

動的なテキスト集合に対する類似検索アルゴリズム ALE-Q の評価

土田 祐将[†] 古賀 久志^{††}

[†] 電気通信大学情報理工学域 〒182-8585 東京都調布市調布ヶ丘 1-5-1

^{††} 電気通信大学大学院情報理工学研究科 〒182-8585 東京都調布市調布ヶ丘 1-5-1

E-mail: [†]yuma.tsuchida@uec.ac.jp, ^{††}koga@sd.is.uec.ac.jp

あらまし 本研究では SNS から投稿内容が似た類似ユーザを見つける状況を想定し、ユーザをテキストストリームでモデル化し動的に変化するテキスト集合を対象とする類似検索を取り扱う。とくにこの問題に対する既存アルゴリズムを転置インデックスを使って高速化することを目指す。転置インデックスはテキスト検索における標準的な要素技術である。しかし、テキスト集合が動的に変化する場合、転置インデックスを更新するオーバーヘッドが生じるため、その適用は自明ではない。本研究では転置インデックスを使用するか否かをアルゴリズム内のテキスト照合パターンに応じて選択することにより、既存アルゴリズムを高速化できることを示す。

キーワード 類似検索、テキストストリーム、転置インデックス、枝刈りアルゴリズム

1 まえがき

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技術として注目されている。これまでのストリームデータを対象とした類似検索は、データストリームをデータベースをみなし、データベース内のデータが増減する環境を多く取り扱っている[1][2][3]。とくに最近は、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問題設定が取り扱われている。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 Koga ら[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に、検索結果を随時更新する必要がある (Continuous Similarity Search)。

Efstathiades ら[6]、Kubo ら[7]は SNS から類似ユーザを探すことを想定し、ユーザ U を投稿したテキスト群でモデル化し、テキスト集合間の類似検索を取り扱った。その中に CTS 問題[7]がある。CTS 問題とは、ユーザ U 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、スライディングウィンドウの類似度が閾値 ϵ_u 以上となる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問題である。ここで X_U はユーザ U が投稿したテキストの集合であり、時間と共に新しいテキストが追加される。 X_U のスライディングウィンドウは U の最近の投稿内容を表す。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。久保ら[7]は CTS 問題に対して遅延評価法という枝刈りベースの高速アルゴリズムを提案した。しかし、遅延評価法ではテキスト検索でよく使われる転置インデックスを使用しておらず、多くの処理時間が必要である。

本研究では CTS 問題を研究対象とし、遅延評価法を転置インデックスにより高速化することを目指す。転置インデックスはテキスト検索における標準的な要素技術である。しかし、CTS 問題ではテキスト集合が動的に変化するため、転置インデックスの更新オーバーヘッドを考慮する必要があり、転置インデックスの適用は自明ではない。本研究では、まず遅延評価法で起こるテキストマッチングのパターンを分析し、その結果を踏まえマッチング時に転置インデックスを使用するか否かを切り替える手法を提案する。そして、切り替えを適切に行うことで遅延評価法を高速化できることを実験評価により示す。なお、切り替え方が適切でないと、転置インデックス更新のオーバーヘッドのため、実行速度が却って遅くなる場合があることも確認した。本研究の新規性はクエリとデータベースの両側でテキスト集合が動的に変化する条件で、転置インデックスの適切な使用法を示したことである。

以下に本稿の構成を述べる。2 節で CTS 問題の定義を述べ、3 節でベースラインとなる遅延評価法を記述する。4 節では転置インデックスについて簡潔に説明し、5 節では提案手法となる遅延評価法の転置インデックスによる高速化方式を記述する。6 節では提案手法に改善を加えた手法を示し、7 節で提案手法を人工データと実データを用いて実験的に評価し、考察する。8 節は結論である。

5 節の提案手法の LE-Q, LE-QD は第 174 回データベースシステム研究発表会で発表済み[8]であるが、本稿では提案手法として記述し、提案手法の改善案である ALE-Q に関しては内容を変更し、再定義する。

2 CTS 問題

本節では、本研究で取り扱う CTS 問題 (Continuous similarity search for Text Streams) 問題[7] の定義を述べる。

CTS 問題ではユーザ A を A が投稿したテキストの集合で特徴づける。テキストは単語の集合である。例えば、Twitter のユーザを投稿した tweet で特徴づけるということになる。各ユーザ A が単位時間毎に新しいテキストを 1 つ投稿するストリーム環境を想定する。

各データストリーム A 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 A 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 A が保有するテキスト集合 A_T は、 $A_T = \{a_{T-W+1}, a_{T-W+2}, \dots, a_T\}$ 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 a_{T+1} が到着し、 a_{T-W+1} が離脱する。

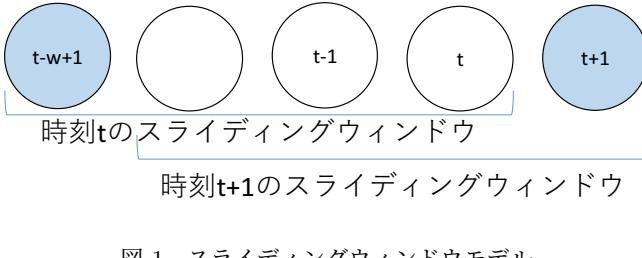


図 1 スライディングウィンドウモデル

CTS 問題は、多数のユーザのテキストストリームを保持したデータベース D とクエリユーザ V のテキストストリームが与えられて、毎時刻クエリ V とのユーザ間類似度 $\text{sim}(V_T, U_T)$ が閾値 ϵ_u 以上であるすべてのユーザ U を D から検索する継続的なレンジ探索である。したがって、スライディングウィンドウが変化する度に類似度判定結果を更新することが要求される。

