Supplemental Material: Accelerating K-Means Clustering for Document Data Sets with an Architecture-Friendly Pruning Method

Kazuo Aoyama
NTT Corporation
kazuo.aoyama@ntt.com

Kazumi Saito
Kanagawa University
k-saito@kanagawa-u.ac.jp

S1 FULL ALGORITHM OF AF-ICP

This section shows in Algorithms S1, S2, and S3 the pseudocode of the full algorithm of AF-ICP in Section 4.2. First, we show the pseudocodes of the assignment step, which correspond to Algorithms 2 and 3. They are rewritten in the form with scaled feature values [6]. The scaling is performed to avoid the multiplications for calculating the upper bounds on similarities. Next, we detail the update step at the rth iteration in AF-ICP in Algorithm S3, which is not described in Section 4.2.

Algorithm S1 Assignment step in AF-ICP algorithm

Input: $\hat{\mathcal{X}}$, $\check{\mathcal{M}}^{[r-1]}$, $\check{\mathcal{M}}^{p[r-1]}$, $\left\{\rho_{a(i)}^{[r-1]}\right\}_{i=1}^{N}$, $t^{[th]}$, $v^{[th]}$

Output: $C^{[r]} = \left\{C_j^{[r]}\right\}_{j=1}^{K}$

▷ Scale values before assignment step starts

1: **for all** $\hat{\boldsymbol{x}}_i = [\,(t_{(i,p)}, u_{t_{(i,p)}})\,]_{p=1}^{(nt)_i} \in \hat{\mathcal{X}}$ **do**
2: $\quad u_{t_{(i,p)}} \leftarrow u_{t_{(i,p)}} \cdot v^{[th]}$ ▷ Scaling

▷ Calculate similarities in parallel w.r.t $\hat{\mathcal{X}}$

3: $C_j^{[r]} \leftarrow \emptyset$, $j = 1, 2, \cdots, K$
4: **for all** $\hat{\boldsymbol{x}}_i = [\,(t_{(i,p)}, u_{t_{(i,p)}})\,]_{p=1}^{(nt)_i} \in \hat{\mathcal{X}}$ **do**
5: $\quad \{\rho_j\}_{j=1}^{K} \leftarrow 0,\ \rho_{(max)} \leftarrow \rho_{a(i)}^{[r-1]},\ \mathcal{Z}_i \leftarrow \emptyset$
6: $\quad$ **for** $j \leftarrow 1$ **to** K **do**
7: $\quad\quad y_{(i,j)} \leftarrow \sum_{t_{(i,p)} \geq t^{[th]}} (u_{t_{(i,p)}})$ ▷ Initializing $y_{(i,j)}$

▷ Gathering phase

8: $\quad$ **if** $xState = 1$ **then** ▷ ICP filtering
9: $\quad\quad \left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_1(\mathcal{Z}_i, \{(\rho_j,\ y_{(i,j)}\}_{j=1}^{K}, \rho_{(max)}, (nMv))$
10: $\quad$ **else**
11: $\quad\quad \left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_0(\mathcal{Z}_i, \{(\rho_j,\ y_{(i,j)}\}_{j=1}^{K}, \rho_{(max)})$

▷ Verification phase

12: $\quad$ **for** $(s \leftarrow t_{(i,p)}) \geq t^{[th]}$ **do** ▷ Region 3
13: $\quad\quad$ **for all** $j \in \mathcal{Z}_i$ **do** $\rho_j \leftarrow \rho_j + u_s \cdot w_{(s,j)}$
14: $\quad$ **for all** $j \in \mathcal{Z}_i$ **do**
15: $\quad\quad$ **If** $\rho_j > \rho_{(max)}$ **then** $\rho_{(max)} \leftarrow \rho_j$ and $a(i) \leftarrow j$
16: $\quad C_{a(i)}^{[r]} \leftarrow C_{a(i)}^{[r]} \cup \{\hat{\boldsymbol{x}}_i\}$

Algorithm S1 shows the assignment-step pseudocode. Scaling the data-object-feature values at lines 1 to 2 is performed once before our K-means clustering algorithm starts. Algorithm S2 shows G_1 function in Algorithm S1. Owing to scaling the data-object-feature values in Algorithm S1 and mean-feature values at line 5 in Algorithm S3 (appearing later), we can obtain the upper bound on similarities by only the addition at line 9 in Algorithm S2.

Algorithm S2 Candidate-gathering function: G_1

Input: $\hat{\boldsymbol{x}}_i$, $\check{\mathcal{M}}^{[r-1]}$, $t^{[th]}$, $v^{[th]}$, $\mathcal{Z}_i$, $\{\rho_j,\ y_{(i,j)}\}_{j=1}^{K}$, $\rho_{(max)}$, (nMv) ▷ (nMv): # moving centroids

Output: $\mathcal{Z}_i$, $\{\rho_j\}_{j=1}^{K}$

▷ Exact partial similarity calculation

1: **for** $(s \leftarrow t_{(i,p)}) < t^{[th]}$ in all term IDs in $\hat{\boldsymbol{x}}_i$ **do** ▷ Region 1
2: $\quad$ **for** $1 \leq q \leq (mfM)_s$ **do** ▷ Endpoint
3: $\quad\quad \rho_{c_{(s,q)}} \leftarrow \rho_{c_{(s,q)}} + u_s \cdot v_{c_{(s,q)}}$
4: **for** $(s \leftarrow t_{(i,p)}) \geq t^{[th]}$ in all term IDs in $\hat{\boldsymbol{x}}_i$ **do** ▷ Region 2
5: $\quad$ **for** $1 \leq q \leq (mfM)_s$ **do** ▷ Endpoint
6: $\quad\quad \rho_{c_{(s,q)}} \leftarrow \rho_{c_{(s,q)}} + u_s \cdot v_{c_{(s,q)}}$, $y_{(i,c_{(s,q)})} \leftarrow y_{(i,c_{(s,q)})} - u_s$

▷ Upper-bound calculation

7: **for** $1 \leq j' \leq (nMv)$ **do**
8: $\quad j'$ is transformed to j.
9: $\quad \rho_j^{[ub]} \leftarrow \rho_j + \boxed{y_{(i,j)}}$ ▷ Region 3 (UB): Scaling

▷ AF filtering

10: $\quad$ **if** $\rho_j^{[ub]} > \rho_{(max)}$ **then** $\mathcal{Z}_i \leftarrow \mathcal{Z}_i \cup \{j\}$

Algorithm S3 shows the pseudocode of the update step at the rth iteration. The algorithm performs the following five processes at lines 1 to 16.

(1) Making tentative mean-feature vector $\boldsymbol{\lambda}$ of the cluster $C_j^{[r]}$ at lines 2 to 5. In particular, at line 5, scaling the mean-feature values by $1/v^{[th]}$ is performed. The $\boldsymbol{\lambda}$ with full expression consists of D feature values λ_s $(s = 1, 2, \cdots, D)$.
(2) Calculating the similarity $\rho_{a(i)}^{[r]}$ of the ith object to the centroid of cluster $C_{a(i)}^{[r]}$ to which the object belong at lines 6 and 7. The $\rho_{a(i)}^{[r]}$ is utilized as the similarity threshold in the gathering phase at the assignment step at the next $(r+1)$ iteration, which is for determining whether the ith object satisfies the condition with respect to the ICP filter at line 10 in Algorithm 3 in Section 4.2.
(3) Constructing mean-inverted index $\check{\mathcal{M}}^{[r]}$ and partial mean-inverted index $\check{\mathcal{M}}^{p[r]}$ using the obtained $\boldsymbol{\lambda}$ at lines 8 to 11. $\check{\mathcal{M}}^{p[r]}$ (its element $w_{(s,j)}$) is utilized for exact similarity calculations in Region 3 at the assignment step in Algorithm S1. $\check{\mathcal{M}}^{p[r]}$ consists of $(D - t^{[th]} + 1)$ inverted-index arrays $\check{\zeta}_s^{[r]}$ whose lengths are K. The jth element in $\check{\zeta}_s$ is mean-feature values $w_{(s,j)}$ if $w_{(s,j)} < v^{[th]}$, otherwise 0. Due to this full expression, the jth element can be directly accessed by using the centroid ID j as the key.
(4) Structuring inverted-index arrays for the AF filter, which are $\check{\xi}_s^{[r]} \in \check{\mathcal{M}}^{[r]}$ $(t^{[th]} \leq s \leq D)$ at lines 14 to 15. Since

Algorithm S3 Update step in AF-ICP algorithm

Input: $\hat{\mathcal{X}},\ C^{[r]}=\left\{C_j^{[r]}\right\}_{j=1}^{K},\ t^{[th]}$

Output: $\check{\mathcal{M}}^{[r]},\ \check{\mathcal{M}}^{p[r]},\ \left\{\rho_{a(i)}^{[r]}\right\}_{i=1}^{N},\ [(mfH)_s]_{s=t^{[th]}}^{D}$

