学　士　学　位　論　文

題　　目

マイクロブログテキストを用いた教師なし文書間類似度評価手法の分析

指 導 教 員

鬼塚真教授

報　告　者

近井厚三

平成 28 年 2 月 19 日

大阪大学工学部　電子情報工学科

卒業研究報告書

マイクロブログテキストを用いた教師なし文書間類似度評価手法の分析

近井厚三

内容梗概

Twitter に代表されるマイクロブログ型の SNS が急速に普及している．通常のブログより，文章量の少ない投稿，かつ，人との情報コミュニケーションに特化しており，リアルタイム性があり実世界を表現する鮮度の高い情報源となっている．自然言語処理分野において，このようなマイクロブログ型のテキストデータの分析精度の向上は急務であると考えられている．テキストデータの分析方法の一つとして，テキストデータ間の差異を観測する文書間類似度が挙げられる．文書間類似度計算の精度向上は，ニューステキストの推薦システムや，人工知能の自動応答システムの品質向上に繋がるものである．特に自動応答システムは，対話ロボットやカーナビ，自動案内の音声システム等のアプリケーションに応用されており，今後需要のある分野の一つとして数えられている．

一般的に，文書間類似度は各文書の単語の共起頻度などを用いて計算される．しかし，マイクロブログ型のテキストデータは一つの投稿が短文であり，このような単語数 (情報量) の少ない短文間の類似度を計算することは，単語の共起頻度が十分でない分，通常の文章の類似度を計算を行うよりも精度が落ちてしまう．

現在，自然言語処理分野において文書間の類似度は様々なアプローチが存在している．本報告では，そのような類似度計算手法を用いて，入力となるツイートが与えられた時の適切な応答を検索・出力するシステムを実装した．実装したシステムを使用し，実際にどのような類似度計算手法が自動応答システムに寄与するかを調査した．具体的には，TF-IDF を用いた bag-of-words，LDA に代表されるトピックモデル，行列分解による低近似ベクトル生成，ニューラルネットワークを用いた単語・文章のベクトル生成 (word2vec，doc2vec 等) を比較している．

結果として，word2vec で類似する単語が含まれている短文を取り出し，それを TF-IDF で類似度計算する方法が最も評価値が高かった．これは最初のステップがフィルタリングのような役割を果たし，純粋な TF-IDF 計算のみよりも，正解データが上位にランキングされ易かったと考えられる．一方で，トピックモデルや doc2vec は軒並み結果が低い値となった．これは，今回用いたデータが Twitter のテキストデータであったため，データの疎性により上手くベクトル空間にモデリングされなかった事が原因と考えられる．

キーワード

マイクロブログ，Twitter，ショートテキスト，文書間類似度，自動応答

目 次

第1章　序論

近年，特に需要が高まっている媒体の一つとしてSNS (Social Networking Service) がある．SNSは他人とコミュニケーションをとる事が主な目的の媒体であり，ソーシャルメディアの一種である．図1.1に日本人SNS利用者数の年代順にまとめたグラフを示す．グラフはICT総研[1]が2015年度の7月29日にSNS利用動向に関する調査結果をまとめたものの一つである．

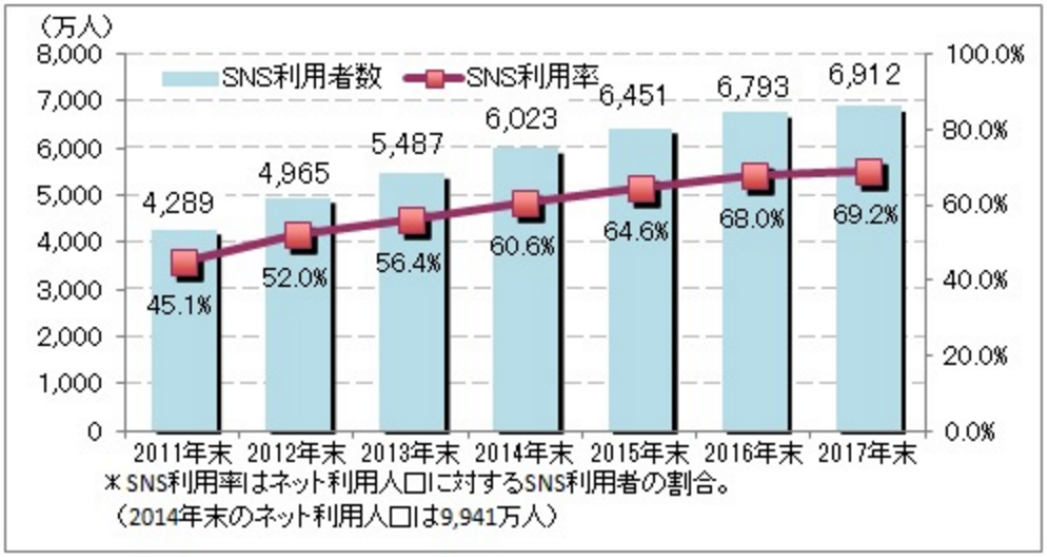

図1.1: 日本におけるSNS利用者数[2]

このグラフによると，日本でのSNS利用者数は年々増加の一途を辿っており，2015年の末には利用者数は6,451万人に達すると見込まれている．また，2014年末での日本国内のインターネットユーザーは9,941万人と概算されているが，SNS利用者はその内，60.6％にあたる6,023万人に登るとされている．2015年の年間純増者数は428万人となる見込みで，１ヶ月平均で約36万人の利用者が増加を続けている．

上記のように年々ユーザ数を伸ばしているSNSであるが，総務省情報通信政策研究所[3]

[1] http://ictr.co.jp/
[2] http://ictr.co.jp/report/20150729000088-2.html
[3] http://www.soumu.go.jp/iicp/

の調査では，日本国内では主に LINE[4]，Twitter[5]，Facebook[6] の三つの SNS サービスの利用率が上昇しているという結果が提出されている．SNS 利用率上昇の参考として，国内の主なソーシャルメディアの利用率を表したグラフを図 1.2 に示す．

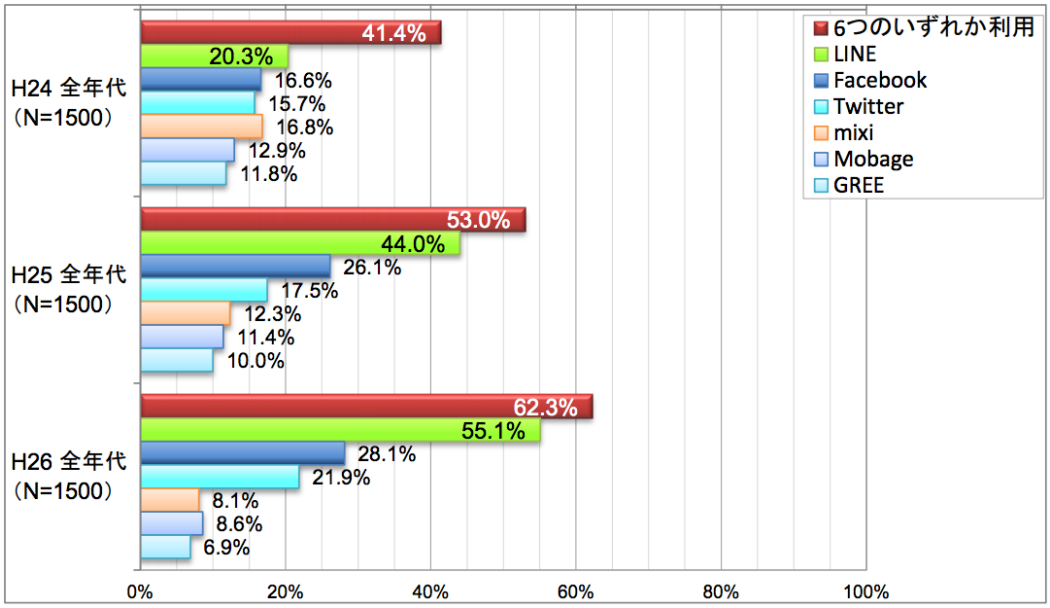

図 1.2: 日本での主なソーシャルメディア利用率 [7]

図 1.2 によると，他の国内発 SNS を抑え，海外発サービスである LINE，Twitter，Facebook の三つの利用率が年を追う毎に増加していることがわかる．

本報告では，特に Twitter 等のマイクロブログに注目する．マイクロブログとは，ブログサービスの一種であり，利用者は現在の心境や状況，雑記等の短文をウェブサイトに投稿することができる．またユーザ間のコミュニケーションも自由に取ることができる．このようなシステムにおいて，ユーザが発する情報は基本的に“今”起こっている出来事や思っている感情であることが多い．従って，この膨大なデータを収集・分析することで，現在の情勢や流行等をテキストで捉えることができる．具体例として，NHK の番組「NEWS WEB」[8] のコーナーのひとつである“つぶやきビックデータ”が挙げられる．このコーナーでは，NTT データ [9] が解析したデータから，前日に比べてツイートされた数が急増した単語を紹介する．このように，集めたテキストデータを分析する事で，現在のトレンドを調べる事が可能となる．

[4] http://line.me/ja/
[5] https://twitter.com/
[6] https://ja-jp.facebook.com/
[7] 総務省情報通信政策研究所　平成 26 年情報通信メディアの利用時間と情報行動に関する調査報告書 http://ictr.co.jp/report/20150729000088-2.html
[8] http://www3.nhk.or.jp/news/newsweb/
[9] http://www.nttdata.com/jp/ja/index.html

また，マイクロブログにおいてユーザが発する情報は出来事だけでなく，他のユーザの投稿に対して反応・返信したテキスト (reply) も含まれている．本研究は NTCIR [10](NII Testbeds and Community for Information access Research) が推進している STC [11](Short Text Conversation) Japanese Task というタスクに参加している．このタスクでは，投稿文と返信文のペアを大量に収集し，自動応答システムに応用する事を提案している．現在の自然言語処理技術を用いても，コンピュータにより自然な応答を生成するのは困難である．そこで自動応答システム実現の第一歩として，入力文と最も類似している文を過去の投稿・返信文データから検索し，それに対する返信を出力することで，自動応答システムを構築する．システムの概要を図 1.3 に示す．

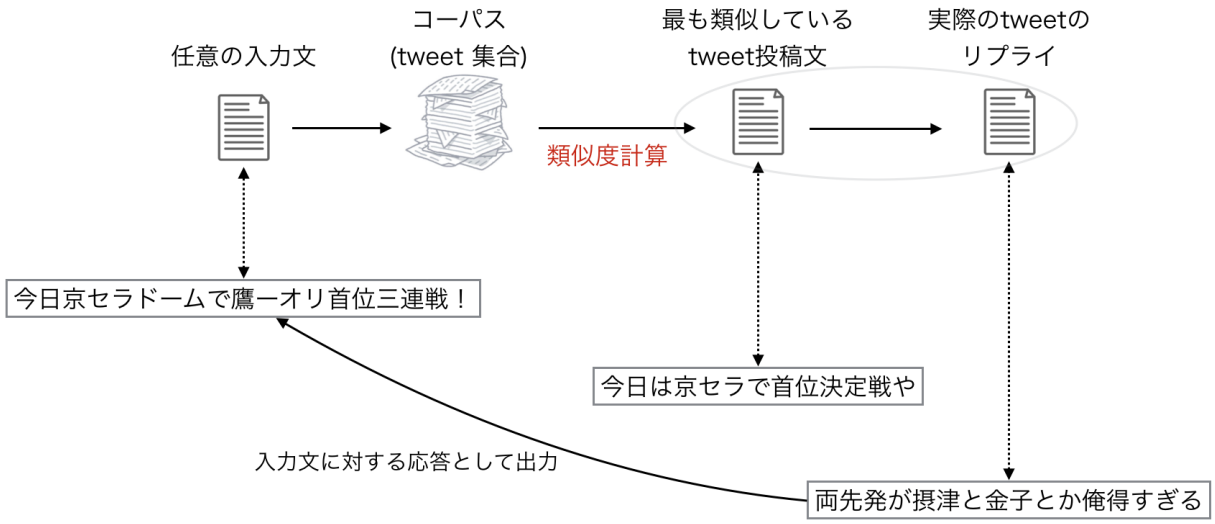

図 1.3: 応答文検索システムのイメージ図

図 1.3 のようなシステムを構築する事で，入力文に対し，似ている短文 (ツイート) をコーパスから抽出し，それに返信しているツイートを応答として出力することによって応答システムとして使用する事ができる．

図 1.3 のシステムを構築する際，文書間類似度を高精度で計算することが重要になってくる．理由として，このシステムでは類似度計算のみで類似ポストを判定しており，少しでも似ていない投稿文が抽出されてしまうと，それに対するリプライが期待していたものと大幅に異なる恐れがあるためである．ここで，Twitter のような短文のデータを分析する際，テキストデータの疎性が非常に大きな問題となる．テキスト間の類似度は，テキストを何かしら適当なベクトル空間に変換し，生成されたベクトルの距離を計算することで求める事ができる．このベクトル空間は，一般的に出現する単語を元に生成されるため，テキストは高次元ベクトルとなる．もしテキストが短文であれば，単語の共起頻度が非常に少な

[10] http://research.nii.ac.jp/ntcir/index-ja.html
[11] http://ntcir12.noahlab.com.hk/japanese/stc-jpn.htm

くなってしまい，高精度な類似度計算ができなくなってしまう．そこで本研究では，現在，自然言語処理分野で使用されている代表的なテキストのベクトル生成手法と類似度計算手法の短文における有効性を検証する．実際の Twitter のテキストデータを用いて，上記に述べたような自動応答システムを各々の類似度計算技術を用いて構築し，評価することで，類似度計算の違いが出力される応答に与える影響を分析した．

第2章　関連研究

自動応答システムを考えるにあたって，自然言語処理分野の中では CQA (Community-based Question Answering) における自動回答システムに関する研究が関連が深い．CQA とは，Yahoo!知恵袋 [1] に代表されるコミュニティ型質問投稿サイトのことであり，ユーザが様々な質問を投稿し，投稿された質問に対し，任意の回答者が回答を投稿するサイトである．これらに関する研究として，過去の QA データを機械に学習させて，入力された質問文から回答を自動的に導き出すタスクが挙げられる．用いられる情報はテキストデータに限らず，質問投稿者・回答者などのユーザ情報や，分類カテゴリ，どの質問に対する回答か，などというサイトのデータ構造も使用する．例えば，Gao Cong ら [1] は，コミュニティから質問を探し当てる方法 (question detection) として系列ラベリングを使用し，また回答を選択する方法 (answer detection) として，PageRank[2][3] アルゴリズムを基にしたグラフベースのランキング手法を提案している．その結果，従来の単語の共起頻度のみを使用した類似度計算に加え，データ構造を利用したグラフ情報も適応する方が，回答検索の精度が向上した．また，Chirag Shah ら [4] は，CQA サイトにおける回答の質の評価を複数の項目に分け (適切性，専門性，信頼性，客観性等)，それを人手で判定し，ロジスティック回帰分析を用いて，実際の評価値と比べることで，どの項目が回答の質に大きく関わっているか調査した．また，CQA から複数のデータ (タイトル長，本文長，参照，ユーザプロフィール等) を自動抽出し，ロジスティック回帰分析から，どの項目が自動で回答選択を行うにあたって重要な要因となるかを調査した．

CQA に関する研究は他にも多く存在している [5][6]．しかし，CQA で取り上げられるテキストは基本的に複数の文から構成されており，一つのテキストの情報量が多く含まれている点において今回の研究とは大きく異なる．さらに本研究では Twitter のテキストデータ (一つの文書が 140 字以内の短文) を用いるため，サイトのデータ構造は使用することができない．

今回データとして用いる Twitter に関する研究としては，主に Twitter のデータ構造 (ユーザのフォロー・フォロアー関係，位置情報，投稿時刻，ハッシュタタグ等) を利用した研究が多く存在している．Rui Yan ら [7] は，ツイート間類似度を使用した無向グラフ，ユーザのフォロー関係を利用した有向グラフの二つのグラフを利用したランキングアルゴリズム

[1] http://chiebukuro.yahoo.co.jp/

を提案し，ツイートの推薦システムを提案している．Wayne Xin Zhao ら [8] は，トピックモデルである LDA[9] を改良した Twitter-LDA モデルを提案し，ツイートのトピックは一つであると仮定した方がトピック同定の精度が高いことを示している．また Twitter-LDA を用いてトピック集合を検索し，トピックに対して topical PageRank を使用し，トピックの中の重要語を抽出する手法を提案している [10].

