DEIM Forum 2021 H24-4

TopK アルゴリズムを用いた連続 KeyGraph による SNS 上のキーワード抽出

根津　裕太†　　三浦　孝夫†

† 法政大学 理工学研究科システム理工学専攻　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †yuta.nezu.6f@stu.hosei.ac.jp, ††miurat@hosei.ac.jp

あらまし　ソーシャルネットワーキングサービス (SNS) は, 世の中のニュースを知るための重要な情報源である．SNS 上のテキストを要約することによって，常に最新のニュースを把握することが可能となる．しかし，文書要約手法は多く存在するが，SNS のようなデータストリームに対応した手法はあまり知られていない．数少ないストリーム上のキーワード抽出法の一つとして，連続 KeyGraph が挙げられる．連続 KeyGraph とは，要約のためのキーワード抽出法である KeyGraph をデータストリームに適用した手法である．本研究では連続 KeyGraph に対して改善を試みる．従来の連続 KeyGraph では単語の出現頻度や共起頻度を Lossy Count によって保持していたが，本研究では TopK アルゴリズムによって保持することで，実行時間の短縮と必要メモリ量の削減を図る．また，従来の KeyGraph との比較を行うことにより，連続 KeyGraph の有用性を示す．

キーワード　データストリーム，ソーシャルネットワーキングサービス，Space Saving，KeyGraph，自然言語処理

1　前　書　き

近年，SNS は個人間のコミュニケーションツールとしての用途のみではなく，世の中のニュースを知るための情報源として重宝されている．Fedoryszak らの調査によると [1]，SNS はニュース収集の主要なソースとして印刷媒体を上回っているとの報告がある．SNS をニュース収集目的で使用する際，新聞やネットニュースと異なる点は，文書の校正（正しさの確認）がなく，投稿者がニュース配信を目的としていない点である．大半の投稿者は今起こったイベントについての感想を述べているだけである．SNS から除法を得ようとするためには，我々は数十件の投稿文を見比べて何が起きているのかを推測する必要がある．

SNS 上のテキストを自動要約することによって上記のような手間を省き，簡単にニュースを理解することができる．しかしながら，文書要約手法は多く存在するが，SNS のようなデータストリームに対応した手法はあまり知られていない．数少ないストリーム上のキーワード抽出法の一つとして，連続 KeyGraph [3] が挙げられる．連続 KeyGraph とは，要約のためのキーワード抽出法である KeyGraph [2] をデータストリームに適用した手法である．

本研究では連続 KeyGraph に対して改善を試みる．従来の連続 KeyGraph では単語の出現頻度や共起頻度を Lossy Count [8] によって保持していたが，本研究では TopK アルゴリズム [9] によって保持することで，実行時間の短縮と必要メモリ量の削減を図る．また，従来の KeyGraph との比較を行うことにより，連続 KeyGraph の有用性を示す．

2 章ではストリームアルゴリズムについて述べ，3 章では連続 KeyGraph について述べる．4 章では提案手法である TopK アルゴリズムによる連続 KeyGraph ついて述べ，5 章で実験を行う．最後に 6 章で結論を述べる．

2　ストリームアルゴリズム

近年，SNS を初めとしたインターネットの普及により，データストリームを対象としたデータマイニングの技術，すなわち，データストリームマイニングの必要性が高くなっている．有村らはデータストリームを (1) 膨大な量のデータが，(2) 高速なストリームを通じて，(3) 時間的に変化しながら，(4) 終わりなく到着し続けるという特性を持つ大規模データであると定義した [4].

また，データストリームを対象としたアルゴリズム（ストリームアルゴリズム）に必要な条件として (1) 非常に少ない計算資源を用いて (2) 長期間安定して動き続け (3) 利用者の要求に即応して (4) 近似的な解を返すという特性を挙げている．

ストリームアルゴリズムは抽出したい統計量に応じて様々な手法が提案されているが，本研究では頻出アイテムの抽出を目的としたアルゴリズムを紹介する．頻出アイテムに焦点を当てたアルゴリズムは大きく分けてスケッチベースと KV ベースの 2 種類に分類される [5].

