トピック空間上での文全体と各単語の類似度に基づく語重み付け手法

田中　雄也†　　田島　敬史††

† 京都大学工学部情報学科　〒 606–8501 京都府京都市左京区吉田本町
†† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町
E-mail: †yuya.tanaka@dl.soc.i.kyoto-u.ac.jp, ††tajima@i.kyoto-u.ac.jp

あらまし　本論文では，Twitter における各投稿のようなショートテキスト中の語に対しても，適切に重み付けを行うことが可能な手法を提案する．例えば，Twitter における各投稿には 140 文字以内という制限があるため，web ページなどとは異なり，同じ語が複数回出現することは少ない．そのため，情報検索において一般的に用いられる手法である tf-idf では，ショートテキスト中における各語の重要度に従って重みを付けることは難しい．本研究では，文書の特徴ベクトルと語の特徴ベクトルのコサイン類似度によって重み付けを行う手法を提案する．本手法では，Word2Vec と Doc2Vec を用いることで文書と語の特徴ベクトルを抽出する．

キーワード　Doc2Vec，重み付け，ショートテキスト，類似度，Twitter，特徴ベクトル

1. はじめに

Twitter(注1) や Facebook(注2) のようなソーシャル・ネットワーキング・サービスの普及は著しく，その重要性は増すばかりである．例えば Twitter における日本国内の月間アクティブ・ユーザー数は，2011 年 3 月時点では 670 万人だったが (注3)，2017 年 10 月には 4500 万人を突破した (注4)．天気の変化や事故の情報など，普段からリアルタイムに現地の情報を手軽に入手できる手段として，多くの人々に利用されている．2011 年に発生した東日本大震災の際には，安否確認や，被害状況を始めとする様々な情報の収集・発信のツールとして，多くの一般人のみならず，自治体やその他の公的機関もこうしたサービスを活用した．また世界中の多くの企業が，広報やカスタマーサポートを目的として，さらには，自社のニュースの配信を目的としてマス・メディアもが，ソーシャル・ネットワーキング・サービスのアカウントを保有・運営しており，今や人々にとってソーシャル・ネットワーキング・サービスは，マス・メディアと並ぶ情報源となりつつあるばかりでなく，情報発信においても重要なものとなりつつある．

その一方で検索者が，必要としている情報を含む投稿を見つけ出すことは難しい．情報検索では一般的に，文書をランキングするために集合内の各文書に対してクエリとの関連度を計算するが，その際，文書中の各語に重みを与え，文書をベクトル化するということが頻繁に行われている．しかしツイートの場合，英語などの言語では 280 文字以内，日本語などでは 140 文字以内という制限があるため，1 つのツイートに含まれる語数が非常に少なく，また，ほとんどの語は一つのツイート中に一回しか現れない．そのため，語に正しく重みを付けることは難しい．

また近年，Twitter を始めとするマイクロブログの検索に関する研究が，情報検索の分野における新たな課題として盛んに行われている．例えば，140 文字，もしくは 280 文字以内という制限によるツイート検索の困難さを克服するために，様々なアプローチから研究されている．また，マイクロブログにおける投稿のトピック分布を推定する手法に関しても，多くの研究が行われている．

こうした問題を解決するために，ハッシュタグや位置情報，投稿したユーザー自身の情報など，投稿の本文以外の情報を用いることで，情報を補完し精度を向上させる手法が一般的に研究されている．しかし，こうした手法はあくまでも特定のサービスに依存した手法であるため，汎用性が低い．そこで本論文では，ツイートのような短文においても文章の意味を正しく表現するように，かつ，特定のサービスに依存しない方法で，語に重み付けを行う手法を考える．この手法を実現するために以下の仮説を立てた．

> 文書において重要な語のトピックは，文書全体のトピックに類似する．

この仮説に従い，文書とその文書内に出現する語のトピック分布をそれぞれ推定し，2 つの分布の類似度によって語に重みを付ける手法を提案する．具体的には，語のトピック分布は Mikolov ら [1] が 提案した Word2Vec を用いて，文書のトピック分布は Le ら [2] が提案した Doc2Vec を用いて，それぞれ推定する．そして，これら 2 つのベクトルのコサイン類似度を語の重みとする．

実験では trec 2011 Microblog Track(注5) のデータセットを用いて Doc2Vec や Word2Vec のモデルを構築し，提案手法による重み付けが有効であることを示すためにツイート検索・ラ

(注1)：https://twitter.com
(注2)：https://www.facebook.com/
(注3)：http://www.huffingtonpost.jp/2016/02/18/twitter-japan_n_9260630.html
(注4)：https://mainichi.jp/articles/20171027/k00/00e/040/370000c
(注5)：http://trec.nist.gov/data/microblog2011.html

ンキングを行う．そして， trec 2012 Microblog Track(注6) の正解データを使用し， nDCG を用いてこのランキングの評価を行う．この評価実験の結果，本手法を用いて重み付けした場合，ランキングの精度が既存手法に比べて大きく向上することが確認された．

本論文の構成は以下の通りである．2. 章では，関連研究を紹介する．3. 章では，本研究で用いる Doc2Vec と Word2Vec について説明した後に，提案手法について説明する．4. 章では，提案手法に対する実験とその結果について説明し，課題を考察する．5. 章では，本論文の結論を述べる．

2. 関 連 研 究

情報検索に関する研究は盛んに行われてきたが，近年，Twitter を始めとするマイクロブログの検索に関する研究が，情報検索の分野における新たな課題として盛んに行われている．例えば，140 文字，もしくは 280 文字以内という制限によるツイート検索の困難さを克服するために，様々なアプローチから研究されている．また，マイクロブログにおける投稿のトピック分布を推定する手法に関しても，多くの研究が行われている．この節では，こうした分野の先行研究を紹介し，本研究の位置付けを明確にする．

