An Intelligent PubSub Filtering System

Zeinah Hmedeh, Cedric Du Mouza, Nicolas Travers

An Intelligent PubSub Filtering System

Zeinah Hmedeh, Cédric du Mouza, and Nicolas Travers

CEDRIC, CNAM, Paris, France

Abstract

Content syndication has become a popular mean for timely delivery of frequently updated information on the Web. It essentially enhances traditional pull-oriented searching and browsing of web pages with push-oriented protocols. In such paradigm, publishers deliver brief information summaries on the Web, called news items, while information consumers subscribe to a number of feeds seen as information channels and get informed about the addition of recent items. However, many Web syndication applications imply a tight coupling between feed producers with consumers, and, they do not help users finding news items with interesting content. This demonstration shows a prototype which integrates the whole processus of thin filtering steps on numerous RSS feeds in memory, based on keyword-based subscriptions. Our system proposes to notify items related to set of keywords either by broad-match semantic, partial matching and a diversity/novelty filtering. In this demonstrate we discuss how our system integrates the three paradigms through dedicated indexes and a window-based structure for diversity/novelty filtering. We will demonstrate the management of notifications and global information on our system.

1 Motivation

Web 2.0 technologies have transformed the Web from a publishing-only environment into a living information place where passive readers have become active information collectors and content generators themselves. In this context, Web syndication formats such as RSS or Atom emerge as a popular mean for timely delivery of frequently updated Web content. According to these formats, information publishers provide brief summaries, called *information items*, of the content they deliver on websites [HVT+11], while information consumers subscribe to a number of RSS/Atom *feeds* and get informed about newly published items. Today, almost every personal weblog, news portal, discussion forum, user group or social media (*e.g.*, Facebook, Twitter, Flickr) on the Web employs RSS/Atom feeds.

Given that the amount of the information generated in Web 2.0 is growing dramatically nowadays. An active information consumer becomes rapidly overwhelmed by the amount of information he receives. There is an urgent need for efficient *real-time filtering methods across feeds*. We focus in this work on a *content-based Publish/Subscribe* paradigm for Web syndication in which information consumers are decoupled (in both *space* and *time*) from information providers and they can express their interest on specific information from items by using content-based subscriptions. Rather than overwhelming users with all items of a feed, *keyword-based subscriptions* can be matched on the fly against the content of incoming items originating from different feeds. Keyword-based subscriptions essentially allow specifying feeds on a custom-basis (according to the Web 2.0 philosophy) from information available "out there" and essentially act as *continuous queries* running into the future rather than trying to go "back in time" through searching information stored in a local store.

We have previously proposed three filtering structures [HKC+12] for this paradigm. Each of these structures reveals interesting characteristics in broad-match semantics. Now, we improve our system by integrating partial matching and diversity/novelty filtering services. In fact, over millions of subscriptions, we show that two large specific sets of non conventional subscriptions are noticed during the filtering process; (a) long subscriptions are never notified since very few items contain all subscription's terms. For this we proposed to relax the broad-match semantic using partial matching, (b) most of short subscriptions are still overwhelmed by too many matching items, especially by duplicate or similar items. To deal with this issue we integrated a diversity and novelty filtering step to our system in order to improve user satisfaction. By doing so, we reduce the amount of duplicate notifications corresponding to similar items. This last step needs to take into account the history of items already received by the user.

This demonstration details the architecture and items life cycle in a Publish/Subscribe system which integrates keyword-based subscriptions. We describe our new index for partial matching with a diversity/novelty filtering step.

2 Architecture

Our system is a Publish/Subscribe engine which gathers information (items) from feeds, filters them by keyword-based subscriptions and notifies the corresponding users with the resulting items. It is based on a push-oriented

architecture which allows users to be notified on real-time. The global architecture of our system is depicted in Figure 1 showing its main modules. It decouples sources from users in order to push directly notifications to them.

