary to store the stemmed terms to avoid useless stemming. It is very useful. In the following, it is the time of both two situations.

```
Time with all stemming is 4.625 seconds
Time with avoiding useless stemming is 0.859375 seconds
```

Reflections:

This week, I have learned the conception of probabilistic model, BM25 model and their formula. I think the concept is not very difficult, but the formula is very complex. The reason is that these formulas are derived from the knowledge of probability theory, such as Naive Bayes.

The BM25 model is an improvement of the BIM (Binary independent model). There are two assumptions about BIM. The first one is binary assumption which means the document can be shown in two situations, relevant and non-relevant. The second assumption is lexical independence assumptions which means the terms are independent. According to some more complicated calculations, the following formula can be got, and it is the same as the IDF vector in the vector space model. The performance is not good, but it is the basic of BM25.

$$\sum_{i:q_i=d_i=1} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/((N - R) - (n_i - r_i) + 0.5)}$$

http://blog.csdn.net/wdxin1322

BM25 focuses on the weight of the word item itself and the length of the document, which is why the performance of BM25 is better than BIM.

BM25F is an improved algorithm of typical BM25 which considers documents as a whole when calculating correlation. However, with the development of search technology, documents are gradually replaced by structured data, and each document will be divided into multiple independent fields. For example, web can be divided in to head field, content field, theme field. The weights of these fields are not the equal. The weight of head field is more important the weight of content field. BM25F is a weighted sum of the score values of each word in each field.

There are some other improvements of BM25, such as BM25L, BM25+, BM25-adpt, BM25T and so on. BM25L and BM25+ can deal with one issue of BM25 that is very short documents would be given too high scores. BM25-adpt and BM25T can deal with the issues about global term applied to all query terms and term-specific.

The course mentioned 3 principles of good term weighting, inverse document frequency, term frequency and document length normalization. Almost complex model is created to meet these three principles. However, I think how to divide the document and term is also a very interesting question. Because the term are not independent in fact, and the weight of document or the part of document is not equal.

Week 7（beginning April 12nd 2021)

Lecture Topic: Evaluation(part1)

My Learning:

There are many measurable quantities of evaluating the IR system, such as processing, user experience and search. Search means how effective is the system in satisfying the user's information need, which is the focus of this lecture. There are 3 sets of documents. The first one is collection which is the set of all available documents. The second set is the answer set that an IR system has returned in response to a query. Finally, it is relevant set that have been judged by the experts to be relevant for that query. In general, the intersection between answer set and relevant set is very important, whose proportional relationship with answer set or relevant set is important factors in evaluating IR system. This week, I will learn precision, recall, P@N, R-precision and Mean Average Precision.s

Precision and Recall are the basic evaluation metrics about the intersection between answer set and relevant set. Precision is the fraction of the set of retrieved documents (Ret) that are relevant. Recall is the fraction of the relevant documents (Rel) that have been retrieved. In the following, it is the formula.

- $Precision = \frac{|RelRet|}{|Ret|}$

- $Recall = \frac{|RelRet|}{|Rel|}$

Precision and Recall are often inversely related: as one increases the other decreases. Both are more suited to their own application scenarios.

P@N means precision at N. It is the proportion of relevant documents amongst the top N results. The performance of P@N depends on the number of relevant documents. If the number of relevant documents is much larger than N, a good performance is easy. R-precision means the precision after top R documents are returned. R is the number of relevant documents.

Mean Average Precision (MAP) is the most commonly used metric to evaluate IR system. There are three steps to calculate it. The first step is that calculating the precision at each recall point. Then dividing the sum of these precision calculations by the total number of relevant documents. The final step is to target multiple queries, it need calculate the mean of the queries' average precision values.

Lab/assignment work:

As for the lab of this week, it is about the vector space model. One interesting question I

need think is choosing which data structure to store the vectors. Because the document id is in order, so both of dictionary or list may can store vectors. But dictionaries are more applicable. As for the data structure of storing each vector, it is an interesting question. When I do the exercise about vector space model, I only the order of one vector is according to the order in which terms appear. When coding, it is difficult to find all terms in all documents, because it needs to read terms for one document after another. That means the order of terms in one vector is difficult to calculate. Consequently, dictionary can used to store each vector, and the key is the name of term.

The solution of this lab is very similar to the last lab. Both of them need to go through all terms, and calculate the occurrence of term. In this lab, the dictionary of store the appeared term needs to be refreshed with the emergence of a new document.

As for the code about representing the query vector, calculating the cosine similarity, and printing the result in order is very easy.

The most complex operation is getting the vector of documents. In fact, only the terms which appear in the query is useful in the Boolean weighting scheme. We may only need to calculate the length of each document, and care about the terms which appeared in the query. That means we need to represent the vector of query first, and get the vector of documents according to the terms appeared in the query. That may can save some time during one query. However, this needs to be recalculated the vector of documents every time once a query is received from a user. In addition, this way is not suitable to the tf-idf scheme.

Reflections:

As for R-precision, there is a very interesting situation in my opinioin. R-precision means the number of relevant documents that are retrieved in the first R that is the total number of relevant documents results divided by R. If the number of retrieved documents is less than R, the top line will be the number of all relevant documents which were retrieved, and the bottom line is the number of all relevant documents. That means R-precision will be equal to recall.

Both recall and precision have more important cases. For example, recall is even more important when retrieving who is at risk of being infected with COVID-19. The reason is that we must find all infected people to avoid the outbreak of COVID-19. On the other hand, when someone want to search some information about their symptoms and medications, precision is more important. Otherwise, wrong information may exacerbate that person’s symptoms. There is a method that can comprehensively consider recall and precision, called F-measure [12]. It is the weighted harmonic mean of Precision and Recall. In addition, the formula is in the following.

$$F = \frac{(\alpha^2 + 1)P * R}{\alpha^2(P + R)}$$

P means precision, and R means recall. The function of a is to indicate which is more important, precision or recall. In general, a is set to be 1. At that time, it is called F1-measure, and the formula is in the following.

$$F1 = \frac{2 * P * R}{P + R}$$

In addition, precision has sone problems. The same precision does not mean their performance is the same. Because the performance of IR system which retrieved before is better. So, Mean Average Precision was created

There is a different indexes and calculation methods besides Recall and Precision. It is Receiver Operating Characteristic (ROC). There are two metrics in ROC, True Positive Rate (TPR), and False Positive Rate (FPR). They represent the rate of true positive in real positive, and the rate of true negative in real negative. The ROC curve with FPR as the abscissa and TPR as the ordinate can intuitively represent Performance. Classification and identification are more appropriate in this way. Retrieval is more suitable for Recall and Precision.

Week 8（beginning April 19th 2021)

Lecture Topic: Evaluation(part2)

My Learning:

The content of this week has two parts, bpref and NDCG.

The bpref means Binary Preference. When the collections are large, and not every document can be judged to be ‘relevant’ or ‘non-relevant’, there will be some document that is un-judged. Un-judged document does not mean it is non-relevant. If the document was treated as a non-relevant document simply, the average precision will be greatly affected. The bpref can eliminate this effect as much as possible. The formula is in the following.

$$B = \frac{1}{R}\sum_{r \in R} 1 - \frac{|n\ ranked\ higher\ than\ r|}{R}$$

This formula is not very easy to understand. The r means the rank of one relevant document, n means the non-relevant document, and n ranked higher than r means the number of non-relevant documents whose rank is less than 4.

As for NDCG, it supports that some documents are more relevant than others. NDCG thinks the highly-relevant documents is more important than mildly relevant ones, and the relevant documents in early position is more important than ones in later position. NDCG means Normalised Discounted Cumulated Gain. For each rank of document, NDCG is equal to DCG/Ideal DCG.