ここでテキスト集合 V_T, U_T 間の類似度 $\text{sim}(V_T, U_T)$ は以下のように定義される。

- (1) V_T と U_T の類似テキストペア間に辺を張り 2 部グラフ $G(V_T, U_T)$ を構成する。ここで 2 つのテキスト $o \in V_T, o' \in U_T$ が類似テキストペアとなる条件は、 o, o' を単語集合と見なした時にその Jaccard 類似度

$$\tau(o, o') = \frac{|o \cap o'|}{|o \cup o'|} \quad (1)$$

が $\tau(o, o') \geq \text{閾値 } \epsilon_{doc}$ という条件を満たすことである。直感的には、共通単語を多く保有するテキストペアが類似テキストペアとなる。 $\tau(o, o')$ をテキスト類似度と呼び、閾値 ϵ_{doc} はレンジ探索の際に指定されるパラメータとなる。

- (2) グラフ G の極大マッチング M のサイズ $|M|$ を $\text{sim}(V_T, U_T)$ とする。

なお、上記で説明した問題は単位時間毎にテキストが投稿されるとしているが、本稿で扱うすべてのアルゴリズムは数単位時間ごとにテキストが投稿されるような、ユーザごとに一定でない時間間隔でテキストが投稿される場合にも対応可能である。

3 従来手法

本節では、久保らが提案した CTS 問題に対する類似検索アルゴリズムである遅延評価法を説明する。この遅延評価法はマッチング判定を行う際、ユーザ間類似度を正確に求めるのではなく、ユーザ間類似度が ϵ_u を越えるかのみを調べることによってマッチング判定回数の削減を行い、高速化を狙ったアルゴリズムである。遅延評価法では以下のようにクエリユーザ V のスライディングウィンドウ V_T 内のそれぞれのテキストを以下の 3 つの集合に分類する。

- V_M マッチングすることが確定したテキスト集合
- V_{NM} マッチングしないことが確定したテキスト集合
- V_{UM} マッチングするか未定なテキスト集合

さらに、ユーザごとにスライディングウィンドウ U_T 内のそれぞれのテキストを以下の 2 つの集合に分類する。

- U_M マッチングすることが確定したテキスト集合
 - U_{UM} マッチングするか未定なテキスト集合
- この遅延評価法は、
- 類似となる条件 $|V_M| \geq \epsilon_u$
 - 非類似となる条件 $|V_{NM}| > W - \epsilon_u$

のどちらかの条件を満たすまでマッチング判定をすることによって必要以上のマッチング判定処理を省き、ユーザ間類似度が閾値 ϵ_u 以上になるユーザを効率よく見つけ出すことを狙っている。遅延評価法の処理の流れを図 2 に示す。 V, U のスライディングウィンドウに入ってくるテキストを IN_V, IN_U 、スライディングウィンドウから離脱するテキストを OUT_V, OUT_U とする。

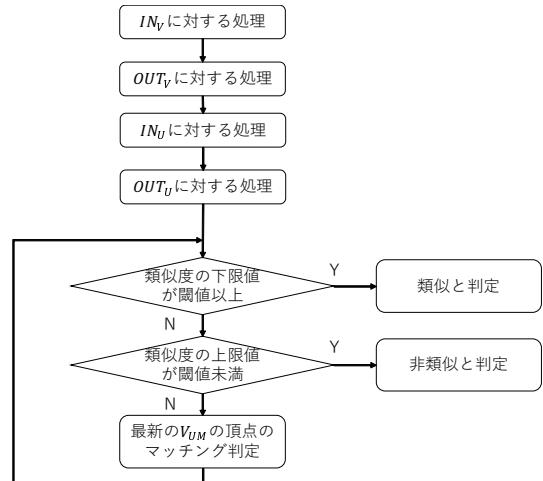


図 2 遅延評価法の処理の流れ

3.1 IN_V に対する処理

クエリ V のテキストストリームに新しいテキスト IN_V が到着し、 $IN_V \in V_{UM}$ とする。

3.2 OUT_V に対する処理

最も古いテキスト OUT_V を V_M, V_{NM}, V_{UM} の所属してい

る集合から除外する。特に $OUT_V \in V_M$ であった時は、 OUT_V のマッチング相手であったユーザ U のテキスト O_U はマッチング相手がいなくなることから O_U を U_M から除外する。さらに、 O_U は V_{NM} のテキストとマッチングする可能性があるため、 O_U と V_{NM} のテキストとのマッチング判定を行う。

- V_{NM} 内のテキスト O_V とマッチングした場合

$$: O_V \in V_M, O_U \in U_M$$

- マッチングしなかった場合 : $O_U \in U_{UM}$

とする。

3.3 IN_U に対する処理

ユーザ U のテキストストリームに新しいテキスト IN_U が到着する時、 IN_U と V_{NM} のいずれかのテキスト同士がマッチングする可能性が発生することから、 IN_U を起点とした V_{NM} 内の各テキストに対するマッチング判定を行う。

