DEIM Forum 2019 B3-2

マイクロブログにおける高速なトピック出現量推移の抽出

福山　怜史† 　若林　 啓††

† 筑波大学大学院 図書館情報メディア研究科　〒 305–8550 茨城県つくば市春日 1-2
†† 筑波大学 図書館情報メディア系　〒 305–8550 茨城県つくば市春日 1-2
E-mail: †s1721691@s.tsukuba.ac.jp, ††kwakaba@slis.tsukuba.ac.jp

あらまし　近年，多くのメディアでは，関係するツイートの出現量が時間経過によって急上昇する話題を対象に情報発信が行なわれており，Twitter の話題の分析において話題の出現量の推移が注目されている．Twitter ではハッシュタグが一部のツイートにしか与えられていないため，全てのツイートに含まれる話題の推移を網羅的に観測することは容易ではない．この問題に対して，Biterm topic model (BTM) によってトピックを推定し，推定したトピックの出現量を利用する方法が有効である．しかし，Twitter ではリアルタイムに膨大なツイートが更新されるため，トピックの推定やトピック出現量の計算において時間的な効率性が求められる．本研究では，ツイートデータを対象に，高速にトピックを学習し，各トピックの単位時間あたりの出現量の計算を効率的に行う手法を提案する．提案手法では，BTM に対してミニバッチ学習を適用し，トピック学習の高速化を図る．またトピック出現量の計算では，一部のデータを用いた近似的な計算を行うことによって，実質的な高速化を図る．実験では，提案手法が既存手法より汎化性能が優れつつ学習における処理時間が短縮できることを確認した．またトピック出現量を近似する方法について複数の方法を示し，近似による誤差の大きさと処理時間の短縮の観点から比較と検討を行った．

キーワード　マイクロブログ，トピック出現量，Biterm topic model，トピックモデル，ミニバッチ学習

1 はじめに

近年，多くのメディアでは，ツイートの出現量が時間経過によって急上昇するような話題を対象に情報発信が行なわれている．このような背景から，Twitter に代表されるマイクロブログにおける話題の分析において，話題の出現量の時間的な推移の利用が注目されており，投稿されたツイートデータから話題の出現量の時間的な推移を推定する手法が求められている．この手法の一つに，話題に関係するハッシュタグが与えられたツイートを収集し，単位時間あたりのツイート数を計算し，話題の出現量とする方法がある．しかしこの手法は，Twitter ではハッシュタグが一部のツイートにしか与えられていないことから，話題に関係するツイートを収集するという観点において網羅的な手法といえない．以上の観点から，ツイートに含まれるトピックを分析し抽出する手法が有効であると考えられる．この手法として，Biterm topic model (BTM) [1] が提案されている．BTM とは，biterm と呼ばれる同じ文書に出現する単語対の集合を分析することで共起しやすい単語の集合としてトピックを抽出する手法である．したがって，BTM によってトピックの抽出を行い，抽出したトピックごとに単位時間あたりの推定ツイート数を計算することによって，Twitter で観測された話題における，全てのツイートを網羅した話題の出現量を得ることができる．

一方で，ハッシュタグを利用する手法に対して，この手法ではトピックの学習および単位時間あたりのトピックの出現量を計算する処理を必要とする．この処理では数日分のツイートから抽出した膨大な biterm を使用するため，時間的なコストが高い．以上の問題を軽減するために，本研究では，BTM の学習とトピック出現量の計算を高速化することで，Twitter における話題の時系列変化を効率的に抽出する手法を提案する．提案手法は，トピックの学習とトピック出現量の計算においてそれぞれ以下のアプローチによって高速化を図る．

- **トピック学習の高速化．**BTM ではトピック学習の効率的な推論アルゴリズムとして，確率的周辺化変分推論 (Stochastic collapsed variational Bayesian inference; SCVB0) [2] が提案されている．SCVB0 では反復処理の各イテレーションにおいて biterm を一つずつ読み込みトピックを学習する．提案手法では，SCVB0 にミニバッチを用いた学習を導入し，トピック学習の高速化を図る．この拡張による高速化の原理について，3.1 節で述べる．
- **トピック出現量計算の高速化．**BTM によって学習したトピックからトピック出現量を計算するためには，各時間で投稿されたツイートに含まれる biterm のトピックを推論する計算処理が必要となる．この処理では，各時間に出現した全ての異なり biterm に対してトピックの推論が行われるため，膨大な処理時間が掛かる．提案手法では，各時間に出現した biterm の内，一部の biterm のみ用いてトピック出現量を近似的に計算する．これにより，トピック出現量には誤差が含まれてしまうが，単位時間に出現する biterm を全て用いる場合と比較してより高速に計算することができる．

実験では，実際に投稿されたツイートを用いてトピックの学習およびトピック出現量の計算を行い，それぞれの手法で高速化の検証を行う．トピック学習の高速化では，既存手法と比較

して，汎化性能が劣らず処理時間が短縮されているか確認する．トピック出現量の高速化では，複数の手法の中から元のトピック出現量に対して，最も誤差が少ないかつ処理時間が短縮されている手法を検討する．

2 前 準 備

2.1 Biterm topic model (BTM)

BTM では，文書の集合を biterm の集まりに変換して扱う．本研究では，ツイートが文書に対応する．D をツイート数，ツイートの集まりを $\Omega = \sigma^{(1)}, \ldots, \sigma^{(D)}$，ツイート $\sigma^{(d)}$ の単語列を $w_1^{(d)}, \ldots, w_{|\sigma^{(d)}|}^{(d)}$ と表記する．それぞれの単語は語彙集合 $\mathcal{V}$ の要素であり，語彙数を $W = |\mathcal{V}|$ とする．また，それぞれのツイートにはタイムスタンプがついている．本稿では，特定の単位時間[1] 内のツイートには同一のタイムスタンプが付与されているとみなして，基準時刻を 1 とした σ のタイムスタンプを $\delta(\sigma) \in [1, T]$ と表す．ただし，$[1, T]$ は 1 から T までの整数の閉区間である．

biterm は，同一のツイートに含まれる 2 つの異なる単語からなる非順序対である．Ω に含まれる異なり biterm の集合 $\mathcal{V}_B \subseteq \mathcal{V}^2$ は以下のように定義される．

$$\mathcal{V}_B = \{\{w_i^{(d)}, w_j^{(d)}\} | d \in [1, D], i, j \in [1, |\sigma^{(d)}|], w_i^{(d)} \neq w_j^{(d)}\}.$$

