連続 KeyGraph によるソーシャルネットワーキングサービス上の要約

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.j

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 SNS では，テキストの更新が頻繁に起こるため, 膨大な量のデータを高速に処理する必要がある．そのため，実行時間が短く使用するメモリが少ないアルゴリズムが望ましい．本研究では，Lossy Counting によって頻出項目を保持し，KeyGraph を用いてニュースを要約する連続 KeyGraph を提案する．KeyGraph とは筆者の主張の抽出を目的とした手法であり高速に結果を出力することが保証されている．また，Lossy Counting を用いることによって,KeyGraph の計算過程で必要となる単語間の共起情報を最低限のみ保持する．さらに，SNS 特有の未知語や不要語に対応するための事前処理を行い, 形態素解析の精度向上を図る．

キーワード　ストリームデータ，ソーシャルネットワーキングサービス，KeyGraph，Lossy Counting，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，SNS では日々大量の投稿がなされており，Twitter に関しては一日に平均して 5 億件の tweet が発信されているとの報告がある [1]．そのため，実行時間が短く使用するメモリが少ないアルゴリズムが望ましい．

文書から話題の大筋をとらえるためのキーワード抽出法として KeyGraph がある [3]．KeyGraph とは筆者の主張の抽出を目的とした手法であり高速に結果を出力することが保証されている．輪島らの研究では [2]，KeyGarph の土台に LDA のトピックを用いることにより土台生成の改善を行い SNS に適用した．

本研究では，Lossy Counting [4] によって頻出項目を保持し，KeyGraph を用いてニュースを要約する連続 KeyGraph を提案する．KeyGraph をオンライン化することにより SNS への適用を可能とする．また Lossy Counting を用いることによって,KeyGraph の計算過程で必要となる単語間の共起情報を最低限のみ保持する．さらに，SNS 特有の未知語や不要語に対応するための事前処理を行い, 形態素解析の精度向上を図る．

2 章では KeyGraph の概要と具体的なアルゴリズムについて述べ，3 章では Lossy Counting について述べる．4 章では提案手法である連続 KeyGraph について述べ，5 章で実験を行う．最後に 6 章で結論を述べる．

2　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 1 のようなグラフとして表現する．

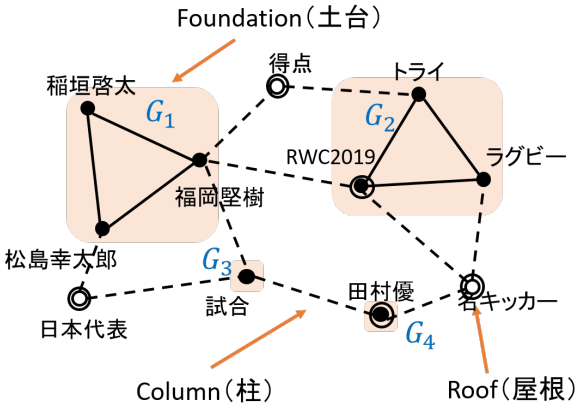

図 1　キーグラフの例

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．

2.1 KeyGraph のアルゴリズム

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 1 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．なお，図 1 のノード「田村優」や「試合」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 1 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 1 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 Lossy Counting

大規模なコーパスでは，一部の高頻度な単語と大量の低頻度な単語で文書が構成されることが多い．低頻度な単語はデータの保持に使用するメモリの大半を占める．しかしながらデータマイニングにおいては，よく出現するアイテムほど重要であり，低頻度なアイテムは無視しても影響が少ない．Manku らはストリームデータにおけるアイテムの頻度測定の手法として Lossy Counting を提案した [4] [5]．Lossy Counting 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項目を保持する．Lossy Counting 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．

アルゴリズム

Lossy Counting 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．ここで N は要素数である．

例えば図 2 の場合，$b_{current} = 1$ から $b_{current} = 2$ のバケットに移るとき，要素「野球」は $f = 1 \leqq b_{current} - \Delta = 1 - 0 = 1$ のため除去される．同様に，$b_{current} = 3$ のバケットに入る際に「サッカー」「バスケ」が除去される．$b_{current} = 3$ の時点で出力が要求されたとき，$N = 15$，$(\gamma - \epsilon)N = (0.4 - 0.2) = 3$ なので，見積もり頻度が 3 以上である「ラグビー」「野球」が出力される．

