共通要素を類似度とするハッシュベース集合間類似検索手法の改善

鈴木　聡† 古賀　久志† GibranFuentes Pineda†† 戸田　貴久†

† 電気通信大学情報理工学研究科　〒 182–8585 東京都調布市調布ケ丘 1–5–1
†† Universidad Nacional Autnoma de Mxico　D.F. 04510 Mexico
E-mail: †{suzuki,koga,takahisa.toda}@sd.is.uec.ac.jp, ††gibranfp@unam.mx

あらまし　Min-Hash は，ハッシュによって Jaccard 係数が高い類似集合を高速に検索するための手法である．Min-Hash では Jaccard 係数が高い集合が同じハッシュ値になるハッシュ関数を用いて，高速な検索を可能にする．しかし，共通要素の多い集合同士を探す類似検索を Min-Hash で実現しようとすると，集合の要素数に大きな差がある場合に，類似度である Jaccard 係数が低下してしまう．このため，ハッシュの衝突が起こらず検索ができないという問題がある．板橋ら [2] はこの問題に対し，要素数の多い集合を分割することで衝突確率の低下を防ぐ手法を提案した．しかし，板橋らの手法を一般化し解析した結果，ハッシュ関数を増やした場合において，衝突確率が共通要素のみに依存するという特性が失われてしまうことがわかった．そこで本研究では，板橋らの手法の問題を改善し，ハッシュ関数の数を増やした場合でも共通要素のみに依存して検索を行える手法を提案する．また，理論解析及び実験によって，提案手法が共通要素類似検索に有効であることを示す．

キーワード　集合，類似検索，ハッシュ，共通要素

1. はじめに

近年，パターン認識などの分野において文章や画像などのデータを集合として扱う機会が増えている．たとえば文章は，それを構成している単語の集合とみなすことができる．このような集合同士を比較することによって類似文章の分類や検索が可能となる．さらに，Web の発展によるデータの急速な増加に伴い，大量のデータの中から類似するものを高速に検索することの必要性が高まっている．Min-Hash [1] ではハッシュを利用することで，高速な類似集合検索を可能にした．ハッシュは，データを高速に検索するためのアルゴリズムの 1 つであり，ハッシュ関数によってデータをハッシュ値とよばれる値に変換することで，ハッシュ値の比較のみでデータを高速に検索することができる．Min-Hash では，2 つの集合 A, B のハッシュ値が一致する確率は Jaccard 係数 $\frac{|A\cap B|}{|A\cup B|}$ に等しくなる．Min-Hash のこの性質を利用し，ハッシュテーブルを用いて高速な検索システムを実現できる．

Min-Hash が Jaccard 係数を類似度とする検索を行う一方，本研究では類似度として共通要素のみに注目したハッシュ検索を目指す．これは，集合 A に対して単純に $|A\cap B|$ が大きい集合 B を検索するものである．本研究ではこの検索を共通要素類似検索と定義する．共通要素類似検索を Min-Hash によって行うことは難しい．これは，Min-Hash が基盤としている Jaccard 係数は，$|A\cap B|$ が大きくとも，$|A\cup B|$ が十分に大きければ小さい値になってしまいハッシュが衝突しなくなるためである．

この問題に対し，板橋ら [2] は，要素数の多い集合を部分集合へ分割することによって要素数の差のよるハッシュ衝突確率の低下を緩和する手法を提案し，共通要素類似検索へ有効であることを理論および実験的に示した．

しかし理論解析を一般化し，複数のハッシュ関数を結合する状況を解析したところ，板橋らの手法はこのような状況で望ましい性能が出ないことが分かった．本研究ではこれに対し，分割した部分集合の全てのハッシュ値を結合するという手法を提案し，理論解析と実験によって提案手法が板橋らの問題点を克服できていることを確認した．

本論文の構成を以下に示す．2 章で Min-Hash を紹介し，3 章で共通要素数類似検索を定義し，問題の有用性について述べる．4 章で共通要素類似検索に対する板橋らの手法を述べる．5 章で板橋らの手法を一般化した場合の理論解析と，そこで分かった問題点について述べる．6 章では提案手法を述べる．7 章で提案手法による実験とその結果について述べる．8 章で本研究のまとめを述べる．

2. Min-Hash

Min-Hash は集合間類似検索のための確率的なハッシュアルゴリズムの一つである．Min-Hash では，集合間の類似度として Jaccard 係数を利用する．以下に，実際に Min-Hash により集合のハッシュ値を求める手順を述べる．まず，並び替え規則 π をランダムに定める．次に，与えられた集合 A について，π に従って各要素を並び替える．並び替えた集合の最小の要素を求め，その集合のハッシュ値とする．以上の一連の操作を mh とし，ある集合 A について求めたハッシュ値を $mh(A)$ とする．これは以下の式 (1) によって表すことができる．

$$mh(A) = \min(\pi(\mathrm{A})) \tag{1}$$

例として，集合の要素種類数を 4 とし，集合 A を $\{1,2,3\}$，並び替え π を $\pi:\{1,2,3,4\}\rightarrow\{4,3,1,2\}$ としたときのハッシュ値を求める．集合 A を π に従って並び替えると，$\pi(A)=\{3,4,2\}$ となる．これは，集合 A の各要素について，並び替え π におけるその要素のインデックスが，並び替え後の要素となっている．