また，重複を含めた biterm の集まりを $\mathbf{B}$ と表記する．1 つのツイート $\sigma^{(d)}$ において同一の biterm $b = \{w_1, w_2\}$ が複数回出現する際には，w_1 と w_2 の出現位置の添字を入れ替えただけの場合を除き，重複して出現したものとみなす．すなわち，ツイート $\sigma^{(d)}$ における biterm $b \in \mathcal{V}_B$ の頻度は以下のように表される．

$$n_{\sigma^{(d)}}(b) = |\{\{i, j\} | \{w_i^{(d)}, w_j^{(d)}\} = b\}|.$$

$\mathbf{B}$ における biterm $b \in \mathcal{V}_B$ の頻度は，$\sum_{d=1}^{D} n_{\sigma^{(d)}}(b)$ と表せる．$\mathbf{B}$ の要素数を $N_{\mathbf{B}}$ と表記し，その要素に便宜的に添字をつけて $\mathbf{B} = \{b_i\}_{i=1}^{N_{\mathbf{B}}}$，$b_i = \{w_{i,1}, w_{i,2}\}$ と表記する．

BTM では，それぞれの biterm b_i について，トピックを表す離散潜在変数 z_i を考える．トピック数を K として，$z_i \in [1, K]$ である．また，K 次元のトピック分布を示すベクトルを $\boldsymbol{\theta} = \{\theta_k\}_{k=1}^{K}$ とし，サイズ $K \times W$ の単語分布の行列を $\Phi = \{\boldsymbol{\phi}_k\}_{k=1}^{K}$ とする．Φ の行列の各行ベクトル $\boldsymbol{\phi}_k$ はトピックごとの単語分布を表し，その次元は W である．$\boldsymbol{\theta}$ および $\boldsymbol{\phi}_k$ の L_1 ノルムの大きさは 1 とする．BTM では以下の過程に従って biterm が生成されることを仮定する．

（1） Draw $\boldsymbol{\theta} \sim$ Dirichlet(α).

（2） For each topic $k \in [1, K]$,

（a） Draw $\boldsymbol{\phi}_k \sim$ Dirichlet(β).

（3） For each biterm $b_i \in \mathbf{B}$,

（a） Draw $z_i \sim$ Multinomial($\boldsymbol{\theta}$).

1：例えば，本稿の実験では，単位時間を 1 時間としている．

（b） Draw $w_{i,1}, w_{i,2} \sim$ Multinomial($\boldsymbol{\phi}_{z_i}$).

BTM を考案した Cheng らはこのモデルを周辺化ギブスサンプリング (Collapsed Gibbs Sampling; CGS) によって学習する手法を提案している [1]．しかし，CGS は $N_{\mathbf{B}}$ 個の潜在変数 z_i の値をメモリに展開しておく必要があり，大規模なデータの学習に適用することが難しいという欠点があることに加え，次節で述べる SCVB0 に比べて学習の進行が遅いことが知られている．

2.2 BTM における SCVB0

BTM を効率的に学習するアルゴリズムとして SCVB0 [2] がある．この手法は，LDA における周辺化変分推論 (Collapsed variational Baysian inference; CVB) [3] に対してトピックの期待値の式のテイラー展開を 0 次に近似したアルゴリズム (CVB0) [4] を，BTM の学習アルゴリズムに適用し，確率的最適化のアルゴリズムに拡張したものである．Awaya らは，BTM における SCVB0 の学習アルゴリズムが BTM における CGS [1] よりトピックの学習が高速であることを確認している [2]．BTM における SCVB0 では，$\mathbf{B}$ から 1 個ずつ biterm b_i を選択し，そのトピック z_i が k である確率を示す変分パラメータ $\hat{z}_{i,k}$ を式 (1) で推定する．

$$\hat{z}_{i,k} \propto (N_k + \alpha) \frac{(N_{w_{i,1}|k} + \beta)(N_{w_{i,2}|k} + \beta)}{(2N_k + W\beta)(2N_k + W\beta + 1)}. \tag{1}$$

コーパスにおけるトピック k の出現頻度の期待値 N_k とトピック k で単語 w が出現する頻度の期待値 $N_{w|k}$ の推定値として $\hat{N}_k$ および $\hat{N}_{w|k}$ をそれぞれ式 (2) および (3) によって計算する．

$$\hat{N}_k = |\mathbf{B}| \hat{z}_{i,k}, \tag{2}$$

$$\hat{N}_{w|k} = \begin{cases} |\mathbf{B}| \hat{z}_{i,k} & \text{if w} \in \text{b}_\text{i}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Robbins-Monro 法による確率的最適化に則り，この期待値の近似値から N_k と $N_{w|k}$ を式 (4) および (5) によって計算する．

$$N_k \leftarrow (1 - \nu^{(s)}) N_k + \nu^{(s)} \hat{N}_k, \tag{4}$$

$$N_{w|k} \leftarrow (1 - \nu^{(s)}) N_{w|k} + \nu^{(s)} \hat{N}_{w|k}. \tag{5}$$

$\nu^{(s)}$ は式 (6) によって計算される．

$$\nu^{(s)} = \frac{1}{(\tau + s)^\kappa}. \tag{6}$$

ただし，$\tau > 0$，$0.5 < \kappa \leqq 1$ は定数である．

また Awaya らは，$N_{w|k}$ の計算を効率的に行う手法を提案している．この手法では，s 回目の行列の更新における $N_{w|k}$ をスカラーと行列の積の形である $\rho_s \tilde{N}_{w|k}$ とする．ρ_s は $\prod_{i=1}^{s}(1 - \nu^{(s)})$ によって計算される．以上の計算によって，biterm b に含まれる単語 w_1, w_2 の更新は $\tilde{N}_{w|k}$ の要素 (k, w_1) および (k, w_2) に対する $\frac{\nu^{(s)}|\mathbf{B}|}{\rho_s}$ の加算のみとなり，$N_{w|k}$ の更新にかかる計算量は，K および W に対しては $\mathcal{O}(1)$ になる．

