Silent Day Detection in Real-Time Summarization Track

Kuang Lu, Peilin Yang, and Hui Fang

Department of Electrical and Computer Engineering
University of Delaware
140 Evans Hall, Newark, Delaware, 19716, USA
{lukuang,franklyn,hfang}@udel.edu

Abstract. In microblog retrieval, a system's ability to detect silent days and "shut up" during these days have an huge impact on its performance [1]. Therefore, in this year's Real-Time Summarization Track, we focus on detecting silent days. More specifically, we propose two silent day detectors **phrase-based weighted information gain** and **local query term coherence**, both of which focus on using query terms *collectively* instead of *individually*. Track results suggest that our methods can effectively detect silent days, which subsequently results in promising performances for both scenarios.

1 Introduction

Real-Time Summarization Track aims to build systems to retrieve tweets for interest profiles in two scenarios: 1.tweets are posted to the evaluation broker as soon as detected relevant; 2. relevant tweets are identified at the end of each day, simulating an email digest summaries for interest profiles. In both scenarios, the retrieved tweets should be both relevant and novel.

According to a previous study about the evaluation of microblog retrieval [1], as well as our experiments on last year's Real-Time Summarization Track, it is clear that a system's ability to detect silent days, meaning days that contain no relevant information, and "shut up" for these days are critical to the performance of the system. Therefore, in this year's RTS track, we focused on detecting silent days. It is important to note that both redundant and irrelevant tweets are treated as irrelevant tweets in the RTS track, but we chose to treat redundant tweets as relevant tweets in silent day detection phrase since finding silent days with only redundant tweets or silent days with no relevant tweets may require different techniques.

The problem of silent day detection is related to query performance prediction. However, we did not simply use the existing query performance prediction techniques since some core differences between these two tasks might significantly weaken the effectiveness of query performance predictors on silent day detection. First, the problem setting is fundamentally different because in query performance prediction, relevant documents are always assumed to be exist, which is clearly not always the case for silent day detection. Moreover, the majority of the predictors assign scores to individual terms and aggregate the scores to calculate the final score of a corpus or a result list. As a result, containing high inverse collection frequency terms would be enough for a silent day to receive a high score. Given these differences, new methods specifically for silent day detection might certainly be required.

In this year's Real-Time Summarization Track, we handled the silent day detection as a classification problem and proposed two silent day detectors **phrase-based weighted information gain** and **local query term coherence**, both of which focus on using query terms *collectively* instead of *individually*. As shown in the final TREC results, the proposed predictors can effectively detect silent days, which subsequently results in promising performances for both scenarios.

2 Silent Day Detection

The problem of silent day detection is naturally addressed as a classification problem since a binary decision of whether a day is silent or not should be made. In this year's track we focused on proposing new features for silent day detection.

2.1 Collectively Using Query Terms

Despite the high similarity between query performance prediction and silent day detection, the state-of-the-art query performance predictors could mistakenly assign high scores to silent days because of two reasons. First, in traditional performance prediction tasks, it is usually assumed, if not ensured [1], that there are at least some relevant documents in the corpus. The assumption leads to the fact that existing performance predictors aim to predict the performance in the realm of *possible*. When also taking *impossible* query-day pairs into account, the theoretical basis and intuitions of the existing predictors might not always hold. Some existing performance predictors are built based on this assumption [3, 4]. Even those that are not, the difference between a non-silent day and a silent day may not reflect on them. For instance, given some of the state-of-the-art predictors which are based on the standard deviation of the scores of the top ranked documents [5, 6, 2], a retrieved list of all irrelevant documents could have very high variance as well, just as that of an easy query.

The other problem is that the majority of the query performance predictors assign scores to individual terms and aggregate the scores to calculate the final score of a corpus or a result list. The score of a term is usually related to its inverse collection frequency [3]. Therefore, only containing terms with high inverse collection frequencies is sufficient for a day to receive a high score that would likely result in the day being labeled as a non-silent day. For instance, for query *MB254*: *cancer and depression* in last year's RTS track, since *cancer* and *depression* both have high high inverse collection frequencies values, a day containing only one of them could have a high score.

Therefore, we make no assumption about whether there are any relevant information, and hypothesize that using query terms *collectively* instead of *individually* might be more suitable and propose two sets of silent day predictors: **phrase-based weighted information gain** (PWIG), and **local query term coherence** (LQC), which we discuss below.

.