1: **for all** $C_j^{[r]} \in C^{[r]}$ **do**

▷ (1) Making tentative mean-feature vector $\boldsymbol{\lambda}$

2: $\boldsymbol{\lambda}=(\lambda_s)_{s=1}^{D} \leftarrow \mathbf{0}$ ▷ Tentative mean-feature vector

3: **for all** $\hat{\boldsymbol{x}}_i=\left[(t_{(i,p)}, u_{t_{(i,p)}})\right]_{p=1}^{(nt)_i} \in C_j^{[r]}$ **do**

4: $\lambda_{t_{(i,p)}} \leftarrow \lambda_{t_{(i,p)}} + u_{t_{(i,p)}}$

5: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}/|C_j^{[r]}|,\ \boldsymbol{\lambda} \leftarrow \boxed{\boldsymbol{\lambda}/(v^{[th]} \cdot \|\boldsymbol{\lambda}\|_2)}$ ▷ Scaling

▷ (2) Calculating similarities $\rho_{a(i)}^{[r]}$

6: **for all** $\hat{\boldsymbol{x}}_i=\left[(t_{(i,p)}, u_{t_{(i,p)}})\right]_{p=1}^{(nt)_i} \in C_j^{[r]}$ **do**

7: $\rho_{a(i)}^{[r]} \leftarrow \sum_{p=1}^{(nt)_i} u_{t_{(i,p)}} \cdot \lambda_{t_{(i,p)}}$

▷ (3) Constructing mean-inverted indexes

8: **for all** term ID s of $\lambda_s > 0$ **do** ▷ Initialize $q \leftarrow 1$

9: $(c_{(s,q)}, v_{c_{(s,q)}}) \in \check{\xi}_s^{[r]} \subset \check{\mathcal{M}}^{[r]} \leftarrow (j, \lambda_s)$ ▷ q++

10: **if** $s \geq t^{[th]}$ and $\lambda_s < v^{[th]}$ **then**

11: $w_{(s,j)}\ (\in \check{\boldsymbol{\xi}}_s^{[r]} \subset \check{\mathcal{M}}^{p[r]}) \leftarrow \lambda_s$

▷ (4) and (5) Structuring $\check{\xi}_s^{[r]} \in \check{\mathcal{M}}^{[r]}$

12: **for all** $[\check{\xi}_s^{[r]}]_{s=1}^{t^{[th]}-1} \subset \check{\mathcal{M}}^{[r]}$ **do**

13: Divide $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mf)_s}$ into two blocks of $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mfM)_s}$ and $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=(mfM)_s+1}^{(mf)_s}$.

14: **for all** $[\check{\xi}_s^{[r]}]_{s=t^{[th]}}^{D} \subset \check{\mathcal{M}}^{[r]}$ **do**

15: Pick up tuples from $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mf)_s} \in \check{\xi}_s^{[r]}$ that satisfy $v_{c_{(s,q)}} \geq v^{[th]}$ and make $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mfH)_s}$ from them.

16: Divide $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mfH)_s}$ into two blocks of $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=1}^{(mfM)_s}$ and $[(c_{(s,q)}, v_{c_{(s,q)}})]_{q=(mfM)_s+1}^{(mfH)_s}$.

▷ Estimating structural parameters (Sections S2 and S3)

17: **if** $r \in \{1, 2\}$ **then**

18: $(t^{[th]}, v^{[th]}) \leftarrow$ *EstParams* $(\check{\mathcal{M}}^{[r]}, \hat{\mathcal{X}},$ other arguments) (detailed in Section S3)

19: Reconstruct $\check{\mathcal{M}}^{[r]}$ using new $(t^{[th]}, v^{[th]})$

mean-feature values of $v_{c_{(s,q)}} \geq v^{[th]}$ are used for calculating exact partial similarities in Region 2, the tuples containing such values are placed from the top of $\check{\xi}_s^{[r]}$ to the $(mfH)_s$th entry.

(5) Structuring inverted-index arrays $\check{\xi}_s^{[r]}$ for the ICP filter at lines 12 to 13 in Region 1 and at line 16 in Region 2.

Furthermore, at the first and second iterations, two structural parameters $t^{[th]}$ and $v^{[th]}$ are estimated by using *EstParams* function in Algorithm S4 and described in Sections S2 and S3. $\check{\mathcal{M}}^{p[r]}$ is constructed after the structural-parameter estimation and then $\check{\mathcal{M}}^{[r]}$ is updated.

S2 OBJECTIVE FUNCTION IN SECTION 5

We derive Prob $(\rho^{[ub]}(i) \geq \rho_{a(i)}\,; s', h)$ in Eq. (10) where s' denotes a candidate of $t^{[th]}$ and h represents $v_h^{[th]}$ that is the hth candidate of $v^{[th]}$ and the objective function in Eq. (6) by using the following three assumptions (ASMs).

ASM 1 A distribution of similarities of the ith object to centroids that are more than or equal to an average similarity $\bar{\rho}(i)$ is expressed as the following distribution in an exponential family:

$$F(\rho|\rho \geq \bar{\rho}(i)) = b \cdot e^{-a(\rho-\bar{\rho}(i))}, \tag{S1}$$

where the similarity distribution ρ is considered with a continuous relaxation of the discrete values ρ_j that denotes the similarity of $\boldsymbol{x}_i$ to the jth centroid, and a and b are parameters.

ASM 2 The distribution of ρ is one of the exponential family and the definite integral value from a median of ρ to the infinity of $F(\rho)$ can be regarded as that from an average of ρ to the infinity. This leads to the following equation:

$$\int_{\bar{\rho}(i)}^{\infty} F(\rho)\, d\rho = \frac{b}{a} = \frac{1}{e}. \tag{S2}$$

ASM 3 Similarity-upper-bound distribution $F^{[ub]}(\rho)$ is expressed by the parallel translation of $F(\rho)$ by $\Delta\bar{\rho}(i; s', h)$ as

$$\begin{aligned} &F^{[ub]}(\rho - \Delta\bar{\rho}(i; s', h)) = F(\rho) \\ &\Delta\bar{\rho}(i; s', h) = \bar{\rho}^{[ub]}(i\,; s', h) - \bar{\rho}(i)\,, \end{aligned} \tag{S3}$$

where the second equation is the same as Eq. (11).

Since an object belongs to a distinct cluster, i.e., hard clustering, the following equation holds:

$$\int_{\rho_{a(i)}}^{\infty} F(\rho)\, d\rho = \left(\frac{1}{e}\right) e^{-a(\rho_{a(i)}-\bar{\rho}(i))} = \frac{1}{K}, \tag{S4}$$

where $\rho_{a(i)}$ denotes the similarity of the ith object to the centroid of the cluster which the object belongs to. For simplicity, the superscript of $[r]$ for $\rho_{a(i)}^{[r]}$, which represents the rth iteration, is omitted. For the first equality in Eq. (S4), Eq. (S2) is used. Using Eqs. (S2) and (S4), the parameters a and b in Eq. (S1) are expressed as

$$a = \frac{\log(K/e)}{\rho_{a(i)} - \bar{\rho}(i)} \qquad \text{and} \qquad b = \frac{a}{e}. \tag{S5}$$

Then, the probability in Eq. (9) is expressed by

$$\begin{aligned} &\text{Prob}\,(\rho^{[ub]}(i) \geq \rho_{a(i)}\,; s', h) \\ &= \int_{\rho_{a(i)}-\Delta\bar{\rho}(i;s',h)}^{\infty} F(\rho)\, d\rho = \left(\frac{1}{K}\right)\left(\frac{K}{e}\right)^{\frac{\Delta\bar{\rho}(i;s',h)}{\rho_{a(i)}-\bar{\rho}(i)}}. \end{aligned} \tag{S6}$$

Eq. (9) is rewritten as

$$(\tilde{\varphi}3)_{(s',h)} = \sum_{i=1}^{N} (ntH)_{(i;s')} \cdot \left(\frac{K}{e}\right)^{\frac{\Delta\bar{\rho}(i;s',h)}{\rho_{a(i)}-\bar{\rho}(i)}}.$$

$\Delta\bar{\rho}(i;s',h)$ is obtained by substituting $\bar{\rho}(i)$ and $\bar{\rho}^{[ub]}(i;s',h)$ in Eqs. (S7) and (S8) into Eq. (S3).

$$\bar{\rho}(i) = \sum_{p=1}^{(nt)_i} \left(u_{t_{(i,p)}} \cdot \frac{1}{K} \sum_{q=1}^{(mf)_s} v_{c_{(s,q)}} \right) \quad \text{and} \quad s = t_{(i,p)}\,, \tag{S7}$$

where $t_{(i,p)}$ and $c_{(s,q)}$ denote the term ID appeared at the pth position in $\hat{\boldsymbol{x}}_i$ and the mean ID (cluster ID) appeared at the qth position in the inverted-index-array with the sth term ID (see Table 1).

$$\bar{\rho}^{[ub]}(i;s',h) = \sum_{s=1}^{s'-1} \left(u_s \cdot \frac{1}{K} \sum_{q=1}^{(mf)_s} v_{c_{(s,q)}} \right) + \sum_{s=s'}^{D} u_s \cdot \frac{1}{K} \left\{ \sum_{q=1}^{(mfH)_s} v_{c_{(s,q)}} + v_h^{[th]}((mf)_s - (mfH)_s) \right\}, \tag{S8}$$

where $(mfH)_{(s,v_h^{[th]})}$ is simplified as $(mfH)_s$ and u_s is $u_{t_{(i,p)}}$ if $u_{t_{(i,p)}}$ exits for $t_{(i,p)} \geq t^{[th]}$, 0 otherwise.