以上の更新を十分な回数繰り返して得られた統計量 N_k と $N_{w|k}$ を用いて，トピック割合 $\boldsymbol{\theta}$ とトピックの単語分布 Φ は，それぞれ式 (7) および (8) によって計算される．$N_\cdot$ は $\sum_{k'=1}^{K} N_{k'}$，

$N_{\cdot|k}$ は $\sum_{w'=1}^{W} N_{w'|k}$ を表す．

$$\theta_k = \frac{N_k + \alpha}{N_{\cdot} + \sum_{k'=1}^{K} \alpha_{k'}}, \tag{7}$$

$$\phi_{k,w} = \frac{N_{w|k} + \beta}{N_{\cdot|k} + W\beta}. \tag{8}$$

2.3 トピック分布のハイパーパラメータの最適化

BTM において，トピック分布 $\boldsymbol{\theta}$ およびトピック k の単語分布 $\boldsymbol{\phi}_k$ はそれぞれ α，β をハイパーパラメータとしたディリクレ分布を事前分布に持つ．Wallach ら [5] は，LDA におけるトピック分布 $\boldsymbol{\theta}$ のハイパーパラメータ α は各要素が異なる値を持つ非対称 Dirichlet 分布，トピック k の単語分布 $\boldsymbol{\phi}_k$ のハイパーパラメータ β は各要素が同一の値を持つ対称 Dirichlet 分布が有用であることを確認している．

この性質を BTM でも考慮し，本研究ではトピック分布 $\boldsymbol{\theta}$ のハイパーパラメータ α の最適化を行う．本研究では，BTM の学習と同時にハイパーパラメータ α の最適化を行い，後述する提案手法では，不動点反復法による式 (9) の反復計算によって更新を行う．

$$\alpha_k \leftarrow \alpha_k \frac{\Psi(N_k + \alpha_k) - \Psi(\alpha_k)}{\Psi(N_{\cdot} + \sum_{k'=1}^{K} \alpha_{k'}) - \Psi(\sum_{k'=1}^{K} \alpha_{k'}))}. \tag{9}$$

2.4 トピック出現量の計算

本研究では，それぞれの単位時間内に含まれるトピック出現量を推定することで，それぞれのトピックの時間的な推移を抽出する．ツイート σ のトピック割合は，式 (10) によって計算される [1].

$$P(z|\sigma) = \sum_{b \in \mathcal{V}_B} P(z|b) P(b|\sigma). \tag{10}$$

biterm $b_i = \{w_{i,1}, w_{i,2}\}$ が与えられた時，b_i がトピック k である確率 $P(z = k|b_i)$ は式 (11) によって計算される．

$$P(z = k|b_i) = \frac{\theta_k \phi_{k,w_{i,1}} \phi_{k,w_{i,2}}}{\sum_{k'} \theta_k \phi_{k,w_{i,1}} \phi_{k,w_{i,2}}}. \tag{11}$$

θ_k および $\phi_{k,w}$ は，それぞれ式 (7) および (8) によって学習結果として求めた値を用いる．また，ツイート中の biterm b の割合 $P(b|\sigma)$ は式 (12) によって定義される．

$$P(b|\sigma) = \frac{n_\sigma(b)}{\sum_{b' \in \mathcal{V}_B} n_\sigma(b')}. \tag{12}$$

ある単位時間 $t \in [1, T]$ 内のトピック k の出現量は，当該の単位時間に含まれるツイート集合 $\Omega_t = \{\sigma | \delta(\sigma) = t\}$ におけるトピック割合の総和 $N_{k|t} = \sum_{\sigma \in \Omega_t} P(z|\sigma)$ と定義する．$P(z|b)$ の値は σ に依存しないため，式 (10) における b についての和と σ についての和を入れ替えることで，$N_{k|t}$ は以下のように計算することができる．

$$N_{k|t} = \sum_{b \in \mathcal{V}_{B,t}} P(z = k|b) \sum_{\sigma \in \Omega_t} P(b|\sigma). \tag{13}$$

ただし，単位時間 t において頻度が 0 でない biterm 集合を $\mathcal{V}_{B,t} = \{b | \exists \sigma \in \Omega_t [n_\sigma(b) > 0]\}$ と定義した．

Algorithm 1 BTM における SCVB0 のミニバッチ学習

```
Randomly initialize N_k and N_{w|k}
for each iteration do
  Sample minibatch of biterms B^(s)
  Compute N_b^(s) in B^(s) for all b ∈ V_B^(s)
  for b ∈ V_B^(s) do
    for each topic k ∈ [1, K] do
      Compute ẑ_{b,k} using Eq. (1)
    end for
  end for
  Update N_k using Eq. (14) and N_{w|k} using Eq. (15)
end for
Compute global parameters θ using Eq. (7) and Φ using Eq. (8)
```

3 提案手法

提案手法では，トピックの学習の高速化のために，SCVB0 をミニバッチ学習に拡張する．また，抽出したトピックの出現量を推定する時，単位時間に出現した biterm から一部に対して推論を適用することで高速化を図る．

