# MicroFilter: Real Time Filtering of Microblogging Content

Ryadh Dahimene
ryadh.dahimene@cnam.fr

Cédric du Mouza
cedric.du_mouza@cnam.fr

CEDRIC Laboratory - CNAM
292, Rue Saint-Martin
Paris, France

## ABSTRACT

Microblogging systems have become a major trend over the Web. After only 7 years of existence, Twitter for instance claims more than 500 million users with more than 350 billion delivered update each day. As a consequence the user must today manage possibly extremely large feeds, resulting in poor data readability and loss of valuable information and the system must face a huge network load. In this demonstration, we present and illustrate the features of MicroFilter (MF in the the following), an inverted list-based filtering engine that nicely extends existing centralized microblogging systems by adding a real-time filtering feature. The demonstration proposed illustrates how the user experience is improved, the impact on the traffic for the overall system, and how the characteristics of microblogs drove the design of the indexing structures.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

## Keywords

Microblogging; Twitter; Filtering

## 1. MOTIVATION

Microblogging web services such as *Twitter* or *Indenti.ca* have become a major trend over the Web 2.0 as well as an important communication vector. In less than six years, Twitter has grown in a spectacular manner becoming now the most widely used Microblogging system in the world with more than half a billion users in January, 2013[1].

For different reasons (security, advertisement, control policy, ...), microblogging systems rely on a fully centralized architecture. Each post published by an account is received by the system that forwards it to *all* followers of the publishing account. Different parameters impact consequently the overall performances of the posting: (*i*) the high update frequency of some accounts (e.g. newspapers, tech-blogs like @techcrunch that posts more than a *tweet* per hour, journalists), and (*ii*) the average number of followers, between 30 and 200 depending on the system considered (see [4] for Twitter), with accounts which reach millions of followers (e.g. celebrity account @ladygaga has more than 36,5 million followers).

From the service provider point of view, the system must consequently face a tremendous amount of posts to forward. For instance Twitter, which claimed in Jully 2011 more than 200 million tweets a day, must deliver daily over 350 billion tweets[2]. This traffic overload (especially for high-followed accounts) represent from an architectural point of view a scalability bottleneck. From the user's point of view, the amount of posts received from the 30-200 accounts he follows, among these some very-productive ones, leads in the very large feeds of posts. This results in poor data readability and potentially loss of valuable information. A direct consequence is the high dynamicity of the graph: to avoid flooding, users follow temporary an account to cover a peculiar event, and then unsubscribe because they can't manage the continuous flow of posts [3] [2].

For the data analyst, leveraging the users manually introduced filters may also provide useful insights that can help analyse the huge microposts datasets.

Based on these observations we introduce the MicroFilter system in order to improve the user experience and reduce the network load of such a systems. We introduce filtering in microblogging systems motivated by the main idea that if a user $A$ follows another user $B$ for some topics, consequently he wants to receive only a subset of $B$'s posts that matches his interest, expressed as keywords. To scale, the *MF* structure must efficiently retrieve for an incoming post all followers of a publishing account whose filter is satisfied by the post. MicroFilter is, to the best of our knowledge, the first attempt to manage filters in existing centralized microblogging systems, and to efficiently introduce them in the underlying graph index.

The goal of this demonstration is to illustrate the benefits of the MicroFilter filtering engine. We will present how the user can interact with the *MF* powered microblogging system and how he can avoid the *flooding* phenomenon. This demonstration will also show the approach benefits from the service provider point of view, *i.e.* the impact on computing, storage, and network load.

## 2. FILTER INDEXING

To improve microblogging systems performances we propose keyword-based filters. We name the social graph whose

[1]http://www.telegraph.co.uk/technology/9837525/Half-a-billion-people-sign-up-for-Twitter.html

[2]http://latimesblogs.latimes.com/technology/2011/07/twitter-delivers-350-billion-tweets-a-day.html


*WWW 2013 Companion,* May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

edges are labeled by filters the *filtered social graph* (*FSG*). We propose and compare in [1] three indexing structures using different factorization criteria. Based on our conclusion regarding space and computation costs, we decide to implement the PTF (*Publisher-Term-Follower*) for our *MicroFilter* filtering engine.

In the *PTF*-index, (as $Publisher - Term - Follower$ index), a key is an account $n \in N$, and the value is the corresponding *posting list* $Postings_{PTF}(n)$. We factorize the posting list on the terms, so each term $t$ is associated to a list of the followers of $n$ that choose $t$ as a filter for the posts of $n$. So $Postings_{PTF}(n) = \{(t_1, \{n^1_{t_1}, n^2_{t_1}, \ldots\}), (t_2, \{n^1_{t_2}, n^2_{t_2}, \ldots\}) , \ldots\}$, with $n_i \in \Gamma^-(n)$ and $t^j_{n_i} \in label(n_i, n)$.

