An On-line Document Clustering Method Based on Forgetting Factors (long version)*

Yoshiharu Ishikawa† Yibing Chen‡** Hiroyuki Kitagawa†

† Institute of Information Sciences and Electronics, University of Tsukuba
‡ Master's Program in Science and Engineering, University of Tsukuba
{ishikawa,kitagawa}@is.tsukuba.ac.jp

Abstract. With the rapid development of on-line information services, information technologies for on-line information processing have been receiving much attention recently. Clustering plays important roles in various on-line applications such as extraction of useful information from news feeding services and selection of relevant documents from the incoming scientific articles in digital libraries. In on-line environments, users generally have interests on newer documents than older ones and have no interests on obsolete old documents.

Based on this observation, we propose an on-line document clustering method F^2ICM (*Forgetting-Factor-based Incremental Clustering Method*) that incorporates the notion of a *forgetting factor* to calculate document similarities. The idea is that every document gradually losses its weight (or memory) as time passes according to this factor. Since F^2ICM generates clusters using a document similarity measure based on the forgetting factor, newer documents have much effects on the resulting cluster structure than older ones. In this paper, we present the fundamental idea of the F^2ICM method and describe its details such as the similarity measure and the clustering algorithm. Also, we show an efficient incremental statistics maintenance method of F^2ICM which is indispensable for on-line dynamic environments.

Keywords: clustering, on-line information processing, incremental algorithms, forgetting factors

1 Introduction

According to the recent information technology development such as the Internet and electronic documents, a huge number of on-line documents (e.g., on-line news articles and electronic journals) are delivered over the network and stored in digital libraries and electronic document archives. Since it is difficult for ordinal users to select required information from such huge document repositories, information filtering to select useful information from delivered new documents and summarization methods to extract important topics from documents have become important research areas. Additionally,

* This paper is an extended version of the paper: Yoshiharu Ishikawa, Yibing Chen, and Hiroyuki Kitagawa: "An On-line Document Clustering Method Based on Forgetting Factors", in *Proceedings of European Conference on Digital Libraries (ECDL2001)*, Darmstadt, Germany, Sept. 2001.

** Current affiliation: Yamatake Building Systems Co., Ltd.

topic detection and tracking (TDT) from on-line information sources has gained much attentions recently [9].

Document clustering is a method to collect similar documents to form document groups (*clusters*), and used as fundamental methods for information retrieval, information filtering, topic detection and tracking, and document categorization [2, 5–8]. To summarize the trend of on-line documents incoming from various information sources (news wire services, Web pages, etc.) and to provide up-to-date information to users, we propose an on-line document clustering method F^{2}ICM (Forgetting-Factor-based Incremental Clustering Method) that can provide clustering results reflecting the novelty of documents; in this method, newer documents highly affect to the clustering results than older documents. The most characteristic feature of F^{2}ICM is that it incorporates the notion of a *forgetting factor*. In F^{2}ICM, we set an initial weight to every document when it is acquired from an information source. Document weights gradually decay as time passes according to the rate specified by the forgetting factor. Since the document similarity measure proposed in this paper is devised to reflect such document weights in computing similarity scores, F^{2}ICM based on the measure can generate clustering results by setting higher importance on recent documents and lower importance on obsolete old documents. Namely, we can say that F^{2}ICM continuously "forgets" old information and focuses mainly on "current" information to generate clusters.

The F^{2}ICM method is an extension of an existing seed-based clustering method C^{2}ICM proposed by Can [3]. When new documents are obtained, F^{2}ICM (also C^{2}ICM) incrementally updates the previous clustering result by computing new seeds and reassigning documents into the seeds. Since F^{2}ICM uses a similarity measure based on the forgetting factor, we have to manage time-dependent statistics for the calculation of similarity scores and have to update these statistics when the document set is updated. In this paper, we show a sophisticated method to update such statistics incrementally with low overheads. Based on the algorithms, F^{2}ICM can adapt to on-line document clustering needs that require current "hot" information, and can be used as an underlying method to support on-line digital library tasks.

The following part of this paper is organized as follows. In Section 2, we briefly introduce the C^{2}ICM clustering method that is the basis of our proposed method F^{2}ICM. Then F^{2}ICM clustering method is described in Section 3. Section 4 focuses on the incremental update algorithm of statistics for the efficient update processing and Section 5 shows the basic incremental cluster update algorithm. In Section 6, we briefly mention the approaches for document expiration and parameter settings and the revised cluster update algorithm. Section 7 briefly reports the results of our experiments. Finally, Section 8 concludes the paper.

2 The C^{2}ICM Clustering Method

The clustering method F^{2}ICM proposed in this paper is partially based on the idea of *C^{2}ICM* (*Cover-Coefficient-based Incremental Clustering Methodology*) proposed by Can [3]. Before introducing F^{2}ICM in Section 3, we briefly describe the algorithms used in C^{2}ICM. For the differences between F^{2}ICM and C^{2}ICM, we mention them on appropriate points in the following discussion.

Suppose that a document set consists of m documents $d_1, \ldots, d_m$ and let all the terms appeared in these documents be $t_1, \ldots, t_n$. C^{2}ICM is a clustering method that is based on probabilistic modeling of document and term similarities [1]. In the method, two probabilities $\Pr(d_j|d_i)$ and $\Pr(t_l|t_k)$ play important roles: the former is the conditional probability that the document d_j is obtained when a document d_i is given. Similarly, the latter is the conditional probability that the term t_l is obtained when a term t_k is given. These probabilities represent the degree of association between two documents or terms. In this section, we assume that these two probabilities are already given and present the clustering algorithms of C^{2}ICM. In Section 3, we describe the derivation of these two probabilities.

2.1 Seed Power

C^{2}ICM is a seed-based clustering method. In its clustering procedure, seed documents are initially selected from the document set then each remaining document is grouped with the most similar seed document and finally clusters are formulated. In this subsection, the notion of a *seed power*, an index used in the seed selection, is explained.

Decoupling Coefficient and Coupling Coefficient First, two important notions used to calculate seed powers in C^{2}ICM are introduced. A probability $\Pr(d_i|d_i)$ that the document d_i is obtained when a document d_i itself is given is called the *decoupling coefficient* δ_i for d_i [3]:

$$\delta_i \stackrel{\text{def}}{=} \Pr(d_i|d_i). \tag{1}$$

Intuitively, δ_i is an index to indicate how d_i is independent (different) from other documents. On the other hand, the *coupling coefficient* φ_i for d_i, given by

$$\varphi_i \stackrel{\text{def}}{=} 1 - \delta_i, \tag{2}$$

is considered to be the degree of dependence of d_i.

Similar to the case of documents, the *decoupling coefficient* δ'_j and *coupling coefficient* φ'_j for a term t_j are defined as follows [3]:

$$\delta'_j \stackrel{\text{def}}{=} \Pr(t_j|t_j) \tag{3}$$

$$\varphi'_j \stackrel{\text{def}}{=} 1 - \delta'_j. \tag{4}$$

Seed Power In [3], document d_i's *seed power* sp_i that evaluates appropriateness of d_i as a cluster seed is given by the following formula:

$$sp_i \stackrel{\text{def}}{=} \delta_i \cdot \varphi_i \cdot \sum_{j=1}^{n} freq(d_i, t_j) \cdot \delta'_j \cdot \varphi'_j, \tag{5}$$

[1] Our notations are rather different from those in [3] to perform more clear probabilistic modeling of document similarities.

where $freq(d_i, t_j)$ is the occurrence frequency of term t_j within document d_i. The intuitive idea behind this formula is to select a document which has moderate dependency within the document set as a cluster seed document: the idea is represented by the part "$\delta_i \cdot \varphi_i$". The remaining summation of Eq. (5) is a normalized weighting by considering the occurrence frequency and the dependency/independency factors for each term.

