A Practical Framework for Real-time Diffusion Analysis in Social Media

Miki Enoki
IBM Research – Tokyo
Ochanomizu University
enomiki@jp.ibm.com

Issei Yoshida
IBM Research – Tokyo
issei@jp.ibm.com

Masato Oguchi
Ochanomizu University
2-1-1 Otsuka, Bunkyou-ku, Tokyo
oguchi@is.ocha.ac.jp

ABSTRACT

In a microblogging service such as Twitter, timely knowledge about what kinds of information are diffusing in social media is quite important for companies. It is also effective to identify the influential users who are retweeted frequently by many users. We are now developing an information diffusion analysis system that enables real-time analysis of streaming social data. However, streaming data is usually divided into segments called windows. The window size is decided by the amount of data or a length of time. This means that we have to use fragmented diffusion data for our diffusion analysis. We propose a customized time-window model by effectively estimating diffusion extinction, which enables an early decision to remove stale data from in-memory data store. We evaluate our implementation in terms of both the efficiency of query processing and effectiveness of our time-window model.

Keywords

Social media, In-memory database, Stream processing

1. REAL-TIME DIFFUSION ANALYSIS SYSTEM

A microblogging service has special characteristics in that the frequency of the users' posts is high and strongly related to transient topics. If a Retweeted message (an RT) is shared widely, then many users may read that tweet, which may have a large impact on the real world. Knowing what kinds of information are being widely broadcast on social media is essential for companies to protect their corporate brand's value and to follow market trends.

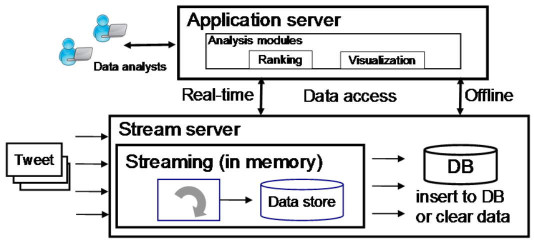

Figure 1. Diffusion analysis system.

CF'15, May 18-21, 2015, Ischia, Italy
ACM 978-1-4503-3358-0/15/05.
http://dx.doi.org/10.1145/2742854.2742899

We focus on a system for analysing diffusion data of tweets retweeted actively in real-time. Figure 1 shows our system for the real-time diffusion analysis system. The framework consists of the stream server and the application server. The application server provides various analysis modules for the diffusion data. For example, the "Ranking" module creates a ranking of retweet counts to detect the current hot tweets or the ranking of influential users for a specified topic. The "Visualization" module displays a diffusion network for a specified tweet.

In stream processing, streaming data is usually divided into segments called windows. The window size is decided by the amount of data or a length of time. Figure 2 shows examples of diffusion data processing for retweets. Each box represents a tweet message, either a tweet or a retweet. The sequentially arriving tweets are divided into the defined time windows, and partitioned by tweet IDs. If a tweet is a retweet, then it is partitioned by the tweet ID of the original tweet. When a tweet is retweeted, a thread for the tweet is created and then we collect all of its retweets in the window.

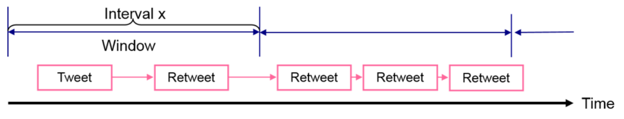

Figure 2. Stream processing on Twitter streams.

Since stream system is real-time processing, the past data from the window may not be found in the system even if the tweet is still actively being retweeted. This means that we have to use fragmented diffusion data for our diffusion analysis.

To handle all of the diffusion data for each tweet retweeted by users, we used an in-memory database to collect the diffusion data in the stream system. We can issue various queries interactively against the stored data. The queries should be expected to be processed quickly, since the data is stored in memory.

2. PROPOSAL OF MAINTENANCE METHOD FOR DIFFUSION DATA IN IN-MEMORY STORE

We have to delete stale data from the in-memory database since the amount of memory is limited, but we should retain the diffusion data as long as it is retweeted frequently. We introduce a customized time-window for each tweet to decide when it is to be removed. We customize the time-window size as shown in Figure 3. We want to set the size so that diffusion data is retained until the extinction of the retweets for each tweet. The start of the window is when the original tweet was posted. The end of the window should be the last retweet, but it is difficult to find that endpoint, because even an old tweet could suddenly be retweeted.

Retweet messages diffuse rapidly within hours or even minutes but usually don't last many days. Therefore we assume that after a certain period from the original tweet posting, the occurrence probability of a retweet is low enough that we can safely remove the diffusion data from the in-memory database. As a simple technique, we use a fixed period of time to determine diffusion extinction for all of the diffusion data. However, the time lag between diffusion extinction and original tweet varies depending on tweets.

