Optimal Query Recommendation Model

Kai-Wen Cheng
r08922015
r08922015@csie.ntu.edu.tw
Learning-based method (BERT)

Cayon Liow
r08922148
r08922148@csie.ntu.edu.tw
Shorten-Content Method (TF-IDF)

Yu-Jia, Liou
b06902008
b06902008@csie.ntu.edu.tw
Smoothing method (LSI)

Bo-Wei, Huang
b06902125
b06902125@csie.ntu.edu.tw
Smoothing method (LM & LDA)

October 3, 2020

1 INTRODUCTION

Usually, there is huge user gap between user space and query space in the information retrieval system. User is impossible to know what term is the best query to find out the documents he wants. Therefore, our goal is to design a Optimal Query Recommendation Model. By giving a document, our system will recommend an optimal query such that the given document can be ranked as top relevant documents by the retrieval model.

Our codes: https://github.com/kevinatntu/Optimal-Query-Recommendation-Model/

2 RELATED WORK

2.1 Vector Space Model

There are multiple types of Vector Space Model(VSM), with different types of vectors, similarity strategies and so on. However, we try to connect our work with Programming Homework 1 in class; hence, there are some default settings in our VSM.

Instead of using different types of vectors, we can only use term-unit vectors. The weights inside are calculated by Okapi BM25 by default. Yet in smoothing and LM topic, the weights are customized. As for similarity, inner product, instead of cosine version, is our method because both Okapi and language model are normalized by document length.

2.2 LATENT SEMANTIC ANALYSIS

TF-IDF is a common weighting factor when it comes to term Vector Space Model(VSM) for ad hoc retrieval. However, it assumes independent term occurrence, which fails to handle absent but core words. Therefore, some resort to Latent Semantic Analysis(LSA), reducing vectors into latent space, ignoring small topics and finding their similarities.

In contrast, we have a restriction on our VSM – it must be implemented in term space – and we also want to select keywords directly from term vectors for efficiency. Thus, we regard LSA as smoothing approach instead of reduction method, reconstructing the term-document back after reduction and topic removal. To do so, the term relationships are added to our term vectors and our goals are satisfied.

2.3 LATENT DIRICHLET ALLOCATION

Latent Dirichlet Allocation(LDA) is another common reduction method. Roughly speaking, it picks a topic distribution for each document and a word distribution for each topic, both of which are generated by Dirichlet distribution.

Plenty of researches take the topic distribution of each document for text analysis or dimension reduction. However, just like LSA, we are proposed to build term vectors. Thus, we construct a topic-related language model for each document with (1). Because of its probabilistic basis, it can be easily combined with maximum likelihood estimated language (MLE LM) (2)[7], creating term vectors with not only frequency but also topic concepts.

$$P_{LDA}(w|\theta_d, \alpha, \beta) = \sum_z P(z|\theta_d, \alpha) \cdot P(w|z, \beta) \quad (1)$$

$$P_{LM}(w|d) = \frac{c(w;d)}{|d|} \quad (2)$$

2.4 BERT

Above methods are using probability or statistic to retrieve the relevant documents. Here we want to add some learning-based model to do the comparison. Nowadays, using a pre-trained deep learning model based on a large number of corpus to fine-tune the downstream task is a trend in NLP field. This kind of model can be regarded as a complicated, entirely data-driven language model. Among those pre-trained structures, BERT [1] might be the pioneer. It has been proved to achieve outstanding effect in a variety of NLP topics, including language-understanding, question-answering and sentiment-analysis. However, its performance depends on the downstream task, too. We wonder

whether BERT can generate a more fluent, sentence-like optimal query for the target document.

3 METHODOLOGY

There are some assumption for the whole system:

- The concepts of relevance documents should be highly similar to the concepts of their query.
- The retrieval model is a Vector Space Model (VSM)[6] [5] and must be implemented in term space.
- Query vector is binary.

We know that the best query to retrieval the specific documents in the VSM model is the whole contents of the documents but it is not instinct for human. Therefore, there are some constraints for the query generated:

- The queries generated should be semantic and human-understandable.
- Ground truth queries should be selected properly.

Several methods are designed to generate the queries:

3.1 SHORTEN CONTENT

As we know that the whole contents of the document is the best query for the retrieval model, we can exclude some unimportant words from the contents to shorten the query.

These unimportant words can be stop words and repeated words. The top 100 frequent words in the whole corpus are removed as stop words. And the repeated words in the contents are removed from the query directly.

Although the length of query is now almost half of its original length, it is still a long sequence of words for human. We decided to remove the words by calculating the weight of term frequency–inverse document frequency(TFIDF)[3]. The value of TFIDF of every words in the query are calculated and sorted in ascending order. The words with smaller TFIDF value are removed one by one and finally the last word in the query is the words that having highest TFIDF value in the contents.