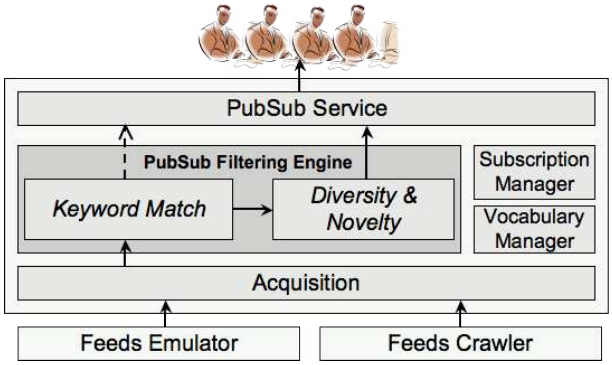

Figure 1: Architecture of the PubSub Filtering System

The **Acquisition** module converts feed items to useful items in our system. In fact, the filtering module is based on set of terms, so a simple transformation is applied on items. An extraction from *title* and *description* is necessary to produce this set of terms. According to this, term identifiers are provided in correlation to the *Vocabulary Manager*. Then, every new item is pushed to the *Keyword Matching* index.

The **Vocabulary manager** gives for each term its identifier and weight. It helps the system to make faster comparison and search during the process. Term weights are useful during the partial matching and novelty/diversity filtering steps.

The **Subscription Manager** links each subscription to the corresponding user information, and stores personal parameters for partial matching and novelty/diversity filtering processes. It also deals with updates of subscriptions on different structures of our system.

The **Publish/Subscribe Service** is in charge the communications between users and the system. It provides subscription management (insert, update, delete, parameters), and notification services. In fact, some specific notification services are necessary to provide new items to the user. It handles notifications by push (on real-time) and/or pull connections (on demand/refresh). Matching items from the *PubSub Filtering Engine* are notified to users, according to linked subscriptions.

The **Keyword Match** filtering system is an index that filters items coming from the *Acquisition* process. It is based on our previous work on the *Broad Matching* semantic [HKC+12], in which we have integrated a partial matching capability (see Section 3.1). Then, the item is pushed to the Diversity & Novelty index in order to filter it for every subscription that it satisfies. It can be also directly notified to the *PubSub service* if no further filters are required by a given user.

Diversity and Novelty filtering is a way to remove two kinds of duplicates in a set of documents. In our case, this filter is applied to each subscription according to its own history of notifications. In order to filter out duplicates for each subscription, a history of already delivered items is necessary to provide new information. Further details are provided in Section 3.2. Every new filtered item is notified to the *PubSub Service* according to linked subscriptions to be delivered to the corresponding users.

3 Filtering in PubSub

Our system is based on a two filtering steps, the first one on partial matching of keywords subscription, and the second on window-based history for filtering items by novelty and diversity according to their subscription. We will detail each filtering step and show how items can be notified.

3.1 Partial Matching

The broad-match semantic can be to strict for few notifications, especially for long subscriptions. We propose to extend our previous work [HKC+12] with the partial matching semantic. It relaxes this constraint by taking into account the weight of each term in a subscription, allowing a subscription to be notified if the "most characteristic" terms are present in the item. Thus, we study in this section the modifications to bring. We proposed subscriptions decomposition approach that consists of indexing all sub-subscriptions of interest. While the decomposition approach is the simplest way to handle partial matching for all structures, Count-based Inverted List (*CIL*) can also be extended to efficiently support partial matching. In this demonstration, we present only *CIL* for partial matching (CIL_p).

3.1.1 Term weight

Our partial matching relies on a matching score between the subscriptions and items, thus we need to define how to weight terms of the subscriptions. We assign a weight to each term in the vocabulary of items which represents

their importance. Several term weighting models are proposed in literature like the *Term Frequency* (TF) combined with *Inverse Documents Frequency* (IDF) [BYRN99], the *Term Discrimination Value* (TDV) [SWY75] or the *Term Precision* [BS74].

In our context, the TDV weighting function is more adapted for items and subscriptions. In fact, an item is a short set of terms (52 terms in average [HVT+11]) where term frequencies cannot be used, so the *TF-IDF* standard function is inappropriate. Moreover, the TDV weighting function measures how a term helps to distinguish a set of documents (*i.e.* the term influence on the global entropy). So basically for our subscriptions set, neither a very frequent term (not a selective filter), nor a very uncommon term (never lead to a notification) have an important TDV value.