上記で述べた手法はフォロー関係・ハッシュタグ等のツイッター特有のデータ構造を用いてシステムを構築している．しかし，短文応答のペア (テキストデータ) のみで手法を構築した方がデータ構造に依存しないため，手法の汎用性が高く，他の SNS やアプリケーションに応用が容易である．そこで本報告では，上に紹介したような Twitter 特融のデータ構造は用いず，文書間の類似度計算に注目した．類似度計算手法として標準的に用いられているモデルとして，bag-of-words に代表されるテキストのベクトル化を用いた類似度計算がある．ベクトルの生成は bag-of-words に限らず，pLSI[11] や LDA[9] に代表されるようなトピックモデルによるベクトル推定法も存在する．また，Weiwei Guo ら [12][13] は行列分解 [14] を利用した WTMF (Weighted Text Matrix Factorization) モデルを提案しており，トピックモデル等よりも情報の疎性に対応した分，文書間類似度の精度が高い事を示している．一方で，文書ベクトルの疎性を改善するために，他のコーパスを用いてベクトルの特徴量を増やす手法も存在している．例えば，E. Gabrilovich ら [15] は，Wikipedia[2] のコーパスを用いて文書ベクトルの生成する手法，ESA(Explicit Semantic Analysis) を提案している．近年では，ニューラルネットワークを用いた単語ベクトル生成手法も存在しており，Mikolov Tomas ら [16][17][18][19] が提案した word2vec や，Jeffrey Pennington[20] らが提案した GloVe が有名である．本研究ではこれらの類似度計算手法がツイートのような短文に対してどれだけ有効か比較検証する．

[2] https://ja.wikipedia.org/wiki/

第3章　類似度推定手法

3.1　ベクトルによる文書表現

本研究で自動応答システムを構築する際，入力ポストに対する類似ポストを見つける必要がある．このような日本語短文の情報検索を行うため，大量の短文データをベクトル空間モデルで表現する．ベクトル空間モデルとは情報を検索する際の手法の一つであり，このベクトル空間モデルを使うことで、文章の類似度をベクトル間の距離や内積によって評価できる．

3.1.1　ベクトル空間モデル

言語データをベクトル空間モデルで表現するには，一般的に bag-of-words を使用した単語文書行列が使用される．bag-of-words とは単語の並び順は考慮せず，文書における単語の出現回数のみを観測して数値化したモデルである．また，単語文書行列とは言語コーパスの表現方法の一つであり，行列における行を単語ベクトル，列を文書ベクトルとして表現する．

$$Corpus = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} \tag{3.1}$$

行列内の各セルには一般的には単語の出現頻度が用いられる．具体的には，行 i に対応する単語で列 j に対応する文書内の出現頻度は，行列内のセル c_{ij} に表現される．実際にベクトルを使った類似度算出手法はベクトルの内積や距離が用いられるが，計算方法は各ベクトル空間モデルに依存するため，詳細は 3.3 節で取り上げる．

3.1.2 日本語の単語分割

日本語の文章は英語等の印欧語と異なり, 単語毎のスペースが存在しないため, コンピュータが単語の区切りを理解できない限り，単語を文書から取り出すことができない．そのため日本語やタイ語，中国語などの単語分割スペースのない (分かち書きのされていない) 言語は，ベクトル表現の際に形態素解析という前処理が必要となる．形態素解析とは，機械が単語の区切りを判定し品詞をタグ付けする操作を指す.

2016 年現在，入手可能な日本語のフリー形態素解析エンジンとして Chasen[1]，JUMAN[2], KAKASI[3], KyTea[4], MeCab[5] 等が挙げられる．本報告では MeCab を形態素解析エンジンとして採用した．MeCab はオープンソースの形態素解析エンジンで，解析モデルとして bi-gram マルコフモデル，学習モデルに CRF (Conditional Random Fields) を採用しており，従来のフリー解析エンジンに比べて高速かつ高精度な解析を実現している．MeCab の出力フォーマットは

表層形　品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用形, 活用型, 原形, 読み, 発音

のようになっており，品詞タグのみでなく，原形や発音等も出力されるようになっている．MeCab の出力例を “MeCab 出力例 (1) ” に挙げておく．

また，この形態素解析エンジンを用いることで，単語を基本形に戻す見出し語化も行うことが可能となる．例えば，例に挙げた文で “積もり ” という単語が抽出されているが，その動詞の基本形は “積もる ” であり，“積もる ” という単語には “積もら ” (ない), “積もれ ” (ば) など，派生形が存在する．この単語を単一の素性 “積もる ” とみなす操作のことを見出し語化という．MeCab では動詞や形容詞，形容動詞の単語に原形がタグ付けされて出力されているので，その単語を用いて見出し語化することができる．

[1] http://chasen-legacy.osdn.jp/
[2] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN
[3] http://kakasi.namazu.org/
[4] http://www.phontron.com/kytea/index-ja.html
[5] http://taku910.github.io/mecab/

MeCab 出力例 (1)

西日本の太平洋側など普段雪の少ない地方も雪が積もり、大雪の恐れがあります。

```
西日本　名詞, 固有名詞, 地域, 一般,*,*, 西日本, ニシニホン, ニシニホン
の　助詞, 連体化,*,*,*,*, の, ノ, ノ
太平洋　名詞, 固有名詞, 地域, 一般,*,*, 太平洋, タイヘイヨウ, タイヘイヨー
側　名詞, 接尾, 一般,*,*,*, 側, ガワ, ガワ
など　助詞, 副助詞,*,*,*,*, など, ナド, ナド
普段　名詞, 副詞可能,*,*,*,*, 普段, フダン, フダン
雪　名詞, 一般,*,*,*,*, 雪, ユキ, ユキ
の　助詞, 格助詞, 一般,*,*,*, の, ノ, ノ
少ない　形容詞, 自立,*,*, 形容詞・アウオ段, 基本形, 少ない, スクナイ, スクナイ
地方　名詞, 一般,*,*,*,*, 地方, チホウ, チホー
も　助詞, 係助詞,*,*,*,*, も, モ, モ
雪　名詞, 一般,*,*,*,*, 雪, ユキ, ユキ
が　助詞, 格助詞, 一般,*,*,*, が, ガ, ガ
積もり　動詞, 自立,*,*, 五段・ラ行, 連用形, 積もる, ツモリ, ツモリ
、　記号, 読点,*,*,*,*,、, ,、, ,、
大雪　名詞, 一般,*,*,*,*, 大雪, オオユキ, オーユキ
の　助詞, 連体化,*,*,*,*, の, ノ, ノ
恐れ　名詞, 一般,*,*,*,*, 恐れ, オソレ, オソレ
が　助詞, 格助詞, 一般,*,*,*, が, ガ, ガ
あり　動詞, 自立,*,*, 五段・ラ行, 連用形, ある, アリ, アリ
ます　助動詞,*,*,*, 特殊・マス, 基本形, ます, マス, マス
。　記号, 句点,*,*,*,*,。, ,。, ,。
EOS
```

本実験ではこの形態素解析エンジンを用いてツイートを形態素解析するが，ここで１つ問題が生じる．それは固有名詞の存在である．例えば，“やしきたかじんさんの番組を観てる。”という文を MeCab に入力した時の結果を例とすると，以下の出力例 (2) のような結果が得られる。

MeCab 出力例 (2)

やしきたかじんさんの番組を観てる。

やし　助動詞,*,*,*, 特殊・マス, 命令ｉ, やす, ヤシ, ヤシ
き　動詞, 自立,*,*, カ変・クル, 連用形, くる, キ, キ
た　助動詞,*,*,*, 特殊・タ, 基本形, た, タ, タ
か　助詞, 副助詞／並立助詞／終助詞,*,*,*,*, か, カ, カ
じん　名詞, 一般,*,*,*,*, じん, ジン, ジン
さん　名詞, 接尾, 人名,*,*,*, さん, サン, サン
の　助詞, 連体化,*,*,*,*, の, ノ, ノ
番組　名詞, 一般,*,*,*,*, 番組, バングミ, バングミ
を　助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
観　動詞, 自立,*,*, 一段, 連用形, 観る, ミ, ミ
てる　動詞, 非自立,*,*, 一段, 基本形, てる, テル, テル
。　記号, 句点,*,*,*,*,。,。,。
EOS

出力例 (2) からもわかるように “やしきたかじん ” という固有名詞を Mecab が認識できずに, 形態素解析が失敗していることがわかる．Twitter の文章は例に挙げたようなくだけた口語体の文章や新語が多く存在しており，小説や新聞などの文章よりも形態素解析を行うのが困難である．

形態素解析の精度を向上させるために，本研究では Wikipedia とはてなキーワード [6] の単語を Mecab の単語辞書に加えた．Wikipedia やはてなキーワードの登録単語には，通常の辞書に載っていないような新語や口語，固有名詞が多く掲載されているため，ツイートの解析に有効であると考えた．Mecab の単語辞書とは言語を分解、品詞を判定する際に必要とする文書知識であり，具体的には品詞等の情報付きの単語リストのことである．今回はこのリストに Wikipedia とはてなキーワードの単語を追加することで固有名詞や造語等を Mecab に認識させることにした．なお，配布リストには一般的な熟語なども含まれているので登録する単語長を 3 文字以上 10 文字以下を満たす単語とした．品詞は全て名詞とした．結果として，1,469,947 単語を新たに Mecab の辞書に追加した．Wikipedia とはてなキーワードの単語を追加した後の Mecab の出力例を次に示す．先程，失敗した “やしきたかじん ” という固有名詞の抽出に成功していることが分かる．

[6] http://d.hatena.ne.jp/keyword/

Mecab 出力例 (3)

やしきたかじんさんの番組を観てる。

やしきたかじん	名詞, 一般,*,*,*,*, やしきたかじん,*,*,wikipedia
さん	名詞, 接尾, 人名,*,*,*, さん, サン, サン
の	助詞, 連体化,*,*,*,*, の, ノ, ノ
番組	名詞, 一般,*,*,*,*, 番組, バングミ, バングミ
を	助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
観	動詞, 自立,*,*, 一段, 連用形, 観る, ミ, ミ
てる	動詞, 非自立,*,*, 一段, 基本形, てる, テル, テル
。	記号, 句点,*,*,*,*,。,。,。
EOS	

3.2 文書ベクトル生成手法

本章では，本研究で比較する TF-IDF (3.2.1 節) ，トピックモデル (3.2.2 節) ，行列分解モデル (3.2.3 節) ，単語の分散表現 (3.2.4 節) の各種文書ベクトル生成手法について述べる．

3.1.1 節で述べた bag-of-words は，セルに入力される数値は単純な単語の出現回数ではなく，TF-IDF と呼ばれるの単語の重み付けをした数値を入力する事が一般的である．そこで本研究では，TF-IDF を用いた bag-of-words ベクトルを用いる．

Bag-of-words は単語の出現でベクトルを作成しており，文書ベクトル同士の類似度はベクトル内の単語共起で決定される．例えば，“車でスーパーに行った。” と “自動車で買い物に出かけた。” という二つの文書は意味的には類似しているが，単語共起という観点から見ると，この二つの文書は類似していないものとされる．このような表記ゆれに対応するには文書ベクトルの生成方法の変更が必要である．

この問題に対応する手法として，トピックモデルが挙げられる．トピックモデルは文書トピックと単語の生起を何かしらの確率分布で仮定するモデルである．bag-of-words において，単語の生起は独立事象として捉えられているが，トピックモデルはトピックから単語の生起が決定されるので，単語出現に偏りを生じさせることが出来る．また，各単語によるベクトル表現ではなく，トピックによるベクトル表現に次元縮退する事により，“車 ”, “自動車 ”などの類義語も集約可能である．今回トピックモデルとして，LDA[9] と HDP[21] の二種類を採用した．これらを用いることで，生データから単語トピックを抽出し，それを用いて類似度の精度を向上させる可能性があると考えた．

トピックモデルに近い手法として，行列分解モデルが存在する．単語の文脈行列を作成

し，その行列を低ランクの行列に分解するモデルである．代表するモデルとして，SVD (特異値分解) [14] や NMF (非負値行列因子分解) [22]，PCA (主成分分析) [23] が挙げられる．行列分解モデルもトピックモデルと同様に，高次元の文書ベクトルを情報量を減少させずに低次元近似させる事ができる．今回はそのモデルの一つとして WTMF モデル [12] を採用した．WTMF モデルは非観測語を情報に取り入れた手法となっており，短文のようなスパースなベクトルを上手く次元縮退させることが示されている．

さらに, 単語を表現するベクトルをその周辺の単語からニューラルネットを用いて学習し，ベクトルを生成する手法も提案されており，単語の分散表現と呼ばれている．Bag-of-words と比べて周辺の単語を観測するため，コーパス内の単語間の特性を正確に表現できると期待される．出力される単語ベクトルは，単語間の距離から単語をトピックにまとめることや，単語ベクトルを新たに情報として加えることで類似度の推定を向上させる要因の一つとなり得る．また，単語ベクトルの線形演算から文書ベクトルを生成することもできる．