2.2 Clustering Algorithms

The clustering algorithms of C^{2}ICM consist of the initial cluster generation algorithm and the incremental clustering algorithm that used in the update time.

Initial Clustering Algorithm

1. Calculate the seed power sp_i for each document d_i in the document set.
2. Select n_c documents which have the largest sp_i values as cluster seeds [2].
3. Each remaining document is appended to the cluster such that its cluster seed is the most similar one to the document. We can specify a threshold value for the assignment. A document that does not have a similarity score to any seeds that is larger than the threshold value is assigned to a special *ragbag cluster*.

Incremental Clustering Algorithm When documents are appended or deleted, the C^{2}ICM method maintains the clusters in an incremental manner:

1. Recompute the seed power of each document in the document set.
2. Select n_c documents which have the largest sp_i values as cluster seeds.
3. Examine each previous cluster: if its seed is re-selected at this time, the cluster is preserved. On the other hand, if the seed is not selected, we delete the cluster.
4. Perform reclustering: new documents, the documents contained in the clusters deleted in Step 3, and the documents in the ragbag cluster are assigned to the most similar clusters based on the similarity scores. As the initial clustering algorithm, a document that does not have a similarity score to any seeds that is larger than the threshold value is assigned to the ragbag cluster.

The point is not to recluster all the documents from scratch but to utilize partial results of the previous clustering because C^{2}ICM focuses on low update processing cost for on-line information processing. Although we did not mention the document deletion method above, we can easily incorporate the deletion phase into the update algorithm.

3 The F^{2}ICM Clustering Method

In this section, the F^{2}ICM (Forgetting-Factor-based Incremental Clustering Method) is introduced. As mentioned before, this method is an extension of C^{2}ICM so that its algorithms are basically based on C^{2}ICM. The main difference between them is that F^{2}ICM

[2] Although the number of clusters n_c is automatically determined in the original C^{2}ICM method [3], this paper assumes that a user specifies n_c. This is for the simplification of the procedure.

assigns temporally decaying weights to documents and utilizes a document similarity measure that incorporates the notion of a forgetting factor. In this section, the document similarity measure used in F^{2}ICM is derived and some probabilities used in the algorithms, such as $\Pr(d_j|d_i)$ and $\Pr(t_l|t_k)$, are introduced.

3.1 Document Forgetting Model

The *document forgetting model* described in this subsection plays an important role in the F^{2}ICM method. The model is based on a simple intuition: the values of news articles delivered everyday and on-line journal articles maintained in digital libraries are considered to be gradually losing their values as time passes. The document forgetting model is based on a rough modeling of such behaviors.

Let the current time be $t = \tau$ and the acquisition time of each document d_i ($i = 1,\ldots,m$) be T_i ($T_i \leq \tau$); for example, we can use issue dates as the acquisition times for on-line news articles. We represent the *value* (also called *weight*) of d_i at time τ by $dw_i|_\tau$. The notation "$|_\tau$" is used to represent the value of a variable at time τ. If the context is clear, we omit "$|_\tau$".

Although we can consider various formulas to represent the decay of the information value of a document, we utilize the following exponential weighting formula:

$$dw_i|_\tau \stackrel{\text{def}}{=} \lambda^{\tau - T_i} \quad (0 < \lambda < 1), \tag{6}$$

where λ is a parameter tuned according to the target document set and the intended application. The smaller the value of λ is, the faster the value decay (forgetting) speed becomes. For this reason, we call λ a *forgetting factor*. The reasons to select this exponential forgetting model are summarized as follows:

1. The model that human memory will decrease as an exponential function depending on time appears as a behavioral law in procedural and declarative human memory modeling, and called the *power law of forgetting* [1][3]. Of course, such a cognitive human memory model and forgetting of document contents in our context do not have direct relationship, but we may be able to regard the human memory model as an informal background of our model.
2. If we use the exponential forgetting factor shown above, we can construct an efficient statistics maintenance method for our clustering method. The details of the maintenance method is described in Section 4.
3. The proposed document forgetting model simply uses one parameter λ to control the degree of weight decay. This means that the information value of every document decays with the same rate and works as a basis of efficient implementation of our cluster maintenance method. Although it is possible to set different λ values for different documents, the approach is not suited to on-line document clustering since its processing cost becomes much higher than that of our simple approach.

[3] But note that our document forgetting model (Eq. (6)) is too much simplified one for convenience; models used in the human cognition research area are more devised ones [1].

3.2 Derivation of the Document Similarity Measure

In this subsection, we derive the document similarity measure from a probabilistic perspective. For the derivation, we take the document forgetting model introduced above into account. In the following, we represent the documents in a document set by d_i ($i = 1, \ldots, m$) and all the index terms in the document set by t_k ($k = 1, \ldots, n$). The number of occurrences of term t_k within document d_i is represented by $freq(d_i, t_k)$. And we assume that the acquisition times of the documents $d_1, d_2, \ldots, d_m$ satisfy the relationship $T_1 \leq T_2 \leq \cdots \leq T_m$.

First we define the total weights of all the documents tdw by

$$tdw \stackrel{\text{def}}{=} \sum_{l=1}^{m} dw_l, \tag{7}$$

and define the probability $\Pr(d_i)$ that the document d_i is randomly selected from the document set by the following *subjective probability*:

$$\Pr(d_i) \stackrel{\text{def}}{=} \frac{dw_i}{tdw}. \tag{8}$$

Namely, we assume that old documents have smaller selection probability than newer ones.

Next, we derive the conditional probability $\Pr(t_k|d_i)$ that a term t_k is selected from a document d_i. We simply derive it based on the number of occurrences of terms in a document:

$$\Pr(t_k|d_i) \stackrel{\text{def}}{=} \frac{freq(d_i, t_k)}{\sum_{l=1}^{n} freq(d_i, t_l)}. \tag{9}$$

Since we can consider that the right hand of Eq. (9) gives the *term frequency* of t_k within d_i, we also denote it as

$$tf(d_i, t_k) \stackrel{\text{def}}{=} \Pr(t_k|d_i). \tag{10}$$

The occurrence probability of term t_k, $\Pr(t_k)$, can be derived by

$$\Pr(t_k) = \sum_{i=1}^{m} \Pr(t_k|d_i) \cdot \Pr(d_i). \tag{11}$$

Since we can consider that $\Pr(t_k)$ represents the *document frequency* of term t_k, we also denote $\Pr(t_k)$ as

$$df(t_k) \stackrel{\text{def}}{=} \Pr(t_k). \tag{12}$$

Also, since we can regard the reciprocal of $df(t_k)$ as the *inverse document frequency* (*IDF*) of t_k, we define

$$idf(t_k) \stackrel{\text{def}}{=} \frac{1}{df(t_k)}. \tag{13}$$

Using the above formulas and Bayes' theorem, we obtain

$$\begin{aligned} \Pr(d_j|t_k) &= \frac{\Pr(t_k|d_j)\Pr(d_j)}{\Pr(t_k)} \\ &= \Pr(d_j) \cdot tf(d_j, t_k) \cdot idf(t_k). \end{aligned} \tag{14}$$

Next, we consider the conditional probability $\Pr(d_j|d_i)$. It is defined as

$$\Pr(d_j|d_i) = \sum_{k=1}^{n} \Pr(d_j|d_i, t_k) \Pr(t_k|d_i). \tag{15}$$

Now we make an assumption that

$$\Pr(d_j|d_i, t_k) \simeq \Pr(d_j|t_k), \tag{16}$$

then we get

$$\begin{aligned} \Pr(d_j|d_i) &\simeq \sum_{k=1}^{n} \Pr(d_j|t_k) \Pr(t_k|d_i) \\ &= \Pr(d_j) \sum_{k=1}^{n} tf(d_i, t_k) \cdot tf(d_j, t_k) \cdot idf(t_k). \end{aligned} \tag{17}$$

Based on the above formula, we also get

$$\begin{aligned} \Pr(d_i, d_j) &= \Pr(d_j|d_i) \cdot \Pr(d_i) \\ &\simeq \Pr(d_i) \Pr(d_j) \sum_{k=1}^{n} tf(d_i, t_k) \cdot tf(d_j, t_k) \cdot idf(t_k). \end{aligned} \tag{18}$$

In the following, we use $\Pr(d_i, d_j)$, the co-occurrence probability of documents d_i and d_j, as the similarity score for d_i and d_j, and define the document similarity measure as follows:

$$sim(d_i, d_j) \stackrel{\text{def}}{=} \Pr(d_i, d_j). \tag{19}$$

Based on the above definition, obsolete documents generally have small similarity scores with any other documents since their occurrence probabilities are quite small.