Each term weight is the TDV value normalized by the sum of TDV values of the subscription terms. So, for a subscription $s=\{t_1, t_2, \ldots, t_n\}$, the subscription term weight $\varrho(t_k, s)$ is equal to: $\frac{tdv(t_k)}{\sum_{t_i \in s} tdv(t_i)}$

3.1.2 Extending Count-based Inverted List: CIL$_p$

In order to take into account partial matching, we need to define a satisfaction threshold κ_p which determines if an item is close enough to a given subscription. An item is notified if the sum of $\varrho(t_k, s)$ of all matched terms ($\sum_{t \in s \cap I} \varrho(t, s) > \kappa_p$) exceeds κ_p.

The main difference between *CIL* and *CIL$_p$* is the `Counter` which is replaced by a *decrementing counter* with entries equal to κ_p for each subscription, and term's weight $\varrho(t, s)$ is stored in postings lists for each subscription's id. Then during the matching attempt, for each term t_j of the item the corresponding value in the `Counter copy` of each subscription s in its postings list is decremented with $\varrho(t_j, s)$. Whenever a counter reaches zero, a matching is reported.

3.2 Novelty and Diversity

The filtering items by Novelty or Diversity is based on the history of received items per subscription. We are not looking of finding the k newest and most dissimilar items among a set of items to notify it to the user. User has already received a set of items (history) and we aim to check if the new published item is new (contains new information) and upgrade the diversity (makes the items less similar to each other). These definitions are based on the weight of item terms, we will use the TDV value for same reasons as explained previously.

3.2.1 Novelty

The objective of filtering items by novelty is to not send to the user the same information several times (identical or contained). Thus, an item I is said to have new information compared to another item I' if it contains more important terms which are not contained in I'. It is measured by the proportion of new terms weight that should exceed a novelty threshold κ_n: $\frac{\sum_{t \in (I \setminus I \cap I')} tdv_t}{\sum_{t' \in I} tdv_{t'}}$.

Notice that the novelty measure should not be symmetric. Regarding the example of a document d and a paragraph p in d, p is not new compared with d, but d is new compared with p. That is why using an asymmetric one, as Jaccard measure, is not suitable where the distance between two items is the ratio of their intersection on the set of their union.

So an item I is called new, according to a subscription with a corresponding history of items H, if there is no item $I' \in H$ that does not prove the novelty of I: $\forall I' \in H, \frac{\sum_{t \in (I \setminus I \cap I')} tdv_t}{\sum_{t' \in I} tdv_{t'}} \geq \kappa_n$.

3.2.2 Diversity

At the opposite of novelty, diversity deals with the global information contained into subscriptions' history. An item is diverse if the information that it contains is not present in the set of items already notified to the user. It measures how much the incoming item can increase the average pairwise distance $D(H)$ between the history's items. Given a distance metric *dist* between two items: $D(H) = \frac{2}{|H|*(|H|-1)} \sum_{I \in H} \sum_{(I' \in H \wedge I' \neq I)} dist(I, I')$.
So, an item I improves the diversity of H if and only if: $D(H \cup \{I\}) > D(H)$ (increases the average pairwise distance).

Several approaches were used in the literature to compute the distance between two items: Cosine [ZCM02], Euclidean [DP12a, PvA08], and Jaccard [DP12b]... Since the importance of diversity is bounded by measuring the amount of non-common terms (dissimilarity), the Euclidean distance is the more appropriate metric (Cosine and Jaccard are based on common terms). So, *dist* is measured by: $dist(I, I') = \frac{\sqrt{\sum_{t \in (I \cup I' \setminus I \cap I')} tdv^2(t)}}{\|I\|^2 \times \|I'\|^2}$ (To take into account items size and the importance of other terms in the items, a normalisation by the product of their Euclidean norm is applied).

