Vol.2020-MPS-131 No.11
2020/12/17

ストリーム環境でのテキスト集合に対する類似検索

久保 幸平[1,a)]　古賀 久志[1,b)]

概要：近年，ソーシャルネットワークや IoT の発展に伴い，データストリームを対象とした類似検索の重要性が増している．その中で最近，データストリームの直近の要素群を動的に変化する集合とみなし，類似データストリームを集合間類似検索に帰着して検索するアプローチが出現した．既存研究では集合の要素がアルファベットであるのに対し，本研究では集合の要素をテキストに拡張しテキストストリームを対象とする類似検索を考える．とくに本研究では，クエリ (テキスト集合) との類似度が閾値 ϵ 以上となるテキストストリームをデータベースから探すレンジ探索 (Continuous similarity search for Text Streams, CTS 問題) を取り扱う．CTS 問題は SNS から類似ユーザを探すシナリオをモデル化している．本論文では CTS 問題に対する枝刈りベースの高速アルゴリズムを提案する．そして，人工データを用いた実験評価により提案アルゴリズムが，単純なベースラインアルゴリズムよりも高速に動作することを示す．なお，クエリとデータベースの両者が動的に変化する問題設定を取り扱った点も本研究の新規性である．

キーワード：データストリーム，類似検索，アルゴリズム

1. はじめに

近年, ソーシャルネットワークや IoT (Internet of Things) の発展に伴い，ストリームデータ解析の重要性が高まっている．その中でストリームデータを対象とする類似検索は，リアルタイムでの情報推薦や異常検出の基盤となる技術として注目されている．ストリームデータでは時間経過に伴って新しいデータが追加される．これまで，ストリームデータを対象とした類似検索では，1 つのデータストリームをデータベースと見なし，データベース内でデータが動的に減ったり，増えたりする状況でクエリと類似したデータを探し続ける問題が多く取り扱われている．[1][2][3]

一方で最近，データストリームをデータの集合と捉え，集合間類似検索により類似データストリームを検索する新しい問題設定が取り扱われ始めている．例えば，Xu ら [4] は 1 データストリームのスライディングウィンドウ内の要素群をクエリ集合として，(時間によらない静的な) 集合のデータベースからクエリと最も類似した上位 k 個の集合を探す検索問題を提唱した．逆に文献 [5] では，データベースに複数のデータストリームが登録された状況で，(時間によらない静的な) 集合をクエリとして，スライディングウィンドウの内容が最も類似した上位 k 個のデータストリームを検索する問題を取り扱った．これらの問題では，時間経過によりデータストリームに新しい要素が来ると，スライディングウィンドウの内容が変わるため，検索結果を随時更新する必要がある (Continuous Similarity Search)．

上記の先行研究 [4], [5] では集合の要素はアルファベットであった．これに対して，本研究は集合の要素をテキストに拡張した問題設定 (CTS 問題，Continuous similarity search for Text Streams) を考える．より，具体的には CTS 問題は，1 つのテキストストリーム X_u のスライディングウィンドウをクエリテキスト集合として，スライディングウィンドウの類似度が閾値 ϵ 以上のテキストストリームをデータベースから探索するレンジ探索である．CTS 問題は，SNS の投稿内容から類似ユーザを探すシナリオをモデル化している．テキストストリーム U は 1 ユーザ U が投稿した tweet の集合であり，時間と共に新しい tweet が投稿される．U のスライディングウィンドウはユーザ U が最近投稿した tweet の集合に対応する．そして，スライディングウィンドウが似たテキストストリームを探すことにより，U と最近の投稿内容が似た他のユーザを見つけられる．本論文では CTS 問題に対する枝刈りベースの高速アルゴリズムを提案し，人工データを用いた実験評価により提案アルゴリズムが，単純なベースラインアルゴリズムよりも高速に動作することを示す．