2.1 ツイート検索に関する先行研究

Luo ら [4] は，ツイートの構造的な情報を用いることでツイート検索の精度を向上させる手法を提案した．Luo らの手法では，ハッシュタグ（例: #iphone）や, 他のユーザーへのメッセージを表す部分（例: @ladygaga），リツイートを表す表現 (例: RT @ladygaga)，URL，コメントといったツイートの構造情報を Twitter Building Blocks (TBB) と定義し，TBB の長さや数，TBB 中におけるクエリ語の位置，TBB における Out Of Vocabulary 語（スペルミスや省略などによって，語彙には含まれない語）の割合といった情報を用いてツイートのランキングを行う手法を提案した．Luo らが行った実験によると，TBB の情報以外にも，ツイートの長さやリンクの有無といった基本的な情報，投稿ユーザーのフォロワー・フォロイー数，リツイート数やハッシュタグ数といった，ソーシャル・メディア特有の情報も加えてランキングすることで，性能が大幅に向上することが確認されている．また，後述のマイクロブログへ適用する LDA に関する研究においても，本文以外の情報を用いることで LDA の性能を向上させる手法が複数提案されている．

投稿の本文の情報のみを使用する手法には，Lau ら [5] の手法がある．Lau らは，語ベースの重み付けとパターンベースの重み付けを組み合わせることで，マイクロブログ検索の性能を向上させる手法を提案した．この手法におけるパターンとは，文書集合内で一定回数以上出現する 3 語以内の語列のことであり，語ベースの重み付けとは tf-idf のことである．Lau らは，ツイートと疑似適合フィードバックによる拡張済みのクエリの両方に対して提案手法を用いた重み付けを行い，Jaccard 係数によってランキングを行った．

2.2 トピックモデルに関する先行研究

文書のトピック分布を抽出する手法はいくつか存在するが，代表的な手法の 1 つに，Blei ら [6] が提案した Latent Dirichlet Allocation (LDA) がある．この手法は，ベイジアン・ネットワークを用いて，与えられた文書集合から各トピックにおける語の生成分布を抽出する教師なし機械学習である．しかしこの LDA は，ツイートのような短文においては精度が落ちることが知られている [7]．そのため，マイクロブログの投稿に対する LDA の性能向上を図る手法は多く研究されている．Mehrotra ら [7] の手法では，140 文字以内というツイートの弱点を克服するために，ハッシュタグによってツイートをプーリングしてから LDA の学習を行う．また，ハッシュタグが付加されているツイートが少ないことから，ハッシュタグ付きのツイートとそうでないツイートのコサイン類似度が一定値以上となれば，そのハッシュタグをハッシュタグの付いていないツイートに付加する．Mehrotra らの実験において，この手法によって LDA の性能が向上することが確認された．Kotov ら [8] は，マイクロブログの投稿の本文に加えて，それに付随する位置情報を使用して LDA の学習を行う手法を提案した．Kotov らの手法では，投稿に付随する位置情報（付随しない場合はユーザーのプロフィールに記載の位置情報）によって投稿をラベリング・クラスタリングし，その各クラスタに対して LDA を適用することで，地理的情報に依存したトピックを得る手法を提案した．Zhao ら [9] は，ツイートにおける 140 文字以内という制限から，1 ツイートあたり 1 トピックと考えることでツイートに対する LDA の性能を向上させる Twitter-LDA という手法を提案した．

2.3 本研究の位置付け

このように，本研究と同じ分野において様々な先行研究が存在しているものの，前述の研究の多くは，本文以外の情報も使用しており Twitter に特化したものである．また前述の Lau らの手法のように，本文のみを使用する手法も存在しているが，文書のトピックを考慮していない．本研究では，Doc2Vec によって推定した文書の特徴ベクトルと，Word2Vec によって推定した，その文書中に出現する各語の特徴ベクトルとの類似度によって重み付けを行う．本文以外の情報を使用していないため，特定のサービスに依存した手法ではない．また，重み付けの際に Doc2Vec と Word2Vec を使用しているため，文書のトピックを考慮することが可能である．

そもそも本手法は，1. で述べた仮説に基づいた重み付け手法であるが，この仮説を一般化して述べると以下のようになる．

> 全体との類似度の高い要素ほど，その全体をよく表現しており重要な要素である．

実際，この考えに基づいた研究は他にも存在している．例えば本研究とは少し異なる分野ではあるが，Gong ら [10] の手法がある．これは文書要約に関する研究であるが，ここで提案されている手法の 1 つにおいて，この考え方が使われている．Gong らの手法では，文書を要約する際に，文書 d を文集合

(注6)：http://trec.nist.gov/data/microblog2012.html

$S = (s_1, s_2, \ldots, s_n)$ に分解し，tf によって d，s_i の特徴ベクトル $\boldsymbol{u}$，$\boldsymbol{v}_i$ を生成し，これらの類似度を内積によって計算する．そしてこの類似度によって s_i をランキングし，上位 k 文を，文書 d の要約文集合の候補に加えるという作業を行っている．