4 連続 KeyGraph

本研究では，SNS のテキストに KeyGraph を適用し，SNS データの更新に伴って KeyGraph を更新する連続 KeyGraph を提案する．即ち，KeyGraph のオンライン化を試みる．

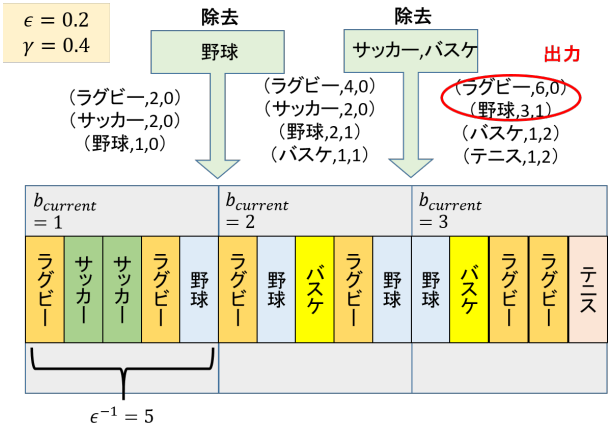

図 2　Lossy Counting の例

従来の KeyGraph では，土台抽出の際に文書から共起頻度を抽出し，柱を抽出する時に再度文書内を探索する．つまり，キーグラフを作成する際に少なくとも 2 回以上文書を読み込む必要がある．そのため，ストリームデータに適用することが困難である．

連続 KeyGraph では単語の頻度と単語同士の共起頻度，土台と単語の共起頻度を保持することによりキーグラフの更新を可能にする．また，Lossy Counting を単語頻度と共起頻度に適用することによりメモリ使用量を抑え，実行時間を短縮する．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．

4.1　Lossy Counting との連動

連続 KeyGraph では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．

連続 KeyGraph では投稿文を一つずつ読んでいき，各シノプスを更新する．更新の操作は以下のとおりである．

1. カウント更新操作：
単語シノプス：読んだ要素 w に対するシノプス S_t がすでにある場合，$f++$
共起シノプス：読んだ要素のペア w_1，w_2 に対するシノプス S_c がすでにある場合，$cf++$
柱シノプス：読んだ要素 w と土台 g に対するシノプス S_col がすでにある場合，$gf++$

2. 挿入操作：
単語シノプス：読んだ要素 e に対するシノプス S_t がない場合，新しいシノプス $(w, 1, 1, -1)$ を作成
共起シノプス：読んだ要素のペア w_1，w_2 に対するシノプス S_c がない場合，新しいシノプス $(S_{t1}, S_{t2}, 1, 1)$ を作成
柱シノプス：読んだ要素 w と土台 g に対するシノプス S_{col} がない場合，新しいシノプス $(S_t, 1)$ を作成

3. 除去操作：新しいバケットに入るとき，
単語シノプス：$f \leqq thr$ を満たすシノプス S_t を消去し，全てのシノプスに対し $thr++$
共起シノプス：上記の操作により対応する S_{t1} または S_{t2} が消去されたシノプス S_c を消去する．また $cf \leqq thr_c$ を満たすシノプス S_t を消去し，全てのシノプスに対し thr_c++
柱シノプス：上記の操作により対応する S_t が消去されたシノプス S_col を消去する

4. 減少操作：T_{reduce} 期間毎に，
単語シノプス：全てのシノプスに対し，$f = r * f$，$thr = r * thr$
共起シノプス：全てのシノプスに対し，$cf = r * cf$，$thr_c = r * thr_c$
柱シノプス：全てのシノプスに対し，$gf = r * gf$
N=r*N

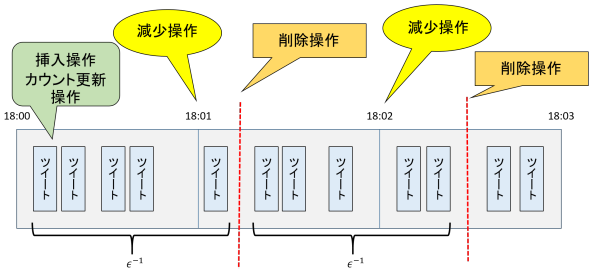

図 3　Lossy Counting の各操作のタイミング

ここで，$r(0 \leqq r \leqq 1)$ 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．また 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 分毎に減少操作を行う．各操作のタイミングについて図**??**にまとめる．