本研究の先行研究に対する新規性は以下の 2 点である．

- 集合の要素をアルファベットからテキストに拡張した．要素がテキストの場合は，2 つの集合に対する積

[1] 電気通信大学大学大学院情報理工学研究科
[a)] kubo@sd.is.uec.ac.jp
[b)] koga@sd.is.uec.ac.jp

集合が自明に定まらないため類似度の算出が複雑になる．本研究では，積集合を 2 部グラフのマッチングに帰着して求める．

- クエリとデータベースの両方がデータストリームとなっている問題設定を初めて取り扱った．先行研究 [4], [5] ではクエリ，データベースの一方だけがデータストリームである．

以下に本論文の構成を述べる．2 節で CTS 問題を正式に定義し，3 節でベースラインとなる CTS 問題に対する単純アルゴリズムを記述する．4 節では提案手法となる枝刈りベースの CTS 問題に対する高速解法を記述する．5 節では提案手法を人工データで実験的に評価する．6 節は結論である．

2. 問題設定

本節では，本研究が定義した CTS 問題の問題設定について述べる．本研究ではユーザ U を U が投稿したテキストオブジェクトの集合で特徴づける．これは twitter における tweet である．テキストオブジェクトは単語の集合になっている．そして，各ユーザ U に 1 つの新しいオブジェクトが時間毎に追加されるストリーム環境を想定する．各データストリーム U には幅 w のスライディングウィンドウが設定され，スライディングウィンドウは U に到着した直近 w 個の要素が含まれる．つまり，CTS 問題では時刻 T のユーザ U が保有するテキスト集合 U_T は，$U_T = \{u_{T-W+1}, u_{T-W+2}, ..., u_T\}$, $|U_T| = W$ となる．時刻 T から $T+1$ に更新された時，図 1 のようにスライディングウィンドウに u_{T+1} が到着し，u_{T-W+1} が離脱する．

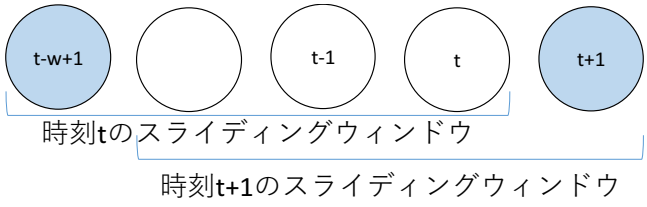

図 1　スライディングウィンドウモデル

CTS 問題は，あるユーザをクエリ V に設定し，毎時刻クエリ V とのユーザ間類似度 $sim(V_T, U_T)$ が ϵ 以上であるユーザ U の集合を検索するレンジ探索である．

ユーザ間類似度を計算するには，2 つのオブジェクトが共通ペアであるかどうかの判定が必要になる．そこで 2 つのオブジェクト o, o' が共通ペアであることの定義を与える．

o と o' のテキスト類似度 $\tau(o, o')$ をテキストに含まれる単語集合の Jaccard 類似度

$$\tau(o, o') = \frac{|o \cap o'|}{|o \cup o'|}$$

とする．o と o' は，$\tau(o, o') \geq$ 閾値ϵ_{doc} という条件を満たすならば，共通ペアである．この定義の直感的な意味は，共通ペアはテキスト情報が似たオブジェクト同士のペアであるということである．この共通ペア同士で辺を結び二部グラフを完成させる．なお，閾値 ϵ_{doc} はレンジ探索の中で指定されるパラメータである．そして，ユーザペア V と U 間の時刻 T における類似度は，V が生成したテキストオブジェクトの集合と U が生成したテキストオブジェクトの集合間の類似度 $sim(V_T, U_T)$ により表現する．$sim(V_T, U_T)$ を式 (1) のように定義する．

$$sim(V_T, U_T) = |M| \tag{1}$$

式 (1) において，$|M|$ は V_T, U_T を部分集合とし，共通ペアで辺を結ぶことで完成する二部グラフの最大マッチングサイズである．

図 2 に $\epsilon_{doc} = 0.5$ の場合のテキストオブジェクトのマッチングの例を示す．図 2 のオレンジの 1 と青の 2 のようにテキスト間の jaccard 類似度が 0.5 以上であるオブジェクトのペアに辺を結ぶ．図 2 の場合のユーザ間類似度は 2 である．

