Information Retrieval Programming HW1

R09944019

April 12, 2021

1 Describe Vector Space Model

There are two important factor of Vector Space Model.

The first is the dimension of query vector and document-term matrix for memory footprints and computation overhead. There are roughly thirty thousand vocabularies and forty seven thousand documents in the model, so it will be a 30,000 * 47,000 matrix if I directly build the matrix in uni-gram fashion. It is a heavy overhead for both memory and computation. In order to fit the time restrictions, for query vector, I only use the contents of query file as the dimension of query vector, and import package "Jieba" to break chinese sentence into vocabularies. Furthermore, I perform the combination of uni-gram and bi-gram fasion based on the result of Jieba after removing stop words. Finally, the dimension can be reduced from 30,000 to roughly 200 which is dependent on the query file. It also mitigates the memory footprints and the time of computing. By the way, I use all of the tags in query file except the end of "narrative" since the end of "narrative" contains the irrelevant words. On the other hand, the dimension of document-term matrix is (the dimension of query) * (the number of documents). The number of row of document-term matrix is the same as the dimension of query vector due to execute matrix multiplication between query vector and document-term matrix. The number of column of document matrix is the same as the number of total documents in order to calculate the similarity between query and each document.

One other important factor of vector space model is how to give the weight of both query vector and document-term matrix. I've tried many weighting methods including directly using term frequency in each document or query, adapting term frequency normalization, considering the document length for document length normalization, and LSI method. The results of experiments of the methods will also be showned in experiments section.

Finally, the best result of the method I tried adapts the following weight rule:

$$\log(\frac{(N-df+0.5)}{(df+0.5)}) * \frac{(k1+1)*tf)}{(k1*(1-b+b*\frac{doclen}{avgdoclen}+tf)} \tag{1}$$

This is OKapi BM25 and the parameters $k1$, b are 1 and 0.75 respectively. The dimension of query vector and the row of matrix is the number of the result of the combination of uni-gram and bi-gram fashion based on the result of jieba after removing stop words for the balance between time restriction and similarity performance.

2 Describe Rocchio Relevance Feedback

I use the Pseudo Relevance Feedback which is a blind relevance feedback method. The manual part of the feedback is automated in this Feedback process so user gets the improved results. The system finds the relevant documents and assumed that the top k documents that ranked are the relevance feedback. Based on the rule of Rocchio Relevance Feedback:

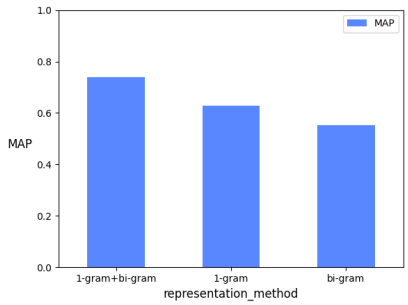
$$q_{new} = \alpha * q_{old} + \frac{\beta}{|D_r|}\Sigma(d_j) - \frac{\gamma}{|D_n|}\Sigma(d_j) \tag{2}$$

The parameters α, β, and γ are 1, 0.7, and 0.4 respectively. The reason why assigning smaller γ value than β is that the relevant vectors are more important than the irrelevant documents in the Pseudo Relevance Feedback.

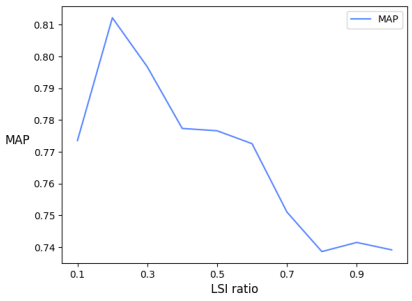
3 Experiments

3.1 Representation Methods for Chinese Texts

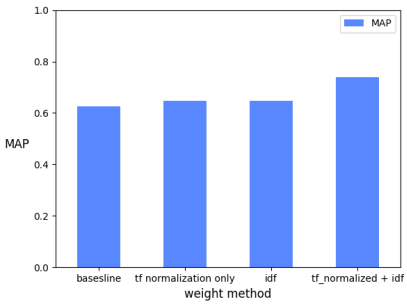
According the provided file - 'inverted-file', there are three representation methods including 1-gram, bi-gram, and the combination of 1-gram and bi-gram. The figure (a) shows the respectively MAP value under different representation methods. We can see that the combination of 1-gram and bi-gram gets the better performance than others because the higher dimension makes the similarity more precise. Under the time restriction, it is the best option for me.