Similarly, if we make an assumption that $\Pr(t_i|d_k, t_j) \simeq \Pr(t_i|d_k)$, we get

$$\begin{aligned} \Pr(t_i|t_j) &= \sum_{k=1}^{m} \Pr(t_i|d_k, t_j) \cdot \Pr(d_k|t_j) \\ &\simeq \sum_{k=1}^{m} \Pr(t_i|d_k) \cdot \Pr(d_k|t_j) \\ &= idf(t_j) \cdot \sum_{k=1}^{m} \Pr(d_k) \cdot tf(d_k, t_i) \cdot tf(d_k, t_j) \end{aligned} \tag{20}$$

and obtain

$$\begin{aligned} \Pr(t_i, t_j) &= \Pr(t_j) \Pr(t_i|t_j) \\ &\simeq \sum_{k=1}^{m} \Pr(d_k) \cdot tf(d_k, t_i) \cdot tf(d_k, t_j). \end{aligned} \tag{21}$$

Now we briefly mention the relationship between two document similarity measures used in our F^{2}ICM method and the C^{2}ICM method [3]. In F^{2}ICM, we have revised the

similarity measure used in C^{2}ICM from a more theoretical perspective and derived the similarity measure based on the probabilistic modeling. While the main difference of F^{2}ICM from C^{2}ICM is the incorporation of a forgetting factor into the probability calculation (e.g., $\Pr(d_i)$), even if we omit forgetting factors from our formulas, the formulas do not completely match the ones of C^{2}ICM. Another difference is that C^{2}ICM defines its document similarity measure by $\Pr(d_j|d_i)$ instead of $\Pr(d_i, d_j)$. However, $\Pr(d_i, d_j)$ and $\Pr(d_j|d_i)$ play almost equivalent roles as far as they are used in the clustering algorithms shown in Section 2.

4 Updating Statistics and Probabilities

We have already shown the basic clustering algorithm of the F^{2}ICM method in Section 2 and derived the document similarity measure in Section 3. Although we can generate and maintain document clusters based on them, we still have a room to improve the clustering method by devising the update procedure for statistics and probabilities used in the clustering. Since some of the statistics and probabilities used in our method (e.g., $\Pr(d_i)$ and $df(t_k)$) change their values when new documents are incorporated into the document set and when time has passed. Therefore, we have to recalculate their new values based on their definitions shown in Section 3. Since the recalculation becomes costly for large data sets, we devise the statistics update method which is based on incremental computation and fully utilizes previous statistics and probabilities to achieve efficient updates. In this section, we show such an incremental update method for statistics and probabilities.

Let the last update time of the given document set consisting of m documents $d_1, \ldots, d_m$ be $t = \tau$. Namely, the most recent documents are incorporated into the document set at $t = \tau$. Then suppose that m' new documents $d_{m+1}, \ldots, d_{m+m'}$ are appended at the time $t = \tau + \Delta\tau$. Therefore, their acquisition times are $T_{m+1} = \cdots = T_{m+m'} = \tau + \Delta\tau$. Let all the index terms contained in the document set at time $t = \tau$ be $t_1, \ldots, t_n$ and the additional terms incorporated by the insertion of documents $d_{m+1}, \ldots, d_{m+m'}$ be $t_{n+1}, \ldots, t_{n+n'}$. In the following discussion, we assume that $m \gg m'$ and $n \gg n'$ hold.

1. Updating of dw_i's: First we consider the update of document weights of documents $d_1, \ldots, d_m$. We have already assigned a weight $dw_i|_\tau$ to each document d_i ($1 \leq i \leq m$) at the last update time $t = \tau$. These weights have to be updated to $dw_i|_{\tau+\Delta\tau}$ in this update time. Since the relationship

$$\begin{aligned} dw_i|_{\tau+\Delta\tau} &= \lambda^{\tau+\Delta\tau-T_i} \\ &= \lambda^{\Delta\tau} dw_i|_\tau \end{aligned} \tag{22}$$

holds between $dw_i|_\tau$ and $dw_i|_{\tau+\Delta\tau}$, we can easily derive $dw_i|_{\tau+\Delta\tau}$ from $dw_i|_\tau$ by simply multiplying $\lambda^{\Delta\tau}$ to $dw_i|_\tau$. This property for the efficient update is due to the selection of the exponential forgetting factor in our document forgetting model.
For the new incoming documents $d_{m+1}, \ldots, d_{m+m'}$, we simply set $dw_i|_{\tau+\Delta\tau} = 1$ ($m + 1 \leq i \leq m + m'$). The computational complexity of this step is estimated as $O(m + m') \approx O(m)$.

2. Updating of tdw: For the total weight of all the documents tdw, we can utilize the following update formula:

$$\begin{aligned} tdw|_{\tau+\Delta\tau} &= \sum_{l=1}^{m+m'} \lambda^{\tau+\Delta\tau-T_l} \\ &= \lambda^{\Delta\tau} tdw|_{\tau} + m'. \end{aligned} \tag{23}$$

   The processing cost is $O(1)$.
3. Calculation of $\Pr(d_i)$'s: $\Pr(d_i)$, the occurrence probability of document d_i, is given by

$$\Pr(d_i)|_{\tau+\Delta\tau} = \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}}. \tag{24}$$

   Since we have already obtained $dw_i|_{\tau+\Delta\tau}$ and $tdw|_{\tau+\Delta\tau}$ in Step 1 and 2, we can easily calculate $\Pr(d_i)$ when it is required.
4. Maintenance of $tf(d_i, t_k)$'s: For $tf(d_i, t_k)$, we decompose it into

$$tf(d_i, t_k) = \frac{freq(d_i, t_k)}{doclen_i}, \tag{25}$$

   then maintain $freq(d_i, t_k)$ and $doclen_i$ instead of $tf(d_i, t_k)$[4], and calculate $tf(d_i, t_k)$ when it is required.
   Since $freq(d_i, t_k)$ and $doclen_i$ do not depend on time, we have to compute them only for the new documents $d_{m+1}, \ldots, d_{m+m'}$. If we roughly suppose that the number of terms contained in each document be a constant c, this step requires $O(cm') = O(m')$ computation time.
5. Updating of $df(t_k)$'s: The formula of $df(t_k)|_{\tau}$ can be transformed as

$$\begin{aligned} df(t_k)|_{\tau} &= \sum_{i=1}^{m} \frac{dw_i|_{\tau}}{tdw|_{\tau}} \cdot tf(d_i, t_k) \\ &= \frac{1}{tdw|_{\tau}} \sum_{i=1}^{m} dw_i|_{\tau} \cdot tf(d_i, t_k). \end{aligned} \tag{26}$$

   Now we define $\widetilde{df}(t_k)|_{\tau}$ as

$$\widetilde{df}(t_k)|_{\tau} \stackrel{\text{def}}{=} \sum_{i=1}^{m} dw_i|_{\tau} \cdot tf(d_i, t_k), \tag{27}$$

   then $df(t_k)|_{\tau}$ is given by

$$df(t_k)|_{\tau} = \frac{\widetilde{df}(t_k)|_{\tau}}{tdw|_{\tau}}. \tag{28}$$

   By storing $\widetilde{df}(t_k)|_{\tau}$ instead of $df(t_k)|_{\tau}$, we can achieve the incremental update. When we need the new value $df(t_k)|_{\tau+\Delta\tau}$, we can compute it from $\widetilde{df}(t_k)|_{\tau+\Delta\tau}$ and $tdw|_{\tau+\Delta\tau}$ using Eq. (28).

