集合間類似度を用いたストリームデータの top-k 類似検索における min-hash を用いた近似解法

古賀研究室 1610615 三原寛寿

2019 年 10 月 1 日

1 はじめに

昨今、リアルタイムで新しいデータが生成され、絶えず流れている。このストリームデータの中で、古いデータや必要のなくなったデータはすぐに破棄し、新しく流れてくるデータを処理しなくてはいけない。このストリームデータ処理技術の向上が課題として挙げられる。そして、ストリームデータの中からクエリに類似したデータを探す類似検索技術において、より正確で、処理の早い技術が求められている。例えば、類似検索技術はあるユーザーに対して、興味のある広告を見せるために使われる技術として用いられる。今回、扱うデータはクエリが固定でデータベースが変わっていくデータである。このデータに対して、類似検索する上で、データの量が多いと多くの時間がかかってしまう。それを改善するために今回、min-hash を用いた近似解法を提案する。

2 ストリームデータの top-k 類似検索

毎時刻アルファベットが一つ到着するストリームデータがあり、アルファベットを要素とする n 個の静的な集合であるデータベース $D = \{S_1, S_2, \cdots, S_n\}$ がある。そして、時刻 t のクエリ Q_t はストリームデータの直近の W 個の要素からなる集合である。

Q_t と類似する上位 k 個の集合を D から毎時刻検索する。類似度計算には Jaccard 係数を用いる

$$sim(S, Q) = \frac{|S| \cap |Q|}{|S| \cup |Q|}$$

Jaccard 係数とは、2 つの集合に含まれている要素のうち共通要素が占める割合である。

3 従来手法

3.1 min-hash

Min-hash とは、m 次元 2 値ベクトルとして表現された集合に対する確率的なハッシュ関数で,Jaccard 係数に対する類似検索技術である。そして、Min-hash によるハッシュ値が一致する確率は Jaccard 係数と一致する。集合に対する類似検索をハッシュテーブルで実現可能である。

3.2 Min-hash を用いた Jaccard 係数の近似解法

Min-hash を用いた方法では、k 個のハッシュテーブルに対して、ハッシュ衝突回数を数える。

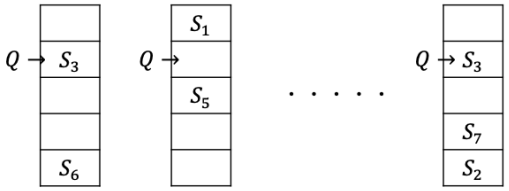

図 1 ハッシュテーブル

例えば、$k = 5$ で、S_3 に対して 2 回クエリが一致している場合は、2/5 となる。

3.3 従来手法での近似解法

まず、以下がハッシュテーブルの状態である。

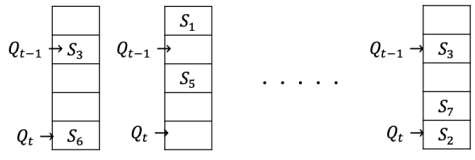

図 2 Q の変化

ハッシュ値が等しいかどうか、4 パターンある。

(1)S_i と Q_t-1 のハッシュ値が異なり、S_i と Q_t のハッシュ値が同じ。

(2)S_i と Q_t-1 のハッシュ値が同じで、、S_i と Q_t のハッシュ値が異なる。

(3)S_i と Q_t-1 のハッシュ値が同じで、、S_i と Q_t のハッシュ値が同じ。

(4)S_i と Q_t-1 のハッシュ値が異なり、、S_i と Q_t のハッシュ値が異なる。

(1) の箇所は Jaccard 係数を $1/|k|$ 増加させ、(2) の箇所は Jaccard 係数を $1/|k|$ 減少させ、(3),(4) の箇所は値を変えない。

このようにすることで、毎回 Jaccard 係数による類似計算することをなくし、簡略化する。

4 本研究で扱う問題

本来とは逆に、本研究では、クエリ Q はアルファベット集合で変化せず、データベース $D_t = \{S_1, S_2, \cdots, S_n\}$ は毎時刻更新されるデータを扱う。

5 提案手法

5.1 研究目的

新しい問題設定に対して、Min-hash を使った高速近似解法を構築する。

5.2 既存手法を単純に拡張した場合

データベースの全集合 S_i に対して、毎時刻ハッシュ値の再計算が必要となる。これはオーバーヘッドが大きくなる。

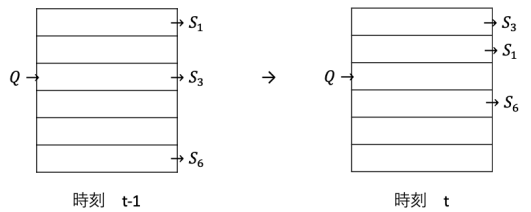

図 3　時刻による変化

5.3 高速化のアイデア

データベースの S_i の部分集合に対してハッシュ値を計算し、ハッシュテーブルに登録する。そのハッシュ値の部分集合の 1 部分がデータストリームから流された段階で、そのストリームの部分集合のハッシュ値を計算をし直すことによって、データが毎回変わるごとにではなくすることによって、ハッシュ値の更新回数を減らす。

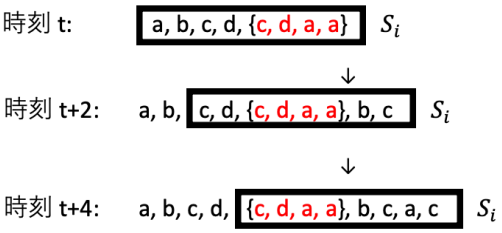

図 3　データベースの S_i のデータストリーム

6 進捗状況

私は、まだテーマを決めただけの段階である。今後はこの高速化のアイデアに対して、実現可能か煮詰めていく。

7 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 KNOWL- EDGE AND DATA ENGINEERING, vol.29, issue.5, May 1 2017