3.2.1 TF-IDF

TF-IDF(Term Frequency - Inverse Document Frequency) は情報検索等で頻繁に用いられるコーパス内の単語の重み付け手法である．TF-IDF は単語の出現頻度 (Term Frequency) と逆文書頻度 (Inverse Document Frequency) の二つの数値を用いることによって計算している．

$$tfidf = tf \cdot idf \tag{3.2}$$

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3.3}$$

$$idf_i = \log \frac{|D|}{|d : d \ni w_i|} \tag{3.4}$$

単語 w_i の文書 d_j における出現回数を n_{ij}，$\sum_k n_{k,j}$ は文書 d_j に出現する単語の総出現回数である．単語の出現頻度は式 (3.3) により求める．また逆文書頻度は式 (3.4) により求める．この数値が高ければ，単語 w_i がその文書のみに頻繁に出現していると考えられる．逆に数値が低ければ，単語 w_i は他の文書にも同様に出現している事を表している．つまり，ここでの idf は出現頻度の一様に高い単語の重みを下げるフィルタリングのような役割を果たしている．

上記のような重み付けを単語に施す事によって，単純な bag-of-words よりも，単語の出現の偏りを表現した文書ベクトルを生成できる．

3.2.2 トピックモデル

文書ベクトルを生成するにあたり，単語の出現を任意の確率分布に従うと仮定することでもベクトルが生成できる．このモデルは確率的言語モデルと呼ばれ，主にトピックモデルに使われる概念である．トピックモデルは単語出現頻度を確率として推定し，各文書の分類や類似度に利用する事が出来る．単語の出現確率の推定は文書内に出現する単語の出現回数を計算し，その数値を用いて推定を行う．トピックモデルでは，出現頻度の高い単語はトピックによって異なっていると仮定される (例: 政治トピックとスポーツトピックでは“選挙 ”というワードの出現確率が異なる)．上記で述べたトピックが文書においてどの単位で存在しているかを考慮する事によってモデルの構造が変化する．全ての文書が一つのトピックから構成されているモデルはユニグラムモデル (unigram model) と定義される．また，各文書が一つのトピックから構成されており，そのトピックから文書の単語が生成されるモデルは混合ユニグラムモデル (Unigram Mixture) と定義される．

今回利用するトピックモデルの一つとして，LDA[9] (Latent Dirichlet Allocation: 潜在ディリクレ配分法) がある．LDA とは，文書の生成過程を確率的にモデル化したトピックモデルの一種であり，基本的な概念として

- 一つの文書が複数の潜在トピックによって表現される．
- 各トピックは単語分布によって特徴づけられる．

が挙げられる．LDA は各文書は任意の複数のトピックから構成されており，各トピックの単語分布の合計から単語が生成されている仮定を置いている．つまり，各文書を複数のトピックで表現し，文書の大まかな特徴 (トピック) を観測することでカテゴリ分類や文書間の差異を数値によって表現することが可能になる．

具体的な文書生成モデルを以下に述べる．ここでは文書集合 M が与えられた時，文書 m 内の n 番目の単語を $w_{m,n}$ とし，$w_{m,n}$ に対する潜在変数を $z_{m,n}$ とする．LDA における文書生成過程は次のように表される．

step 1

各トピック $k = 1, \ldots, K$ に対して，ディリクレ分布に従って単語分布 ϕ を生成する．

$$\phi_k \sim Dir(\boldsymbol{\beta})$$

step 2

各文書 $m = 1, \ldots, |M|$ に対して，

step 2-1

ディリクレ分布に従ってトピック分布 θ を生成する．

$$\theta_m \sim Dir(\boldsymbol{\alpha})$$

step 2-2

文書 m 内の単語 $n = 1, \ldots, N_m$ に対して

step 2-2-1

多項分布に従って各単語のトピックを決定

$$z_{mn} \sim Multi(\theta_m)$$

step 2-2-2

多項分布に従って単語を生成

$$w_{mn} \sim Multi(\phi_{z_{mn}})$$

また，LDA におけるグラフィカルモデルを図 3.1 に示す．グラフィカルモデルとは，統計や確率論，機械学習等で用いられる確率変数間の依存構造を表したグラフ表現である．$\boldsymbol{\alpha}$ と

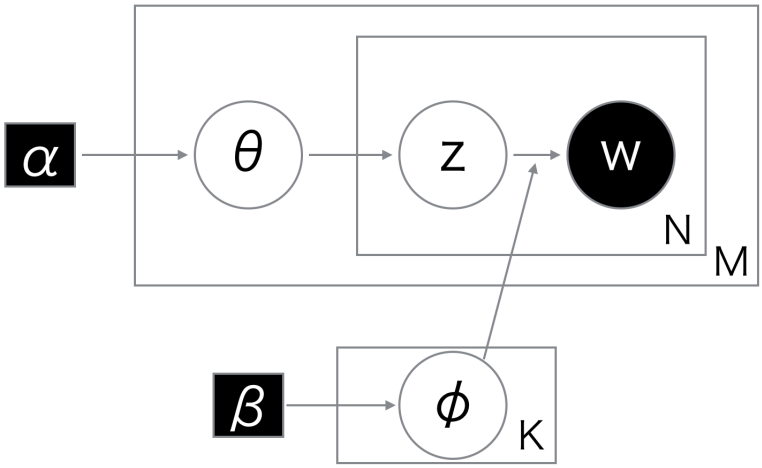

図 3.1: Graphical model representation of LDA

$\boldsymbol{\beta}$ は，モデルに対するハイパーパラメータであり，この数値によって単語やトピックの出現頻度に偏りを表現する．K, M, N はそれぞれトピック数，文書数，単語数となっている．グラフィカルモデルにおいて，矢印は入力に従い事象が決定する操作，長方形は指定された操作を右下の回数分試行することを表現している．

グラフィカルモデルに沿い，順を追って説明する．まず，**step 1** の $\phi_k \sim Dir(\boldsymbol{\beta})$ の操作は，$\boldsymbol{\beta}$ に従って V 次元単語分布 ϕ を K 個生成する操作である．ディリクレ分布とは，多

項分布に生成確率を与える分布である．パラメータを $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$，実数ベクトルを $\boldsymbol{x} = (x_1, \ldots, x_K)$ とし，これを確率変数と定義すると，$K-1$ 次元ディリクレ分布の確率密度関数 $P(x; \boldsymbol{\alpha})$ は下記のように定義される．

$$P(x; \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\boldsymbol{\alpha}_i - 1} \tag{3.5}$$

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\boldsymbol{\alpha_i})}{\Gamma(\prod_{i=1}^{K} \boldsymbol{\alpha}_i)} \tag{3.6}$$

ただし，$B(\boldsymbol{\alpha})$ はベータ関数であり，$\boldsymbol{x}$ は K 次元単体上の点であり，$\sum x_i = 1$，$x_i \geq 0$ である．ディリクレ分布を仮定し $\boldsymbol{\alpha}$ を設定する事により，多項分布を得ることができる．$\boldsymbol{\phi}$ は各トピックの単語分布を表した V 次元多項分布となっており，分布の各要素がそのトピックにおける単語の出現頻度に直結している．次に，**step 2-1** の $\theta_m \sim Dir(\boldsymbol{\alpha})$ は $\boldsymbol{\alpha}$ に従って K 次元トピック分布 $\boldsymbol{\theta}$ を M 個生成する操作である．この $\boldsymbol{\theta}$ が文書内の単語毎にトピックを決定する K 次元分布である．この分布は文書の数だけ生成されるが，その分布は文書によって異なる．$\boldsymbol{\theta}$ は文書のトピックの構成比を決定付けることからトピック混合比ともいわれる．**step 2-2-1** では，作成した K 次元分布を用いて，各文書の単語毎にトピックを決定する．グラフィカルモデルでは $\boldsymbol{\theta} \rightarrow \boldsymbol{z}$ がその操作を表現している．最後に **step 2-2-2** で，各単語のトピックが決定した後，V 次元分布を使用してトピックに対する単語を生成する．LDA の操作は文書を単語次元からトピック次元に縮退している．

LDA の解法として，変分ベイズ法，Collapsed 変分ベイズ法，Gibbs サンプリング，固有値計算，期待値伝搬法等 [24] が提案されている．本報告では変分ベイズ法 [9] を採用した．この手法は確率モデルのベイズ推定を行うための近似解法であり，LDA に用いられる一般的な手法である．変分ベイズ法では観測された W について，パラメータ $\boldsymbol{\theta}, \boldsymbol{\phi}$ と潜在変数 $\boldsymbol{z}$ を確率変数として，確率分布を求める．実際には解析的に解く事が困難な事後分布 $p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi} | \boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ を求めるのではなく，事後分布に近似する変分事後分布を学習する．変分事後分布のパラメータ推定は以下の更新式で計算される．

$$\tilde{\alpha}_{m,k} = \alpha_{m,k} + \sum_{v=1}^{N} \frac{D_{m,v} \tilde{\theta}_{m,k} \tilde{\phi}_{k,v}}{\tilde{\psi}_{m,v}}, \qquad \tilde{\theta}_{m,k} = exp(E_k[\tilde{\boldsymbol{\alpha}}_m]) \tag{3.7}$$

$$\tilde{\beta}_{k,v} = \beta_{k,v} + \sum_{m=1}^{N_m} \frac{D_{m,v} \tilde{\theta}_{m,k} \tilde{\phi}_{k,v}}{\tilde{\psi}_{m,v}}, \qquad \tilde{\phi}_{k,v} = exp(E_v[\tilde{\boldsymbol{\beta}}_k]) \tag{3.8}$$

$$E_k[\alpha] = \Psi(\alpha_k) + \Psi(\sum_{k'=1}^{K} \alpha_{k'}) \tag{3.9}$$

$D_{m,v}$ は文書 m 内の単語 v の出現回数, $\Psi(x)$ は diagamma 関数である．また，文書 m 内で単語 v が得られる確率を $\psi_{m,v} = \sum_{k=1}^{K} \theta_{m,k}\phi_{k,v}$ とする．従って，$\boldsymbol{\phi}$ と $\boldsymbol{\theta}$ を与えられた時，各単語は他の単語と独立で，単語の語順等の構造的な情報は考慮に入れない。

一例として，ツイートデータ 8000 件を入力データとした時のトピック例を図 3.2 に挙げておく．図 3.2 は LDA で求めたトピックを WordCloud [7] で出力した画像を出している．図 3.2 のトピックでは “打席 ”，“満塁 ”，“バレンティン ”，“試合 ”，“回 ” など，野球に関連したトピックワードがまとめられている．このようなトピックを教師なし学習で自動で抽出することが可能な点がトピックモデルの利点である．

図 3.2: 抽出されたトピックの一例

LDA は予めトピック数をユーザ側で指定し，事後分布を学習していくが，データセットに応じてパラメータの値を自動で調整する手法も存在する．その一つに HDP [21]（階層的ディリクレ過程: Hierarchical Dirichlet Process）が存在する．HDP は LDA と異なり，トピック生成にディリクレ分布ではなく，ディリクレ過程を用いている．ディリクレ過程はディリクレ分布の一般化であり，次数が可変のディリクレ分布である．HDP はこのディリクレ過程のパラメータにディリクレ過程を与えている (ディリクレ過程の階層化)．このようなモデルを形成することで，LDA ではユーザ指定であったトピック数を自動で決定する．基本的に文書数が多い方がトピック数も多くなる傾向にある (“スポーツ ” であれば “サッカー ”，“野球 ” などより細かいトピックに分類される) .

LDA と比べて，モデルの本質な違いはほとんどないので，最適なパラメータを選択した LDA と精度は同等であると言われている．解法として，棒折り過程や中華料理店過程が存在する [21].

[7] http://amueller.github.io/word_cloud/index.html

3.2.3 Weighted Text Matrix Factorization

3.2.2 節で述べたトピックモデルは文書を高次元の単語空間から低次元のトピック空間に縮退させることができるが，今回取り扱う Twitter のような短文で構成されたコーパスでは精度が落ちることが報告されている [25]．これは，トピックモデルが与えられた文書内の観測された単語の共起頻度に注目するが，コーパスの疎性 (sparse) により，トピック空間において文書をベクトル化することが不完全であることがその原因として挙げられている．

また，今回のようなスパースな文書ベクトル同士の類似度計算を行う時，文書ベクトルはどちらもスパース (sparse vector) である．従って，ほとんどのベクトル要素が 0 で単語共起が不十分であるため，正確な類似度計算が困難になってしまう [12][13].

このような疎な文書集合における適切なモデル化を目的としたモデルが WTMF モデル [12] (Weighted Text Matrix Factorization) である．WTMF モデルは基本的な考え方として，“注目する文に対してその非観測語は関連すべきではない” という考えに基づいている．ここでの各文における非観測語の定義は，全単語集合から観測語を引いた単語集合を指す．従来の手法では主に単語の共起頻度のみに着目してきたが，非観測語も利用しモデル化する事で文書ベクトルの情報量を増加させる意図がある．

まず，文書集合を (3.1) のような単語文書行列 (共起行列) で表現する．各行が単語に，各列が文に対応している．また，行列内の各セルは TF-IDF 値を格納している．行列を (3.10) に示す．

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \tag{3.10}$$

WTMF モデルの大まかなアプローチは Singular Value Decomposition [14](SVD) と類似している．図 3.3 のように，WTMF は単語文書行列 X を $K \times M$ 行列である P，$K \times N$ 行列である Q の二つの行列の内積で近似させる事を目的としたモデルである．このモデルにおいて，ある文 s_j は K 次元潜在ベクトル $Q_{\cdot,j}$ で表現される．同様に，単語 w_i はベクトル $P_{\cdot,i}$ に対応している．

$$\sum_{i,j} W_{i,j}(P_{\cdot,i} \cdot Q_{\cdot,j} - X_{i,j}) + \lambda||P||_2^2 + \lambda||Q||_2^2 \tag{3.11}$$

$$W_{i,j} = \begin{cases} 1 & (\text{if } \ X_{i,j} \neq 0) \\ w_m & (\text{if } \ X_{i,j} = 0) \end{cases} \tag{3.12}$$

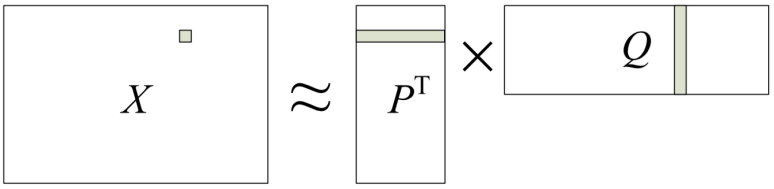

図 3.3: Weighted Textual Matrix Factorization

WTMF モデルのパラメータ行列 P，Q は式 (3.11) の目的関数を最小化することによって最適化される．λ は正則化項であり，パラメータが過学習する事を防ぐ目的がある．$W_{i,j}$ は非観測語の重みであり，式 (3.12) で定義される．単語文書行列 X のほとんどのセルが 0(非観測語) である為，観測語の影響が減少される．従って，観測語の影響を保存する為に，小さな重み w_m を非観測語 (X のセル = 0) に割り当てる．

また，K 次元潜在ベクトル $Q_{\cdot,j}$，$P_{\cdot,i}$ の演算には直感的な意味が存在している．

(1) 単語ベクトル $P_{\cdot,i}$ と文ベクトル $Q_{\cdot,j}$ の内積が X_{ij} に近似される (図 3.3 の射影部分).