3.2 SMOOTHING STRATEGY

When using TF-IDF[6] to represent document vector and calculate score[5] with query vector, we can see that if the query contains a word that doesn't appear in the document, the word will not make the score higher. However, sometimes the user will type some words not in the document but has similar meanings, just like 'ship vs boat', 'good vs great'. So, we tried to use LSA[2] to smooth the origin TF-IDF matrix. After decompose the origin matrix with SVD, we'll get an approximate matrix. We then use the new matrix to generate our document vector and use dot product with query vector to calculate the score. Thus, the words not in the document may be selected when generating queries.
In the experiment, we will set different latent dimensions to see how it will affect the query we generate. We expect that the higher dimensions we use, the words are more weird or are proper noun. In contrast, the lower dimensions we use, the optimal query contains more 'concept' words and more readable.

3.3 LANGUAGE MODEL

We try to pick keywords directly by the ranking of MLE LM just like TF-IDF method for each document. Also, we remove 50 words with highest document frequency to get rid of stop words and frequent words.

Note that, this method has the same effect as considering term frequency only. Yet in many cases, frequent words are professional terms and hard to include the main idea of the document. Therefore, similar to LSA, we implement Latent Dirichlet Allocation (LDA) for topic-smoothing extension on the language model. After fitting the whole dataset into LDA model(topic number = 32), we can construct a topic-related language model according to (1), and then find the keywords with highest probability.

However, in the experiment part, we find out that most keywords selected by LDA model are coarse topic words, such as *images, models*, and so on, making poor performance in our task. Thus, we combine LDA model with MLE LM to strike the balance between their pros and cons.

$$P(w|\theta_d) = 0.5 \cdot P_{LM}(w|\theta_d) + 0.5 \cdot P_{LDA}(w|\theta_d, \alpha, \beta) \quad (3)$$

In this case, term weights are assigned as $log(P(w|\theta_d))$, and the inner product of query vector and document vector for similarity can be equivalent to product of probabilities.

3.4 BERT

For learning-based model, we generate and choose the optimal query based on the latent similarity generated by BERT. First we fine-tune the BERT on our dataset, and the task is thesis type classification. In this way the distant of similar papers in the latent space will be closer, which help improve the mAP scores. Next, we randomly sample specific number of words from original document as query candidates. Due to high cost of sampling all possible combination, we instead sample 10000 candidate at a time, and repeat several times to calculate the average as result. Finally, we compare the similarity between each candidate's latent and original document's latent by computing their mean-square error (MSE), and select the one with minimum MSE as this document's optimal query.

In details, we use the Bert-large-cased pretrained model from HuggingFace [8] repository to do the downstream training task. It takes about 7 hours to converge, and the final classification accuracy is 73%.

4 EXPERIMENTS

The dataset with 7000 article from TBrain is chosen for the experiments. Here we test our four methods with four benchmarks - distribution of target document's ranking, mAP of a set of documents, user-friendly and readability, and query generating time.

4.1 DISTRIBUTION OF TARGET DOCUMENT RANKING

All the methods are asked to generated 5-words optimal query for random 100 documents over the whole 7000 documents and the rankings of respective documents from its queries are observed.

For the shorted content method, the ranking of respective documents are always 1.

For LSI part, different dimension will get different ranking distribution. In the table below, we can see that the higher dimension we use, the higher rank 1 percentage we get.

Rank 1(%) in different dimension(LSI)

Dim	100	500	1000	5000
rank1(%)	13	80	94	100

For language model, different ratios between MLE LM and LDA model can cause different results. When using MLE LM only, 100% documents reach top 1. When using LDA model only, 2% documents get top 1. In fact, the poor performance of LDA tells us although topic words are highly related to documents, they are hard to be regarded as optimal queries. Thus, we decide to combine MLE LM and LDA model (1:1), leading an acceptable result on performance(99% top 1) and more reasonable queries simultaneously.

For BERT method, only 65% documents get rank 1, and 76% are in top 5. The performance is much worse than other methods.

4.2 mAP OF A SET OF DOCUMENTS

In this part, we assume that the user want to retrieve a set of similar document by one optimal query. Here we choose 5 documents, which are all mentioned "YOLO" [4] in their abstract. Then we calculate the mAP of these documents in the whole corpus. This benchmark tests whether the optimal query can simultaneously achieve high ranking for each documents among the same set.

For shorten content, we directly generate document id D01350's optimal query without considering other YOLO documents and use to rank all documents, and the result is shown below:

Stage	mAP
Whole content	0.2045
Remove stop words	0.3001
Remove repeated words	0.4230
Remain top 5 highest TF-IDF words	0.4318
Remain top 1 highest TF-IDF words	0.2019

From above, we can know that queries with top 5 highest TF-IDF words get the best mAP, as the words is important and also maybe appear in other relevant documents.

mAP in different dimension(LSI)

Dim	100	500	1000	5000
mAP	0.02492	0.40340	0.40403	0.40650

For LSA, the table above shows that the mAP under different dimenion with query length equal to 5. Since that we will get more conceptive words if we use the lower dimension, so the score will be lower. And we also find that using different documents in same topic(e.g. with keywords 'yolo') to generate query will also get different mAP, but the trend is the same.