また本研究と類似の手法に，Wilson ら [11] が提案した手法がある．この手法では，LDA を用いて文書のトピック分布を推定し，文書におけるトピックの確率と語がそのトピックから生成される確率との積によって重み付けを行う．よって，文書のトピックを考慮し，本文の情報のみを用いて語の重み付けを行うという点において，本研究とは同じである．しかしこの手法では，どのようなトピックからも生成されやすい語の重みが必ず高くなるが，こうした語は，tf-idf では低い重みとなる語である．つまり，重要である可能性が少ないと考えられる．一方，本手法では，文書において重要な語のトピックは，文書全体のトピックに類似するという仮説に基づいた手法であるため，LDA による重み付けの問題点は解消されていると考えられる．

他にも，Doc2Vec や LDA によって文書とクエリのトピック分布をそれぞれ推定し，これらの類似度によって文書をランキングするような手法も考えられる．しかし，この手法にも問題が考えられる．ここで次のような文書を考える．

> I hope they rappinitup bcuz frm dc-bmore alone the hiv/aids EPIDEMIC is so not worth it. Bsides its just nasty; & I'm a dancer

この場合，“epidemic” に関する語が多く含まれているため，この文書のトピック分布は，“epidemic” に関する部分の値が大きくなる．これを次のクエリによってランキングする．

> Bedbug epidemic

この場合，“bedbug” も “epidemic” と関わりのある語と考えられるので，この文書のトピック分布も，“epidemic” に関する部分の値が大きくなる．したがって，2 つのベクトルの類似度は大きくなり，ランキングの順位は上位になると考えられる．しかし上記文書は “budbug” とは何の関係もないため，これは誤りである．本手法では，各語に重みを付けることで得られるベクトルを用いる．したがって，文書には含まれない “bedbug” のスコアは 0 となるため相対的に文書のスコアは小さくなり，順位は下がると考えられる．つまり，提案手法を用いると全てのクエリ語を等しく考慮することが可能となっている．そのため本例のように，大きな粒度のトピックであればクエリ全体と文書とで共通しているが，トピックの粒度を小さくすると文書とは異なるトピックのクエリ語が存在している場合であっても，その異なるクエリ語が文書内に含まれていなければ，順位を低くすることができる．一方，文書をトピック空間上で扱うと，その次元は高々 100 であるため，クエリ側の “Bedbug” のような粒度の低いトピックは考慮されず，“Epidemic” のように比較的粒度の大きいトピックの影響力が強まってしまい，正しくランキングすることができないと考えられる．こうした点から，本手法のほうがより適切であると考えられる．以上の事柄を確かめるために，後述の実験における比較手法として，Wilson ら [11] の手法，Doc2Vec によって文書とクエリの特徴ベクトルそれぞれ推定し，これらの類似度によって文書をランキングする手法，LDA によって文書とクエリのトピック分布をそれぞれ推定し，これらの類似度によって文書をランキングする手法の合計 3 つを用いることとする．

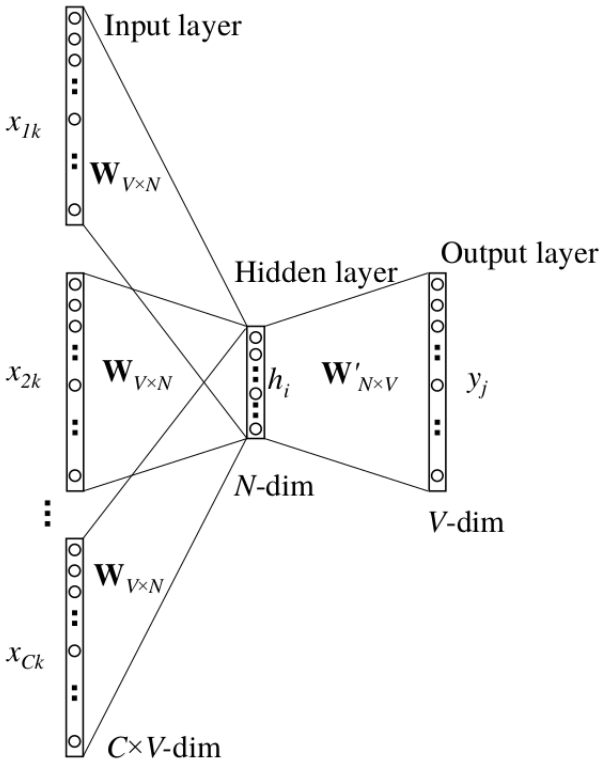

図 1　CBOW のモデル （Xin [12] より引用）

3. 提案手法

この節では，提案手法において使用する Word2Vec [1] と Doc2Vec [2] のアルゴリズムについて解説した後に，提案手法について解説する．

3.1 Word2Vec

Word2Vec には，周辺単語群から該当単語を推定する Continuous Bag-of-Words Model (CBOW) と，該当単語から周辺単語群を推定する Continuous Skip-gram Mode (Skip-gram) の 2 種類のモデルが存在する．本研究では CBOW を用いるため，ここでは CBOW の解説のみを行う．

図 1 は CBOW のモデル図であるが，この図から分かるように，Word2Vec の中身は 3 層のニューラルネットワークである．文書中において連続する語を同時に入力し，出力における学習対象の語の確率が最大となるようなベクトル表現を学習する．以下ではこのニューラルネットワークの学習過程を，入力から順番に見ていくこととする．

コーパス内の語彙を $w_1, w_2, \ldots, w_V$，該当単語の前後合わせて $C-1$ 語を周辺語群とする．入力単語 w_i' は V 次元の One-hot ベクトル $\boldsymbol{x}^w_{w_i'}$ で表現される．入力は