4.2　KeyGraph の更新

本稿では連続 KeyGraph のアルゴリズムについて述べる．キーグラフを更新するタイミングは更新間隔 $T_{interval}$ で制御する．例えば $T_{interval} = 10$ 分の場合，10 分毎にキーグラフを更新する．

Step1: 不要語の除去
SNS では新聞記事などの形式が整った文書と異なり，表現が多様で単語分布の変動が激しいため，SNS コーパスに対応した動的な不要語を用いる方がよい．本稿で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 [6]．以下の指標を用いて

不要語を抽出し，不要語以外の語について以下のアルゴリズムを適用する．

$$TFIG(w_i) = \log_2 \frac{tf(w_i)}{G(w_i)} \tag{6}$$

Step2: 土台語の抽出

$f \geqq (\gamma - \epsilon)N$ 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 Step3，ない場合は Step5 へ進む．また，初回の場合は Step3 へ進む．

Step3: 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 $g = -1$）．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．

$$C(w,g) = \frac{1}{N_g} \sum_{k=1}^{N_g} \log_2 \frac{p(w,g_k)}{p(w)p(g_k)} \tag{7}$$

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．Step4 へ進む．

Step4: 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．

$$gf(w,g) = \frac{1}{N_g} \sum_{k=1}^{N_g} cf(w,g_k) \tag{8}$$

Step5 へ進む．

Step5: 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．

$$Neighbors(g) = \sum_{g \in G} gf(w,g) \tag{9}$$

$$key(w) = 1 - \prod_{g \subset G} \left(1 - \frac{gf(w,g)}{Neighbors(g)}\right) \tag{10}$$

key 値上位 R 語を屋根として抽出する．

4.3 従来の KeyGraph との比較

従来の KeyGraph との対応について述べる．従来の KeyGraph における Step1 では土台語の抽出を行う．提案手法では Step2 に対応する．従来では土台語の数を定数個抽出したのに対し，提案手法では Lossy Counting に従って閾値以上の語を抽出する．

従来の KeyGraph では Step2，Step3 にかけて土台の抽出を行うが，このステップが提案手法との一番の違いである．従来では共起度 Co の高い順にリンクを張り，極大連結部分を土台として抽出する．対して提案手法では PMI を基準として最も共起度の高いクラスタに土台語を振り分けていく．この方法を用いることにより土台の更新を可能にする．

Step4 での柱の抽出では，従来法では文書を探索し共起をカウントするのに対し，提案手法では予め保持していた共起情報から近似する．

Step5 での屋根の抽出に関しては，KeyGraph のアイデアの心臓部にあたるため，ほとんど同じ計算を行う．

5 実　　験

本章では，オフラインにおける KeyGraph をベースラインとし，再現率とメモリ使用量，実行時間において連続 KeyGraph と比較し，提案手法の有用性を実験により示す．

5.1 実験環境

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 4，図 5 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 4　取得キーワード

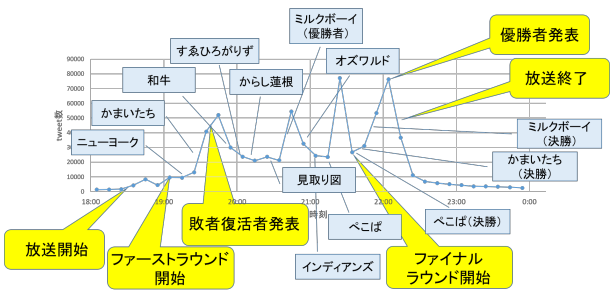

図 5　期間内の主なイベントとツイート数の変動

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 [6]．

5.2 実験結果

コーパスを $T_{interval} = 1$ 分毎に区切り，各区間で独立にキーグラフの生成を行う．これらの結果をオフラインと定義する（図 6）．

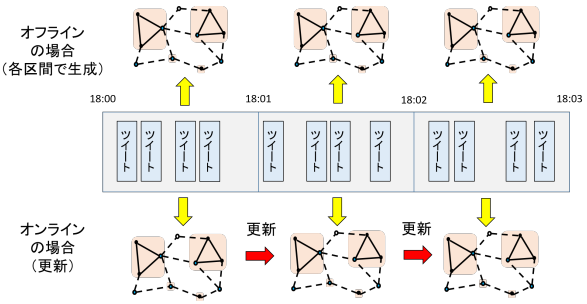

図 6　オフラインとオンラインの違い

各パラメータは $\gamma = 0.05$，$Thr_{new} = 1$，不要語の閾値 $= 1.5$ とし，これらのパラメータについては他の実験でも変化させずに行う．