Phrase-Based Weighted Information Gain (PWIG). This predictor is inspired by the use of term proximity features, which are ordered or unordered groups of terms, in the query performance predictor weighted information gain (WIG) [7]. In WIG, both single term and term proximity features are used to predict the performance of a query over a collection, and the query term features would dominate the final scores of WIG, which is the second problem discussed above. Therefore, *phrase-based weighted information gain* (PWIG) is proposed so that single term features are removed and different weights for sequential dependence (considering order) and full dependence (without considering order) features are introduced.

More specifically, a query, its sequential dependence feature set as well as full dependence feature set are noted as Q, $S(Q)$, and $F(Q)$, respectively. $R(Q)$ is the union of $S(Q)$ and $F(Q)$. Given a term proximity feature ξ, the probability that it appears in a document D or a day C are denoted as $P(\xi|D)$ and $P(\xi|C)$. Finally, given a query which has a retrieved list L_K containing K documents $D_i \in \{D_1, D_2, ..., D_K\}$, we define PWIG as follow:

$$PWIG = \frac{1}{K} \sum_{D_i \in L_K} \sum_{\xi \in R(Q)} \lambda_\xi log \frac{P(\xi|D_i)}{P(\xi|C)}, \text{ if } |R(Q)| > 0 \tag{1}$$

where

$$\lambda_\xi = \begin{cases} \frac{\lambda}{\sqrt{|R(Q)|}}, & \xi \in F(Q) \\ \frac{1-\lambda}{\sqrt{|R(Q)|}}, & \xi \in S(Q) \end{cases} \tag{2}$$

In PWIG, we estimate the relevance of the retrieved list only using query terms' collective occurrences. WIG is computed for single-term queries.

[1] In fact for the track collection disk4&5, the query 672, for which no relevant documents were found in the original TREC 2004, is often excluded for performance predictor evaluation by previous work [2].

Local Query Term Coherence (LQC). *Local query term coherence* (LQC) is the other set of features we propose, which infers the difficulty of a query by estimating how coherent the query is. If, in a collection with respect to a day, the query terms are closely associated, the query is easy to the collection. Therefore, the day is not a silent day. The coherence between query terms are calculated as the coherences of groups of query terms with sizes range from 2 to the size of the original query, and the sub-coherences are aggregated to the final coherence of the query. This way of computing is chosen since the associations between query terms can vary significantly and it is not necessary for a query term to be closely related to all other terms. Moreover, we hope that by computing coherences for all possible query term groups, we cover all association conditions of a query. Another important computational detail about LQC is that these sub-coherences are computed *locally*. In other words, only top K tweets of the retrieved list are used instead of the whole collection. The reason behind it is that there can be more than one meaning of a query term, and the meaning of it in top ranked tweets are more likely to be the same as its original meaning in the query. The calculation details are as follow: the set of all possible multi-query-term groups of query Q is noted as $F(Q)$. The occurrence of ξ where $\xi \in F(Q)$ indicates a certain degree of coherence of terms in ξ. As a result, more tweets in top K contain ξ means higher degree of coherence between these terms:

$$C(\xi) = \frac{\#\text{ of documents containing }\xi}{K} \tag{3}$$

Intuitively, longer multi-query-term groups tend to be more important. Therefore, ξ are grouped by their sizes i through function $A()$:

$$C_i(Q) = A(\{C(\xi) : |\xi| = i\}) \tag{4}$$

We employed three implementations of $A()$: Max, Average, and Binary. Binary means that if there is at least one occurrence of ξ, the final value would be 1. The function Binary is employed so that even in the extreme case where there is only one appearance of a multi-query-term group, **LQC** is still able to detect it. This scenario could be the result of only one relevant tweet of a day. $C_i(Q)$ are then aggregated further by using a weighted sum of them to calculate the raw LQC ($rLQC$):

$$rLQC = \sum_{i=2}^{|Q|} log(i) \times C_i(Q) \tag{5}$$

$$LQC = \frac{rLQC}{iLQC} \tag{6}$$

The weight is implemented as the logarithm of the size. This weighting scheme is used to favor longer query-term collection while not overly penalizing the weight of short ones when the query is long. $rLQC$ is then divided by the ideal LQC to obtain the final value of LQC.

2.2 Training the Silent Day Classifiers

The proposed two sets of predictors were then used as the features of Naive Bayes classifiers to detect silent days. We now discuss the details about how the classifiers were trained.

Training Data The data from 2015 Microblog Track and 2016 RTS Track were the obvious choices for the training data. In order to obtain more training data in the hope of improving the classifiers, we modified the data used in 2011 and 2012 Microblog Track by identifying silent days according to the timestamps of the tweets and queries so that they could be used to train the classifiers.