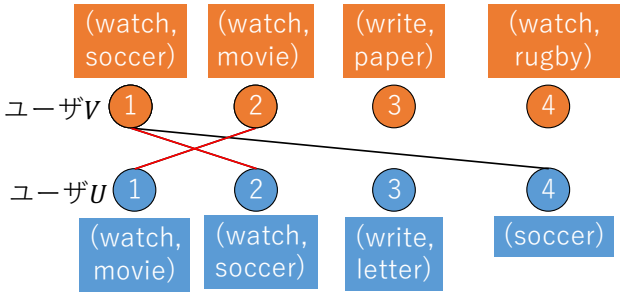

図 2　テキストオブジェクトのマッチングの例

CTS 問題は時刻 T が変化するとクエリ V_T とユーザ U_T も変化し，検索結果も変わるので検索結果を更新することが求められる．つまり，CTS 問題は，検索結果を毎時刻更新するため毎時刻グラフマッチングを求める必要がある．しかし，グラフ頂点が更新される度に最大マッチングの計算を行うことは計算量的に困難である．そこで本研究ではユーザ間の極大マッチングのサイズが閾値 ϵ 以上である場合に類似と判定する近似レンジ探索を行う．これは以下の 2 点により妥当である．

(1) 極大マッチングのサイズは最大マッチングのサイズを超えない
(2) 極大マッチングのサイズは最大マッチングのサイズの半分以上である [6]

極大マッチングに変更することで一度成立したマッチングを変更しなくなる．

3. 単純なアルゴリズム

本節では，CTS 問題を解く単純なアルゴリズム SIMPLE

を説明する．本稿では，SIMPLE は提案手法に対するベースラインとなる．SIMPLE では常に完成した極大マッチングを保持する．とくに，スライディングウィンドウが変化するたびに，極大マッチングを更新し完成させる．CTS 問題は極大マッチングサイズが ϵ 以上であるかを決定する問題であるが、SIMPLE では極大マッチングを完成させた後にマッチングサイズが ϵ 以上であるかを判定する．

以下では，時刻が $T-1$ から T に変化した時に SIMPLE がどう極大マッチングを更新するかを述べる．時刻が $T-1$ から T に変化した時，U，V のスライディングウィンドウがそれぞれ U_{T-1} から U_T, V_{T-1} から V_T に変化する．U_T に新規に追加されたテキストオブジェクト IN_U，U_{T-1} から離脱したテキストオブジェクトを OUT_U とする．V 側についても同様に IN_V，OUT_V を定義する．つまり，時刻が $T-1$ から T に変化した時に以下の 4 つのイベントが発生する．

(1) IN_U の到着
(2) OUT_U の離脱
(3) IN_V の到着
(4) OUT_V の離脱

これらの 4 種類のイベントに対して極大マッチングの更新手続きを定める必要があるが，SIMPLE は U,V に対して対称に動作するので，本稿に IN_V の到着, OUT_V の離脱に対する処理のみ記述する．なお，SIMPLE では時刻 $T-1$ の極大マッチングにおいて，離脱する OUT_U, OUT_V と関連しないマッチングは時刻 T でもマッチング判定をやり直すことなく維持する．

3.1 IN_V に対する処理

IN_V が到着した時，IN_V はマッチングサイズを増加させる可能性がある．よって極大マッチングを完成させるには IN_V のマッチング判定が必要になる．図 3 のように IN_V とマッチングに属さない U のオブジェクト群との間でマッチング判定を行う．IN_V とマッチングする U 側のオブジェクトが見つかった場合は，極大マッチングが完成するので残りのオブジェクトとのマッチング判定は打ち切る．

