Term Project Report

Real Time Adhoc Retrieval

CSC849 Search Engine

Fall 2016

Xuan Zhang

Introduction

Twitter is becoming more popular nowadays, and it is one of biggest social media around the world as well as the primary information publishing and information source recently. It plays a very important role in information retrieval. For example, there are millions of topics every day on twitter. By the means of twitter, we could get the most updated exact information all of the world. Since it becomes more and more important in IR fields, we should find and research the effective ways to do tweets retrieval.

However, microblog are different from traditional information retrieval. Mainly because the following aspects:

First, tweets are very short. Based on the regulation of Twitter, each tweet should be less than 140 words, which is very different from traditional documents.

Second, twitter provides a large volumes of data in real time. Every seconds there are millions of people updating the new tweets.

Third, very short tweets has very distinct features—that they have pretty much standard structures, such as normal text, URL, hashtags and so on, which is very different from traditional documents.

Therefore, we have to find new effective methods and research them to get very adaptive ways to deal with microblog retrieval.

Now I selected two papers which I think could represent the some effective ways to deal with microblog retrieval.

Related Works:

1. Improving Twitter Retrieval by Exploiting Structural Information [1]

One of paper I selected is Improving Twitter Retrieval by Exploiting Structural Information.

Though tweets are very short, they have very obvious limited structures—can be divided into a couple of blocks, such as plain texts, hashtags, links, mentions (Ahnizer et al. (2004) [3]
). Besides, the sequence of these blocks also reflect the quality of the tweets. A same term can behave differently in particular blocks.This paper utilize the structure information to achieve higher performance in tweets search.

For example, like figure1, we can see that Yao's tweets are only plain texts. BBC News are messages followed by URL links. And Lady Gaga uses a mixture of hashtags, links and mentions.

A tweet could be divided into different Tweet Building Blocks (TBB), And the distribution of sequence of TBB structures in real world is like:

MSG, MET MSG, MSG URL, OHTERS, COM URL, MSG TAG, MSG URL TAG, RWT MSG, TAG MSG, TAG MSG URL, RWT MSG URL, COM RWT MSG, MET MSG URL, MSG MET MSG, RWT MSG TAG and so on.

In their work, they followed a related work Duan et.al(2010) [2] which gives the best published results on the Twitter search task. In Duan's work, they found out that the tweets with links (URLs) are the most effective feature for

tweet retrieval. And the paper I chose continued to study which TBB structure (or in another word, which kind of sequence of blocks) has high quality and might be highly relevant to the query. Because the different TBB structures of tweets has different search quality. Such as BBC news may be pretty much more formal, informative and relative than a normal tweet sent by Lady Gaga.

In their work, they choose 2000 samples of English tweets, using other toolkits to do TBB classification and TBB boundary detection in order to tokenize terms in different blocks. Then they crawled and indexed 800,000 tweets, and get high MAP value.

The conclusion is that all their results suggest that structural information of tweets can improve search. And TBB structures containing the “URL” block are highly valued for ranking, especially the tweets with “MSG URL” structure are more likely to be relevant than other structures with URL.

Their work shows that although tweets are very short, their structural information can improve microblog retrieval.

2. USC/ISI at TREC 2011: Microblog Track [4]

The second paper I chose was USI/ISI at TREC 2011: Microblog Track, which mainly use 3 approaches:

1. Markov random field model (MRF) (full dependence model)
2. Pseudo relevant feedback (PRF) using Latent Concept Expansion (LCE)
3. Learning-to-Rank model

They did 4 official runs using those different combinations of methods above.

In their experiments, they first download the dataset and used text scoring, which they make use of Markov random full dependence model (MRF),

which assumed that all terms are dependent on each other. And this is especially better for ranking short documents like tweets. And then using Indri to get the top query results to do the query expansion.

The second step is to do the query expansion which they make use of Latent Concept Expansion (LCE), which is a PRF technique.

After that, they use leaning-to-rank model to do the query search. In this paper, they use 8 features. At the end, text_score was found to be the best feature to help tweet retrieval, and other very good features include has_url in Duan et al, which is also the first paper referenced based on the structure information of tweets. For each query, they got 30 results.

Therefore, their results showed that both learning-to-rank and pseudo relevance feedback methods were effective in tweet retrieval.

My Approach:

Analysis of the Given Material and Find the Benchmark for this Project:

First, I analyze the given material in general when I build the inverted index for the whole folders. There are 7 folders in the given materials: 20110123, 20110124, 20110127, 20110128, 20110129, 20110131, and 20110201 which represents 7 days in microblog2011. Actually, it should be 17 days from January 23 to February 8 2011 seventeen days in Trec microblog 2011. So, the given material is not complete and doesn't cover all the

documents, and not all the relevant documents included in the given folders.