[4] The reason to maintain $freq(d_i, t_k)$ and $doclen_i$ independently is that we need $freq(d_i, t_k)$ to calculate the seed power sp_i using Eq. (5).

We can derive the update formula for $\widetilde{df}(t_k)$ as follows:

$$\begin{aligned}
\widetilde{df}(t_k)|_{\tau+\Delta\tau} &= \sum_{i=1}^{m+m'} dw_i|_{\tau+\Delta\tau} \cdot tf(d_i,t_k) \\
&= \sum_{i=1}^{m} dw_i|_{\tau+\Delta\tau} \cdot tf(d_i,t_k) + \sum_{i=m+1}^{m+m'} dw_i|_{\tau+\Delta\tau} \cdot tf(d_i,t_k) \\
&= \sum_{i=1}^{m} \lambda^{\Delta\tau} dw_i|_{\tau} \cdot tf(d_i,t_k) + \sum_{i=m+1}^{m+m'} 1 \cdot tf(d_i,t_k) \\
&= \lambda^{\Delta\tau} \sum_{i=1}^{m} dw_i|_{\tau} \cdot tf(d_i,t_k) + \sum_{i=m+1}^{m+m'} tf(d_i,t_k) \\
&= \lambda^{\Delta\tau} \cdot \widetilde{df}(t_k)|_{\tau} + \sum_{i=m+1}^{m+m'} tf(d_i,t_k) \qquad (29)
\end{aligned}$$

Now we define $\Delta t f_{\text{sum}}(t_k)$ as

$$\Delta t f_{\text{sum}}(t_k) \stackrel{\text{def}}{=} \sum_{i=m+1}^{m+m'} tf(d_i,t_k), \qquad (30)$$

then we get a simplified update formula

$$\widetilde{df}(t_k)|_{\tau+\Delta\tau} = \lambda^{\Delta\tau} \cdot \widetilde{df}(t_k)|_{\tau} + \Delta t f_{\text{sum}}(t_k). \qquad (31)$$

Since it takes $O(m')$ time to compute a $\Delta t f_{\text{sum}}(t_k)$ value, we need $O(m' \cdot (n+n')) \approx O(m'n)$ time for all the documents.

6. Calculation of δ_i's: We can transform the decoupling coefficient formula for documents as follows:

$$\begin{aligned}
\delta_i|_{\tau} &= \Pr(d_i|d_i)|_{\tau} \\
&= \Pr(d_i)|_{\tau} \sum_{k=1}^{n} tf(d_i,t_k)^2 \cdot idf(t_k)|_{\tau}. \qquad (32)
\end{aligned}$$

For the documents $d_1 \ldots, d_m$, $\delta_i|_{\tau+\Delta\tau}$ is given as follows:

$$\begin{aligned}
\delta_i|_{\tau+\Delta\tau} &= \Pr(d_i)|_{\tau+\Delta\tau} \sum_{k=1}^{n+n'} tf(d_i,t_k)^2 \cdot idf(t_k)|_{\tau+\Delta\tau} \\
&= \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}} \sum_{k=1}^{n} tf(d_i,t_k)^2 \cdot \frac{1}{df(t_k)|_{\tau+\Delta\tau}} \\
&= \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}} \sum_{k=1}^{n} tf(d_i,t_k)^2 \cdot \frac{tdw|_{\tau+\Delta\tau}}{\widetilde{df}(t_k)|_{\tau+\Delta\tau}} \\
&= dw_i|_{\tau+\Delta\tau} \sum_{k=1}^{n} \frac{tf(d_i,t_k)^2}{\widetilde{df}(t_k)|_{\tau+\Delta\tau}} \qquad (33)
\end{aligned}$$

Although we cannot derive $\delta_i|_{\tau+\Delta\tau}$ incrementally from $\delta_i|_\tau$, we can achieve $O(cm) = O(m)$ computation time using appropriate inverted index structures.
For $d_{m+1}, \ldots, d_{m+m'}$, we can use the formula

$$\begin{aligned}
\delta_i|_{\tau+\Delta\tau} &= \Pr(d_i)|_{\tau+\Delta\tau} \sum_{k=1}^{n+n'} tf(d_i, t_k)^2 \cdot idf(t_k)|_{\tau+\Delta\tau} \\
&= \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}} \sum_{k=1}^{n+n'} tf(d_i, t_k)^2 \cdot \frac{1}{df(t_k)|_{\tau+\Delta\tau}} \\
&= \frac{1}{tdw|_{\tau+\Delta\tau}} \sum_{k=1}^{n+n'} tf(d_i, t_k)^2 \cdot \frac{tdw|_{\tau+\Delta\tau}}{\widetilde{df}(t_k)|_{\tau+\Delta\tau}} \\
&= \sum_{k=1}^{n+n'} \frac{tf(d_i, t_k)^2}{\widetilde{df}(t_k)|_{\tau+\Delta\tau}}. \qquad (34)
\end{aligned}$$

It takes $O(m')$ time. Therefore, the overall computation cost in this step is $O(m + m') \approx O(m)$.

7. Updating δ'_i's: The formula of $\delta'_i|_\tau$ is transformed as follows:

$$\begin{aligned}
\delta'_i|_\tau &= \Pr(t_i|t_i) \\
&= idf(t_i)|_\tau \sum_{k=1}^{m} \Pr(d_k)|_\tau \cdot tf(d_k, t_i)^2 \\
&= \frac{tdw|_\tau}{\widetilde{df}(t_i)|_\tau} \sum_{k=1}^{m} \frac{dw_k|_\tau}{tdw|_\tau} \cdot tf(d_k, t_i)^2 \\
&= \frac{1}{\widetilde{df}(t_i)|_\tau} \sum_{k=1}^{m} dw_k|_\tau \cdot tf(d_k, t_i)^2 \qquad (35)
\end{aligned}$$

By defining $\widetilde{\delta}'_i|_\tau$ as

$$\widetilde{\delta}'_i|_\tau \stackrel{\text{def}}{=} \sum_{k=1}^{m} dw_k|_\tau \cdot tf(d_k, t_i)^2, \qquad (36)$$

we get

$$\delta'_i|_\tau = \frac{\widetilde{\delta}'_i|_\tau}{\widetilde{df}(t_i)|_\tau}. \qquad (37)$$

We store $\widetilde{\delta}'_i|_\tau$ instead of $\delta'_i|_\tau$ to enable the incremental update of δ'_i.
Since

$$\begin{aligned}
\widetilde{\delta}'_i|_{\tau+\Delta\tau} &= \sum_{k=1}^{m+m'} dw_k|_{\tau+\Delta\tau} \cdot tf(d_k, t_i)^2 \\
&= \sum_{k=1}^{m} dw_k|_{\tau+\Delta\tau} \cdot tf(d_k, t_i)^2 + \sum_{k=m+1}^{m+m'} dw_k|_{\tau+\Delta\tau} \cdot tf(d_k, t_i)^2 \\
&= \sum_{k=1}^{m} \lambda^{\Delta\tau} dw_k|_\tau \cdot tf(d_k, t_i)^2 + \sum_{k=m+1}^{m+m'} 1 \cdot tf(d_k, t_i)^2
\end{aligned}$$

$$= \lambda^{\Delta\tau} \cdot \widetilde{\delta}'_i|_\tau + \sum_{k=m+1}^{m+m'} tf(d_k, t_i)^2 \tag{38}$$

holds, by defining

$$\Delta t\, f_{\mathrm{sqsum}}(t_i) \stackrel{\mathrm{def}}{=} \sum_{k=m+1}^{m+m'} tf(d_k, t_i)^2, \tag{39}$$

we obtain the update formula

$$\widetilde{\delta}'_i|_{\tau+\Delta\tau} = \lambda^{\Delta\tau} \cdot \widetilde{\delta}'_i|_\tau + \Delta t\, f_{\mathrm{sqsum}}(t_i). \tag{40}$$