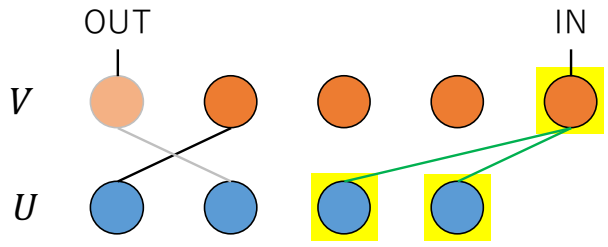

図 3　IN_V に対する処理

3.2 OUT_V に対する処理

OUT_V が極大マッチングに属していなかった場合, OUT_V が離脱してもマッチングが変化しないので何もする必要はない．逆に OUT_V が極大マッチングに属していた場合は，OUT_V の離脱によりマッチングサイズが 1 減る．この時 OUT_V とマッチングしていた U 側のオブジェクト (O_U とする) はマッチングサイズを増加させる可能性がある．そこで，図 4 のように O_U とマッチングに属さない V のオブジェクト群との間でマッチング判定を行う．O_U とマッチングする V 側のオブジェクトが見つかった場合は，極大マッチングが完成するので残りのオブジェクトとのマッチング判定は打ち切る．

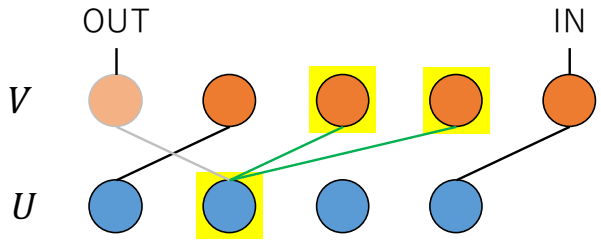

図 4　OUT_V に対する処理

4. 提案手法

本節では，マッチング判定の回数を減らして CTS 問題を高速に解くアルゴリズムを提案する．3 章で述べた SIMPLE は極大マッチングを完成させるため，$\forall T$ に対して U_T, V_T に追加されたオブジェクトは即座にマッチング判定される．この結果，全オブジェクトが到着時にマッチングされる．これに対して提案手法では極大マッチングを完成させず，$sim(U_T, V_T)$ が閾値 ϵ 以上かどうかを判定するのに必要な分だけオブジェクトをマッチング判定する．つまり，マッチング判定を行う頂点を限定することで計算時間を短縮する．到着したオブジェクトのマッチング判定をできるだけ遅らせる特性から提案手法を遅延評価法と名づける．

遅延評価法では，V_T に含まれるオブジェクトを U_T とマッチングするかに基づいて 3 種類に分類して管理する．

(1) V_M: U_T とのマッチングに属するオブジェクト集合
(2) V_{NM}: U_T とのマッチングに属さないことが確定したオブジェクト集合
(3) V_{UM}: マッチングに所属するかが未定なオブジェクト集合

同様に U_T 内で V_T とのマッチングに属するオブジェクト集合を U_M とする．マッチングの定義より $|V_M| = |U_M|$. V_{NM} の定義において「U_T とのマッチングに属さないことが確定した」とは、U_M 以外のオブジェクト全て，つまり $U_T - U_M$ の全オブジェクトと (テキストが似ていない理由で) マッチングできないことが確定したということである．したがって，V_{NM} に含まれるオブジェクトは $U_T - U_M$ 内

のオブジェクトすべてとマッチング判定済みである．一方，V_{UM} に含まれるオブジェクト (O_V とする) に対しては，$U_T - U_M$ 内に O_V とのマッチング判定が済んでいないオブジェクトが存在する．なお，SIMPLE アルゴリズムでは V_{UM} に属するオブジェクトは存在しないので，常に $V_{UM} = \phi$ となる．

遅延評価法では，

- $|V_M| \geq \epsilon$ となり U_T と V_T が類似と判定される
- $|V_{NM}| > (W - \epsilon)$ となり U_T と V_T が非類似と判定される

のいずれかが成立するまで V_{UM} に属するオブジェクトのマッチング判定を続ける．

時刻が $T-1$ から T に変化した時の遅延評価法の処理の流れを図 5 に示す．発生する 4 つのイベント (1)IN_V の到着 (2)OUT_V の離脱 (3)IN_U の到着 (4)OUT_U の離脱を処理したあと，(5)U_T と V_T が類似か非類似かの判定を実施する．以下では，(1) から (5) の処理を順に説明する．