mAP for LMs (given document D01350)

Keyword num	10	5	1
MLE LM	0.200	0.201	0.206
Hybrid	0.202	0.204	0.203
LDA	0.008	0.108	0.168

For LM models, the queries generated by only D01350 has worse performance than TF-IDF method. The main reason might be LMs don't consider IDF explicitly, which filters some accurate keywords for specific domains of documents, such as *yolo*.

mAP for LMs (given 5 yolo documents)

Keyword num	10	5	1
MLE LM	0.804	0.701	0.133
Hybrid	1.000	0.440	0.134
LDA	0.005	0.108	0.006

On the basis of probability, LM series can be extended to generate one query given 5 yolo documents; that is, $P(w|5\ \textit{yolo documents})$. That leads us to observe the ability to summarize a chunk of documents. As the table shows, hybrid model improves MLE LM when there are 10 keywords, indicating LDA works when we pull appropriate amount of topic words into query. Also, when evaluated by VSM model, LM series overwhelm BERT method, which means counting-based model is more suitable for VSM.

For BERT method, we construct the optimal query by calculating the sum of MSE between each query candidate and total 5 YOLO documents. Also, we normalize the scores for fear that single document might dominate the MSE.

The mAP is not very high, either. mAP is at most 0.2 when the max query length is 5. In fact, this is not beyond our imagination, since the task we use to fine-tuned our BERT is just a rough classification. Therefore, it is hard to perfectly distill the detailed category, like YOLO, from the whole corpus. We also find that most top-retrieved documents are in computer vision and reinforcement learning fields, but did not mention YOLO strategy in their abstract. It shows that to some extent BERT can achieve the general clustering ability.

4.3 HUMAN-READABLE QUERIES

For our query recommendation model, generating human-readable queries is an important benchmark to qualify our model.

For the shorted content method, the quality of query is not maintained since it depends on the words in the whole

content. [5] For example, the query *'generalisation hippocampal entorhinal generalise memories'* is expert terminology but the query *'contextualized ckr sparql rdf inferences'* is not instinct for human.

For LSI method, the query is more readable if we use lower dimension. For example, when we set dim=100, we generate query likes *'object recognition detection neural images'*, which is instinct for human. When we set dim=5000, we generate query likes *'pepper yolo card robustly tk1 320x320'*, which contains some numbers, device type and proper noun.

For MLE LM, the keywords might contain too many high frequency words or even professional terms, such as *adversarial, yolo, alpr, detectors*, etc.; whereas, the queries generated by LDA contain more familiar topic words, like *image, object, detection, algorithm*, etc. However, these topic words are too coarse to construct an optimal query. Therefore, we combine MLE LM and LDA to generate a query with *adversarial, image, object, yolo*, taking care of both performance and readability.

For BERT, it clearly demonstrates BERT's characteristic in this benchmark. Not only do most optimal queries contain keywords of corresponding documents, but they are natural and meet the rule of a normal sentence. For example, for the paper *"Integrating Task-Motion Planning with Reinforcement Learning for Robust Decision Making in Mobile Robots"*, the optimal query is *"by corresponding planning-learning executed RL"*, which comprises both keywords and POS structure.

4.4 QUERY GENERATION TIME

The generation time of query is important for the model to be convenient and easy to use.

The query generation time of shorten content method is always instant since there is only operation of 1-time TF-IDF calculation and operation of removing word.

For LSI, the query generation time depends on how large dimension we use. It takes about 5 seconds when dim=100, but it takes about 1500 seconds when dim=5000.

For LDA, modeling time is about 1 minute. However, when facing new documents, only 5 to 10 seconds are needed for inference.

As for BERT, the query-generating process is really time-consuming. Apart from the long fine-tuned process, in average it takes 3 to 5 minutes to generate an optimal query for the target document. The large model size and considerable number of query candidates needed to try are main reasons. Users are hardly willing to wait for such a long time.

4.5 Summary of our experiment

Within all method we tried, we can find that TF-IDF and LDA with LM method has the highest score in rank 1, which means that the optimal query will rank the origin document to rank 1. Next, to measure the quality of the queries, the words in LDA and LSI are more conceptive. In contrast, the words selected with origin TF-IDF are quite involved and abstruse. For the readability, BERT method can get a sentence-like query with keywords and concept in it. Finally, all method need little time to generate a query except BERT method.

5 CONCLUSIONS

In this final project, we have designed four model to meet our goal – to generate the optimal query of the target document for the user. Considering that our retrieval model, VSM, is too simple, the TF-IDF strategy is well enough to achieve our goal. The only drawback is that sometimes the query might contain keywords that is too specific for users to know in advance, which makes it not user-friendly. For this issue, the two smoothing and learning-based methods can help. Due to the lack of time, we have not tested in other kinds of retrieval model, but we expect these methods can perform better in more complex retrieval model.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology (ARIST)*, 38:189–230, 2004.

[3] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[4] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[5] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[6] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[7] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, 2006.

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.