$$X_{ij} \approx P_{\cdot,i} \cdot Q_{\cdot,j}$$

(2) 式 (3.11) は，文が非観測語と関連付けないように立式されている．

$$X_{ij} = 0 \rightarrow P_{\cdot,i} \cdot Q_{\cdot,j} = 0$$

(3) 二つの文 s_j，$s_{j'}$ 間の類似度は，$Q_{\cdot,j}$，$Q_{\cdot,j'}$ 間のコサイン類似度を計算することで求めることが可能である．

$$sim(s_j, s_{j'}) = \frac{Q_{\cdot,j} \cdot Q_{\cdot,j'}}{|Q_{\cdot,j}||Q_{\cdot,j'}|}$$

このように行列 P, Q を定義し，Q から文書同士のコサイン類似度を計算することでスパースで，かつ非観測語を考慮した文書の類似度計算が可能となる．実際の P, Q の推論に関しては，ALS(交互最小二乗法: Alternating Least Square)[26] によって決定される．

$$P_{\cdot,i} = (Q\tilde{W}^{(j)}Q^T + \lambda I)^{-1} Q\tilde{W}^{(j)}X_{i,\cdot}^T \quad (3.13)$$

$$Q_{\cdot,j} = (P\tilde{W}^{(i)}P^T + \lambda I)^{-1} P\tilde{W}^{(i)}X_{\cdot,j}^T \quad (3.14)$$

$$\tilde{W}^{(i)} = diag(W_{i,\cdot}) \quad (3.15)$$

$$\tilde{W}^{(j)} = diag(W_{\cdot,j}) \quad (3.16)$$

ただし，$\tilde{W}^{(i)}$ は weight matrix W の i 番目の行を対角成分に配置した対角行列である．また，$\tilde{W}^{(j)}$ は weight matrix W の j 番目の列を対角成分に配置した対角行列である．

3.2.4 単語・文書の分散意味表現

近年, 機械学習の中で研究が盛んになっている分野の一つとして, 深層学習 (Deep Learing) が挙げられる．自然言語処理分野では，その技術の先駆けとして word2vec[16][17][18][19] が取り挙げられる事が多い．word2vec はテキストデータから単語の生起を学習するニューラルネットワークを用いた言語モデルの一つである．学習された出力データは各単語の低次元密ベクトルであり，各ベクトルが単語概念を表現しており距離計算や，今まで困難とされてきた単語間の演算も可能となった．

word2vec の学習モデルとして，CBOW (Continuous Bag-Of-Words) モデルと Skip-gram モデルの二つが提案されている． 図 3.4 において，左のモデルが CBOW であり，周辺の文脈

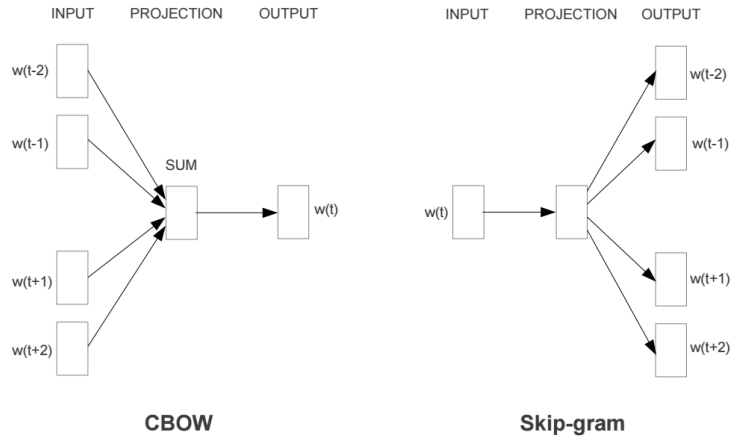

図 3.4: CBOW モデルと Skip-gram モデル [8]

(周辺単語) から単語を予測するモデルである．逆に，右の Skip-gram はある単語から周辺の単語を予測するモデルである．どちらのモデルでも，入力層と中間層間の重み行列 (各単語に対する重みベクトルの集合) が，word2vec が最終的に生成する単語ベクトルの集合 (分散表現) となる．この単語ベクトルは，類似している単語がベクトル空間において近い位置にマッピングされるように表現されている．興味深い事にこのベクトルは，線形演算にも意味を含蓄しており，例えば “$king$” − “man” + “$woman$” = “$queen$” となるような結果も提出されている [16][19].

具体的な計算方法を Skip-gram を例に述べる．Skip-gram のトレーニング目的は, 周辺語の予測に利用出来る単語表現を見つけることである．つまり，訓練用の単語列 $w_1, w_2, \ldots, w_T$ が与えられた時, Skip-gram の最終目標は式 (3.17) の対数確率を最大化する事に帰着される．

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{3.17}$$

[8] 出典: Efficient Estimation of Word Representations in Vector Space[17]

c は文脈窓のサイズ (観測する周辺の距離) であり，トレーニング精度と計算量のトレードオフの関係となっている．データによるが，一般的に 5〜10 程度で定義される [17][18]．式 (3.17) で定式化されている $p(w_{t+j}|w_t)$ は softmax 関数を用いて以下のように表される．

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^{\mathrm{T}} v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^{\mathrm{T}} v_{w_I})} \tag{3.18}$$

v_w と v'_w はそれぞれ，w の入力ベクトルと出力ベクトルである．W は辞書内の単語総数である．式 (3.18) は w_I から w_O が出現する確率を，全単語中から対象単語を選択するような softmax 関数で定義している．この関数の分母に注目すると全単語分の和を計算するので，この勾配計算の計算コストが跳ね上がってしまう．この計算オーダーを減らす方法として，階層的 softmax と負例サンプリングが存在する．

階層的 softmax は単語の出現の代わりに意味を使用したモデルであり，具体的には単語をクラスタリングし，バイナリの 2 分木を生成し，各単語の出現確率を根 (root) からその単語までの各ノードのベクトルと内積を取り，その値を入力値としたシグモイド関数の積で単語の生起確率を近似する方法である．具体的な計算式を以下に記す．

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot {v'_{n(w,j)}}^{\mathrm{T}} v_{w_I}) \tag{3.19}$$

$n(w,j+1)$ は，根から w までの道における j 番目のノードであり，$L(w)$ はこの道の距離である．従って，$n(w,1) = root$，$n(w,L(w)) = w$ である．任意の内部ノード n に対して，任意の修正された n の子ノードを $ch(n)$ としている．$[x]$ は x が真であれば 1，偽であれば -1 と定義する．また，$\sigma(x) = \frac{1}{(1+\exp(-x))}$ である．$\sum_{w=1}^{W} p(w|w_I) = 1$ であることから，$p(w_O|w_I)$ の計算量は $L(w_O)$ に比例し，計算量の平均は $\log W$ を超えない [18]．実際の木の生成方法としてハフマン木が挙げられる．単語ごとにハフマン符号を割り当てる際に，頻出単語に短い符号を割り当てる事で高速アクセスを可能にしている．

負例サンプリングは，式 (3.18) の分母の全単語に関して和をとる代わりに，単語分布に関する期待値計算に近似させる手法である．期待値計算に関しては，複数個のサンプルを取って計算する．実際には，5〜20 個程度のサンプルで良いとの結果が出ており [18]，全単語数のループがわずか数回のループで終わるというメリットがある．計算式を以下に挙げる．

$$\log P(w_O|w_I) = \log \sigma({v'_{w_O}}^{\mathrm{T}} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \{\log \sigma(-{v'_{w_i}}^{\mathrm{T}} v_{w_I})\} \tag{3.20}$$

σ 内の期待値計算は，式の通り k 個 (5〜20) の負例サンプルを抽出して近似させる．この計

算はロジスティック回帰分布を用いてノイズ分布 $P_n(w)$ から抽出された w_O を区別する意味合いを含んでいる．$P_n(w)$ は 1-gram 頻度分布の $\frac{3}{4}$ 乗に比例させた時がもっとも性能が高いことが報告されている [18].

モデルは確率的勾配降下法 (SGD : Stochastic Gradient Descent) を用いて最適化され，ベクトルと重みベクトルの更新を単語毎に更新する．

word2vec は単語の低次元ベクトルを求めていたが，word2vec を文書のベクトル表現に適応させたモデルも提案されており，その一つとして doc2vec (Paragraph Vector)[27] が挙げられる．doc2vec も word2vec と同様に文書ベクトルを低次元密ベクトルで表現する方法であり，生成モデルとして，DM (Distributed Memory Model) と DBoW (Distributed Bag-of-Words Model) の二つの手法が提案されている．各モデルの違いを図 3.5，3.6 に示す．DM はモデルのコンセプトとしては CBOW とほぼ同様のモデルである．唯一異なる点として，CBOW は単語の予測に周辺の単語を入力として予測していたが，DM は周辺単語だけでなく，文書のベクトルも入力ベクトルとして使用する点が挙げられる．DBoW は Skip-gram と同様に文書情報から文脈中の単語の出現を予測出来るように学習させるモデルである．文書ベクトルを抽出して単語を推定，出力された単語と文書に実際含まれる単語との誤差をバックプロパゲーションすることで文書行列が最適化されていき，文内の単語生起を予測することができる．

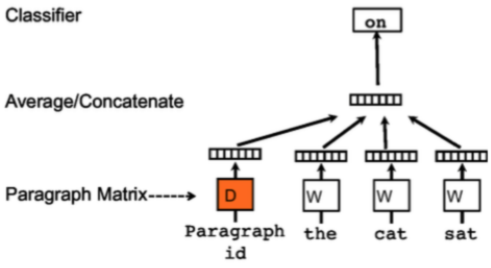

図 3.5: Distributed Memory Model[12]

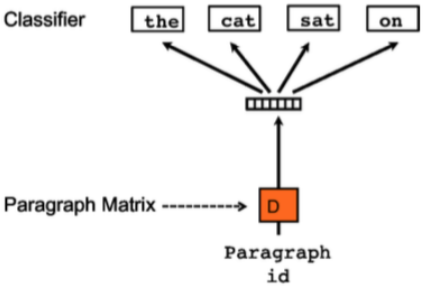

図 3.6: Distributed Bag-of-Words Model[12]

3.3 文書間類似度の計算

ベクトル空間モデルで表現した文書の類似度を計算する方法の中で，もっとも一般的なものとしてコサイン類似度があげられる．

[12]出典：Distributed Representations of Sentences and Documents[27]

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} \tag{3.21}$$

コサイン類似度は式 (3.21) のように定義され，$\vec{x}$，$\vec{y}$ が対象の文書ベクトルである．この数値が 1 に近ければ近い程，$\vec{x}$ と $\vec{y}$ は似ている文書である．逆に 0 に近づけば $\vec{x}$ と $\vec{y}$ は似ていないものとみなされる．また文書ベクトルを正規化し，これを離散確率分布とみなした時，KL ダイバージェンス (Kullback-Leibler divergence) を用いて分布の差異を計算することができる．離散確率分布 P と Q の，P における Q の KL ダイバージェンスは以下のように表せる．

$$D_{KL}(P||Q) = \sum_i P(i) \log_2 \frac{P(i)}{Q(i)} \tag{3.22}$$

KL ダイバージェンスは常に正の値を取り，値が 0 に近ければ近い程，二つの確率分布は類似しているとみなす．KL ダイバージェンスは対称性がない ($D_{KL}(P||Q) \neq D_{KL}(Q||P)$) ので，正確には距離の尺度ではない (距離の公理を満たさない)．そこで，KL ダイバージェンスに対称性を持たせたものが JS ダイバージェンス (Jensen-Shannon divergence) である．JS ダイバージェンスは下記のように定義される．

$$D_{JS}(P, Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \qquad (M = \frac{1}{2}P + \frac{1}{2}Q) \tag{3.23}$$

また，Yangqiu ら [28] は，word2vec をコサイン類似度 (式 (3.21)) に組み込んだ短文のための類似度計算手法を提案している．従来の手法では，精度向上が困難であった短文間の類似度計算を，word2vec による単語間の類似度を加える事で，ベクトル演算における疎性を解消し，短文により情報量を増やす事ができる．本報告では，具体的な計算方法として式 (3.24)，式 (3.25) を用いる．

$$S_A(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n_x||\boldsymbol{x}|| \cdot n_y||\boldsymbol{y}||} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} x_{a_i} y_{b_j} \phi(a_i, b_j) \tag{3.24}$$

$$S_M(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \sum_{i=1}^{n_x} x_{a_i} y_{b_j} \max_j \phi(a_i, b_j) \tag{3.25}$$

まず，式 (3.24) の式は，word2vec の全ての各単語間の類似度の平均値をとったものであり，最も直感的な融合方法である．n_x，n_y は各文書ベクトルの非 0 要素の数，a_i，b_j はショートテキストに含まれる各単語であり，$\phi(a_i, b_j)$ は 2 単語間の word2vec 値，x_{a_i}，y_{b_j} はベクトルの各要素値，$||\cdot||$ は 2-ノルムである．しかしながら，式 (3.24) の手法において，$\phi(a_i, b_j)$ の値の多くは 0 に近い値をとる．このようなケースでは，未知の単語間の類似度を求めるどころか，ノイズの原因になってしまう事がある．従って，単語間の最大マッチを考えた計算法が式 (3.25) である．具体的には，$argmax_{j'}\phi(a_i, b_{j'})$ として j を選出し，a_i と b_j 間の類似度をそのまま $\boldsymbol{x}$，$\boldsymbol{y}$ の類似度として代用する計算である．

本実験では，コサイン類似度に加え，この二つの類似度計算も採用して，類似度計算を行った．尚，提案された論文において，ベクトルの各要素値 x_{a_i}，y_{b_j} は ESA[15](Explicit Semantic Analsys) で求めた要素を導入していたが，今回の実験においては，文書ベクトルの TF-IDF 値を使用した．

第4章　評価指標

本研究では入力ポストの返答 (reply) に適している Tweet をランキング形式で返すシステムを実装している．従って，評価実験で用いる評価指標はランキング全体を評価できるような手法が望ましい．そこで，P@k，TOP@k，MRR，MAP，nDCG@k の５つを評価手法として用いる．

4.1　P@k

P@k(Presicion at k) は上位 k 件の精度を表したモデルである．$[0, 1]$ 間に値が得られ，値が 1 に近い程，上位 k 件の正解率が高い．

$$P@k = \frac{1}{N} \sum \frac{c}{k} \tag{4.1}$$

P@k の式を (4.1) に示す．c は上位 k 件に入っていた正解データ (tweet) の個数，N は評価対象のランキング数である．今回 k の値は，1，5，10，20 で実験を行った．

4.2　TOP@k

TOP@k は上位 k 位以内に正解データ (本実験では tweet) が入っているかどうかを表したものである．式を (4.2) に示す．T はランキング内の正解データの有無の二値である．

$$TOPk = \frac{1}{N} \sum T \tag{4.2}$$

$$T = \begin{cases} 0 & \text{(上位 } k \text{ 位以内に正解 Tweet が存在しなかった場合)} \\ 1 & \text{(上位 } k \text{ 位以内に正解 Tweet が存在した場合)} \end{cases}$$