We can obtain the objective funciton by substituting Eqs. (7), (8), and (12) into Eq. (6) as

$$J(s', v_h^{[th]}) = \sum_{s=1}^{s'-1} (df)_s \cdot (mf)_s + \sum_{s=s'}^{D} (df)_s \cdot (mfH)_{(s,v_h^{[th]})} + \sum_{i=1}^{N} (ntH)_{(i,s')} \cdot \left(\frac{K}{e}\right)^{\frac{\Delta\bar{\rho}(i;s',h)}{\rho_{a(i)} - \bar{\rho}_i}}.$$

S3 PARAMETER ESTIMATION IN SECTION 5

This section describes our proposed efficient practical algorithm for simultaneously estimating $t^{[th]}$ and $v^{[th]}$. For the parameter estimation, we suppose that an appropriate $t^{[th]}$ exists near D on the grounds of the universal characteristics (UCs) in Section 3, i.e., the skewed forms of both the document frequency (df) and mean frequency (mf) with respect to term ID sorted in ascending order of df in Figs. 2 and 3, and $v^{[th]}$ is in the range of small values due to the skewed form of the mean-feature-values in each inverted-index array. This allows us to narrow down a search space for the two parameters.

The estimation algorithm minimizes the objective function in Eq. (6) that is rewritten as

$$(t^{[th]}, v^{[th]}) = \underset{\substack{v_h^{[th]} \in V^{[th]} \\ s_{(min)} \leq s' \leq D}}{\arg\min} \left(J(s', v_h^{[th]}) \right)$$

$$J(s', v_h^{[th]}) = \tilde{\phi}_{(s',h)} = (\phi 1)_{s'} + (\phi 2)_{(s',h)} + (\tilde{\phi} 3)_{(s',h)}\,,$$

where s' denotes the term ID that is a candidate of $t^{[th]}$ and decremented one by one from D to the pre-determined minimum term ID of $s_{(min)}$ and h represents the hth candidate of $v^{[th]}$, i.e., $v_h^{[th]}$. Based on the presumption of $t^{[th]}$ exists near D, the estimation algorithm efficiently calculates $\tilde{\phi}_{(s',h)}$ for each $v_h^{[th]}$[1] by exploiting a recurrence relation with respect to s' expressed by

$$\begin{aligned} \tilde{\phi}_{(s';h)} &= \tilde{\phi}_{(s'+1;h)} - (df)_{s'} \cdot (mfL)_{(s';v_h^{[th]})} \\ &\quad + (\tilde{\phi} 3)_{(s';h)} - (\tilde{\phi} 3)_{(s'+1;h)} \qquad \text{(S9)} \\ \tilde{\phi}_{(D+1;h)} &= \sum_{s=1}^{D} (df)_s \cdot (mf)_s \triangleq \phi \\ (\tilde{\phi} 3)_{(D+1;h)} &= 0\,, \end{aligned}$$

where $(mfL)_{(s';v_h^{[th]})} = (mf)_{s'} - (mfH)_{(s';v_h^{[th]})}$, i.e., $(mfL)_{(s';v_h^{[th]})}$ denotes the number of centroids whose feature values are lower than $v_h^{[th]}$.

Moreover, $(\tilde{\phi} 3)_{(s';h)}$ itself in Eq. (S9) is represented with a recurrence relation. We rewrite the approximate number of multiplications with respect to the ith object in Eq. (12) as

$$(\tilde{\phi} 3)_{(i,s';h)} = (ntH)_{(i,s')} \cdot \left(\frac{K}{e}\right)^{\frac{\bar{\rho}^{[ub]}(i,s';h) - \bar{\rho}_i}{\rho_{a(i)} - \bar{\rho}_i}}. \tag{S10}$$

We consider two cases of the ith object contains the term whose ID is s', i.e., $s' \in \{t_{(i,p)}\}_{p=1}^{(nt)_i}$, and the other. In the latter case, $(\tilde{\phi} 3)_{(i,s';h)}$ is invariant from $(\tilde{\phi} 3)_{(i,s'+1;h)}$ as

$$(\tilde{\phi} 3)_{(i,s';h)} = (\tilde{\phi} 3)_{(i,s'+1;h)}\,. \tag{S11}$$

In the former case, we derive the recurrence relation while focusing on $(ntH)_{(i,s')}$ and $\bar{\rho}^{[ub]}(i,s';h)$ in Eq. S10. First, the following equation with respect to $(ntH)_{(i,s')}$ holds.

$$(ntH)_{(i,s')} = (ntH)_{(i,s'+1)} + 1\,. \tag{S12}$$

Next, $\bar{\rho}^{[ub]}(i,s';h)$ is expressed by

$$\bar{\rho}^{[ub]}(i,s') = \bar{\rho}^{[ub]}(i,s'+1) + \underbrace{\left\{ \sum_{q=(mfH)_{s'}+1}^{(mf)_{s'}} (v_h^{[th]} - v_{c_{(s,q)}}) + (K - (mf)_{s'}) v_h^{[th]} \right\} \left(\frac{u_{s'}}{K}\right)}_{\Delta\bar{v}_{(i,s')} \cdot u_{s'}}, \tag{S13}$$

where $u_{s'}$ denotes $u_{t_{(i,p)=s'}}$ and $\Delta\bar{v}_{(i,s')}$ is the average difference between $v_h^{[th]}$ and the exact mean-feature values in Region 3 at the s'th term ID. Then, $\bar{\rho}^{[ub]}(i,s';h)$ is expressed with the recurrence relation as

$$\begin{aligned} (\tilde{\phi} 3)_{(i,s';h)} &= (ntH)_{(i,s')} \cdot \left(\frac{K}{e}\right)^{\frac{\bar{\rho}^{[ub]}(i,s') - \bar{\rho}_i}{\rho_{a(i)} - \bar{\rho}_i}} \\ &= \left(\frac{K}{e}\right)^{\gamma(i,s')} \left[\left\{ (ntH)_{(i,s'+1)} + 1 \right\} \left(\frac{K}{e}\right)^{\frac{\bar{\rho}^{[ub]}(i,s'+1) - \bar{\rho}_i}{\rho_{a(i)} - \bar{\rho}_i}} \right] \qquad \text{(S14)} \\ \gamma(i,s') &= \frac{\Delta\bar{v}_{(i,s')} \cdot u_{s'}}{\rho_{a(i)} - \bar{\rho}_i}\,, \qquad \text{(S15)} \end{aligned}$$

where the values in the square bracket are already obtained at the $(s'+1)$th term ID. The boundary conditions are set as

$$(\tilde{\phi} 3)_{(i,D+1;h)} = 0, \quad (ntH)_{(i,D+1)} = 0, \quad \bar{\rho}^{[ub]}(i,D+1) = \bar{\rho}_i\,. \tag{S16}$$

By applying Eqs. (S11) and (S14) to Eq. (S9), we make the recurrence relation. To efficiently calculate $\tilde{\phi}_{(s';h)}$ with it, we structure the data-object set.

[1] This process for each $v_h^{[th]}$ is executed in parallel processing.

