A Novel Clustering-based RSS Aggregator

Xin Li[1], Jun Yan[2], Zhihong Deng[1], Lei Ji[2], Weiguo Fan[3], Benyu Zhang[2], Zheng Chen[2]

[1]National Laboratory on Machine Perception, School of Electronics Engineering and Computer Science Peking University, Beijing 100871, P.R.China
{lix, zhdeng}@cis.pku.edu.cn

[2]Microsoft Research Asia
{junyan, leiji, byzhang,zhengc}@microsoft.com

[3] Virginia Polytechnic Institute and State University
wgfan@vt.edu

ABSTRACT

In recent years, different commercial Weblog subscribing systems have been proposed to return stories from users' subscribed feeds. In this paper, we propose a novel clustering-based RSS aggregator called as RSS Clusgator System (RCS) for Weblog reading. Note that an RSS feed may have several different topics. A user may only be interested in a subset of these topics. In addition there could be many different stories from multiple RSS feeds, which discuss similar topic from different perspectives. A user may be interested in this topic but do not know how to collect all feeds related to this topic. In contrast to many previous works, we cluster all stories in RSS feeds into hierarchical structure to better serve the readers. Through this way, users can easily find all their interested stories. To make the system current, we propose a flexible time window for incremental clustering. RCS utilizes both link information and content information for efficient clustering. Experiments show the effectiveness of RCS.

Categories and Subject Descriptors

H.4.3 **[Information Systems Applications]**: Communications Applications – *Information browsers.* I.5.3 **[Pattern Recognition]**: Clustering – *Algorithms.*

General Terms: Algorithms, Management.

Keywords: RSS, Weblog, clustering, story.

1. INTRODUCTION

Weblog, or called as blog, is attracting a lot of attentions due to the rapid growth of RSS technology. According to a recent survey made by Technorati [7], there are about 75,000 new RSS feeds and 1.2 million new stories daily. Thus it is impossible for a user to read all new stories related to his/her interested topics. Based on RSS subscribing, several commercial systems [3, 4, 5, 6] have been proposed to help Weblog readers find their information requirements. However, many of recent systems only return the list of all stories come from the subscribed feeds. To better serve the readers, further data organization is highly desire. In this paper we propose a novel clustering-based RSS Clusgator System (RCS) to help blog reading.

RCS clusters all collected stories into a hierarchical structure. If a user subscribes to a group of RSS feeds, the clusters which include the stories of these subscribed feeds are returned to the user. Through this way, the user could brows the returned hierarchies to find their interested topics quickly without going through all subscribed stories. On the other hand, if a user is interested in a particular topic and wants to know how other feeds have discussed the same topic, he can browse the returned hierarchy and find other stories/feeds talking about the same topic. The main technique behind RCS is clustering on the story dataset. The dynamically changing of stories requires the clustering algorithm to have the ability of processing data incrementally. In this work, we propose a flexible time window for incremental story clustering. In addition, to make the clustering procedure efficient, we utilized both link information and content information of stories.

Figure 1 shows the user interface of RCS. It clusters stories from all collected RSS feeds into a hierarchical structure. Each node is denoted by a key-phrase extracted from text documents. If a user subscribes to a group of feeds, RCS returns the specific leaf nodes which represent the stories of these subscribed feeds. If more than one cluster within the same branch of the hierarchy is returned, their parent node should also be returned.

Figure 1. User interface of RCS.

2. RCS ARCHITECTURE

Figure 2 shows the architecture of RCS. There are three main modules, *crawler*, *clustering module* and *topic extraction module*. We have a collection of pre-defined RSS feeds. The stories from all user-subscribed feeds and these pre-defined feeds are daily crawled. The daily crawled data are stored in a story database. The size of this database increases over time. We delete the outdated posts according to a sliding time window which will be introduced in the next Section. Once the story database gets updated, the clustering module runs the incremental clustering algorithm to update clusters of posts. And then RCS can update the hierarchical cluster structure of all stories. After that, the topics of clusters are extracted by the topic extraction module. In this work, [2] is utilized for topic extraction.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

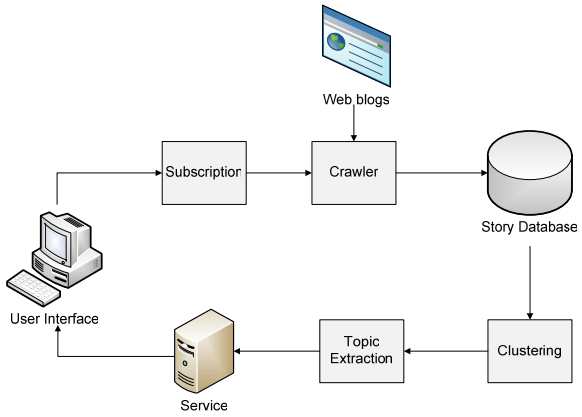

Figure 2. Architecture of RCS.