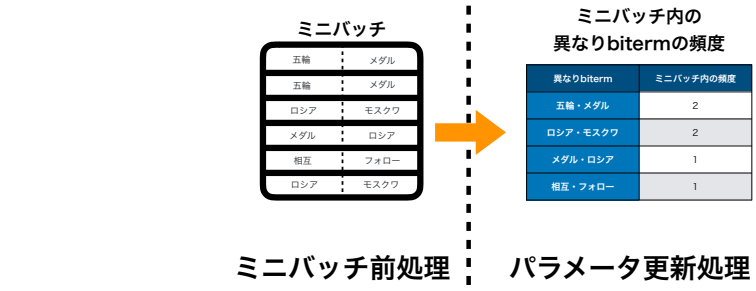
3.1 BTM における SCVB0 によるミニバッチ学習

訓練するデータの中から複数のデータをサンプリングし，それを全体のデータの近似として学習する手法をミニバッチ学習と呼ぶ．SCVB0 によるミニバッチ学習では，BTM を SCVB0 のアルゴリズムで学習する際に $\mathbf{B}$ からあらかじめ決定したミニバッチサイズの数だけ biterm をサンプリングし，ミニバッチごとに N_k と $N_{w|k}$ の更新を行う．SCVB0 によるミニバッチ学習のアルゴリズムを Algorithm 1 に示す．$\mathbf{B}^{(s)}$ は s 回目のイテレーションにおけるミニバッチに含まれる（重複を含む）biterm の集まり，$\mathcal{V}_B^{(s)}$ は $\mathbf{B}^{(s)}$ に含まれる異なり biterm の集合，$N_b^{(s)}$ は $\mathbf{B}^{(s)}$ に含まれる biterm b の頻度を表す．ミニバッチ学習による，SCVB0 の更新式を式 (14) および (15) とする．

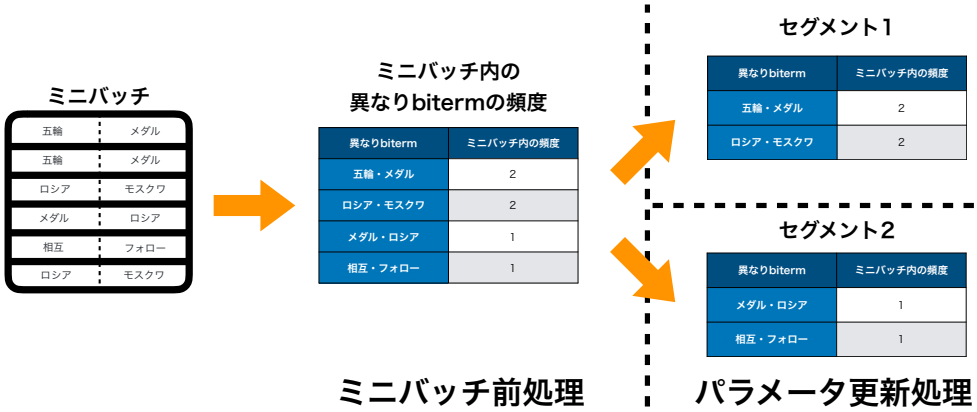
$$N_k \leftarrow (1 - \nu^{(s)}) N_k + \nu^{(s)} \frac{|\mathbf{B}|}{|\mathbf{B}^{(s)}|} \sum_{b \in \mathcal{V}_B^{(s)}} \hat{z}_{b,k} N_b^{(s)}, \tag{14}$$

$$N_{w|k} \leftarrow (1 - \nu^{(s)}) N_{w|k} + \nu^{(s)} \frac{|\mathbf{B}|}{|\mathbf{B}^{(s)}|} \sum_{b \in \mathcal{V}_B^{(s)}} \hat{z}_{b,k} N_b^{(s)} \, \delta(w \in b). \tag{15}$$

ただし，$\hat{z}_{b,k}$ は，添字の biterm b を当該の biterm が元々出現する位置 i に読み替えて式 (1) で計算する．このアルゴリズムでは，ミニバッチとして複数の biterm をサンプリングした際に，ミニバッチ内に出現する各 biterm の頻度を数え上げ，biterm $b \in \mathcal{V}_B^{(s)}$ の変分パラメータ $\hat{z}_{b,k}$ とその biterm の頻度 $N_b^{(s)}$ の積の総和を N_k と $N_{w|k}$ の更新量として計算する．これにより，トピック学習のボトルネックとなるミニバッチ内の各 biterm の変分パラメータの計算回数は，ミニバッチ内に含まれる異なり biterm 数まで削減される．



(a) SCVB0 のミニバッチ学習



(b) ミニバッチをセグメントに分割する学習 (ミニバッチ–セグメント学習)

図 1: SCVB0 におけるミニバッチ学習とミニバッチ–セグメント学習．提案手法 1 ではミニバッチ全体でパラメータ更新処理を行うことに対して，提案手法 2 ではミニバッチをさらにセグメントに分割してセグメントごとにパラメータ更新を行う．

3.2 ミニバッチをセグメントに分割する学習 (ミニバッチ–セグメント学習)

本研究では，ミニバッチ内で重複する biterm が存在する性質を利用し，さらに効率的な学習手法を提案する．3.1 節で説明しているミニバッチ学習では，ミニバッチサイズが大きければ大きいほど重複する異なり biterm の種類の数が多くなるため，変分パラメータの計算回数の削減量が大きくなることが期待される．一方で，ミニバッチサイズを大きくするほどミニバッチに含まれる異なり biterm 数が増加するため，変分パラメータの更新の時間的な周期が長くなり，結果的に実時間に対する学習の進行速度が低下する可能性がある．

ミニバッチ–セグメント学習では，ミニバッチサイズを大きく保ちながら更新の時間的な周期を短くするために，ミニバッチ内の biterm の頻度を数え上げた後，あらかじめ決めた種類数ずつ異なり biterm 集合 $\mathcal{V}_B^{(s)}$ を分割し，この分割 (セグメント) ごとに N_k と $N_{w|k}$ の更新を行う．本研究では，各分割に含まれる異なり biterm の数をセグメントサイズと呼び，提案手法 2 を “ミニバッチ–セグメント学習” と呼ぶ．図 1 に示すように，ミニバッチ学習では，サンプリングした biterm 全てを用いて N_k と $N_{w|k}$ を更新することに対し，ミニバッチ–セグメント学習ではサンプリングした異なり biterm を複数のセグメントに分割し，このセグメントごとに N_k と $N_{w|k}$ の更新を行う．これにより，ミニバッチ学習の場合と比較して，サンプリングした biterm の数が同一でも，より短い周期で N_k と $N_{w|k}$ の更新を行うことができるため，トピック学習の進行がより高速になると考えられる．