4.3　MRR

MRR(Mean Reciprocal Rank) はランキングにおいて最上位の正解データのみを観測して評価を行う．ランキングの中で一位から順に観測し，初めて出現した正解データに対して

順位の逆数 (RR: Reciprocal Rank) を取り，それを合計し，全てのランキング数で割ったものである．出力数値は [0, 1] であり，1 に近ければ近い程良いランキングであると言える．

$$MRR = \frac{1}{N}\sum_{i}^{N}\frac{1}{r} \tag{4.3}$$

MRR の式を (4.3) に示す．r は正解データが出現した順位である．

4.4 MAP

MAP(Mean Average Precision) はランキング結果に対して，結果内の不正解データの少なさと正解データの漏れの少なさを評価する指標である．数値は [0, 1] であり，1 に近ければ近い程良いランキング精度であると言える．

$$AP_i = \frac{1}{R_i}\sum_{j} I(j)\frac{count(j)}{j} \tag{4.4}$$

$$MAP = \frac{1}{N}\sum_{i}^{N} AP_i \tag{4.5}$$

MAP の式を (4.4)，(4.5) に示す．R_i はランキング i における実際の正解数，$I(j)$ は j 位のデータが正解かどうか (0 or 1) の二値，$count(j)$ は j 位までの正解データ数，AP_i はランキング i における Average Precision である．

4.5 nDCG@k

nDCG(Normalized Discounted Cumulative Gain) とは，ランキングの各順位に評価値 (スコア) が付随している時のランキング評価手法である．まず, DCG(Discounted Cumulative Gain) では関連度の高いデータが上位にあればある程評価値が高いという概念に基づく．DCG の定義式を式 (4.6) に示す．

$$DCG_k = rel_1 + \sum_{i=2}^{k}\frac{rel_i}{\log_2 i} \tag{4.6}$$

rel_i は 生成されたランキングに対する i 位の評価値を表している．k はランキングの上位 k 位まで観測することを表現している．DCG は評価値の高いデータが下位にランキングされた時にランキングに対して対数反比例的にペナルティが課せられる．

式 (4.6) によって DCG は定義されるが，評価値として使用する場合，ランキングに応じて DCG 値の大きさが異なるため，理想的なランキングの DCG 値 ($IDCG_k$) を 1.0 として正規

化する必要がある．正規化された DCG 値を nDCG(Normalized Discounted Cumulative Gain) と呼び，式 (4.7) で定義される．

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{4.7}$$

式 (4.7) のように正規化を施すことにより，全てのランキングの nDCG 値はランキングアルゴリズムの平均パフォーマンス値を測定することが可能となる．数値は [0, 1] であり，1 に近ければ近い程正解に近いランキングである．

第5章　実験設定

本研究ではTwitterテキストを入力ポストとした時，その入力ポストに対するコメントをランキング形式で返すシステムを実装している．本実験は，この出力ランキングをP@1，P@5，P@10，P@20，TOP20，TOP50，MRR，MAP，nDCG@20の9つの手法で評価する．

5.1　実装環境

本実験の開発環境を以下に記す．

実装環境

使用言語　：Python 3.5.0 :: Anaconda 2.4.0 (x86_64)
使用マシン：iMac (Retina 4K, 21.5-inch, Late 2015)
プロセッサ：3.3 GHz Intel Core i7
メモリ　　：16 GB 1867 MHz DDR3

今回の手法の内，TF-IDF，LDA，HDP，word2vec，doc2vecの実装はpythonのGensim[1]という科学計算ライブラリを使用した．Gensimは分かち書きされた文書集合を入力すると，出現単語をIDで振り分け，単語IDと出現頻度をまとめたコーパスと単語-ID辞書を生成することができる．さらに，作成されたコーパスと辞書からTF-IDFの重み付けや，LDA等のトピックモデル，word2vec等の学習まで試行することが可能である．

5.2　実験データ

本実験において使用したtweetデータは，TwitterのAPIを用いてクロールしてきた日本語テキストデータを用いた．報告者はNTCIR[2] (NII Testbeds and Community for Information

[1] https://radimrehurek.com/gensim/
[2] http://research.nii.ac.jp/ntcir/index-ja.html

access Research) が運営している STC (Short Text Conversation) Japanese Task[3] に参加している．従って，今回使用するデータの形式としては，NTCIR が提供している 500,000 件のツイート ID のペア (ツイートとそれに対応した返信ツイート (reply)) を Twitter API を用いてクロールし，データレポジトリ (コーパス) として使用する．しかし，今回公開されているツイート ID は 2014 年に投稿されたツイートであるので，対象ツイートが削除されていたり，投稿したアカウントが非公開アカウントになっていると，データをクロールすることが不可能である．よって今回の実験では，ツイートペアの内，投稿されたツイートを Post Tweet，返信ツイートを Comment Tweet，片方しか取得できなかったツイートを Residue Tweet と呼ぶ事とする．クロールの結果，ツイートペア 428,124 ペア，ペアの片方が存在しないツイート 64, 395 件，合計 920,643 件のツイートを取得した．

また実験で使用する評価用ポストツイートについても同様に，NTCIR が公開している評価用のツイート ID セットを使用する．評価用のデータセットには，入力ポストとして使用するツイートの ID と，それに対応したツイートをコーパスから 10 件取り出し，そのツイートが入力ポストの返答として適切かどうかを 10 人のアノテータが 0〜2 (もしくは NA) の三段階評価したものが付随している．評価の判断基準としては

0 ：返答として意味を成さない

1 ：返答として意味が通る

2 ：良い返答である

NA ： 入力ポスト自体が意味を為していないと判断した場合

となっている．今回はこの評価値を平均して (“NA”は除外した) 評価値が 0.7 以上のデータを正解データとして使用した．また，nDCG に使用する評価値もこの値を使用し，アノテートされていないツイートに関してはすべて評価値を “0”とした．データクロールの結果，179 件の評価用の入力ポスト (ツイート) を取得した．最後に，上に述べたデータ構造のイメージ図を図 5.1 に示す．また，付録 A にデータの一例を挙げる．

5.3 構築した自動応答システム

今回実装した自動応答システムの概要を図 5.2，図 5.3 に示す．大まかな流れとして，

(i) 入力ポストが与えられる．

(ii) 入力ポストと予めクロールした各ツイートで類似度計算を施す．

[3] http://ntcir12.noahlab.com.hk/japanese/stc-jpn.htm

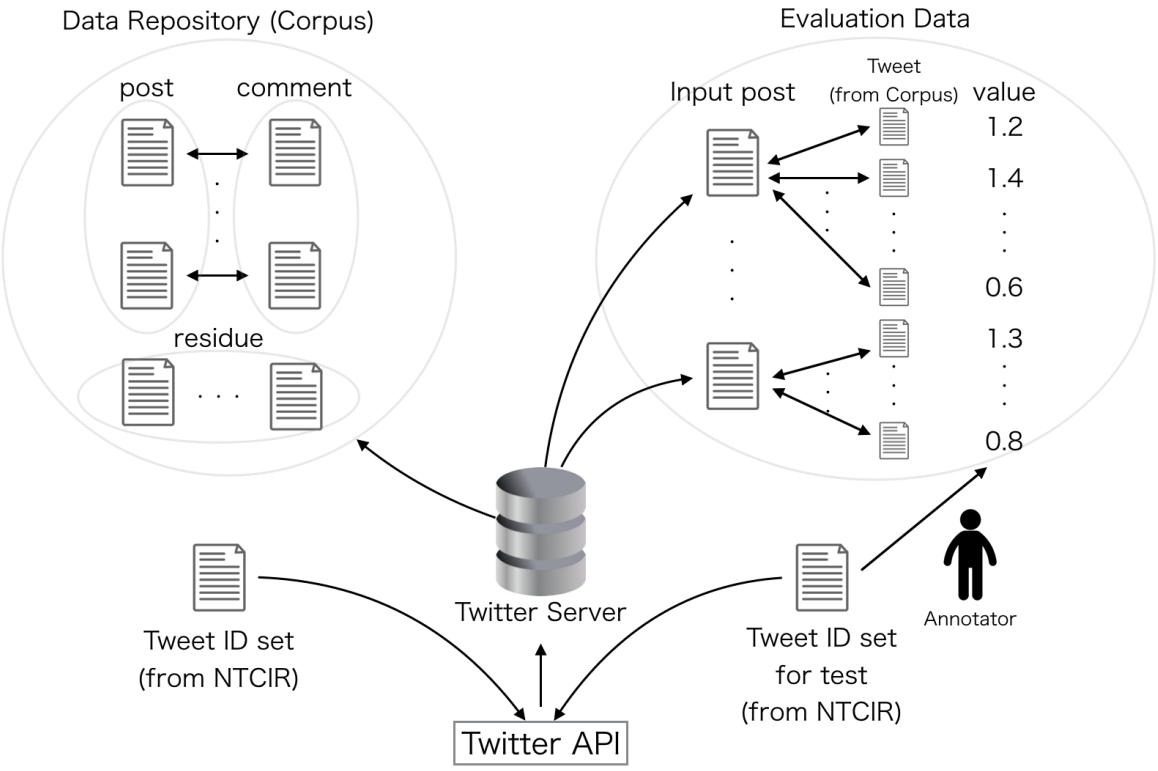

図 5.1: データ構築イメージ

(iii) 類似度の高いものから順にランキングを行う．
(今回は上位 50 件のみ観測する事とした)

(iv) 生成されたランキングを出力として返す．

の 4 行程から成っている．以下に行程の詳しい実装を述べる．

5.3.1 文書間類似度計算手法

(ii) の類似度計算において，本実験では，文書ベクトル生成と類似度計算の組み合わせを 9 通り実装し，それぞれから生成されたランキングを評価する．手法を以下に挙げる．

① TF-IDF

各ツイートに対して TF-IDF の重みを付与し，コサイン類似度 (式 (3.21)) で類似度計算する．

② LDA → TF-IDF

入力ポストをコーパスから生成した LDA モデルに変換した際に得られるトピック分布から，各トピックの生起確率の高い単語上位 30 個を取り出

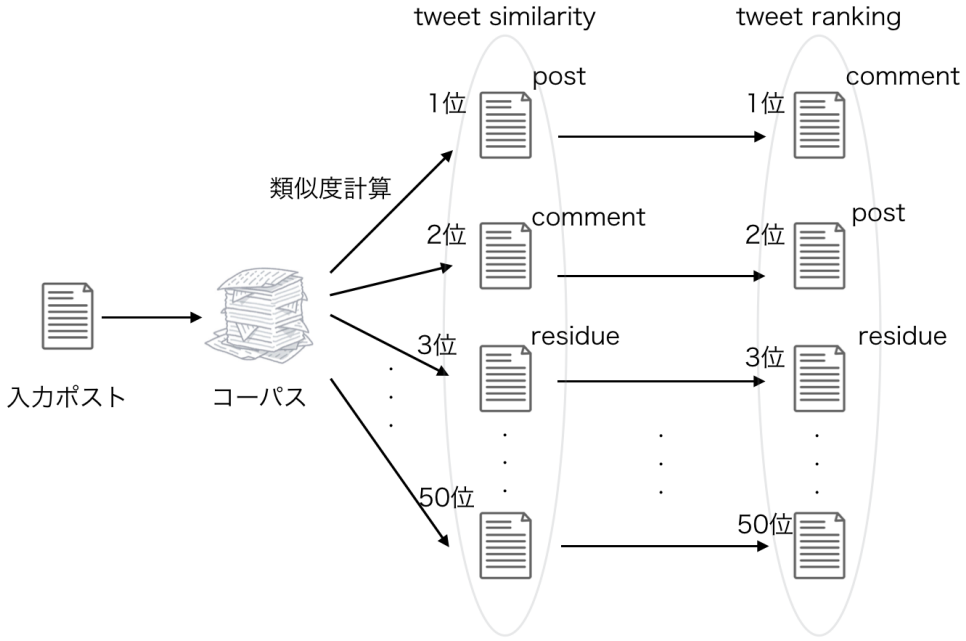

図 5.2: Pair Raking Method

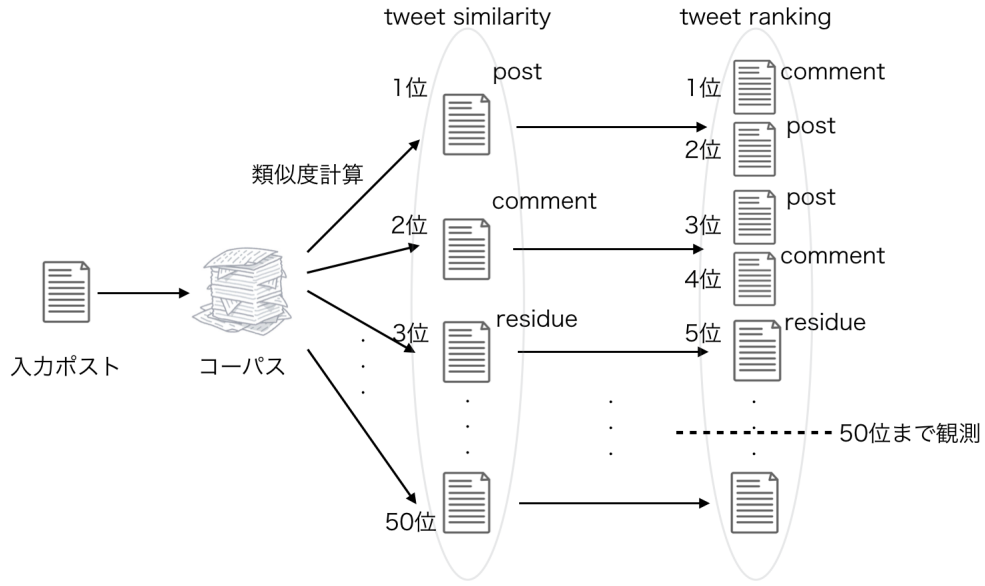

図 5.3: All Raking Method

し，その単語が含まれているツイートを検索する．その後 TF-IDF と同様，コサイン類似度 (式 (3.21)) で類似度を計算する．LDA の学習モデルの生成トピック数は 200，300，400 で実験を試行する．

③ HDP → TF-IDF

HDP を用いて，LDA と同様の手順で行う．

④ WTMF model

行列分解を仮定した際の，行列 Q(式 (3.11)) を計算し，コサイン類似度 (式 (3.21)) で類似度計算する．

⑤ word2vec

Wikipedia のダンプデータ [4] を使用し，word2vec を用いて単語ベクトルを生成する．その後入力ポストと各ツイートに対して，ツイート内の単語を word2vec でベクトル表現させ，それの線形和でツイートを表す．類似度計算は同様にコサイン類似度 (式 (3.21)) を使用する．word2vec のモデルは単語ベクトルの次元は 300，ウィンドウサイズを 3，4，5 で設定して実験を行う．word2vec の計算方法は CBOW モデル，負例サンプリングを採用した．