- V_{NM} 内のテキスト O_V とマッチングした場合

$$: O_V \in V_M, IN_U \in U_M$$

- マッチングしなかった場合 : $IN_U \in U_{UM}$

とする。

3.4 OUT_U に対する処理

スライディングウィンドウ内の最も古いテキスト OUT_U を U_M, U_{UM} の所属している集合から除外する。特に $OUT_U \in U_M$ であった時、 OUT_U とマッチングしていたテキスト O_V を $O_V \in V_{UM}$ とする。

3.5 類似・非類似の判定処理

ここでは各時刻 T でユーザ間類似度が ϵ_u を超えるかを判定するプロセスを示している。 $V_M (= U_M)$ の大きさは常に $\text{sim}(V_T, U_T)$ 以下であることから、 V_M の大きさがユーザ間類似度の下限値といえる。さらに V_{NM} に所属するテキストはマッチングすることはないため、スライディングウィンドウの大きさ W を用いて $W - |V_{NM}|$ がユーザ間類似度の上限値となる。したがって、ユーザ間類似度を正確に求めるのではなく、類似となる条件式

$$|V_M| \geq \epsilon_u \quad (2)$$

または、非類似となる条件式

$$W - |V_{NM}| < \epsilon_u \quad (3)$$

のどちらかを満たすまでマッチング判定を行えばよい。そこで、式 (2) または式 (3) のいずれかが成立するまで V_{UM} の各テキストを起点とした U_{UM} 内のテキストに対する以下のマッチング判定処理を繰り返し行う。

- U_{UM} のテキスト O_V と U_{UM} のテキスト O_U がマッチングした場合、 $O_V \in V_M, O_U \in U_M$ とする。この時、 V_M の大きさが 1 増加する。

- U_{UM} のテキスト O_V が U_{UM} のどのテキストともマッチングしなかった場合、 $O_V \in V_{NM}$ とする。このとき V_{NM} の大きさが 1 増加する。

3.6 マッチング判定を行うペアの順番

ユーザ間類似度を求める際にに行うマッチング判定は、 V_{UM} 内のテキストと U_{UM} 内のテキスト同士でマッチング判定を行うが、それぞれの集合内のどのテキストからマッチング判定を行うかは自由度が存在する。そこで遅延評価法では常に新しいテキストから優先してマッチング判定を行う。このとき、新しいテキスト同士がマッチングすることで長時間マッチングの関係が維持でき、どちらか片方のテキストがスライディングウィンドウを離脱したときのマッチング判定のやり直しの回数の削減が見込める。

4 転置インデックス

本節では本問題に対する転置インデックスの管理・利用について説明する。

遅延評価法はマッチング判定の際に、共通単語を 1 つも持たないテキストに対してもマッチング判定を行っている。共通単語を 1 つも持たないとき、Jaccard 類似度は 0 となり類似判定をする必要がない。転置インデックスを用いると共通単語を含むテキストのみを参照することができ、Jaccard 類似度が 0 になるテキストへのマッチング判定を避けることができる。よって遅延評価法に転置インデックスを用いたマッチング判定処理を導入することで類似性判定の処理の高速化を狙う。

4.1 データ構造

マッチング判定はスライディングウィンドウ内のテキストを対象としているので、ユーザ U が持つ時刻 T でのテキスト集合 U_T は、スライディングウィンドウの大きさ W を用いて $U_T = \{u_{T-W+1}, u_{T-W+2}, \dots, u_T\}$ と表せる。そしてテキスト u_T が持つ単語は、 $u_T = \{t_1, t_2, \dots, t_{|u_T|}\}$ と表せる。各テキストはそれが持つ単語によって索引付けされ、単語 t_i に対するリスト L_i は $\langle ID(u_T), f_i \rangle$ と表されるテキスト情報を持ち、 $ID(u_T)$ はテキスト u_T が持つテキスト ID、 f_i はテキスト u_T 内の単語 t_i の出現回数である。 u_T のハッシュ値を計算し、その値にリスト L_i を対応させたハッシュテーブルを転置インデックスとする。

4.2 構築

静的なテキスト集合を対象とした転置インデックスであれば、テキスト集合の内容は変化せず、転置インデックスを更新する必要はない。しかし、本問題は動的なテキスト集合を扱うため転置インデックスを更新する必要がある。そのため動的なテキスト集合に適した転置インデックスのデータ構造を提案する。テキストストリームには毎時刻スライディングウィンドウに新しいテキストが到着し、古いテキストが離脱する。このことから単語 t_i に対応付けられるリスト L_i にはキューを用いる。新しいテキストが到着したときには L_i の先頭に追加、古いテキストが離脱するときには L_i の末尾からテキストを削除することで管理コストを抑えつつ動的なテキスト集合に対応した転置インデックスを管理できると考える。

よって転置インデックスの単語 t_i に対するリスト L_i には先頭

に近いほど新しいテキストが格納されることになる。

4.3 利用

テキスト集合 U_T とテキスト集合 V_T のテキストについて極大マッチングを求めるを考える。テキスト集合 V_T に対して転置インデックス ii_{V_T} が構築されているとする。まず、テキスト集合 U_T からテキスト u_T を取り出す。テキスト u_T が持つ単語 $t_1, t_2, \dots, t_{|u_T|}$ に対応するリストを ii_{V_T} から取得するが、ここで取得したリストを用いてテキストを参照する方法として DAAT を用いる。

