Liu YB, Cai JR, Yin J *et al.* Clustering text data streams. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(1): 112–128 Jan. 2008

Clustering Text Data Streams

Yu-Bao Liu[1] (刘玉葆), Jia-Rong Cai[1] (蔡嘉荣), Jian Yin[1] (印　鉴), and Ada Wai-Chee Fu[2] (傅蔚慈)

[1] *Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China*

[2] *Department of Computer Science and Engineering, the Chinese University of Hong Kong, Hong Kong, China*

E-mail: {liuyubao, issjyin}@mail.sysu.edu.cn; kelvin2004_cai@163.com; adafu@cse.cuhk.edu.hk

Received June 25, 2007; revised November 14, 2007.

Abstract Clustering text data streams is an important issue in data mining community and has a number of applications such as news group filtering, text crawling, document organization and topic detection and tracing etc. However, most methods are similarity-based approaches and only use the TF∗IDF scheme to represent the semantics of text data and often lead to poor clustering quality. Recently, researchers argue that semantic smoothing model is more efficient than the existing TF∗IDF scheme for improving text clustering quality. However, the existing semantic smoothing model is not suitable for dynamic text data context. In this paper, we extend the semantic smoothing model into text data streams context firstly. Based on the extended model, we then present two online clustering algorithms OCTS and OCTSM for the clustering of massive text data streams. In both algorithms, we also present a new cluster statistics structure named cluster profile which can capture the semantics of text data streams dynamically and at the same time speed up the clustering process. Some efficient implementations for our algorithms are also given. Finally, we present a series of experimental results illustrating the effectiveness of our technique.

Keywords clustering, database applications, data mining, text data streams

1 Introduction

Clustering text data streams is an important issue in data mining community and has a number of applications such as news group filtering, text crawling, document organization and topic detection and tracing etc.

Can you imagine you check the mails in different categories when you open the email box? Text data streams clustering approach can easily implement such a system. For example, a part of the text streams is given in Fig.1, where four emails continuously arrive in a twenty minutes interval. From Fig.1, we can see that Email1 and Email3 are about the working business that are urgent and need to be handled immediately, whereas Email2 and Email4 are about the entertainment such as gathering and dating. When Kevin checks his mails at 10:30 pm, it would be more convenient for him to reply the emails if the emails can be automatically classified into different categories. Another example of text data streams clustering is the thread detection in text streams in which the Instant

...
Email1 arrives at 10:00 pm. Jan. 18th, 2007:
hello Kevin,
Can you send me the project proposal before 24th this month? Thank you.
Tom.

Email2 arrives at 10:05 pm. Jan. 18th, 2007:
Long time no see Kevin; we have a gathering at 20th, would you like to join us. Reply the mail no matter whether you would come.
Jack.

Email3 arrives at 10:15 pm. Jan. 18th, 2007:
hello Kevin,
The codes of the project have some bugs; we would have an online meeting tomorrow. Please attend on time.
Mr. Robinson.

Email4 arrives at 10:17 pm. Jan. 18th, 2007:
Shall we go to see a movie this weekend?
Marry.
...

Fig.1. Example of text data streams.

Regular Paper

Supported by the National Natural Science Foundation of China under Grant Nos. 60573097, 60703111, 60773198, the Natural Science Foundation of Guangdong Province under Grant No. 06104916, the Specialized Research Foundation for the Doctoral Program of Higher Education under Grant No. 20050558017 and the Program for New Century Excellent Talents in University of China under Grant No. NCET-06-0727.

Messaging (IM) and Internet Relay Chat (IRC) text message streams are classified[1]. In such text data stream applications, text data comes as a continuous stream and this presents many challenges to traditional static text clustering. For example, the whole text data cannot be fit into memory at once and multiple scans of the data kept in secondary storage are not possible due to real-time response requirements etc.[2]

Actually, the clustering problem has recently been studied in the context of numeric data streams and categorical data streams[3−5]. However, the clustering of text data streams is still in early stage. In [6], an online algorithm framework based on traditional numeric data streams clustering is presented for categorical and text data streams. In [7], the single pass clustering method for online event detection of text data streams is presented. This method is also an extension of traditional numeric data streams clustering. In [8], several methods based on three batch topic models are presented for the clustering of text data streams.

Recently, [9] argues that a text document is often full of class-independent "general" words (such as stop words that may be shared by different classes of two documents) and short of class-specific "core" words (such as the related topic words occur in the document), which often leads to poor document clustering quality. The model-based clustering approaches based on semantic smoothing, which is widely used in information retrieval (IR)[10−12], are presented for static text data clustering.

Actually, most existing clustering algorithms for text data streams are similarity-based approaches and often employ the heuristic TF∗IDF scheme to discount the effect of "general" words. As shown in [9], semantic smoothing model is often better than TF∗IDF scheme in improving the clustering quality. However, the existing semantic model is not suitable for the dynamic text context.

In this paper, we extend the semantic smoothing model into text data streams context and try to improve the clustering quality by the semantic smoothing model. To the best of our knowledge, there are no works on this problem. The key contributions of this paper are as follows.

1) We analyze the characteristics of text data streams and point out the existing semantic smoothing model is not suitable for the text data streams context. Then we present an extended semantic smoothing model for the text data streams (please see Section 2).

2) Based on the extended semantic smoothing model, we present two online clustering algorithms OCTS (stands for online clustering of text streams) and OCTSM (stands for online clustering of text streams with merge) for the clustering of massive text data streams. In both algorithms, we also present a new cluster statistics structure named cluster profile, which can capture the semantics of text data streams dynamically and at the same time speed up the clustering process. Some efficient implementations for our algorithms are also given.

3) We have performed a series of experiments to compare our technique with some existing algorithms on the 20ng-newsgroups (20NG) corpus dataset, which is widely used in the text data clustering. The experimental results illustrate the effectiveness of our technique①.

1.1 Related Work

There are some related works on our problem. The existing static text clustering methods (or referred to as batch text clustering) are related to our work including the spherical k-means, the bisecting k-means, and the documents model-based clustering etc.[14,15] Those methods are the basis of our work.

The data streams clustering methods are also related. In [5, 16], based on the traditional k-means, S. Guha *et al.* firstly propose the data streams clustering algorithm named STREAM. In STREAM, the centroid is used to represent the clusters, and a layered clustering strategy is used to enhance the algorithm efficiency. Different from STREAM, we use cluster profile, which is a kind of statistics structure, to represent the clusters. In [3], Aggrawal *et al.* propose a novel data stream clustering framework called as CluStream, which views the data streams clustering as an evolutionary process with time. This framework includes two sub-processes, that is, the online process and offline process. That is similar to our algorithms in which the online process and offline process are also included. But, our algorithms aim to handle text data streams that consist of more complicated semantics than the general data streams. Similar to [3], a projected data stream clustering algorithm called HPStream is presented for the clustering of high dimensional numeric data stream. In [6], the concept of cluster droplet is used to store the real-time condensed cluster statistics information. When a document comes, it would be assigned to the suitable cluster and then the corresponding cluster droplet is updated. In clustering process, they utilize the online

①The initial results on OCTS also appear in [13].

version of the traditional TF∗IDF scheme called incremental TF∗IDF to dynamically represent the streaming documents. Similar to [6], our cluster profile is also a kind of statistics information structure. However, in cluster profile, our extended semantic smoothing model, which is not used in the cluster droplet, captures the semantics. The incremental TF∗IDF strategy is also used in [7]. Different from [6], the time window is used to describe the changes of text data streams in [7]. Similar to [6], the fading functions are also used in our algorithms. Different from [6, 7], the static spherical k-means text clustering method is extended for online text data streams clustering[17], and it only keeps cluster centers instead of cluster droplets. When a document comes, it would not be clustered at once. Instead, it is accumulated as a portion of a segment. When a segment (a fixed number of documents) forms, the online spherical k-means algorithm would be employed and then the cluster centers are updated.

In [18], a feature-pivot clustering technique is presented for the detection of a set of bursty events from a text stream. They fully utilize the time information to determine a set of bursty features, which may occur in different time windows based on the feature distributions. Based on [18], a dynamical document representation method based on bursty features is applied in the text streams environment[19]. But both assume the text stream is a finite process, which contains a fixed number of documents. Different from [18, 19], our algorithms are used to classify the document topics but the bursty features from the text data streams in which the documents continuously come.

Recently, [8] proposed a practical heuristic for hybrid topic modeling, which learns online topic models on streaming text and intermittently runs batch topic models on aggregated documents offline. Compared to the semantic smoothing model, the hybrid topic models including vMF, LDA and DCM are more complicated. In addition, the time window is used to describe the changes of text data streams, which is also different from our work in which the fading function is used. In [20], an efficient extraction algorithm is presented for the text data streams with noise.

There are also some other related works. The work in [21–23] focuses on the problem of concept drift of data streams. In [24], a new mining problem called temporal text mining is studied, whose purpose is to discover and summarize the evolutionary theme patterns in text data streams context. In [25–29], some interesting problems on data stream mining such as the approximate querying, frequent itemsets mining etc. are discussed. In [30, 31], the multi-dimensional data stream processing is discussed as the problem of stream cube. In [32], we also present an extended semantic smoothing model for the static text data clustering.

1.2 Paper Organization

The rest of this paper is organized as follows. The existing semantic smoothing model and its extension are given in Section 2. In Section 3, we present the clustering algorithms OCTS and OCTSM based on the extended semantic smoothing model. Some efficient implementations for our algorithms are also given in this section. The experimental results are reported in Section 4. The conclusions are drawn in Section 5.

2 Semantic Smoothing Model

Many previous approaches use word extraction method and single word vector as the document features. However, they suffer from the context-insensitivity problem. The terms in these models may have ambiguous meanings. In contrast, the semantic smoothing model uses the multiword phrases as topic signatures (document features). For example, the multiword phrase "fixed star" (denotes a celestial body) has clearer meaning than the single word "star" (denotes either a celestial body or a pop star).

After phrase extraction, the training process determines the probability of translating the given multiword phrase to terms in the vocabulary. For example, if the word "planet" frequently appears in the documents whose topic contains "fixed star", then "fixed star" and "planet" must have some specific relationship. The translation model finds out such relationship and assigns a degree to describe it. In the following process (e.g., clustering), if we encounter a document contains the topic signature "fixed star" (but not "planet"), we can also assign a rational probability count to the word "planet" for the document.