Algorithm S4 Parameter-estimation function: *EstParams*

Input: $\check{\mathcal{M}},\ \hat{\mathcal{X}},\ \check{\mathcal{X}}^p,\ V^{[th]} = \left\{v_1^{[th]}, \cdots, v_{|V^{[th]}|}^{[th]}\right\},\ s_{(min)}$

Output: $\left(t^{[th]},\ v^{[th]}\right)$

▷ Initialization

1: $\phi = \sum_{s=1}^{D} (df)_s \cdot (mf)_s$

2: **for all** $\hat{\boldsymbol{x}}_i \in \hat{\mathcal{X}}$ **do**

3: $\quad \bar{\rho}_i \leftarrow (1/K) \sum_{p=1}^{(ntH)_i} \sum_{q=1}^{(mf)_{s=t_{(i,p)}}} (v_{c_{(s,q)}} \cdot u_{t_{(i,p)}})$

▷ Parallel processing

4: **for all** $v_h^{[th]} \in V^{[th]}$ **do**

5: $\quad \tilde{\phi}_{(D+1,h)} \leftarrow \phi$

6: $\quad (ntH)_{(i,D+1)} \leftarrow 0,\ \Delta v_{(i,D+1)} \leftarrow 0 \quad \text{for} \quad 1 \le i \le N$

7: $\quad$ **for** $s' \leftarrow D$ **to** $s_{(min)}$ **do** ▷ s': $t^{[th]}$ candidate

8: $\quad\quad$ **for all** $i = o_{(s',q')} \in \check{\boldsymbol{\eta}}_{s'} \subset \check{\mathcal{X}}^p$ **do**

9: $\quad\quad\quad \tilde{\phi}_{(s',h)} \leftarrow \tilde{\phi}_{(s'+1,h)} - (ntH)_{(i,s'+1)} (K/e)^{\gamma_{(i,s'+1)}}$

10: $\quad\quad\quad \Delta v_{(i,s')} \leftarrow \Delta v_{(i,s'+1)} + \sum_{q=(mfH)'_{s'}+1}^{(mf)_{s'}} (v_h^{[th]} - v_{c_{(s',q)}}) + (K - (mf)_{s'}) \cdot v_h^{[th]}$

11: $\quad\quad\quad \Delta \bar{v}_{(i,s')} \leftarrow \Delta v_{(i,s')}/K$

12: $\quad\quad\quad \gamma_{(i,s')} \leftarrow (\Delta \bar{v}_{(i,s')} \cdot \check{u}_i)/(\rho_{a(i)} - \bar{\rho}_i)$

13: $\quad\quad\quad \tilde{\phi}_{(s',h)} \leftarrow \tilde{\phi}_{(s',h)} + (ntH)_{(i,s')} (K/e)^{\gamma_{(i,s')}}$

14: $\quad\quad \tilde{\phi}_{(s',h)} \leftarrow \tilde{\phi}_{(s',h)} - (df)_{s'} \cdot (mfL)_{(s';v_h^{[th]})}$

15: $\quad\quad J(s', v_h^{[th]}) = \tilde{\phi}_{(s',h)}$

16: $\quad (t_h^{[th]}, v_h^{[th]}) \leftarrow \arg\min_{s_{(min)} \le s' \le D} \left(J(s', v_h^{[th]})\right)$

17: $(t^{[th]}, v^{[th]}) \leftarrow \arg\min_{1 \le h \le |V^{[th]}|} \left(J(t_h^{[th]}, v_h^{[th]})\right)$

Table S1: Notation for *EstParams* function

Symbol	Description and Definitions
$J(s', v_h^{[th]})$	**Function** of $(s', v_h^{[th]})$ that returns the approximation number of the multiplications s': Term ID as $t^{[th]}$ candidate $v_h^{[th]}$: Mean-feature-value as $v^{[th]}$ candidate $v_h^{[th]}$ can be represented with h for simplicty.
$V^{[th]}$	**Set** of $v^{[th]}$ candidates, $v_h^{[th]}$, $h = 1, \cdots, \lvert V^{[th]}\rvert$
$(\tilde{\phi})_{(s',h)}$	Approximate number of multiplications for similarity calculations, given s' and h
$(df)_s$	Document frequency of sth term ID
$(mf)_s$	Mean frequency of sth term ID
$s_{(min)}$	Minimum term ID of candidates (s') of $t^{[th]}$, $s_{(min)} \le s' \le D$
$\check{\mathcal{X}}^p$	**Partial inverted-index** of objects, each column of which is object-tuple array $\check{\boldsymbol{\eta}}_s,\ s \ge s_{(min)}$ $\check{\boldsymbol{\eta}}_s = [(o_{(s,q')}, \check{u}_{o_{(s,q')}})]_{q'=1}^{(df)_s}$ $o_{(s,q')}$: q'th object ID appeared in $\check{\boldsymbol{\eta}}_s$ $\check{u}_{o_{(s,q')}}$: q'th feature value appeared in $\check{\boldsymbol{\eta}}_s$ $(df)_s$: Document frequency of term ID, s
$\bar{\rho}_i$	Average similarity of ith object to all centroids
$(ntH)_{(i,t^{th})}$	Number of terms in ith object whose ID is higher than or equal to given $t^{[th]}$
$(mfH)_{(s;v_h^{[th]})}$	Mean frequency of term ID s where $v_{(s,q)} \ge v_h^{[th]}$, where q is a position in $\check{\xi}_s$, given $v_h^{[th]}$
$(mfL)_{(s;v_h^{[th]})}$	$(mfL)_{(s;v_h^{[th]})} = (mf)_s - (mfH)_{(s;v_h^{[th]})}$

When using the recurrence relation, we need to access a data-object- and a mean-feature value at the sth term ID. Although the mean-feature value is accessed with the mean-inverted index $\check{\mathcal{M}}$, it is difficult to select only the data-object feature value with the current data structure of $\hat{\mathcal{X}}$ with a low computational cost. To overcome the difficulty in the architecture-friendly manner (AFM) in Section 2, we introduce a partial object-inverted-index $\check{\mathcal{X}}^p$ for the term IDs from $s_{(min)}$ to D with low memory consumption. By using the partial object-inverted-index, we can access both the feature values at the sth term ID without using conditional branches causing many mispredictions or loading the large array of the object data set, resulting in calculating $(\tilde{\phi}3)_{(i,s';h)}$ efficiently.

Algorithm S4 and Table S1 show a pseudocode of the practical algorithm *EstParams* and its notation, respectively. Given the $v^{[th]}$ candidate $v_h^{[th]} \in V^{[th]}$, at lines 4 to 16, the tuple of $(t_h^{[th]}, v_h^{[th]})$ is obtained for each $v_h^{[th]}$, where $t_h^{[th]}$ denotes the s' that minimizes $J(s', v_h^{[th]})$. At line 17, $(t^{[th]}, v^{[th]})$ is determined as the tuple that minimizes $J(t_h^{[th]}, v_h^{[th]})$ among all the tuples. At lines 7 to 15, the recurrent relation is used with the initial state of $s' = D$. By using the inverted-index-array $\check{\eta}_{s'}$ in the data-object inverted index $\check{\mathcal{X}}^p$, we can efficiently access the object-feature value $\check{u}_i$ having the s' term ID and calculate the approximate number of the multiplications $\tilde{\phi}_{(s',h)}$ at lines 8 to 15.

To confirm the effectiveness of the parameter-estimation algorithm *EstParams* in Algorithm S4, we incorporated it to our proposed K-means clustering algorithm AF-ICP in Section 4 and applied AF-ICP with $K = 80\,000$ to the 8.2M-sized PubMed data set in Section 6. We executed *EstParams* twice, at the first and the second iteration, in AF-ICP. The purpose of its execution at the first iteration is only to reduce the elapsed time in the second iteration. Clustering results at the first iteration strongly depends on initial settings and some centroids often change their positions significantly. At the second iteration, *EstParams* determines parameter values that are utilized at the successive iterations.

We compared the approximate number of the multiplications obtained by *EstParams* with the corresponding actual number calculated by AF-ICP, where $t^{[th]}$ and $v^{[th]}$ were set at the values estimated at the second iteration. Figure S1 shows the comparison with the approximate and actual number of the multiplications along $v_h^{[th]}$ at which $t_h^{[th]}$ at line 16 in Algorithm S4 was a different value except the small $v_h^{[th]}$ values. The parameters in *EstParams* were set as $s_{(min)} = 1.22 \times 10^5$ and $0.020 \le v_h^{[th]} \le 0.060$ by 0.001 step. We observed that the approximate number of the multiplications agreed with the actual number in all the range and each value of $v_h^{[th]}$. The minimum number of the multiplications was observed at the identical value of 0.038 between the estimated and the actual number. Figure S2 shows the actual number of the multiplications when $t^{[th]}$ was set at the various fixed values from 1.22×10^5 to 1.32×10^5. Comparing the approximate number of the multiplications in Fig. S1 with the actual ones in Fig. S2, we notice that the approximate number corresponds to the lowest envelop curve of the actual numbers. Thus, our parameter-estimation algorithm *EstParams* can find appropriate values for $t^{[th]}$ and $v^{[th]}$ simultaneously.

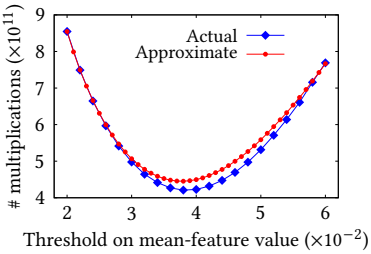

Figure S1: Comparison of the approximate and actual number of the multiplications along threshold $v_h^{[th]}$ on the mean-feature value for 8.2M-sized PubMed with K=80 000. The other threshold s' on term ID at each point differs from others in general.