As the computation cost for a $\Delta t\, f_{\mathrm{sqsum}}(t_i)$ value is $O(m')$, the overall processing cost for the terms $t_1, \ldots, t_n$ becomes $O(m'n)$.
For the new terms $t_{n+1}, \ldots, t_{n+n'}$, we can use the formula

$$\widetilde{\delta}'_i|_{\tau+\Delta\tau} = \Delta t\, f_{\mathrm{sqsum}}(t_i) \tag{41}$$

by setting $\widetilde{\delta}'_i|_\tau = 0$ in Eq. (40). The calculation cost is $O(m'n')$. Therefore, the overall cost of this step is $O(m'(n+n')) \approx O(m'n)$.

Based on the above discussion, the total cost to update statistics and probabilities in an incremental manner is given by

$$O(m) + O(1) + O(m') + O(m'n) + O(m) + O(m'n) \approx O(m + m'n). \tag{42}$$

On the other hand, the naive scheme that calculate statistics and probabilities on each update has $O((m+m') \cdot (n+n')) \approx O(mn)$ computation time (see Appendix) and is expensive for on-line document clustering applications.

Now we summarize the above ideas. We persistently store and incrementally maintain the following statistics: dw_i's, tdw, $freq(d_i, t_k)$'s, $doclen_i$'s, $\widetilde{df}(t_k)$'s, and $\widetilde{\delta}'_k$'s, and achieve the update cost $O(m+n)$. Other statistics and probabilities ($\Pr(d_i)$'s, $tf(d_i, t_k)$'s, $df(t_k)$'s, δ_i's, and δ'_i's) are computed when they are needed.

In Fig. 1, we show the incremental statistics update algorithm. Although the algorithm shown in Fig. 1 is for the use in incremental updates, we can also use the same algorithm for the initialization case by setting $m = n = 0$. In this algorithm, we assume that the following statistical values are persistently maintained and readily available in its processing.

1. dw_i ($i = 1, \ldots, m$): document weights – $O(m)$ storage cost
2. tdw: the total weight of the documents – $O(1)$ storage cost
3. $freq(d_i, t_k)$ ($i = 1, \ldots, m$): document term frequencies – if we roughly assume that c terms are contained in a document, the total storage cost is $O(cm) = O(m)$.
4. $doclen_i$ ($i = 1, \ldots, m$): document lengths – $O(m)$ storage cost
5. $\widetilde{df}(t_k)$ ($k = 1, \ldots, n$): document frequencies of terms – $O(n)$ storage cost
6. $\widetilde{\delta}'_k$ ($k = 1, \ldots, n$): decoupling coefficients for terms – $O(n)$ storage cost

Therefore, the total storage cost becomes $O(m+n)$. This cost is worse than non-incremental update approach with the storage cost $O(m)$ (shown in Appendix), but is still linear for m and n.

```
1  // Preconditions:
2  // 1) ∀t : Δt f_sum(t) = 0, Δt f_sqsum(t) = 0
3  // 2) For i = m+1,...,m+m', ∀t : freq(d_i, t) = 0
4
5  // Only for the initialization case
6  if init_flag = true then
7     m := 0;
8     n := 0;
9     tdw := 0;
10 end
11
12 // Step 1: collect frequency information for new documents
13 for i := m+1 to m+m' do
14    doclen_i := 0;
15    foreach t in d_i do
16       doclen_i := doclen_i + 1;
17       freq(d_i, t) := freq(d_i, t) + 1;
18    end
19    foreach t such that freq(d_i, t) > 0 do
20       tf(d_i, t) := freq(d_i, t)/doclen_i;
21       Δt f_sum(t) := Δt f_sum(t) + tf(d_i, t);
22       Δt f_sqsum(t) := Δt f_sqsum(t) + tf(d_i, t)^2;
23    end
24 end
25
26 // Step 2: set or update document weights
27 for i := 1 to m do
28    dw_i := λ^Δτ · dw_i;
29 for i := m+1 to m+m' do
30    dw_i := 1;
31 tdw := λ^Δτ tdw + m';    // update the total weight of documents
32
33 // Step 3: set or update d̃f values
34 for k := 1 to n do
35    d̃f(t_k) := λ^Δτ · d̃f(t_k) + Δt f_sum(t_k);
36 for k := n+1 to n' do
37    d̃f(t_k) := Δt f_sum(t_k);
38
39 // Step 4: recompute decoupling coefficients for documents
40 for i := 1 to m+m' do
41    δ_i := 0;
42    foreach term t such that freq(d_i, t) > 0 do
43       tf(d_i, t) := freq(d_i, t)/doclen_i;
44       δ_i := δ_i + tf(d_i, t)^2 · dw_i/d̃f(t);
45 end
46
47 // Step 5: update decoupling coefficients for terms
48 for i := 1 to n do
49    δ̃'_i := λ^Δτ · δ̃'_i + Δt f_sqsum(t_i);
50    δ'_i := δ̃'_i/d̃f(t_i);
51 end
52 for i := n+1 to n+n' do
53    δ̃'_i := Δt f_sqsum(t_i);
54    δ'_i := δ̃'_i/d̃f(t_i);
55 end
```

Fig. 1. The Incremental Statistics Update Algorithm

5 The Incremental Cluster Update Algorithm

Figure 2 shows the incremental cluster update algorithm at the time $t = \tau$, where we assume that the statistical values required in the algorithm are already obtained in the statistics update algorithm shown in the previous section.

Here we roughly estimate its computation cost:

1. Step 1: The seed power formula at $t = \tau$ is given from Eq. (1) to Eq. (5):

$$sp_i|_{\tau+\Delta\tau} = \delta_i|_{\tau+\Delta\tau} \cdot (1 - \delta_i|_{\tau+\Delta\tau}) \cdot \sum_{j=1}^{n+n'} freq(d_i, t_j) \cdot \delta'_j|_{\tau+\Delta\tau} \cdot (1 - \delta'_j|_{\tau+\Delta\tau}). \quad (43)$$

   Since we already have $\delta_i|_{\tau+\Delta\tau}$, $\delta'_j|_{\tau+\Delta\tau}$, and $freq(d_i, t_j)$, we only have to compute the above summation. As this computation should be performed only for t_j such that $freq(d_i, t_j) > 0$, the computation time for each document is $O(c) = O(1)$. Therefore, the total cost for the entire document set becomes $O(1 \cdot (m + m')) \approx O(m)$.
2. In Step 2, the seed power values sp_i $(i = 1, \ldots, m)$ are sorted, and the largest n_c ones are selected as the seeds: we select top n_c documents from from *sorted_list* and insert into *new_seed_set*. If a new document, which is just now inserted into the document set, is selected as a seed, we delete it from *new_doc_set*. When the seed selection process is finished, we assign the value of *new_doc_set* to *nonseed_set* (the set of non-seed documents).
   As it takes $O(m \log m)$ time for the sorting and n_c is assumed to be rather smaller than m, this step roughly requires $O(m \log m)$ computation cost.
3. In Step 3, re-selection and deletion of clusters are performed, where *cluster_set* represents the current set of clusters. The formula $c \rightarrow seed()$ represents that we iteratively perform the following process for each cluster c.
   When s is re-selected as a seed, we only delete it from *new_seed_set*; otherwise we delete the cluster c and the documents in c are appended to *nonseed_set*. Finally, we generate a cluster for each new seed.
   It is difficult to estimate the computational cost of this step. However, we can expect that this step does not require much time compared to other steps in this algorithm.
4. In Step 4, we insert non-seed documents into the clusters. If we estimate the number of non-seed documents as $O(m)$ and the number of iterations as $O(n_c)$ (worst-case), the computation cost becomes $O(mn_c f(m, n))$, where f represents the computation cost of the function *sim* that calculates the similarity value between documents. If we roughly suppose $f(m, n)$ is constant, the worst-case cost of this step is given by $O(mn_c)$.