スケッチベースアルゴリズム：スケッチベースアルゴリズムではハッシュによって全てのアイテムの頻度を記録するが，ハッシュの衝突を許容している．そのため，異なるアイテムで同じカウンタをインクリメントした場合，カウントの誤差が生じる．多くの場合，ハッシュ関数の異なる複数の配列で構成されたデータ構造を用いる．

例えば，CM スケッチ（Count-Min Sketch）[6] では，複数の配列でカウントを行い，頻度を要求された際は全ての配列でのカウントのうち最小の値を返す．また，アイテムのデクリメ

ントができない代わりに，カウント誤差の平均値を CM スケッチと比較して半分まで抑えた CU スケッチ (count-min sketch with Conservative Update) [7] なども提案されている．

基本的にはサイズを固定したハッシュ構造を用いるため，アイテム数に関わらず必要なメモリは変わらない．また，ハッシュの衝突を許すため，アイテムの探索時間が O(1) である．これらのアルゴリズムは出現するアイテムの種類が予め限られている場合（例えば商品の購買データ）には極めて有効であるが，自然言語等の新しいアイテム（語）が度々出現するデータには不適である．

KV アルゴリズム：KV ベースアルゴリズムでは頻度の高いアイテムのサブセット <ID, 頻度 > を保持する．大規模なコーパスにおいては，頻出アイテムが重要であり，低頻度アイテムを無視しても影響が少ないケースが多い．そのため，KV ベースでは低頻度アイテムの情報を随時破棄する．

Lossy Count [8] ではアイテムの頻度を閾値とし，閾値未満のアイテムを定期的に破棄する．Space Saving [9] ではランクを閾値とし，データ構造に空きがなくなると，保持しているデータのうち一番頻度が少ないアイテムと入れ替える．

本節では上記 2 つのアルゴリズムについて解説する．

2.1 Lossy Count

Lossy Count では頻度の誤差を許容する代わりに少ないメモリで頻出アイテムを保持する．Lossy Count では常にイプシロン劣シノプスを保持する．イプシロン劣シノプスとは以下のデータ集合を出力できる状態を示す．

(1) $F \geqq \gamma N$ を満たすデータ
(2) $(\gamma - \epsilon)N \leqq f \leqq F$ を満たす f

ここで F は真の頻度，f は見積もり頻度，γ は頻出項目の閾値，ϵ は誤り許容率を示す．

直観的には，Lossy Count を行うことにより各バケットに平均して 1 回以下しか出現しなかった要素を削除される．結果的に，頻度が閾値 ϵN 以下の要素が除外される．ここで N は全要素の総出現数である．

アルゴリズム

Lossy Count ではシノプスと呼ばれる頻出データの候補を保持する．シノプスは要素（またはその ID）e，見積もり頻度 f，f の最大許容誤差 Δ の集合である．ストリームを ϵ^{-1} 個ずつのデータが入ったバケットに区切り，各バケットに 1 から順番に番号をつける．アルゴリズムはデータストリーム上の要素を一つずつ読んでいき，シノプスを更新する．更新の操作は以下の 3 つである．

1. **カウント更新操作**：読んだ要素 e に対するシノプス (e, f, Δ) がすでにある場合，$f++$
2. **挿入操作**：読んだ要素 e に対するシノプスがない場合，新しいシノプス $(e, 1, b_{current} - 1)$ を作成
3. **除去操作**：新しいバケットに入るとき，$f \leqq b_{current} - \Delta$ を満たすシノプスを全て消去する．

また，出力を要請された場合は $f \geqq (\gamma - \epsilon)N$ を満たす e と f のペアを出力する．

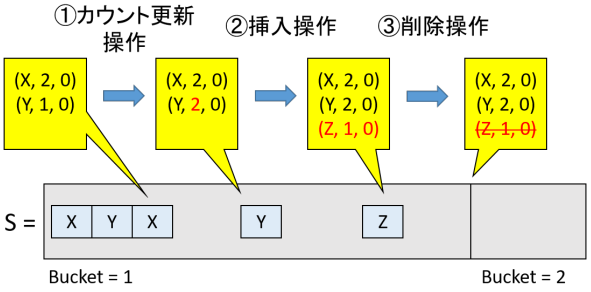

図 1　Lossy Counting の例

例えば図 1 の場合，① では要素 Y が既に格納されているため，Y の頻度をインクリメントする（カウント更新操作）．② では，新しい要素 Z が出現したため，シノプス $(Z, 1, 1-1=0)$ を格納する（挿入操作）．③ では，要素 Z が $1 \leqq 1-0=1$ より $f \leqq b_{current} - \Delta$ を満たすため削除する（削除操作）．