3.3 単位時間あたりのトピック出現量の計算と近似

本研究では，各単位時間のトピック出現量を求めるために，その時間 t に投稿されたツイート Ω_t に含まれる biterm に対して BTM で抽出したトピックの割合を計算し，ツイートに含まれるトピック k の割合の単位時間あたりの合計 $N_{k|t}$ を式 (13) によって計算する．式 (13) の計算には，$\mathcal{V}_{B,t}$ に含まれる全ての biterm のトピック割合を式 (11) によって計算する必要があり，これがトピック出現量の計算のボトルネックになっていると考えられる．

各単位時間のトピック出現量の計算時間を短縮するために，本研究では，$\mathcal{V}_{B,t}$ から一部を取り出し，取り出された biterm のトピックに基づいて各トピックの出現量の計算を近似的に行う．トピック出現量に対する寄与の大きい biterm を特定する手がかりとして，式 (13) で用いられる以下の項を “biterm 重み” $\omega_{t,b}$ として用いる．

$$\omega_{t,b} = \sum_{\sigma \in \Omega_t} P(b|\sigma). \tag{16}$$

この定義に基づいて，取り出す biterm の決定方法として以下の手法を比較・検討する．

- biterm 重みの離散分布からサンプリングする．
- 異なり biterm からランダムに選択する．
- biterm 重みの高い順に選択する．

biterm 重みの離散分布からサンプリングする手法では，$\omega_{t,b}$

を biterm 重みの総和 $\sum_{b' \in \mathcal{V}_{B,t}} \omega_{t,b'}$ で割った値を biterm b についての離散分布のパラメータとみなし，この分布からあらかじめ決めた回数だけランダムにサンプリングを行い，このサンプリングされた biterm の集まり Ω'_t およびこれに含まれる異なり biterm 集合 $\mathcal{V}'_{B,t}$ を用いて式 (13) を計算する．異なり biterm からランダムに選択する手法では，異なり biterm をランダムに一定数選択し，トピック出現量の計算では元の biterm 重みを使う．biterm 重みの高い順に選択する手法では，異なり biterm の数が一定になるまで，biterm 重みの高い順に異なり biterm を取り出す．異なり biterm からランダムに選択する手法と同様に，トピック出現量の計算では元の biterm 重みを使う．いずれの手法も，元の式 (11) で計算されるトピック出現量に対する近似値を計算する手法である．

4 評 価 実 験

提案手法の有効性を検証するため，実際のツイートデータから提案手法によってトピック出現量の推定を行う．

4.1 実験データ

実験では，2012 年 8 月 1 日から 2012 年 8 月 7 日の間に投稿されたツイートデータを用いる．ツイートデータの総数は 286,223,678，1 日あたりの平均ツイート数は 40,889,096 である．本研究では，ノイズとなる語を除去するために，ツイートに含まれる語彙に対して以下の制約を設ける．

（1） 固有名詞・一般名詞・サ変接続の名詞以外の品詞を持つ単語を除く．

（2） 2 字以上の漢字・ひらがな・カタカナ・数字の組みあわせで構成されている単語以外を除く．

（3） リツイートを示す「RT」・リプライを示す「@ユーザ名」・ハッシュタグを示す「#」に続く文字列および URL を除く．

（4） 笑いを意味する「w」・「W」・「笑」の単語は除く．

除去した結果，1 つも単語を含まないツイートをツイートの総数は 243,600,235 で，1 日あたりの平均ツイート数は 34,800,034 である．上記の制約を満たしたツイートデータの単語数について集計した結果を図 2 に示す．図 2 より，含まれる単語の数が 10 以下であるツイートが多くの割合を占めることがわかる．

BTM の学習のためにこれらのツイートから biterm の抽出を行う．抽出された biterm の総数は 3,077,224,154 であり，異なり biterm 数は 355,980,502 である．本研究では，データ数削減のために頻度が 10 以下の biterm を処理対象から除く．この結果，実験に用いる biterm の総数は 2,453,945,076，異なり biterm 数では 32,016,826 である．

実験は，OS が Ubuntu 16.04，CPU が Intel Xeon E5-2630 (2.40GHz) 8core 2 機の計算機上で行った．前処理における形態素解析器は ipadic を用いた MeCab [6]，各アルゴリズムの実装は Python で行った．

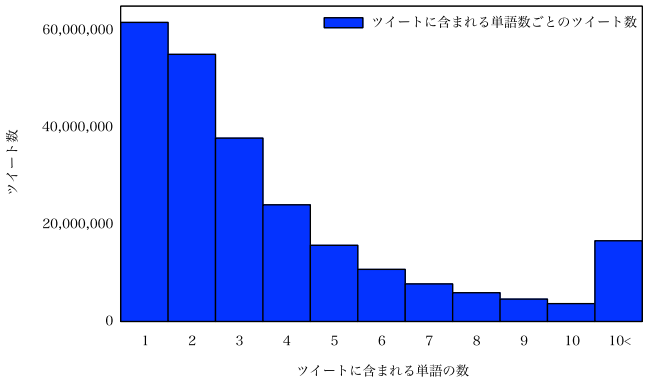


図 2: ツイートに含まれる単語数ごとのツイート数

表 1: ミニバッチに含まれる異なり biterm 数の平均値．各値はそれぞれのデータに対してミニバッチを 100 回抽出した時の異なり biterm 数の平均値である．

ミニバッチサイズ	異なり biterm 数の平均値
5	5.000
10	10.000
100	99.989
1,000	997.740
10,000	9860.823
100,000	93673.470
1,000,000	779655.323

4.2 トピック学習の高速化の評価

提案手法によってトピックの学習が高速化されていることを検証するため，既存手法との比較を行なった．

4.2.1 比 較 手 法

実験データとして抽出した biterm に対して，以下の 3 種類の手法によってトピックを学習する．

SCVB0 提案手法のベースとなる BTM における SCVB0 の推論アルゴリズムを，ベースラインとして用いる．この手法では，biterm を一つずつランダムに抽出し，式 (4) および (5) によって N_k と $N_{w|k}$ の更新を行う．