例えば，A の 1 番目の要素 1 は，π では 3 番目に並び替えられているので，$\pi(A)$ では 3 へ変換される．このとき，並び替えた後の集合で最小となる要素が，A のハッシュ値 $mh(A)$ となる．この場合，$mh(A)$ は $\pi(A)$ の最小値である 2 である．Min-Hash の性質として，2 つの集合 A, B のハッシュ値 $mh(A), mh(B)$ が一致する確率は A, B の Jaccard 係数に等しくなることが示されている．

$$P[mh(A) = mh(B)] = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

2.1 ハッシュテーブルを用いた類似検索システム

Min-Hash の性質を利用し，ハッシュテーブルによる高速な類似集合検索システムを構築することができる．図 1 にシステムの概要を示す．

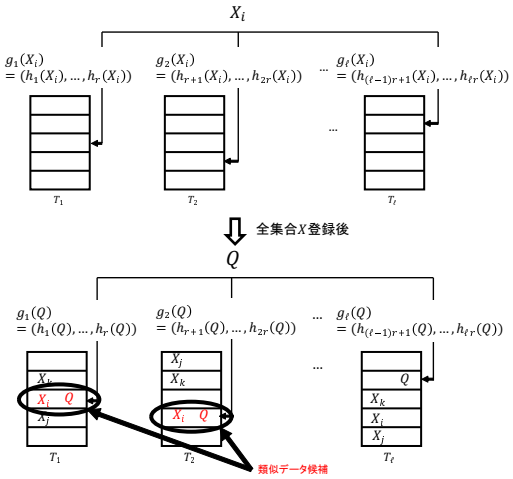

図 1　Min-Hash による類似検索

このシステムにおける類似集合の検索の手順を以下に述べる．

ハッシュテーブルへの集合の登録

（1） まず，ランダムに選択する π の異なる r 種類のハッシュ関数 $(mh_1, mh_2, \cdots, mh_r)$ を用意し，r 個のハッシュ値を結合して出力する関数を g とする．すなわち $g(A) = (mh_1(A), mh_2(A), \cdots, mh_r(A))$ となる．

（2） 関数 g を $g_1, g_2, \cdots, g_l$ の l 種類用意し，それぞれの関数に対応するハッシュテーブル $T_1, T_2, \cdots, T_l$ を作成する．

（3） ハッシュテーブルを作成したら，データベース内の集合 X_i について，それぞれの関数 $g_j (1 \leqq j \leqq l)$ でハッシュ値 $g_j(X_i)$ を求める．

（4） 求めたハッシュ値をインデックスとして，対応するハッシュテーブル T_j のバケツ $b(g_j(x_i))$ へ集合を格納する．

（5） データベース内のすべての集合に対して，以上の手順によってハッシュテーブルへの登録を行う．

クエリ集合 Q に対する類似検索

（1） クエリ集合 Q についてハッシュ値 $g_j(Q)$ を計算し，構築したそれぞれのハッシュテーブル T_j のバケツ $b(g_j(Q))$ に格納されている集合群を Q の類似候補とする．

（2） 最後に，クエリと類似候補の Jaccard 係数を厳密計算することで，類似集合を決定する．

3. 共通要素類似検索

本研究の目的は，クエリ集合 A に対して，共有する要素を最も多く持つような集合 B を発見できるような類似検索を実現することである．具体的には，クエリ A に対して $\frac{|A \cap B|}{|A|}$ の値が大きくなるような集合 B を発見する．ここで，分母の $|A|$ はクエリのみに依存するため，結局 $|A \cap B|$ の値が大きくなる集合 B を見つければ良いことになる．本論文では，このような集合を検索することを共通要素類似検索と定義している．

Min-Hash で利用される Jaccard 係数は，$\frac{|A \cap B|}{|A|}$ に比例している．しかし，$\frac{|A \cap B|}{|A|}$ が大きい場合でも，A と B の集合の要素数の差がある程度大きい場合は，分母である $|A \cup B|$ の値が増加し，最終的な類似度の値は低くなり，Min-Hash を用いた検索は困難になる．例えば，B の要素数が A の α 倍 $(\alpha > 1)$ の場合，$|A \cap B| \leqq |A|$ かつ $|A \cup B| \geqq \alpha|A|$ となるため，類似度 $sim(A, B) \leqq \frac{1}{\alpha}$ となる．これは，類似度が共通要素 $|A \cap B|$ によらず，要素数の差によって低くなってしまうことを示している．以上の理由から，Min-Hash を共通要素類似検索へと適用することは適切でないと言える．

共通要素類似検索が適用できる実際のアプリケーションとして，文章の剽窃検知が挙げられる．これは，ある文章の一部分が他の文章と酷似している場合に，それを発見するという問題である．この問題を Min-Hash を用いて実現することを考える．例として，ある文章 D_1 を完全に含んでいる別の文章 D_2 があるとする．この場合 D_2 は D_1 を剽窃している可能性が高いため，検知されるべきである．しかし，D_2 の文章の長さが D_1 に比べて十分に大きい場合，先述した理由により類似度の低下が起こり Min-Hash の衝突は起こらない可能性がある．実際，検知される文章の長さが一定であるとは限らないため，Min-Hash でこのアプリケーションを実現するのは現実的ではない．

4. 板橋らの手法

集合 $\boldsymbol{B = \{x_1, x_2, \ldots, x_{14}\}}$　ハッシュ関数$\boldsymbol{\{mh_1, mh_2\}}$

分割(閾値4)
&
マージ

$$\boldsymbol{B^1 = \{x_1, x_3, x_6\ x_{14}\}}$$
$$\boldsymbol{B^2 = \{x_2, x_5, x_{12}, x_{11}\}}$$
$$\boldsymbol{B^3 = \{x_4, x_{10}, x_8, x_{13}, x_7, x_9\}}$$

ハッシュ計算
・結合

$$\left.\begin{aligned} \boldsymbol{g(B^1) = \{mh_1(B^1), mh_2(B^1)\}} \\ \boldsymbol{g(B^2) = \{mh_1(B^2), mh_2(B^2)\}} \\ \boldsymbol{g(B^3) = \{mh_1(B^3), mh_2(B^3)\}} \end{aligned}\right\} \boldsymbol{mh(B)}$$

図 2　部分集合への分割

前章で述べた Min-Hash における衝突確率の低下は，集合間の要素数の差が大きいことが原因である．そこで板橋らは，要素数の多い集合をいくつかの部分集合へ分割し，要素数の差を無くすことで衝突確率が低下することを緩和する手法を提案した．以下にこの手法の手順を述べる．