The formula of Discounted Cumulated Gain (DCG) is in the following.

- $$DCG[i] = \begin{cases} G[1], & i = 1; \\ \frac{G[i]}{log_2 i} + DCG[i-1] & i > 1 \end{cases}$$

The i means the rank of each rank, G[i] means the graded relevance judgements which is used to represent the degree of relevance of the document. The bigger G[i] is, the more degree of relevance of that document. Log2i is used to let the later documents add less to the gain.

The Ideal DCG is the DCG that the IR system had the best retrieval. That means it begins with all documents who has the highest relevance level, and then it includes all documents who has the second highest relevance level.

The advantage of NDCG over DCG is that it is easy to compare, because it has a fixed range of values, 0 to 1.

Lab/assignment work:

This week has an exercise and a lab.

As for the exercise, it is not difficult. By overserving the retrieved set, I found it is not a good system, because a lot of relevant documents were found lately. By computing, the results of evaluation metrics are in the following.

1. Precision=0.33
2. Recall=0.75
3. P@10=0.1
4. R-precision=0.125
5. MAP=0.23
6. Bpref=0.2
7. NDCG@10=0.27

The precision of this IR system is not good, the recall is better. All of MAP, Bpref and NDCG can show the bad performance of this system. The situation of NDCG is better, it may because it found a very important document early, though it found few relevant documents in the first 10 documents. The reason why Bpref shows the worst performance is that there are too many non-relevant documents were found before relevant documents. If the number of non-relevant documents is more than the total of relevant documents, the rest of relevant documents are meaningless for Bpref.

As for the lab, it is very similar to the last week. The only different thing is the way of calculating the weighting. When running one loop, we need to calculate as much as thing we can calculate. As for how to calculate TF-IDF, we need to consider how to calculate each other. TF is equal to that the frequency of one term in one document divided by any maximum frequency term in that document. The frequency of one term and the maximum frequency can be calculated as terms are iterated in one document. IDF=log2N/ni, N is the number of documents, n is the number of documents where term i is. The ni need to be counted as the documents are iterated. If we want to store the information about IDF straightly, it is very hard to calculate. At this situation, we have to store every ni and the value of every log2N/ni with the change of ni. Consequently, we can calculate ni when calculating the frequencies. Then, we can get the value of IDF of each term. At that time, the log2N/ni of each term only need to be calculated once.

After watching the video about solutions, there three keys about optimization. The first one is the data structure of storing stopwords. Because reading information from set is far fast than reading information from list. Stopwords need not be changed, so set is a very suitable data structure of storing stopwords to fasten our program. The second point is removing useless stemming that I was mentioned two weeks ago. We can create one dictionary to store the stemmed terms to avoid useless stemming. The third point is we should care about which part of code only need to be ran once, and which part of code need to be ran over and over again. For the lab in this week, the code before calculating

the vector and the length of query only need to be ran once, so plenty of calculation need to be finished in this part. With the change of terms, the rent of code may need to be ran over and over again, so we need to reduce the amount of computation as much as possible in this part.

Reflections:

As for far, I have learned a lot of matrices to evaluation the performance of IR system. However, I think almost matrices need to judge if the documents are relevant, even the graded relevance. In fact, it is very hard to judge the relevance. Relevance needs to be considered from several perspectives, such as system perspective, cognitive perspective, situational perspective and motivational perspective [13]. If there are a lot of documents, it is very hard to determine their relevance. So, I think the best way to evaluate the IR system is evaluating a limited sample of documents using multiple queries. We can test a lot of terms, instead of using a large number of sample documents.

There are two ways to calculate the average of queries, macro average and micro average. Micro average means calculating the sum of testing documents, and get the average value. Macro average means calculating specific value of each testing, and get the average specific value. For example, there are two queries, q1 and q2. The relevant documents of q1, q2 are 10, 20 respectively. The q1 has 20 retrieved documents in which there are 5 relevant documents, q2 has 100 retrieved documents in which there are 10 relevant documents. The precision of q1 is 0.25, the recall of q1 is 0.5; The precision of q2 is 0.1, the recall of q2 is 0.5. The macro average of precision is (0.25+0.1)/2=0.175, the macro average of recall is (0.5+0.5)/2=0.5; the micro average of precision is (5+10)/ (20+100) =0.125, the micro average of recall is (5+10)/ (10+20) =0.5. As long as the sample size is stable, either method will work. At the same time, the difference between the results of the same system should be small, and both methods should be applicable. However, if we let the number of samples fluctuate, we can judge whether the system is stable or not based on the difference between the macro average and micro average.

In general, we can calculate the MAP (mean average precision). Sometimes, we can use MRR (Mean Reciprocal Rank) to evaluate the performance of IR system. For a query, if the first relevant document is in the NTH place, the RR score is N. If there is no relevant document, RR score is 0. MRR is equal to the average of RRs. If the precision or recall has bad performance in one system, MRR may can used to evaluate it.

Week 9（beginning April 26th 2021)

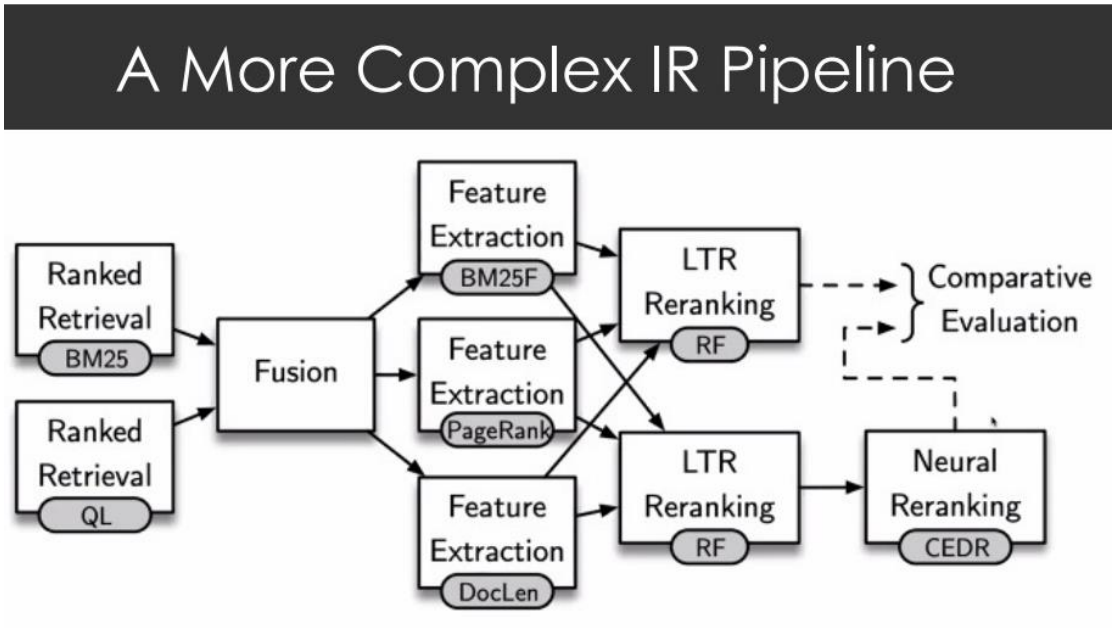
Lecture Topic: IR Pipelines and Modern IR

My Learning:

The course of this week is about finding the issues of classical IR system, and find the way to solve them in modern IR system. This course is an introduction for the following courses.