提案手法 (ミニバッチ学習) ミニバッチサイズの数だけ biterm を抽出し，式 (14) および (15) によって N_k と $N_{w|k}$ の更新を行う．ミニバッチサイズが 1 の時，比較手法となる SCVB0 と同一の手法となる．

提案手法 (ミニバッチ–セグメント学習) ミニバッチサイズの数だけ biterm を抽出し，ミニバッチ内の異なり biterm をセグメントサイズずつ分割し，セグメントごとに N_k と $N_{w|k}$ の更新を行う．セグメントサイズが 1 の時，ミニバッチ学習と同一の手法となる．

4.2.2 パラメータ設定

BTM のトピック数 K を 512 で固定する．またハイパーパラメータ α と β はそれぞれ $\alpha = 50/K$ と $\beta = 1/W$ とする．ただし，α は BTM の学習と同時に式 (9) によってトピックごとに更新される．減衰重み ν に関するパラメータである τ と κ

表 2: 各手法における学習用データを学習させた時の評価用データの平均対数尤度と学習に掛かる処理時間

手法	ミニバッチサイズ	イテレーション	セグメントサイズ	平均対数尤度	処理時間 [sec]
SCVB0	1	100,000,000	-	−19.231	10,496 ± 103
ミニバッチ学習	10,000	10,000	-	**−19.088**	7,253 ± 154
	100,000	1,000		−19.706	7,077 ± 174
	1,000,000	100		−21.517	6,315 ± 149
ミニバッチ–セグメント学習	1,000,000	100	10	**−18.998**	5,797 ± 144
			100	**−18.987**	4,822 ± 9
			1,000	**−18.995**	5,282 ± 123
			10,000	**−19.100**	5,931 ± 143
			100,000	−19.875	6,074 ± 40

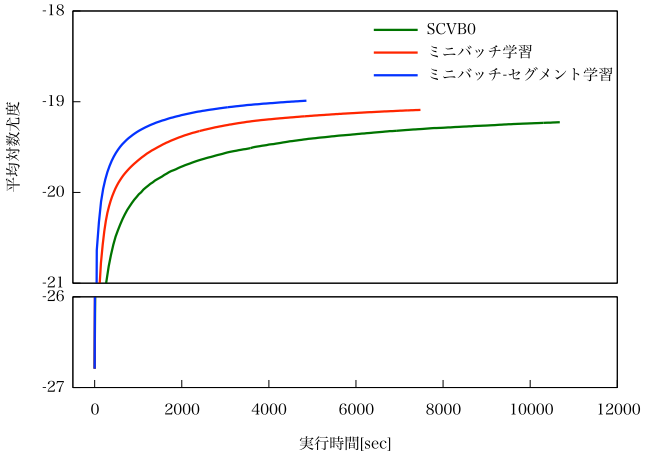


図 3: 各手法の実行時間と平均対数尤度の変化

は，Awaya ら [2] の実験と同一のパラメータである $\tau = 1,000$ と $\kappa = 0.8$ とする．

4.2.3 実 験 方 法

本実験では提案手法が，SCVB0 と比較して，モデルの汎化性能を劣化させることなく，処理時間が短縮できることを確認する．実験では，実験用に 1 億個の biterm をサンプリングし，9 : 1 の割合で学習用データと評価用データに分割し，学習用データで学習したモデル [2] に対する評価用データの平均対数尤度を比較する．本研究では，評価用データに対する平均対数尤度を式 (17) で計算する．

$$\frac{1}{|\mathbf{B}_{\text{test}}|} \sum_{b_i \in \mathbf{B}_{\text{test}}} \log \sum_{k=1}^{K} \theta_k \phi_{k,w_{i,1}} \phi_{k,w_{i,2}}. \tag{17}$$

トピック分布のハイパーパラメータの更新は，biterm を 10,000,000 個サンプリングするごとに行う．この最適化における不動点反復法の反復回数は 20 回とする．各手法で学習時に抽出される総 biterm 数は 1 億とする．SCVB0 では，1 サンプルの学習を 1 イテレーションとして扱い，このイテレーションを 1 億回行う．提案手法のミニバッチ学習では，ミニバッチサイズ分の学習を 1 イテレーションとして扱い，ミニバッチサイズとイテレーション回数の積が 1 億になる組み合わせで行なう．提案手法のミニバッチ–セグメント学習では，ミニバッチサイズを 1,000,000，イテレーション回数を 100 に固定し，セグメントサイズを変更する．

各手法では，alias 法 [7] [8] によって biterm をサンプリングする．aliash 法は，与えられた離散分布について ailas テーブルと呼ばれるデータ構造を構築することで，当該の離散分布からのサンプリングを $\mathcal{O}(1)$ で実行可能とする手法である．alias 法における離散分布は，各 biterm の出現頻度をその総和から割った値によって求める．

本研究では，ミニバッチ学習におけるミニバッチサイズとイテレーション回数の組み合わせの候補を決定するために，予備実験を行った．予備実験では，学習用データおよび全ての biterm による実験データにおいてミニバッチを 100 回抽出し，ミニバッチに含まれる異なり biterm 数の平均値を計算した．表 1 に示した予備実験の結果より，ミニバッチサイズを大きくすればするほどミニバッチサイズに対する異なり biterm 数の割合が小さくなることがわかる．一方で，ミニバッチサイズの大きさが 1,000 以下の場合，biterm の推論回数の削減される割合が 1%未満となることから，ミニバッチ学習による高速化が期待できない．したがって本研究では，ミニバッチ学習の手法においてミニバッチサイズを 10,000 以上と設定し，実験を行う．

4.2.4 実 験 結 果

各手法における学習用データを学習させた時の評価用データの平均対数尤度と学習に掛かる処理時間を表 2 に示す．ベースラインである SCVB0 と比較して，提案手法であるミニバッチ学習およびミニバッチ–セグメント学習の処理時間はいずれの条件においても短縮されていることがわかる．特にミニバッチ学習では，ミニバッチサイズを大きくすればするほど処理時間が短縮されていることから，ミニバッチに含まれる異なり biterm の頻度の数え上げにより効率的な処理が行われていることがわかる．