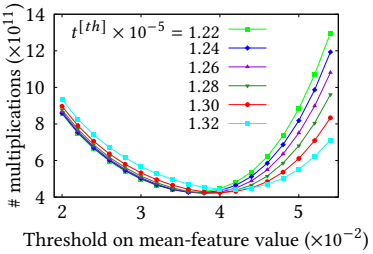

Figure S2: Actual number of the multiplications along threshold on the mean-feature value with various $t^{[th]}$ for 8.2M-sized PubMed with K=80 000.

S4 ABLATION STUDY

We analyze the contributions of components in our proposed algorithm AF-ICP to its performance. We focus on the main filter AF since the performance of only the auxiliary filter ICP is evaluated in Section 6.3. The AF filter exploits the tight upper bounds on the similarities obtained by using the three regions divided by the two parameters of $t^{[th]}$ and $v^{[th]}$. We regarded the components as the two parameters and prepared three algorithms without ICP in addition to the baseline MIVI.

The first algorithm was AF that employed the two parameters. The second was ThV that utilized only the parameter $v^{[th]}$ estimated by $t^{[th]} = 0$ [2]. ThV calculated the upper-bound similarities based on the $v^{[th]}$ in the range of all the term IDs. These upper bounds were looser than AF since the exact partial similarities in Region 1 of AF were replaced with the corresponding upper bounds. For calculating the exact similarities of unpruned centroids, ThV needed mean-inverted index $\check{M}^{p[r]}$ of the memory size of $\{D \times K \times \text{sizeof(double)}\}$ bytes. The last was ThT that utilized $t^{[th]}$ estimated by $v^{[th]} = 1.0$ [3]. The upper bounds $\rho^{[ub]'}_{(j;i)}$ corresponding to those in Eq. (3) are expressed by $\rho^{[ub]'}_{(j;i)} = (\rho 1')_{(j;i)} + \|\boldsymbol{x}_i^{p'}\|_1$, where $\|\boldsymbol{x}_i^{p'}\|$ and $(\rho 1')_{(j;i)}$ denote the partial L_1 norm of the ith object and the exact partial similarity of the object to the jth centroid based on ThT's $t^{[th]}$.

[2]ThV's $v^{[th]}$ was 0.032 while AF's 0.038 for 8.2M-sized PubMed with $K = 80\,000$.

[3]ThT's $t^{[th]}$ was 140 904 while AF's 128 090 for 8.2M-sized PubMed with $K = 80\,000$.

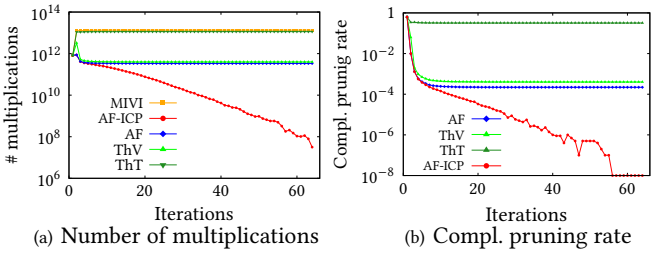

Figure S3: Ablation study in 8.2M-sized PubMed data set with K=80 000: (a) Number of multiplications and (b) Complementary pruning rate along iterations until convergence.

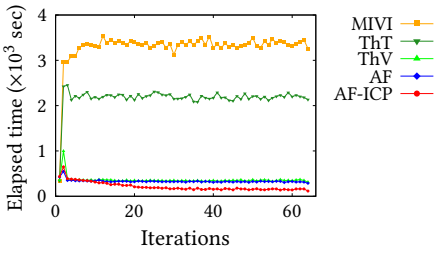

Figure S4: Elapsed time along iterations until convergence in 8.2M-sized PubMed data set with K=80 000.

Table S2: Ablation study of proposed algorithm AF-ICP in 8.2M-sized PubMed data set with K=80 000. Value in each column is rate to that of AF-ICP.

Algo.	Avg Mult	Avg time	Inst	BM	LLCM	Max MEM
AF	3.793	1.546	1.782	1.658	3.205	0.998
ThV	4.832	1.676	1.925	1.956	3.461	5.777
ThT	119.4	10.16	13.39	7.173	11.54	0.5094

Figure S3(a) and (b) show the number of the multiplications (Mult) and the complementary pruning rate (CPR) [4] along iterations until convergence when the algorithms were applied to the 8.2M-sized PubMed data set with $K = 80\,000$. AF and ThV successfully reduced both the multiplications and the CPR at the early stage in the iterations while ThT barely did. This observation shows that ThT had much looser upper bounds than AF and ThV. We know that the AF's pruning performance was mainly supported by the upper bounds based on the parameter $v^{[th]}$.

Figure S4 shows the elapsed time that each algorithm required in the same setting in Fig. S3. Table S2 shows the performance rates of the compared algorithms to AF-ICP. AF and ThV had the similar characteristics to each other in terms of the elapsed time and the performance-degradation factors of Inst, BM, and LLCM. ThV needed the high memory capacity for its partial mean-inverted index $\check{M}^{p[r]}$. We consider that the difference between these two algorithms and AF-ICP comes from the number of unpruned centroids in Fig. S3. ThT had the inferior performance to the others except the maximum memory usage. This shows that the parameter $t^{[th]}$ contributed to not the pruning performance but rather the low memory usage for the partial mean-inverted index $\check{\mathcal{M}}^{p[r]}$ of $\{(D - t^{[th]} + 1) \times K \times \text{sizeof(double)}\}$-byte capacity. Compared with

[4]The values of CPR are bounded below by 1×10^{-8}.

MIVI in Table 4, ThT slightly decreased Mult and Inst with the aid of its filter while it increased BM and LLCM. This was caused by ThT's poor filter passing many centroids in Fig. S3(b). Then, many branch mispredictions occurred in the conditional branch at the gathering phase, which judges whether $\rho^{[ub]}_{(j;i)}$ is larger than $\rho_{(max)}$ or not and mean-feature values in the partial mean-inverted index for the exact similarity calculations. were loaded many times.

In reference, the comparisons of actual performance of AF-ICP, AF, ThV, and ThT are shown in Tables S3, S4, S5, and S6. Tables S3 and S4 correspond to those when the algorithms were applied to the 8.2M-sized PubMed data set with $K = 80\,000$ and Tables S5 and S6 correspond to those when they were applied to the 1M-sized NYT data set with $K = 10\,000$.

Consequently, the parameters of $v^{[th]}$ and $t^{[th]}$ mainly contribute to the high pruning performance and the low memory usage, respectively. The proposed algorithm AF-ICP with the two filters of AF and ICP efficiently worked at high speed and with low memory consumption from the early to the last stage in the iterations.

S5 ALGORITHM PERFORMANCE IN SECTION 2

The algorithm performance of MIVI, DIVI and Ding$^+$ in Section 2 are shown as the rates to the baseline algorithm MIVI. This section shows the actual values of their performance.

Tables S7 shows the average values per iteration of the number of multiplications and the elapsed time (sec) when the algorithms were executed at an identical initial state in the 8.2M-sized PubMed data set with $K = 80\,000$ until the convergence where they needed 64 iterations. Table S8 shows their perf results (total amounts), where the numbers of branches and last-level-cache (LLC) loads are added in addition to the evaluation items in Table 2 in Section 2. We notice two facts in the perf results. One is that Ding$^+$ and DIVI needed many LLC-loads and respectively caused LLC-load misses of around 99% and 80% of the loads. The other is that Ding$^+$ failed around 10% of conditional-branch predictions while MIVI did only 0.04%. These caused the increase of elapsed time of Ding$^+$ and DIVI.

S6 COMPARED ALGORITHMS IN SECTION 6.3

This section details TA-ICP and CS-ICP in Section 6.3. Both the algorithms employ the mean-inverted index that is partitioned by the structural parameter $t^{[th]}$ (threshold on term IDs) into two regions like the proposed algorithm AF-ICP. The differences from AF-ICP are in their main upper-bound-based pruning (UBP) filters and data structures related to the filters.

S6.1 TA-ICP

TA-ICP is characterized by using (1) the UBP filter every an individual object and (2) two mean-inverted indexes for moving centroids and all the centroids. Each array of the mean-inverted index is sorted in descending order of the mean-feature values. Remind that AF-ICP's array is not sorted but classified in Section 4.1.