In the feedback file which will help to evaluate the result “qrels”, there are all the relevant documents for 110 queries. The format for each line is like:

1 0 34952194402811904 0

or

1 0 32080604065431552 1

The last integer “0” represents non relevant whereas “1” represents relevant. However, when I use program to get only relevant lines, I find out that there are 11 queries which don’t have relevant record in the file (Query 11, 12, 15, 18, 24, 49, 50, 53, 63, 75, and 76).

Besides, when I test the characteristic of the given material, I extracted all the true relevant documents from the 7 folders, and found out that there are 5 queries that have relevant records, however their relevant records are not in the give material (Query 30, 55, 58, 66, 80). So, the effective queries should be 110 - 11 - 5 = 94 queries. When I use Indri to get the results, only the 94 queries would be effective, So I modified the feedback file “qrels” and remove those noneffective queries in order to get a more accurate evaluation.

And I extracted all the relevant records from the given materials. There are total 1650 relevant feedback tweets in the given materials for those 94 queries. On average, 1650 / 94 = 17.55 relevant feedback tweets per query. The distribution of number of relevant feedback for each query could be found in my file CountRel in the code and result file part. Actually, the median value is 12 relevant tweets per query. (#of relevant <= 11: 46 queries; #of relevant =12: 4 queries; #of relevant >12: 44 queries). In

other words, more than half of the queries don't have the true relevant result larger than 12. If we use P@30 as the benchmark, even if there is a "perfect" search engine which make the top return all true relevant, more than half of the queries evaluation would be still less than 12/30= 40%.

# of true relevant for the query	<= 11	12	> 12
# of queries	46	4	44

So , due to the limited resource in this project, I think P@10 or P@15 more meaningful than P@30, though many paper prefer P@30 since their dataset is complete.

Know the Indri Limitation:

Firstly, I want to test if there is a "perfect" search engine, what is the evaluation limitation. That is the system limitation. So I extract all the relevant documents for each query. And select the top 20 words (if the number of relevant documents is smaller than 20, then the number of expanded words depends on the number of relevant documents). The evaluation result is as below:

	Original	Ideal Case
P@5	0.3830	0.8191
P@10	0.3511	0.6319
P@15	0.3170	0.5340
MRR	0.6326	0.9840

Definitely in the formal run we could not make the top returns are all true relevant document. This is not the real case. The formal run later would not

be better (and actually much worse) than this “best performance” way. But we can see the limitation of using Indri for this project’s noncompete given documents.

My Formal Run:

Tools:

I use Indri-5.11 and Trec_Eval.9.0 and my own programs in Java in this project.

Method:

I do query expansion based on tf-idf score.

For each documents: weight(t, d) = [1 + log10(**tf(t,d)**)]* log10(N/**df(t)**).

(here N = 2704533 tweets in 7 folders)

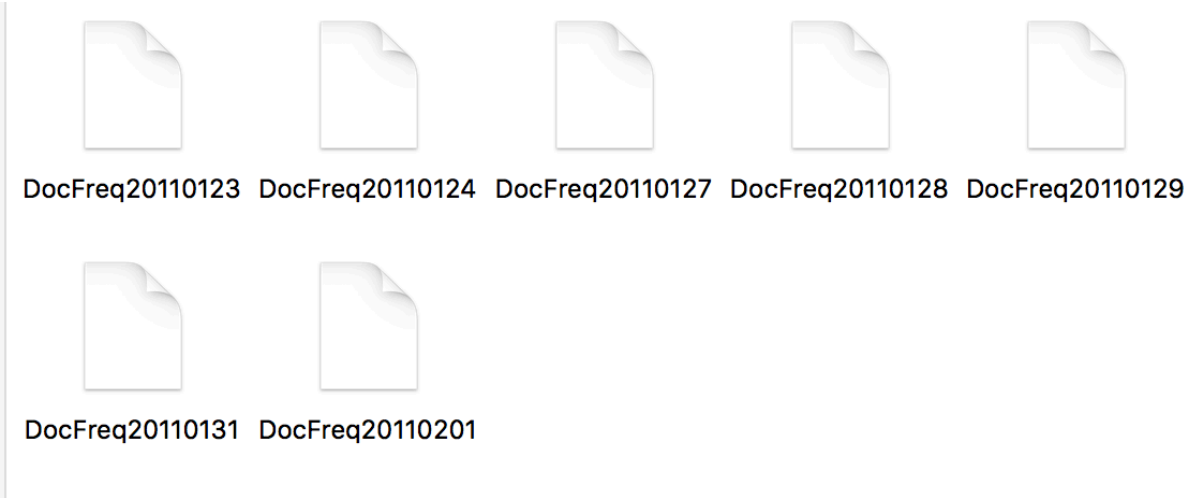
We can see that the term frequency tf(t, d) is related to the Indri top returns (pseudo feedback), but df(t) is related to the all documents, relevant and non relevant documents. So, when I build inverted index, I deal with them separately. (Due to the volume of large file to build the inverted index of the whole files, I use the inverted index only related to the document frequency without position list for each folders. And for tf(t, d) I build separate inverted index with position list for each query.)