For each phrase t_k, it would have a set of documents (D_k) containing that phrase. Since not all words in D_k center on the topic signature t_k, we assume D_k is generated by a mixture language model (i.e., all terms in the document set are either translated by the given topic signature model $p(w|t_k)$ or generated by the background collection model $p(w|C)$). (1)∼(3) are used to iteratively compute the translation probabilities[9,10].

$$p(w|D_k) = (1-\beta)p(w|t_k) + \beta p(w|C), \quad (1)$$

$$\hat{p}^{(n)}(w) = \frac{(1-\beta)p^{(n)}(w|t_k)}{(1-\beta)p^{(n)}(w|t_k) + \beta p(w|C)}, \quad (2)$$

$$p^{(n+1)}(w|t_k) = \frac{c(w, D_k)\hat{p}^{(n)}(w)}{\sum_i c(w_i, D_k)p^{(n)}(w_i)}. \quad (3)$$

Here, β is a coefficient accounting for the background noise, $p(w|t_k)$ denotes the translation model for topic signature t_k, $c(w, D_k)$ is the frequency count of term w in document set D_k (means the number of appearance of w in D_k), n is the number of iterations, and C denotes the background collection, which is the set of all word occurrences in the corpus. In practice, the EM algorithm is used to estimate the translation model in (2) and (3).

The cluster model with semantic smoothing (or referred to as semantic smoothing model) is estimated using a composite model $p_{bt}(w|c_j)$, which means the likelihood of each vocabulary word w generated by a given document cluster c_j after smoothing. It has two components: a simple language model $p_b(w|c_j)$ and a topic signature (multiword phrase) translation model $p_t(w|c_j)$. The influence of two components is controlled by the translation coefficient (λ) in the mixture model.

$$p_{bt}(w|c_j) = (1-\lambda)p_b(w|c_j) + \lambda p_t(w|c_j), \quad (4)$$

$$p_b(w|c_j) = (1-\alpha)p_{ml}(w|c_j) + \alpha p(w|C), \quad (5)$$

$$p_t(w|c_j) = \sum_k p(w|t_k)p_{ml}(t_k|c_j). \quad (6)$$

In simple language model (5), α is a coefficient controlling the influence of the background collection model $p(w|C)$ and $p_{ml}(w|c_j)$ is a maximum likelihood estimator cluster model. They can be computed using (7) and (8). In translation model (6), t_k denotes the topic signatures (multiword phrases) extracted from documents in cluster c_j. The probability of translating t_k to individual term (word) is estimated using (1)$\sim$(3). The maximum likelihood estimator of t_k in cluster c_j can be estimated with (9). In (7), (8), and (9), $c(w, c_j)$ denotes the frequency count of word w in cluster c_j, $c(w, C)$ is the frequency count of word w in the background corpus, and $c(t_k, c_j)$ is the frequency count of topic signature t_k (multiword phrase) in cluster c_j. The function of the translation model is to assign reasonable probability to an unseen core word in a given cluster.

$$p_{ml}(w|c_j) = \frac{c(w, c_j)}{\sum_{w_i \in c_j} c(w_i, c_j)}, \quad (7)$$

$$p(w|C) = \frac{c(w, C)}{\sum_{w_i \in C} c(w_i, C)}, \quad (8)$$

$$p_{ml}(t_k|c_j) = \frac{c(t_k, c_j)}{\sum_{t_i \in c_j} c(t_i, c_j)}. \quad (9)$$

Due to the likelihood of word w generated by a given cluster $p(w|c_j)$ can be obtained by the cluster model with semantic smoothing, the remaining problem for the clustering of text document is how to estimate the likelihood of a document d generated by a cluster. The log likelihood of document d generated by the j-th multinomial cluster model is described in (10), where $c(w, d)$ denotes the frequency count of word w in document d and V denotes the vocabulary.

$$\log p(d|c_j) = \sum_{w \in V} c(w, d) \log p(w|c_j). \quad (10)$$

Compared to semantic smoothing model, traditional similarity-based approaches just use the technique of frequency count of words (which is similar to the simple language model $p_b(w|c_j)$) and does not take into account the translation model $p_t(w|c_j)$). As shown in [9], semantic smoothing model is efficient to improve the clustering quality for traditional static text document clustering. However, it cannot be directly used in the dynamical text data streams environment. The key reason is that, in text data streams, the text data comes as a continuous stream and it is hard to get the background collection model of all document of text data streams in advance. That is, it is hard to determine $p(w|C)$ in $p_b(w|c_j)$.

In this paper, we present an extended semantic smoothing model in which the background model is not included and set coefficient α as zero. Then we define the semantic smoothing model as follows.

$$p_{bt}(w|c_j) = (1-\lambda)p_{ml}(w|c_j) + \lambda p_t(w|c_j). \quad (11)$$

From (11), we can see that the extended semantic smoothing model also consists of two components, one is $p_{ml}(w|c_j)$ (which consists of frequency count of words) and the other is

$$p_t(w|c_j) = \sum_k p(w|t_k)p_{ml}(t_k|c_j).$$

3 Proposed Online Clustering Algorithms

3.1 Basic Concepts

In text data streams environment, text document comes as a continuous stream. In order to account for the evolution of the data stream, we assign a time-sensitive weight to each document. It is assumed that each data point has a time-dependent weight defined by the function $f(t)$. The function $f(t)$ is also referred to as the fading function. The fading function $f(t)$ is a monotonic decreasing function which decays uniformly

with time t. In order to formalize this concept, we will define the half-life of a point in the data stream.

Definition 1 (Half Life). *The half life t_0 of a point is defined as the time at which $f(t_0) = (1/2)f(0)$.*

The aim of defining a half life is to define the rate of decay of the weight associated with each data point in the stream. The decay-rate is defined as the inverse of the half life of the data stream.

Similar to [6], we denote the decay rate by $\zeta = 1/t_0$ and the fading function is defined as follows.

Definition 2 (Fading Function). *Consider the time point t, the fading function value is defined as $f(t) = 2^{-\zeta t}$, where $\zeta = 1/t_0$ and t_0 is the half life of the stream.*

3.2 Cluster Statistics Structure

In order to achieve greater accuracy in the clustering process, we also maintain a high level of granularity in the underlying data structures. We refer to such cluster statistic structure as cluster profile in which the semantics are captured by the extended semantic smoothing model. In cluster profile, considering the evolution characteristics of text data streams, we associate the components $p_{ml}(w|c_j)$ and $p_t(w|c_j)$ in (11) with weighted factor $f(t - T_d)$, where $T_d$is the arrival time of document d and t is the current time. Then we get the following formula.

$$\begin{aligned} p'_{bt}(w|c_j) &= (1-\lambda)\sum_{d\in c_j} f(t-T_d)p_{ml}(w|c_j) \\ &\quad + \lambda \sum_{d\in c_j} f(t-T_d)p_t(w|c_j) \\ &= (1-\lambda)\frac{\sum_{d\in c_j} f(t-T_d)c(w,d)}{\sum_{w_i\in c_j}\sum_{d\in c_j} f(t-T_d)c(w_i,d)} \\ &\quad + \lambda \sum_k p(w|t_k) \\ &\quad \cdot \frac{\sum_{d\in c_j} f(t-T_d)c(t_k,d)}{\sum_{t_i\in c_j}\sum_{d\in c_j} f(t-T_d)c(t_i,d)} \\ &\quad \text{(using (6), (7) and (9))} \\ &= (1-\lambda)\frac{\sum_{d\in c_j} f(t-T_d)c(w,d)}{\sum_{w_i\in c_j}\sum_{d\in c_j} f(t-T_d)c(w_i,d)} \\ &\quad + \lambda \frac{\sum_k p(w|t_k)\sum_{d\in c_j} f(t-T_d)c(t_k,d)}{\sum_{t_i\in c_j}\sum_{d\in c_j} f(t-T_d)c(t_i,d)}. \end{aligned}$$

Based on this result, we introduce the following definitions.

Definition 3 (Weighted Sum of Frequency Count). *The weighted sum of frequency count for word w_i in cluster c is defined as $w\_c(w_i, c) = \sum_{d\in c} f(t - T_d)c(w_i, d)$. Here, $c(w_i, d)$ denotes the frequency count of w_i in document d, T_d is the arrival time of document d and $f(t - T_d)$ is the weight for word w_i in document d. Similarly, the weighted sum of frequency count for topic signature t_k in the cluster c is defined as $w\_c(t_k, c) = \sum_{d\in c} f(t-T_d)c(t_k, d)$, where $c(t_k, d)$ denotes the frequency count of t_k in document d.*

Definition 4 (Weighted Sum of Translation). *The weighted sum of topic signature translation probability in cluster c for word w_i is defined as*

$$\begin{aligned} w\_t(w_i, c) &= \sum_k p(w_i|t_k)w\_c(t_k, c) \\ &= \sum_k p(w_i|t_k)\Big(\sum_{d\in c} f(t-T_d)c(t_k,d)\Big). \end{aligned}$$

Here, $c(t_k, d)$ denotes the frequency count of topic signature t_k in document d, $p(w_i|t_k)$ denotes the probability of translating topic signature t_k to word w_i, T_d is the arrival time of document d and $f(t - T_d)$ is the weight for topic signature t_k in document d.

Based on the Definition 3 and Definition 4, we give the following definition of cluster profile.

Definition 5 (Cluster Profile, CP). *A cluster profile $D(t, c)$ for a document cluster c at time t is defined as a quadruple ($\overline{\boldsymbol{DF2}}$, $\overline{\boldsymbol{DF1}}, s, l$), and consider wb denotes the number of distinct words in the dictionary V, and each vector is defined as follows.*

- *The vector $\overline{\boldsymbol{DF2}}$ contains wb entries and the i-th entry $\overline{DF2}_i$ is defined as $w\_c(w_i, c)$, $1 \leqslant i \leqslant wb$.*
- *The vector $\overline{\boldsymbol{DF1}}$ contains wb entries and the i-th entry $\overline{DF1}_i$ is defined as $w\_t(w_i, c)$, $1 \leqslant i \leqslant wb$.*
- *$s = \sum_k w\_c(t_k, c)$ denotes the summation of $w\_c(t_k, c)$ for all the topic signature t_k in the cluster c.*
- *The entry l denotes the last time when cluster c is updated (i.e., add some documents into c).*

Then we can estimate the cluster model with semantic smoothing using (12) in which $\overline{DF2}_i/\sum_i \overline{DF2}_i$ and $\overline{DF1}_i/s$ denote the weighted form of the components $p_{ml}(w|c_j)$ and $p_t(w|c_j)$ respectively.

$$p'_{bt}(w_i|c_j) = (1-\lambda)\frac{\overline{DF2}_i}{\sum_i \overline{DF2}_i} + \lambda\frac{\overline{DF1}_i}{s}. \qquad (12)$$

Interestingly, we find that *CP* structure also has some similar properties for clustering process as the cluster droplet[6].