The UBP filter of TA-ICP is designed, inspired by the threshold algorithm (TA) in Fagin$^+$ and Li$^+$ algorithms. This filter utilizes a threshold ($v^{[th]}_{(ta)i}$) on mean-feature values, which is defined every the ith object by Eq. (16) as follows.

$$v^{[th]}_{(ta)i} = \rho_{(max)} / \|\boldsymbol{x}_i\|_1 ,$$

where $\|\boldsymbol{x}_i\|_1$ and $\rho_{(max)}$ denote the L_1-norm of the ith object-feature vector and the similarity of ith object to the centroid to which the object belongs at the last iteration. Due to the threshold for the individual object, it is difficult to structure the mean-inverted index when incorporating the auxiliary ICP filter unlike AF-ICP. TA-ICP employs the two mean-inverted indexes for moving and all the centroids.

Algorithm S5 Assignment step at the rth iteration in TA-ICP

1: **Input:** $\hat{\mathcal{X}}$, $\check{\mathcal{M}}^{[r-1]}$, $\check{\mathcal{M}}^{p[r-1]}$, $\left\{\rho^{[r-1]}_{a(i)}\right\}_{i=1}^{N}$, $t^{[th]}$, $\check{\mathcal{M}}^{[r-1]}_{mv}$
2: **Output:** $C^{[r]} = \left\{C^{[r]}_j\right\}_{j=1}^{K}$
3: $C^{[r]}_j \leftarrow \emptyset$, $j = 1, 2, \cdots, K$
▷ `Calculate similarities in parallel w.r.t` $\hat{\mathcal{X}}$
4: **for all** $\hat{\boldsymbol{x}}_i = [\,(t_{(i,p)}, u_{t_{(i,p)}})\,]_{p=1}^{(nt)_i} \in \hat{\mathcal{X}}$ **do**
5: $\mathcal{Z}_i \leftarrow \emptyset$, $\{\rho_j\}_{j=1}^{K} \leftarrow 0$, $\rho_{(max)} \leftarrow \rho^{[r-1]}_{a(i)}$, $\{y_{(i,j)}\}_{j=1}^{K} \leftarrow \sum_{t_{(i,p)} \geq t^{[th]}} u_{t_{(i,p)}}$
6: $\boxed{v^{[th]}_{(ta)i} \leftarrow \rho_{(max)} / \|\boldsymbol{x}_i\|_1}$ ▷ `Individual threshold`
▷ `Gathering phase`
7: **if** $xState = 1$ **then** ▷ `Object satisties Eq. (4).`
8: args $\leftarrow \mathcal{Z}_i$, $\{\rho_j, y_{(i,j)}\}_{j=1}^{K}$, $\rho_{(max)}$, $v^{[th]}_{(ta)i}$
9: $\left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_{(ta)1}(\text{args})$
10: **else**
11: $\left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_{(ta)0}(\text{args})$
▷ `Exact-similarity calculation for unpruned centroids`
12: **for** $(s \leftarrow t_{(i,p)}) \geq t^{[th]}$ **do**
13: **for all** $j \in \mathcal{Z}_i$ **do**
14: **if** $\boxed{w'_{(s,j)} < v^{[th]}_{(ta)i}}$ **then**
15: $\rho_j \leftarrow \rho_j + u_s \cdot w'_{(s,j)}$
▷ `Verification phase: The same as that in Algorithm S1.`
16: $C^{[r]}_{a(i)} \leftarrow C^{[r]}_{a(i)} \cup \{\hat{\boldsymbol{x}}_i\}$

Algorithms S5 and S6 show the pseudocodes of the assignment step in TA-ICP. At line 6 in Algorithm S5, the individual threshold is set. In the gathering phase from lines 7 to 11, the candidate-centroid-ID set $\mathcal{Z}_i$ for exact similarity calculations is made by the UBP and ICP filters. The exact similarity calculations for the centroids with IDs in $\mathcal{Z}_i$ are performed at lines 12 to 15. $w'_{(s,j)}$ at line 14 denotes the mean-feature value of the jth centroid in the sth array in the partial mean-inverted index that is different from the AF-ICP's counter part in including all the mean-feature values.

Algorithm S6 shows the pseudocode of the candidate-gathering function $G_{(ta)0}$ at line 11 in Algorithm S5. $G_{(ta)0}$ is used for the object that does not satisfy the condition in Eq. (4). When calculating the exact partial similarity of the ith object to the centroids whose

Algorithm S6 Candidate-gathering function: $G_{(ta)0}$

Input: $\hat{\boldsymbol{x}}_i,\ \check{\mathcal{M}}^{[r-1]},\ t^{[th]},$
$\mathcal{Z}_i,\ \{\rho_j, y_{(i,j)}\}_{j=1}^{K},\ \rho_{(max)},\ v_{(ta)i}^{[th]}$
Output: $\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}$

▷ Exact partial similarity calculation
1: **for** $(s \leftarrow t_{(i,p)}) < t^{[th]}$ for p in $\hat{\boldsymbol{x}}_i$ **do** ▷ Region 1
2: **for** $1 \le q \le (mf)_s$ **do**
3: $\rho_{c_{(s,q)}} \leftarrow \rho_{c_{(s,q)}} + u_s \cdot v_{c_{(s,q)}}$
4: **for** $(s \leftarrow t_{(i,p)}) \ge t^{[th]}$ for p in $\hat{\boldsymbol{x}}_i$ **do** ▷ Region 2
5: **for** $1 \le q \le (mf)_s$ **do**
6: **If** $v_{c(s,q)} < v_{(ta)i}^{[th]}$ **then** break
7: $\rho_{c_{(s,q)}} \leftarrow \rho_{c_{(s,q)}} + u_s \cdot v_{c_{(s,q)}}$
8: $y_{(i,c_{(s,q)})} \leftarrow y_{(i,c_{(s,q)})} - u_s$
▷ Gathering phase
9: **for** $1 \le j \le K$ **do**
10: **If** $\rho_j = 0$ **then** continue
11: $\rho_j^{[ub]} \leftarrow \rho_j + v_{(ta)i}^{[th]} \cdot y_{(i,j)}$ ▷ UB calculation
12: **if** $\rho_j^{[ub]} > \rho_{(max)}$ **then** ▷ UBP filter
13: $\mathcal{Z}_i \leftarrow \mathcal{Z}_i \cup \{j\}$

ID is $c_{(s,q)}$, $G_{(ta)0}$ judges whether its feature value satisfies the condition of $v_{c_{(s,q)}} \ge v_{(ta)i}^{[th]}$ or not at line 6. After the exact partial-similarity calculations in Regions 1 and 2 (defined by $v_{(ta)i}^{[th]}$), the upper bound on the similarity to the centroid whose exact partial-similarity is not zero is calculated. If the exact partial-similarity is zero, the upper bound is smaller than or equal to $\rho_{(max)}$ from the definition of $v_{(ta)i}^{[th]} = \rho_{(max)} / \|\boldsymbol{x}_i\|_1$. In the gathering phase, the centroid-IDs of the centroids passing through the TA filter are collected in $\mathcal{Z}_i$.

The function $G_{(ta)1}$ at line 9 in Algorithm S5 has the identical structure to $G_{(ta)0}$. The difference is that $G_{(ta)1}$ uses the moving-centroid mean-inverted index instead of $\check{\mathcal{M}}^{[r-1]}$. For this difference, $(mf)_s$ at lines 2 and 5 in $G_{(ta)0}$ is replaced with $(mfM)_s$ in $G_{(ta)1}$.

S6.2 CS-ICP

CS-ICP is characterized by using (1) the Cauchy-Schwarz inequality that is applied to mean-feature vectors in a subspace of an individual object-feature vector, (2) an additional mean-inverted index that stores squared mean-feature values, and (3) no threshold on the mean-feature values unlike AF-ICP and TA-ICP.

Algorithm S7 shows the pseudocode of the assignment step in CS-ICP. The $\|\boldsymbol{x}_i^p\|_2$ and $\check{\mathcal{M}}_{sq}^{p[r-1]}$ in the inputs respectively denote the L_2-norm of the ith object's *partial* feature vector and the *partial* squared mean-inverted index in the range of the term IDs (s) satisfying $s \ge t^{[th]}$. At lines 5 to 7, the candidate centroids for the exact similarity calculations are collected by the candidate-gathering functions $G_{(cs)0}$ and $G_{(cs)1}$. The other parts have the same algorithm structure as those in Algorithm S1 although each array in the partial mean-feature-inverted index for CS-ICP contains all the mean-feature values unlike AF-ICP whose array contains mean-feature values smaller than $v^{[th]}$ in Region 3.