Build Inverted Index 1 for all the files to get document frequence:

First, I build the inverted index for all the files. Since it will cover all documents, no matter whether it is relevant or non relevant, and only df(t) should cover all documents, term frequency tf(t,d) is only related to the term in the pseudo relevant documents returned from Indri. So, this inverted index is without the position information. I build the position list in the pseudo relevant documents inverted index (which is inverted index 3).

When I build the big inverted index for all the files, at the beginning I just want to traverse the 7 folders, however, due to the large volume, this method failed because it is too large and it miss a lot of information.

Then, I separate the inverted index. I first build 7 inverted index for the 7 folders respectively. And then I got 7 inverted index file with only document frequence but no position list. Actually, if we want to get the document frequence for a given term, we have to get the 7 inverted index together to get the total document frequency for that term in all 7 folders. And if we want to get the tf-idf of that term in a certain documents, we have to know the position information as well. We will do those jobs later after we get the Indri pseudo relevant returns.



7 Files of inverted index without position list

Using Indri to get the top 10 returns for 94 queries:

Then I use Indri to get the pseudo relevant feedback. I programed to get the param file for the total queries.

For each query: it is like:

<query>

<type>indri</type>

<number>1</number>

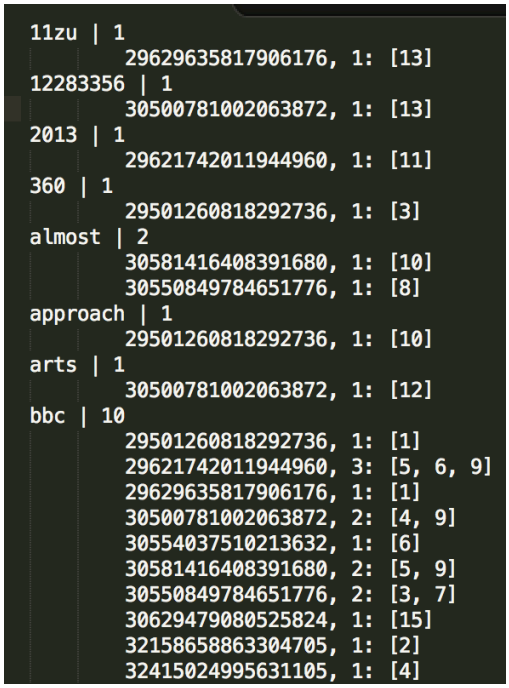
<text>#filreq(#less(requested_id 34952194402811904) #combine(BBC World Service staff cuts))</text>

</query>

I use this way to filter the documents which is after the query date. And for each query, I got 100 returns. And I only use the top 10 returns to do the query expansion. (Why I use the top 10? Because I also test the top 20, but the result is worse. And many paper selected the top 10, and I also select the top 10.)

Build Inverted Index 3 with position list for the pseudo relevant documents for each query.

The inverted index 3 is like below:



Please pay attention, the document frequence in inverted index 3 is only the document frequence in pseudo relevant documents, when computing the tf-idf score, we have to use the document frequence in inverted index 2, which is the total document frequence in all documents.

Then I programmed to extract all these pseudo relevant documents based on the indri score from microblog2011. But how do I know the exact document frequence of a given term? I have to add the inverted index 1's 7 document frequences together. But here I did not directly merge the 7 inverted index files, instead, for each query, only the term shows up in the top 10 returns of Indri need to be calculated the tf-idf score. So for each query, I build an inverted index (or only the document frequence) by traversing the 7 inverted index 1 and add the df for that term together. This inverted index without position list (or document frequence file) is called inverted index 2 for each query.

Query Expansion:

Now we have extracted the pseudo relevant feedback, and we can get term frequence tf(t, d) from inverted index 3 and document frequence from inverted index 2. So we can get the tf-idf score and do query expansion.