⑥ doc2vec

文書ベクトル 500，ウィンドウサイズ 3，4，5，6 で実験を試行し，生成された文書ベクトルに対し，コサイン類似度 (式 (3.21)) で類似度計算する．尚，doc2vec の実際の計算方法として，Distributed Memory model (PV-DM) と負例サンプリングを採用した．

⑦ word2vec → TF-IDF

⑤の word2vec モデルを使用して，入力ポストの各単語に最も近い単語をそれぞれ 20 個取り出し，その単語を含んでいるツイートを検索し，TF-IDF で重み付け，コサイン類似度で類似度計算する．

⑧ word2vec(average) + TF-IDF

各ツイートの類似度計算に式 (3.24) を用いる．

⑨ word2vec(max) + TF-IDF

各ツイートの類似度計算に式 (3.25) を用いランキングを生成する．

[4] http://dumps.wikimedia.org/jawiki/latest/jawiki-latest-article.xml

5.3.2 ランキング手法

自動応答システムにおいて，行程 (iii) のランキング方法に関して，今回の実験では二通りの方法で実装した．一つ目は，(ii) で類似度計算を行った後，類似度上位から順にランキングしていくが，類似度上位 50 位にランクインしたツイートに付随するツイートを出力ランキングとして使用する (図 5.2)．二つ目は，ランクインしたツイートに，1：付随するツイート，2：元のツイート，の順にランキングさせたものを出力ランキングとして使用する (図 5.3)．本報告では前者を Pair Ranking Method，後者を All Ranking Method と呼ぶ．

例えば，類似度計算で，クロールしたツイートの内 Post Tweet に該当するツイートがランクされた時，当該ツイートと対を成しているツイート，即ち Comment Tweet を出力ランキングとして使用するのが Pair Ranking Method である．それに対して，対になっている Comment Tweet だけでなく当該ツイート自身 (Post Tweet) もランキングに含ませる方法が All Ranking Method である．ランクしたツイートが Comment Tweet の際は Post と Comment が逆になる．ただし，ペアになっていないツイート (Residue Tweet) がランクされた時はそのままランキングに使用する．以上のような手法を用いてランキングを出力し，出力されたランキングを評価する．

この二つのランキング手法を試行した理由として，類似度計算を行った際，付随するペアを抽出してランキングを生成するか，類似度の高いツイートそれ自身もランキングに含んだ方が精度が向上するかを調べる必要性があると考えたためである．本実験はランキングの評価なので，もし Pair Ranking Method が All Ranking Method に比べて検索精度が上であったら，図 1.3 のような流れが上手く構築されていることが言える．また，All Ranking Method が Pair Ranking Method に比べて精度が高い場合，類似度の高いツイートそれ自身も返答対象になりうる事が言える．後者は，Twitter のデータが質問–応答のペアだけでなく，日常系の会話文も含まれていることから起こりうる事例であると思われる．つまり，ポストとコメントのペアが反対でも成り立つ場合である (例: “夏の北海道の気候って最高だよな ” と，“今日の北海道めっちゃ涼しい ”)．この場合どちらのツイートに対しても正解アノテートがされている場合が高い．以上の理由から二つのランキング手法を採用した．

第6章　実験結果

実験で使用した評価用データは，テストデータが存在しておらず，デベロップデータのみであったため，各手法のパラメータチューニングは同一の評価用データセットで行った．従って，各手法のパラメータチューニングは参考として付録 B に掲載するものとした．本章に記載するグラフは各手法の中で NDCG@20 の数値が最も高かったパラメータをグラフデータとして使用した．結果として，Pair Ranking Method，All Ranking Method のどちらも② LDA はトピック数 400，④ WTMF は $\lambda = 0.10$，⑥ doc2vec と⑧ word2vec(average) + TF-IDF，⑨ word2vec(max) + TF-IDF はウィンドウサイズ 4，⑦ word2vec → TF-IDF はウィンドウサイズ 5 を採用した．(また，⑤ word2vec と，⑥ doc2vec の Pair Ranking Method に関しては，ウィンドウサイズによらずどの数値も 0 であったため，考慮する必要がなかった．)

6.1　All Ranking Method

All Ranking Method における，各評価値を図 6.1，表 6.1 に示す．

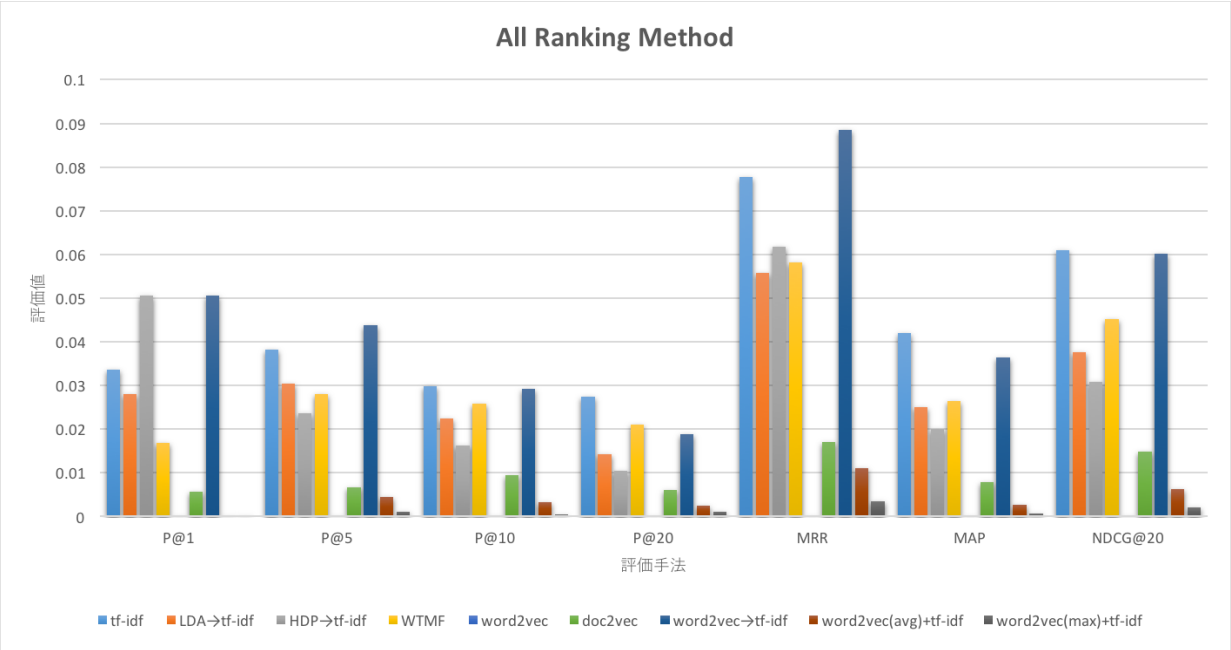

図 6.1: All Ranking Method の結果

表 6.1: All Ranking Method の各評価値

	tf-idf	LDA → tf-idf	HDP → tf-idf	WTMF	word2vec	doc2vec	word2vec → tf-idf	word2vec (avg) + tf-idf	word2vec (max) + tf-idf
P@1	0.0337	0.0280	0.0505	0.0168	0.0	0.0056	0.0505	0.0	0.0
P@5	0.0382	0.0303	0.0235	0.0280	0.0	0.0067	0.0438	0.0044	0.0011
P@10	0.0297	0.0224	0.0162	0.0258	0.0	0.0095	0.0292	0.0033	0.0005
P@20	0.0275	0.0143	0.0103	0.0210	0.0	0.0061	0.0188	0.0025	0.0011
TOP@20	0.2528	0.1516	0.1179	0.1910	0.0	0.0617	0.2022	0.0449	0.0224
TOP@50	0.3651	0.2078	0.1516	0.2977	0.0	0.0842	0.2808	0.0674	0.0393
MRR	0.0776	0.0558	0.0618	0.0582	0.0	0.0171	0.0884	0.0109	0.0034
MAP	0.0420	0.0250	0.0201	0.0264	0.0	0.0079	0.0363	0.0026	0.0007
NDCG@20	0.0609	0.0376	0.0307	0.0451	0.0	0.0149	0.0600	0.0062	0.0021

図 6.1 の結果から，相対的に性能の高いものとして，① TF-IDF，⑦ word2vec → TF-IDF が挙げられる．④ WTMF モデルに関しては全体的に TF-IDF のデータよりも少し低い評価値になっている．また，② LDA → TF-IDF，③ HDP → TF-IDF に関しては④ WTMF と同様かそれ以下の結果となった．⑤ word2vec，⑥ doc2vec，⑧ word2vec(average) + TF-IDF と，⑨ word2vec(max) + TF-IDF は他の結果に比べて精度が低い結果となった．

All Ranking Method でのランキング方法を使用した時のランキングの中に正解データが含まれているかどうかを観測した結果 (TOP 値) を図 6.2 に示す．

図 6.2: All Ranking Method におけるランキング内正解含有率調査

図 6.2 より，最も正解データが含まれている確率が高い手法は① TF-IDF であった．④ WTMF や⑦ word2vec → TF-IDF も数値は他と比べて高い数値であったが，どちらも① TF-IDF の比べ，5%程数値が低かった．

6.2 Pair Ranking Method

Pair Ranking Method における，各評価値を図 6.3，表 6.2 に示す．また，TOP@k の結果を図 6.4 に示す．

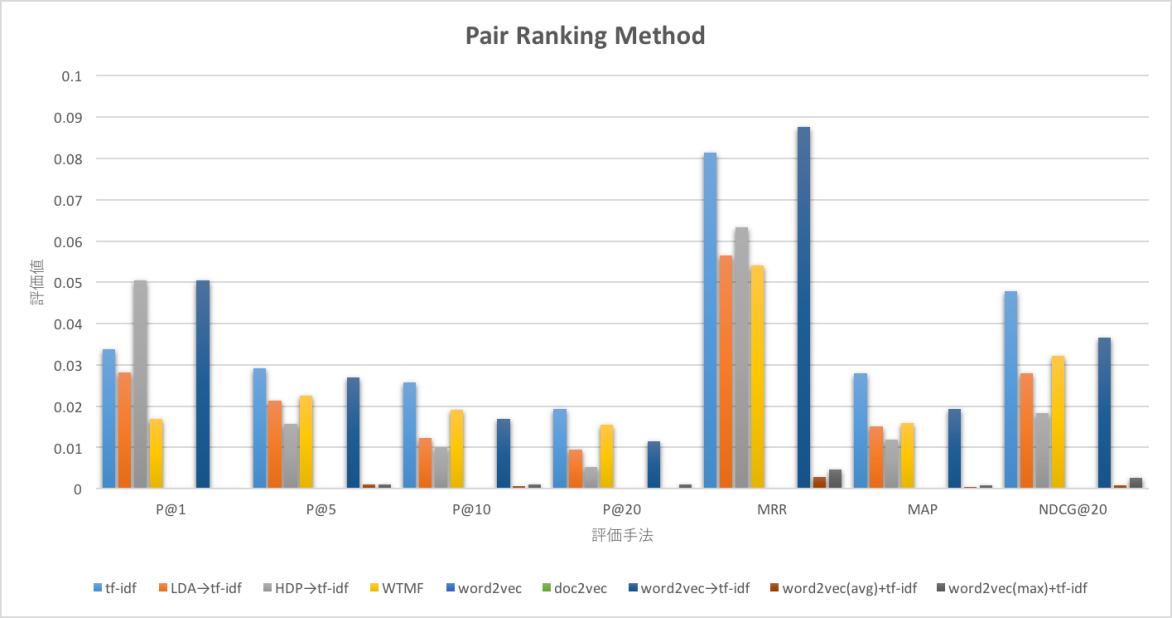

図 6.3: Pair Ranking Method の結果

表 6.2: Pair Ranking Method の各評価値

	tf-idf	LDA → tf-idf	HDP → tf-idf	WTMF	word2vec	doc2vec	word2vec → tf-idf	word2vec (avg) + tf-idf	word2vec (max) + tf-idf
P@1	0.0337	0.0280	0.0505	0.0168	0.0	0.0	0.0505	0.0	0.0
P@5	0.0292	0.0213	0.0157	0.0224	0.0	0.0	0.0269	0.0011	0.0011
P@10	0.0258	0.0123	0.0101	0.0191	0.0	0.0	0.0168	0.0005	0.0011
P@20	0.0193	0.0095	0.0053	0.0154	0.0	0.0	0.0115	0.0002	0.0011
TOP20	0.2640	0.1348	0.1011	0.2191	0.0	0.0	0.1966	0.0056	0.0224
TOP50	0.3820	0.1910	0.1348	0.3202	0.0	0.0	0.2640	0.0112	0.0280
MRR	0.0813	0.0564	0.0633	0.0540	0.0	0.0	0.0876	0.0029	0.0046
MAP	0.0279	0.0151	0.0118	0.0159	0.0	0.0	0.0194	0.0004	0.0009
NDCG@20	0.0479	0.0279	0.0182	0.0322	0.0	0.0	0.0366	0.0008	0.0026

Pair Ranking Method も概ね All Ranking Method と同様の結果となった．二つのランキング方法を比較するために，図 6.1，図 6.3 の違いを観測すると，All Ranking Method の方が MAP，NDCG@20，P@k の値が大きい事が分かった．また，MRR に関しては概ね Pair Ranking Method の方が高い傾向となった．TOP 値に関しては，① TF-IDF や④ WTMF モデ

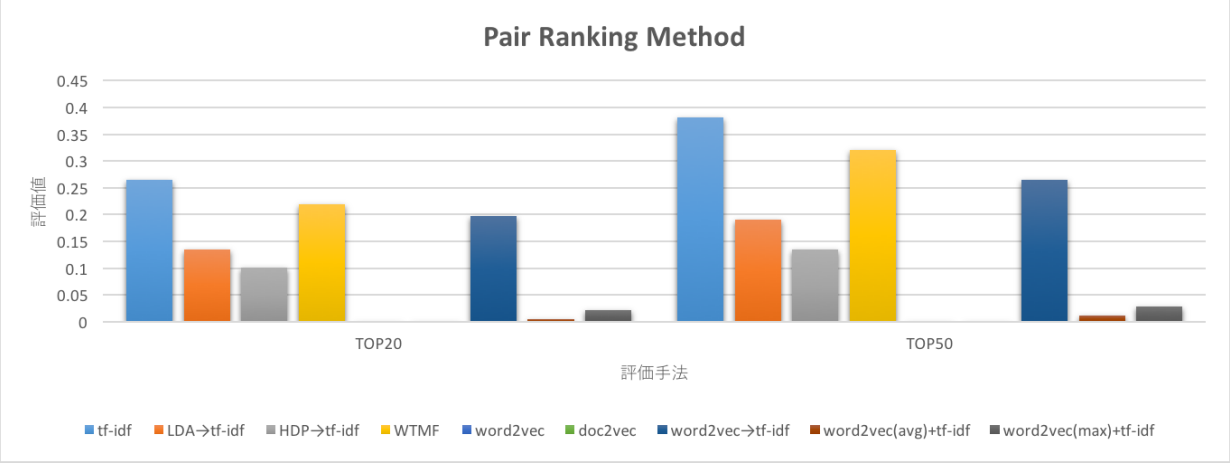

図 6.4: Pair Ranking Method におけるランキング内正解含有率調査

ルは Pair Ranking Method の方が All Ranking Method に比べ高い数値であったが，② LDA → TF-IDF，③ HDP → TF-IDF ，⑦ word2vec → TF-IDF は All Rank Method の方が高い数値となった.

