Udel @ NTCIR-11 IMine Track

Ashraf Bah
Computer and Information Sciences
University of Delaware
Newark, DE 19716
ashraf@udel.edu

Ben Carterette
Computer and Information Sciences
University of Delaware
Newark, DE 19716
carteret @udel.edu

Praveen Chandar
Computer and Information Sciences
University of Delaware
Newark, DE 19716
pcr@udel.edu

ABSTRACT

This paper describes our participation in the Intent Mining track of NTCIR-11. We present our methods and results for both document ranking and subtopic mining. Our ranking methods are based on several data fusion techniques with some variations. Our subtopic mining method is a very simple technique that uses query dimensions' items to form a subtopic

1. INTRODUCTION

The same query, when provided by different people, may be intended to satisfy completely different information needs. It is thus important for search engines to incorporate in their rankings coverage of as many intents and aspects as necessary to satisfy many users information need. The retrieval task of the NTCIR IMine track is aimed at addressing this issue.

2. DOCUMENT RANKING RUNS

The 2014 NTCIR IMine dataset for English document retrieval consists of 50 queries for which the organizers provided query suggestions from Bing, Google and Yahoo as well as query dimensions generated using the method proposed by Dou et al. [3]. The task of the participants is to provide a diversified ranking of no more than 100 documents per query for the 50 queries, covering as many intents as possible. The collection on which the retrieval is performed is the ClueWeb12-B13 subset which contains about 52 million web pages crawled in 2012. More detail about the task can be found in the track overview paper [2].

2.1 Udel-D-E-1A

In this experiment, our aim is to leverage query suggestions to improve the diversity ranking. We consider query suggestions from Google, Yahoo and Bing – provided by the track organizers – to be various representations of the original query. We think of them as different formulations that have the goal of satisfying different (yet possibly related) intents. Our approach strives to combine evidence from these different formulations of the original query by using a rank aggregation method that we call CombCAT – a variant of CombMNZ [1]. The idea behind this is to promote documents that cover the larger number of aspects and/or intents.

All of the runs submitted rely on the CombCAT rank aggregation. CombCAT is a variation of CombMNZ wherein we not only take the overall similarity values into account, but more importantly we also account for the frequency of a document. In CombCAT, for each query, we first group documents into different categories such that documents that appeared in n different rankings are put in the same category $category_n$. Then we proceed with our re-ranking by promoting documents that appear in the largest number of rankings. And within the same category, documents that have the largest sum take priority over others. Both CombMNZ and CombCAT explicitly reward documents that appear in the largest number of rankings – though in different ways.

2.2 Udel-D-E-2A

In this experiment, we make use of the dimensions' information provided by the organizers. This run is similar to the previous run in that we use the same aggregation method. But contrary to the previous run, instead of using query suggestions, we use the query dimensions' information. In particular, we consider only the first five dimensions, then we use items that were marked "selected" to augment the original query. Thus each generated related query will be of the form original_query + selected_item where "+" is the concatenation operator, selected_item is an item that was marked "selected" and "original_query" is the original query. For instance the first four related queries for query 51 and 52 will be as shown in Table 1.

2.3 Udel-D-E-3A

This experiment also makes use of the dimensions provided by the organizers. This run is similar to the previous run in that we use the same aggregation method and we consider only the first five dimensions. But contrary to the previous run, we construct related queries by concatenating values of the items that were marked "selected" as well as the original query. Basically for this method the largest number of related queries is only five (and this has an impact on the results, as will be discussed in the results section). In contrast, the number of related queries for the first run (obtained using all the query suggestions from Bing, Google and Yahoo!) is much larger. For instance the first two related queries for query 51 and 52 will look as shown in Table 2.

2.4 Udel-D-E-4A

This run also makes use of query dimensions. In this approach, we first group related queries per dimension. So for query 51 for instance, selected-items from dimension 1 would be the first group of related queries, and selected-items from dimension 2 would be the second group of related queries, and so on. Thus for query 51, we would have 27 groups of related queries since we have 27 different dimensions. However, in our implementation, we limit ourselves to 5 groups. In each group, we aggregate the rankings of the related queries using CombCAT. Then we proceed to the final re-ranking by doing the following (starting with the top document of each of the five aggregated rankings): doc_from_group1 followed by doc_from_group2 followed by doc_from_group3 followed by doc_from_group4 followed by doc_from_group5, and we repeat until we exhaust the list of documents. Doc_from_group1 means a document that comes from the first of our five aggregated rankings, and doc_from_group2 means document that comes from the second of our five aggregated rankings.