Based on the above consideration, we might be able to say that the total computation cost is $O(m \log m + mn_c)$.

The algorithm to compute a similarity score is given in Fig. 3. It is based on Eq. (19), but we have represented it by using available statistical values.

```
1  // Let the set of appended documents be new_doc_set
2  new_doc_set := {d_{m+1}, ..., d_{m+m'}};
3
4  // Only for initialization case
5  if init_flag = true then
6     m := 0;
7     cluster_set := ∅;
8  end
9
10 // Step 1: compute seed powers of documents
11 for i := 1 to m + m' do
12    sp_i := 0;
13    foreach t such that freq(d_i, t_k) > 0 do
14       sp_i := sp_i + freq(d_i, t_k) · δ'_k · (1 − δ'_k);
15    sp_i := sp_i · δ_i · (1 − δ_i);
16 end
17
18 // Step 2: select seeds
19 new_seed_set := ∅;
20 sorted_list := sort(sp_1, ..., sp_{m+m'}); // select
21 for i := 1 to n_c do
22    s := head(sorted_list);    // larger n_c seeds
23    sorted_list := tail(sorted_list);
24    new_seed_set := new_seed_set ∪ {s};
25    if s in new_doc_set then
26       new_doc_set := new_doc_set − {s};
27 end
28 nonseed_set := new_doc_set;    // non-seed document set
29
30 // Step 3: Update clusters
31 foreach c in cluster_set do
32    s := c→seed();    // obtain the seed of c
33    if s in new_seed_set then    // s is re-selected as a seed
34       new_seed_set := new_seed_set − {s};
35    else    // s is not selected: the cluster is deleted
36       cluster_set := cluster_set − {c};
37       nonseed_set := nonseed_set ∪ c→entries();
38       delete(c);    // delete the cluster c
39    end
40 end
41 foreach s in new_seed_set do
42    c := new_cluster(s);    // generate a new cluster
43    cluster_set := cluster_set ∪ {c};
44 end
45
46 // Step 4: remaining documents are inserted into clusters
47 foreach d in nonseed_set do
48    max_simval := 0;
49    foreach c in cluster_set do
50       simval := sim(d, c→seed());
51       if simval > max_simval then
52          max_simval := simval;
53          max_cluster := c;
54       end
55    end
56    max_cluster→add(d);
57 end
```

Fig. 2. The Incremental Clustering Algorithm

```
1  function sim(d_i, d_j)
2  begin
3     simval := 0;
4     foreach t such that tf(d_i, t) > 0 and tf(d_j, t) > 0 do
5           simval := simval + (freq(d_i,t)/doclen_i) · (freq(d_j,t)/doclen_j) · (tdw/(df(t))^2);
6     simval := simval · (dw_i/tdw) · (dw_j/tdw);
7  end
```

Fig. 3. The Similarity Score Calculation Algorithm

6 Document Expiration and Parameter Setting Methods

6.1 Expiration of Old Documents

We have not mentioned deletion of old documents until now. Since the F^{2}ICM method weights each document according to the novelty of the document, old documents have small document weights (dw_i's) and do not have effects on the clustering results. Since F^{2}ICM is based on the philosophy to neglect obsolete documents, we can remove too old documents from the targets of the clustering. Such removal will improve the storage overhead and the update overhead of F^{2}ICM.

To remove obsolete documents from the clustering target documents, we take the following approaches:

1. First we consider the deletion condition of old documents. In this paper, we take a simple approach: if the document weight dw_i for a document d_i satisfies the condition
$$dw_i \leq \varepsilon \tag{44}$$
for a small positive constant ε, we delete the document d_i. In practice, we delete the document weight dw_i, maintained as described in Section 4, from a persistent storage.
2. When we delete dw_i of the deleted document d_i, we have to propagate the deletion to other statistics. For tdw, the total weight of all the documents, we have to modify it as $tdw = tdw - dw_i$ according to its original definition. However, since now $dw_i \approx 0$, $tdw - dw_i \approx tdw$ holds so that we do not have to modify tdw actually.
3. We also need to delete $freq(d_i, t_k)$'s, the term occurrence frequencies for d_i, to reduce the storage cost. Therefore, we simply delete $freq(d_i, t_k)$'s for all the term t_k's that satisfy $freq(d_i, t_k) > 0$.
4. Additionally, we have to delete $\widetilde{df}(t_k)$ and $\widetilde{\delta}'_k$ for each term t_k contained in d_i, but we should remind that the term t_k may be contained in other documents. In such a case, we should not delete these values because they are still active. To solve this problem, we simply use a reference counter for each term: when the reference counter becomes zero, we can safely delete the statistics values for the term.

The incremental statistics update algorithm that incorporates the document deletion is shown in Fig. 4. The underlined parts are included for the deletion process. As easily estimated, the deletion does not take much cost when the numbers of deleted documents are not large.

Note that line 31 for the deletion of document d_i. In this algorithm, we assume that all the related information for d_i is deleted at this point and the following processes do not use the information. For example, we assume that d_i is omitted in the iteration from line 55 to line 60. This is for the simplification of the algorithm description and is easily implementable. Similar convention is used for the deleted terms.

Finally note that the incremental clustering algorithm shown in Fig. 2 can also be used for this statistics update algorithm without modification.

```
1  // Preconditions
2  // ∀t : Δt f_sum(t) = 0, Δt f_sqsum(t) = 0
3  // For i = m+1, ..., m+m', ∀t : freq(d_i, t) = 0
4
5  // Only for the initialization case
6  if init_flag = true then
7     m := 0;
8     n := 0;
9     tdw := 0;
10 end
11
12 // Step 1: collect frequency information for new documents
13 for i := m+1 to m+m' do
14    doclen_i := 0;
15    foreach t in d_i do
16       doclen_i := doclen_i + 1;
17       freq(d_i, t) := freq(d_i, t) + 1;
18    end
19    foreach t such that freq(d_i, t) > 0 do
20       tf(d_i, t) := freq(d_i, t)/doclen_i;
21       Δt f_sum(t) := Δt f_sum(t) + tf(d_i, t);
22       Δt f_sqsum(t) := Δt f_sqsum(t) + tf(d_i, t)^2;
23       ref_cnt(t) := ref_cnt(t) + 1;
24    end
25 end
26
27 // Step 2: set or update document weights
28 for i := 1 to m do
29    dw_i := λ^Δτ · dw_i;
30    if dw_i ≤ ε then
31       delete(d_i);    // delete d_i
32       delete(dw_i);
33       foreach t such that freq(d_i, t) > 0 do
34          delete(freq(d_i, t));
35          refcnt(t) := refcnt(t) − 1;
36          if refcnt(t) = 0 do    // delete information for t
37             delete(t_k);    // delete t_k
38             delete(d̃f(t_k));
39             delete(δ̃_k);    // also delete δ̃_k
40          end
41       end
42    end
43 for i := m+1 to m+m' do
44    dw_i := 1;
45 tdw := λ^Δτ tdw + m';    // update the total weight of documents
46
47 // Step 3: set or update d̃f values
48 for k := 1 to n do
49    d̃f(t_k) := λ^Δτ · d̃f(t_k) + Δt f_sum(t_k);
50 end
51 for k := n+1 to n' do
52    d̃f(t_k) := Δt f_sum(t_k);
53
54 // Step 4: recompute decoupling coefficients for documents
55 for i := 1 to m+m' do
56    δ_i := 0;
57    foreach t such that freq(d_i, t) > 0 do
58       tf(d_i, t) := freq(d_i, t)/doclen_i;
59       δ_i := δ_i + tf(d_i, t)^2 · dw_i/d̃f(t);
60 end
61
62 // Step 5: update decoupling coefficients for terms
63 for i := 1 to n do
64    δ̃'_i := λ^Δτ · δ̃'_i + Δt f_sqsum(t_i);
65    δ'_i := δ̃'_i/d̃f(t_i);
66 end
67 for i := n+1 to n+n' do
68    δ̃'_i := Δt f_sqsum(t_i);
69    δ'_i := δ̃'_i/d̃f(t_i);
70 end
```