# 3. DEMONSTRATION

## 3.1 System overview

Figure 1 shows the general architecture of *MicroFilter*. At the storage level we find the microblogging system *Filtered Social Graph*. For efficiency reasons and in order to avoid costly disk inputs/outputs, the system indexes the microblogging data grabbed from the storage layer in main memory using the *PTF*-index structure described in the model section. The generated structures can fit in main memory, and are used by *MF* to quickly route the incoming *tweets* flow.

At the user layer, clients can interact with the *MF* filtering

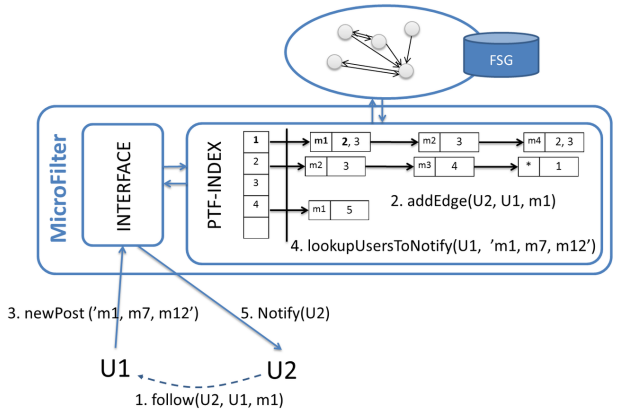


**Figure 1: Architecture of MicroFilter**

engine using a classical browser interface. This client-server model handles the interaction between users and the system trough the system web interface. Clients can use the interface to manage their filters and receive the posts that match their interests.

The user typical use case scenario illustrated in the figure 1 is described as follows:

1. At the beginning the user *U2* choose to follow the user *U1*, he wants to receive *U1*'s update that are related to the term *m1*, so that he explicitly filters the posts with the filter *m1*.
2. The *MicroFilter* system stores the new edge in the social graph by adding the new filterd *following* relationship in his in memory *PTF*-index.
3. The user *U1* posts a new update which contains the three terms *m1*, *m7* and *m12* using the web interface.
4. Now the system use the *PTF*-index to find users to notify, it starts with the hashtable entry of the update poster and find the users to notify. In this case *U2*.
5. Finally, the system can notify the right users for the new update. *U2* is then notified for *U1*'s update which contains the term *m1*.

The described architecture is used by *MicroFilter* to handle filtering over the continuous flow of posts we can meet in the microblogging world.

## 3.2 Implementation

We have implemented the the *PTF*-index structure on top of a JAVA platform. The users can interact with the system through a browser interface. The users are able to receive updates from accounts they follow. They can use two different modes. The first mode rely on the *all or nothing* paradigm *i.e* users retrieve all updates from their direct neighbourhood (it is actually the current state in all microblogging platforms). In the second mode, the users can activate the *MicroFilter* filtering engine and specify their interests by setting a list keywords which represent their interests. In this mode, the user can compare the average matching times of his filters with the content and can experience the real time filtering. From the service provider point of view, the challenging point consists on the ability to handle the filtering for the complete dataset of users/*tweets* and fit the inverted lists index in main memory.

The experiment we conduct in the demonstration consists of indexing in main memory the whole filtered social graph of our *Twitter* dataset (social graph + tweets for more than 148.5 million publisher-follower relationships, described in [1]) and then launch a continuous flow of incoming tweets (up to 15,7 million tweets) to evaluate how the system handles the matching. By tuning some parameters (number of users, size of filters) we can provide real time statistics on matching time, memory occupation and network load gain to illustrate the benefits of our approach. We observed that the *PTF*-index exhibit a linear growth and can handle the full 148.5 million edges dataset in 3.5 *Gb* of memory.

We observed also that the *PTF*-index can handle a post with an almost constant matching time (15 $\mu s$ in average). A result is that with the *PTF*-index we are able to handle, in a single centralized system, up to 66k posts per second. It means that we can easily manage, for instance, the *Twitter* historical peaks (around 12k tweets per second[3]).

# 4. REFERENCES


[1] R. Dahimene, C. du Mouza, and M. Scholl. Efficient filtering in micro-blogging systems: We won't get flooded again. In *SSDBM*, pages 168–176, 2012.

[2] F. Kivran-Swaine, P. Govindan, and M. Naaman. The Impact of Network Structure on Breaking Ties in Online Social Networks: Unfollowing on Twitter. In *CHI*, pages 1101–1104, 2011.

[3] H. Kwak, H. Chun, and S. B. Moon. Fragile Online Relationship: A First Look at Unfollow Dynamics in Twitter. In *CHI*, pages 1091–1100, 2011.

[4] H. Kwak, C. Lee, H. Park, and S. B. Moon. What Is Twitter, a Social Network or a News Media? In *WWW*, pages 591–600, 2010.


[3]http://blog.twitter.com/2012/02/post-bowl-twitter-analysis.html