4.1 IN_V に対する処理

ここでは V_T に新オブジェクトに IN_V が追加される．SIMPLE アルゴリズムとは異なり，遅延評価法では IN_V に対して即座にマッチング判定を行うことなく，$\mathrm{IN}_V \in V_{UM}$ とする．

4.2 OUT_V に対する処理

ここでは V_{T-1} から OUT_V が離脱する．この時，OUT_V を離脱直前の状態に応じて V_M，V_{UM}，V_{NM} のいずれかから除外する．

とくに $\mathrm{OUT}_V \in V_M$ であった場合は，OUT_V とマッチングしていたオブジェクト O_U はマッチング相手がいなくなるので U_M から除外する．この時，V_{NM} 内の各オブジェクト O_V は O_U とマッチングする可能性が発生し，「U_T とのマッチングに属さないことが確定しない」状態になる．そこで，O_V と O_U とのマッチング判定を行い，

- マッチングしないなら $O_V \in V_{NM}$
- マッチングするなら $O_V \in V_M$, $O_U \in U_M$

とする．

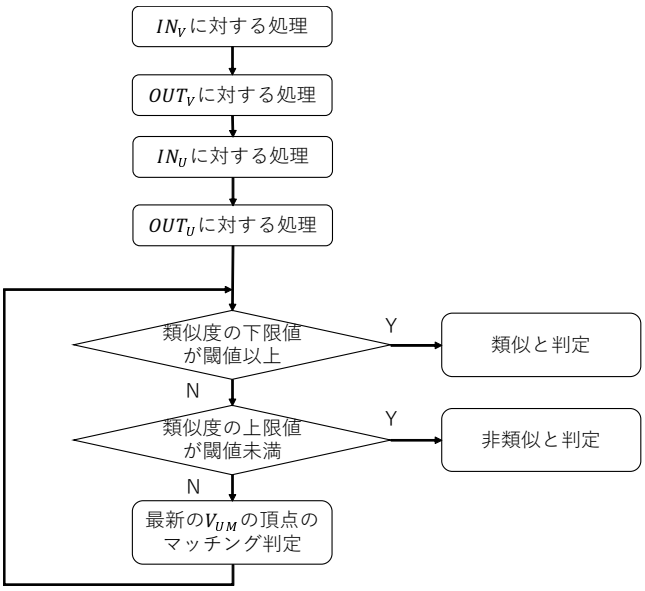

図 5 遅延評価法の処理の流れ

4.3 IN_U に対する処理

ここでは U_T に新オブジェクトに IN_U が追加される．その結果，V_{NM} 内の各オブジェクト O_V は IN_U とマッチングする可能性が発生し，「U_T とのマッチングに属さないことが確定しない」状態になる．そこで，O_V と IN_U とのマッチング判定を行い，

- マッチングしないなら $O_V \in V_{NM}$
- マッチングするなら $O_V \in V_M$, $\mathrm{IN}_U \in U_M$

とする．IN_U は V_{UM} に属するオブジェクトとはマッチング判定されない点に注意されたい．

4.4 OUT_U に対する処理

ここでは U_{T-1} から OUT_U が離脱する．この時，$\mathrm{OUT}_U \in U_M$ であった場合は OUT_U を U_M から除外する．さらに，OUT_U とマッチングしていたオブジェクト O_V はマッチング相手がいなくなるので，V_M から V_{UM} に移動する．この処理では新たにマッチング判定が実行されない．

4.5 U_T と V_T との類似性判定

ここでは，$|V_M| \geq \epsilon$ (類似) あるいは $|V_{NM}| > (W - \epsilon)$ (非類似) のいずれかが成立するまで，V_{UM} に属するオブジェクト O_V を 1 つずつマッチング判定する．具体的には O_V を $U_T - U_M$ 内のオブジェクト群とマッチングし，

- O_V とマッチングするオブジェクトが $U_T - U_M$ に存在するならば，$O_V \in V_M$ とする．この結果，V_M は 1 増加する．
  この時，O_V とマッチングしたオブジェクト O_U は $O_U \in U_M$ とする．
- O_V が $U_T - U_M$ のどのオブジェクトともマッチングしないならば，$O_V \in V_{NM}$ とする．この結果，V_{NM} は 1 増加する．