As for the issues of classical IR system, this course introduced the classical IR pipeline at first. The pipeline is linear pipeline which has 4 stages, preprocessing, indexing, ranking and evaluation. These stages happen sequentially, and the output of one process becomes the input of the next process. If some stages can happen at the same time, it can save some time or make the result more accurate. These is one of the issues of traditional IR system. The second issue is the vocabulary and ranking model mismatch, If some stages can happen in parallel, a combination of ranking models and vocabularies can be tried to use. The third issue is that only searching the document contents is not enough, there is a lot of information in other place, such as title and link. In addition, the classical IR system doesn't make use of recent advances in machine learning, natural language processing and natural language understanding. Last but not least, the speed issue is very important, it is not fast enough. The issues 2-4 was caused by the issue 1, so we can redesign the pipeline of classical IR system to solve these issues.

In the following, it is a more complex IR pipeline which can solve some issues of classical IR system.



In this IR pipeline, multiple rank models, query expansions, feature extractions and

reranking can solve the issues 1-3 which mentioned before. Firstly, it can use pseudo-relevance feedback that means re-running the search over and over again with assuming that the top k ranked documents are relevant to evaluate and optimize the system. There are a lot of ways about query expansion, such as adding related terms. The related terms can be got by manually-created thesaurus, automatically-created Thesaurus, Word embeddings, Query Log and Target-corpus based techniques. Fusion can make the results better by the scores calculated by the Ranked Retrieval systems, the rank of the documents in each results list and the probability of a document be relevant. In addition, machine learning can used to re-rank the document. PageRank which considers the link structure of the web is one of the features of learning to rank the models. As for the neural reranking, Bidirectional Encoder Representations from Transformers (BERT) is a very important model.

Lab/assignment work:

This lab has 7 steps.

The first step is getting the query documents, the relevance of documents, result documents, the rank of document and the score of documents. All of information can be stored in document, the key can be the document id. List also can store the query documents and result documents, but reading list is slower than reading dictionary for python. The sample set is small, so both of them is fine.

The second step is getting some number of specific sets which are very useful for the lab, such as the number of relevant documents in the query documents and the number of relevant documents in the results documents.

The third step is calculating precision and recall according to the numbers which were calculated in the above step. This step is very easy.

The fourth step is calculating the precision 10 and R-precision. In this step, we need create get the number of relative documents in the first N (R or 10) results documents. As long as the traversal N the result set and record related document number is ok.

The fifth step is calculating MAP. This step is very similar to the last step, the only different thing is that we need to record the number of relative documents whose rank is smaller than one relative document divided by the rank of that documents. These results are then added up and divided by the number of relevant documents.

The sixth step is calculating the Bpref. Before writing the code, I should think about what value should I use according to the formula. All of the number of relevant documents, the number of relevant documents in the result sets, and the number of non-relevant documents whose rank is smaller than one relevant document. Both of the number of relevant documents is known, so we only need to calculate the number of non-relevant

documents, and record each 1 - the non-relevant documents divide by the number of relevant documents (this devisor cannot bigger than 1). These results are then added up and divided by the number of relevant documents in the result set. In the following, it is the code of this step.

```
def bpref(resultset,relevance,queryset,Numberofrel, Numberofrelret):
    result = 0
    i = 0
    Numberofnonrel = 0
    for doc in resultset:
        if resultset[doc] in queryset and relevance[doc] != 0:
            i = i + 1
            result = result + 1 -Numberofnonrel / Numberofrel
        else:
            if resultset[doc] in queryset and Numberofnonrel <= Numberofrel:
                Numberofnonrel = Numberofnonrel + 1
        if i == Numberofrelret:
            break
        Bpref = result/Numberofrel
        print(Bpref)
```

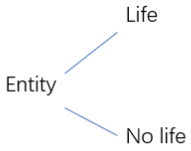
The last step is calculating NDCG@10, we need to calculate DCG of rank 10 and ideal DCG of rank 10. The ways of calculating them is very similar. After sort the result list by the relevance of documents, the way of calculation is same. It's worth noting that the dictionary is converted to a list after sorting. When coding, treat the data structure that stores the relevance as a List. Calculate the DCG and Ideal DCG of the rank 10 according to the formula, and then come up with it.

The most difficult thing is counting the non-relevant documents for Bpref. But after careful analysis, it's not very difficult.

Reflections:

I think this week is used to introduce what we should learn in the following. So, the course is not difficult, and I do not have a lot of thinking about it. So, I will explain some terms I do not understand before. The course introduces a lot of way to find the related words, some ways have some examples. In the following, I will explain Wordnet and word embedding.

As for Wordnet, it is a semantic network covering a wide range of English words. Nouns, verbs, adjectives, and adverbs are each organized into a network of synonyms. Each synonym set represents a basic semantic concept, and these sets are connected by relationships. In addition, a polysemy will appear in the synonym set for each of its meanings. In the first version of WordNet, there was no connection between the four different parts of speech networks. A very important of parts is nouns, the relationships in this part main are hierarchical relationship. In the following, it is a very simple example. In fact, it is more complex. The deepest level of the noun hierarchy is 16.



The word embedding means to represent each word as a vector and then reduce the dimension. In the beginning, every word is a dimension, it is called one-hot representation. The distributed representation can decrease the dimensions. In the following, It is an example of one-hot representation.
Child (1,0,0,0,0,0,0,0)
Adult (0,1,0,0,0,0,0,0)
Male (0,0,1,0,0,0,0,0)
Female (0,0,0 1,0,0,0,0)
Boy (0,0,0,0,1,0,0,0)
Girl (0,0,0,0,0,1,0,0,)
Woman (0,0,0,0,0,0,1,0)
Man (0,0,0,0,0,0,0,1)

Because the child of male is boy, the adult of male is man, the child of female is girl, the adult of female is women. For the distributed representation, the example terms can be shown as following.
Child (1,0,0,0)
Adult (0,1,0,0)
Male (0,0,1,0)
Female (0,0,0,1)
Boy (1,0,1,0)
Girl (1,0,0,1)
Woman (0,1,0,1)
Man (0,1,1,0)

This reduces the dimension by half. But in general, dimensionality reduction is realized by deep learning and neural network based on context. A very useful way to test word embedding is whether it realize the analogy. For example, Apple-fruit is equal to Cat-animal.

Week 10（beginning May 3rd 2021)

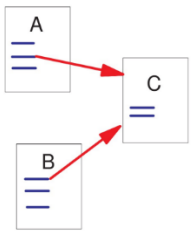
Lecture Topic: PageRank

My Learning:

The course in this week introduces the concept of PageRank, and how to calculate the PageRank score.

For the web, there is an interconnection between the documents. Two pages can link to one another without sequential order. This may cause some mathematical difficulties. In addition, there may be a lot of pages which link to one page, or one page may link to a lot of pages because anyone can let their page connect to a lot of documents.

As for how to calculate the Page Rank score, there are two cases for the page who should have high score. There first case is that the page is linked to by many documents, and the second one is that the page is linked to by documents that have a high PageRank score. To explain the formula to calculate the PageRank Score, there are two terms which need to be understood. In the following, it is the explain of these two terms, backlinks and outlinks.



In this situation, we say that A and B are backlinks of C, and A and B have outlinks to C. Then, the following formula can be explained.

$$R(u) = \sum_{v \in B_u} \frac{R(v)}{N_v}$$

- R(u) is the PageRank score for document u.
- B_u is the set of all backlinks of document u.
- R(v) is the PageRank score for document v.
- N_v is the number of outlinks in document v.