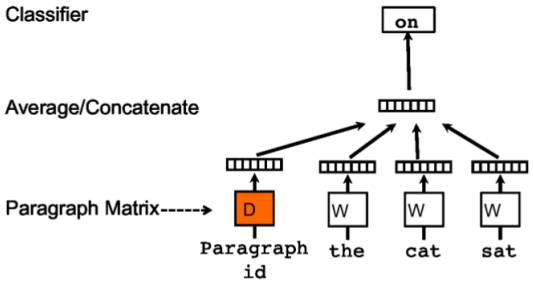

図 2　PV-DM のモデル （Le ら [2] より引用）

$$\boldsymbol{X} = (\boldsymbol{x}^{w}_{w'_1}\ \boldsymbol{x}^{w}_{w'_2}\ \ldots\ \boldsymbol{x}^{w}_{w'_C}) \tag{1}$$

である．入力層 → 隠れ層の重み行列 $\boldsymbol{W}$ は $N \times V$ であり，隠れ層の入力は

$$\boldsymbol{WX} = (\boldsymbol{v}_{w'_1}\ \boldsymbol{v}_{w'_2}\ \ldots\ \boldsymbol{v}_{w'_C}) \tag{2}$$

となる．隠れ層の出力は

$$\boldsymbol{h} = \sum_{i=1}^{C} \boldsymbol{v}_{w'_i} \tag{3}$$

である．隠れ層 → 出力層の重み行列は $\boldsymbol{W^T}$ であるから，出力層の入力は

$$\boldsymbol{y} = \boldsymbol{W^T h} = \begin{pmatrix} \boldsymbol{v}^T_{w_1} \cdot \boldsymbol{h} \\ \boldsymbol{v}^T_{w_2} \cdot \boldsymbol{h} \\ \vdots \\ \boldsymbol{v}^T_{w_V} \cdot \boldsymbol{h} \end{pmatrix} \tag{4}$$

となり，出力は

$$\boldsymbol{z} = \mathrm{Softmax}(\boldsymbol{y}) \tag{5}$$

$$\boldsymbol{z}_i = \frac{\exp(\boldsymbol{y}_i)}{\sum_{k=1}^{V} \exp(\boldsymbol{y}_k)} = p(w'_i | w'_1, \ldots, w'_{i-1}, w'_{i+1}, \ldots, w'_C) \tag{6}$$

となる．最後に，$p(w'_i | w'_1, \ldots, w'_{i-1}, w'_{i+1}, \ldots, w'_C)$ が最大となるように，$\boldsymbol{W}$ を更新する．以上のサイクルを繰り返した後に得られる

$$\boldsymbol{v}_{w_i} = \boldsymbol{W}\boldsymbol{x}^{w}_{w_i} \tag{7}$$

が，w_i の特徴ベクトルとなる．

3.2　Doc2Vec

Doc2Vec には，周辺単語群と文書 ID から該当単語を推定する Paragraph Vector: A distributed memory model (PV-DM) と，文書 ID から文書内に含まれる単語群を推定する Paragraph Vector without word ordering: Distributed bag of word (PV-DBOW) の 2 種類のモデルが存在する．本研究では PV-DM を用いるため，ここでは PV-DM の解説のみを行う．

図 2 は PV-DM のモデルである．図 1 よりも簡略化されているが，中身は Word2Vec の CBOW と同じ 3 層のニューラルネットワークである．したがって，アルゴリズムのほとんどは Word2Vec の CBOW のものと同じである．しかし図 2 から分かるように，入力には周辺語群の他に文書 ID が存在している．これによって，語の特徴ベクトルを学習する時と同じように，文書の特徴ベクトルを学習することが可能となっている．以下ではこのニューラルネットワークの学習過程を，入力から順番に見ていくこととする．

コーパス内の文書を $d_1, d_2, \ldots, d_U$，コーパス内の語彙を $w_1, w_2, \ldots, w_V$ ，該当単語の前後合わせて $C-1$ 語を周辺語群とする．入力層 → 隠れ層の重み行列 $\boldsymbol{W}, \boldsymbol{D}$ は，それぞれ $N \times V, N \times U$ であり，文書 ID d'_j は U 次元の One-hot ベクトル $\boldsymbol{x}^{d}_{d'_j}$ で，入力単語 w'_i は V 次元の One-hot ベクトル $\boldsymbol{x}^{w}_{w'_i}$ で表現される．Word2Vec のアルゴリズムと異なるのは，以下の 2 式のみである．

$$\boldsymbol{h} = \boldsymbol{D}\boldsymbol{x}^{d}_{d'_j} + \sum_{i=1}^{C} \boldsymbol{W}\boldsymbol{x}^{w}_{w'_i} \tag{8}$$

$$\boldsymbol{z}_i = \frac{\exp(\boldsymbol{y}_i)}{\sum_{k=1}^{V} \exp(\boldsymbol{y}_k)} = p(w'_i | w'_1, \ldots, w'_{i-1}, w'_{i+1}, \ldots, w'_C, d'_j) \tag{9}$$

最後に，$p(w'_i | w'_1, \ldots, w'_{i-1}, w'_{i+1}, \ldots, w'_C, d'_j)$ が最大となるように，$\boldsymbol{W}$ と $\boldsymbol{D}$ を更新する．以上のサイクルを繰り返した後に得られる

$$\boldsymbol{u}_{d_j} = \boldsymbol{D}\boldsymbol{x}^{d}_{d_j} \tag{10}$$

$$\boldsymbol{v}_{w_i} = \boldsymbol{W}\boldsymbol{x}^{w}_{w_i} \tag{11}$$

が，それぞれ d_j，w_i の特徴ベクトルとなる．このように，文書 ID が文書全体の文脈を保持する 1 つの語として振る舞うことで，Word2Vec とほぼ同じアルゴリズムを用いて，文書の特徴ベクトルを得ることを可能にしている．