第7章　考察

7.1　ランキング方法

今回の実験で実装した Pair Ranking Method と All Rank Method であるが，実験の結果から All Ranking Method は MAP，NDCG@20，P@k，Pair Ranking Method は MRR が相対的に大きく，TOP 値に関しては手法によって異なる，という結果となった．このことから，All Rank Method の方が Pair Ranking Method に比べてランキング全体の質が良いという事が分かった．従って，類似しているツイート自身も今回のデータでは返答になりうる可能性を示唆している．MRR においてのみ Pair Ranking Method の評価値が高い原因として，Pair Ranking Method が付随するツイートのみを評価値として使用し，かつ All Ranking Method では不正解データとしてランクされている類似ツイート (この場合自身が類似しているため，ランクされたツイートを指す) が存在するため，付随するツイートが正解データであった場合，このツイートは All Rank Method よりも上位にランキングされる可能性が高くなることが挙げられる．また，All Rank Method でのランキング順位が 50〜70 位の付随正解ツイートの場合，Pair Ranking Method では 50 位以内にランキングされる可能性が高いことから，MRR 値・TOP 値の微小な差異が生じると推察される．

つまり，今回のこの二つのランキング手法は，上位にランクされる “類似正解ツイート ” を取るか，All Rank Method で 50〜70 位の “付随正解ツイート ” をランキングに含めるかのトレードオフのような関係に成っていると考えられる．今回のデータセットに関しては，上位にランクされてるであろう正解類似ツイートをランキングに含めた方が MAP 値や NDCG 値が良かったため，All Rank Method の方が全体的な精度は良かったと考えられる．ただ，この結果はデータ依存によるものなので，コーパスがただの会話文でなく質問 – 回答文のみで構成されていた場合は All Rank Method の方が評価が低くなると考えられる．

表 7.1 に① TF-IDF (All Ranking Method) の結果を載せている．表 7.1 内の sim と rep は，類似度が高いツイートとしてランクしたか (sim)，付随ツイートとしてランクしたか (rep) の違いである．この結果を見ても分かる通り，正解ツイートとしてアノテートされているツイートは基本的にペアでアノテートされている可能性が高いため，類似ツイートを含める All Ranking Method の方が正解データをランキング内により多く含めることができたと考える．

表 7.1: TF-IDF(All Ranking Method) によるランキング 1(一部抜粋)

入力ポスト: 金沢まで特急あるけど金沢は車で行きたいかな				
順位	正解	post/comment/residue	similarity/reply	text
20		cmnt	sim	金沢参戦するんですか？？
21	●	cmnt	rep	小矢部、森本間通行止めらしいぞー
22	●	post	sim	土曜の 23 時ぐらいに高速で金沢行かないといけないんだが生きて金沢着けるかな・・・
23		post	rep	雨の中キャリーを引きつつお出かけ　待ってて金沢 のゥの
24		cmnt	sim	おろ？ スパイさんも金沢？ 自分も家族旅行で金沢でござる。
25		post	rep	無事帰宅！ 昨夜の金沢 vanvan v4 お集まり頂いた皆さんありがとうございました！ 昨日のお客さんもカッコ良かった！ THE NEAT BEATS はもう言わずもがなっすね★すげぇに決まってる！ 笑 自分達としては初の金沢ライブでしたが、精一杯歌って来ました！ また必ず帰って来ます！ 感謝！
26		cmnt	sim	お疲れ様でした。無事のご帰宅よかったです！ 金沢の初ライブ行きたかったなー。
27		post	rep	マジか、神奈総が上がってこなかったのが意外すぎる そして新羽は流石すぎ っていうか神奈総と金総って分かりにくい、どっちも音は「かなそう」だし、どっちも神奈川だし 神奈川総合と金沢総合・・・だっけ？
28		cmnt	sim	金沢総合と金沢高校も……
29	●	post	rep	金沢に遊びに行きました！ お絵かきもした！！ 楽しかった！！
30	●	cmnt	sim	僕金沢在住です笑
31	●	resi	sim	金沢から福井は特急ですぐだったな 対バンの時もすぐ行けるか
32		cmnt	rep	さすが pioneer
33		post	sim	あ、明日の 7 時半から金沢にいますので遊んでください
34	●	post	rep	金沢へゴー ?″ ・屑 ?″ ・屑 ?″ ・屑
35	●	cmnt	sim	げんちゃんも今日金沢やーいってらっしゃいー！
36		post	rep	Facebook でイベントの招待する相手が遠方で絶対来んやろって距離なのに参加しますっていう返事きたら、嬉しいけど嘘つくな！ って思う。

7.2 ベクトル生成手法の有効性

All Rank Method と Pair Ranking Method では同様の傾向を示しているため，より性能の高かった All Rank Method の結果に基づいて考察する．本実験において評価値が最も高かったものは，主に① TF-IDF と⑦ word2vec → TF-IDF であった．両者の違いとして，TOP 値・MAP 値が高かったのは TF-IDF であり，MRR・P@k は word2vec → TF-IDF が概ね高い結果となった．この理由として，TF-IDF のみの結果はコーパス全てに対して類似度計算を行っているのに対し，word2vec → TF-IDF は word2vec で入力ポストの各単語に対して類似している単語を取り出し，その単語を含むツイートのみに対して類似度計算しているので，word2vec が単語によるトピックフィルタリングのような役割を果たしていると考えられる．従って，TF-IDF は全体的にランキングに正解データが入る割合が高くなるが，その分不正解データもランキングに入っているので，ランキングの質は低下する．それに対し，word2vec → TF-IDF は類似単語によって観測するツイートを制限するため，正解データが漏れてしまう可能性が存在するが，正解データがランキングに入った場合，ノイズデータが取り除かれているため，比較的上位にランクされやすいと考えられる．

また，トピックモデルである② LDA → TF-IDF，③ HDP → TF-IDF は純粋な TF-IDF に比べ，MAP 値や NDCG 値が低く，ランキングの質が悪いという結果となった．これはトピックモデルが，Twitter テキストデータの疎性のために，コーパスをベクトル空間にモデル化することに失敗していることが原因として考えられる．実験で扱ったツイートデータがスパースであるかどうか検証するために，データの単語数に着目した．また，今回のコーパスで単語生起頻度が 1 回の単語と高頻度語は，類似度計算にほとんど寄与しないことから，データから除去している．その結果を表 7.2 に示す．表 7.2 から分かる通り，抽出された

表 7.2: コーパスデータ内訳

全ツイート数	920,643
Post(Comment) tweet 数	428,124
Residue tweet 数	64,395
全単語数	202,584
除去後の単語数	110,934

た単語 202, 584 語の内，約半数の 9 万語近くは出現頻度が一回しかないものであった．このことから今回扱ったデータがスパースなデータであったことが分かる．

本実験で，⑤ word2vec，⑥ doc2vec，⑧ word2vec(average) + TF-IDF，⑨ word2vec (max) + TF-IDF は，全く正解データを当てる事が出来ないという結果となった．doc2vec は，Dis-

tributed Memory Model を採用したが，このモデルでは，CBOW モデルに文書を表したベクトルの情報を入力に追加して文書ベクトルを求める (図 3.5)．従って，文書内の語順を情報に組み込むことで通常の単語生起によるベクトル生成よりも一般に精度が高いと言われているが，今回のコーパスは表 7.2 から分かるようにデータのスパース性が大きいものであるので，ウィンドウサイズの存在 (周辺単語を観測すること) がデータの疎性によるモデルの劣化に繋がったのではないかと考えられる．

⑤は，word2vec の単語ベクトルの線形和によってテキストベクトルを生成したが，結果は最も評価値の悪い結果となった．word2vec のモデルの精度を観測するために，図 7.1 と図 7.2 に注目単語と類似する単語上位 20 件を可視化した．この結果を見る限りでは, word2vec は単語のトピックをうまく分類しているように見える．しかし，実際のランキングには正解データが入ることはなかった．具体例として，表 7.3 に実際の⑤ word2vec の結果の一部を示す．この例では，入力ポストが “食” をテーマにしたテキストになっており，ランキングに挙げられる類似ポストの多くは食べ物の単語で構成されている．しかし，この入力ポストに含まれる “ローストビーフ” や，“アヒージョ” という食べ物の種類の近いツイートはランキングに入らなかった．従って，word2vec の線形和によるベクトル演算は今回の短文のトピックまでは分けることができるが，より具体的な内容や応答文を抽出する精度に至らなかったと考える．

⑧ word2vec(average) + TF-IDF，⑨ word2vec (max) + TF-IDF に関しても，コーパス全体から単語の重みを付ける TF-IDF と word2vec のよる単語ベクトルから単語間距離の情報を組み合わせることにより，テキストの情報量が増すと考えたが，組み合わせることにより，word2vec が逆にノイズになってしまい，① TF-IDF に比べて結果が悪くなってしまったものと思われる．

また，TF-IDF がもっとも高い原因の一つとして，正解データのアノテート方法が TF-IDF に近い方法でなされたのではないかと考えられる．表 7.4 に具体例を載せる．この例では，特に入力ポストの “社会不適合者” という単語に対して類似しているテキストが TF-IDF によって取り出されている．正解データに注目すると “社会不適合者” という単語の入ったツイートのポストとコメントのペアに対してアノテートされていることから，このツイートに関しては TF-IDF に近い方法でこのツイート群を探し出し，それをペアごとアノテートした可能性が考えられる．このことから，④ WTMF が TF-IDF に比べて低い値を取っているのも正解データのアノテートの偏りも要因の一つとして考えられる．

図 7.1: “ツイート” と類似する単語

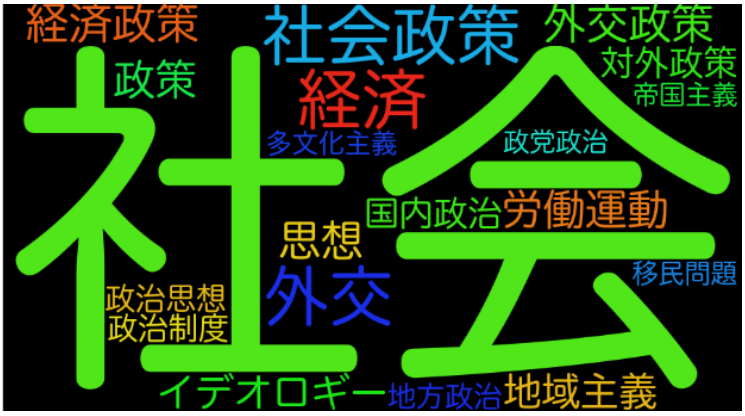

図 7.2: “政治” と類似する単語

表 7.3: word2vec(All Ranking Method) によるランキング (一部抜粋)

入力ポスト: 帰ったら夜ご飯がローストビーフとアヒージョだった www お洒落か！！！				
順位	正解	post/comment/residue	similarity/reply	text
1		cmnt	rep	きくも食べたいよぅー作ってー？
2		post	sim	兄貴兄貴ー！ クレープ！ パフェ！ 宇治金時ーー！！！！ あとあとー、ホットケーキ！ たーべたい！
3		cmnt	rep	それに対してこばやしは…
4		post	sim	前から気になってたけど、モリケンさん毎日ラーメンとビール飲んでるのに太らないんだけど
5		cmnt	rep	カラスミパスタがんばります
6		post	sim	多分ブチャラティがいたら妖怪ポルチーニ茸を延々と出し続けるババアになってしまうからカラスミソーススパゲッティ班とホタテのオーブン焼き班は頼んだ

表 7.4: TF-IDF(All Ranking Method) によるランキング 2(一部抜粋)

入力ポスト: おーい言われてんぞ、社会不適合者だって				
順位	正解	post/comment/residue	similarity/reply	text
1	●	cmnt	rep	ととろが社会適合者みたいな
2	●	post	sim	社会不適合者ほど辞めたくなるのが大学というものだと思います
3	●	cmnt	rep	「社会が俺に適合すれば良いんだよ！」という感じに強く
4	●	post	sim	少しずつ環境が変わっていく。この先、今までみたいにぬるま湯に浸かったままじゃいられないんだよな。強くならなきゃいけない。社会不適合者は社会不適合者なりに、強く。
5	●	cmnt	rep	ネガネガしてると全てに影響してくるぞ 気持ちだけでも前向きに行きましょうぜ
6	●	post	sim	私みたいな社会不適合者を学校に通わせる事自体、間違ってると思います！！！
7		post	rep	昨日マック行ったら店員さんめっちゃ態度悪くてすぐ帰った
8		cmnt	sim	そういうの是正されないってほんとおかしいよね 社会不適合者がどんどん増えてくと思うんだけど
9		post	rep	初出勤…気を使われているのが分かってより辛い。分かってはいるけれど。
10		cmnt	sim	私は社会不適合者なので復職不能でした。出勤しているだけでも凄いことだと思います。
11	●	cmnt	rep	間違ってないから困る
12	●	post	sim	やめなよ学園祭のこと社会不適合者の見本市みたいな言い方するの
13		post	rep	うんちょこそ人間のつくりだした最高の一品…

第8章　結論

本報告では，Twitterの日本語テキストデータを用いて，入力文に対し適切な応答文となるようなテキストを検索するアプローチにより，自動応答システムを実装した．ツイートのような短文を検索するには，出現単語が疎となるため，類似度評価が困難となる．そこで類似度計算に用いる文書ベクトル生成方法として，トピックモデルや，行列分解，ニューラルネットワークなどの手法を用いて，短文の類似度推定に適するベクトル生成手法を検証した．結果として，word2vecによる生成した単語ベクトルを用いて，入力ツイートに現れる単語の類似単語を含むツイートを抽出し，TF-IDFを用いて生成したベクトルによる類似度を計算し，ランキングする手法が最も評価値が高かった．しかし，ランキング20位以内に正解データがランクされる割合は，35%に留まった．一因として，正解データとして全てのツイートが検証されているわけではなく，本来正解となるはずのツイートもアノテーションの欠如により不正解となること，応答として適するかどうかと類似しているかどうかが必ずしも一致しないことが挙げられる．

本研究では，システムの汎用性を考慮してテキストデータのみを使用したが，Twitterであればユーザのプロフィールデータなどの外部のデータなどを用いて情報を追加することによって，評価値を向上させることができる可能性も考えられる．また，教師あり学習でポストとコメントのペアを学習させることでも異なる結果が得られると考えられる．上記のような点を，システムの改善策として，今後の研究に生かす予定である．

謝辞

本研究の全過程を通して多大な御教授，御援助を賜りましたマルチメディア工学専攻ビックデータ工学講座の鬼塚真教授に厚く感謝の意を表します．

また，本研究を遂行し，本報告書作成にあたり，日々，懇切丁寧な御指導，御協力を頂きました本研究室の荒瀬由紀准教授に心より御礼を申し上げます．そして，研究テーマ発表会等を通じて御討論，御支援を頂きました博士前期課程 1 年の近藤舜介氏，藤森俊匡氏，水嶋海都氏，水野陽平氏，本研究室学部 4 年の伊藤竜一氏，高田祥平氏，筒井道紀氏に感謝致します．

参考文献

[1] G. Cong, L. Wang, C.Y. Lin, Y.I. Song, and Y. Sun, “Finding question-answer pairs from online forums,” Proc. of SIGIR, pp. 20–24, July 2008.