Fig. 4. The Incremental Statistics Update Algorithm with Document Deletion

6.2 Methods for Parameter Setting

The F^{2}ICM method uses two parameters in its algorithms:

- a forgetting factor λ ($0 < \lambda < 1$) that specifies the speed of forgetting
- an expiration parameter ε ($0 < \varepsilon < 1$), the threshold value for document deletion

To help the user's decision for the parameter settings, we use the following metaphors to give intuitive meanings to them.

To set the parameter λ, we assume that the user gives a *half-life span* value β. It specifies the period that a document loses half of its weight. Namely, β satisfies $\lambda^\beta = \frac{1}{2}$. Therefore, λ can be derived as

$$\lambda = \exp\left(-\frac{\log 2}{\beta}\right). \tag{45}$$

For the parameter ε, we assume that the user gives a *life span* value γ. The γ value specifies the period that a document is "active" as the target of clustering. Therefore the expiration parameter ε can be derived by

$$\varepsilon = \lambda^\gamma. \tag{46}$$

These parameter setting methods are more intuitive than the direct setting of λ and ε and more easier for ordinal users.

7 Experimental Results

7.1 Dataset and Parameter Settings

In this section, we show two experimental results performed using F^{2}ICM. As the test dataset, we use an archive of Japanese newspaper articles of Mainichi Daily Newspaper for the year 1994. The archive is available as a CD-ROM format and articles in it are categorized by their issue dates and subject areas. A news article is typically assigned 50 to 150 keywords: we use such keywords as the index terms for an article. In the experiment, we mainly utilize articles on international affairs that issued in January and February in 1994. News articles are basically issued per-day basis and the number of news articles for each day is from 15 to 25.

In the experiments, we assume to perform the clustering procedure once in a day. The numbers of clusters n_c is fixed as $n_c = 10$ throughout the experiments. We set the half-life span parameter as $\beta = 7$. Namely, we assume that the value of an article reduces to $1/2$ in one week. Also, we set the life span parameter as $\gamma = 30$. Therefore, every document will be deleted from the clusters after 30 days from its incorporation.

7.2 Computational Cost for Clustering Sequences

First we show the experimental result on computation cost for daily clustering. Figure 5 plots the CPU time and the response time for each clustering performed everyday. The

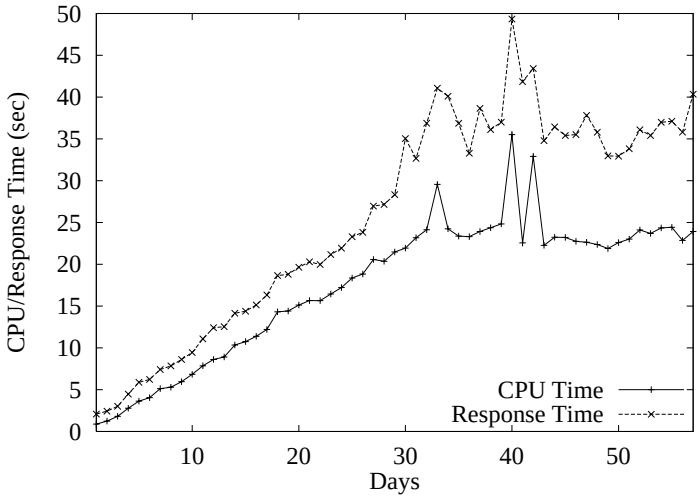

Fig. 5. CPU and Response Times of Clustering

x-axis represents passed days from the start date (January 1st, 1994) and ends with 57th day (March 1st, 1994)[5].

As shown in this figure, the CPU and response times increase almost linearly until 30th day. This is because the size of the target document set increases almost linearly until 30th day and because F^2ICM has near-linear computational cost. After 30 days, the processing cost turns to be almost constant. This is because after 30 days, not only new articles are inserted into the target document set, but also old articles are deleted from it. Therefore, the size of the target document set becomes almost constant after 30 days. We can observe the abrupt increases of the processing cost at 33rd, 40th, and 42nd days. The reason would be because there are many articles particularly for these three days. Based on this experiment, we can say that F^2ICM has constant processing cost for continual clustering tasks which are required in on-line environments.

7.3 Overview of the Clustering Results

In this subsection, we show the experimental results of the clustering from the standpoint of their qualities. Unfortunately, for the target dataset, there are no relevance judgments or ideal clustering results to be used for the comparison purpose. Therefore, we briefly review the result of the manual observation of the clustering results.

As an example, we summarize the clusters obtained after the clustering process at January 31, 1994 (30th day).

1. East Europe, NATO, Russia, Ukraine
2. Clinton(White Water/politics), military issue(Korea/Myanmar/Mexico/Indonesia)

[5] Although there are 60 days in this period, three no issue days exist in this dataset.

3. China(import and export/U.S.)
4. U.S. politics(economic sanctions on Vietnam/elections)
5. Clinton(Syria/South East issue/visiting Europe), Europe(France/Italy/Switzerland)
6. South Africa(ANC/human rights), East Europe(Boznia-Herzegovina, Croatia), Russia(Zhirinovsky/ruble/Ukraine)
7. Russia(economy/Moscow/U.S.), North Korea(IAEA/nuclear)
8. China(Patriot missiles/South Korea/Russia/Taiwan/economics)
9. Mexico(indigenous peoples/riot), Israel
10. South East Asia(Indonesia/Cambodia/Thailand), China(Taiwan/France), South Korea(politics),

Another example at March 1st, 1994 (57th day) is as follows:

1. Boznia-Herzegovina (NATO/PKO/UN/Serbia), China(diplomacy)
2. U.S. issue (Japan/economy/New Zealand/Boznia/Washington)
3. Myanmar, Russia, Mexico
4. Boznia-Herzegovina(Sarajevo/Serbia), U.S.(North Korea/economy/military)
5. North Korea(IAEA/U.S./nuclear)
6. East Asia(Hebron/random shooting/PLO), Myanmar, Boznia-Herzegovina
7. U.S.(society/crime/North Korea/IAEA)
8. U.N.(PKO/Boznia-Herzegovina/EU), China
9. Boznia-Herzegovina(U.N./PKO/Sarajevo), Russia(Moscow/Serbia)
10. Sarajevo(Boznia-Herzegovina), China(Taiwan/Tibet)

Based on the observation, we would be able to say that our method groups similar articles into a cluster as far as an appropriate article representing a specific topic is selected as a cluster seed, but a cluster obtained as the result of clustering usually contains multiple topics. This would partly due to the effect of the terms that commonly appear in news articles (e.g., U.S., China, military, president). To alleviate this problem, it would be beneficial to devise more sophisticated term weighting methods or to use thesauri to select effective index terms for document clustering.