Algorithm S7 Assignment step at the rth iteration in CS-ICP

Input: $\hat{\mathcal{X}},\ \check{\mathcal{M}}^{[r-1]},\ \check{\mathcal{M}}^{p[r-1]},\ \left\{\rho_{a(i)}^{[r-1]}\right\}_{i=1}^{N},\ \{\|\boldsymbol{x}_i^p\|_2\}_{i=1}^{N},$
$\check{\mathcal{M}}_{sq}^{p[r-1]},\ t^{[th]}$
Output: $C^{[r]} = \left\{C_j^{[r]}\right\}_{j=1}^{K}$

1: $C_j^{[r]} \leftarrow \emptyset\ ,\ j = 1, 2, \cdots, K$
▷ Calculate similarities in parallel w.r.t $\hat{\mathcal{X}}$
2: **for all** $\hat{\boldsymbol{x}}_i = [(t_{(i,p)}, u_{t_{(i,p)}})]_{p=1}^{(nt)_i} \in \hat{\mathcal{X}}$ **do**
3: $\{\rho_j\}_{j=1}^{K} \leftarrow 0,\ \mathcal{Z}_i \leftarrow \emptyset,\ \rho_{(max)} \leftarrow \rho_{a(i)}^{[r-1]}$
▷ Gathering phase
4: **if** $xState = 1$ **then** ▷ Object satisties Eq. (4).
5: $\left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_{(cs)1}(\mathcal{Z}_i, \{\rho_j\}_{j=1}^{K}, \rho_{(max)})$
6: **else**
7: $\left(\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}\right) = G_{(cs)0}(\mathcal{Z}_i, \{\rho_j\}_{j=1}^{K}, \rho_{(max)})$
▷ Exact-similarity calculation for unpruned centroids:
The same algorithm structure as that in Algorithm S1
▷ Verification phase: The same as that in Algorithm S1
8: $C_{a(i)}^{[r]} \leftarrow C_{a(i)}^{[r]} \cup \{\hat{\boldsymbol{x}}_i\}$

Algorithm S8 Candidate-gathering function: $G_{(cs)0}$

Input: $\hat{\boldsymbol{x}}_i,\ \check{\mathcal{M}}^{[r-1]},\ \mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K},\ \rho_{(max)},\ t^{[th]},$
$\|\boldsymbol{x}_i^p\|_2,\ \check{\mathcal{M}}_{sq}^{p[r-1]}$
Output: $\mathcal{Z}_i,\ \{\rho_j\}_{j=1}^{K}$

1: $\{\|\boldsymbol{\mu}_j^p\|_2^2\}_{j=1}^{K} \leftarrow 0$ ▷ Initialization
▷ Exact partial similarity calculation
2: **for** $(s \leftarrow t_{(i,p)}) < t^{[th]}$ in all term IDs in $\hat{\boldsymbol{x}}_i$ **do**
3: **for** $1 \le q \le (mf)_s$ **do**
4: $\rho_{c_{(s,q)}} \leftarrow \rho_{c_{(s,q)}} + u_s \cdot v_{c_{(s,q)}}$
▷ Calculate squared mean-L2-norm in object-subspace
5: **for** $(s \leftarrow t_{(i,p)}) \ge t^{[th]}$ in all term IDs in $\hat{\boldsymbol{x}}_i$ **do**
6: **for** $1 \le q \le (mf)_s$ **do**
7: $\|\boldsymbol{\mu}_{c(s,q)}^p\|_2^2 \leftarrow \|\boldsymbol{\mu}_{c(s,q)}^p\|_2^2 + v_{c(s,q)}^2$
▷ Gathering phase
8: **for** $1 \le j' \le (nMv)$ **do** ▷ nMv: #moving centroids
9: j' is transformed to j.
10: $\rho_j^{[ub]} \leftarrow \rho_j + \|\boldsymbol{x}_i^p\|_2 \times \sqrt{\|\boldsymbol{\mu}_j^p\|_2^2}$ ▷ UB calculation
11: **if** $\rho_j^{[ub]} > \rho_{(max)}$ **then** ▷ UBP filter
12: $\mathcal{Z}_i \leftarrow \mathcal{Z}_i \cup \{j\}$

Algorithm S8 shows the pseudocode of the candidate-gathering function $G_{(cs)0}$ at line 7 in Algorithm S7. $G_{(cs)0}$ is used for the object that does not satisfy the condition in Eq. (4). At lines 2 to 4, the exact partial similarities $\rho_{c_{(s,q)}}$ are calculated in the range of $t_{(i,p)} < t^{[th]}$. At lines 5 to 7, the squared L_2-norm of the partial mean-feature vector $\|\boldsymbol{\mu}_{c_{(s,q)}}^p\|_2^2$ is calculated in the subspace of the ith object-feature vector, where $v_{c_{(s,q)}}^2$ denotes the squared mean-feature value in $\check{\mathcal{M}}_{sq}^{p[r-1]}$. At lines 8 to 12, the gathering phase is performed and the centroid-ID set $\mathcal{Z}_i$ for exact similarity calculations is returned. At line 10, the upper bound on the similarity to the jth centroid $\rho_j^{[ub]}$ is calculated by the sum of the

foregoing exact partial similarity ρ_j and the product of the precalculated L_2-norm of the ith object's partial feature vector $\|\boldsymbol{x}_i^p\|_2$ and $\sqrt{\|\boldsymbol{\mu}_{c_{(s,q)}}^p\|_2^2}$. In this product calculation, the square-root operation is performed, which requires high computational cost.

The function $G_{(cs)1}$ at line 5 in Algorithm S7 is similar to $G_{(cs)0}$. The difference is in the endpoint of the loops at lines 3 and 6. $G_{(cs)0}$ uses $(mf)_s$ while $G_{(cs)1}$ does $(mfM)_s$ in the mean-inverted index consisting of the moving centroids.

S6.3 Performance Comparison

This section shows the actual performance of AF-ICP, ICP, CS-ICP, and TA-ICP when they were applied to the 8.2M-sized PubMed data set with $K = 80\,000$ in Tables S9 and S10 and the 1M-sized NYT data set with $K = 10\,000$ in Tables S11 and S12.

From the both results in the two distinct data sets and settings, in addition to the facts described in Section 6.3, we note the following two facts. (1) CS-ICP and TA-ICP required more elapsed time than the baseline ICP although they operated using the smaller or around equal number of (completed) instructions or multiplications. This is because they caused the larger numbers of branch mispredictions (BMs) and last-level-cache misses (LLCMs). In particular, TA-ICP did much more branch mispredictions than the others while their numbers of branch instructions were not much different. This led to its worse performance. (2) The elapsed-time differences of the four algorithms came from those in the assignment step which we focused on. The four algorithms worked in not much different average update time although they constructed the different data structures. In particular, AF-ICP and CS-ICP took less update time than the baseline ICP because they processed fewer centroids owing to their high pruning rates.

In the NYT results in Table S11, we note that surprisingly, AF-ICP took less elapsed time in the assignment step than that in the update step. For any further acceleration in such data sets and settings, we will need to change our strategy of the acceleration in the assignment step to that in both the steps including the update.

These, in addition to the results in Section 6.3, show that only reducing the numbers of multiplications or instructions does not always lead to reducing elapsed time required by an algorithm. Suppressing the performance-degradation factors in Section 2 realizes the acceleration. In other words, by carefully designing an algorithm so that it suppresses the numbers of instructions, last-level cache misses (LLCMs), and branch mispredictions (BMs), we can obtain an efficient algorithm that operates in the *architecture-friendly manner*.

We also note that reducing the maximum memory size required by AF-ICP, which is described as the remaining task in Section 6.3, is not a serious problem with respect to the actual memory size and the LLCMs in the current data sets and settings from Tables S9 and S11. The memory sizes used for the partial mean-inverted indexes $\check{\mathcal{M}}^{p[r]}$ in the 8.2M-sized PubMed and 1M-sized NYT data sets in the foregoing settings were around 8.3 GB and 3.2 GB, respectively. Even for the PubMed data set, total amount size was 16.72 GB. A current standard computer system for scientific and technical computing equips a memory system whose size is much larger than the foregoing amount. Regarding LLCMs, the number of accesses to $\check{\mathcal{M}}^{p[r]}$ were small, i.e., $\check{\mathcal{M}}^{p[r]}$ was rarely loaded to the caches because AF-ICP's filters reduced unpruned centroids that were targets for exact similarity calculations using $\check{\mathcal{M}}^{p[r]}$.

S7 MAIN-FILTER COMPARISON

There may be a doubt of the combination of the main UBP and auxiliary ICP filters in the compared algorithms weakens the main filter's positive effect on the performance, To clear up the doubt, this section compares only the main filters of AF-ICP, TA-ICP, and CS-ICP, where algorithms with the filters are respectively called AF-MIVI, TA-MIVI, and CS-MIVI. The algorithms were applied to the 8.2M-sized PubMed data set with $K = 80\,000$ in Tables S13 and S14 and the 1M-sized NYT data set with $K = 10\,000$ in Tables S15 and S16.

The most important fact is that no algorithm without the ICP filter improved its elapsed time in the data sets and settings, compared with the corresponding algorithm with the ICP filter. This is because the algorithm with the ICP filter suppressed the performance-degradation factors of the numbers of instructions, branch mispredictions (BMs), and last-level-cache misses (LLCMs).