R(u) is to enhance the influence of the high score backlink, Nv is to reduce the backlink has too many outlinks influence. Because the pages are linked to each other, so we need to give an arbitrary score to every document to be the beginning of calculation. After a few iterations, the data is generally stable. For a group of pages, they may do not link to anywhere else outside the group, that will cause their scores may not be influenced by the outside world. In addition, no backlinks page may have 0 PageRank. However, if the link to others, they should make difference. The following improved formula can solve these problems.

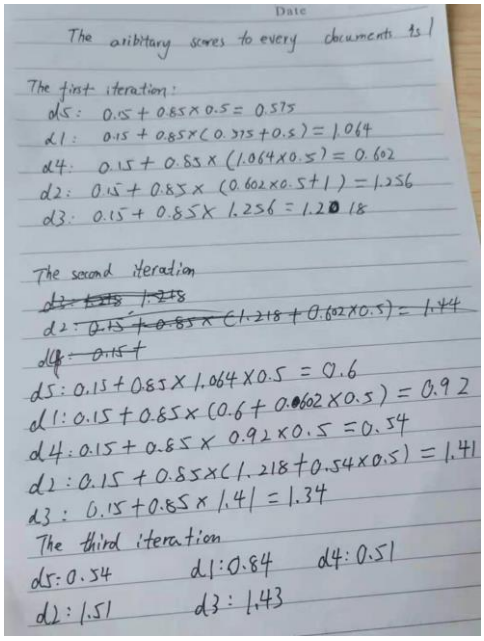
$$R(u) = (1-d) + d \times \sum_{v \in B_u} \frac{R(v)}{N_v}$$

The d means damping factor which means not all of a document's PageRank is passed on via its outlink. Damping factor may be 0.85 in general. Using this formula, most scores tend to stabilize after several iterations. What's more, the number of iterations does not increase rapidly with the increase of data. That means it is suitable for the large-scale data collections.

Lab/assignment work:

There is no lab in this week. So, I will write something about the exercise in this part.

This exercise has three basic question, and one advanced question. In the following, it is the solution of question 3, which is an example solution of basic questions.



By doing the exercise, I find the trend is the same for each document at the beginning. When the score of one document increases once, it may keep on increasing. By looking at the lines that we talked about in class, I found the trends for all documents will change at the same time. When the trend of a score levels off or changes suddenly, the trends in other scores will change as well.

As for the advanced question, I think dictionary will be a suitable way to store the scores of documents. The key is the document ID, and the value is some lists which is used to store the score of one document per iteration. The lists can be used to find when all scores are stable by the difference between two adjacent values, and what the final results are. Each time a document is traversed, its score is calculated based on the data.

Reflections:

I think the course in this week is very useful. PageRank is a very popular way to sort web pages, and it is easy to understand.

There are some interesting situations that was mentioned in the class of PageRank. One situation is called rank sink. When a group of pages have links to each other and have backlinks from other pages in the graph but have no outlinks to other pages in the graph, it is rank sink. When traverse to this page, it is like a black hole, can only spin inside, cannot come out. In this situation, if the user use simplified version of PageRank, the score of that group will be 1, and the score of other documents will be 0. The other situation is that one web has no backlinks. That will cause that all of the score are 0 after iterations, because that web is the end point. Luckly, damping factor can solve these problems. However, PageRank still has some disadvantages, such as not friendly to new web pages and not filtering spam links. Both of these disadvantages are because that PageRank treats every link fairly. The weight PageRank (WPR) takes into account the importance of a page's links and links, and assigns ranking scores based on the popularity of the page [14].

$$PR(u) = (1-d) + d \sum_{v \in B(u)} PR(v) W^{in}_{(v,u)} W^{out}_{(v,u)}$$

$W^{in}_{(v,u)}$ is the weight of link(v, u) calculated based on the number of backlinks of page u and the number of backlinks of all reference pages of page v. $W^{out}_{(v,u)}$ is the weight of link(v, u) calculated based on the number of outlinks of page u and the number of outlinks of all reference pages of page v. This method can consider the weight of link.

There is anther way to sort the web, named HITS [14]. Hub page and Authority page are the two basic definitions of HITS algorithm [15]. Authority page is high quality web pages related to a particular field or topic. For example, YouTube I a authority page about video. Hub page is a Web page that contains a lot of links to high-quality Authority pages, such as the browser home page. HITS algorithm is from a large number of web pages to find the Hub page and Authority page which are about the topic of user. Compared with PageRank, HITS algorithm is closely related to user queries, and has low computational efficiency. It is more suitable to deal with specific user queries, and its structure is unstable. Overall, PageRank is better, but the HITS algorithm is better suited for client deployment.

Week 11（beginning May 10th 2021)

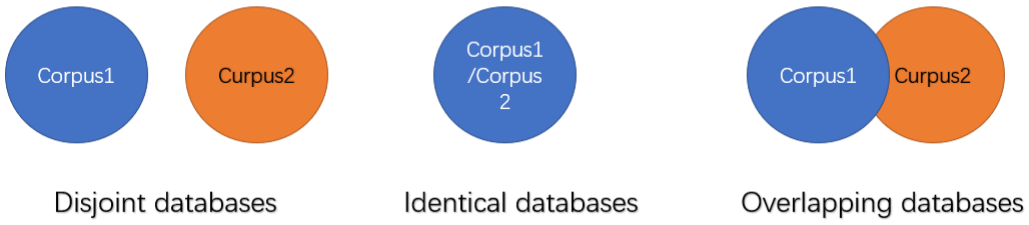
Lecture Topic: Fusion (part1)

My Learning:

The course in this week introduces the fusion and explain how to calculate the tank based fusion.

Fusion is combining the outputs of multiple different information retrieval systems into a single ranked result set which can make the result set be better quality. In general, better recall is easier to implement. As for how fusion system works, it calculating a score for each document first, each underlying IR system can make difference on the scores. But it is influenced by the position/ rank of the documents and the quality of the system. Then the fusion system returns the user a documents List sorted by score. In addition, fusion can be used for Metasearch, Distributed information Retrieval, Internal Metasearch.

The corpus of each IR system is different. Corpus Overlop has three levels, and for each level, Fusing will process the result seft differently. In the following, it is a figure which can explain the three level of corpus overlap.



Our focus is Identical databases, the document appear in multiple result sets can prove it more relevance, because all IR system will deal with that document. This is called data fusion. As for the overlapping databases, it is difficult to find whether the document is not in the corpus of the system, or the system think that document is irrelevant. In addition, the fusion of disjoint corpora is frequently known as Collection Fusion.

There are three ‘effects’ of fusion algorithm, the first one argues the top documents from each result set has better performance named the skimming effect. The second is called the chorus effect. It argues that the documents were though are relevant by multiple systems is more important. The last one is dark horse effect that is where the resuits of system is different from others, it may be unusually accurate or unusually inaccurate.

There are 3 principal categories of Data Fusion, rank-based fusion, score-based fusion and segment-based fusion. Rank-based fusion only care about the rank, Score-based fusion consider the relevance scores, and the segment-based fusion divide the corpus into groups of documents, instead of single ranks or scores. This course introduces how to calculate the rank-based fusion.