As an another problem, we can observe that clustering results get worse in some cases because two or more articles belonging to the same topic are often selected as seed articles. This phenomenon is well observed in the result of March 1st, 1994 shown above. Since five seed articles are related to the topic "Boznia-Herzegovina issue", articles belonging to this topic are separately clustered in different clusters. This is because F^{2}ICM only uses seed powers in its seed selection step and does not consider similarities among the selected seed documents[6]. Based on this observation, we can say that we should devise a more sophisticated scheme for seed selection. As an another improvement, it may be useful to use two-step clustering approach (as in Scatter/Gather [4]) that consists of the first clustering step that clusters part of the documents with a costly, but high-quality clustering scheme, and the second clustering step that clusters remaining documents with a low-cost clustering scheme utilizing the result of the first clustering.

[6] In the paper of C^{2}ICM [3], it is mentioned that selection of seeds belonging to a same topic can be avoided using a threshold value to evaluate their similarity. But it is not clear how to set this parameter appropriately.

8 Conclusions and Future Work

In this paper, we have proposed an on-line document clustering method F^{2}ICM that is based on the notion of a forgetting factor to compute document similarities and to derive clustering results. The feature of F^{2}ICM is to "forget" past documents gradually and put high weights on newer documents than older documents to generate clusters. We have described the document similarity measure used in F^{2}ICM that incorporates the notion of a forgetting factor, the clustering algorithms, and the incremental statistics maintenance algorithm for the efficient update of clusters. We have briefly shown our experimental results performed on daily newspaper articles and analyzed the behaviors of F^{2}ICM.

As future work, we are planning to revise our clustering algorithms to improve the quality of the generated clusters. Also, we aim to devise an automatic estimation method of the number of clusters and a semi-automatic parameter setting method for the forgetting factor λ to achieve good clustering results. We are also planning to make more detailed experiments using other test data collections.

References

1. J.R. Anderson (ed.), *Rules of the Mind*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.
2. R. Baeza-Yates and B. Ribeiro-Neto. (eds.), *Modern Information Retrieval*, Addison-Wesley, 1999.
3. F. Can, "Incremental Clustering for Dynamic Information Processing", *ACM TOIS*, 11(2), pp. 143–164, 1993.
4. D.R. Cutting, D.R. Karger, J.O. Pedersen, "Constraint Interaction-Time Scatter/Gather Browsing of Very Large Document Collections", *Proc. ACM SIGIR*, pp. 126-134, 1993.
5. W.B. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structure & Algorithms*, Prentice-Hall, 1992.
6. A.K. Jain, M.N. Murty, P.J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, 31(3), 1999.
7. G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.
8. C.J. van Rijsbergen, *Information Retrieval* (2nd ed.), Butterworth, 1979.
9. Y. Yang, J.G. Carbonell, R.D. Brown, T. Pierce, B.T. Archibald, X. Liu, "Learning Approaches for Detecting and Tracking News Events", *IEEE Intelligent Systems*, 14(4), 1999.

Appendix: Statistics Recomputation Cost

We estimate the recomputation cost of statistics when new documents $d_{m+1}, \ldots, d_{m+m'}$ are inserted at $t = \tau + \Delta\tau$.

1. Updating $\Pr(d_i)$ ($i = 1, \ldots, m$): When $t = \tau + \Delta\tau$, they are given by

$$\Pr(d_i)|_{\tau+\Delta\tau} = \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}} = \frac{dw_i|_{\tau+\Delta\tau}}{\sum_{l=1}^{m+m'} dw_l|_{\tau+\Delta\tau}}.$$

   To compute them directly, the following processes are required:

(a) Compute $dw_i|_{\tau+\Delta\tau}$ for $i = 1, \ldots, m$.
(b) Compute

$$tdw|_{\tau+\Delta\tau} = \sum_{i=1}^{m+m'} dw_i|_{\tau+\Delta\tau} = \sum_{i=1}^{m} dw_i|_{\tau+\Delta\tau} + m'.$$

(c) For $i = 1, \ldots, m$, calculate

$$\Pr(d_i)|_{\tau+\Delta\tau} = \frac{dw_i|_{\tau+\Delta\tau}}{tdw|_{\tau+\Delta\tau}},$$

and for $i = m+1, \ldots, m+m'$, calculate

$$\Pr(d_i)|_{\tau+\Delta\tau} = \frac{1}{tdw|_{\tau+\Delta\tau}}.$$

Each computation of (a), (b), and (c) takes $O(m)$ time. Therefore, the total computation cost becomes $O(m)$. Note that we have to store the timestamps T_i ($i = 1, \ldots, m$) for the computation. The storage cost is $O(m)$.

2. For $tf(d_i, t_k)$, we use the same approach we took in Section 4. Therefore, the computational cost is $O(m')$.
3. For document frequencies, we need to calculate

$$df(t_k)|_{\tau+\Delta\tau} = \sum_{i=1}^{m+m'} \Pr(d_i)|_{\tau+\Delta\tau} tf(d_i, t_k)$$

for $k = 1, \ldots, n+n'$. Since we have already $\Pr(d_i)|_{\tau+\Delta\tau}$'s and $tf(d_i, t_k)$'s, the remaining process is to multiply them and take the summation. It looks like that it takes $O(m+m') \approx O(m)$ time, but we can estimate more realistic estimation considering the fact that not necessarily all of $d_1, \ldots, d_{m+m'}$ contain the term t_k.
We consider as follows. Let the probability that a term t is contained in a document be p and p is constant (does not depend on time and documents). Then we can estimate that the document set consisting of m documents has mp document that contains t. Since p is constant, we can assume that $O(mp) = O(m)$ documents contain t.
Therefore, we can assume that there are $O(m+m') \approx O(m)$ documents within $d_1, \ldots, d_{m+m'}$ that contain t_k ($k = 1, \ldots, n+n'$). If we assume that the computation cost of $\Pr(d_i)|_{\tau+\Delta\tau} tf(d_i, t_k)$ is constant, the process takes $O(m)$ time. Therefore, we can estimate the cost is $O(m \cdot (n+n')) \approx O(mn)$ for the total document set.
4. For decoupling coefficients

$$\delta_i|_{\tau+\Delta\tau} = \Pr(d_i)|_{\tau+\Delta\tau} \sum_{k=1}^{n+n'} tf(d_i, t_k)^2 \cdot idf(t_k)|_{\tau+\Delta\tau}$$

($i = 1, \ldots, m+m'$), we only compute the multiplication and summation since we already have $\Pr(d_i)|_{\tau+\Delta\tau}$, $tf(d_i, t_k)$, and $idf(t_k)|_{\tau+\Delta\tau}$. If we assume that each $tf(d_i, t_k)$ value that satisfies $tf(d_i, t_k) > 0$ is easily enumerable and each document contains c (constant) terms, the total computation cost for the document set becomes $O(1 \cdot (m+m')) \approx O(m)$.

5. For decoupling coefficients for term t_k,

$$\delta_i'|_{\tau+\Delta\tau} = idf(t_i)|_{\tau+\Delta\tau} \sum_{k=1}^{m+m'} \Pr(d_k)|_{\tau+\Delta\tau} \cdot tf(d_k, t_i)^2$$

($k = 1, \ldots, n+n'$), we can use the similar approach to the case of document frequencies, and get $O(m \cdot (n+n')) \approx O(mn)$ computation cost.

Based on the above consideration, the overall cost is estimated as

$$O(m) + O(m') + O(mn) + O(m) + O(mn) = O(mn). \tag{47}$$

For these computations, we need to maintain the following statistics:

- T_i ($i = 1, \ldots, m$): document timestamps – $O(m)$ storage cost
- $freq(d_i, t_k)$ ($i = 1, \ldots, m$): document frequencies – $O(m)$ storage cost
- $doclen_i$ ($i = 1, \ldots, m$): document lengths – $O(m)$ storage cost

Therefore, this approach requires $O(m)$ storage cost.