3.3　提 案 手 法

本研究における提案手法は次の通りである．文書 d_j 中の語 w_i の重みを s_{w_i,d_j} とし，文書と語の特徴ベクトルを，それぞれ (10) 式，(11) 式とする．このとき

$$s_{w_i,d_j} = \frac{\boldsymbol{v}_{w_i} \cdot \boldsymbol{u}_{d_j}}{|\boldsymbol{v}_{w_i}||\boldsymbol{u}_{d_j}|} \tag{12}$$

とする．

また式 (12) では，同じ語が文書中に複数回出現することは考慮されない．そのため，本研究では提案手法のバリエーションとして以下を提案する．

$$s_{w_i,d_j} = \frac{\boldsymbol{v}_{w_i} \cdot \boldsymbol{u}_{d_j}}{|\boldsymbol{v}_{w_i}||\boldsymbol{u}_{d_j}|} \cdot \mathrm{tf}(w_i, d_j) = \frac{\boldsymbol{v}_{w_i} \cdot \boldsymbol{u}_{d_j}}{|\boldsymbol{v}_{w_i}||\boldsymbol{u}_{d_j}|} \cdot n_{w_i} \tag{13}$$

ただし，n_{w_i} は文書集合内において語 w_i が出現する文書数とする．

Salton と Buckley [3] は，語に対して適切に重みを付けるために，語の重み付け手法が考慮すべき 3 要素を検討した．その 3 要素は以下のとおりである．

（1） 文書内における語の重要性

（2） 語が持つ，文書集合内における文書識別能力
（3） 文書長

式 (12) も式 (13) は，文書 d_j 内のローカルな基準のみに基づいた重み付け手法である．文書集合内のグローバルな要素も考慮するために，提案手法のバリエーションの 2 つめとして以下を提案する．

$$s_{w_i,d_j} = \frac{\boldsymbol{v}_{w_i} \cdot \boldsymbol{u}_{d_j}}{|\boldsymbol{v}_{w_i}||\boldsymbol{u}_{d_j}|} \cdot \mathrm{idf}(w_i) = \frac{\boldsymbol{v}_{w_i} \cdot \boldsymbol{u}_{d_j}}{|\boldsymbol{v}_{w_i}||\boldsymbol{u}_{d_j}|} \cdot \log_2(\frac{N}{n_{w_i}}) \quad (14)$$

ただし，N は文書集合内の文書数，n_{w_i} は文書集合内において語 w_i が出現する文書数とする．

4. 実　　験

本節では，提案手法の有効性を確かめるために行う実験について述べる．

4.1 概　　要

本実験には実験 1 と実験 2 が存在しているが，どちらの実験においても trec 2011 Microblog Track のデータセットを使用する．予めこのデータセットを用いて，Doc2Vec と LDA のモデルを学習させておく．実験 1 では，クエリ毎にツイートを 4.2 節の各手法を用いてランキングを行い，それぞれ nDCG によって評価，比較する．実験 2 では，2.3 節にて述べた本手法の正当性を検証する．そのために，2.3 節において例として示した文書とクエリとの類似度や，ランキングの順位を調べる．

なお，Doc2Vec や LDA のモデル構築，特徴ベクトルの推定には，Python のライブラリである gensim(注7) を利用する．

4.2 手　　法

本実験において使用する手法は，以下の 6 つである．2.3 節でも述べたが，改めてここで整理しておく．

提案手法 1　式 (12) の手法
提案手法 2　式 (13) の手法
提案手法 3　式 (14) の手法
比較手法 1　文書におけるトピックの確率と，語がそのトピックから生成される確率との積によって重み付けを行う Wilson ら [11] の手法
比較手法 2　文書とクエリの特徴ベクトルを Doc2Vec によってそれぞれ推定し，これらのコサイン類似度によってランキングを行う．
比較手法 3　文書とクエリのトピック分布を LDA によってそれぞれ推定し，これらの JS-divergence によってランキングを行う．
比較手法 4
ランダム　ランダムに文書を並べたものをランキングとする．

KL-divergence は，離散確率分布 $\boldsymbol{p}$，$\boldsymbol{q}$ に対して $KL(\boldsymbol{p}||\boldsymbol{q}) = \sum_i \boldsymbol{p}_i \log \frac{\boldsymbol{p}_i}{\boldsymbol{q}_i}$ となるため，$\boldsymbol{q}_i = 0$ となる場合があると ∞ へ発散してしまう．本実験では，実際にこのような現象が起こりうるため，ここでは JS-divergence を使用する．

(注7)：https://radimrehurek.com/gensim/index.html

表 1　特徴ベクトルの次元数ごとの nDCG@100（入力語数：15）

	提案手法 1	提案手法 2	提案手法 3	比較手法 2
10	0.25440831	0.25279099	0.24707562	0.09957822
25	0.25761855	0.24821565	0.25043788	0.10470439
50	0.24784969	0.25214559	0.24477418	0.09959917
100	0.26198452	0.25666877	0.25616871	0.10239643
200	0.24648823	0.25158932	0.24649039	0.0931755

表 2　入力語数ごとの nDCG@100（次元数：100）

	提案手法 1	提案手法 2	提案手法 3	比較手法 2
10	0.24658349	0.25291932	0.24507438	0.09270935
15	0.26198452	0.25666877	0.25616871	0.10239643
20	0.24922384	0.24920894	0.24984324	0.09652193
30	0.26175858	0.24809502	0.25521608	0.09963189
40	0.25982737	0.25093671	0.25679767	0.10417619