Indri Expansion based on top 20:

First I tried to expand 10 terms based on the top 20 returns of indri. However, the result is as below:

	original	Expansion based on Indri returns (top20)
P@5	0.3830	0.3894
P@10	0.3511	0.3617
P@15	0.3170	0.3099

	original	Expansion based on Indri returns (top20)
MRR	0.6326	0.5435

Analysis:

The result is not good. I think because the P@20 itself is only 0.2941, the pseudo relevant results would include many non relevant documents, which leads to the expansion is not good. And if we choose the top 10 Indri returns to do query expansion, the result might be a little better.

Indri Expansion based on top 10

I choose to expand 10 terms for each query since I see some paper prefer this way. Thus, for each query I got the top 10 terms based on their tf-idf score and these terms should not be in the original query. After the expansion, I use trec_eval.9.0 again to evaluate the results:

	original	Expansion based on Indri returns (top 10)
P@5	0.3830	0.4000
P@10	0.3511	0.3559
P@15	0.3170	0.3154
MRR	0.6326	0.5615

Analysis:

We can see the result is not good. The improvement is not obvious, only P@5 improve 4% comparing original P@5, and some other features such as P@15 and MRR even worse than original. I think this is because even in the top 10 returns of indri the precision is not high, only 0.3511. When I do expansion, there is lots of pseudo relevant feedback are non relevant at all.

And this results in that the top 10 terms expanded into the query is non relevant to the topic.

The expansion based on the tweet itself is not good, then I tried to think if I can use external evidence to expand the query effectively.

Test on the External Evidence:

I want to manually test if this could work. For each query, I googled. If there is a wikipedia page in the first search page of google, I use wikipedia. In order to make sure the date was before the query date, I choose the wiki version before the query date. For example, for Query 23, “Amtrak Train Service”, I select the old version: 02:34, 7 February 2011. For Query 36, “Moscow airport bombing” I select the old version: 22:52 24 January 2011.

For the queries if there are not wikipedia page shown in the first google search result page, I choose the first relevant page which was in 2011.

I copied the text part of the relevant page. And then I build a inverted index for those external evidence, and get the top 5 frequently used terms in the external page. (I used the top frequently used terms as in the paper[5].) And then I combine the original query with expansions from indri return as well as the expansion from external. The evaluation result is as the following table:

	original	Expansion based on Indri returns	Expansion based on Indri returns + External Evidence
P@5	0.3830	0.4000	0.3957
P@10	0.3511	0.3559	0.3585
P@15	0.3170	0.3154	0.3106

	original	Expansion based on Indri returns	Expansion based on Indri returns + External Evidence
MRR	0.6326	0.5615	0.5752

Analysis:

From the result, we can see that my way of using external evidence is useless for improving the precision at any level. I guess it is because of the several following reasons:

1. Because I only use the frequently used term in the page not the tf-idf score, some words are very frequently used but not relevant to the query. For example, for query 1, the 5 expanded terms from external evidence is “said foreign language week began”, or in query 16, “rumsfeld bush president we know”, there are a lot of frequently used words but not relevant.

2. Due to the limited pool, for each query, the true relevant documents are very limited, not all the relevant words are shown in those relevant tweets.

3. Some of the expansion terms are already in the expansion terms based on Indri returns. For example, for query 2: the 5 terms expanded from external are “qatar cup world host worker”, but “qatar”, “cup” and “world” are already in the expansion from tweets.

4. I only copied one external page, I think from other paper[5], at least we need download 10 external pages to do expansion. One page for each query is not enough and noneffective from the result.

Conclusion:

Query Expansion from the indri returns based on tf-idf score using my way only has subtle improvement at P@5. And using external evidence of only 1 page for each query is not effective.

I think if I have additional time, my system can be improved by other methods:

Using more proximity operator in the original queries.

Using large amount of external evidence.

References

[1]Zhunchen Luo, Miles Osborne, Saša Petrovic, Ting Wang, Improving Twitter Retrieval by Exploiting Structural Information

[2]Y.Duan,L.Jiang,T.Qin,M.Zhou,andH.-Y.Shum. An empirical study on learning to rank of tweets. In *Proc. 23rd Intl. Conf. on Computational Linguistics*, pages 295–303, 2010.

[3]Ahnizeret, K.; Fernandes, D.; Cavalcanti, J.; deMoura, E.; and da Silva, A. 2004. Information Retrieval Aware Web Site Modelling and Generation. In *Proceedings of the 23th Internacional Conference on Conceptual Modeling*, pages 402-419.

[4] Donald Metzler, Congxing Cai, USC/ISI at TREC 2011: Microblog Track

[5] Zhihua Zhang, Man Lan, Estimating Semantic Similarity between Expanded Query and Tweet Content for Microblog Retrieval