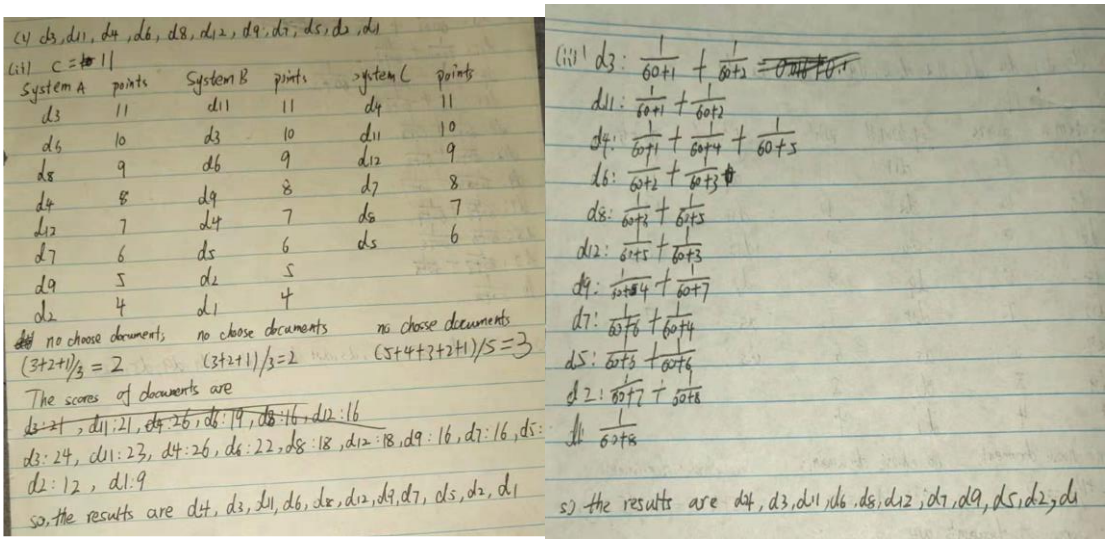
There are three ways to calculate rank-based fusion, interleaving, Borda-Fuse and Reciprocal Rank Fusion. The interleaving is just adding the documents from top to bottom in round. As for Borda Fuse, it gives each document a point according to the rank, and the point of un-choose document is 1+..+the last rank/the last rank-1. The score of Reciprocal Rank Fusion is calculated by the following formula.

- $RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k+r(d)}$
- where *r(d)* is the rank of document *d* in result set *r*, and *k*=60 (set by experiment)

Lab/assignment work:

There is no lab in this week. So, I will write something about the exercise in this part.

This exercise has 2 basic questions, and one advanced question. In the following, it is the solution of question 2, which is an example solution of basic questions.



The results of three ways to calculate the rank-based fusion are very similar. For the small corpus, the Reciprocal Rank Fusion is not vey suitable, because it cares about the times of one document appears into different system too much. K=60 is not suitable for the small corpus. As for the Borda Fuse, it makes more sense because it focuses on both the rank and the number of occurrences on multiple IR systems.

As for the advanced question, interleaving is easy. It iterates through each document in multiple systems, and then stores the non-duplicated documents in a new dictionary. As for the Borda Fuse, it needs to calculate the points according to the number of documents and the number of retrieved documents. Using a dictionary to store the points, the key is document id, and the value is another dictionary which used to store the points for each system. The key is system name, the value is the points. Then creating a list to store the final points and documents id, and the list is ordered by the final points. The way to

implement Reciprocal Rank Fusion is very similar to the Borda Fuse.

Reflections:

I think the most interesting content of this course is the collection fuse, which can be used in the disjoint database. I think it is very amazing, I don't know how to increase the result by totally different corpus. There are two fusion strategies about Collection Fusion Strategies, the relevant document distribution fusion strategy and the query clustering fusion strategy [16]. Both of these ways are useful for collection fuse. In the following, it is the diagram of the steps.

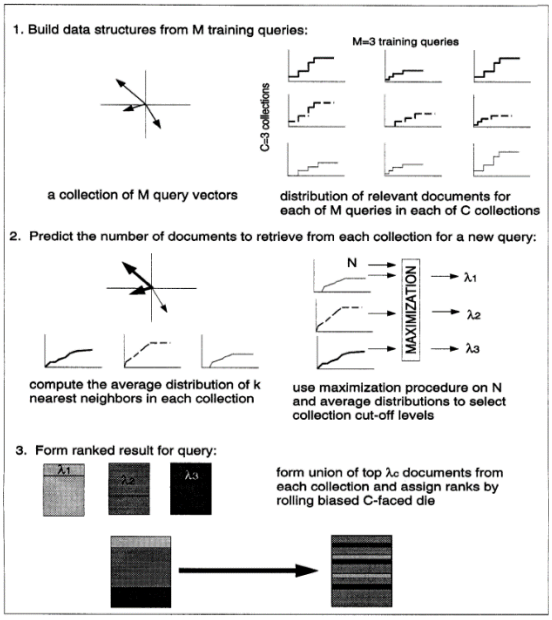


Figure 2: The relevant document distribution fusion strategy.

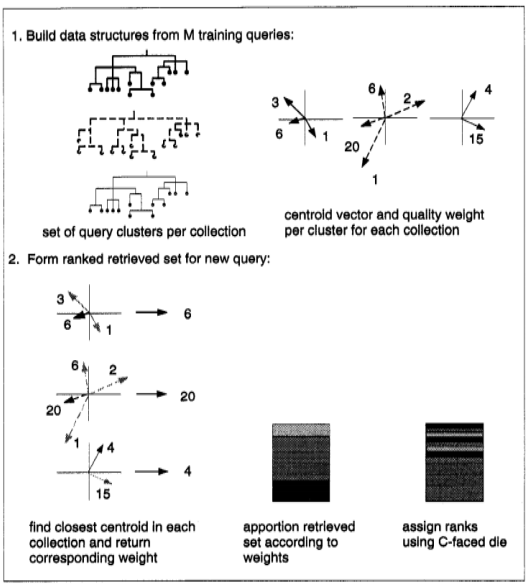


Figure 3: The query clustering fusion strategy.

Fusion has a few input IR systems which is very important, because fusion need to optimize the result set according to the input IR system. If the input IR system is not good, the fusion will get bad results. Jia and Deng mentioned there are some factors of the IR system, which will influence the fusion [17]. One of the influencing factors is whether the information retrieval system involved in fusion are efficient and have similar performance. In this case, when the information retrieval systems with large performance differences are sorted and combined, the weighted fusion of information retrieval modes is needed. The second factor is whether the information retrieval systems involved in the fusion have great cognitive differences. If they are cognitively similar or even identical, the resulting fusion performance will not be as ideal. The third point is whether the information retrieval system involved in the fusion are comparable. The fourth is whether the information retrieval system involved in the fusion are evenly distributed in the sample space composed of all the information retrieval modes. Uniform distributed information retrieval system has great cognitive difference and can reflect the query demand from multiple levels in the cognitive sense. The fifth is whether the number of information retrieval modes involved in the fusion is sufficient. The last point is that the selection of information fusion baseline information (including score-based information and

ranking-based information) affects the selection of information fusion methods, so as to affect the performance of information retrieval fusion.

Week 12（beginning May 17th 2021)

Lecture Topic: Fusion (part2)

My Learning:

This course introduces how to calculate score-based fusion and probability-based fusion.

There are two ways to calculate score-based fusion, CombSUM and CombMNZ. Both of the final scores are calculated according to the score of each IR system, but the scoring criteria for each IR system are different. So, we need to do some normalization to let each system have the same weight. The most common way is standard normalization, which is calculated by the following formula.

$$normalised\_score = \frac{unnormalised\_score - min\_score}{max\_score - min\_score}$$

As for the CombSum, it is very easy and useful. It is just calculated by adding the score of each IR system. The CombMNZ is calculated by multiplying each CombSum score by the number of IR system that think the document is relevant.

The weight to each IR system may be not the same, so we need a way to apply the weighting. At first, each document's normalised scores are multiplied by the weighting of IR system. Then, the calculation way is the same as CombSum. There are two ways to calculate the weights, the CORI algorithm and LMS. CORI algorithm was calculated by document frequency and inverse collection frequency. As for LMS, longer result set results are in higher weight.