一方で平均対数尤度は，パラメータによって優劣に差がみられる．SCVB0 よりも平均対数尤度が高い手法は，表中で太字で示している．特にミニバッチ–セグメント学習は，セグメントサイズが 10・100・1,000 の場合，SCVB0 およびミニバッチ

2 : 各手法では，N_k と $N_{w|k}$ の更新で使われる式 (1) による biterm のトピック確率の計算を，ループ処理ではなく，行列やベクトルに変形し，更新で使われる全ての biterm に対して一度に計算する．本研究では効率的な数学演算ライブラリとして，Intel Math Kernel Library(https://software.intel.com/en-us/mkl) を用いる．

表 3: 元のトピック出現量に対する各手法の RMSE と処理時間．RMSE および処理時間は 5 回の実験の平均値とする．
Table 3　RMSE of each method for baseline. We calculated the average of each value from 5 experiments.

手法	異なり biterm 数	平均 RMSE	処理時間 [sec]
全ての biterm 重みを使う (元のトピック出現量)	$4,026,528$	-	$28,351 \pm 108$
biterm 重みの離散分布からサンプリングする	$100,000$	60.565	$2,263 \pm 162$
	$500,000$	22.509	$5,503 \pm 21$
	$1,000,000$	13.340	$10,263 \pm 34$
異なり biterm からランダムに選択する	$1,000,000$	213.840	$7,788 \pm 319$
biterm 重みの高い順に選択する	$1,000,000$	217.605	$8,526 \pm 51$

学習の全ての条件と比較して，平均対数尤度が高い．これは，ミニバッチサイズに対して小さなセグメントサイズを設定することで変分パラメータの更新頻度が増加し，学習が早く進んだためであると考えられる．更新頻度が増加しているにも関わらず，同じミニバッチサイズのミニバッチ学習よりも処理時間が高速な理由は，CPU のキャッシュ効果の影響などが考えられるが，より詳細な分析は今後の課題とする．

SCVB0・ミニバッチ学習 (ミニバッチサイズ 10,000 かつイテレーション 10,000)・ミニバッチ–セグメント学習 (セグメントサイズ 100) における学習の実行時間に対する平均対数尤度の変化を図 3 に示す．図 3 より，SCVB0 と比較して，ミニバッチ学習およびミニバッチ–セグメント学習は実行時間の経過に対する平均対数尤度の収束が早いことがわかる．またミニバッチ学習とミニバッチ–セグメント学習を比較した場合，ミニバッチ–セグメント学習の方が平均対数尤度が早い段階で収束することがわかる．以上の結果より，適切にミニバッチサイズとセグメントサイズを設定することで，提案手法は SCVB0 よりも汎化性能が向上しつつ，処理時間が高速化されることがわかる．

4.3 biterm の一部を抽出しトピック出現量を近似する方法の評価

4.3.1 実 験 方 法

提案手法の有効性を示すため，全ての biterm を使って推定されるトピック出現量に対する近似精度と推定に掛かる実行時間を比較する．本実験では，式 (13) によって計算された各単位時間におけるトピック出現量に対して，3.3 節で挙げた手法で近似したトピック出現量との平均平方二乗誤差 (Root mean square error; RMSE) とトピック出現量の計算に掛かる処理時間を比較する [3]．

近似したトピック出現量 $\hat{N}_{k|t}$ は元のトピック出現量 $N_{k|t}$ に対して単位時間あたりのトピックの出現量の総和が異なるため，式 (18) によってスケールを揃えた近似値 $\tilde{N}_{k|t}$ を求める．

$$\tilde{N}_{k|t} = \frac{\sum_{k'=1}^{K} N_{k'|t}}{\sum_{k'=1}^{K} \hat{N}_{k'|t}} \hat{N}_{k|t}. \tag{18}$$

全ての biterm によって計算されたトピック出現量 $N_{k|t}$ に対する $\tilde{N}_{k|t}$ の RMSE の平均値を式 (19) で計算する．

$$\frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(N_{k|t}-\tilde{N}_{k|t}\right)^2}. \tag{19}$$

式 (19) による RMSE の平均値は単位時間あたりの各トピックの出現量の誤差の平均値を意味する．

biterm 重みの離散分布からサンプリングする手法では，各単位時間で異なり biterm 数が 100,000・500,000・1,000,000 をそれぞれ超えるまで biterm のサンプリングを続ける．この手法では，4.2 節と同様に alias 法 [7] [8] によって biterm をサンプリングする．異なり biterm からランダムに選択する手法と biterm 重みの高い順に選択する手法では，選択する biterm の数を 1,000,000 とする．以上の手法によるトピック出現量の計算をそれぞれ 5 回行い，その RMSE と処理時間の平均値を比較する．ただし biterm 重みの高い順に選択する手法では，一意な RMSE が計算されるため，処理時間のみ平均値を計算する．