4.5.1 マッチング判定を行う順序

上述した U_T と V_T との類似性判定において

(1) V_{UM} に属するどのオブジェクトを優先してマッチング判定するか．

(2) O_V に対して $U_T - U_M$ 内のどのオブジェクトを優先してマッチング判定するか．

は自由度がある．遅延評価法では (1) については V_{UM} になった時刻が遅い新しいオブジェクトからマッチング判定を行い，(2) に関しても $U_T - U_M$ に所属した時刻が遅い新しいオブジェクトからマッチング判定を行う．新しいオブ

ジェクトからマッチング判定を行う理由は 2 つある．

- U_T 内のオブジェクト O_U と V_T 内のオブジェクト O_V 間でマッチングが一度成立すると，そのマッチングは O_U あるいは O_V がスライディングウィンドウから離脱するまでは有効である．すなわち，新しいオブジェクト同士がマッチングするほどマッチングが長期間有効になり，マッチングのやり直し回数が減少する．
- V_{UM} に属するオブジェクトは一度もマッチング判定されていないことが多々ある．これは V_T に到着したオブジェクト IN_V を何もしないで V_{UM} に投入することによる．U_T と V_T との類似性判定を行う時に，V_{UM} 内の古いオブジェクトをできるだけマッチング判定しないことで，一度もマッチング判定されないまま V のスライディングウィンドウから離脱するオブジェクト数が増えることが期待できる．

4.6 V_{NM} への再確定処理の保留

遅延評価法は，4.2 節の OUT_V と 4.3 節の IN_U に対する処理で V_{NM} に含まれるオブジェクト O_V が，V_{NM} に引き続き含まれるかを判定する．このように V_{NM} に引き続き含まれるかを判定する処理のことを「V_{NM} への再確定」と呼ぶ．V_{NM} への再確定を実施する理由は，O_V が U のどのオブジェクトをマッチング判定済みであるかを管理しないで済ませるためである．$O_V \in V_{NM}$ であれば，$U_T - U_M$ の全オブジェクトとマッチング判定済みであると言える．

一方で類似性判定の観点からは，V_{NM} への再確定は U_T と V_T が類似と判定される場合には無駄である．そこで本節では U_T と V_T が類似と判定されるかを予測し，類似と判定されそうな場合に V_{NM} への再確定を保留する改善案を提案する．具体的には，V_T 内の $|V_M|$ と $|V_{NM}|$ との割合を考え，

$$\frac{|V_M|}{|V_M| + |V_{NM}|} > \frac{\epsilon}{W} + \alpha \qquad (2)$$

であれば類似と判定されると予測する．$\alpha(> 0)$ は予測誤差をカバーするための定数パラメータである．

V_{NM} への再確定を保留されたオブジェクトは，$\frac{|V_M|}{|V_M|+|V_{NM}|} \leq \frac{\epsilon}{W} + \alpha$ が成立した時刻に V_{NM} に再確定される．しかし，U_T と V_T が継続して類似し続けた場合は，オブジェクトが再確定を保留されたままスライディングウィンドウから離脱するため，マッチング判定回数が削減されることになる．

5. 実験

本節では，遅延評価法の性能を実験的に評価する．実験のプラットフォームは，メモリ 16GB，Ubuntu20.04，Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz である．

本研究は人工データを用いて実験を行う．本研究で扱う人工データは以下の 3 種類のテキストをランダムに生成する．

A クエリユーザが 7 割の確率で生成
B クエリでないユーザが 7 割の確率で生成
C すべてのユーザが 3 割の確率で生成（共通要素）

つまり，クエリユーザのデータストリームは A のテキスト 7 割，C のテキストが 3 割の確率で追加される．一方で，クエリでないユーザは B のテキスト 7 割，C のテキストが 3 割の確率で追加される．これらのストリームのスライディングウィンドウを動かすことでクエリとデータベースを再現した．