ハッシュテーブルへの集合の登録

STEP1： まず，分割のための閾値 T を定める．テーブルへ登録する集合 B の要素数が T 以上のとき，要素数 T の部分集合へとランダムに分割する．この際 $|B|$ が T で割り切れず，$|B| \bmod T$ の部分集合が発生した場合，これを他の部分集合のどれかに結合する．この操作により，集合 B は $\lfloor \frac{|B|}{T} \rfloor$ 個の部分集合 $B^1, B^2, \cdots, B^{\lfloor \frac{|B|}{T} \rfloor}$ へと分割される．

STEP2： 次に，分割したそれぞれ部分集合 $B^i (1 \leqq i \leqq \frac{|B|}{T})$ に対して r 個のハッシュ関数 $mh_1(), \cdots, mh_r()$ を適用し，ハッシュ値 $mh_1(B^i), mh_2(B^i), \cdots, mh_r(B^i)$ を得る．

STEP3： STEP2 で得られた r 個のハッシュ値を結合し，これを $g(B^i) = (mh_1(B^i), mh_2(B^i), \cdots, mh_r(B^i))$ とする．

STEP4： STEP3 で結合されたそれぞれの部分集合のハッシュ値の集合 $g(B^1), \cdots, g(B^{\frac{|B|}{T}})$ を B のハッシュ値とする．

STEP5： 全ての部分集合 B^i について，対応するハッシュバケツ $b(g(B^i))$ へと格納する．

クエリ集合 A に対する類似検索

STEP1： クエリ集合 A について B と同様に STEP1~STEP4 の手順で $g(A^1), \cdots, g(A^{\frac{|A|}{T}})$ を求める．

STEP2： A のそれぞれの部分集合に対応するハッシュバケツ $b(g(A^i))$ に B の部分集合のどれかが格納されていた場合，集合 B をクエリ A の類似候補とする．なお，クエリ A の検索を行う場合，クエリを分割するかどうかはオプションとする．これは，適用するアプリケーションによってはクエリの要素数が小さいことがあらかじめわかっている場合，クエリを分割する必要がないためである．クエリを分割しない場合は単純にクエリ A について $g(A)$ を求め，対応するハッシュバケツ $b(g(A))$ に B の部分集合のどれかが格納されていた場合に集合 B をクエリ A の類似候補とする．

図 2 に板橋らの手法におけるハッシュ値の計算の概要図を示す．要素数 14 個の集合 B は閾値 4 により，3 つの部分集合 B^1, B^2, B^3 にランダムに分割される．分割された 3 つの部分集合それぞれに対し，2 つのハッシュ関数 mh_1, mh_2 によりハッシュ値を計算し，これを結合し $g(B^1), g(B^2), g(B^3)$ を得る．結局，集合 B のハッシュ値 $mh(B)$ は $\{g(B^1), g(B^2), g(B^3)\}$ の 3 つになる．この手法は，従来の Min-Hash の僅かな変更によって実現することができ，実装が容易である．

4.1 理論解析

本節では，板橋らの手法が共通要素類似検索において理論的に望ましい性質を持っていることを述べる．

ここで，それぞれの部分集合へ適用するハッシュ関数の数 $r = 1$ とする．2 つの集合 A,B を考え，これらの要素数を分割閾値 T の定数倍とする．また，集合 B の要素数 $|B|$ は集合 A の要素数 $|A|$ の α 倍 $(\alpha \geqq 1)$ とする．このとき，2 つの集合の要素数は，自然数の定数 c を用いて $|A| = cT, |B| = \alpha cT$ とかける．2 つの集合の共通要素 $|A \cap B|$ を $k (0 \leqq k \leqq cT)$ とすると，Min-Hash におけるハッシュの衝突確率 $P[mh(A) = mh(B)]$ は以下の式によって表すことができる．

$$p[mh(A) = mh(B)] = \frac{|A \cap B|}{|A \cup B|} = \frac{k}{(\alpha + 1)cT - k} \qquad (3)$$

式 (3) は α の値が増加するにつれて 0 へと収束する．このことは B の要素数が A に対して十分に大きい場合に Min-Hash でのハッシュ衝突が起こらなくなってしまうことを示している．

次に，集合を分割する場合について述べる．分割閾値を T とすると，集合 A,B はそれぞれ部分集合 $A^1, A^2, \cdots, A^c$ および $B^1, B^2, \cdots, B^{\alpha c}$ へと分割される．分割によって，集合が持つ共通要素 $|A \cap B|$ は各部分集合へ偏りなく配分されると仮定すると A と B の部分集合 $A^i (1 \leqq i \leqq c)$ と $B^j (1 \leqq j \leqq \alpha c)$ の持つ共通要素の期待値は $\frac{k}{\alpha c^2}$ となる．したがって，分割された A，B の部分集合同士が衝突する確率 $P[mh(A^i) = mh(B^j)]$ は以下の式によって表される．

$$P[mh(A^i) = mh(B^j)] = \frac{\frac{k}{\alpha c^2}}{T + T - \frac{k}{\alpha c^2}} = \frac{k}{2\alpha c^2 T - k} \qquad (4)$$

式 (4) は，Min-Hash の場合と同様に $\alpha \to \infty$ で 0 へ収束してしまうが，板橋らの手法において，集合 A，B が最終的に衝突する確率 $P[imh(A) = imh(B)]$ は A，B それぞれの部分集合間のうち少なくともどれか 1 つが衝突する確率となる．これは，すべての部分集合間でハッシュ値が合致しない確率を 1 から引いたものであり，次に示す式 (5) によって表すことができる．

$$P[imh(A) = imh(B)] = 1 - \left(1 - \frac{k}{2\alpha c^2 T - k}\right)^{\alpha c^2} \qquad (5)$$