はじめに，Lossy Counting の有効性を示すための実験を行う．$T_{interval} = 10$ 分，$T_{reduce} = 10$ 分，$r = 0.5$ とし，ϵ の値を変化させて実験を行う．各時刻毎に抽出されたキーワードの再現率を算出し，その平均値を評価する．また，$R = 5, 10, 15, 20$ それぞれの場合で実験を行う．データを読み込み始めたばかりである最初の 1 時間は評価対象外とし，19:10 から 23:59 までの出力を扱う．なお，再現率は以下の式で定義する．

$$Recall = \frac{key_{offline} \cap key_{online}}{key_{offline}} \tag{11}$$

ここで，$key_{offline}$ はオフライン KeyGraph によって抽出されたキーワードの数，$key_{offline} \cap key_{online}$ は提案手法とオフライン KeyGraph の両方で抽出されたキーワードの数である．再現率，保持している単語シノプスと共起シノプスの数と実行時間（平均値）を表 3，表 4 に示す．

表 3　各 ϵ における再現率

ϵ	R=5	R=10	R=15	R=20
0.0001	0.78	0.803333333	0.766666667	0.741666667
0.0002	0.78	0.803333333	0.766666667	0.736666667
0.0005	0.766666667	0.806666667	0.764444444	0.731666667
0.001	0.76	0.79	0.757777778	0.715
0.002	0.766666667	0.806666667	0.773333333	0.73
0.005	0.713333333	0.8	0.748888889	0.735
0.01	0.693333333	0.756666667	0.728888889	0.728333333

基本的に ϵ が高いほどオフラインに対する再現率の低下が確認された．また，保持するシノプス数においては大幅に減少した．具体的には，共起シノプスにおいて $\epsilon = 0.001$ のときは

表 4　各 ϵ におけるシノプス数と実行時間の比較

ϵ	単語シノプス	共起シノプス	実行時間
オフライン	10493.43333	524602.1	17572.2
0.0001	7657.433333	362508.6667	10981.63333
0.0002	5301.366667	220129.1	8503.7
0.0005	3305.733333	117475.1667	5978.666667
0.001	2260.066667	71165.6	5307.6
0.002	1412.633333	38214.56667	5021.733333
0.005	794.7666667	18439.33333	7601.266667
0.01	480.6	9801.766667	10347.53333

13.6%，$\epsilon = 0.0001$ のときには 1.9%にまで削減できた．これにより使用するメモリを大幅に抑えることが確認できた．一方実行時間においては，$\epsilon = 0002$ のときが最も短く，それ以降は ϵ が高くなるにつれて実行時間が長くなった．これはおそらく ϵ が高くなるにつれて除去操作を行う回数が増えたことが原因だと思われる．

次に，減少係数 r についての実験を行う．$T_{interval} = 1$ 分，$T_{reduce} = 1$ 分，$\epsilon = 0.001$ とし，r の値を変化させて実験を行う．データを読み込み始めたばかりである最初の 1 時間は評価対象外とし，19:01 から 23:59 までの出力を扱う．結果を表 5，6 に示す．

表 5　各 r における再現率

r	R=5	R=10	R=15	R=20
0.9	0.604	0.675666667	0.642888889	0.6405
0.8	0.675333333	0.709666667	0.689333333	0.690666667
0.7	0.683333333	0.733333333	0.726	0.726666667
0.6	0.718	0.749666667	0.748222222	0.751
0.5	0.712666667	0.772	0.766888889	0.771333333
0.4	0.764	0.791666667	0.786666667	0.7925
0.3	0.768	0.796666667	0.794	0.803666667
0.2	0.791333333	0.807666667	0.800888889	0.810333333
0.1	0.777333333	0.81	0.809333333	0.815333333