2.2 Space Saving

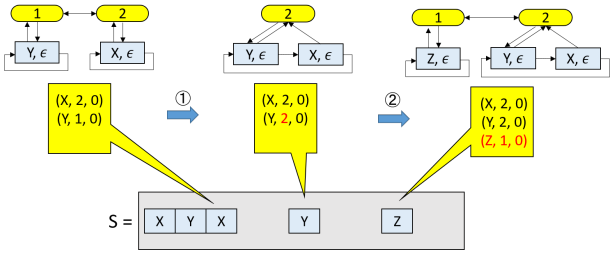

図 2　Stream Summary の例

Space Saving では Stream-Summary と呼ばれるデータ構造を使用して頻出アイテムを保持する．Stream-Summary とは図 2 のようなデータ構造である．頻度を格納したリスト（頻度リスト）とリストの各要素とリンクしたサブセット $< ID, \epsilon >$ で構成される．ϵ とは頻度の最大誤差を示す．Stream-Summary では各操作を要求された際に以下の内部処理を行う．

カウント更新操作 $(< I >, val)$：対象アイテム I のサブセットを更新後の頻度 val の要素にリンクさせる．更新後の頻度の要素がない場合は新たに頻度リストに追加する．
挿入操作 $(< I, \epsilon >, val)$：サブセット $< I, \epsilon >$ を追加し，頻度 val の要素にリンクさせる．頻度 val の要素がない場合は新たに頻度リストに追加する．
除去操作 $(< I >)$：対象アイテム I のサブセットを削除する．

例えば図 2 の場合，① はカウント更新操作 $(< Y >, 2)$ に対応し，② は挿入操作 $(< Z, 0 >, 1)$ に対応する．

アルゴリズム

Space Saving では頻度が TopK 番目までのアイテムを保持

する．データストリームに新しいアイテム I が出現したとき，以下の操作を行う．

$(N < K+1)$ の場合：挿入操作 $(< I, 0 >, 1)$ を行う．
$(N = K+1)$ の場合：頻度が一番低いアイテムのサブセット $(< I', \epsilon >, val)$ に対して削除操作 $(< I' >)$ を行い，挿入操作 $(< I, \epsilon + 1 >, val + 1)$ を行う．

ここで N はデータ構造に格納しているアイテムの種類数である．

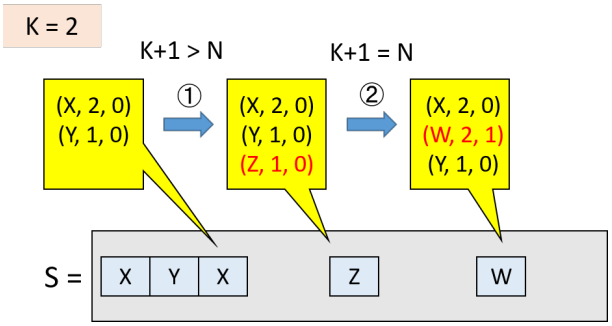

図 3　Space Saving の例

例えば図 3 の場合，① では $(N < K+1)$ を満たしているため，要素 Z を頻度 $= 1$，$\epsilon = 0$ で挿入する．② では $(N = K+1)$ なので，頻度が最も低い要素 Z に対して削除操作を行い，新たに出現した要素 W を頻度 $= 1+1 = 2$，$\epsilon = 0+1 = 1$ で挿入する．

3　KeyGraph

KeyGraph とは，文書から主張と概要を表すキーワードを抽出する手法である．抽出されたキーワードは文書検索や要約に用いることを想定している．各語をノード，語と語の共起をリンクで表し，図 4 のようなグラフとして表現する．

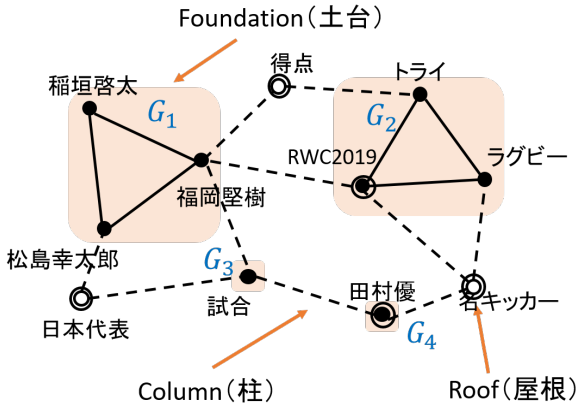

図 4　キーグラフの例

KeyGraph は TF*IDF に匹敵する高速なアルゴリズムであるため，リアルタイムでの応答が求められるシステムにおいても適用することが可能である．

KeyGraph では，文章中で繰り返し出現する高頻度語を土台と定義し，土台によって説明される主張となる語を屋根と定義する．また，土台と文書中の各語の共起関係の強さを柱と定義する．土台とは，文書を展開していく上で基本となる概念である．屋根とは筆者が伝えたい主張を指しており，多くの土台との柱に支えられている語を屋根として抽出する．