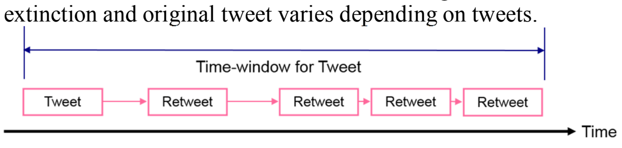

Figure 3. Custom time-window.

Another propagation model of retweet has been studied [1] and follows a log-normal distribution. Therefore we regard the distribution of retweets over time in each time slot as a probability distribution. Figure 4 shows an example of the distribution of the retweet delay times after the original tweet.

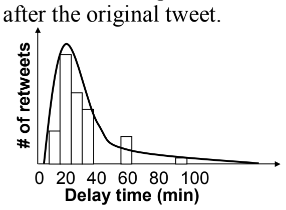

Figure 4. Retweet distribution of a tweet.

We fit the distribution for the probability density function of the log-normal distribution. The probability density function of the log-normal distribution with parameters μ and σ is given by

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(\ln x - \mu)^2}{2\sigma^2}} \quad , \ x > 0$$

We estimate μ and σ with diffusion data for the first one hour from when the original tweet was posted. Then we calculate the 90 percentile point in the estimated distribution and define the point as the time-window size for the tweet. This is regarded as a time when the diffusion will be almost extinct. This means we should retain about 90% of the diffusion data of the tweet in the database.

3. PERFORMANCE EVALUATION

We measure the query performance of our system using typical analysis scenarios with real diffusion data from Twitter. We also evaluate our method for customized time-windows.

3.1 Evaluation Methodology

(1) Query processing

We measure the response times of the following queries. They are used in typical analysis scenarios.

Query1: Obtain the retweet diffusion network data of specified tweets

Query2: Find popular tweets (returns the top 10 most retweeted tweets)

Query3: Find influential users (returns top 10 users who retweeted most often from among the specified tweets)

(2) Customized time-window

We determine the time of diffusion extinction by fitting a log-normal distribution and estimate retweet curve and calculate the 90 percentile point for that curve to determine the time-window size for the tweet. We count the total number of retweets within the time-window and calculate its percentage of the total retweet count.

We use Japanese retweet data from Twitter for the period from 1/19 to 2/11/2014. This period included the Tokyo governor's election in Japan. We collected the tweets and retweets that were posted from the five official accounts representing the candidates in the election, as well as the activities of heavy users of social media. The total number of tweets that were retweeted was 1,611. The total retweet count of these tweets was 76,593.

Our test server has 2 Xeon X5670 CPUs (2.93GHz, 6 cores) with 32 GB of RAM, and Red Hat Linux 5.5. A client for measurement of the query performance ran in the same server. We use the H2 database [2] as the in-memory mode for the in-memory database.

3.2 Experimental Result

(1) Query processing

The average response times of Query1, 2 show that the query processing is extremely fast, while the response time of Query3 is much longer than the other queries. It is because the computational cost is higher for the aggregation operations (Table 1). We created a special view to use the Threshold Algorithm [3] to calculate the top-k rankings more rapidly. The query performance of Query3 is greatly improved by using the view (Query3 optimized in Table 1).

Table 1. Query performance result

Query1	Query2	Query3	Query3 optimized
0.2 ms	0.4ms	5.6ms	0.09ms

(2) Customized time-window

We calculate the average percentages using a fixed period of time to determine the diffusion extinction for all tweets for comparison. We set the fixed time (= t) to the value of 1, 10, 11 and 12 hours. When t is 1 hour, the average coverage is only 55%. This indicates that about a half of the retweets occurred after more than one hour from the original tweet posting. When t is 11 hours, the coverage is the same as with our method. In this case, diffusion data of each tweet is stored in the database for 11 hours because the time-window size is set to 11 hours.

Table 2. The average coverage of retweet diffusion

t=1	t=10	t=11	t=12	**our method**
55.0%	89.1%	89.9%	90.6%	**89.9%**

The average difference time between the actual and the estimated results is 368 minutes, while the difference between the actual results and 660 minutes (= time-window size of 11 hours) is 535 minutes. This indicates that the accuracy of our method is better than the fixed time-window size.

4. CONCLUSION

In this paper, we introduced a framework for a diffusion analysis system that enables real-time analysis of streaming social data. We measured the query performance of our system using typical analysis scenarios and showed that the query processing was extremely fast. We also evaluated our method to determine diffusion extinction. The accuracy of our method was better than fixed time-window size.

References

[1] Galuba, W., Chakraborty, D., Aberer, K, Despotovic, Z., and Kellerer,W., "Outtweeting the Twitterers – Predicting Infor-mation Cascades in Microblogs.", WOSN, 2010.

[2] H2 Database Enginehttp://www.h2database.com/html/main.html

[3] Ronald F., Amnon L., and Moni N., "Optimal aggregation algorithms for middleware. " PODS, pp.102–113, 200