式 (5) は $\alpha \to \infty$ で単調減少するものの，0 へ収束することなく一定の値へと収束する．このときの収束値は以下の式で求められる．

$$\begin{aligned} &\lim_{\alpha \to \infty} 1 - \left(1 - \frac{k}{2\alpha c^2 T - k}\right)^{\alpha c^2} \\ &= \lim_{\alpha \to \infty} 1 - \left(\frac{1}{e}\right)^{\frac{\alpha c^2}{\frac{2T}{k} \alpha c^2 - 1}} \\ &= 1 - \left(\frac{1}{e}\right)^{\frac{k}{2T}} = 1 - e^{-\frac{k}{2T}} \end{aligned} \qquad (6)$$

式 (6) から，収束する値は α によらず，共通要素数 k のみに依存していることがわかる．これは，板橋らの手法が集合の要素数に関係なく共通要素の数が多い程ハッシュが衝突することを示しており，共通要素類似検索に適した性質を持っていると言える．

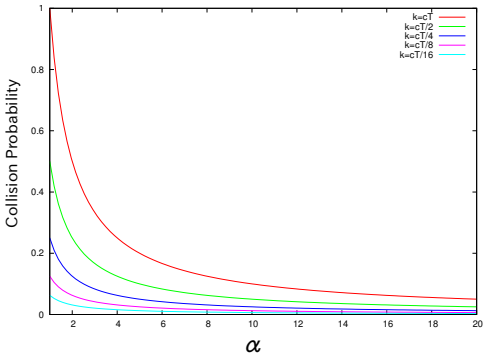

図 3　Min-Hash における衝突確率

図 3，4 に，$c = 1$ とし，k と α を変えシミュレーションを行った場合の Min-Hash および板橋らの手法における衝突確率

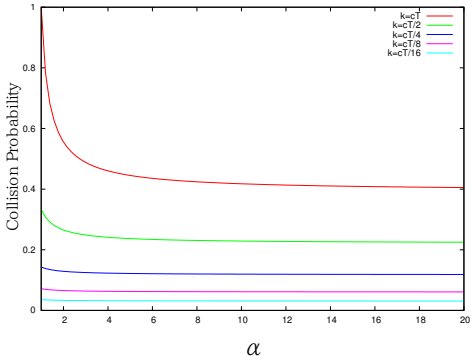

図 4　板橋らの手法における衝突確率

の遷移を示す．図 3 から，Min-Hash におけるハッシュの衝突確率は共通要素数 k に依らず，$\alpha \to \infty$ で 0 へと収束することが読み取れる．これは Min-Hash が共通要素類似検索について適切でないことを示している．これに対し，図 4 では，$\alpha \to \infty$ において，衝突確率は 0 でない一定の値へと収束していることが分かる．さらにその収束値は共通要素数 k の値に応じて異なっており，板橋らの手法が共通要素類似検索に適していることを示している．

5. 板橋らの手法の問題点

先行研究では，$r = 1$ という条件で，板橋らの手法は共通要素類似検索に適した性質を持っていることを示した．しかし，2 章で述べたように，Min-Hash を用いた実際の検索システムでは，複数のハッシュ関数を結合することで 1 つのハッシュ値を計算することが多い．これは，真に近い集合同士のみが衝突するようにすることで検索における擬陽性を減らし，検索精度を向上するためである．

このことを考慮し，本研究では板橋らの手法を $r \geqq 1$ の一般化された条件下で理論解析を行った．集合 A,B および分割後の部分集合をそれぞれ A^i, B^j とおく．このとき，部分集合 A^i, B^j 同士が衝突する条件は，$(mh_1(A^i) = mh_1(B^j)) \wedge (mh_2(A^i) = mh_2(B^j)) \wedge \cdots \wedge (mh_r(A^i) = mh_r(B^j))$ となる．すなわち，分割されたある部分集合同士について，ハッシュが衝突するためには，r 個すべてのハッシュ値同士が合致する必要がある．これは理論的には以下の式によって表すことができる．

$$1 - \left(1 - \left(\frac{k}{2\alpha c^2 T - k}\right)^r\right)^{\alpha c^2} \tag{7}$$

式 (7) を $\alpha \to \infty$ とした極限を考える．

$$\begin{aligned}
&\lim_{\alpha\to\infty} 1 - \left(1 - \left(\frac{k}{2\alpha c^2 T - k}\right)^r\right)^{\alpha c^2} \\
&= \lim_{\alpha\to\infty} 1 - \left(1 - \frac{1}{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}\right)^{\alpha c^2} \\
&= \lim_{\alpha\to\infty} 1 - \left(\left(1 - \frac{1}{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}\right)^{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}\right)^{\frac{\alpha c^2}{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}} \\
&= \lim_{\alpha\to\infty} 1 - \left(\frac{1}{e}\right)^{\frac{\alpha c^2}{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}}
\end{aligned} \tag{8}$$

式 (8) の $\left(\frac{2T}{k}\alpha c^2 - 1\right)^r$ の部分は，$r \geqq 2$ において $\alpha \to \infty$ で 0 へと収束し，$\left(\frac{1}{e}\right)^{\frac{\alpha c^2}{\left(\frac{2T}{k}\alpha c^2 - 1\right)^r}}$ は 1 へと近づくため，結局式 (8) は $\alpha \to \infty$ で 0 へと収束してしまう．これは，$r \geqq 2$ の場合に板橋らの手法の利点が失われることを示している．

6. 提 案 手 法

本研究では，板橋らの手法における衝突確率低下を緩和するための手法を提案する．提案手法は，複数のハッシュ関数において分割した部分集合同士の全てのペアでハッシュ値の結合を行う．

例として，集合 B が n 個の部分集合へと分割され，r 個のハッシュ値を結合する場合を考える．従来手法では，分割されたそれぞれの部分集合毎に複数のハッシュ値を求めそれらを 1 つに結合する．このとき，ある部分集合 B^i に対する 1 つ目のハッシュ値 $mh_1(B^i)$ に結合されるハッシュ値は，その部分集合に対するハッシュ値 $mh_2(B_i), \cdots, mh_r(B_i)$ に限られる．そのため，最終的に得られるハッシュ値の数は集合の分割数 n に等しくなる．