Property 1 (Additivity). *Additivity describes the variation of cluster profile after two clusters c_1 and c_2 are merged as $c_1 \cup c_2$. Suppose two cluster profiles are $D(t, c_1) = (\overline{\boldsymbol{DF2}}(c_1), \overline{\boldsymbol{DF1}}(c_1), s_{c_1}, l_{c_1})$*

and $D(t,c_2) = (\overline{\boldsymbol{DF2}}(c_2), \overline{\boldsymbol{DF1}}(c_2), s_{c_2}, l_{c_2})$. *Then* $D(t, c_1 \cup c_2) = (\overline{\boldsymbol{DF2}}(c_{12}), \overline{\boldsymbol{DF1}}(c_{12}), s_{c_{12}}, l_{c_{12}})$ *can be defined by tuple* $(\overline{\boldsymbol{DF2}}(c_1) + \overline{\boldsymbol{DF2}}(c_2), \overline{\boldsymbol{DF1}}(c_1) + \overline{\boldsymbol{DF1}}(c_2), s_{c_1} + s_{c_2}, \max(l_{c_1}, l_{c_2}))$.

Property 2 (Updatability). *Updatability describes the variation of cluster profile after a new document is added into the clusters. Suppose a new document d is merged into a cluster c, the current cluster profile* $D_b(t, c_b) = (\overline{\boldsymbol{DF2}}(c_b), \overline{\boldsymbol{DF1}}(c_b), s_{cb}, l_{cb})$. *Then the updated cluster profile is denoted as* $D_a(t, c_a) = (\overline{\boldsymbol{DF2}}(c_a), \overline{\boldsymbol{DF1}}(c_a), s_{c_a}, l_{c_a})$. *Here* $\overline{DF2}(c_a)_i = \overline{DF2}(c_b)_i + c(w_i, d), \overline{DF1}(c_a)_i = \overline{DF1}(c_b)_i + \sum_k p(w_i|t_k)c(t_k, d), s_{c_a} = s_{c_b} + \sum_k c(t_k, d)$ *and* $l_{c_a} = t$.

Actually, Property 2 can also be viewed as the special case of Property 1, in which one of the two clusters to be merged only consists of one document.

Property 3 (Fading Property). *Fading Property describes the variation of cluster profile with time. Consider the cluster profile at the time* $t1$ *is* $D(t1, c_{t1}) = (\overline{\boldsymbol{DF2}}(c_{t1}), \overline{\boldsymbol{DF1}}(c_{t1}), s_{c_{t1}}, l_{c_{t1}})$ *and there is no document added to the cluster* c_{t1} *during* $[t1, t2]$. *Then the cluster profile at the time* $t2$ *is defined as* $D(t2, c_{t2}) = (\overline{\boldsymbol{DF2}}(c_{t2}), \overline{\boldsymbol{DF1}}(c_{t2}), s_{c_{t2}}, l_{c_{t2}})$, *where* $c_{t2} = c_{t1}$, $\overline{\boldsymbol{DF2}}(c_{t2}) = \overline{\boldsymbol{DF2}}(c_{t1}) \times 2^{-\zeta(t2-t1)}$, $\overline{\boldsymbol{DF1}}(c_{t2}) = \overline{\boldsymbol{DF1}}(c_{t1}) \times 2^{-\zeta(t2-t1)}$, $s_{c_{t2}} = s_{c_{t1}} \times 2^{-\zeta(t2-t1)}$ *and* $l_{c_{t2}} = l_{c_{t1}}$.

3.3 Online Clustering Algorithm OCTS

Our online clustering algorithm OCTS includes two phases: 1) offline initialization process, 2) online clustering process. The detailed description of OCTS algorithm framework is given in Fig.2.

The offline initialization process corresponds to lines 1∼2. In detail, OCTS first reads in the retrospective documents stored in disk as the training text dataset. From the training document set, OCTS generates the topic signature translation model. In our algorithm implementation, the topic signature translation model is estimated by matrix $\boldsymbol{M}(wb \times tb)$, where wb denotes the number of vocabulary words and tb denotes the number of topic signatures, and the data element M_{ik} of matrix represents $p(w_i|t_k)$. The extraction process of words and phrase are similar to [9], that is, we use Xtract to identify phrases in documents. Xtract is a kind of statistical extraction tool with some syntactic constraints. Then OCTS reads in the first k documents from the text data streams and build the initial cluster profiles *CP*s (each for one cluster) using Definition 5.

The online clustering process corresponds to lines 3∼21. In this process, as a new text document arrives, firstly all the *CP*s would be updated using Property 3. Next, the probability $p'_{bt}(w_i|c_j)$ is computed by (12). Then the similarity between document d and each cluster is estimated using (10). Then if the similarity between document d and its most similar cluster is less than specified threshold *MinFactor*, the most inactive cluster is deleted, and a new cluster is created. We associate the new cluster with the ID same to the deleted cluster's ID. Then the document d is assigned to the new cluster using Property 2. Otherwise, document d would be assigned to the most similar cluster using Property 2. In detail, Steps 5∼6 are to update all *CP* using the fading property, and Steps 7∼9 are the online building of the cluster model. Steps 11∼13 are to find the most similar cluster to document d, Steps 15∼18 are to delete the most inactive cluster and create a new cluster (Step 15 is to get the most inactive cluster's ID, if there are more than one inactive clu-

Algorithm. OCTS
Inputs: A stream of text data, D, k is the number of clusters, *MinFactor* is the threshold.
Output: A set of k cluster profiles $(D(t, c_1), \ldots, D(t, c_k))$
Method:
1. Extract words and multiword phrases, and build the translation model for the training data set D.
2. Read in the first k documents and generate k cluster profiles $(D(t, c_1), \ldots, D(t, c_k))$;
3. **While** (the stream is not empty) **do**
4. **Begin**
5. t = GetCurrentTimestamp ();
6. Update all cluster profiles with t;
7. **For** each cluster j **do**
8. **For** each word w_i **do**
9. The cluster model with semantic smoothing $p'_{bt}(w_i|c_j)$ is estimated;
10. Read in the next document d and extract words and multiword phrases from d;
11. **For** each cluster j **do**
12. The similarity $p(d|c_j)$ between d and c_j is estimated;
13. $AssignID = \text{argmax}_j p(d|c_j)$;
14. **If** $(p(d|c_{assignID}) < MinFactor)$ **then** {
15. $NID = \text{argmin}_j D(t, c_j).l$;
16. Delete the most inactive cluster c_{NID};
17. Create a new empty cluster with $ID = NID$ and build its cluster profile;
18. Assign document d to the new cluster profile; }
19. **Else**
20. Assign document d to cluster profile with $ID = AssignID$;
21. **End**

Fig.2. Description of OCTS algorithm.

sters then we randomly choose one). Step 20 is to assign document d to its most similar cluster.

3.4 Further Improvements of OCTS

As shown in our experiments, OCTS algorithm can achieve the good clustering quality. However, we also find that the clustering accuracy of OCTS varies largely with the evolution of the text data streams. We attribute this phenomenon to the frequent deletion of the most inactive cluster (i.e., the historical data) when a new cluster needs to be created. Although the historical data is less important for the dynamic text data streams, it does not mean they are completely useless. Intuitively, if the historical data are very similar to the most recent data, they should not be immediately deleted. Based on this observation, we propose another extended algorithm OCTSM, which is showed in Fig.3.

Algorithm. OCTSM (The inputs and outputs are the same to OCTS with an additional input argument *MergeFactor*)
Method:
1. Extract words and multiword phrases, and build the translation model for the training data set D.
2. Read in the first k documents and generate k cluster profiles $(D(t, c_1), \ldots, D(t, c_k))$;
3. **While** (the stream is not empty) **do**
4. **Begin**
5. t = GetCurrentTimestamp ();
6. Update all cluster profiles with t;
7. **For** each cluster j **do**
8. **For** each word w_i **do**
9. The cluster model with semantic smoothing $p'_{bt}(w_i|c_j)$ is estimated;
10. **If** Merge(*MergeFactor*) then goto Step 22.
11. Read in the next document d and extract words and multiword phrases from d;
12. **For** each cluster j **do**
13. The similarity $p(d|c_j)$ between d and c_j is estimated;
14. $AssignID = \text{argmax}_j p(d|c_j)$;
15. **If** $(p(d|c_{assignID}) < MinFactor)$ **then** {
16. $NID = \text{argmin}_j D(t, c_j).l$;
17. Delete the most inactive cluster c_{NID};
18. Create a new empty cluster with $ID = NID$ and build its cluster profile;
19. Assign document d to the new cluster profile; }
20. **Else**
21. Assign document d to cluster profile with $ID = AssignID$;
22. **End**

Fig.3. Description of OCTSM algorithm.

The key framework of OCTSM is similar to that of OCTS. The key difference is that, in OCTSM, we introduce a new Merge process with a new argument *MergeFactor* (Step 10 in Fig.3) and use this process to determine the similar clusters. In detail, before assignment, the similarity between each existing cluster pairs is calculated. If there exist two highly similar clusters, i and j, they would be merged as a new cluster i using Property 1 and document d would be classified into a new created cluster with the ID = j using Property 2 (see the description of Merge process in Fig.4). After that, the OCTSM algorithm restarts a new loop since the new document has been assigned. The purpose of the merge process is to give the inactive cluster a second chance. If an inactive cluster is highly similar to a recent cluster, it means the inactive cluster is not useless and should not be ignored.

In this situation, the merge operation help the inactive cluster to update its time stamp, which actually reactivates the historical data. With the introduction of the merge operation, the clustering process would become more stable and the clustering accuracy is further enhanced since in most text streams the historical text can help the recent clustering process to determine the cluster more effectively.