- DAAT (Document-At-A-Time)

テキスト単位で処理を行う。

- 手順 1 最も投稿時間の新しいテキストが根に来るヒープ H を用意する。
 - 手順 2 u_T から単語 t_1 を取り出し、対応するリストの先頭を H に追加する
 - 手順 3 手順 2 をすべての単語に対して行う。
 - 手順 4 ヒープ H から最も新しいテキストを取り出し、所属するリストの次のテキストを再びヒープ H に追加する。
- 新しいテキストから優先してテキストを参照することができるため、新しいテキストからマッチング判定をすることが可能である。DAAT の手順を図 3 に示す。

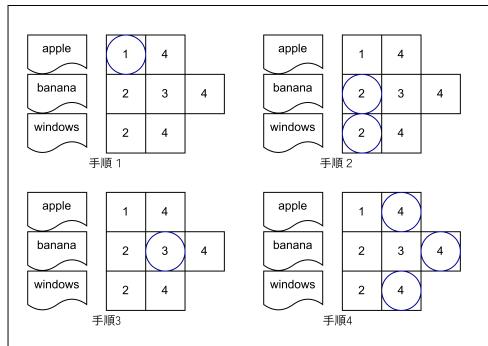


図 3 DAAT の手順

5 提案手法

遅延評価法ではマッチング判定を行う際、クエリとデータベースにおいて共通する単語が 1 つもない場合でも、テキスト類似度を求めている。共通する単語が存在しない状況においてはテキスト類似度は 0 となり、マッチング判定対象に含める必要がない。よって、転置インデックスを用いて共通する単語が存在するテキストを抽出し、そのテキスト集合のみにマッチング判定処理を行うアルゴリズムを提案する。転置インデックスは 4 節に従い、毎時刻テキストが到着し、スライディングウィンドウの内容が変化する度に更新するものとする。

遅延評価法のマッチング処理には以下で示す 2 つのパターンが存在する。

パターン 1 U_T 内のテキスト $o \in U_T$ を起点とするマッチング判定 (図 2 における OUT_V に対する処理、IN_U に対する処理)

パターン 2 V_T 内のテキスト $o \in V_T$ を起点とするマッチング判定 (図 2 における最新の V_{UM} の頂点のマッチング判定)

5.1 LE-Q

遅延評価法のパターン 1 の部分に対して転置インデックスを用いたマッチング判定を取り入れたアルゴリズムを LE-Q(Lazy Evaluation with inverted indices for Query) として提案する。このアルゴリズムを用いる場合、クエリ 1 人分のスライディングウィンドウに対する転置インデックスを管理する必要がある。

5.2 LE-QD

遅延評価法のパターン 1、パターン 2 の部分に対して転置インデックスを用いたマッチング判定を取り入れたアルゴリズムを LE-QD(Lazy Evaluation with inverted indices for Query and Database) として提案する。このアルゴリズムを用いる場合、クエリ 1 人分に加えデータベース上のユーザ全員のスライディングウィンドウに対する転置インデックスを管理する必要がある。

5.3 マッチング判定方法

マッチング判定を行う対象は新しいテキストから優先してマッチングすることが優位であることが遅延評価法で示されている。キューには到着した順に先頭からテキスト情報を格納しているため、マッチング判定は先頭から DAAT 方式で参照していく。判定の処理に関しては、どちらのパターンにおいても共通の処理を行い、そのマッチング判定方法を以下に示す。

- (1) 調べるものとテキスト u_T のそれぞれの単語 $\{t_1, t_2, \dots, t_{|u_T|}\}$ に対応するリストを転置インデックスを用いて取得する。
- (2) DAAT 方式に則って取得したリストを先頭から新しいテキスト優先で並列に走査していく。
- (3) 参照したテキストが
 - パターン 1 では V_{UM}
 - パターン 2 では U_{UM}

に含まれていれば以下の処理を行う。

- (4) 同一 ID をもつテキストごとに共通する単語がいくつ存在するかを数える。
- (5) 異なるテキスト ID が登場する、または最後のテキストを参照したときに、式 (1) を用いてテキスト類似度を計算する。
- (6) 式 (1) の値が ϵ_{doc} 以上であればマッチング、そうでなければ非マッチングとする。

6 適応的な LE-Q (ALE-Q)

LE-Q ではデータベース側に転置インデックスを持たないため、パターン 2 の V_T を起点とするマッチング判定で転置インデックスを使用しない。

ここでユーザ間類似度 $sim(V_T, U_T)$ が ϵ_u と近い時、パター

ン2の処理で V_{UM} の多くのテキストをマッチング判定しないと類似、非類似を確定できない。つまり、枝刈りの効果が期待できない。このように枝刈りの効果が弱い場合に、 V_{UM} に属するテキストを起点とする転置インデックスを使わないマッチング判定を、 U_{UM} に属するテキストを起点としてクエリ側の転置インデックスを使用するマッチング判定に切り替える改善案を提案する。具体的には ϵ_r を閾値パラメータとして、以下の基準でマッチング判定モードを切り替える。

- モード1: $|V_{UM}| < \epsilon_r$ ならば、 U_{UM} に属するテキストを起点としてマッチング判定
- モード2: $|V_{UM}| \geq \epsilon_r$ ならば、 V_{UM} に属するテキストを起点としてマッチング判定