AF-MIVI showed the best performance regardless of the data sets and settings. We note that the AF filter was effective by itself and from the AF-ICP's results in Tables S9 to S12, both the filters of AF and ICP worked without losing each other's advantages. When combining plural filters like the UBP and ICP filters, we should carefully design the combined algorithm that operates in the architecture-friendly manner.

Table S3: Ablation study in terms of average number of multiplications and average elapsed time until convergence in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
AF-ICP	9.391×10^{10}	213.8 [185.5, 28.79]	16.72
AF	3.562×10^{11}	330.6 [304.0, 27.00]	16.69
ThV	4.538×10^{11}	358.4 [329.37, 29.50]	96.59
ThT	1.122×10^{13}	2172 [2149, 23.81]	8.52

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S4: Ablation study in terms of perf results until convergence in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
AF-ICP	6.157×10^{14}	8.417×10^{13}	9.569×10^{10} (0.11)	1.043×10^{13}	1.738×10^{12} (16.7)
AF	1.097×10^{15}	1.659×10^{14}	1.586×10^{11} (0.10)	3.586×10^{13}	5.569×10^{12} (15.53)
ThV	1.185×10^{15}	1.725×10^{14}	1.872×10^{11} (0.11)	4.218×10^{13}	6.015×10^{12} (14.26)
ThT	8.244×10^{15}	9.009×10^{14}	6.864×10^{12} (0.08)	2.713×10^{14}	2.004×10^{13} (7.39)

Table S5: Ablation study in terms of average number of multiplications and average elapsed time until convergence in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
AF-ICP	2.411×10^{10}	15.77 [5.394, 10.50]	7.914
AF	9.411×10^{10}	26.00 [15.72, 10.41]	7.907
ThV	1.430×10^{11}	32.79 [21.99, 10.94]	43.00
ThT	2.385×10^{12}	238.5 [229.6, 9.016]	4.752

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S6: Ablation study in terms of perf results until convergence in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
AF-ICP	7.003×10^{13}	1.239×10^{13}	4.127×10^{10} (0.33)	8.470×10^{11}	1.044×10^{11} (12.3)
AF	1.538×10^{14}	1.993×10^{13}	7.044×10^{10} (0.35)	3.046×10^{12}	2.727×10^{11} (8.95)
ThV	1.931×10^{14}	2.324×10^{13}	7.395×10^{10} (0.32)	3.408×10^{12}	4.071×10^{11} (11.95)
ThT	2.017×10^{15}	2.091×10^{14}	8.155×10^{10} (0.04)	2.508×10^{13}	2.215×10^{12} (8.83)

Table S7: Performance of MIVI, DIVI, and Ding$^+$ in 8.2M-sized PubMed data set with K=80 000. Number of iterations unitl convergence is 64.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec)
MIVI	1.326×10^{13}	3.302×10^{3}
DIVI	1.326×10^{13}	3.372×10^{4}
Ding$^+$	3.029×10^{12}	9.552×10^{3}

Table S8: perf results of MIVI, DIVI, and Ding$^+$ in 8.2M-sized PubMed data set with K=80 000. Number of iterations unitl convergence is 64.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
MIVI	1.024×10^{16}	1.013×10^{15}	3.929×10^{11} (0.04)	2.790×10^{14}	1.767×10^{13} (6.33)
DIVI	1.006×10^{16}	1.019×10^{15}	2.743×10^{12} (0.27)	7.983×10^{14}	6.444×10^{14} (80.7)
Ding$^+$	6.691×10^{15}	1.949×10^{15}	1.937×10^{14} (9.94)	6.637×10^{14}	6.577×10^{14} (99.1)

Table S9: Performance comparison of AF-ICP, ICP, CS-ICP and TA-ICP in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
AF-ICP	9.391×10^{10}	204.8 [176.4, 28.86]	16.72
ICP	2.960×10^{12}	759.5 [729.9, 30.05]	8.285
CS-ICP	1.733×10^{11}	901.7 [875.2, 26.99]	18.31
TA-ICP	9.069×10^{11}	1042 [1006, 36.35]	19.07

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S10: perf results of AF-ICP, ICP, CS-ICP, and TA-ICP in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
AF-ICP	6.197×10^{14}	8.471×10^{13}	9.623×10^{10} (0.11)	1.037×10^{13}	1.619×10^{12} (15.6)
ICP	2.876×10^{15}	2.934×10^{14}	2.796×10^{11} (0.10)	8.277×10^{13}	4.467×10^{12} (5.40)
CS-ICP	2.346×10^{15}	2.786×10^{14}	3.127×10^{11} (0.11)	9.297×10^{13}	8.025×10^{12} (8.63)
TA-ICP	1.476×10^{15}	2.067×10^{14}	1.859×10^{12} (0.90)	4.593×10^{13}	2.209×10^{13} (48.10)

Table S11: Performance comparison of AF-ICP, ICP, CS-ICP and TA-ICP in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
AF-ICP	2.411×10^{10}	15.83 [5.466, 10.50]	7.914
ICP	3.947×10^{11}	68.13 [57.22, 11.05]	4.147
CS-ICP	2.137×10^{10}	86.16 [76.89, 9.380]	8.419
TA-ICP	2.909×10^{11}	107.6 [94.44, 13.32]	8.645

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S12: perf results of AF-ICP, ICP, CS-ICP, and TA-ICP in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
AF-ICP	7.041×10^{13}	1.246×10^{13}	4.094×10^{10} (0.33)	8.340×10^{11}	1.051×10^{11} (12.6)
ICP	4.065×10^{14}	4.363×10^{13}	5.652×10^{10} (0.13)	4.734×10^{12}	4.196×10^{11} (8.86)
CS-ICP	3.437×10^{14}	4.322×10^{13}	6.778×10^{10} (0.16)	5.783×10^{12}	1.456×10^{12} (25.18)
TA-ICP	4.264×10^{14}	5.876×10^{13}	4.321×10^{11} (0.74)	6.894×10^{12}	2.103×10^{12} (30.50)

Table S13: Performance comparison of MIVI, AF-, CS-, and TA-MIVI in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
MIVI	1.326×10^{13}	3302 [3278, 23.81]	8.251
AF-MIVI	3.562×10^{11}	266.7 [237.8, 29.38]	16.69
CS-MIVI	7.601×10^{11}	2760 [2733, 27.53]	18.28
TA-MIVI	3.856×10^{12}	3380 [3342, 37.99]	17.21

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S14: perf results of MIVI, AF-, CS-, and TA-MIVI in 8.2M-sized PubMed data set with K=80 000. Number of iterations until convergence is 64.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
MIVI	1.024×10^{16}	1.014×10^{15}	3.929×10^{11} (0.04)	2.790×10^{14}	1.767×10^{13} (6.33)
AF-MIVI	1.062×10^{15}	1.677×10^{14}	1.842×10^{11} (0.11)	3.592×10^{13}	4.522×10^{12} (12.6)
CS-MIVI	9.065×10^{15}	1.042×10^{15}	6.201×10^{11} (0.06)	3.073×10^{14}	2.711×10^{13} (8.82)
TA-MIVI	4.824×10^{15}	6.895×10^{14}	7.428×10^{12} (1.08)	1.754×10^{14}	6.626×10^{13} (37.77)

Table S15: Performance comparison of MIVI, AF-, CS-, and TA-MIVI in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	Avg. # multiplications per iteration	Avg. elapsed time per iteration (sec): [assignment, update]†	Maximum memory size (GB)
MIVI	1.955×10^{12}	272.4 [263.8, 8.723]	4.134
AF-MIVI	9.411×10^{10}	26.06 [15.64, 10.55]	7.907
CS-MIVI	7.536×10^{10}	346.7 [337.2, 9.568]	8.412
TA-MIVI	1.367×10^{12}	280.6 [267.2, 13.56]	8.030

† The average elapsed time does not exactly match the sum of the assignment and the update time because the algorithm terminated at the end of the assignment step of the last iteration.

Table S16: perf results of MIVI, AF-, CS-, and TA-MIVI in 1M-sized NYT data set with K=10 000. Number of iterations until convergence is 81.

Algorithm	# instructions	# branches	# branch misses (%)	# LLC-loads	# LLC-loads misses (%)
MIVI	1.804×10^{15}	1.716×10^{14}	7.717×10^{10} (0.04)	2.720×10^{13}	2.077×10^{12} (7.64)
AF-MIVI	1.518×10^{14}	2.003×10^{13}	7.343×10^{10} (0.37)	3.090×10^{12}	2.774×10^{11} (8.98)
CS-MIVI	1.540×10^{15}	1.766×10^{14}	1.287×10^{11} (0.07)	2.907×10^{13}	6.011×10^{12} (20.68)
TA-MIVI	1.851×10^{15}	2.393×10^{14}	1.677×10^{12} (0.70)	2.101×10^{13}	3.689×10^{12} (17.56)