However, a single weight does not accurately assess the overall performance of a system. Probability-based fusion, it can reflect the ranks/positions in the results sets where the IR system tend to return relevant documents. There are three ways to calculate Probability-based fusion, ProbFuse, SegFuse and SlideFuse.

As for the ProbFuse, the result set is divided into x equal-sized segments. The final score of the document is based on the probability that the document is relevant in that segment. Before calculating the scores, we need to do training phase. First, calculate the probability of the relevant documents in each segment, and then calculate the average of the probability of the segment in the same position in each system. Finally, calculating the average over all training queries. It exploits the Chorus Effect, so it is only suitable for data fusion.

There are a lot of closed documents has the same score. The size of segments on SegFuse increase exponentially as we go down the result set. For each document, the probability score of its segment was multiplied by its normalized score. As for training SegFuse, it is very similar to ProbFuse. The only difference is the size of segment. The

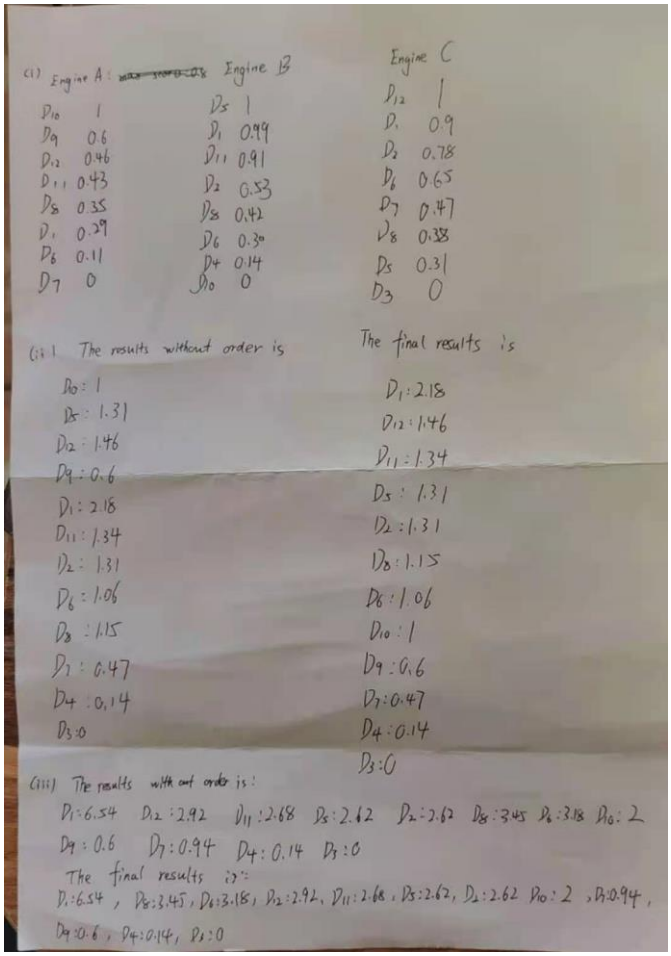
size can be equal to (10 x 2k-1) – 5 where k is the segment number.

SegFuse can improve the performance than ProbFuse, especially using MAP matric. However， the adjacent documents are treated totally differently. As for SlideFuse, we use each rank's neighbors to get the score. For the training phase, calculating the probability of relevance at each rank. We need to do multiple queries in a system. The way to calculate is that the number of relevant documents returned at rank k divided by number of the times of training queries. Then, calculating the average probability of each of the five adjacent positions including that rank, as the score of that rank for one IR system. The final score is the sum of the scores for each IR system.

Lab/assignment work:

There is no lab in this week. So, I will write something about the exercise in this part.

This exercise has 3 basic questions, and one advanced question. In the following, it is the solution of question 2, which is an example solution of basic questions.



This exercise is very easy. As for the CombSUM, I thought it emphasize Chorus Effect a lot. However, it is not enough after I calculating this exercise. The score of documents with high score may has high final score than the documents in more results sets. CombMNZ

emphasize Chorus Effect more. The effect is very clear. If a document has high score in one document, the reason may be the system is not accurate. In my opinion, one document that were thought as a relevant document by multiple system is more important than it was thought as a high relevant document by seldom IR system.

In addition, one interesting thing is that if the document is the last document in a set, the normalization of score is 0. At that time, I need to think it is in that document although the score is 0 when we calculate the CombMNZ.

Reflections:

Score-based fusion and probability-based fusion are similar in my opinion. Probability based fusion is a little harder to calculate, but it is more accurate. Compared with Rank-based Fusion, score-based Fusion and probability-based Fusion pay more attention to Chorus Effect. However, all of them are the fusions of the result set. Li Pei mentioned that there are other kinds of fusion [18]. The first one is the fusions of description, the retrieval results of multiple methods are better than any one method. For example, the retrieval effect of combining title and keyword is better than that of single title or keyword. The second point if query fusion. People's different way of querying will get different query results. The fusion of different Boolean query expressions, the combination of Boolean expressions, natural language, and weighted words can improve the retrieval efficiency. And the third kind of fusion is the fusion of methods, which is a mixture of models that we've studied before. Another fusion method is corpus fusion, mainly Collection Retrieval Inference network (CORI), Modeling Relevant Document Distributions (MRDD) and Query Clustering (QC). In general, the purpose of this fusion method is to better find the corpus that is closer to the search results. The last one is Web fusion, which is the specific application of multiple data sets fusion in the Web environment. Different Web search engines can be regarded as multiple data sets in this environment.

There are three specific approaches to Web Fusion, MetaCrawler, ProFusion and SavvySearch. They all use some fusion algorithm to merge and sort the results from multiple search engines. Metacrawler uses a simple method of combining normalized score values. ProFusion uses a weighted merge approach in which the normalized score is weighted by the search engine efficiency of calculating the query results from the training set. SavvySearch also uses a method of combining weighted values based on search engine performance. Unlike Profusion, however, it measures search engine performance with statically accessible statistics about the search process, such as links returned and followed.

Week 13（beginning May 24th 2021)

Lecture Topic: Relevance Feedback and Query Expansion

My Learning:

This course is mainly about relevance feedback and query expansion.

There are three ways of Relevance Feedback, common relevance feedback, pseudo relevance feedback and indirect relevance feedback. All of these ways are to find the relevant documents, then a relevance feedback algorithm will be used to compute a better representation of the information need. The common relevance feedback is that the users judge the document is relevant or nonrelevant. In practice, users generally do not wish to provide feedback. As for pseudo relevance feedback, it assumes the top k ranked documents are relevant, which is very useful in practice. Indirect relevance feedback is the evidence from other sources, such as the time the user browses, the number of times the user browses.

As for the Rocchio Algorithm, it is a best-known relevance feedback algorithm. On it, each document and query are represented as vectors, and the algorithm want to find a query vector that can maximize the similarity of related documents, and minimize the similarity of unrelated documents. In the following, it is the formula.

$$\vec{q}_{opt} = \arg\max_{\vec{q}}[sim(\vec{q}, C_r) - sim(\vec{q}, C_{nr})]$$

- where:
  - $sim(...)$ is a measure of similarity (e.g. cosine similarity)
  - C_r is the set of relevant documents
  - C_{nr} is the set of nonrelevant documents

The centroid is the center point of a set of vectors, the algorithm can subtract the centroid of the irrelevant set from that of relevant set by the following formula.

$$\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r}\vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j \in C_{nr}}\vec{d}_j$$

In practice, we use this algorithm to get a modified query vector according to the known non-relevant document and known relevant document. The following formula is used to modified query vector. I think it is the most useful one.