3.1　KeyGraph アルゴリズム

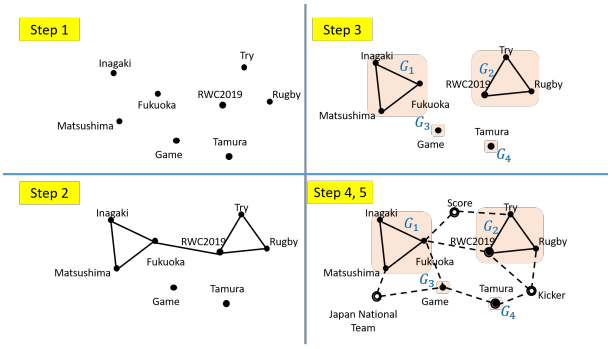

図 5　KeyGraph アルゴリズム

以下では KeyGraph のアルゴリズムについて述べる．

Step1: 土台語の抽出

文書中の単語の出現頻度を算出し，頻出語上位 M 語を土台語とする．図 5 では黒いノードにあたる．

Step2: 土台語のクラスタの生成

土台語同士の共起度 Co を算出し，共起度上位 M-1 番目までの単語のペアにリンクを張る．共起度 Co については以下の式によって定義する．

$$Co(w_i, w_j) = \sum_{s \in D} min(|w_i|_s, |w_j|_s) \tag{1}$$

ここで，$|w_i|_s$，$|w_j|_s$ は文書 D 内の文章 s における単語 w_i，w_j それぞれの出現回数を示す．

Step3: 土台の抽出

Step2 で張ったリンクのうち，強連結を形成するリンク以外を削除する．強連結とは対となるノード w_i，w_j において w_i，w_j 間のリンクを削除しても他のノードを経由することによって他方のノードに到達できる場合のことを指す．Step1 から Step3 の手順を経てできた極大連結部分をそれぞれ 1 つのクラスタとして抽出する．

図 5 では「Fukuoka」と [RWC2019」間のリンクを削除した．なお，ノード「田村優 (Tamura)」や「試合（Game)」のような他のどのノードともリンクが張られていないノードにおいては，それ自身が 1 つのクラスタを形成する．

Step4: 柱の抽出

文書内の全ての単語と土台の共起度 Based を算出する．共起度 Based については以下の式によって定義する．

$$Based(w, g) = \sum_{s \in D} |w|_s |g - w|_s \tag{2}$$

$$|g - w|_s = \begin{cases} |g|_s - |w|_s & if\ w \in g \\ |g|_s & if\ w \notin g \end{cases} \tag{3}$$

ここで $|g - w|_s$ は文章 s に出現した土台語 g の出現回数を示

す．ただし，w が土台語に含まれる場合は w 以外の g との共起回数とする（式 (3)）．図 4 のノード「ラグビー」と土台 G_2 の場合，「ラグビー」と「トライ」または「RWC2019」の共起回数となる．

Step5: 屋根の抽出

ノード w が全ての土台から支えられている力を $key(w)$ とする．$key(w)$ はいずれかの土台と共起している条件付き確率で定義され，以下の式で算出される．

$$Neighbors(g) = \sum_{s \in D} \sum_{w \in s} |w|_s |g - w|_s \tag{4}$$

$$key(w) = 1 - \prod_{g \subset G} \left(1 - \frac{Based(w,g)}{Neighbors(g)}\right) \tag{5}$$

key 値上位 R 語を屋根として抽出する．この屋根こそが筆者の主張を示すキーワードとなる．

ここで注意すべき点は，土台語もキーワードになりえる点である．図 4 では「RWC2019」と「田村優」が土台語であると同時にキーワードとしても抽出された．

3.2 連続 KeyGraph

連続 KeyGraph とは，KeyGraph をデータストリームに適用したアルゴリズムである．即ち，時間の経過に伴って KeyGraph を更新する手法である．

従来の KeyGraph では Step1，Step2，Step4 にて文書を読み込む必要がある．しかし，データストリームにおいて複数回データを読み込むことは望まれない．連続 KeyGraph では KeyGraph の生成に必要な情報として以下のシノプスを保持する．

- 単語シノプス $S_t(w, f, thr, g)$
- 共起シノプス $S_c(S_{t1}, S_{t2}, cf, thr_c)$
- 柱シノプス $S_{col}(S_t, g, gf)$