2.5 Udel-D-E-5A

This last run is a two-step method. As in Udel-D-E-5A, we start by constructing related queries by concatenating values of the items that were marked "selected" as well as the original query. But in the second step, we use an iterative method that selects the document with the highest estimated utility score in each iteration. At rank 1, we simply select the document with the highest relevance score. From rank 2 onwards, we select the document with highest score using the following formula:

$$ArgMax_{d_k \in S}\left[\sum_{q \in R_q} rel_{d_i,q} * (1-\alpha)^{\sum_{i=1}^{k} rel_{d_j,q}}\right]$$

Where, S represents a set of unranked documents, R is a set of related queries, $rel_{d\_i,q}$ is the probability that document d_i is relevant to the related query *q*. And *alpha* is a parameter that penalizes redundant documents and *k* is the rank of the document.

Table 1. Example queries for run udel-D-E-2A

First four related queries for query 51	**First four related queries for query 52**
apple iphone apple ipad apple apple tv apple ipod	cathedral giving tuesday cathedral annual giving cathedral faith in the future cathedral planned giving

Table 2. Example queries for udel-D-E-3A

First two related queries for query 51	**First two related queries for query 52**
- apple iphone ipad apple tv ipod iphone 4 iphone 3g mac ios os x itunes ipod touch app store iphone 5	- cathedral giving tuesday annual giving faith in the future planned giving student raffle
- apple ipod shuffle macbook ipod touch macbook pro ipod nano macbook air ipod classic ibook imac mac pro apple tv mac mini ipad air ipod shuffle 4th generation	- cathedral sunday saturday

Table 3. Example of subtopics generated for query 51

First-level subtopics	**Corresponding second-level subtopics**
apple iphone ipad apple tv ipod	apple iphone apple ipad apple apple tv apple ipod apple iphone 4 apple iphone 3g apple mac apple ios apple os x apple itunes
apple ipod shuffle macbook ipod touch macbook pro	apple ipod shuffle apple macbook apple ipod touch apple macbook pro apple ipod nano apple macbook air apple ipod classic apple ibook apple imac apple mac pro
apple adam blossom brandy apples	apple adam apple blossom apple brandy apple apples apple aphid apple bee apple green apple alligator apple applaud apple blight

3. Subtopic mining task

For this task, participants are asked to provide subtopics related to the 50 queries. A subtopic may be either an aspect of the original query or a different interpretation of the original (ambiguous) query. Specifically, participants must provide a two-level hierarchy of the subtopics. To aid in that endeavor, organizers provided query suggestions from Bing, Google and Yahoo as well as query dimensions generated by [3]. These are the same resources mentioned in the above section.

3.1 udel-S-E-1A

This is a very simple run that simply uses query dimensions' items to form a subtopic. We consider the first five dimensions only. Each first level subtopic for each query is just: original_query + first_5_items_of_dimension$_i$ where "+" is the concatenation operator, original_query is the original query and first_5_items_of_dimension$_i$ is the concatenation of the top five items in the ith dimension.

Each first-level subtopic has five second-level subtopics each of which is simply: original_query + item$_j$_of_dimension$_i$ where "+" is the concatenation operator, original_query is the original query and item$_j$_of_dimension$_i$ is the jth of the top 10 items of the ith dimension.

For each query, we provided five top-level subtopics and ten second-level subtopics. These cutoffs as well as the number of dimensions items used to form first-level subtopics (i.e. five) did not undergo any tuning. This, as well as the simplicity of the method, leaves large room for improvement.