これに対し提案手法では，ハッシュ関数 $mh_p(1 \leqq p \leqq r)$ に対して，部分集合のハッシュ値の集合 $S_p = \{mh_p(B^1), mh_p(B^2), \cdots, mh_p(B^n)\}$ を生成する．そして，$S_1, S_2, \cdots, S_r$ からハッシュ値を 1 つずつ選んで組み合わせる．このとき，組み合わせの場合の数は n^r となるが，それに対応する n^r 個のハッシュ値を生成する．

図 5，6 に，分割数 $n = 3$，ハッシュ関数の数 $r = 2$ における従来手法および提案手法におけるハッシュ値生成の具体例を示す．いずれの手法においても，集合 B はまず 3 つの部

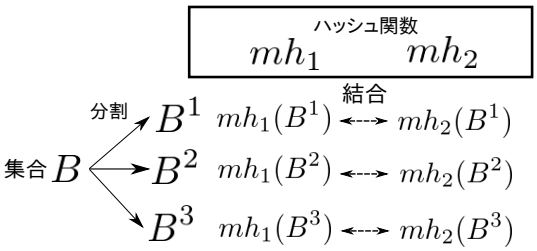

図 5　従来手法におけるハッシュ値の結合

分集合 B^1, B^2, B^3 に分割される．その後，分割された部分集合について 2 つのハッシュ関数 mh_1, mh_2 によってハッシュ値を計算し，結合を行う．この時，従来手法では図 5 に示すように，$mh_1(B^1)$ については $mh_2(B^1)$，$mh_1(B^2)$ については $mh_2(B^2)$ というようにその部分集合についてのハッシュ値のみが結合される．そのため，生成されるハッシュ値は分割数と同じ 3 個となる．

これに対し従来手法では図 6 に示すように，$mh_1(B^1)$ について $mh_2(B^1)$ の他にも，異なる部分集合のハッシュ値 $mh_2(B^2), mh_2(B^3)$ も結合されることになり，結局，生成されるハッシュ値の数は $3^2 = 9$ 個となる．ここで，提案手法におけるハッシュ衝突の条件は，それぞれのハッシュ関数において分割された部分集合のハッシュ値の少なくとも 1 つが衝突す

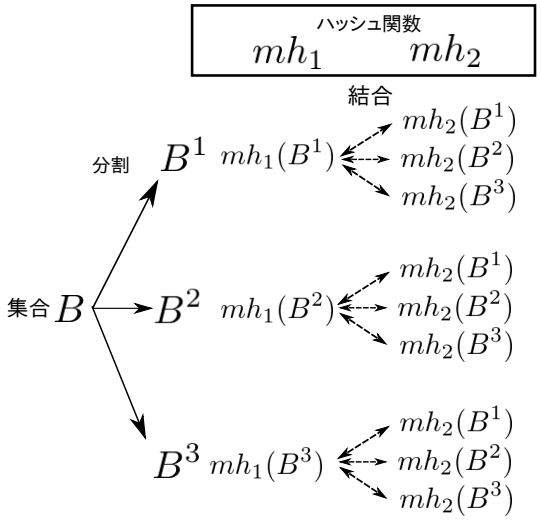

図 6　提案手法におけるハッシュ値の結合 ($r = 2$)

る確率である．これは以下の式によって表すことができる．

$$\left(1 - \left(1 - \frac{k}{2\alpha c^2 T - k}\right)^{\alpha c^2}\right)^r \tag{9}$$

式 (9) は板橋らの手法の衝突確率を単純に冪乗したものになるため，板橋らの手法における性質を失うことなくハッシュ関数の結合が行うことができると言える．

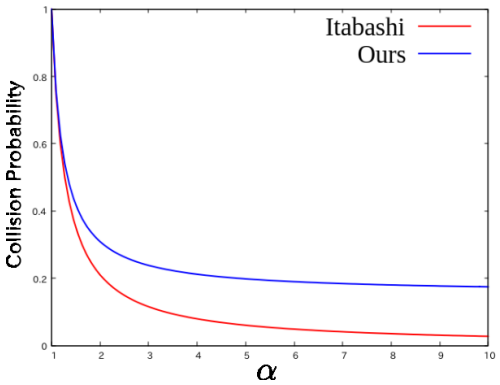

図 7　$r = 2$ における各手法の衝突確率の遷移

図 7 に，ハッシュ関数の数 $r = 2$，共通要素数 $k = cT$ で $c = 1$ としたときの板橋らの手法 (式 (7)) および提案手法 (式 (9)) のハッシュ衝突確率の遷移を示す．図から，板橋らの手法は $r = 2$ で衝突確率が 0 へと収束してしまうのに対し，提案手法は一定の値へ収束していることがわかる．このことから，提案手法はハッシュを増やした場合においてもハッシュ衝突確率の低下を緩和し，共通要素類似検索を行えると言える．

6.1　提案手法の実装

提案手法において生成されるハッシュ関数の数は，n^r となるが，これは r に伴い指数的に増加するため，実用においては計算量上好ましくない．そこで本研究では，$r \geqq 3$ の場合は $r - 1$ までのハッシュ値は従来通り結合し，r 番目のハッシュ値の結合の際にだけすべての組み合わせを行うことにした．これにより，r の数に依らず生成されるハッシュ値の数は n^2 に抑えられる．図 8 に，結合するハッシュの数が 3 の場合における提案手法でのハッシュ結合の様子を示す．この場合，mh_1 と mh_2 の結合は従来通り 1 つの部分集合だけで行い，最後の mh_3 を結合する際にのみ，すべての部分集合のハッシュ値を組み合わせる．この実装は，r が大きくなると衝突確率は低下してしまうため，必ずしも望ましいとは言えない．しかし，Min-Hash の実際のアプリケーションでは，r =2 や 3 などハッシュ関数の数は小さな値に設定されることが多い．例えば [11] や [9] の研究は，画像中の複製物を検知する問題で $r = 2$ としている．従って提案手法はこのようなアプリケーションに対して実用性を失うことは無いと言える．