モード1の U_{UM} に属するテキストを起点とするマッチング判定処理を以下に述べる。 $\forall O_U \in U_{UM}$ に対してマッチング判定を行っており、極大マッチングを完成させる点に注意されたい。

Step 1: $\forall O_U \in U_{UM}$ を起点とし、 V_{UM} 内のテキストを対象としたマッチング判定を転置インデックスを用いて行う。その結果、 O_U と V_{UM} のテキスト O_V がマッチングした時、 $O_U \in U_M$, $O_V \in V_M$ とする。

Step 2: U_{UM} の全テキストのマッチング判定後、 V_{UM} に残っているテキストをすべて V_{NM} に分類する。

以上をまとめると、 $|V_{UM}|$ が小さく枝刈りが期待できない状況では遅延評価をせずに、モード1で転置インデックスを使って高速に極大マッチングを完成させるということになる。これを ϵ_r を用いたマッチング切替と呼ぶ。

一方で、 $|V_{UM}|$ が極端に小さい時、転置インデックスを用いたマッチング判定が転置インデックスを用いないマッチング判定よりも遅い場合があると考える。

U_{UM} に属するテキストを起点とした転置インデックスを用いるマッチング判定では、

- (1) テキストに含まれるそれぞれの単語の転置リストを取得
- (2) DAAT方式でテキストを走査

を行う。転置インデックスはクエリユーザの全テキストにより構成されるため、 V_M, V_{NM}, V_{UM} 所属のテキストが混在する。よってマッチング判定を行う際、パターン2において

- マッチング相手が V_{UM} 所属であること

を満たすテキストにのみマッチング判定をする必要がある。このことから V_{UM} に属するテキストが少ないと、転置インデックスを参照しても V_{UM} に属さないテキストが取り出され、マッチング判定となる V_{UM} に属するテキストとなかなか遭遇しない。この場合、マッチング判定は高速に行えるが V_{UM} に所属するテキストを見つけることに多くの時間がかかり、結果的に転置インデックスを用いないマッチングの方が早いと考える。よって V_{UM} の大きさが閾値 ϵ_m 以下のとき、マッチング方法を転置インデックスを用いないマッチングに変更する。これを ϵ_m を用いたマッチング切替と呼ぶ。

このように、遅延評価を実施するかを適応的に切り替えるこ

とから、LE-Qに ϵ_r を用いたマッチング切替と ϵ_m を用いたマッチング切替を取り入れたアルゴリズムをALE-Q(Adaptive LE-Q)と名付ける。

7 実験

従来手法となる遅延評価法、本稿で提案したLE-Q, LE-QD, ALE-Qの性能を比較する評価実験を行った。各実験において、

- (1) 実行時間
- (2) マッチング判定回数

の比較を行う。また、実験を行う環境として、Intel(R) Core(TM)i7-6700 CPU @3.40GHz, 16GB メモリ、Ubuntu20.04を用意した。各実験において特に指定しないとき、以下の設定を用いる。

- スライディングウィンドウの大きさ 100
- シミュレートする時刻 1 から 1000
- テキスト類似度の閾値 ϵ_{doc} を 0.1
- ユーザ間類似度の閾値 ϵ_u を 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 と変化させた。

7.1 人工データセット

人工データセットを以下のルールに則って生成した。全ユーザは3単語で構成されるテキストを1000個ずつ持つ。単語集合 Φ_q, Φ_{nq} を用意し、

- $\Phi_q \cap \Phi_{nq} = \phi$
- $|\Phi_q| = 1000$
- $|\Phi_{nq}| = 1000$

とする。

- クエリユーザのテキストは Φ_q から重複ありでランダムに選んだ単語で構成する。
 - データベース側にはユーザを100人用意する。 i 番目($1 \leq i \leq 100$)のユーザの時刻 t ($1 \leq t \leq 1000$)のテキストは、
 - p の確率で $t - 100 \leq t' < t + 100$ を満たす t' をランダムに決定し、クエリの時刻 t' のテキストとする。
 - $1 - p$ の確率で Φ_{nq} から重複を許してランダムに選んだ単語とする。

以上の条件をもとに $p = 0.96^{(i-1)}$ としたときのデータセットを作成する。

- i が小さいほどクエリユーザのテキストを選ぶ確率が大きくなるため、 i が大きいユーザほど、クエリユーザとの類似度の期待値が小さくなる。

このデータセットを用いて、各時刻に1つのテキストが到着する環境をシミュレートする。

7.2 CoPhIR データセット

実データとしてFlickrアーカイブからメタデータを抽出したCoPhIR(Content-based Photo Image Retrieval)[9][10]データセットを用いる。

CoPhIRデータセットに含まれる写真のメタデータから写真につけられたタグを単語とみなし、写真1枚に対して1つのテ

キストとし、本実験に適用した。なお、データセットは9108種類のタグと、100人のユーザから構成される。

ユーザごとに持つ写真の数は異なるが、すべて100個以上であるため、スライディングウィンドウの大きさ以上のテキストを持つ。テキストの数が1000に満たないユーザは古い時刻のテキストから順番に複製し、全ユーザが1000個のテキストを持つデータセットとする。