Algorithm. Merge (*MergeFactor*)
Method:
1. Calculate the similarity between cluster i and cluster j $(1 \leqslant i \neq j \leqslant k)$ using the following formula $(\Delta(c_i, c_j) \equiv \sum_{w \in V} p(w|c_i) \log [p(w|c_i)/p(w|c_j)]$ and $\Delta(c_j, c_i) \equiv \sum_{w \in V} p(w|c_j) \log [p(w|c_j)/p(w|c_i)]$;
2. **If** $(1/\min(\Delta(c_i, c_j), \Delta(c_j, c_i)) > MergeFactor)$ {
3. Clusters i and j are merged as new cluster i using Property 1;
4. A new empty cluster with ID = j is created using Definition 5 and d is assigned to cluster j using Property 2;
5. Update cluster i and j with Property 3;
6. **Return** true;}
7. **Return** false;

Fig.4. Description of OCTSM algorithm.

3.5 Efficient Implementation for Clustering Algorithms

The main difficulty for our proposed methods comes from the storage and access of the translation matrix $\boldsymbol{M}$. It is impossible to store the whole topic signature translation probability matrix $\boldsymbol{M}$ as a hard disk file in advance. In addition, in the text data streams context, frequency hard disk accesses are infeasible and that would lead to the delayed response.

We observe that for the corpus, especially for the corpus that has the similar topics in different single documents, the visited items in the matrix would probably be visited again soon. Inspired by the principle of the associate memory in operating system, we propose an effective solution for our algorithm implementations.

The solution consists of three layers: associate memory (stored in cache), main memory and a text file (stored in hard disk). The most frequently used phrase and word pairs are stored in the associate memory, and the other parts are stored in main memory and hard disk (as a text data file) respectively. Initially, we randomly select the frequent and non-frequent items from the data dictionary. We apply the bisearch method for the fast access of the main memory.

The mapping process is used to access the different layers and described as follows. When we need to use a word and phrase pair $\langle w_i, t_k \rangle$, we first try to match it with the pairs in the associate memory using hash mapping method. If matched, the translation probability is returned. If not, we would further search in the main memory using the bisearch method. If matched, the translation probability is returned and it would replace the item in the associate memory using the FIFO (first in first out) strategy. If we cannot find any matched pair in the associate memory and main memory, we would search in the disk file. If matched, the translation probability is returned and it would replace the item in the main memory using the FIFO strategy. If not found, it means we encounter a new multiword phrase. Then we would update the matrix. In detail, we simply get all the words from the phrase and add the phrase and word pairs into the part of matrix in associate memory. For example, if "fixed star" is a new phrase, then the pairs ⟨fixed star, fixed⟩ and ⟨fixed star, star⟩ with an initial translation probability (in order to emphasize the added phrase we can set the probability as a constant "1") would be added into the associate memory.

4 Experimental Results

In the experimental studies, we compare our text streams clustering method with the framework proposed in [6] (denoted as Ostc) and the single pass clustering method in [7] (denoted as SPstc) in terms of clustering accuracy, stability and efficiency. All clustering algorithms are implemented by Java 1.4. Our experiment is performed on AMD 1.60GHz CPU, 240MB memory, 40GB hard disk and Window XP OS. The normalized mutual information (NMI) evaluation function is used to evaluate the clustering quality, and 20ng-newsgroups (20*NG*) (20 000 documents) corpus[9] is used as the test dataset. To simulate the real stream environment, we randomly give every document an arrival timestamp and create three different text data streams with different document sequence (denoted as Stream 1, Stream 2 and Stream 3). By default, in our experiments, we randomly choose 500 documents as the training set and set the translation coefficient $\lambda = 0.6$, cluster number $k = 20$, stream speed is 100doc/s, half life = 1s and $MinFactor = 0.05$, $MergeFactor = 0.4$.

4.1 Results of Clustering Accuracy

The first set of experiments is about the clustering quality with different streams and the results are given in Figs. 5∼10. In this set of experiments, there are two kinds of results with different test granularities.

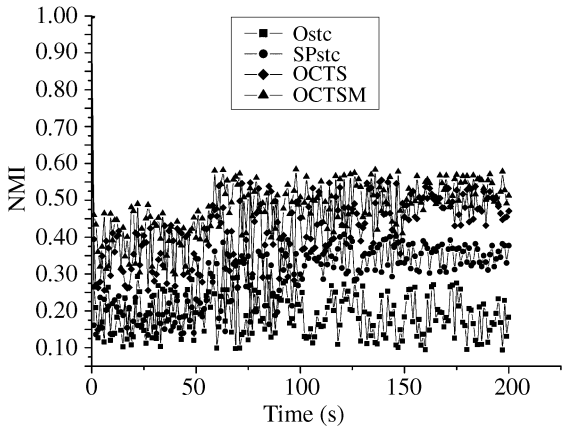

Fig.5. Clustering quality comparison of Stream 1 (fine granularity).

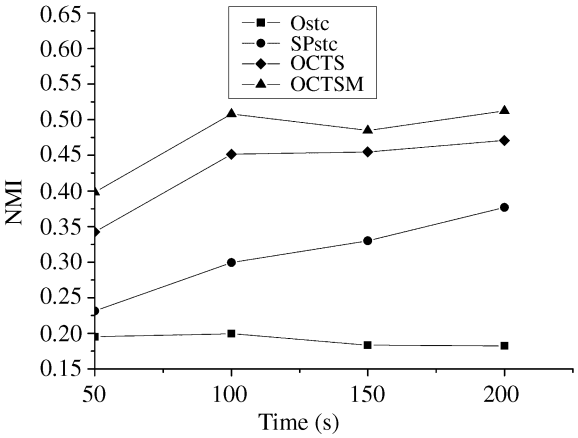

Fig.6. Clustering quality comparison of Stream 1 (coarse granularity).

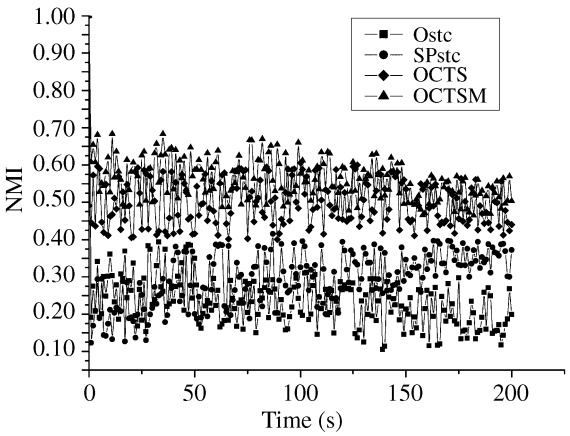

Fig.7. Clustering quality comparison of Stream 2 (fine granularity).

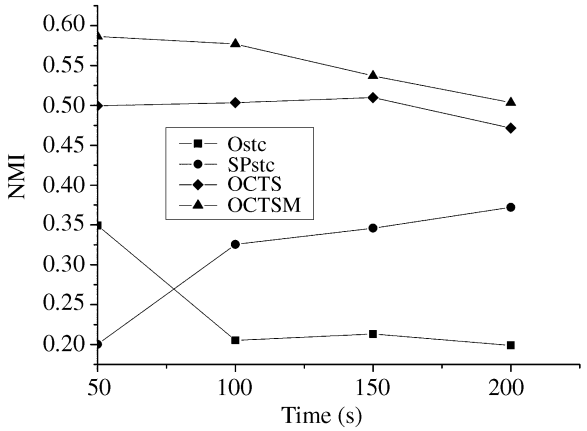

Fig.8. Clustering quality comparison of Stream 2 (coarse granularity).

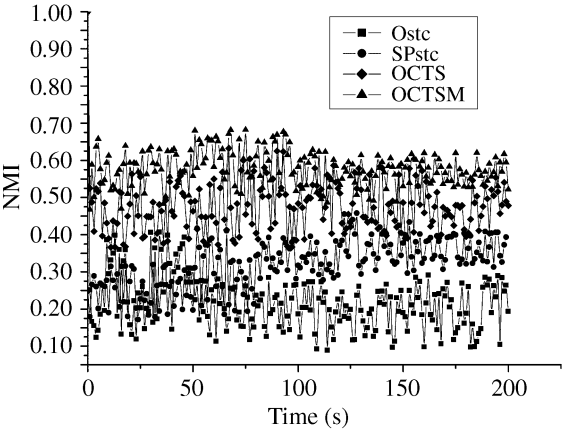

Fig.9. Clustering quality comparison of Stream 3 (fine granularity).

In Figs. 5, 7 and 9, the NMI value temporal variance is set as fine granularity, that is, the NMI values are compared by every 1 second interval. In Figs. 6, 8 and 10, we compare the NMI value in coarser granularity, and the NMI value is estimated by every 50 seconds interval.

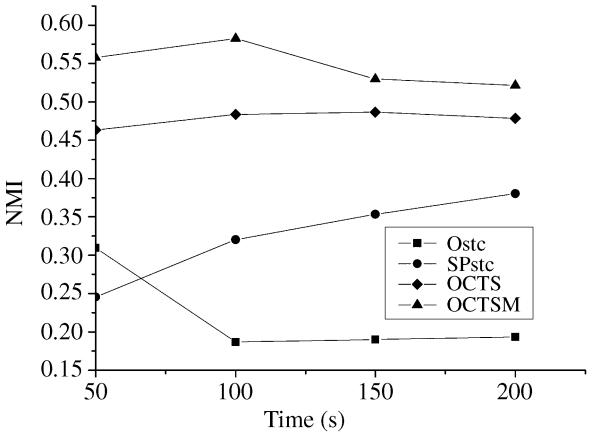

Fig.10. Clustering quality comparison of Stream 3 (coarse granularity).

From the above results, it is easy to know our proposed method OCTS obviously outperforms Ostc and SPstc in which both TF[6] and incremental IDF[7] schemes are used. The results show the effectiveness of semantic smoothing model for improving the clustering quality. For the results, it is also known that OCTSM can yield higher clustering accuracy and is more stable than OCTS, which shows the merge process in OCTSM is effective.

4.2 Results of Clustering Accuracy with Translation Coefficient

The second set of experiments about the clustering quality of OCTS with different translation coefficient (λ) and different text data streams is also studied. The results for OCTS are given in Figs. 11∼14. The results for OCTSM are given in Figs. 15∼18. From the results, it is easy to know that the NMI values of OCTS and OCTSM with different λ are also better than the NMI values of Ostc and SPstc (referred to Figs. 6, 8 and 10). In general, from these results, we can also know that NMI values of OCTS and OCTSM will increase with the increase of the translation coefficient till the peak point (between 0.2 and 0.6) and then go downward. That is, our method is most effective as λ is between 0.2 and 0.6.