単語シノプスは，単語 w，w の見積もり頻度 f，f の閾値 thr，所属する土台 IDg(無所属なら-1) の集合である．また，共起シノプスは 2 つの単語シノプスのポインタ S_{t1}，S_{t2}，S_{t1} と S_{t2} の見積もり共起頻度 cf，cf の閾値 thr_c の集合であり，柱シノプスは単語シノプスのポインタ S_t，土台 IDg，S_t と土台 g の見積もり共起頻度の集合である．これらのシノプスを常に保持することによって，1 回の読み込みで KeyGraph の生成を可能とする．

従来の KeyGraph では，単語のペアが同じ文章に出現することを共起とし，1 つの文書に対し 1 つのキーグラフを生成する．提案手法では，特定の話題について言及している複数の投稿文を 1 つの文書と見なし，単語ペアが同一投稿文に出現することを共起ととらえる．

連続 KeyGraph では投稿文を一つずつ読んでいき，Lossy Count アルゴリズムに従って各シノプスを更新する．以下では KeyGraph の更新方法について述べる．

Step1: 土台語の抽出

$f \geq (\gamma - \epsilon)N$ を満たす単語を土台語とする．$g \neq -1$ である単語シノプス S_t の w の集合（前回の土台語）と抽出された土台語を比較し，変化があった場合は Step2，ない場合は Step4 へ進む．また，初回の場合は Step2 へ進む．

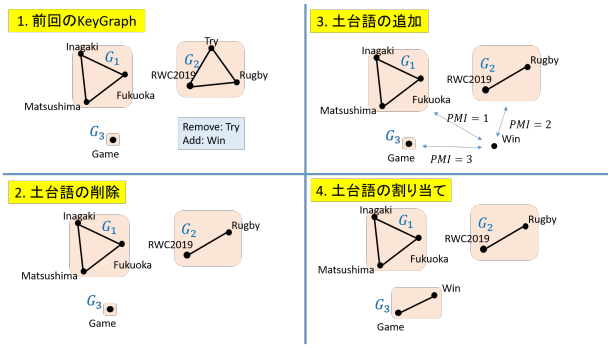

図 6　連続 KeyGraph における土台クラスタの割り当て

Step2: 土台クラスタの割り当て

土台語でなくなった単語を土台クラスタから外す（S_t の $g = -1$）．図 6 では単語「Try」を土台 G_2 から外す．また，新たな土台語と既存の土台との共起度 C を共起シノプスを用いて算出する．C は自己相互情報量（PMI）の平均として以下の式で定義する．

$$C(w,g) = \frac{1}{N_g} \sum_{k=1}^{N_g} \log_2 \frac{p(w,g_k)}{p(w)p(g_k)} \tag{6}$$

ここで N_g は土台 g に所属する単語の数，g_k は土台 g に所属する単語を示す．C の値が最も高い土台に新たな土台を所属する．ただし，最も高い C が閾値 Thr_{new} を下回る場合，新しい土台クラスタを生成する．図 6 では単語「Win」を PMI の平均値が最も高かった土台 G_3 に追加する．Step3 へ進む．

Step3: 柱シノプスの再計算

所属する単語が変化した土台に関する柱シノプス S_col の gf を共起シノプスを用いて再計算する．以下の式を用いる．

$$gf(w,g) = \frac{1}{N_g} \sum_{k=1}^{N_g} cf(w,g_k) \tag{7}$$

Step4 へ進む．

Step4: 屋根の抽出

柱シノプスを用いて $key(w)$ を算出する．以下の式を用いる．

$$Neighbors(g) = \sum_{g \in G} gf(w,g) \tag{8}$$

$$key(w) = 1 - \prod_{g \subset G} \left(1 - \frac{gf(w,g)}{Neighbors(g)}\right) \tag{9}$$

key 値上位 R 語を屋根として抽出する．

4 提 案 手 法

本研究では，連続 KeyGraph の改良を試みる．

従来の連続 KeyGraph では，KeyGraph の生成に必要な 3 つの統計量 (単語頻度，共起頻度，単語と土台の共起頻度) を Lossy Count によって保持している．Lossy Count を適用することによって，KeyGraph の生成にあまり影響を与えない低頻

度語を削除し，統計量の保持に必要なメモリ量と実行時間を短縮することができる．低頻度語の閾値はパラメータ ϵ によって調整することが可能で，ϵ の値が高いほど多くの低頻度語を除外し，必要メモリ量を削減する．しかし，ϵ の値を高くし過ぎると，Lossy Count における削除操作の実行回数が多くなり，全体としての実行時間が長くなる．即ち，実用性を考慮すると削減できるメモリ量には限りがある．