7.3 転置インデックスの性能評価

遅延評価法、LE-Q、LE-QDのアルゴリズムを人工データセット、CoPhIRデータセットを利用し、全ユーザの類似性を各時刻すべてにおいて求める実験を行った。人工データセットにおける実験結果を図4に示す。結果よりLE-QはLE-QDよりも高速に動作することが分かった。LE-QDはマッチング比較回数をかなり抑えることができたものの、転置インデックスの構築に時間がかかってしまい、結果的にLE-Qよりも多くの実行時間を要することとなったと考えられる。

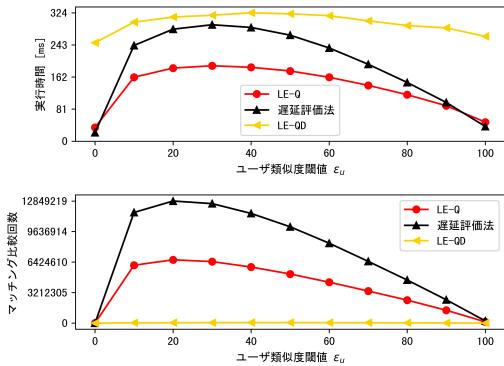


図4 性能評価(人工データセット)

CoPhIRデータセットにおける実験結果を図5に示した。図4と同様な結果を示し、LE-QDと遅延評価法よりLE-Qが高速であると言える。

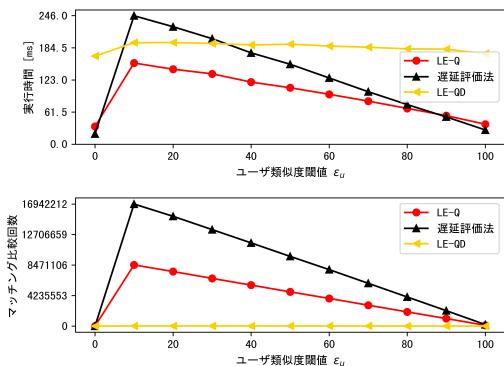


図5 性能評価(CoPhIRデータセット)

7.4 マッチング順序の評価

7.3節の実験からLE-QDよりもLE-Qの方が優れていると考えられるため、本実験ではLE-Qを対象とする。3.6節で示したとおりマッチング順序に関しては新しいテキストから優先してマッチング判定をすることが効率的であると考えられるが、この優位性が転置インデックスを用いたアルゴリズムでも確認できるかを調べる。そのためLE-Qにおけるマッチング処理を古いテキスト優先のマッチング処理に置き換え、全ユーザの類似性を各時刻すべてにおいて求める実験を行った。人工データセットにおける実験結果を図6に示した。結果より、新しいテキストから優先的にマッチングする方が実行時間が短縮され効果的であることが示された。これは新しいテキスト優先することでマッチング比較回数を削減することが出来た影響であると考える。

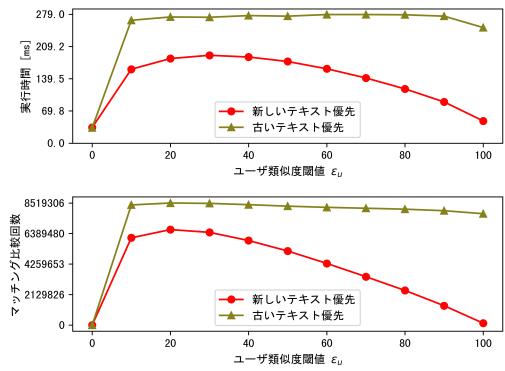


図6 LE-Qのマッチング順序の優位性評価(人工データセット)

CoPhIRデータセットにおける実験結果を図7に示した。図7は人工データによる実験と同様の結果を示し、テキストのマッチング順序として新しいテキストから優先することは有効であると確認できた。

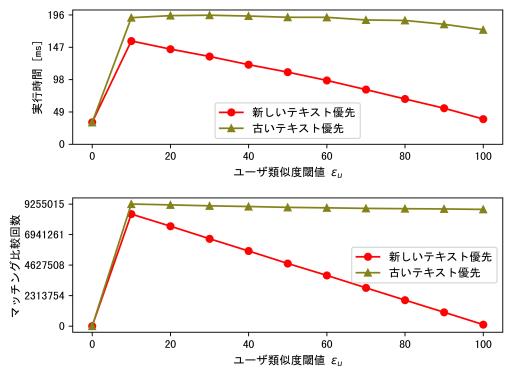


図7 LE-Qのマッチング順序の優位性評価(CoPhIRデータセット)

7.5 ALE-Q の評価

遅延評価法, LE-Q, LE-Q の改善案である ALE-Q を比較するために, 全ユーザの類似性を各時刻すべてにおいて求める実験を行った. ϵ_r を 55, ϵ_m を 5 としたときの人工データセットにおける実験結果を図 8 に示した. 結果より, ALE-Q の方が LE-Q より実行時間が短くなっていることがわかる. これはある程度少ない数のテキストのマッチング判定を行わなくてはいけないとき, 転置インデックスを用いたマッチングに変更することで共通の単語を 1 つも持たないテキストへのマッチングを避けることができたからであると考える. さらにかなり少ない数のテキストのマッチング判定を行わなくてはいけないとき, 転置インデックスを用いないマッチング方法にすることでマッチング判定を行うテキストの個数を減らすことが出来たからであると考える.