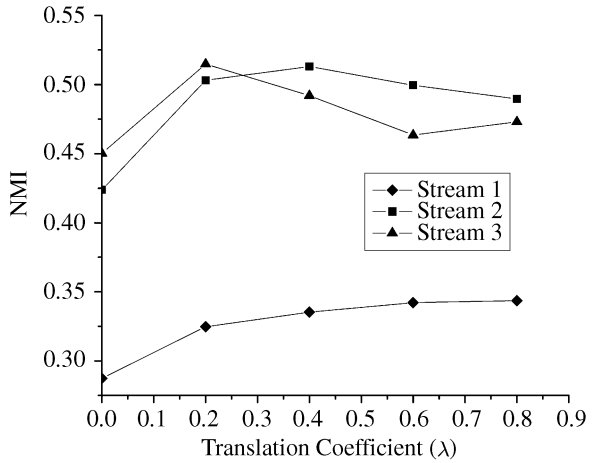

Fig.11. Change of clustering quality with λ of OCTS (test interval: 50 seconds).

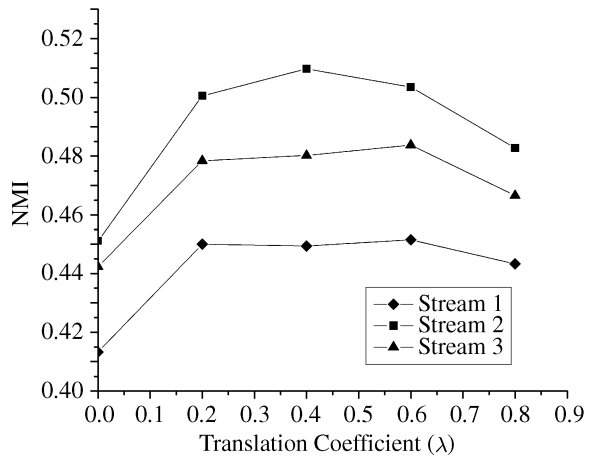

Fig.12. Change of clustering quality with λ of OCTS (test interval: 100 seconds).

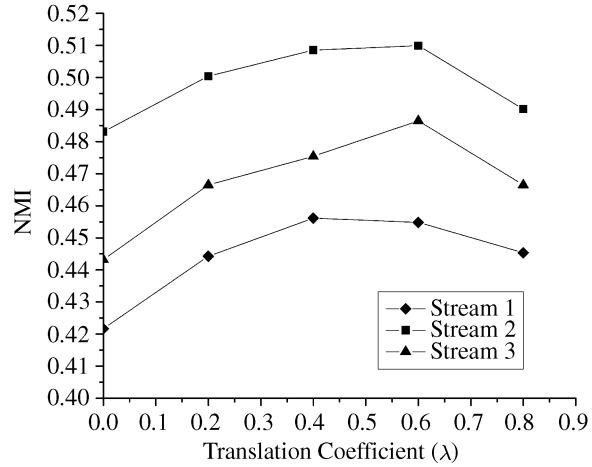

Fig.13. Change of clustering quality with λ of OCTS (test interval: 150 seconds).

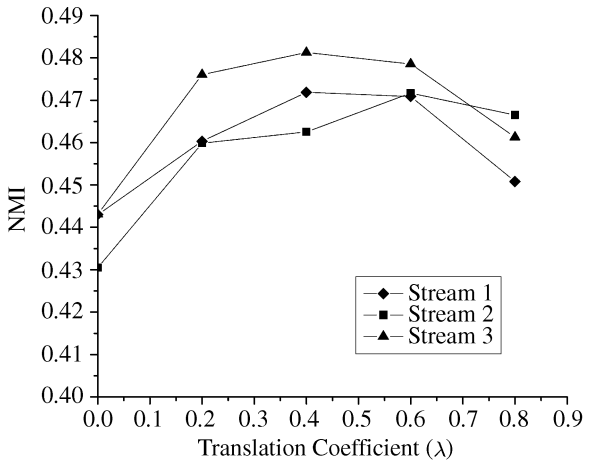

Fig.14. Change of clustering quality with λ of OCTS (test interval: 200 seconds).

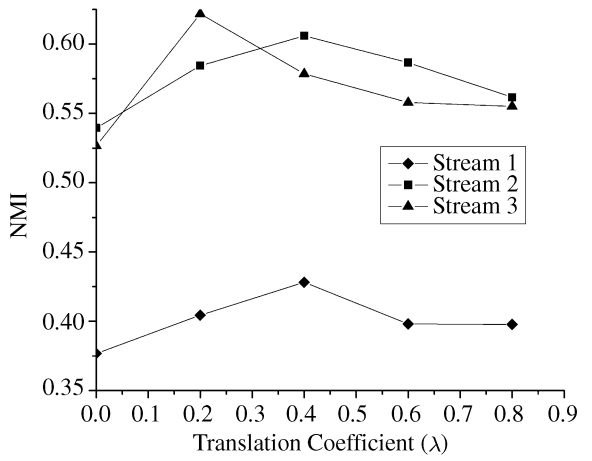

Fig.15. Change of clustering quality λ of OCTSM (test interval: 50 seconds).

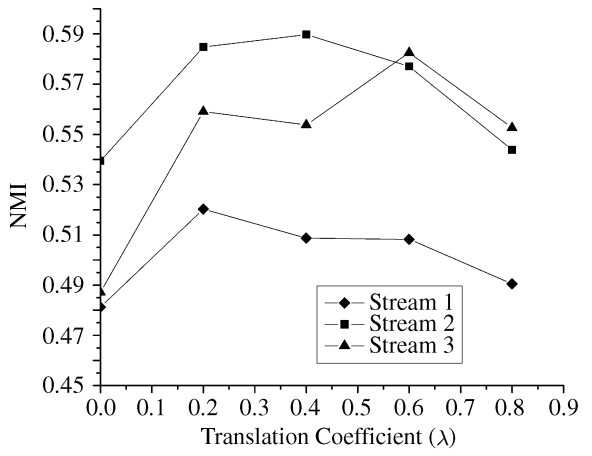

Fig.16. Change of clustering quality with λ of OCTSM (test interval: 100 seconds).

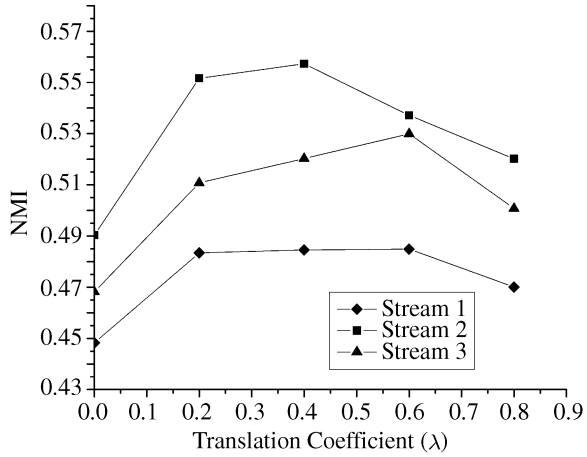

Fig.17. Change of clustering quality with λ of OCTSM (test interval: 150 seconds).

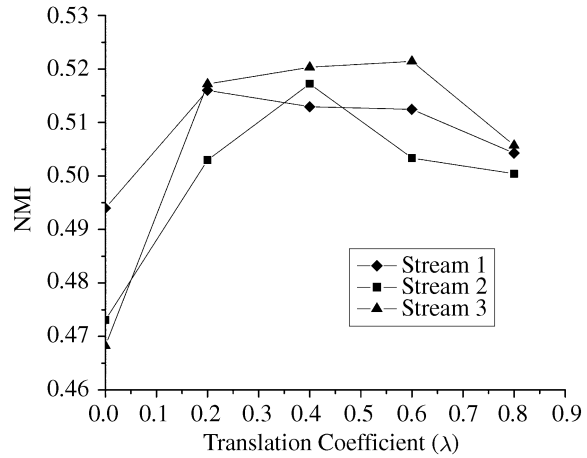

Fig.18. Change of clustering quality with λ of OCTSM (test interval: 200 seconds).

4.3 Results of Algorithm Runtimes

The third set of experiments is about the runtimes of OCTS and OCTSM with different documents. In this set of experiments, the documents in the text data streams do not come as a fixed speed but come after one document is clustered. The results are given in Figs. 19∼21. From the results, we can see that the runtimes of OCTS and OCTSM are increased with the increment of documents in different text data streams. In general, the runtimes quickly increase between 10 000 and 15 000 documents. This is because there are more historical cluster deletions and new cluster creations whose costs are more expensive than the simple cluster assignment operation. Besides, the runtime of the algorithms with the efficient impleme-

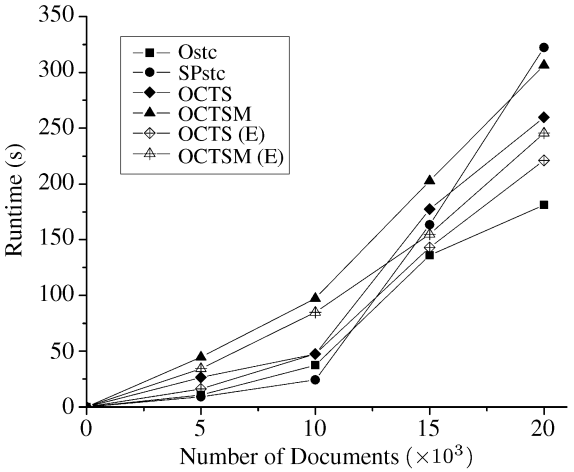

Fig.19. Runtime comparison of Stream 1.

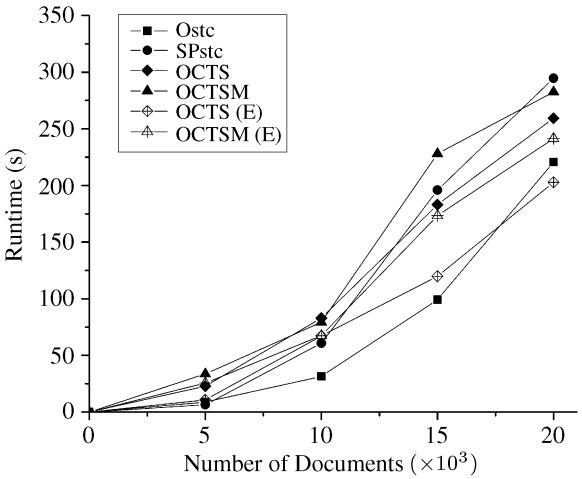

Fig.20. Runtime comparison of Stream 2.