表 3　トピック数ごとの nDCG@100

	比較手法 1	比較手法 3
200	0.11656867	0.18789896
400	0.13560341	0.19983731
800	0.17036754	0.19727132
1200	0.15055062	0.1973041
1600	0.14141588	0.19038578

4.3 データセット

本実験では，trec 2011 Microblog Track のデータセットを使用する．このデータセットは，以下のデータによって構成されている．

- 2011 年 1 月 24 日から 2011 年 2 月 8 日の間に投稿された約 1600 万のツイート
- クエリとそれに対する各ツイートの適合度（ 0，1，2 の 3 段階）の対応表

なお今回は，この約 1600 万ツイートのうち，クエリとの関連度が示されている約 10 万ツイートを使用することとする．また，このデータセットに含まれる適合度の対応表を正解データとして用いる．

4.4 パラメーター

本実験における Doc2Vec と LDA のパラメーターを定めるために，予備実験を行った．予備実験の内容は実験 1 とほとんど同じであるが，ツイートの前処理には，4.5 節に列挙した処理に加えて，ポーターのステミング処理 [13] を採用する．評価には nDCG@100 を使用し，この値が最大となるパラメーターを採用する．

表 1，表 2，表 3 に予備実験の結果を示す．表 1，表 2，表 3 はそれぞれ，Doc2Vec の特徴ベクトルの次元数，Doc2Vec の入力語数，LDA のトピック数の各値と，各手法の各パラメーター値における nDCG@100 との対応を示している．1 行目はパラメーターの値，それ以降の行は nDCG@100 の値である．この予備実験より各パラメーターには，表 4，表 5 の値を採用する．

4.5 前　処　理

ツイートには，他のユーザーへのメッセージを表す部分（例:

表 4　Doc2Vec のパラメーター

特徴ベクトルの次元	100
入力語数	15
学習する語の最低出現回数	2
エポック数	10
その他	gensim のデフォルト値

表 5　LDA のパラメーター

トピック数	800
最小確率	0
その他	gensim のデフォルト値

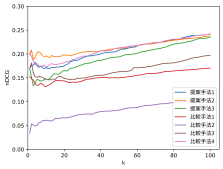

図 3　SWR:有，STM:有

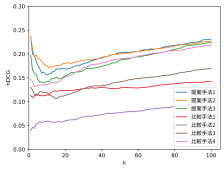

図 4　SWR:有，STM:無

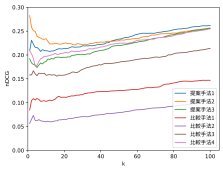

図 5　SWR:無，STM:有

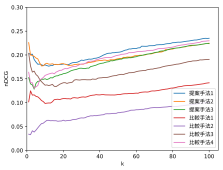

図 6　SWR:無，STM:無

@ladygaga）や URL といった，本文以外の追加情報が含まれている．しかし今回の実験の趣旨として，2.3 節でも述べたように，こうした本文以外の情報は不要である．加えて，Doc2Vec や LDA のモデルの学習精度を下げてしまいかねない要素は排除したい．そのためモデル学習前に，全ツイートに対して以下の処理を行う．

- 全ての文字を小文字化
- タグの除去（例： tag¿）
- 記号の除去
- 連続して出現するホワイトスペース（半角・全角スペースやタブなどを含む）の除去
- 数字の除去
- URL の除去
- 他のユーザーへのメッセージを表す部分の除去（例：@ladygaga）
- リツイートを表す部分の除去（例： RT @ladygaga）
- 英語以外の言語で記述されたツイートの除去

上記処理に加えて，stop word 除去やステミング処理も行うことが一般的である．しかし，これらがランキングに及ぼす影響は手法によって異なると考えられる．ここでは予備実験を行い，各手法において stop word 除去やステミング処理を行うかどうか決定する．実験の内容は実験 1 とほとんど同じである．評価に関しては nDCG@100 を使用し，この値が最大となる処理の組合せを採用することとする．なお，ここで用いるステミング処理は，ポーターのステミング処理 [13] である．

表 6，図 3，図 4，図 5，図 6 に予備実験の結果を示す．表 6

stop word 除去	有	有	無	無
ステミング処理	有	無	有	無
提案手法 1	0.24192605	0.23112926	0.26198452	0.23501813
提案手法 2	0.23843401	0.22706986	0.25666877	0.22472684
提案手法 3	0.23512372	0.22400021	0.25616871	0.22385292
比較手法 1	0.17036754	0.14244705	0.14679245	0.14167576
比較手法 2	0.10382023	0.10058236	0.10239643	0.10097361
比較手法 3	0.19727132	0.16997874	0.21377084	0.1907285
比較手法 4	0.24286583	0.21946339	0.25461981	0.22973489

表 6　各手法の nDCG@100

	stop word 除去	ステミング処理
提案手法 1	無	有
提案手法 2	無	有
提案手法 3	無	有
比較手法 1	有	有
比較手法 2	有	有
比較手法 3	有	有
比較手法 4	無	有

表 7　採用する前処理

は，前処理の組合せと各手法の nDCG@100 の対応を示している．図 3，図 4，図 5，図 6 は，前処理の各組み合わせを適用した際のランキングの nDCG@k を $1 \leqq k \leqq 100$ の範囲で示したものである．なお，“STR” は stop word 除去を，“STM” はステミング処理を表す．

stop word を除去すると，Doc2Vec を用いる 2 手法の性能が低下しているのに対して，LDA を用いる 2 手法では性能が向上していることが分かる．2.3 節において述べたように，“どのようなトピックからも生成されやすい語” が LDA のモデル学習や重み付けに悪影響を及ぼしていると考えられる．一方で，Doc2Vec では連続する C 語を同時に入力するため，stop word のような機能語によって得られる語間関係も学習に利用することで，より正確に文脈を把握しているのではないかと推測される．そのため，stop word を除去することで性能が低下してしまっているのではないかと考えられる．