各時刻 t において，クエリ V とユーザ間類似度 $sim(V_T, U_T)$ が閾値 ϵ 以上となるユーザ U を検索する実験を行った．時刻を $t = 1 \sim 1000$ まで進め，1000 回のレンジ探索にかかる合計の処理時間を測定した．また，本実験はスライディングウィンドウサイズ $w = 100$，データベースの集合数 100，という条件の下でマッチングサイズの閾値を変更して実行時間の比較を行う．

まず，遅延評価法の V_M と V_{NM} の数を利用した枝刈りと 4.5.1 項で述べたマッチング判定を行う順序に対する工夫の性能を評価するために下記の 3 つのアルゴリズムを実行時間で比較した．

(1) ベースラインアルゴリズム
(2) 遅延評価法
(3) 枝刈り+古いオブジェクト優先

実行時間の比較結果を図 6 に示す．図 6 から遅延評価法がどの状況でも 2 つのアルゴリズムよりも実行時間が短くなっていることが示された．また，4.5.1 項で述べたマッチング判定を行う順序に対する工夫の性能を評価するために一度もマッチング判定されないでスライディングウィンドウから離脱した V のオブジェクトの割合を比較した．一度もマッチング判定されないでスライディングウィンドウから離脱した V のオブジェクトの割合を図 7 に示す．図 7 から遅延評価法が古いオブジェクト優先のアルゴリズムと比べて多くのオブジェクトがマッチング判定を行わずにスライディングウィンドウから離脱したことが示された．したがって，遅延評価法のマッチングサイズの V_M と V_{NM}