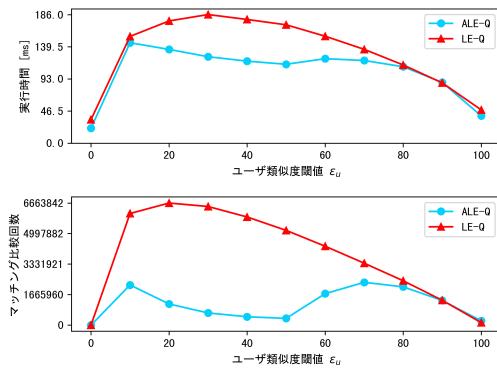


図 8 ALE-Q の性能比較 (人工データセット)

ϵ_r を 40, ϵ_m を 5 とした CoPhIR データセットにおける実験結果を図 9 に示した. 人工データの実験の図 8 と同様に ALE-Q の方が LE-Q より早くなっていることがわかる. 閾値 ϵ_r , ϵ_m を用いた転置インデクス利用の切り替えを適応的に行った結果, マッチング比較回数を削減することに繋がり, p の値によらず LE-Q よりも ALE-Q の方が実行時間が短くなるという結果が示された.

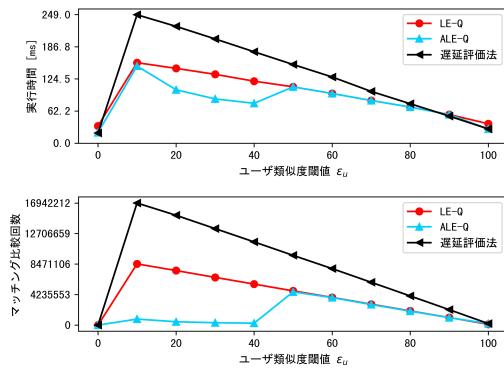


図 9 ALE-Q の性能比較 (CoPhIR データセット)

7.5.1 ALE-Q における切り替えの評価

ALE-Q では, ϵ_r を用いたマッチング切り替え, ϵ_m を用いたマッチング切り替えが有効であるかを示すために, ALE-Q-1 ($5(\epsilon_m) < |V_{UM}| < 55(\epsilon_r)$ のときモード 1 に切り替え) と, ALE-Q-2 ($|V_{UM}| < 55(\epsilon_r)$ のときモード 1 に切り替え) を比較し, 性能を評価した. 人工データセットにおける実験結果を図 10 に示す. 結果より, ϵ_m を用いたマッチング切り替えは有効であることが示された.

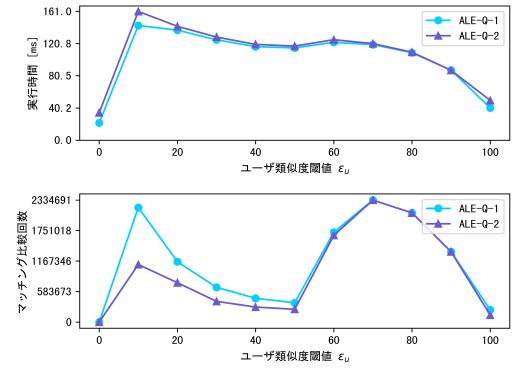


図 10 ϵ_m の評価 (人工データセット)

7.6 クエリ単語種類数の影響評価

以上の実験は全てクエリ側に存在する単語種類数 (Φ_q) を 1000 としていたが, Φ_q が 1000 より小さい場合, 転置インデクスに登録される単語の数が減り, 転置インデクスの 1 つの単語に対応するキーに多くのテキスト情報が格納され, キューが長くなると考えられる. よって Φ_q の方が実行時間に及ぼす影響の有無を実験によって確認した. 人工データセットにおいて Φ_q の値を 10 としたときの実行時間を図 11 に, 50 としたときの実行時間を図 12 に, 100 としたときの実行時間を図 13 に, 500 としたときの実行時間を図 14 に示した. 結果より, Φ_q が小さいほど LE-Q, ALE-Q の実行時間と遅延評価法の実行時間の差が減っていることがわかる. これは転置インデクスを参照する際, 非マッチングと判断するためにキーに格納された多くのテキスト情報を見なくてはいけないからであると考えた.

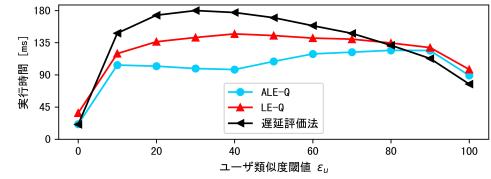


図 11 $\Phi_q = 10$ での比較 (人工データセット)

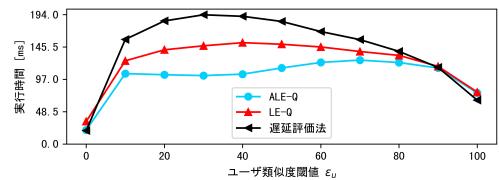


図 12 $\Phi_q = 50$ での比較 (人工データセット)

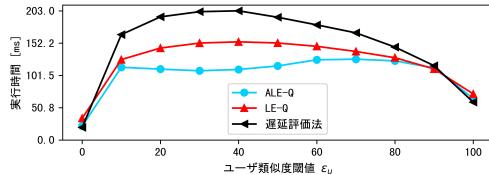


図 13 $\Phi_q = 100$ での比較 (人工データセット)