ステミング処理に関しては，LDA を用いる 2 手法では行う方が性能が良いことが確認された．ステミング処理を行うことで，時制による語尾の変化などによって全く異なる語として認識されるのを抑えることができる．そのため，ステミング処理を行うことでトピックの推定精度が向上していると考えられる．しかし Doc2Vec を用いる 3 手法においては，提案手法では行うほうが性能が向上する一方，比較手法 2 では行わないほうが性能が向上した．Doc2Vec では stop word と同様に，ステミング処理によって失われる情報も学習に利用し，より正確に文脈を把握していると考えられる．しかし提案手法では語に重みを付けるため，クエリ語と時制などまで完璧に一致していない限り，語のスコアは常に 0 となる．必要以上に不適合と判定される文書が増加し，性能が低下していると考えられる．

よって，本節の最初に列挙した処理に加え，表 7 で示した組合せの処理も前処理として採用する．

4.6 実　験　1

本節では，4.1 節で述べた実験 1 の詳細とその結果，考察を述べる．

4.6.1 ランキング方法

あらかじめ，4.3 節のデータセットに含まれるツイートに前処理を行う．Doc2Vec と LDA のモデルは前処理済みのツイートを用いて学習させ，このモデルによってツイートと語の特徴ベクトルを推定する．そしてデータセットに含まれるクエリを使用し，前処理済みのツイートを検索，ランキングする．ランキングの際には，ツイートのベクトルとクエリのベクトルとのコサイン類似度もしくは JS-divergence を計算し，この値をツイートのスコアとすることでランキングを行う．こうして得られたランキングを nDCG@k によって比較，評価する．

なお，提案手法や Wilson ら [11] の手法を用いてランキングする際のツイートのベクトルは，これらの手法によって計算された語の重みが，対応する次元の要素となるようなベクトルとする．クエリは Binary 方式で重み付けしベクトル化する．つまり，クエリ語に対応する次元の要素が 1 で，その他の要素は 0 となるベクトルである．

4.6.2 評 価 指 標

この実験では，ツイートをクエリとの類似度によってランキングを行う．また正解データには，ツイートとクエリの適合度が 3 段階で示されている．そのため，normalized Discounted Cumulative Gain (nDCG) を用いて各手法によって作成されたランキングを評価する．

nDCG は Jrvelin ら [14] が提案した手法で，多値適合性と検索順位を考慮することで性能評価を行う手法である．ランキング k 位までの nDCG（以下 nDCG@k ）は，DCG@k と理想的なランキングにおける DCG@k (これを iDCG@k とする) の 2 つより計算される．なお，

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{15}$$

であり，rel_i は，ランキング i 位の文書の適合度である．本実験においては，$rel_i = 0, 1, 2$ であり，$rel_i = 0$ は不適合，$rel_i = 1, 2$ は適合である．この時，

$$\mathrm{nDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{iDCG@}k} \tag{16}$$

となる．

4.6.3 結　　果

実験の結果を図 4.6.3 に示す．この図は，クエリ毎に作成されたランキングの nDCG@k の平均値と，k の値の関係を $1 \leqq k \leqq 100$ の範囲で示したものである．縦軸が nDCG@k の値，横軸が k の値を表している．

この実験により，提案手法はどちらも，比較手法よりもショートテキスト検索の性能が良いことが確かめられた．しかし提案手法同士を比較すると，提案手法 2 のほうが性能が悪いことが分かる．提案手法 2 では，語重み付けの 3 要素 [3] を満たすことを目的として idf を組合せているが，実際は，Doc2Vec の学習時にコーパス全体を使用するため，Doc2Vec によって得られた特徴ベクトルには，既にコーパス全体のグローバルな基準が含まれていると考えられる．そのため提案手法に idf を組み合わせると，このグローバルな基準が重複してしまい，性能の低下に繋がったと推測される．

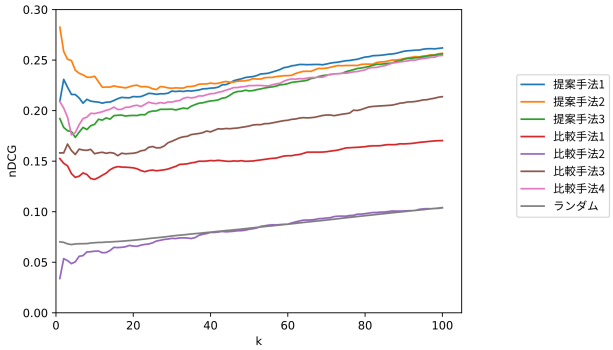

図 7　実験 1 の結果

また今回の実験において，提案手法は比較手法に比べて性能が良いことが確認されたが，$1 \leqq k \leqq 100$ における nDCG@k の値は，全て 0.26 以下となっている．さらに，全体的な傾向として nDCG@k の値が右肩上がりである．つまり，ランキングの上位に位置する関連文書が少ないということを示しており，提案手法の性能にはまだ課題が残っている．考えられる改善方法としては，文書の特徴ベクトルの推定精度を上げることがある．そのための手段として，例えば，Doc2Vec のモデルを学習させる際にツイートの集合を使用するのではなく，web ページの集合を使用するという方法が考えられる．ツイートには 140 文字，もしくは 280 文字以内という制限があるが，web ページにはこのような制限がないため文書長が長い．そのため，より効果的にモデルを学習させることができ，提案手法の性能が向上すると考えられる．