(a) Representation method



(b) LSI ratio



(c) Weighting method

3.2 Latent Semantic Indexing

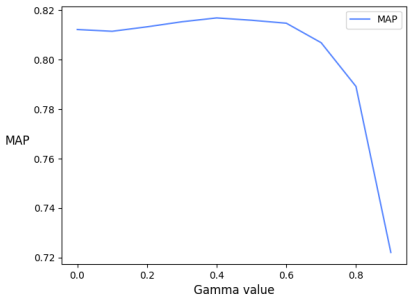
Latent Semantic Indexing is a method used to smooth the document matrix which can extract the latent semantic of document. The method is implemented by pruning a part of unimportant eigenvalues to produce low-rank document-term matrix. Since the number of eigenvalue is depending on the dimension of query vector, the number of pruned eigenvalue should be different according to different query. Instead of using fixed rank document-term matrix, I adapt ratio-based document-term matrix which defines a ratio that represents how many ratio of eigenvalue should be remained to reconstruct document-term matrix. The comparison between different ratios is showed in figure (b). The best ratio is roughly 0.2. Lower ratio makes eigenvalues remain too few that is hard to extract latent semantic. Higher ratio makes the matrix not smooth enough.

3.3 Methods to assign weights

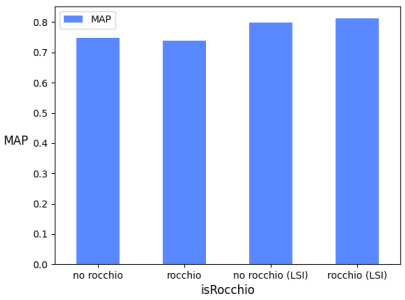
I tried many weighting methods which are used to assign the weights of document-term matrix including directly using term frequency in each document or query, adapting term frequency normalization, considering the document length for document length normalization, and combination of some of above methods. The MAP values in different assigning weight methods are showed in figure (c). I get the best result with the combination of tr-normalization and idf.

3.4 Relevance Feedback

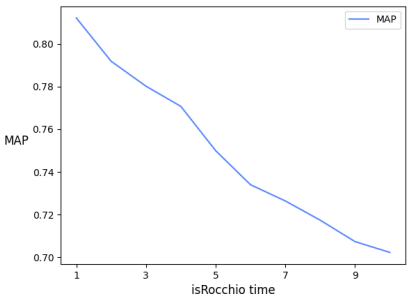
Figure (d) and Figure (e) shows the MAP values of different γ and the result of using Relevance Feedback and no Relevance Feedback respectively. The former shows that the MAP decreases dramatically when the gamma value is larger than 0.6 and the later shows rocchio improves performance only when performing LSI simultaneously. The reason is that Pseudo Relevance Feedback method is dependent on the accuracy of original system. Furthermore, it considers top k documents as relevant documents and others are considered as irrelevant documents. It contains the problem that it may label irrelevant documents as relevant documents and label relevant documents as irrelevant documents. Therefore, if the system misses predictions, the query will adjust in wrong direction. Figure (f) shows the MAP values of the number of running rocchio algorithm. The best number of running rocchio algorithm is 1. I consider that the reason of poor performance with higher running time is that the original system is not good enough to use pseudo relevance feedback duplicately since it is dependent on the accuracy of original system.



(d) The affect of gamma value



(e) Relevance Feedback



(f) Time of Rocchio

4 Discussion

Using Latent Semantic Indexing to improve the system should consider the rank of original document-term matrix. If the original rank is few, the LSI method may hurt the performance of system. Besides, Rocchio algorithm is also sensitive to the rank of query vector. Because more ranks make it able to update query in more factors which improves the system. Last but not least,

the problem of Pseudo Relevance Feedback deeply affects the performance of this homework, it should be used when you ensure that the system is good enough.

5 Imported packages

- numpy: used to execute matrix computation and use function ‘np.linalg.svd‘ for implementing LSI.
- jieba: use function ‘jieba.cut‘ to break chinese sentence into vocabularies.
- tqdm: used to show the progress
- argparse: used to set option for the program
- untangle: used to parse csv into friendly using format