本研究では，3 つの統計量を Space Saving によって保持する．低頻度の閾値を出現頻度のランクで管理することにより，統計量の保管に必要なメモリ量の上限を指定することが可能となり，より少ないメモリで連続 KeyGraph を実行することが可能である．

4.1 データ構造とパラメータ

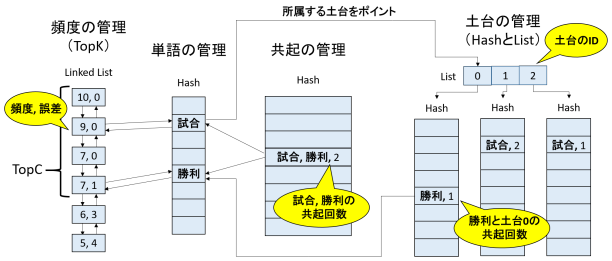

図 7　提案手法のデータ構造

図 7 に提案手法のデータ構造を示す．本研究では 7 つのパラメータ K，C，M，r，T_{reduce}，T_{output}，Thr_{new} によって Space Saving を制御する．K は TopK の K に対応し，頻度上位 K 番目までの単語を保持する．S は Space Saving における信頼区間を示す．アルゴリズムの仕組みを考慮すると，ランクが下位の要素ほど頻度の誤差が大きくなると考えられる．そのため，TopK のうちある程度信頼できる区間 TopC を指定する．TopC 内の単語同士の共起頻度と，単語と土台の共起頻度を Hash に格納し，単語が TopK から除外されると，その単語に関する共起頻度も削除する．M は土台語の閾値に対応する．即ち，TopM の単語を土台語とする．

r，T_{reduce} は減少操作に関するパラメータである．減少操作が要求されると，全ての頻度 f に対して $f = f \times r$ を行う．$r(0 \leqq r \leqq 1)$ は減少係数であり，r の値が 0 に近いほど最新のデータを重要視する．T_{reduce} は減少操作を行う時間間隔である．例えば $T_{reduce} = 10$ 分の場合，10 分毎に減少操作を実行する．

T_{output} は KeyGraph の更新の時間間隔である．Thr_{new} は連続 KeyGraph の Step2 における土台クラスタの閾値を示す．

5 実　　験

本研究では，提案手法における最適なパラメータを求めるための 2 つの実験と，従来の連続 KeyGraph との性能比較実験の計 3 種類の実験を行う．なお，本実験で求めるパラメータはあくまで今回実験に用いたデータに最適化された値であり，全てのデータにおいて最高のパフォーマンスを示すものではない．

5.1 実 験 環 境

まずは，本実験で用いるコーパスについて説明する．本実験では 2019 年 12 月 22 日に放送されたテレビ番組「M-1 グランプリ 2019」に関する tweet 文を対象とする．この tweet 文は twitterAPI を用いて，キーワード検索によって指定期間内の指定キーワードを含む全ツイート文を収集されたコーパスである（ただし RT 文は除外する）．選定したキーワードは，「#m1 グランプリ」と番組内で漫才を披露した 10 組のコンビ名を用いた（「すゑひろがりず」については表記ゆれを考慮し「すゑひろがりず」と「すえひろがりず」の両方をキーワードとした）．詳細な情報を表 1, 表 2, 図 8，図 9 に示す．

表 1　ツイートの取得期間とツイート数

取得期間	ツイート数
2019/12/22/18:00 - 2019/12/22/23:59	750540

表 2　ツイートの取得期間の詳細

取得期間	ツイート数
18:00 - 18:59	21267
19:00 - 19:59	154905
20:00 - 20:59	176653
21:00 - 21:59	236104
22:00 - 22:59	141598
23:00 - 23:59	20013

#m1 グランプリ，インディアンズ，オズワルド，かまいたち，からし蓮根，すゑひろがりず，すえひろがりず，ニューヨーク，ぺこぱ，見取り図，ミルクボーイ，和牛

図 8　取得キーワード

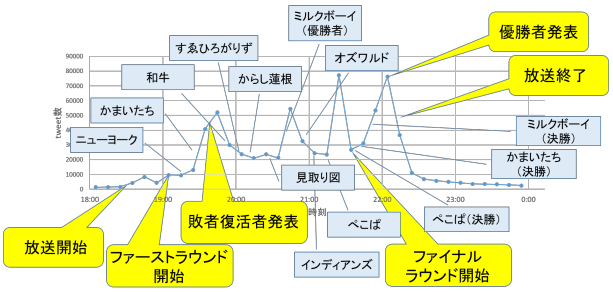

図 9　期間内の主なイベントとツイート数の変動

収集したコーパスからテキスト情報と時間情報を抽出する．文書に対して MeCab により形態素解析を行う．システム辞書には「mecab-ipadic-NEologd」を使用する．また，形態素解析を SNS に対応させるためにいくつかの前処理を行う [10]