Parameter Settings We tuned the parameter settings of the proposed silent day detectors on the training data in order to obtain the optimal parameter settings for the submitted runs. For phrase-based weighted information gain, the number of tweets K was set to 5 and λ was set to 0.2. Whereas, for local query coherence we set the number of tweets as 5, 10, 100 for the aggregation functions Max, Average, and Binary, respectively.

Table 1: Summary of Silent List Phenomenon

Collection	**# of Silent Days**	**# of Silent Lists**[*]
2011&2012	355	255
2015	126	121
2016	131	171

[*] silent days are excluded for this number

Different Labeling Strategies: Silent Day vs. Silent List. During the experiments of exploring silent day detection, an interesting phenomenon was discovered: given a query, it is possible that there are some relevant tweets in the day, but the retrieval method fails to retrieve any relevant documents in the top 10 results (10 is chosen here since it is the tweets-per-day limit). In other words, the retrieved list is ”silent” but the day is not. When using the retrieval function F2EXP [8], the number of silent lists in tracks of different years are shown in Table 1. As it can be clearly seen, this phenomenon is frequent. Therefore, besides labeling silent days only based on the judgment information, there is another reasonable labeling strategy which labels days with silent lists as silent days. There are two reasons why this could be more desirable. First, since there are no relevant tweets in a silent list, in terms of performance, returning any tweets from the top of the list is the same as treating the day as a silent day and returning no tweets. More importantly, since the proposed detectors are post-retrieval and rely on the top retrieved tweets, they may receive similar values for silent days and silent lists. As a result, assigning them the same label could be more benefitial for the final classification performance than assigning them with different labels. Both labeling strategies were explored in our submitted runs.

3 Submitted Runs

We use the same corpus we crawled using twitter ”sample” end point of the twitter stream api[2] during the evaluation for both scenarios. Non-English tweets were filtered out. For the downloaded tweets, only ”id” and ”text” fields were used. If a tweet was a retweet, the ”text” field of the original tweet was used since retweets would be normalized to the underlying tweets for evaluation. For the interest profiles, we only used ”title” field as initial queries for our system.

3.1 Scenario A

In scenario A, we tested the effectiveness of the proposed silent day detectors as well as explored how different silent day labeling strategies affect the performance. There are 4 steps for generating our scenario A runs: 1) Given a day and a query representing an interest profile, retrieve an initial result at the end of day using some ranking function; 2) Use a silent day classifier to detect whether a day is a silent day or not; 3) If the day is a silent day, do nothing, but if it is not, use the top 10 retrieved tweets as candidate tweets 4) Perform redundancy detection on the candidate tweets and return non-redundant tweets. We submitted three runs, which all used F2exp [9] as the ranking function and jaccrad distance and threshold 0.5 for the redundancy test. The differences among these runs are the queries for interest profiles, the labeling strategy for training silent day classifiers, and the number of returned tweets for non-silent days. The details of each runs are described below:

- $UDInfoBL$: In this run, ”title” field of the interest profiles were used as queries. We trained the classifier without considering silent lists. For each non-silent day, only the top non-redundant tweet was returned. The decision of returning one tweet was based on our experiment results on the previous year’s tracks.
- $UDInfoSDWR$: Similar to the previous run, except that the classifier was trained by treating silent lists as silent days.

[2] https://dev.twitter.com/streaming/reference/get/statuses/sample

- $UDInfoEXP$: In this run, we searched the "title" field on commercial search engines such as Google and Bing before the evaluation period. The snippets of the search results were subsequently crawled and used to expand the "title" field [8]. The expanded queries were used to retrieve results for each day for each query. The classifier used in this run is the same as the one in $UDInfoBL$. For each non-silent day, all the non-redundant tweets of the candidate tweets (up to 10) were returned.

There are two things need to be noted for the submitted scenario A runs. First, in run $UDInfoBL$ and $UDInfoSDWR$, we chose to return only 1 tweet for a non-silent day since when we conducted experiments on previous years' tracks about how the number of returned tweets would affect the performance, returning 1 tweet provided the best performance. Moreover, since we initially planed to use the relevance feedback provided from the broker, we decided to return the maximum number of tweets allowed for a non-silent day in run $UDInfoEXP$ to maximize the amount of relevance feedbacks we could receive. However, because the relevance feedback containing conflicting judgments and we did not come up with a reasonable way of using them, we did not use the feedbacks for our scenario B runs.

3.2 Scenario B

In scenario B, we wanted to investigate different redundancy detection methods. Therefore, we submitted three runs which were all based on $UDInfoSDWR$, which was expected to be the best run for our scenario A at that time. The only difference between these runs is how redundant tweets were detected. The detailed explanation of them are listed below:

- $UDInfoJac$: It is the same as $UDInfoSDWR$ except returning 10 tweets per day instead of 1.