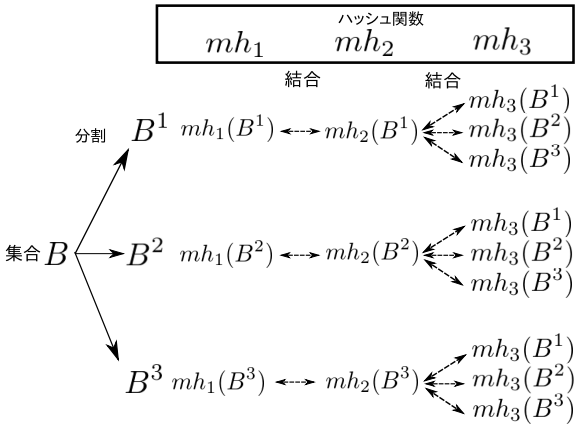

図 8　提案手法におけるハッシュ値の結合 (r=3)

7.　関 連 研 究

このような Min-Hash を用いたシステムは様々なアプリケーションへと応用されている．[3] では，画像から局所特徴を抽出し，画像を局所特徴から構成される集合とみなすことで，Min-Hash を用いた高速な複製画像検索を実現している．また，[4] では，複数の画像に共通して現れる局所特徴の集合をオブジェクトとみなすことで，高速な画像からのオブジェクト発見手法を提案した．

Min-Hash のアルゴリズム自体の改良に関する研究も盛んに行われてきた．[5] では 1 回の入れ替えについて，最小値と最大値の 2 つをハッシュ値とすることにより，ハッシュテーブルの数を半分に減らした．b-bit minwise hashing [6] では，ハッシュ値の下位 b ビットのみを使用することによって空間計算量を大幅に削減した．One Permutation Hashing [7] では集合の要素を入れ替えたものを複数の bin へ分割し，それぞれからハッシュ値を生成する事で，1 つの入れ替え規則で複数のハッシュ値を生成する手法を提案した．[8] では，Min-Hash での類似画像検索において，局所特徴の位置的な情報を利用することにより，検索精度を向上させた．[9] は，画像を小領域へ分割し，それぞれの領域毎にハッシュ値を生成することで，画像の局所的な類似性を考慮した検索を実現した．[10] はこの手法を改良し，領域分割を木構造にすることで，検索性能を更に向上させた．[11] では，画像中の 1 つの特徴と，その周辺にある特徴をまとめて Bundle とし，ハッシュ値を計算することで画像中に

あるロゴを検出する手法を提案した．

8. 実験評価

8.1 人工データによる実験

提案手法の妥当性を評価するため，人工データを使用した検索実験を行った．以下に実験の手順を述べる．

（1） まず，要素種類数 20000 の条件で要素をランダムに選択し，要素数 200 のクエリ集合 $(Q_1, Q_2, \cdots, Q_{100})$ を作る．

（2） 次に，それぞれのクエリ集合に対し，共通要素を多く含む類似集合を 50 個ずつ作成し，データベースとする．このとき，類似集合に含ませる共通要素数は 32 から 130 までの 2 刻み $\{32, 34, \cdots, 128, 130\}$ とし，集合の要素数は $\{400, 600, 800, 1000, 1200, 1400, 1600\}$ の中からランダムに決定し，クエリとの要素数の比が集合毎に異なるようにする．

実験は作成したクエリとデータベースを用いて，ハッシュテーブルにより検索を行う．実験のパラメータとして，ハッシュテーブルの数を 500，分割閾値 T を 200 とする．比較対象は Min-Hash 及び板橋ら [2] の手法とする．

8.1.1 スピアマンの順位相関係数を用いた評価

実験の評価には，各クエリの類似集合におけるハッシュ衝突回数と共通要素数間のスピアマンの順位相関係数を用いた．スピアマンの順位相関係数 ρ は，順位が与えられている変数 X，Y について，対応する X と Y の順位の差を D，値のペアの数を N，同順位の数をそれぞれ t_i，t_j とおくと，式 (10) で表される．

$$\rho = \frac{T_X + T_Y - \sum D^2}{2\sqrt{T_X T_Y}}$$
$$T_X = \frac{N^3 + N - \sum(t_i^3 - t_i)}{12} \qquad (10)$$
$$T_Y = \frac{N^3 + N - \sum(t_j^3 - t_j)}{12}$$

ここでは，クエリとデータベースについて，共通要素の多い集合についてハッシュの衝突回数が多い傾向がある場合に順位相関係数は高くなる．したがって，この値が大きいハッシュ関数は共通要素類似検索に適した性能を持っているといえる．評価にあたって，100 個のクエリそれぞれの順位相関係数を求め，平均化することによって，各手法の評価値とした．実験の結果を表 1 に示す．

表 1 実験結果: ハッシュテーブル数 l =500

ハッシュ関数の数	Min-Hash	板橋 [2]	提案手法
1	0.688	0.977	
2	0.619	0.751	0.966
3	0.442	0.446	0.710

実験結果から，提案手法はハッシュ関数の数が 1 から 2 に増えた場合でも相関係数の値がほぼ変わらないことがわかる．またハッシュ関数の数が 3 に増えた場合は，相関係数は若干低下してまうものの一般的に高い相関が認められる 0.7 よりも高い値を示している．このことから，提案手法はハッシュ結合を行う場合でも共通要素類似検索に適した性質があると言える．