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

- Here:
- $\vec{q}_0$ is the original query provided by the user.
- D_r and D_{nr} are the documents that have been identified by the user as being relevant and nonrelevant respectively.
- α, β and γ are weights that can affect the behaviour of the formula.

a = 1, β = 0.75 and γ= 0.15 or 0. In addition, there are two assumptions about relevance feedback. One is the user must know how to choose the relevant documents, and the other is relevant documents close to each other in a cluster.

As for the query expansion, it is the process of adding terms to a query in order to increase recall, it may reduce precision. There are two kinds of query expansion, interactive query expansion and automatically query expansion. The first one is that the system supports some relevant terms to user that they can choose to query. Aa for the automatically query expansion, the most useful way is thesaurus-based query expansion. The thesaurus is a database to store the synonyms, and it can be created by analysing the distribution of words in documents. When the words co-occur with similar words or occur in particular grammatical relations with the same words, they may be similar. In fact, creating thesaurus is expansive, so there are two other ways to realize automatically query expansion. The first one is word embedding which is a representation of word as vectors in an n-dimensional space. It is assumed that words that are close to one another in the vector space have similar meaning which is very similar to relevance feedback. The second one is query log mining which is storing the information of previous queries and user behavior.

Lab/assignment work:

There is no lab or exercise in this week. From this week, this part will show how I finish my assignment and some thoughts about it.

I have finished BM25 part of the assignment. It is very similar to the lab we did before. At first, we need to consider which step should we calculated before querying, and then consider which value we need to store and get to calculate BM25 score and choose the suitable and fast data structure to store these values. Then, calculating these values and BM25 score. Finally, calculating the querying result according to the query user input.

As for the first step, I think I should calculate the value as much as possible before querying. The reason is that we need to querying over and over again, and the result of pre-processing can be stored. So, I decide calculate the BM25 of each term in each document before querying. It can save a lot of time. According to the requirement of assignment, I also need to choose a way to store the BM25 scores. The scores will be stored in a dictionary, and json is a very useful way to store and load a dictionary as a file.

The second step is I need to know which value should I calculated and when should I calculate it. According to the formula. I need to know the average length of the document (AVG), the length of each document (Length), the appeared times of each term (Times), the frequency of each term in each document (Frequency). All of these values can be calculated when going through all terms. After calculating them, we can calculate the BM25 according to the formula.

As for the rest step, it is not difficult. I just need to calculate the sum of terms which is in the query in one document, and rank documents.

There are other points I need to pay attention to in this part. How does a point handle a particular symbol in the corpus? I choose to use strip to remove these symbols from the end of terms. The other point is how I judge if the user has pre-processing our project? I choose to use try-except to find if the index file exists.

Reflections:

I think Query expansion is very useful ,because Query it can increase the Recall, and it may increase the precision sometimes. I have learned a lot of kinds of way to improve IR system performance to improve recall in the course. However, to improve the recall rate and accuracy also needs the user's own efforts. Precision may be difficult to improve because it may be affected by the user's expressive ability. There are some ways can help the user know how to improve the recall of their querying.

The first way is trying more querying using synonyms, synonyms, related words, different times, singular, plural, and abbreviations of term. They IR system also may consider these things by query expansion. However, the user's understanding of relevant words is likely to be different from the machines'.

The second method is to use a truncated word, usually "?" and "*", which mean different things in different databases. But '?'usually means one character, and ' * 'usually means more than one character.

The third way is to use superposition words, which refer to a wider range of words, such as subject words.

The fourth way is using Boolean expression. It is useful, and OR can get more result than AND.

The fifth method is to reduce the restrictive conditions and add sub-subject words. If there is a restriction in the previous retrieval, relax or remove the restriction; If you use a combination of subject and sub-subject words, see if you have missed any possible sub-subject words.

There is also a way to improve the precision. It is suitable for some search engines, such as Baidu. This is done by placing a precondition before the search term. These preconditions include the location of term in the document type (title or text), file type, website type, time, etc. For example, you can add ‘intitle:’ before the term, it can help only get the result whose title has the term. It is useful to remove the advertisement and other non-relevant information on Baidu. There are some other pre-condition can be added on Baidu, ‘filetype:’, ‘intext’, ’inurl’ and ‘AAAA ... BBBB’ in which AAAA is the start year, and BBBB is the end year.

Week 14（beginning May 31st 2021)

Lecture Topic: Web IR

My Learning:

This course has two parts, web crawling and adversarial information retrieval.

As for the web crawlers, it is the programs that automatically search documents on the web. It works by extracting hyperlinks from each page it finds and then directing them back to the new page to be indexed. It finds the information according to the breadth-first spider search and depth-first search. They are similar and runs in parallel. Breadth-first spider search is adding “seed” URLs to the queue of pages to visit. When there ate URLS inn the queue, it removes the first URL and download the page, and then extract the link. Lastly, these links were added to the queue if they have not been visited or are not in the queue.

There are for crawlers polite.

- Obeying robots.txt files and other methods to give instructions to crawlers.
- Not overloading the website (obey Crawl-Delay).
- Not consuming too much bandwidth.
- Identifying who they are and who they belong to (“Googlebot”, “Baiduspider”).

The first point is very important. The developer uploads a robots.txt file to tell the person who want to search information which information they are allowed to get and which is not. It can give different allowable limits to different groups. There are a lot of rules to upload robot.txt file, such as it must be in the root directory. There are another two polite. The first one is ignoring pages that have <meta name=”robots” conten=”noindex”/>, the other one is get away from any links from pages that have <meta name=”robots” content=”nofollow”/>

Because there are dynamic environment and static environment, so the crawling system must decide when should re-read the document to get the newest one. There are some challenges for crawling. For example, a lot of pages try to refuse crawling. In addition, it is difficult for compute to get the words from image. Last, but not least, one challenges is that the web is too big.

If we want to know the concept of adversarial IR, we need to understand Search Engine Optimisation (SEO) which involves taking steps to improve the ranking of result sets. There are two ways of techniques of SEO, white hat and black hat. White hat is operating ethically and following search engine rules and guidelines to get higher rankings, and black hat is violating search engine guidelines and rules to try and get an artificial advantage: Adversarial IR.

There are six ways of Adversarial IR. The first way is Meta Tags that means the develop can use <meta> tags to describe their contents. It can let the information be the description, keywords of the page. The second way is hidden text. A good way to hide text is to adjust the color of the text to match the background. It can increase the term frequency, and add extra keywords. The third way is cloaking which means let the user and crawlers see different contents. The fourth way is named sneaky JavaScript. It uses <noscript> tag to let the user or crawler who cannot run JavaScript see the content. The fifth way is exploiting PageRank. The developer can create or use 'link farms' to improve PageRank. The page also can allow the third-part commenting are abused to boost PageRank. The last way is Doorway Pages which means using doorway pages to optimize the particular search query, and that doorway pages will let the user go to the same destination. This is only for the high rank.

Lab/assignment work:

This week, I finish the evaluation part. That means I finish my assignment.

This part is very similar to the lab7. It is not very difficult. For the small corpus, it need not consider the unjudged document, it is easier. Before we can iterate through the result set and calculate the relevant Matrix, we need to read the qrels file to get the number of relevant documents and how relevant each document is (It is not necessary, because all of the document in this file is relevant, and we need not to know the accurate relevance to calculate NDCG. Then we traverse each query, get the result set in each query and traverse the result set, calculate the number of relevant documents in each stage in the result set (the tenth, the RTH, and the last), and also ask for the MAP and bpref of each stage and sum them. Finally, we can calculate the results according to the formula. When we calculate the result set, we can write the output file

The different between large corpus and small corpus is about how to deal with the bpref, and the way to get the number of relevant documents. For the large corpus, we need to read the qrels file and judge if the relevance is 0. If the relevance is 0, it is not relevant document. When calculating bpref, we also need to counter the non-relevant documents.