- $UDInfoW2Pre$: This run is similar to $UDInfoJac$ except the fact that we changed the redundancy detection method to a word embedding based method. More specifically, we represent a tweet with a vector that is the averaged sum of the vectors of words in the tweet. The cosine distance between the vector representations of two tweets were used to decide whether they are redundant. The word vectors used were the pre-trained vectors trained on part of Google News dataset[3], which was provided by the authors of the toolkit word2vec [10]. The redundancy threshold was set to 0.9, which was tuned on previous years' data.

- $UDInfoW2VTWT$: It is similar to $UDInfoW2VTWT$. The difference is that, instead of using pre-trained vectors, we trained the vectors on the tweets we crawled before the evaluation period for about a 7-day period. The redundancy threshold was 0.9, which was tuned on previous years' data.

4 Results and Analysis

The performances of submitted runs as well as the track medians for different scenarios are shown in Table 2. Note that only primary metrics, which are EG-p for scenario A, and nDCG@10-p scenario B, are reported.

For scenario A, it can be clearly seen that UDInfoSDWR and UDInfoBL can outperform track median significantly, which shows that the silent day detection methods we used are very effective. However, UDInfoSDWR, which uses the classifier that employs the labeling strategy considering silent lists the same as silent days, performs worse than UDInfoBL, which uses the classifier that employs the labeling strategy without considering silent lists. It is inconsistent with our experiments on past years' data. We discovered through additionally analysis and experiments that it might be caused by the fact that there are less silent days in this year. In this year there are only 17.7% of days that are silent days, which are less than that of 2015 (24.7%) and 2016 (23.4%). As a result, in this year precision tends to be more impactful on the system's performance. However, the classifier trained by treating silent lists as silent

[3] https://code.google.com/archive/p/word2vec/

Table 2: Performances of Submitted Runs

(a) Performances of Scenario A

Run Name	**EG-p**
UDInfoBL	0.3226
UDInfoSDWR	0.2907
UDInfoEXP	0.2025
Track Median	0.2194

(b) Performances of Scenario B

Run Name	**nDCG@10-p**
UDInfoJac	0.2886
UDInfoW2Pre	0.2933
UDInfoW2VTWT	0.2906
Track Median	0.1865

days has advantageous performance in past years because it predicts silent days more aggressively, which leads to higher recall but lower precision. Thus, using it is less effective on this year's data.

For scenario B, there seems to be no difference among the redundancy detection method in terms of performance, which might suggest that the word embedding based method does not prove to be superior over jaccard distance.

5 Conclusion

In this year's Real-Time Summarization Track, we directly address the problem of silent day detection. It was handled as a classification problem and two sets of silent day detectors using query terms collectively are introduced. Track results indicate that the proposed silent day detectors can indeed be effective for RTS tracks, and thus prove their usefulness to microblog retrieval tasks when silent days are somewhat frequently present.

References

1. Tan, L., Roegiest, A., Lin, J., Clarke, C.L.: An exploration of evaluation metrics for mobile push notifications. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '16, New York, NY, USA, ACM (2016) 741–744
2. Shtok, A., Kurland, O., Carmel, D., Raiber, F., Markovits, G.: Predicting query performance by query-drift estimation. ACM Trans. Inf. Syst. **30**(2) (May 2012) 11:1–11:35
3. Cronen-Townsend, S., Zhou, Y., Croft, W.B.: Predicting query performance. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '02, New York, NY, USA, ACM (2002) 299–306
4. Cummins, R.: Document score distribution models for query performance inference and prediction. ACM Trans. Inf. Syst. **32**(1) (January 2014) 2:1–2:28
5. Pérez-Iglesias, J., Araujo, L.: Standard deviation as a query hardness estimator. In: Proceedings of the 17th International Conference on String Processing and Information Retrieval. SPIRE'10, Berlin, Heidelberg, Springer-Verlag (2010) 207–212
6. Cummins, R., Jose, J., O'Riordan, C.: Improved query performance prediction using standard deviation. In: Proceedings of SIGIR '11. (2011)
7. Zhou, Y., Croft, W.B.: Query performance prediction in web search environments. In: Proceedings of SIGIR '07. (2007)
8. Fang, H., Zhai, C.: Semantic term matching in axiomatic approaches to information retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '06, New York, NY, USA, ACM (2006) 115–122
9. Fang, H., Zhai, C.: An exploration of axiomatic approaches to information retrieval. In: Proceedings of SIGIR '05. SIGIR '05 (2005)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)