8.2 実データによる実験

次に，実データを用いた実験により，提案手法が実際に共通要素類似検索に適しているかを評価した．実データを用いた実験では，データセットとして MNIST [12] を用いた．MNIST は 28x28=784 ピクセルから構成される手書き数字の画像データセットである．データセットには，train データとして 60000 枚，test データとして 10000 枚の合計 70000 枚の画像が含まれている．画像の各ピクセルは，白黒で 0∼255 の値を取るが，Min-Hash や提案手法で扱えるデータは集合であるため，次の手順で画像を集合へと変換した．

- まず画像の各ピクセルを，128 を閾値として二値化し，0 または 1 の二値ベクトルとする．
- 次に値が 1 であるピクセルのインデックスの集合を求め，これを画像に対する集合表現とする．

以上の操作によって，それぞれの画像は白の画素の位置を要素とする集合となる．集合の要素数は，画像中の白の画素数となる．図 9，10 に，train，test それぞれに含まれるデータを集合に変換した際の要素数のヒストグラムを示す．

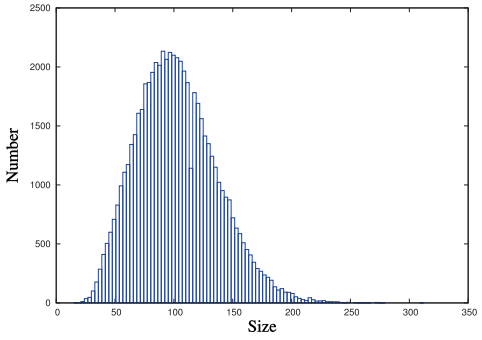

図 9 MNIST train に含まれる集合の要素数のヒストグラム

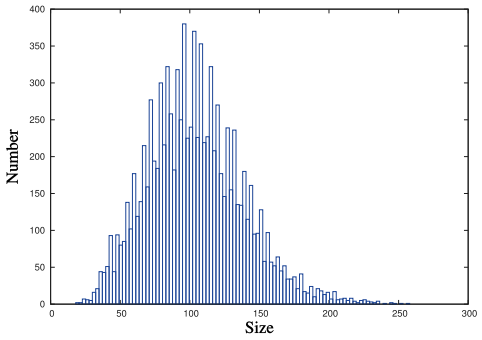

図 10 MNIST test に含まれる集合の要素数のヒストグラム

図 9，10 から，train および test に含まれる集合の要素数の分布はおおよそ同じあることが分かる．つまり，8.1 章の実験とは異なり，検索におけるクエリとデータベース間での集合の要素数に大きな差が無いと言える．

8.2.1 実験手順

実験で用いる検索システムのパラメータとして，ハッシュテーブルの個数 $l = 100$，結合するハッシュ関数の数 $r = 1, 2$ とし，分割の閾値 $T = 50$ に設定した．実験ではまずハッシュテーブルに 60000 枚の train データを登録し，これに対し test データの 10000 枚をクエリとして類似検索を行った．

8.2.2 実験評価

実験の評価方法について述べる．本実験では，データ間のハッシュの衝突確率と共通要素数の間に相関があるかを評価する．そこで，以下の手順で評価を行った．

（1） まず，各クエリデータ Q について，train データから共通要素を多く含む上位 100 個の集合を求め，これを Q に対する正解データとする．

（2） Q をクエリ集合として検索実験を行い，全 train データを Q と同じバケツに入った回数の降順に並び替える．

（3） 並び替えたデータに対して上位から $i (i \geqq 1)$ 番目の集合について，それが正解であるかを判定する．

（4） 判定したら，上位 i 個に含まれる正解数 $a(Q)$ を求め，式 (11) から Precision および Recall を計算する．

$$Precision = \frac{a(Q)}{i}, Recall = \frac{a(Q)}{100} \tag{11}$$

（5） i を 1 増やし，同様に Precision，Recall を計算する．

（6） 1 つの Recall に対して複数の Precision をとる場合は，Precision の平均値を計算し，Recall に対してただ 1 つの Precision を求める．

（7） 最後に全てのクエリについてそれぞれの Recall に対する Precision の値を平均化し，Precision-Recall カーブを得る．

今回の実験によって作成した Precision-Recall カーブを以下に示す．図 11 は $r = 1$，図 12 は $r = 2$ の条件での結果を示している．

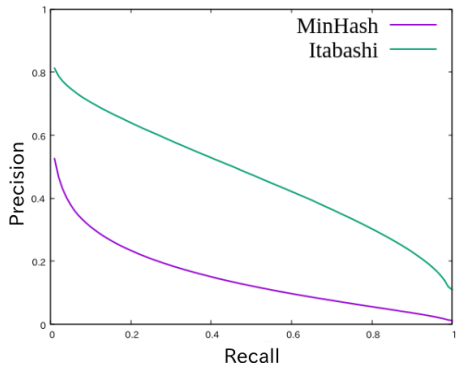

図 11　Precision-Recall カーブ ハッシュ関数 1

$r = 1$ では，ハッシュ関数の結合は行われないため，通常の Min-Hash と板橋らの手法の比較となる．図 11 より，板橋らの手法は Min-Hash と比較して高い Precision の値をとっており，共通要素の多い集合を検索できていることがわかる．

次に $r = 2$ の場合を見ると，板橋らの手法は $r = 1$ の場合

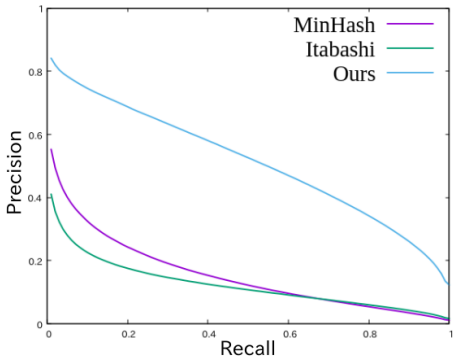

図 12　Precision-Recall カーブ ハッシュ関数 2

と比較して Precision の値が著しく低下していることが分かる．これに対して、提案手法は $r = 1$ の場合での板橋らの手法とほぼ同程度の Precision を示している．この結果は，ハッシュ関数の数を増やした場合に