In this assignment, the problem I finally solved was how to use the command line to realize the operation. I had never touched similar requirements before. After asking my classmates, I learned that sys.argv could solve this problem. It is a list of store the path of read the coding file. The argv[0] is the name of python file, so we need to use them from argv[1].

Reflections:

Adversarial IR is not a good thing for IR. So, why we need to learn it. I think there are two reasons.

The first reason that Adversarial IR is useful to avoid the crawler. Adversarial IR is designed to allow machines and people to recognize the content of the website differently. This allows the machine to misjudge the content of the site and give the wrong crawler results. As for the way of avoiding crawler, a very important thing is avoiding multiple access over an IP period. Therefore, a very effective method is to prohibit IP multiple access. In addition, adding human-computer interaction can also prevent crawlers.

The second reason is that I need to know how to improve our IR system according to solve adversarial IR problem. Jansen mentioned that it is very important to avoid cheating by adversarial IR [19]. For today's information retrieval systems, clicks are money. Adversarial IR to get high ranking scam click rate, is scam money, so must effectively curb it. Jansen mentioned 5 ways to avoid it [19]. The first way is automated and human filter. More sophisticated automated filtering and data mining techniques can be effectively upgraded automated and human filter. The second way is pay-per-action paradigm, which is a shift in paradigm from pay-per-click to pay-per-action. However, this approach is not a technical solution. The third way is block blacklisted IP addresses. There are various blacklisted IP databases, such as those that maintain IP lists of known E-mail spam sites. Search engines can take these IP's and deny them access. The fourth way is aggressive monitoring of click fraud perpetrators. This is not the technical solution either. The last way is search engines must make efforts to ensure trust. It's not a concrete plan either. The methods are similar to those used to avoid crawlers, with the exception of some ineffective ones. This is why we learn crawler.

Week 15（beginning June 7th 2021)

Lecture Topic: Overall experience

What was my overall learning experience like this semester?:

This term I learned a lot of things about IR. The knowledge is mainly concentrated in the following aspects, traditional information retrieval model, evaluation and some of the processes in modern IR.

The traditional information retrieval model is the most important knowledge in my opinion, because other knowledge is based on the traditional information retrieval model. There are four kinds of model I learned, Boolean model, Vector space model, Probabilistic model and BM25 model. There is no order in the result set of Boolean model, and the result set in other model is in order. The rank of result document is a very attribute for evaluation.

The evaluation part is also important, which is the key to judge whether an IR system is good or not. In the class, we learned recall, precision, MAP, R-precision, Bpref and NDCG. The first four points are applicable to a wide range of principles and are simple to understand. However, it is not good at handling more details and has some deviation from reality. In fact, some documents are unjudged, bpref can solve this issue. In addition, the relevance degree of documents is not the same in some situation, and NDCG can solve this. It is important to note that a system is evaluated by multiple queries against the same corpus, rather than processing against multiple corpus.

As for the modern IR part, it has 5 parts, PageRank, fusion, relevance feedback, query expansion and web IR. I think this is an extension and upgrade of the traditional IR system. Compared with the traditional IR system, the precision and recall of modern IR system are both increased. At the same time, modern IR systems begin to pay attention to content other than the main content, such as link and title of websites.

What things (if any) were better about online learning compared to the usual face-to-face classes?

I can review the knowledge whenever I want, that is important for me. I also can adjust my learning speed by myself.

What things (if any) were worse about online learning compared to the usual face-to-face classes?

I think all of the class are good.

Did anything happen in other classes that you think might have made COMP3009J better?

I think COMP2009 is one of the best classes I have learned. It is very good.

Is there anything about this semester that you would like to see continue, even when face-to-face classes resume in the future?

I think the transitional chapters in the course design are very important, such as IR pipelines and Modern IR. This kind of course can help me master the structure of the course, and let me know what will I learn. In addition, I think it is very important to use the discussion part in BS.

Any other opinion you have about your experiences this semester.

There are two experiences in this semester are very interesting. The first one is about writing learning journal. At first, I think the learning journal is only used to record what I learned in the class. Therefore, in the early stage of this semester, there were a lot of lessons in my weekly Learning Journal. In the middle and late period, I reduced the writing of the content in class and added the content of Lab, feeling and extracurricular expansion.

The second thing is about the help you give me outside class. In addition to answering the discussion and the questions in the email, the paper you wrote is also of great help to me. For example, I do not understand the sliding windows in the probabilistic data fusion part. Your article called *extending the probabilistic data fusion using sliding window* helped me understand the sliding window [20]. For the window boundaries and ranking score, I have deep understanding. It really helped me a lot.

Reference:

[1] Croft, W. B., Turtle, H. R., & Lewis, D. D. (1991, September). The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 32-45).

[2] Turtle, H. (1994). Natural language vs. Boolean query evaluation: A comparison of retrieval performance. In *SIGIR'94* (pp. 212-220). Springer, London.

[3] Zhijun Cui (2015), The nature of indexing in information retrieval, https://my.oschina.net/zjzhai/blog/464446.

[4] Pugh, W. (1990). Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, *33*(6), 668-676.

[5] https://www.runoob.com/python3/python3-tutorial.html

[6] Ceshine Lee, [NLP] Four Ways to Tokenize Chinese Documents, https://medium.com/the-artificial-impostor/nlp-four-ways-to-tokenize-chinese-documents-f349eb6ba3c3

[7] https://github.com/fxsjy/jieba

[8] https://blog.csdn.net/smartgay/article/details/1757927

[9] Salton G, Wong A, Yang C S. A vector space model for automatic indexing[J]. Communications of the ACM, 1975, 18(11): 613-620.

[10] Lan, M., Tan, C. L., Su, J., & Lu, Y. (2008). Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence*, *31*(4), 721-735.

[11] Trotman, A., Puurula, A., & Burgess, B. (2014, November). Improvements to BM25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium* (pp. 58-65).

[12] Powers, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061.

[13] Saracevic, T. 1996. "Relevance reconsidered '96". In P. Ingwersen and N. O. Pors. Information Science: Integration in Perspective. Copenhagen: Royal School of Library and Information Science.

[14] Xing, W., & Ghorbani, A. (2004, May). Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.* (pp. 305-314). IEEE.

[15] https://blog.csdn.net/hguisu/article/details/8013489

[16] Towell, G., Voorhees, E. M., Gupta, N. K., & Johnson-Laird, B. (1995). Learning collection fusion strategies for information retrieval. In *Machine Learning Proceedings 1995* (pp. 540-548). Morgan Kaufmann.

[17] Jia Lingling, & Dengjian. (2014). The influencing factors for information retrieval fusion. Knowledge of library and information, 000(005), 81-90.

[18] Li Pei. Information Fusion Model in Information Retrieval. Library Work and Research (6), 3-7.

[19] Jansen, B. J. (2006, August). Adversarial Information Retrieval Aspects of Sponsored Search. In *AIRWeb* (pp. 33-36).

[20] Lillis, D., Toolan, F., Collier, R., & Dunnion, J. (2008, March). Extending probabilistic

data fusion using sliding windows. In European Conference on Information Retrieval (pp. 358-369). Springer, Berlin, Heidelberg.