3.2.3 Sliding window-based filtering

Our PubSub system is faced to a large amount of subscriptions (more than 10M), and must deal with a heavy bandwidth of incoming items (tens thousands of items per hour). The *CIL$_p$* index was design to efficiently

match items and subscriptions. For diversity and novelty computation, we take into account history of each subscription. Over time, this requirement is highly memory consuming. Fortunately, old information is no more relevant in term of filtering, that is why we adapted a sliding window-based on time to manage the history of items. All old items are removed from the system wherever they are older than a given period of time.

Rather than storing a history for each subscription, we will store each item once. This will reduce the memory space used by histories. Also, we will benefit to optimize the filtering process by computing once the novelty and distance between stored items and the new published item. For each subscription, we keep links between the subscription and the corresponding items of its history. An example of a common history is illustrated in Figure 2 where common items between subscriptions S_1 and S_2 are stored only once. Links between items and subscriptions means that those items have been notified by the corresponding subscription.

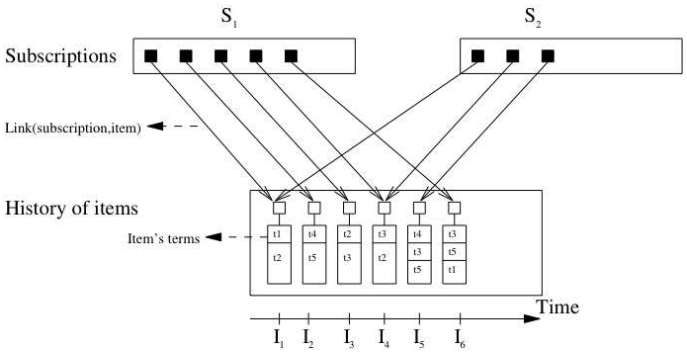

Figure 2: Common History

4 Demonstration

In our demonstration, we will show real-time impact of different parameters on our system. We will use a real data set of items composed of a total number of 10,7M items collected originating from 8K productive feeds (spanning over 2K different hosting sites) [HVT+11]. The subscriptions have been automatically generated based on the ALIAS sampling method [Wal77]. This method aims to generate subscriptions whose distinct terms follow occurrences in a real distribution, and their size respect the distribution of web queries size reported in [BJC+04]. It is characterized among others by a maximal size equal to 12 and an average equal to 2.2. For all sketches, the same set of subscriptions and items will be played in order to show the impact of each modification for different parameters.

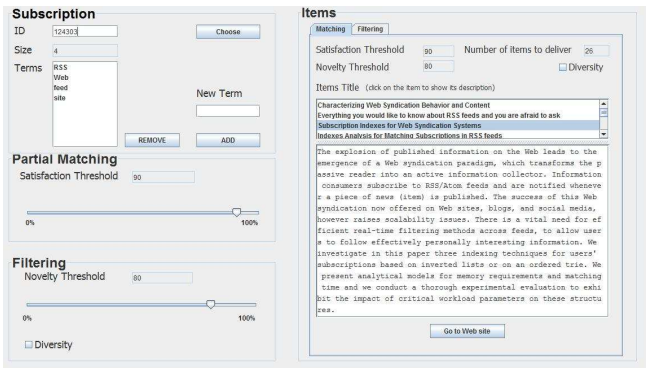

Figure 3: User interface

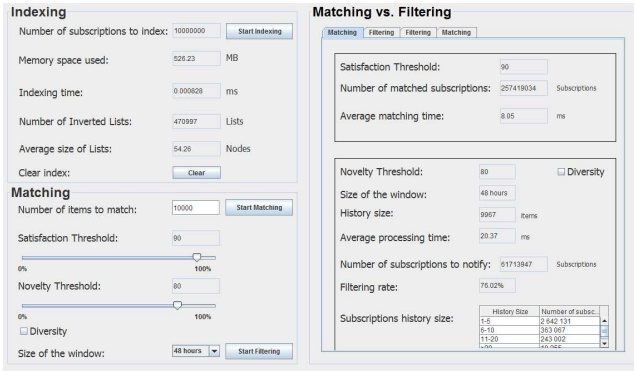

Figure 4: Admin interface

4.1 Subscription Scenario

The demonstration setting is corresponding to the user interaction on the system. As we said above, we will play the same set of subscriptions, so we do not allow user to add or remove subscriptions. But, user can choose the subscription to be considered and, add or remove its terms to see the impact of their weight on matching rate. All interaction will be shown on the user interface (Figure 3), and see on real-time notifications of items after the keyword-match index and the Novelty/Diversity filter. This scenario will show each step of tuning an intelligent filtering system based on keyword subscriptions.

