Query Expansion for Real-time Summarization Track

Jianbo Pei

Introduction

The purpose of Real-time Summarization Track is to construct a system that can retrieve tweets timely that are relevant to interests profiles, and deliver the relevant tweets to the users in two scenarios: A. push relevant tweets immediately to users as push notifications. B. send an email that summarizes relevant tweets to users at the end of the day.

Two major requirements for the retrieved tweets are the relevance and the novelty. The relevance means that the obtained tweets need to be relevant to interest. And the novelty indicates that similar tweets are considered redundant and only one of them needs to be selected. Therefore, all results should meet these two requirements. For the overall project, there would be seven substantial components which are interest profile, query expansion, tweet stream, tweets pre-processing, relevance filter, novelty detection, and tweets push. And this paper will only focus on the query expansion component and illustrate the methods for query expansion in detail.

Query Expansion

The query expansion is essentially based on interest profiles. Each interest profile contains four fields which are “topid”, “title”, “description”, and “narrative”. All fields except “topid” provide information need about a user interest. The differences can be explained as following: the title only consists of several keywords; the description is a one-sentence statement of information need; the narrative is a paragraph providing detailed information need. Considering the importance of all information from interest profiles, the query expansion will be applied on all three informative fields except the topid field.

The final expanded query will be a combined query that eliminates same terms from expanded title query, expanded description query, and expanded narrative query. The narrative query expansion varies on the number of sentences a narrative has. Essentially, if a narrative has only one sentence, the narrative will contribute 10 terms to final query. Otherwise, each sentence in a narrative will contribute 5 terms to final query.

Title Query Expansion

The approach for title query expansion is first obtaining the relevant tweets, then computing the TF-IDF score, and finally selecting top-10 scored terms as the expanded query for the title field.

To obtain the relevant tweets based on the title, the tweepy API is used. To successfully implement the tweepy API, a Twitter account is needed because there are four fields of authorization required to declare. The maximum number of retrieved tweets for each title is set to be 100 which is also the maximum at each search.

After successfully retrieving relevant tweets for a title query, next step is to compute TF-IDF score for each term from the retrieved tweets. The libraries implemented here are NLTK and sklearn. NLTK is for stemming all terms and sklearn is for computing the TF-IDF score. The retrieving process will ignore the retweets and urls in tweets to obtain better results. The TF-IDF scores will be ranked from high to low and only return the top-10 scored terms. The final results will be the original forms of top-10 stemmed terms.

Because the Twitter search API has a limit rate of 180 calls per 15 minutes which is 900 seconds, the interval waiting time for each search is set to be 5.05 seconds to avoid from exceeding the limit rate.

Description Query Expansion

The relevant search results for description query expansion is based on Google search engine. For each description as a search query, top 30 snippets from Google search are crawled using BeautifulSoup and formed as a corpus. In case some certain descriptions will not generate 30 snippets, the query expansion will only crawl available snippets to avoid the occurrence of crawling error.

After a corpus is generated, top-10 scored terms will be retrieved based on the TF-IDF score, which is similar to title query expansion. Eventually, the original forms of the top-10 stemmed terms will be returned as a final query.

To bypass the captcha from Google search engine, an API called UserAgent is implemented. The UserAgent API will randomly generate different headers for each search. It will deceive the search engine to consider each search is an action taken by a human being. Therefore, it will avoid the occurrence of captcha. Also, in case the search runs too frequently to cause the occurrence of captcha, a 20 second waiting time is set among each search.

Narrative Query Expansion

Similarly, narrative query expansion also uses Google search engine to form corpus from crawled snippets. However, because each narrative contains several sentences, it will not generate nice results if the whole narrative is taken as a search query. Therefore, each sentence itself in a narrative will become a search query to capture 30 snippets. To increase the pertinence of each sentence, the title in the same interest profile will be added after each sentence.

If a narrative contains only one sentence, a top-10 scored terms from corpus will be generated. Otherwise, for each sentence, a top-5 scored terms from corpus will be returned. Therefore, a narrative normally could result in 10 to 20 query terms based on how many sentences a narrative consists of. The final query will be the original terms of the top stemmed terms.

Since narrative query expansion also uses Google search engine, a captcha will occur when Google search engine detects a robotic action. Therefore, a UserAgent API and 20 second waiting time are used likewise as description query expansion.

Problems and Solutions

One major problem is the limit rate for Twitter search. Because the limit rate is only 180 for each 15 minutes, a reasonable and efficient solution is needed to avoid from exceeding the limit rate. Therefore, after calculating and tuning, a 5.05 second waiting time is set to be the best solution to this problem.

Another major problem is the occurrence of captcha from Google search. If the Google search engine detects too frequent search calls and robotic actions, a captcha will be generated and any following search will be terminated unless waiting for half an hour without using the search call. It is not feasible to bypass the captcha with just waiting some time among each search call. The detection technique behind it is sophisticated. There are many ways to bypass the captcha to allow continuing search. However, most them are

difficult to implement. Differently, UserAgent API solves the problem in an easy and tricky way. By generating random headers for each search and adding a 20 second waiting time among each search, the Google search engine will not generate the captcha to terminate search calls.

Evaluation Results

The data sets used to evaluate the final expanded query are 56 judged topics from 2016 track. The evaluation measurement used is expected gain. It is the sum of gains divided by total number of tweets. Non-relevant tweets will receive a gain of 0. Relevant tweets will receive a gain of 0.5. And highly-relevant tweets will receive a gain of 1.0. For testing the query expansion performance, 100 tweets for each interest profile were retrieved to evaluate the expanded queries. To find best results, 5 and 10 of k for top-k terms, 30 and 50 snippets corpus for description and narrative query expansion, and different weights for final queries were tested. The weighting is assigned differently on which field the final query terms appear in. The best result was obtained from the top-k terms described before and 30 snippets crawled for each description and narrative query. The assigned weights are shown below in the chart.

30 snippets(weighting)	Title	No appearance	Description or Narrative	Expected Gain
	0.7	0.1	0.3	0.19625
	0.7	0.2	0.4	0.192857
	0.7	0.2	0.3	0.197143
	0.7	0.3	0.6	0.179107
	0.7	0.3	0.5	0.18625
	0.7	0.3	0.4	0.192679
	0.7	0.3	0.3	0.193036
	0.8	0.2	0.3	0.197312
	0.9	0.2	0.3	0.199643
	1	0.2	0.2	0.198036
	1	**0.2**	**0.3**	**0.202679**
Raw query				0.181429