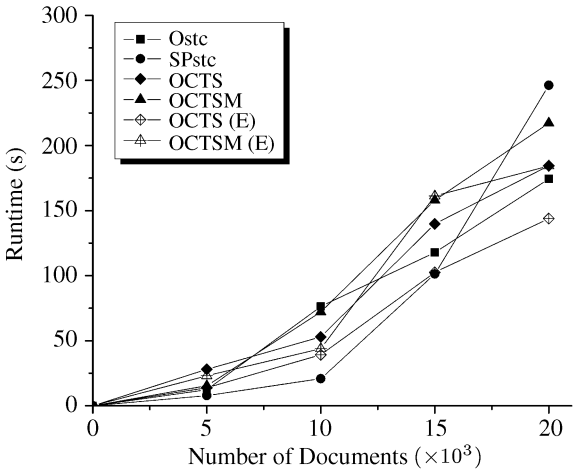

Fig.21. Runtime comparison of Stream 3.

ntation in Subsection 3.5 (denoted as OCTS(E) and OCTSM(E)), is smaller than that of the original OCTS and OCTSM, which shows the effectiveness of the three-layer implementation strategy. Specially, the runtime of the OCTS(E) is comparable with the simple Ostc framework.

4.4 Results of OCTS with MinFactor

We study the clustering quality of OCTS with threshold *MinFactor*. The results are showed in Figs. 22∼25. From the results, the curve goes upward first and soon reaches the peak point (around 0.05∼0.1). Then it would go downward all through until to a very poor point. The reason for this is that when *MinFactor* is too small (extremely equal 0), there are very few new clusters created, which makes some documents have to be added into the dissimilar cluster. Conversely, when *MinFactor* is very large, too many new clusters are created and too many historical clusters are deleted, which largely influences the cluster quality.

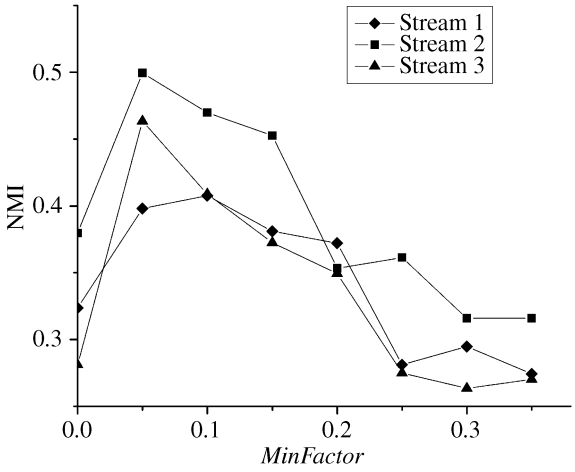

Fig.22. Change of clustering quality with *MinFactor* of OCTS (test interval: 50 seconds).

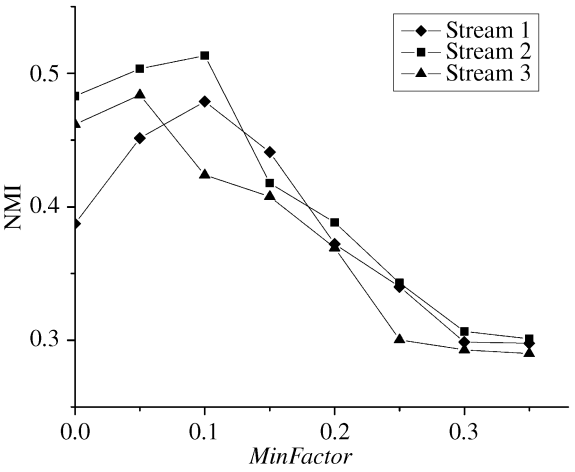

Fig.23. Change of clustering quality with *MinFactor* of OCTS (test interval: 100 seconds).

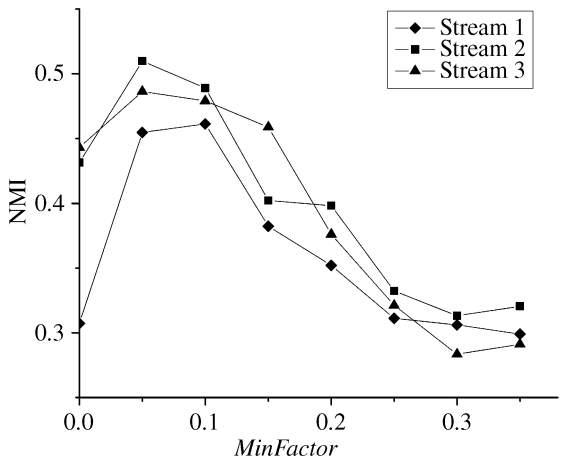

Fig.24. Change of clustering quality with *MinFactor* of OCTS (test interval: 150 seconds).

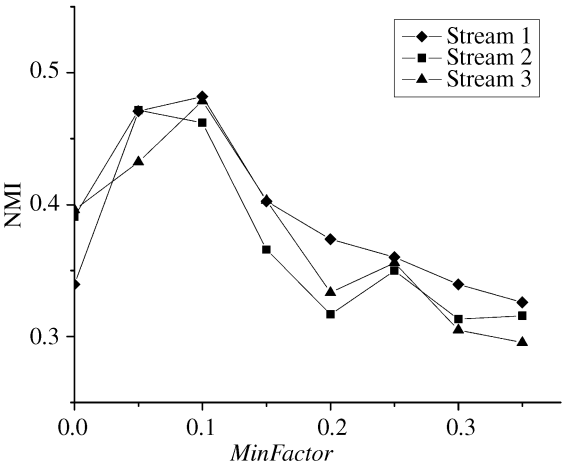

Fig.25. Change of the clustering quality with *MinFactor* of OCTS (test interval: 200 seconds).

The results in Figs. 26, 27 are the runtimes with different values of *MinFactor* when the number of the documents reaches 15 000 and 20 000 at each stream. From the curves, we can see that the runtimes first increase as the value of *MinFactor* increases till a point (around 0.2∼0.3). Then the variance of the runtimes becomes subtle (or nearly no change). This is because as the value of *MinFactor* increases, there are more and more historical cluster deletions and new cluster

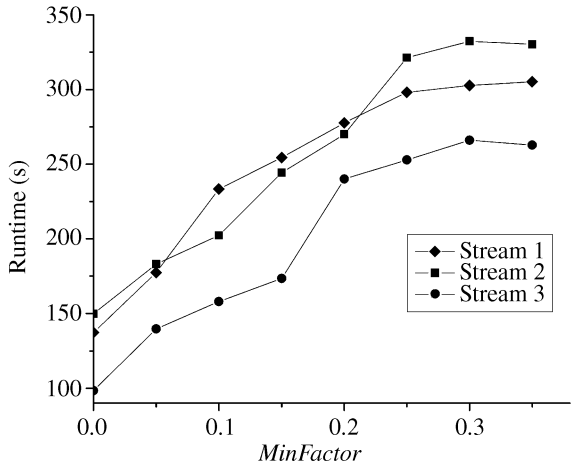

Fig.26. Runtimes with different values of *MinFactor* of OCTS (test point: 15 000 documents).

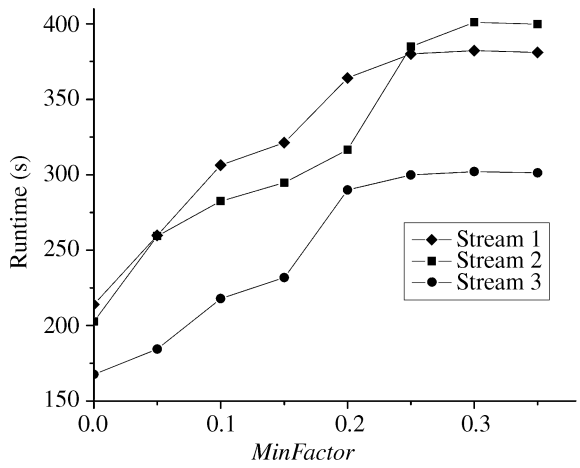

Fig.27. Runtimes with different values of *MinFactor* of OCTS (test point: 20 000 documents).

creations. Then when the *MinFactor* becomes large enough, nearly all the operations are historical cluster deletions and new cluster creations in the whole clustering process. So the variation of runtimes becomes nearly the same.

4.5 Results of OCTSM with MergeFactor

The fifth set of experiments is to test the cluster quality with different *MergeFactor* in OCTSM. In this set of experiments, the documents in the text data streams do not come as a fixed speed but come after one document is clustered. The results are reported in Figs. 28∼31. The experimental results show the clustering results of OCTSM are the best when *MergeFactor* = 0.3∼0.45. In general, NMI values will increase

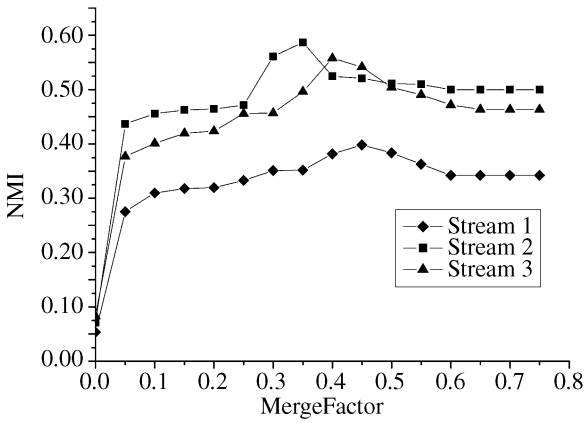

Fig.28. Change of clustering quality with MergeFactor of OCTSM (test interval: 50 seconds).