In the top left part of the interface, the subscription size will be displayed to see the impact of short and long subscriptions on notifications rate. The partial matching threshold, the novelty threshold and take into account diversity in the filtering can be modified by the user in the part just below. Note that, these values will be the same for all subscriptions indexed.

For each play of items (partial matching or filtering by novelty and diversity) in our system, for the specified subscription, the results of filtering will be shown on the right hand-side with a list of items on which we can see its information (title, description, link). Each play of items will generate a new tab containing the new corresponding set of items. This will help users to compare the different sets of items to notify for different parameters.

4.2 Workload Scenario

This scenario of our demonstration focuses on system administration. We would like to show real-time workload and memory space on the whole system by varying all parameters. In the left part of the management interface (Figure 4), all system's parameters (number of subscriptions to index, number of items to match, satisfaction threshold, novelty threshold, take into account diversity or not, and the size of the window to manage histories) can be specified. Also in this part, one of the three functions of our system (indexing, partial matching and filtering by novelty and diversity) can be executed. The right part shows information about the filtering system: memory, workload, subscriptions statistics, thresholds, histories statistics, number of items in the sliding window and emulation speed.

A first sketch will show the indexing phase in our system. We will set the number of subscriptions to index. This setting will show the evolution of memory and workload, according to different number of subscriptions. Generation of subscriptions and corresponding statistics can be seen on the top-left hand side of the interface (memory space used, average indexing time by subscription, number of postings lists and their average size).

Then, a partial matching for a set of items on the index is done. The impact satisfaction threshold will be shown by varying it. For each play, a tab in the right part of the interface is created to visualize the matching time and the number of satisfied subscriptions (top part). This will help users to compare the impact of satisfaction threshold on matching time and number of subscriptions to notify.

The filtering step can be executed after each matching step, by default it is computed for the last matched set of items. The statistics of the filtering is shown in the bottom-right part of the interface. We are interested in shown the processing time of items and the number of subscriptions to notify, in order to compare it to the number of satisfied by the partial matching. Also, we will show the size of the global history, and repartition of histories for subscription (less than 5 items, between 6 and 10, between 11 and 20, and more than 20). The impact of diversity, novelty threshold and size of the window will be shown by several executions of filtering by varying their values.

References

[BJC+04] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. A. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *SIGIR*, pages 321–328, 2004.

[BS74] A. Bookstein and D.R. Swanson. Probabilistic Models for Automatic Indexing. *J. Am. Soc. Inf. Sci.*, 25(5):312–318, 1974.

[BYRN99] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[DP12a] M. Drosou and E. Pitoura. Disc diversity: result diversification based on dissimilarity and coverage. *PVLDB*, 6(1):13–24, 2012.

[DP12b] M. Drosou and E. Pitoura. Dynamic diversification of continuous data. In *EDBT*, pages 216–227, New York, NY, USA, 2012. ACM.

[HKC+12] Z. Hmedeh, H. Kourdounakis, V. Christophides, C. du Mouza, M. Scholl, and N. Travers. Subscription indexes for web syndication systems. In *EDBT*, pages 311–322, 2012.

[HVT+11] Z. Hmedeh, N. Vouzoukidou, N. Travers, V. Christophides, C. du Mouza, and M. Scholl. Characterizing web syndication behavior and content. In *WISE*, pages 29–42, 2011.

[PvA08] K. Pripužić, I.P. Žarko, and K. Aberer. Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w. In *DEBS*, pages 127–138. ACM, 2008.

[SWY75] Gerard Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, 1975.

[Wal77] A.J. Walker. An efficient method for generating discrete random variables with general distributions. *TOMS*, 3:253–256, 1977.

[ZCM02] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, pages 81–88, New York, NY, USA, 2002. ACM.