4. Results and Discussion

Table 4 shows the D#-nDCG results for our runs. Our first run achieved the highest score in terms of coarse-grained results, and the third highest in terms of fine-grained results. The first run is the one that aggregates over query suggestions. The main difference between the first and the second run is that, in the second, we use query dimensions' information instead of query suggestions for our aggregation. This results in a big difference (0.6297 vs 0.3900). This suggests that the query dimensions as they are used here are not as useful for this task as the query suggestions. It would be interesting to see whether a different (maybe more careful) selection of query dimension items may lead to better performance. Another interesting investigation would be the impact that using query dimension in addition to query suggestions would have on our performance.

Table 4. D#-nDCG Results for the 5 runs

Runs	Coarse-grain results (evaluated with first-level subtopics)	Fine-grain results (evaluated with second-level subtopics)
udel-D-E-1A	0.6297	0.5469
udel-D-E-2A	0.3900	0.3181
udel-D-E-3A	0.0985	0.0784
udel-D-E-4A	0.3472	0.2808
udel-D-E-5A	0.0932	0.0877

The second and third runs use the same query dimension items, but in a rather different ways. The first difference is that the third run concatenates selected-items from a dimension in order to get one related query, while the second one uses each of the selected-items as a related query. As a consequence, the second difference is that the third run has much fewer related queries than the second run, which could explain why the second run largely outperforms the third run (0.3900 vs 0.0985).

The poor performance of the third and fifth runs (udel-D-E-3A and udel-D-E-5A) may require more investigation in order to have a clearer assessment of everything that went wrong. One very important probable cause for the performance hits is the kind of related queries we use for both runs. For both runs, the related queries are obtained by concatenating selected-items from a dimension in order to get one related query and there ends up being much fewer related queries than in udel-D-E-1A, udel-D-E-2A and udel-D-E-4A. As explained in section 2.3, we consider only the first five dimensions. Thus the largest number of related queries is only five.

Table 5. α-nDCG@10 and α-nDCG@20 results for the 5 runs as well as the run udel-D-E-6A

Runs	α-nDCG@10	α-nDCG@20
udel-D-E-1A	0.703403	0.696239
udel-D-E-2A	0.485831	0.513147
udel-D-E-3A	0.154978	0.183296
udel-D-E-4A	0.433588	0.45472
udel-D-E-5A	0.065483	0.070904
udel-D-E-6A	0.624203	0.644779

The number of related queries used in our experiments is clearly impacting our results. The quality of the related queries appears to be a very important factor as well. For example, using all the query suggestions from Bing, Google and Yahoo! appears to be more useful for the aggregation method than using dimension items. In fact, the results we obtain for udel-D-E-6A solidifies that statement. Udel-D-E-6A is a run that was created to investigate what happens when we combine the related queries generated for udel-D-E-1A with the related queries generated for udel-D-E-2A in order to obtain a much bigger number of related queries before aggregating the rankings. Table 5 shows α-nDCG@10 and α-nDCG@20 measures for udel-D-E-6A as well as for the 5 runs we submitted. Comparing udel-D-E-6A to udel-D-E-1A, we can see that increasing the number of related queries does not necessarily improve the final ranking. In fact, in the case of udel-D-E-6A, it decreases the performance achieved by udel-D-E-1A. This is probably because, when it comes to combining evidence from different formulations of the original query, dimensions' items are lower quality alternative formulations – in this context – than the combination of Bing, Google and Yahoo! query suggestions.

It is important to note that even though udel-D-E-6A performs worse than udel-D-E-1A, it does perform better than udel-D-E-2A, which is probably because Bing, Google and Yahoo! query suggestions help bring in better quality alternative formulations than the ones in udel-D-E-2A.

Also of note is that α-nDCG results correlate very well with the official measures.

5. REFERENCES

[1] Belkin, N. J., Kantor, P., Fox, E. A., & Shaw, J. A. (1995). Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management, 31*(3), 431-448.

[2] Liu, Y., Song, R., Zhang, M., Dou, Z., Yamamoto, T., Kato, M., ... & Zhou, K. Overview of the NTCIR-11 IMine Task. In NTCIR (Vol. 14).

[3] Zhicheng Dou, Sha Hu, Yulong Luo, Ruihua Song and Ji-Rong Wen.: Finding Dimensions for Queries. In CIKM. (2011).