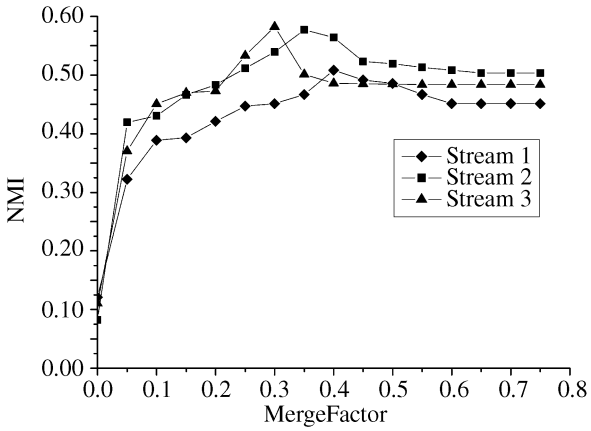

Fig.29. Change of clustering quality with MergeFactor of OCTSM (test interval: 100 seconds).

with the increase of *MergeFactor* till the peak point (*MergeFactor* = 0.3∼0.45) and then decrease until a specified stable point (*MergeFactor* = 0.6∼0.65) after which the value do not vary and that equals the result of OCTS. The reason for this variation is, when *MergeFactor* is too small, many useless historical clusters are merged, which reduce the clustering accuracy. Conversely, when *MergeFactor* is too large, very few historical clusters can be merged and the merge operation becomes useless. Specially, when *MergeFactor* is large enough, OCTSM is equal to OCTS since no clusters would be merged.

4.6 Results on Algorithm Memory Costs

The last set of experiments is to study the memory cost of our proposed clustering methods. The result of OCTS is showed in Fig.32 and the result of OCTSM

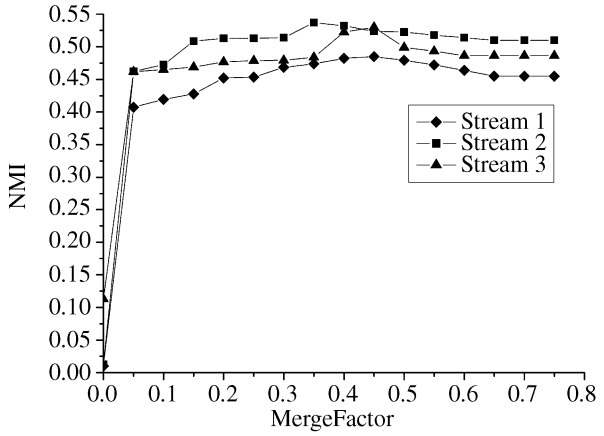

Fig.30. Change of clustering quality with MergeFactor of OCTSM (test interval: 150 seconds).

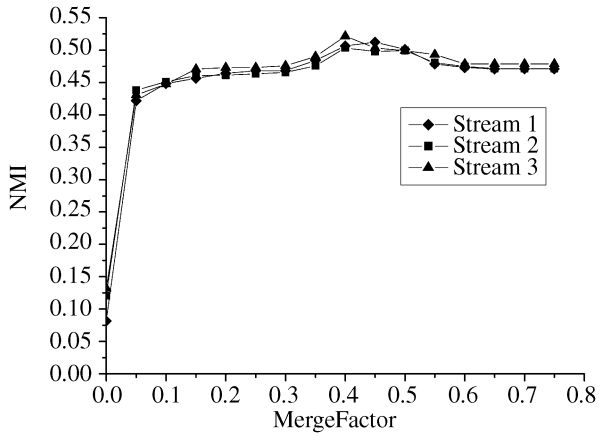

Fig.31. Change of clustering quality with MergeFactor of OCTSM (test interval: 200 seconds).

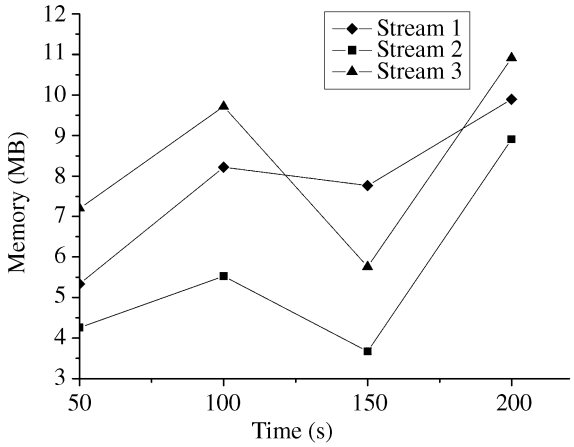

Fig.32. Memory cost of OCTS.

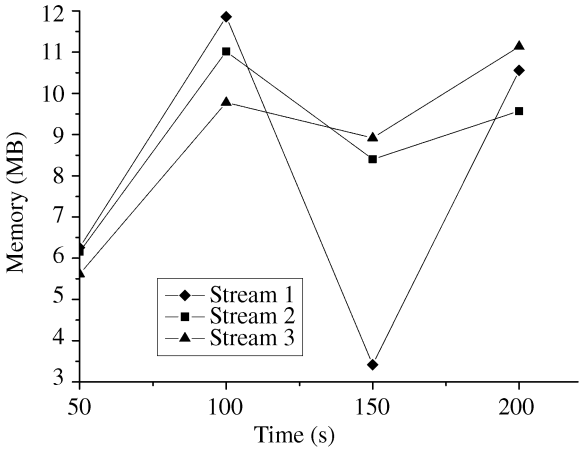

Fig.33. Memory cost of OCTSM.

is in Fig.33. In all figures, the maximal memory usage cost does not exceed 12MB. From this experimental results, the memory cost increases between 50∼100s, and then decreases between 100∼150s, and increases again between 150∼200s. This can be attributed to the variations of the number of the operations of historical cluster deletion, new cluster creation, and assignment operation on different text data streams.

5 Conclusion

In this paper, we extend semantic smoothing model into the text data streams context and present an extended semantic smoothing model. Based on the extended model, two online clustering algorithms OCTS and OCTSM are presented for the clustering of massive text data streams. In our algorithms, we present a new cluster statistics structure named cluster profile, which can capture the real-time semantics of text data streams and speed up the clustering process at the same time. We also present a series of experimental results illustrating the effectiveness of our technique. In particular, the clustering quality of our algorithms obviously outperforms the existing clustering algorithms based on the TF∗IDF scheme, the runtime is close to the existing Ostc algorithm, and the memory cost can satisfy the requirements of real time clustering of text data streams. Integrating further our clustering algorithms into the real applications, such as the real email clustering application, is our future work.

Acknowledgements We would like to thanks the anonymous reviewers for their helpful comments on the early version of this paper.

References

[1] Dou Shen, Qiang Yang, JianTao Sun, Zheng Chen. Thread detection in dynamic text message streams. In *Proc. ACM SIGIR* 2006, Seattle, Washington, August 6–11, pp.35–42.

[2] Aggarwal C C. A framework for diagnosing changes in evolving data streams. In *Proc. ACM SIGMOD* 2003, San Diego, June 9–12, pp.575–586.

[3] Agrawal C C, Han J, Wang J, Yu P S. A framework for clustering evolving data streams. In *Proc. VLDB 2003*, Berlin, September 9–12, 2003, pp.81–92.

[4] Agrawal C C, Han J, Wang J, Yu P S. A framework for projected clustering of high dimensional data streams. In *Proc. VLDB 2004*, Toronto, August 31–September 3, pp.852–863.

[5] O'Callaghan L, Mishra N, Meyerson A, Guha S. Streaming data algorithms for high-quality clustering. In *Proc. ICDE 2002*, San Jose, CA, February 26–March 1, pp.685–704.

[6] Aggarwal C C, Yu P S, A framework for clustering massive text and categorical data streams. In *Proc. SIAM Conference on Data Mining*, Bethesda, MD, April 20–22, 2006, pp.407–411.

[7] Yang Y, Pierce T, Carbonell J. A study of retrospective and on-line event detection. In *Proc. ACM SIGIR*, Melbourne, August 24–28, 1998, pp.28–36.

[8] Arindam Banerjee, Sugato Basu. Topic models over text streams: A study of batch and online unsupervised learning. In *Proc. SIAM Conference on Data Mining*, Minneapolis, April 26–28, 2007, pp.437–442.

[9] Xiaodan Zhang, Xiaohua Zhou, Xiaohua Hu. Semantic smoothing for model-based document clustering. In *Proc. ICDM06*, Hong Kong, December 18–22, pp.1193–1198.

[10] Zhou X, Hu X, Zhang X, Lin X, Song I Y. Context-sensitive semantic smoothing for the language modeling approach to genomic IR. In *Proc. ACM SIGIR*, Seattle, Washington, August 6–11, 2006, pp.170–177.

[11] Zhai C, Lafferty J. Two-stage language models for information retrieval. In *Proc. ACM SIGIR*, Tampere, August 11–15, 2002, pp.49–56.

[12] Zhai C, Lafferty J. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. ACM SIGIR*, New Orleans, September 9–13, 2001, pp.334–342.

[13] Yubao Liu, Jiarong Cai, Jian Yin, Ada Wai-Chee Fu. Clustering massive text data streams by semantic smoothing model. In *Proc. ADMA*, Harbin, August 6–8, 2007, pp.389–400.

[14] Zhong S, Ghosh J. Generative model-based document clustering: A comparative study. *Knowledge and Information Systems*, 2005, 8(3): 374–384.

[15] Steinbach M, Karypis G, Kumar V. A comparison of document clustering techniques. In *Proc. Text Mining Workshop*, *KDD 2000*, Boston, August 20–23, pp.1–20.

[16] Guha S, Mishra N, Motwani R, O'Callaghan L. Clustering data streams. In *Proc. FOCS 2000*, California, November 12–14, pp.359–366.

[17] Shi Zhong. Efficient streaming text clustering. *Neural Networks*, 2005, 18(5-6): 790–798.

[18] Fung G P C, Yu J X, Yu P S, Lu H. Parameter free bursty events detection in text streams. In *Proc. VLDB* 2005, Trondheim, August 30–September 2, pp.181–192

[19] Qi He, Kuiyu Chang, Ee-Peng Lim, Jun Zhang. Bursty feature representation for clustering text streams. In *Proc. SIAM Conference on Data Mining* 2007, Minneapolis, April 26–28, pp.491–496.

[20] Xu-Bin Deng, Yang-Yong Zhu. L-tree match: A new data extraction model and algorithm for huge text stream with noises. *Journal of Computer Science and Technology*, 2005, 20(6): 763–773.

[21] Gabriel Pui Cheong Fung, Jeffery Xu Yu, Hongjun Lu. Classifying text streams in the presence of concept drifts. In *Proc. PAKDD 2004*, Sydney, May 26–28, pp.373–383.