不要語については，私たちが提案した不要語フィルターによる抽出法を適用する [10]．以下の指標を用いて不要語を抽出し，不要語を除去したコーパスで実験を行う．

$$TFIG(w) = \log_2 \frac{TF(w)}{IG(w)} \tag{10}$$

ここで，$TF(w)$ は語 w の出現頻度，$IG(w)$ は語 w の情報利得を示す．

5.2 実験 1: 信頼区間 C の推定

Space Saving アルゴリズムにおいて，TopK に対して上位 X%における各数値の信頼性を示す．コーパスに対して通常のカウント（正解データ）と Space Saving によるカウントを行い，正解データに対して TopK のうちの上位 X%までの精度を調査する．精度の評価として，F 値，ケンドール順位相関係数，カウント誤差（絶対値）の平均値を用いる．K の値を 100 から 10000 まで 100 ずつ変化させて実験を行い，全ての結果の平均値を表 3 に示す．

表 3　上位 X%における信頼性の比較

上位 X%	F 値	ケンドール順位相関係数	平均カウント誤差
10%	0.998685009	0.999005519	0.184
20%	0.987580433	0.981243371	0.968
30%	0.935132962	0.910355212	2.136
40%	0.795489263	0.719516282	4.215
50%	0.50259204	0.41298427	8.811
60%	0.203247581	0.272736026	21.842
70%	0.088947643	0.343494293	34.402
80%	0.062666378	0.401125694	44.128
90%	0.053874728	0.458356181	43.758
100%	0.048805187	0.488848623	42.603

結果より，ランクが下位の単語ほど誤差が大きくなることが確認された．以降の実験では，上位 20%において F 値とケンドール順位相関係数がともに 0.95 を上回ったため，$C = K \times 0.2$ とする．

5.3 実験 2: 最適な K の推定

提案手法において，最も性能が高くなる K の値を実験で確かめる．性能の評価として，実行時間，キーワードの F 値，最大保持シノプス数を用いる．各評価の定義は以下に示す．

実行時間：コーパスを読み込み始めてから最後の KeyGraph の更新が終わるまでの時間

キーワードの F 値：T_{output} 分毎に KeyGraph を更新し，各時点で得られたキーワードと正解キーワードとの F 値を求める．全ての時点での F 値の平均値を評価値とする

最大保持シノプス数：アルゴリズム実行中に保持していた単語頻度と共起頻度の要素数の和の最大値

ここで，正解キーワードは通常の KeyGraph を実行して得られたキーワードとする．連続 KeyGraph では T_{output} 分毎に KeyGraph を更新するのに対し，通常の Keygraph ではコーパスの最初から出力する時点までのコーパスを読み込んで生成する．また，連続 KeyGraph と同様に通常の KeyGraph でも減少操作を行う．最大保持シノプス数は必要メモリ数に匹敵する．

連続 KeyGraph のパラメータは $C = 0.2K$，$M = 20$，$r = 0.5$，$T_{reduce} = 10$ 分，$T_{output} = 10$ 分，$Thr_{new} = 0.5$ とする．K の値を 500 から 1500 まで 50 づつ変化させて実験を行った．結果を表 4 に示す．

結果より，$K = 1050$ のときに F 値が最も高くなり，かつ同様の F 値が得られた結果の中で最小限の実行時間と最大保持シノプス数を示した．以降の実験では $K = 1050$ を採用する．

表 4　各 K における性能の比較

K	実行時間 [ms]	キーワードの F 値	最大保持シノプス数
500	29774	0.848063328	25030
550	25825	0.848301566	19374
600	25576	0.848301566	19316
650	26135	0.84909183	19618
700	28418	0.850144018	20501
750	25847	0.850144018	22476
800	26634	0.850144018	23011
850	26556	0.847437722	23716
900	25960	0.847437722	24834
950	26096	0.846385533	25508
1000	27871	0.846385533	26103
1050	25424	0.854944672	28256
1100	26863	0.854944672	28691
1150	27627	0.854944672	30286
1200	28005	0.854944672	31197
1250	28197	0.849380666	33261
1300	29308	0.849380666	32884
1350	28350	0.847394771	34827
1400	28006	0.847524547	35261
1450	28066	0.847524547	36097
1500	28331	0.847524547	37379