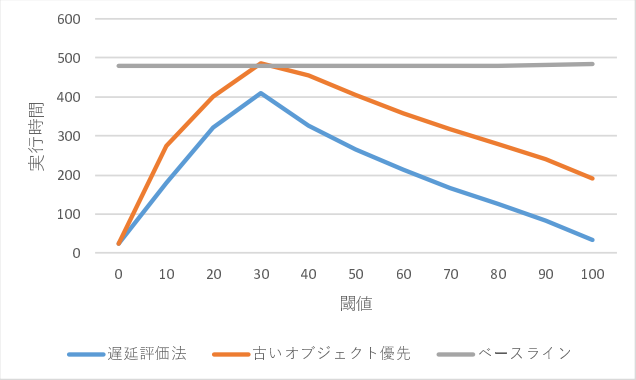

図 6 実行時間によるベースラインとの比較

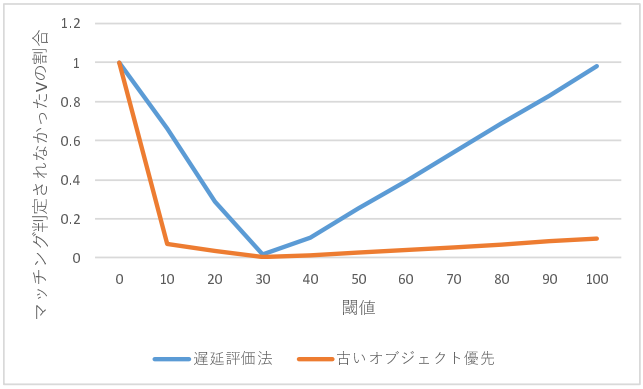

図 7 マッチング判定されなかった V のオブジェクトの割合

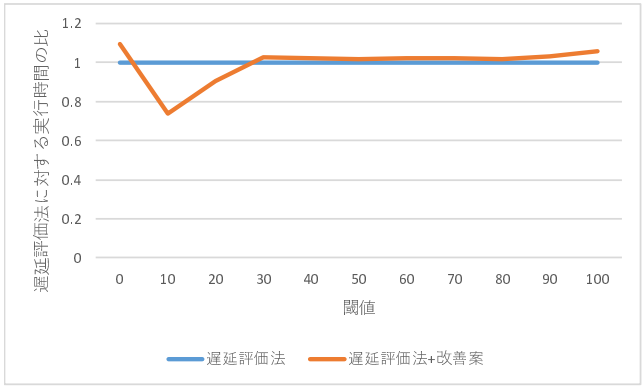

図 9 実行時間の比による遅延評価法と改善案の比較

の数を利用した枝刈りとマッチング判定を行う順序に対する工夫が有効であるといえる．

次に，4.6 節で述べた V_{NM} への再確定処理の保留を評価するために下記のアルゴリズムの比較を行った．

(1) 遅延評価法
(2) 遅延評価法+4.6 節で述べた改善案

遅延評価法と改善案を追加したアルゴリズムの実行時間による比較結果を図 8 に示す．また，遅延評価法と改善案の実行時間の比による比較結果を図 9 に示す．図 8 から類似と判定される状況で実行時間が短くなっていることが示された．また，図 9 から閾値が 10 の場合に実行時間が 2 割以上短くなっていることが示された．したがって，類似と判定されそうな場合に V_{NM} への再確定を保留する改善案が有効であるといえる．

6. おわりに

本研究は [4][5] が扱ったストリーム環境で動的に変化する集合間の類似検索において集合の要素をアルファベットからテキストに拡張した（類似度の導出が二部グラフのマッチングになる）類似検索問題（CTS 問題）を提唱した．また，本研究は CTS 問題を効率的に解くアルゴリズムを提案した．提案アルゴリズムにおける工夫は 3 つある．1 つ目はオブジェクト間のマッチング判定をユーザ間の類似や非類似の判定に必要な分だけに限定することである．2 つ目はマッチング判定なしでスライディングウィンドウから離脱するオブジェクトの数を増やすために，新しいオブジェクト優先でマッチング判定を行うことである．3 つ目は類似と判定されそうな場合にマッチングしないことが確定しているオブジェクトの再確定を保留することである．

そして，人工データを用いて提案したアルゴリズムの工夫を実行時間で比較した．遅延評価法はベースラインと比較して実行時間を大幅に減らすことができた．また，本研究では人工データのみで実験を行ったため，実データを用いた実験を行うことが今後の課題である．

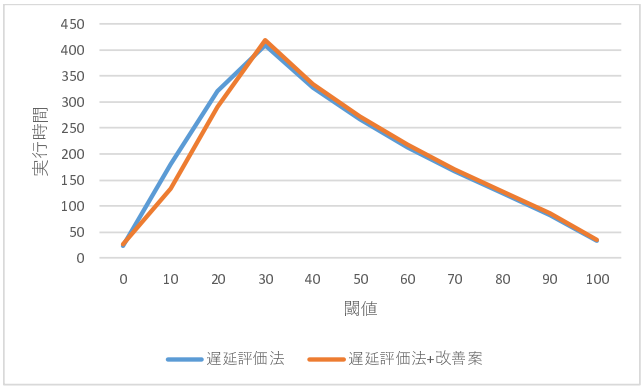

図 8 実行時間による遅延評価法と改善案の比較

謝辞 本研究は科研費基盤研究 (C)18K11311 の助成を受けたものである．

参考文献

[1] Leong Hou U, Junjie Zhang, Kyriakos Mouratidis, and Ye Li: Continuous Top-k Monitoring on Document Streams, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, vol.29, issue.5, May 1 2017
[2] Kyriakos Mouratidis, Spiridon Bakiras, Dimitris Papadias: Continuous monitoring of top-k queries over sliding windows, SIGMOD ’ 06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 635–646 (2006).
[3] Di Yang, Avani Shastri, Elke A. Rundensteiner, Matthew O. Ward: An Optimal Strategy for Monitoring Top-k Queries in Streaming Windows, Conference: EDBT 2011, 14th International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011.
[4] X. Xu, C. Gao, J. Pei, K. Wang, and A. Al-Barakati: Continuous similarity search for evolving queries, Knowledge and Information Systems, vol.48(3), pp.649-678, 2016.
[5] H. Koga and D. Noguchi: Continuous Similarity Search for Evolving Database, in Proc. 13th International Conference on Similarity Search and Applications(SISAP 2020), springer LNCS Vol. 12440, pp. 155-167, 2020.
[6] B. Korte and D. Hausmann: An Analysis of the Greedy Heuristic for Independence Systems, Annals of Discrete Math., 2:65–74, (1978).