[22] Haixun Wang, Jian Yin, Jian Pei, Philip S Yu, Jeffrey Xu Yu. Suppressing model over-fitting in mining concept-drifting data streams. In *Proc. KDD 2006*, Philadelphia, August 20–23, pp.736–741.

[23] Weiheng Zhu, Jian Pei, Jian Yin, Yihuang Xie. Granularity adaptive density estimation and on demand clustering of concept-drifting data streams. In *Proc. DaWaK 2006*, Krakow, September 4–8, pp.322–331.

[24] Qiaozhu Mei, Chengxiang Zhai. Discovering evolutionary theme patterns from text – An exploration of temporal text mining. In *Proc. KDD 2005*, Chicago, August 21–24, pp.198–207.

[25] Shouke Qin, Weining Qian, Aoying Zhou. Approximately processing multi-granularity aggregate queries over data streams. In *Proc. ICDE 2006*, Atlanta, April 3–8, p.67.

[26] Dong-Hong Han, Guo-Ren Wang, Chuan Xiao, Rui Zhou. Load shedding for window joins over streams. *Journal of Computer Science and Technology,* 2007, 22(2): 182–189.

[27] Zhi-Hong Chong, Jeffrey Xu Yu, Zhen-Jie Zhang, Xue-Min Lin, Wei Wang, Ao-Ying Zhou. Efficient computation of k-medians over data streams under memory constraints. *Journal of Computer Science and Technology,* 2006, 21(2): 284–296.

[28] Jian Pei, Haixun Wang, Philip S Yu. Online mining of data streams: Applications, techniques and progress. In *Proc. KDD 2004* (Tutorials), Seattle, WA, August 22–25, pp.1–60.

[29] Joong Hyuk Chang, Won Suk Lee. Effect of count estimation in finding frequent itemsets over online transactional data streams. *Journal of Computer Science and Technology,* 2005, 20(1): 63–69.

[30] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W Wah, Jianyong Wang, Y Dora Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distributed and Parallel Databases,* 2005, 18(2): 173–197.

[31] Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W Wah, Jianyong Wang. Multi-dimensional regression analysis of time-series data streams. In *Proc. VLDB 2002,* August 20–23, Hong Kong, pp.323–334.

[32] Yubao Liu, Jiarong Cai, Jian Yin, Zhilan Huang. Document clustering based on semantic smoothing approach. In *Proc. AWIC 2007,* Fontainebleau, June 25–27, pp.217–222.

Yu-Bao Liu received his B.Eng. and M.Eng. degrees, both in computer science, from Xiangtan University, China in 1997, 2000, respectively, and his Ph.D. degree in computer science from Huazhong University of Science and Technology in 2003, China. He is currently a lecturer in the Department of Computer Science of Sun Yat-Sen University. He has published more than 30 refereed journal and conference papers. His research interests include database system, data warehouse and data mining. He is also a member of China Computer Federation (CCF).

Jia-Rong Cai is a graduate student of the Department of Computer Science of Sun Yat-Sen University. His research interests include data mining and information retrieval.

Jian Yin received the B.S., M.S., and Ph.D. degrees from Wuhan University, China, in 1989, 1991, and 1994, respectively, all in computer science. He joined Sun Yat-Sen University in July 1994 and now he is a professor with Information Science and Technology School. He served as a PC member of ICMLC 2005, PRICAI 2006, PAKDD 2006, 2007, 2008. His current research interests are in the areas of data mining, database, artificial intelligence, and machine learning.

Ada Wai-Chee Fu received her B.Sc. degree in computer science from the Chinese University of Hong Kong in 1983, and both M.Sc. and Ph.D. degrees in computer science from Simon Fraser University of Canada in 1986, 1990, respectively. She worked at Bell Northern Research in Ottawa, Canada from 1989 to 1993 on a wide-area distributed database project and joined the Chinese University of Hong Kong in 1993, where she is now an associate professor in the Department of Computer Science and Engineering. Her research interests include issues in database systems, data mining, and P2P data retrieval. She is a member of the ACM and IEEE associations.

Appendix. Proofs of the Properties

1. Proof for Property 1

1) For the i-th entry of $\overline{\boldsymbol{DF2}}(c_{12})$,

$$\begin{aligned}
&\overline{DF2}(c_{12})_i = w_c(w_i, c_1 \cup c_2)\\
&= \sum_{d\in c_1\cup c_2} f(t-T_d)c(w_i,d)\\
&= \sum_{d\in c_1} f(t-T_d)c(w_i,d) + \sum_{d\in c_2} f(t-T_d)c(w_i,d)\\
&= w_c(w_i,c_1) + w_c(w_i,c_2) = \overline{DF2}(c_1)_i + \overline{DF2}(c_2)_i.
\end{aligned}$$

2) For the i-th entry of $\overline{\boldsymbol{DF1}}(c_{12})$,

$$\begin{aligned}
&\overline{DF1}(c_{12})_i = w_t(w_i, c_1 \cup c_2)\\
&= \sum_k p(w|t_k) w_c(t_k, c_1 \cup c_2)\\
&= \sum_k p(w_i|t_k)\Big(\sum_{d\in c_1\cup c_2} f(t-T_d)c(t_k,d)\Big)\\
&= \sum_k p(w_i|t_k)\Big(\sum_{d\in c_1} f(t-T_d)c(t_k,d)\\
&\quad + \sum_{d\in c_2} f(t-T_d)c(t_k,d)\Big)\\
&= \sum_k p(w|t_k)(w_c(t_k,c_1) + w_c(t_k,c_2))\\
&= \sum_k p(w|t_k) w_c(t_k,c_1) + \sum_k p(w|t_k) w_c(t_k,c_2)\\
&= w_t(w_i,c_1) + w_t(w_i,c_2)\\
&= \overline{DF1}(c_1)_i + \overline{DF1}(c_2)_i.
\end{aligned}$$

3) For $s_{c_{12}}$,

$$\begin{aligned}
s_{c_{12}} &= \sum_k w_c(t_k, c_1 \cup c_2)\\
&= \sum_k \sum_{d\in c_1\cup c_2} f(t-T_d)c(t_k,d)\\
&= \sum_k \Big(\sum_{d\in c_1} f(t-T_d)c(t_k,d) + \sum_{d\in c_2} f(t-T_d)c(t_k,d)\Big)\\
&= \sum_k w_c(t_k,c_1) + \sum_k w_c(t_k,c_2)\\
&= s_{c_1} + s_{c_2}.
\end{aligned}$$

4) For $l_{c_{12}}$, the proof is trivial since the last updated time of the merged cluster is the later of the original two. □

2. Proof for Property 2

1) For the i-th entry of $\overline{\boldsymbol{DF2}}(c_a)$,

$$\begin{aligned}
\overline{DF2}(c_a)_i &= \overline{DF2}(c_b)_i + f(t-t)c(w_i,d)\\
&= \overline{DF2}(c_b)_i + c(w_i,d)
\end{aligned}$$

2) For the i-th entry of $\overline{\boldsymbol{DF1}}(c_a)$,

$$\begin{aligned}
\overline{DF1}(c_a)_i &= \overline{DF1}(c_b)_i + \sum_k p(w_i|t_k) f(t-t)c(t_k,d)\\
&= \overline{DF1}(c_b)_i + \sum_k p(w_i|t_k)c(t_k,d)
\end{aligned}$$

3) For s_{c_a},

$$\begin{aligned}
s_{c_a} &= s_{c_b} + \sum_k f(t-t)c(t_k,d)\\
&= s_{c_b} + \sum_k c(t_k,d).
\end{aligned}$$

4) For l_{c_a}, the proof is trivial since the current time is the last updated time. □

3. Proof for Property 3

1) For the i-th entry of $\overline{\boldsymbol{DF2}}(c_{t_2})$,

$$\begin{aligned}\overline{DF2}(c_{t_2})_i &= w\_c(w_i, c_{t_2}) \\ &= \sum_{d\in c_{t_2}} f(t_2 - T_d)c(w_i, d) \\ &= \sum_{d\in c_{t_2}} f(t_2 - t_1 + t_1 - T_d)c(w_i, d) \\ &= \sum_{d\in ct1} 2^{-\zeta(t_2-t_1+t_1-T_d)} \times c(w_i, d) \\ &= \sum_{d\in ct1} 2^{-\zeta(t_2-t_1)} \times 2^{-\zeta(t_1-T_d)} \times c(w_i, d) \\ &= 2^{-\zeta(t_2-t_1)} \times \sum_{d\in ct1} 2^{-\zeta(t_1-T_d)} \times c(w_i, d) \\ &= \overline{DF2}(c_{t_1})_i \times 2^{-(t_2-t_1)}.\end{aligned}$$

2) For the i-th entry of $\overline{\boldsymbol{DF1}}(c_{t_2})$,

$$\begin{aligned}&\overline{DF1}(c_{t_2})_i = w_t(w_i, c_{t_2}) \\ &= \sum_k p(w|t_k) w\_c(t_k, c_{t_2}) \\ &= \sum_k p(w_i|t_k)\Big(\sum_{d\in ct_2} f(t_2 - T_d)c(t_k, d)\Big) \\ &= \sum_k p(w_i|t_k)\Big(\sum_{d\in ct_2} f(t_2 - t_1 + t_1 - T_d)c(t_k, d)\Big) \\ &= \sum_k p(w_i|t_k) \sum_{d\in ct_1} 2^{-\zeta(t_2-t_1)} \times 2^{-\zeta(t_1-T_d)} \times c(w_i, d) \\ &= \overline{DF1}(c_{t_1})_i \times 2^{-\zeta(t_2-t_1)}.\end{aligned}$$

3) For $s_{c_{t_2}}$,

$$\begin{aligned}s_{c_{t_2}} &= \sum_k w\_c(t_k, c_{t_2}) \\ &= \sum_k \sum_{d\in c_{t_2}} f(t_2 - T_d)c(t_k, d) \\ &= \sum_k \sum_{d\in c_{t_1}} f(t_2 - t_1 + t_1 - T_d)c(t_k, d) \\ &= \sum_k \sum_{d\in c_{t_1}} 2^{-\zeta(t_2-t_1)} \times 2^{-\zeta(t_1-T_d)} \times c(t_k, d) \\ &= s_{c_{t_1}} \times 2^{-\zeta(t_2-t_1)}.\end{aligned}$$

4) For $l_{c_{t_2}}$, the proof is trivial since no document is added to cluster c_{t_1} during $[t_1, t_2]$. □