（1） 板橋らの手法における，ハッシュ衝突確率が共通要素に依存するという性質が失われること．

（2） 提案手法が，上記の性質を失うことなく共通要素類似検索を行えること．

を示していると言える．

また，$r = 2$ における板橋らの手法に注目すると，Min-Hash よりも低い Precision になっている．これは，集合の要素数の分布が原因になっていると考えられる．図 9，10 を見ると，クエリとデータベース間で集合の要素数の差が少なく，MNIST では，要素数の倍率である α が小さな値になる．集合の要素数に大きな差がなければ，Jaccard 係数は共通要素数に依存しやすくなる．そのため，分割するよりも良い性能を出せているものと考えられる．また，本実験においてクエリに対して正解データは共通要素数のみで決定したが，今回のような手書き数字データセットでは，クエリと同一のラベルを持つデータが正解になる可能性が高い．その場合，クエリと正解データ間の要素数の差は小さくなり，Min-Hash にとって有利な条件になると考えられる．提案手法はこのような状況においても，Min-Hash よりも良い値を示しており，共通要素類似検索において，汎用性のある手法であると言える．

9. まとめ

本研究では，クエリ集合に対して共通要素の多い集合を検索する共通要素類似検索を論じた．集合間類似検索に対するハッシュに Min-Hash があるが，Min-Hash が類似度としている Jaccard 係数は，集合間の要素数に差がある場合，共通要素が多い集合間であっても類似度が低下し，ハッシュの衝突確率が低下してしまうという問題がある．

板橋ら [2] は，この問題に対して，集合をいくつかの部分集合に分割することで要素数の差を減らし，衝突確率の低下を抑制する手法を提案した．板橋らの手法は，理論的，実験的にも共通要素類似検索に適していることが示されていたが，複数のハッシュ値を結合することを考慮されていなかった．

本研究では，板橋らの手法について，結合するハッシュ関数の数を r とおいて一般化し，理論解析を行った．理論解析の結果，r が 2 以上の状況では，板橋らの手法におけるハッシュ衝突確率は集合間の要素数の差に伴って 0 へと近づいてしまい，共通要素類似検索に対する利点が失われてしまうことが分かった．この問題を解決するため，本研究では分割された部分集合について複数のハッシュ値を結合する場合全ての部分集合の組み合わせについてハッシュ値を結合する手法を提案した．集合の分割数を n，結合するハッシュ関数の数を r とした場合，板橋らの手法では，それぞれの部分集合毎にハッシュ値を結合するため，n のハッシュ値が生成されるのに対し，提案手法では分割された部分集合すべての組み合わせで結合を行うため，n^r 個のハッシュ値が生成されることになる．これは，r が増えれば，生成されるハッシュ値の数は指数的に増加してしまう．そのため，実装では $r \geqq 2$ の場合は $r-1$ のハッシュ値までは従来通り結合し，最後の組み合わせのみを結合するようにした．理論解析によって，提案手法は $r = 2$ の場合においても，ハッシュ衝突確率が 0 へ近づくことなく，共通要素の数に依存した一定の値へ収束し，共通要素類似検索に適した性質を失わないことが分かった．

人工データを用いた実験では，ハッシュの衝突回数と，共通要素数の間でのスピアマンの順位相関係数を求め，評価を行い，従来手法は r が 1 から 2 に増やすと相関係数が低下するのに対し，提案手法では高い値を保つことが確認できた．実データとして MNIST を用いた実験では，　作成した Precision-Recall カーブから，提案手法は従来手法よりも同一 Recall で高い Precision の値を取ることが確認できた．MNIST は人工データと比較して，集合間の要素数に差があまり無いため，Min-Hash が有利になる状況であるが，提案手法はこのような状況でも Min-Hash より高い精度で共通要素類似検索が実現できることを示した．

謝　　辞

本研究は科研費基盤研究 (C)15K00148 の助成を受けたものである．

文　　献

[1] A. Z. Broder, “On the resemblance and containment of documents” in the Compression and Complexity of Sequences, pages 21-29, 1997.

[2] 板橋 大樹, 古賀 久志,Gibran Fuentes Pineda GIBRAN, 戸田 貴久, “共通要素数を重視したハッシュベース集合間類似検索” DEIM Forum 2016 D7-3, 2016

[3] O. Chum, J. Philbin and A. Zisserman, “Near Duplicate Image Detection: Min-Hash and Tf-Idf Weighting”, in Proc. of BMVC, 2008.

[4] G. F. PINEDA,H. Koga and T. WATANABE, “Scalable Object Discovery: A Hash-based Approach to Clustering Co-occurring Visual Words”, IEICE Trans. on Information and Systems,Vol.E94-D(10),pp.2024–2035, 2011.

[5] J. Ji, J. Li, S. Yan, Q. Tian and B. Zhang, “Min-Max Hash for Jaccard Similarity”, in Proc. IEEE ICDM 2013, pp.301–309, 2013.

[6] P. Li and A. C. König, “b-Bit minwise hashing”, in Proc. WWW 2010, pp.671–680, 2010.

[7] P. Li, A. B. Owen and C. H. Zhang, “One permutation hashing”, in Proc. NIPS’12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2 pp.3113–3121, 2012.

[8] O. Chum, M. Perd’och and J. Matas, “Geometric min-Hashing: Finding a (thick) needle in a haystack”, in CVPR,2009

[9] D. C. Lee, Q. Ke and M. Isard, “Partition Min-Hash for Partial Duplicate Image Discovery”, in Proc. ECCV 2010: Computer Vision, pp.648-662

[10] Q. Zhang, H. Fu and G. Qiu, “Tree partition voting min-hash for partial duplicate image discovery”, in Proc.IEEE international conference on multimedia and expo (ICME), pp 1-6

[11] S. Romberg and R. Lienhart, “Bundle min-hashing for logo recognition”, in Proc. ICMR, pp.113-120, 2013

[12] http://yann.lecun.com/exdb/mnist/