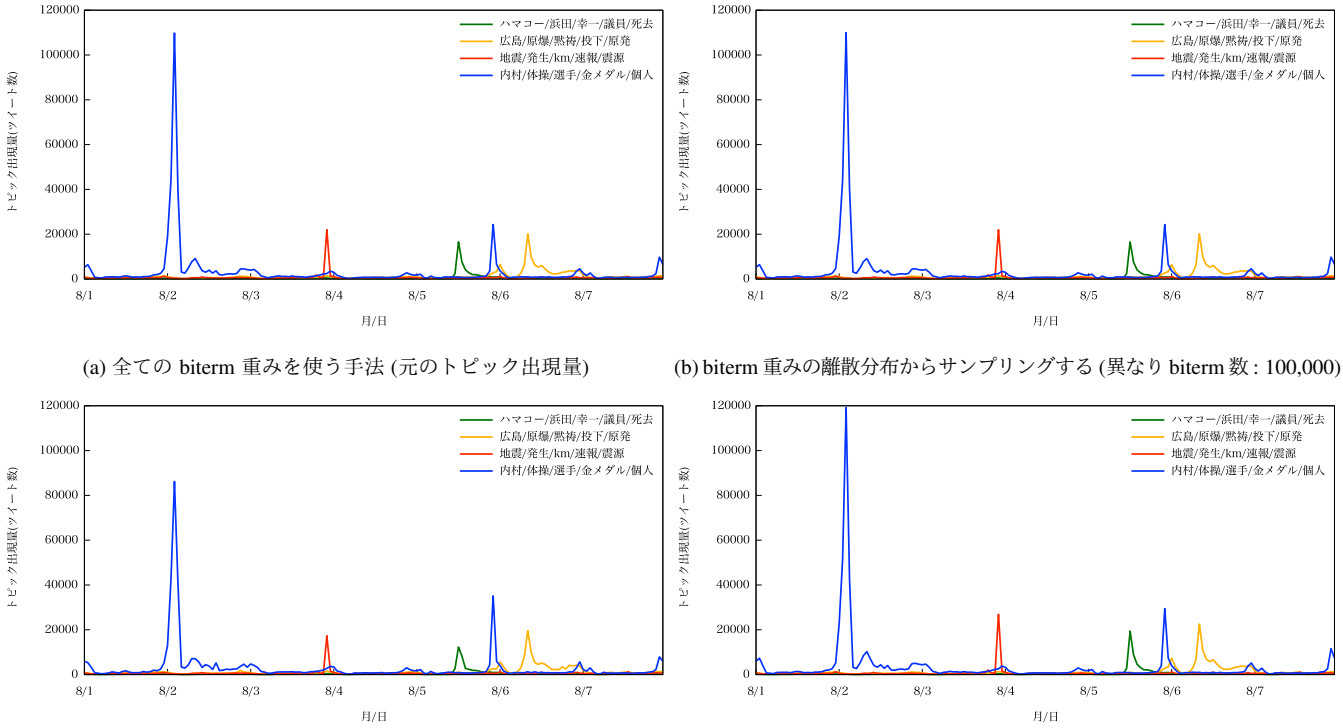
4.3.2 実 験 結 果

元のトピック出現量に対する RMSE と計算に掛かった処理時間を表 3 に示す．表中の異なり biterm 数は，各単位時間あたりの平均値を示している．元のトピック出現量の計算では，異なり biterm が単位時間あたりに平均 4,026,528 個存在し，その処理時間が 7 時間 53 分ほど掛かるが，提案手法により大幅に処理時間を削減できることがわかる．

異なり biterm 数 1,000,000 の条件においては，異なり biterm からランダムに選択する手法や biterm 重みの高い順に選択する手法の方が，biterm 重みの離散分布からサンプリングする手法に比べて処理時間が短いが，元のトピック出現量に対する RMSE は大きくなってしまう．また，biterm 重みの離散分布からサンプリングする手法は，用いる異なり biterm 数を少なくすることで，異なり biterm からランダムに選択する手法や biterm 重みの高い順に選択する手法よりも処理時間が短くなり，かつ，RMSE も低くなることから，より優れた手法といえる．

実際に各手法によって計算されたトピック出現量を図 4 に示す．いずれの手法もスケールに誤差が生じているが，時間経過によるトピック出現量の増減の概形は同じであることから，トレンドを把握する目的には大きな問題にはならないと考えられ

3 : トピック出現量を計算するトピックは，4.2 節の実験において最も平均対数尤度の高いトピックが抽出されたミニバッチ–セグメント学習で推定する．ミニバッチ–セグメント学習の設定は，ミニバッチサイズ 1,000,000・イテレーション 100・セグメントサイズ 1,000 の設定で行なった．この設定は，全ての biterm を使って学習したトピックにおいて，学習した biterm から計算した平均対数尤度が最も高い設定である．



(a) 全ての biterm 重みを使う手法 (元のトピック出現量)

(b) biterm 重みの離散分布からサンプリングする (異なり biterm 数 : 100,000)

(c) 異なり biterm からランダムに選択する

(d) biterm 重みの高い順に選択する

図 4: 各手法によって計算されるトピック出現量．凡例は各トピックで出現確率の高い語を順に左から並べたものである．

る．また，biterm 重みの離散分布からサンプリングする手法に基づいて計算された図 4b の結果では，スケールに対する誤差も小さいことがわかった．

5 結　　論

本研究では，SCVB0 に対して，ミニバッチ学習をベースとした拡張と近似的なトピック出現量の計算によって，より高速な学習とトピック出現量の計算を行う手法を提案した．実験では，2012 年 8 月 1 日から 8 月 7 日に投稿されたツイートデータに対して，提案手法を適用し，トピックとトピック出現量を抽出した．この結果，提案手法によるトピック抽出では既存手法に劣らない汎化性能を保ちつつ，高速な学習が行われていることが確認された．また最も誤差が少ない近似手法が biterm 重みの離散分布からサンプリングする手法であることを確認した．

今後の展望として，BTM のトピックの継続的な利用が考えられる．ツイートのトピックを継続的に追跡する場合，提案手法では何度もモデルを学習し直す必要があるため，時間的なコストが掛かる．今後は，一度使用したモデルを再利用する手法を考案し，継続的な利用が可能であるか検証を行う．

謝　　辞

本研究の一部は，JSPS 科研費（課題番号 16H02904）および筑波大学図書館情報メディア系プロジェクト研究の助成によって行われた．

文　　献

[1] Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. BTM: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941, 2014.

[2] N. Awaya, J. Kitazono, T. Omori, and S. Ozawa. Stochastic collapsed variational bayesian inference for biterm topic model. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3364–3370, 2016.

[3] Yee Whye Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 1353–1360, Cambridge, MA, USA, 2006.

[4] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 27–34, Arlington, Virginia, United States, 2009.

[5] Hanna M. Wallach, David M. Mimno, and Andrew McCallum. Rethinking lda: Why priors matter. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1973–1981. Curran Associates, Inc., 2009.

[6] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis,. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 230–237, 2004.

[7] Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, 1977.

[8] Jingbo Wang, Wai Wan Tsang, and George Marsaglia. Fast generation of discrete random variables. *Journal of Statistical Software*, 11, 2004.