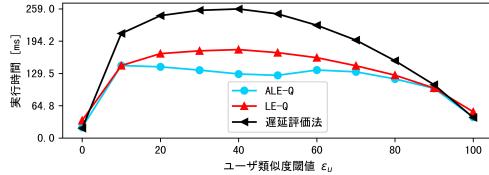


図 14 $\Phi_q = 500$ での比較 (人工データセット)

7.7 テキスト類似度の閾値 ϵ_{doc} の影響評価

以上の実験ではテキスト類似度の閾値 ϵ_{doc} を 0.1 としていたが、これを大きくしたときに実行時間に影響があるかを実験によって確認する。人工データにおいて $\epsilon_{doc} = 0.5$ としたときの実行時間を図 15, $\epsilon_{doc} = 1.0$ としたときの実行時間を図 16 に示す。結果より、テキスト類似度閾値 ϵ_{doc} によらず、ALE-Q は LE-Q, 遅延評価法より高速に動作することを示せた。

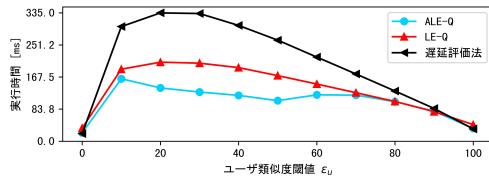


図 15 $\epsilon_{doc} = 0.5$ での比較 (人工データセット)

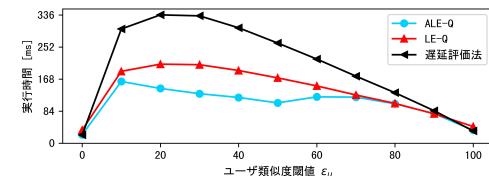


図 16 $\epsilon_{doc} = 1.0$ での比較 (人工データセット)

8 結 論

本研究では、久保ら [7] が提唱したストリーム環境で動的に変化するテキスト集合を対象とした類似検索である CTS 問題を取り扱った。そして、CTS 問題を解く既存手法 (遅延評価法) の転置インデックスを用いた高速化に取り組んだ。

CTS 問題では、クエリとデータベースの両者が時間と共に変化するため、更新オーバーヘッドを考慮して転置インデックスを導入することが求められる。本稿では、クエリテキスト集合 V_T にのみ転置インデックスを構築し、データベース側のテキスト集合 U_T には転置インデックスを保持しない LE-Q の方が、クエリ

とデータベース両者に転置インデックスを用意する方式 LE-QD よりも処理時間が短縮でき、適切であった。これはデータベース内の多数のテキストストリームに対して転置インデックスを作成すると、更新オーバーヘッドが膨大になるためである。

そして、LE-Q をさらに高速化するため、遅延評価法で枝刈り効率が悪い状況で V_T を起点とするマッチング判定を U_T を起点とするマッチング判定に適応的に切り替える手法 ALE-Q を提案した。 U_T を起点としてマッチング判定した場合、類似度を厳密計算することになり枝刈り効率は低下する一方で、マッチング判定そのものは転置インデックスにより高速化される。人工データ、実データを用いた実験により、ALE-Q が遅延評価法、LE-Q よりも実行速度が早くなることを示した。

今後の課題としては、ALE-Q でマッチング判定方法を切り替える条件の最適化が挙げられる。

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。

文 献

- [1] Junjie Zhang, Kyriakos Mouratidis, Ye Li, et al. Continuous top-k monitoring on document streams. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29, No. 5, pp. 991–1003, 2017.
- [2] Kyriakos Mouratidis, Spiridon Bakiras, and Dimitris Papadias. Continuous monitoring of top-k queries over sliding windows. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 635–646, 2006.
- [3] Di Yang, Avani Shastri, Elke A Rundensteiner, and Matthew O Ward. An optimal strategy for monitoring top-k queries in streaming windows. In *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 57–68, 2011.
- [4] Xiaoning Xu, Chuancong Gao, Jian Pei, Ke Wang, and Abdulla Al-Barakati. Continuous similarity search for evolving queries. *Knowledge and Information Systems*, Vol. 48, No. 3, pp. 649–678, 2016.
- [5] Hisashi Koga and Daiki Noguchi. Continuous similarity search for evolving database. In *International Conference on Similarity Search and Applications*, pp. 155–167, 2020.
- [6] Christodoulos Efstathiades, Alexandros Belesiotis, Dimitrios Skoutas, and Dieter Pfoser. Similarity search on spatio-textual point sets. In *EDBT*, pp. 329–340, 2016.
- [7] 久保幸平, 古賀久志ほか. ストリーム環境でのテキスト集合に対する類似検索. 研究報告数理モデル化と問題解決 (MPS), Vol. 2020, No. 11, pp. 1–6, 2020.
- [8] 土田祐将, 古賀久志ほか. 転置インデックスを用いた動的なテキスト集合に対する類似検索の高速化. 研究報告データベースシステム (DBS), Vol. 2021, No. 1, pp. 1–6, 2021.
- [9] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. CoPhIR: a test collection for content-based image retrieval. *CoRR*, Vol. abs/0905.4627v2, , 2009.
- [10] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, and Fausto Rabitti. Enabling content-based image retrieval in very large digital libraries. In *Proceeding of the Second Workshop on Very Large Digital Libraries*, pp. 43–50. DELOS: an Association for Digital Libraries, 2009.