4.7 実　験　2

本節では，4.1 節で述べた実験 2 の詳細とその結果を述べる．

4.7.1 内　　容

本実験では提案手法 1 と比較手法 2 を比較することで，2.3 節において述べた

> クエリ語同士のトピックが近いようなクエリを用いて，クエリ中のある語のトピックとは類似するが，他のクエリ語とは関係がないような文書を検索する場合に，単純にトピック分布同士の類似度によってランキングする手法では，正しくランキングすることができないと考えられる．こうした点から，本手法のほうがより適切であると考えられる．

を検証する．検証には 2.3 節において使用した

> I hope they rappinitup bcuz frm dc-bmore alone the hiv/aids EPIDEMIC is so not worth it. Bsides its just nasty; & I'm a dancer

を文書として使用し，クエリには

> Bedbug epidemic

表 8　実験 2 の結果

	クエリとの類似度	順位
提案手法 1	0.14436401279280694	88
比較手法 2	0.9097659	2

を使用する．Doc2Vec のモデルは，4.3 節のデータセットを使用し，4.4 節の通りにパラメーターを設定して学習させる．そして，提案手法 1 と比較手法 2 によってクエリとの類似度をそれぞれ計算し，結果を比較する．

4.7.2 結　　果

実験の結果を表 8 に示す．クエリとの類似度，ランキングにおける対象文書の順位共に，提案手法 1 のほうが比較手法 2 よりも低くなっていることが確認された．2.3 節において述べたように，提案手法を用いると全てのクエリ語を等しく考慮することが可能となっている．そのため本実験のように，大きな粒度のトピックであればクエリ全体と文書とで共通しているが，トピックの粒度を小さくすると文書とは異なるトピックのクエリ語が存在している場合であっても，その異なるクエリ語が文書内に含まれていなければ，順位を低くすることができている．一方，比較手法 2 では文書をトピック空間上で扱うが，その次元は高々 100 である．そのため，クエリ側の “Bedbug” のような粒度の低いトピックは考慮されず，“Epidemic” のように比較的粒度の大きいトピックの影響力が強まってしまっていると推測される．

5. 結　　論

本論文では， Twitter における各投稿のようなショートテキスト中の語に対しても，適切に重み付けを行うことが可能な手法を提案した．この提案手法では，文書において重要な語のトピックは，文書全体のトピックに類似するという仮説に基づき，Word2Vec と Doc2Vec を用いて語と文書の特徴ベクトルをそれぞれ推定し，これらのコサイン類似度を計算することによって，文書中の各語に重み付けを行った．

また本論文では，提案手法による語の重み付けの性能を確認するために，trec 2011 Microblog Track のデータセットを使用し，ツイート検索・ランキングをタスクとする評価実験を行った．その結果，ツイートのようなショートテキストの検索に対して，提案手法は有効であることを確認した．一方で，提案手法の性能が絶対的には高くないことを確認した上で，その原因と改善案を考察した．また，2.3 節において述べた本研究の位置付けについても検証を行った．

今後は，4.6.3 節にて述べた改善案を用いて実験し，その性能を確認するとともに，データセットをサブセットではなく，1600 万ツイート全てを使用して実験を行う予定である．また提案手法は，ショートテキストに限らず，web ページのような一般的な文書に対しても適用可能な手法であるため，こうした文書に対する性能を測定する実験も行う予定である．

謝　　辞

本研究は，JST，CREST（#JPMJCR16E3）の支援を受けたものである．

文　　献

[1] Mikolov, Tomas, et al. “Efficient estimation of word representations in vector space.” arXiv preprint arXiv:1301.3781 (2013).
[2] Le, Quoc, and Tomas Mikolov. “Distributed representations of sentences and documents.” Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014.
[3] Salton, Gerard, and Christopher Buckley. “Term-weighting approaches in automatic text retrieval.” Information processing & management 24.5 (1988): 513-523.
[4] Luo, Zhunchen, et al. “Improving Twitter Retrieval by Exploiting Structural Information.” AAAI. 2012.
[5] Lau, Cher Han, YueFeng Li, and Dian Tjondronegoro. “Microblog Retrieval Using Topical Features and Query Expansion.” TREC. 2011.
[6] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. “Latent dirichlet allocation.” Journal of machine Learning research 3.Jan (2003): 993-1022.
[7] Mehrotra, Rishabh, et al. “Improving LDA topic models for microblogs via tweet pooling and automatic labeling.” Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013.
[8] Kotov, Alexander, Yu Wang, and Eugene Agichtein. “Leveraging geographical metadata to improve search over social media.” Proceedings of the 22nd International Conference on World Wide Web. ACM, 2013.
[9] Zhao, Wayne Xin, et al. “Comparing twitter and traditional media using topic models.” European Conference on Information Retrieval. Springer, Berlin, Heidelberg, 2011.
[10] Gong, Yihong, and Xin Liu. “Generic text summarization using relevance measure and latent semantic analysis.” Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2001.
[11] Wilson, Andrew T., and Peter A. Chew. “Term weighting schemes for latent dirichlet allocation.” human language technologies: The 2010 annual conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010.
[12] Rong, Xin. “Word2Vec parameter learning explained.” arXiv preprint arXiv:1411.2738 (2014).
[13] Porter, Martin F. “Snowball: A language for stemming algorithms.” (2001).
[14] Jrvelin, Kalervo, and Jaana Keklinen. “Cumulated gain-based evaluation of IR techniques.” ACM Transactions on Information Systems (TOIS) 20.4 (2002): 422-446.