表 6　各 r におけるシノプス数と実行時間の比較

r	単語シノプス	共起シノプス	実行時間
オフライン	2805.87	89156.52333	2582.193333
0.9	2132.66	64505.46667	1538.253333
0.8	2086.19	62042.93333	1397.35
0.7	2055.93	60656.54667	1353.426667
0.6	2033.26	59772.74667	2487.316667
0.5	2009.636667	58746.62667	1304.3
0.4	1999.746667	58365.51667	1666.44
0.3	1989.766667	57930.82333	1893.636667
0.2	1984.21	57688.09	1470.16
0.1	1981.45	57545.87	1886.306667

r が低いほど最新の情報を重要視するため，再現率が高くなることが確認できた．一方，r が低いほどシノプスの数が減少傾向にあるにも関わらず，実行時間は単調減少ではない．これは，r が低いほど土台語の変動が激しく，柱シノプスの生成に

時間がかかったためだと思われる．表 7 はコーパスを最後まで読み込んだ過程で発生した土台クラスタの延べ数である．途中で消滅したクラスタもカウントに含まれている点に注意してもらいたい．

表 7　各 r におけるクラスタの延べ数

r	土台の数
0.9	25
0.8	35
0.7	48
0.6	61
0.5	73
0.4	77
0.3	93
0.2	100
0.1	125

5.3 考　　察

図 5 で挙げたイベントのうち,「ファーストラウンド開始」(19:12),「敗者復活者発表」(19:45),「ファイナルラウンド開始」(21:38),「優勝者発表」(22:05) の 4 つの時点にて実際に抽出されたキーワードを表 8，表 9，表 10 表 11 に示す．各パラメータはそれぞれ $T_{interval} = 1$ 分，$T_{reduce} = 1$ 分，$\epsilon = 0.001$ とし，$r = 0.5$ である．

表 8　ファーストラウンド開始時のキーワード

key 値上位 R 語	キーワード
1-5	#M1 グランプリ，#M1，選手，稲垣，三連単
6-10	投票，予想，あなた，も GYAO，サイト
11-15	敗者復活，GYAO，順位予想，3 位，注目
16-20	草，優勝，かまいたち，2 位，名前

表 9　敗者復活者発表時のキーワード

key 値上位 R 語	キーワード
1-5	かまいたち，敗者復活，#M1 グランプリ，面白い，優勝
6-10	私，ミキ，順位予想，敗者復活戦，#M1
11-15	三連単，1 位，予想，投票，ネタ
16-20	M-1，2 位，M-1 グランプリ，決勝，あなた

表 10　ファイナルラウンド開始時のキーワード

key 値上位 R 語	キーワード
1-5	#M1 グランプリ，和牛，面白い，べこぱ，芸人
6-10	優勝，ミルクボーイ，ストロング，#私，かまいたち
11-15	来年，ネタ，ほしい，決勝，応援
16-20	U+0001F62D，漫才，#和牛，こそ，M-1

表 11　優勝者発表時のキーワード

key 値上位 R 語	キーワード
1-5	ミルクボーイ，優勝，#M1 グランプリ，面白い，かまいたち
6-10	べこぱ，ネタ，ほしい，漫才，推す
11-15	勝つ，決勝，売れる，組，個人的
16-20	おもろい，和牛，来年，応援，芸人

ファーストラウンド開始時では，ゲストとして登場した稲垣選手がキーワードとなった．また，Twitter 上で M1 グランプリの優勝者予想企画が行われていたため,「投票」「三連単」「順位予想」などといった単語がキーワードになった．敗者復活者発表時では,「敗者復活」などがキーワードとなった．ファイナルラウンド開始時では，決勝戦進出を果たした「ミルクボーイ」「かまいたち」「べこぱ」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」がキーワードとなった．

6 結　　論

本研究では，Lossy Counting によって頻出項目を保持し，KeyGraph を用いてニュースを要約する連続 KeyGraph を提案した．実験ではオフラインでの KeyGraph と比較し，$\epsilon = 0.001$ の時において再現率 0.75 に対し実行時間を 69.8%，単語保持に必要なメモリを 86.5%短縮した．

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] 輪島　幸治, 古川　利博, 嶋田　茂, ”KeyGraph による主張点の極性評価 -LDA の潜在トピックを用いて-”，第 7 回データ工学と情報マネジメントに関するフォーラム (DEIM), 2015

[3] 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

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

[5] 徳山　豪, ”オンラインアルゴリズムとストリームアルゴリズム”, 共立出版, 2007

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