3. INCREMENTAL STORY CLUSTERING

Note the daily crawled data could be treated as a data stream. When new stories come, they should be clustered to the existent clusters incrementally or be detected as new topics. On the other hand, too outdated posts should be removed to keep the system always current for users. Motivated by these, we introduce a flexible half bounded sliding window which dynamically extends its ending point during a period of time and periodically re-cluster all stories. Figure 3 shows an example of the proposed time window.

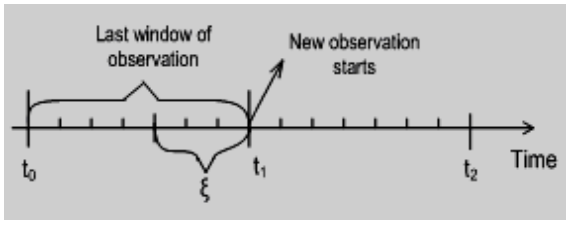

Figure 3. Flexible time window for clustering.

Suppose t_0 is the starting point of the first time window, we crawl all stories at time t_0 as an initial dataset. And then we cluster them into topics using the static clustering algorithm such as K-means. After that the length of this window grows as time goes by. When a group of daily updated new posts are added to the dataset, we combine them into the initial topics or detect them as new topics by the K-nearest neighbor and threshold. In other words, we find the k nearest stories of each new story. If less than k of these similarities are smaller than a threshold, it is treated as a new topic. Otherwise the topic of this new story is voted by the k nearest neighbors. To avoid the window of stories growing infinitely and keep the system current, we set a maximum window length $t_{\max}$ and start a new window at time $t_1 = t_0 + t_{\max}$ to remove the outdated stories posted in the past. When we start a new time window, to capture the potential temporal relationships between old and forthcoming stories, we preserve the posts in a ξ adjacent area instead of removing all posts in last window. We treat posts from $t_1 - \xi$ to t_1 as the initial dataset of new time window for re-clustering. And then the length of the time window is increased from t_1 to $t_1 + t_{\max}$ as time goes by. In this work, all stories are represented by Bag of Words model and the traditional Cosine similarity is utilized.

In this extend abstract, a basic assumption we made is that once there is a link between v_1 and v_2, they have good chance to talk about the same topic. Based on this assumption, we cluster all posts linked together into one topic. And then, we give a threshold which is learned from some training data. If the similarity between two linked stories is smaller than this threshold, we split them into two clusters. On the other hand, if the similarity between the centroids of two clusters is larger than this threshold, we merge them into the same cluster. After cluster the first layer stories, we cluster the second layer of the hierarchy by the same procedure and by treating each cluster as a story.

4. EXPERIMENTS

To evaluate RCS system, we gathered 3000 RSS feeds. The data set consists of about 200,000 pieces of stories collected over a period of two months. The clustering performance of our proposed algorithm is measured by the clustering purity [1]. The average clustering purity of pure content based k-means, our proposed algorithm and the link based algorithm are 0.78, 0.77, 0.31 respectively. Though the pure content based clustering is a little better, its time cost is exponentially increased as the number of stories increase. The time cost of our algorithm is smaller than a constant. To show the user evaluation of our system, we did a user study in contrast to Bloglines.

Table 1. Average interest stories finding time (min).

User	1	2	3	4	5	6	7	8	9	10
Bloglines	58	60	**15**	#	45	30	30	45	20	35
RCS	**20**	**30**	30	#	**15**	**10**	**20**	**15**	**5**	**10**

We ask them to find all their interested stories from these two systems on the same data collection. The average time on each day are shown in Table 1. From this table we can see that to most users, RCS can really help people to save a lot of time on finding their interested stories. Through RCS, users do not need to go through all posts from their subscribed feeds. Instead they can find their interests directly in the topic hierarchy. However, there are two frequent blog readers who can find their interested posts quicker in Bloglines. The first person (number 3) usually was interested in specific feed instead of specific topics. The second person (number 4) did not give the detailed time. He only told Bloglines can save time since he is familiar with it.

For future works, we will continue improve the clustering performance; add more features to the user interface; and develop term extraction algorithm which can reflect the hierarchical information of different layers. In addition, we will try to improve the result ranking within each cluster.

5. REFERENCES

[1] A. Solomonoff, A. Mielke, M. Schmidt, and H. Gish, "Clustering speakers by their voices", Proc. ICASSP'98, pp. 757-760, 1998.

[2] J. L. Chen, B. Y. Zhang, J. Yan, and Q. Yang, "Diverse Topic Phrase Extraction through Latent Semantic Analysis", To appear in proceeding of the IEEE International Conference on Data Mining, 2006.

[3] http://www.bloglines.com/

[4] http://megite.com/

[5] http://tailrank.com/

[6] http://tech.memeorandum.com/

[7] http://www.technorati.com/