5.4 実験 3: 各手法との比較実験

通常の KeyGraph，通常のカウント（Lossy Count や Space Saving を用いない）による連続 KeyGraph，Lossy Count による連続 KeyGraph（ベースライン）と Space Saving による連続 KeyGraph（提案手法）の性能を比較する．ベースラインでのパラメータは元論文で採用された値を使用する．通常の KeyGraph における実行時間は，実験 2 で述べた方法で各時点でのキーワードを抽出し，全ての時点でのキーワードを抽出するまでの累計時間を採用する．結果を表 5 に示す．

表 5　各手法との比較

K%	実行時間 [ms]	キーワードの F 値	最大保持シノプス数
通常 KeyGraph	372012	1	55721
通常カウント	51388	0.843828176	3036231
Lossy Count	35139	0.864640974	289524
Space Saving	25424	0.854944672	28256

この結果から，提案手法はベースラインと比較して，実行時間が 27.6%，最大保持シノプス数は 90.2%削減された．しかし，キーワードの F 値は 1.2%減少した．

5.5 考　　察

連続 KeyGraph で得られたキーワードについて確認する．図 9 で挙げたイベントのうち,「敗者復活者発表」(19:45),「優勝者発表」(22:05) をそれぞれ含んだ 19:50 時点と 22:10 時点にて実際に抽出されたキーワードを表 6，表 7 に示す．

表 6　敗者復活者発表のキーワード

key 値上位 R 語	キーワード
1-5	#M1 グランプリ，敗者復活，和牛，#M1，順位予想
6-10	三連単，U+0001F62D，かまいたち，きた，私
11-15	投票，2 位，1 位，予想，3 位
16-20	M-1 グランプリ，キャンペーン，あなた，も GYAO，サイト

表 7　優勝者発表のキーワード

key 値上位 R 語	キーワード
1-5	ミルクボーイ，かまいたち，#M1 グランプリ，面白い，ぺこぱ
6-10	優勝，おめでとう，和牛，なぁ，好き
11-15	U+0001F3FB，見る，U+0001F62D，ネタ，を
16-20	なー，ほしい，漫才，すごい，目

敗者復活者発表時では,「敗者復活」がキーワードとなった他,敗者復活者に選ばれた「和牛」がキーワードとなった．優勝者発表時では,「優勝」や優勝者である「ミルクボーイ」,「おめでとう」等がキーワードとなった．

6 結　　論

本研究では，連続 KeyGraph に Space Saving アルゴリズムを適用することにより，実行時間や必要メモリ量において改善を試みた．実験では，提案した手法が従来の手法と比較し，キーワードの F 値が 1.2%減少した代わりに，実行時間が 27.6%，使用メモリ量に相当する最大保持シノプス数は 90.2%削減することを示した．

文　　献

[1] Mateusz Fedoryszak, Vijay Rajaram, Brent Frederick, Changtao Zhong, "Real-time Event Detection on Social Data Streams", Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD), 2019

[2] Yukio Ohsawa, Nels E. Benson, Masahiko Yachida, "KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor", Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries(ADL), 1998

[3] Yuta Nezu, Taka Miura, "Extracting Keywords on SNS by Successive KeyGraph", 32th International Conference on Tools with Artificial Intelligence(ICTAI), 2020

[4] 有村 博紀, 喜田 拓也, "データストリームのためのマイニング技術", 情報処理, Vol46 No1 p4-p11, 2005

[5] Jizhou Li, Zikun Li, Yifei Xu, Shiqi Jiang, Tong Yang, Bin Cui, Yafei Dai, Gong Zhang, "WavingSketch: An Unbiased and Generic Sketch for Finding Top-k Items in Data Streams", Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD), 2020

[6] Graham Cormode, S Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications", Journal of Algorithms, 55(1), 2005.

[7] Amit Goyal, Jagadeesh Jagarlamudi, Hal Daum III, Suresh Venkatasubramanian, "Sketching Techniques for Large Scale NLP", WAC@NAACL-HLT,p17-p25, 2010

[8] Gurmeet Singh Manku, Rajeev Motwani, "Approximate Frequency Counts over Data Streams", Proceedings of the 28th international conference on Very Large Data Bases(VLDB), 2002

[9] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. "Efficient computation of frequent and top-k elements in data streams", In International Conference on Database Theory. Springer, 2005.

[10] Yuta Nezu, Taka Miura, "Statistical Processing of Stopwords on SNS", 30th International Conference on Database and Expert Systems Applications(DEXA), 2019