[2] S. Brin, and L. Page, “The anatomy of a large-scale hypertextual web search engine,” Computer Network and ISDN Systems, vol. 30, no. 1–7, pp. 107–117, April 1998.

[3] S. Brin, and L. Page, “The PageRank citation ranking: Bringing order to the web,” Technical report, Stanford University, January 1998.

[4] C. Shah, and J. Pomerantz, “Evaluating and predicting answer quality in community QA,” Proc. of SIGIR, pp. 19–23, July 2010.

[5] W. Xin-Jing, T. Xudong, F. Dan, and Z. Lei, “Ranking community answers by modeling question-answer relationships via analogical reasoning,” Proc. of SIGIR, pp. 179–186, July 2009.

[6] Z. Guangyou, C. Li, Z. Jun, and L. Kang, “Phrase-based translation model for question retrieval in community question answer archives,” Proc. of ACL-HLT, pp. 653–662, June 2011.

[7] R. Yan, M. Lapata, and X. Li, “Tweet recommendation with graph co-ranking,” Proc. of ACL, pp. 516–525, July 2012.

[8] W.X. Zhao, J. Jiang, J. Weng, J. He, E.P. Lim, H. Yan, and X. Li, “Comparing Twitter and traditional media using topic models,” Proc. of ECIR, pp. 338–349, April 2011.

[9] D.M. Blei, A.Y. Ng, and M. Jordan, “Latent Dirichlet Allocation,” Journal of Machine Learning Research, vol. 3, pp. 993–1022, May 2003.

[10] W.X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.P. Lim, and X. Li, “Topical keyphrase extraction from Twitter,” Proc. of ACL, pp. 379–388, June 2011.

[11] T. Hofmann, “Probabilistic latent semantic indexing,” Proc. of SGIR, pp. 50–57, August 1999.

[12] W. Guo, and M. Diab, "Modeling sentences in the latent space," Proc. of ACL, pp. 864–872, July 2012.

[13] W. Guo, H. Li, H. Ji, and M. Diab, "Linking Tweets to news: A framework to enrich short text data in social media," Proc. of ACL, pp. 239–249, August 2013.

[14] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," arXiv preprint arXiv:1410.0342, 2014.

[15] E. Gabrilovich, and S. Markovitch, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," Proc. of IJCAI, pp. 1606–1611, January 2007.

[16] M. Tomas, Y. Wen-tau, and Z. Geoffrey, "Linguistic regularities in continuous space word representations.," Proc. of HLT-NAACL, pp. 746–751, June 2013.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint, arXiv:1301.3781, 2013.

[18] M. Tomas, S. Ilya, C. Kai, C.G. S, and D. Jeff, "Distributed representations of words and phrases and their compositionality," in Advances in Neural Information Processing Systems 26, eds. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, pp. 3111–3119, Curran Associates, Inc., 2013.

[19] M. Tomas, L.Q. V, and S. Ilya, "Exploiting similarities among languages for machine translation," arXiv preprint arXiv:1309.4168, 2013.

[20] J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation," Proc. of EMNLP, vol. 32, pp. 1532–1543, October 2014.

[21] Y.W.Teh, M.I.Jordan, M.J.Beal, and D.M.Blei, "Hierarchical dirichlet processes," Journal of the American Statistical Association, vol. 101, pp. 1566–1581, November 2006.

[22] D.D. Lee, and H.S. Seung, "Learning the parts of objects by non-negative matrix factorization," Nature, vol. 401, no. 6755, pp. 788–791, October 1999.

[23] H. Harold, "Analysis of a complex of statistical variables into principal components.," Journal of educational psychology, vol. 24, no. 6, pp. 417–441, 1933.

[24] D.N. YW Teh, and M. Welling, "A collapsedvariational bayesian inference algorithm for latent Dirichlet allocation," In Advances in Neural Information Processing Systems, vol. 19, pp. 542–548, May 2007.

[25] B. Edward, H. Aria, and B. Regina, "Event discovery in social media feeds," Proc. of ACL-HLT, vol. 1, pp. 389–398, June 2011.

[26] N. Srebro, and T. Jaakkola, "Weighted low-rank approximations," Proc. of ICML, vol. 3, pp. 720–727, August 2003.

[27] Q.V. Le, and T. Mikolov, "Distributed representations of sentences and documents," Proc. of ICML, vol. 32, pp. 1188–1196, June 2014.

[28] Y. Song, and D. Roth, "Unsupervised sparse vector densification for short text similarity," Proc. of HLT-NAACL, pp. 1275–1280, May 2015.

付録A: Tweetデータセットの一例

NTCIR から配布されている評価用 Tweet ID セット一例

```
post                comment             annotate
613587908235112448 496282184613761024 0 0 0 0 0 0 0 0 0 0
613587908235112448 496279274530152448 0 0 0 1 2 0 1 0 0 0
613587908235112448 480913653370077184 1 0 1 0 1 NA 0 0 0 1
613587908235112448 480913383592427523 2 0 0 2 0 0 2 0 2 2
613587908235112448 447863828080914432 0 0 0 0 0 0 0 0 0 0
613587908235112448 447856545980620800 0 0 0 2 0 0 0 1 0 0
613587908235112448 529252432614678530 2 2 0 2 2 2 1 2 2 2
```

実際にクロールした Tweet セット (ID–Text) 一例

```
420140126195294208    どーしたの？　最近メンタル大丈夫？
420136048274849792    ワッショーイ！ 風吹いてきてますね!!
420136351183286272    前回どこいったん？ そーいや
420141356565032960    まじかーーー！ 早く帰らなくちゃ！
420137040655577088    え、これ分隊支援火器の類じゃないンすか…？
420137446454472704    何と無くシステム作って、何と無く敵出してるのが分かるんだよね。俺は詳しいんだ
420137615984041985    番号ネットで検索すべし!!!
420402134048194561    ぱぱー！ 付き合うよ！ 12 はひっとみとボードだから 12 はー？
420139896871731200    スチサイ二回縛りハウケタハードの出番ですな
420141380170563586    あまりに綺麗に空くから調子こいてベルト穴増やしすぎたりフローリングを抉ってしまうよね！
420141134837334017    痛いの痛いのとんでけー！！ って、妹ちゃんじゃあるまいしきかないか、ええと、大丈夫か?治るまで看病してやるから、しょげんな、な？
420140410342621184    今さら iPhone5 なん？ 笑 s とか c じゃないん？
420140405196206080    お疲れさまですー！
```

付録B: 各手法のパラメータチューニング

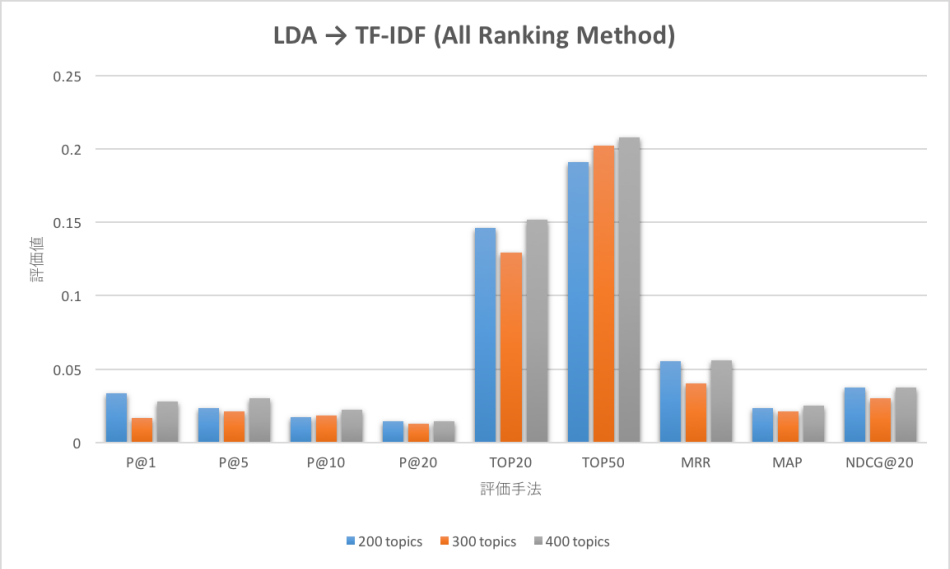

図 8.1: LDA (All Ranking Method) チューニング結果

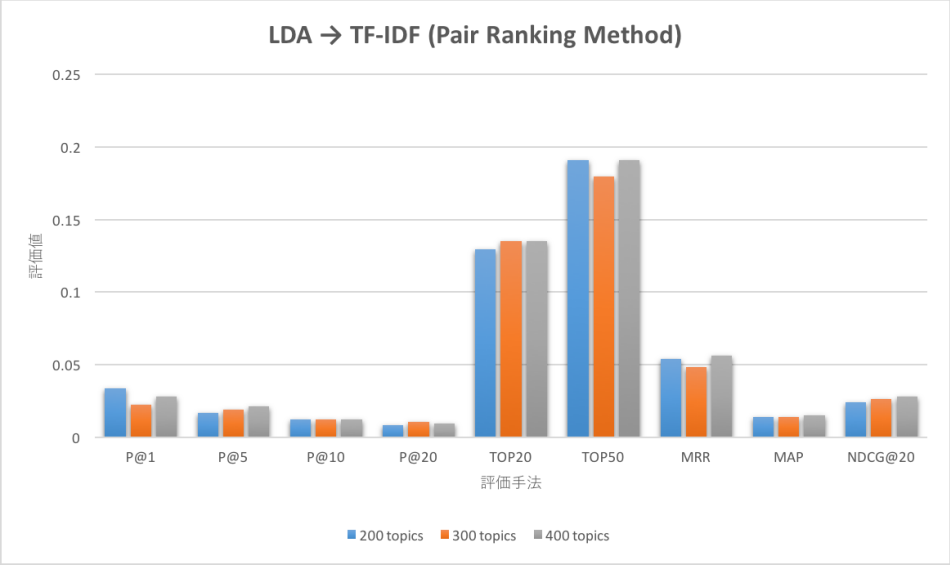

図 8.2: LDA (Pair Ranking Method) チューニング結果

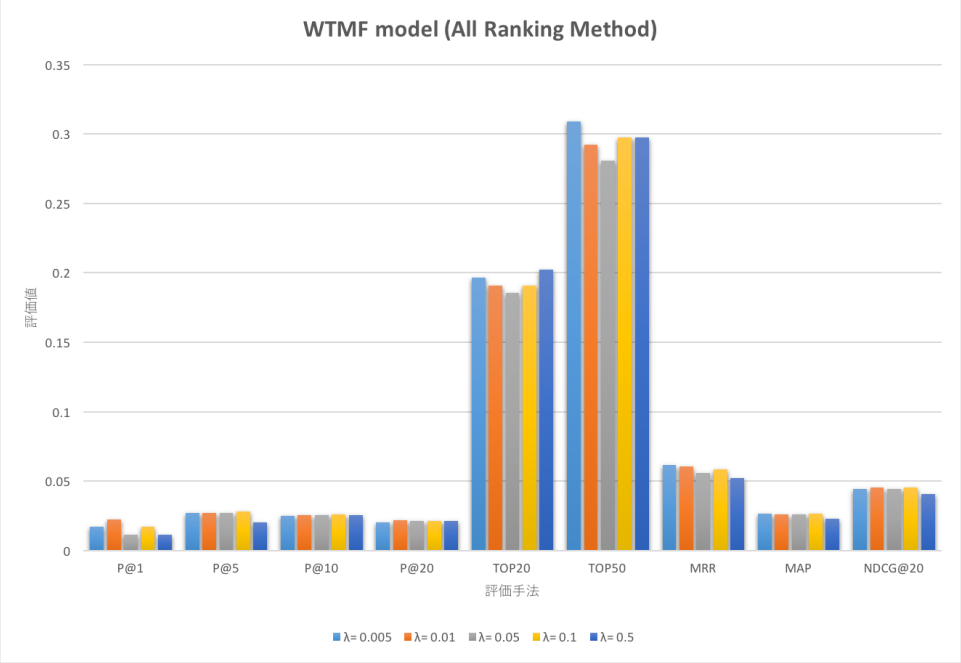

図 8.3: WTMF (All Ranking Method) チューニング結果

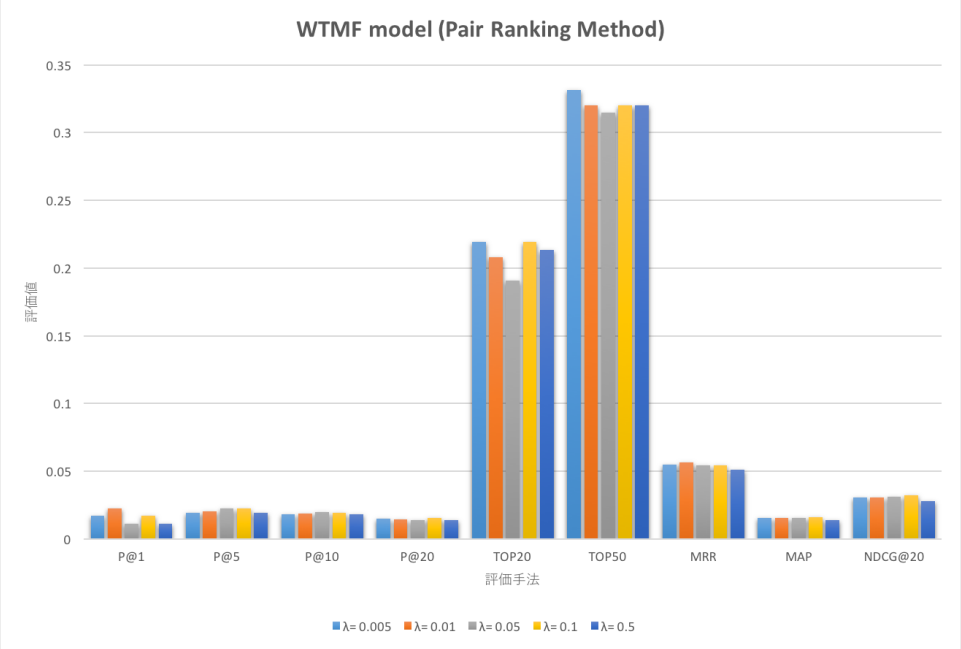

図 8.4: WTMF (Pair Ranking Method) チューニング結果

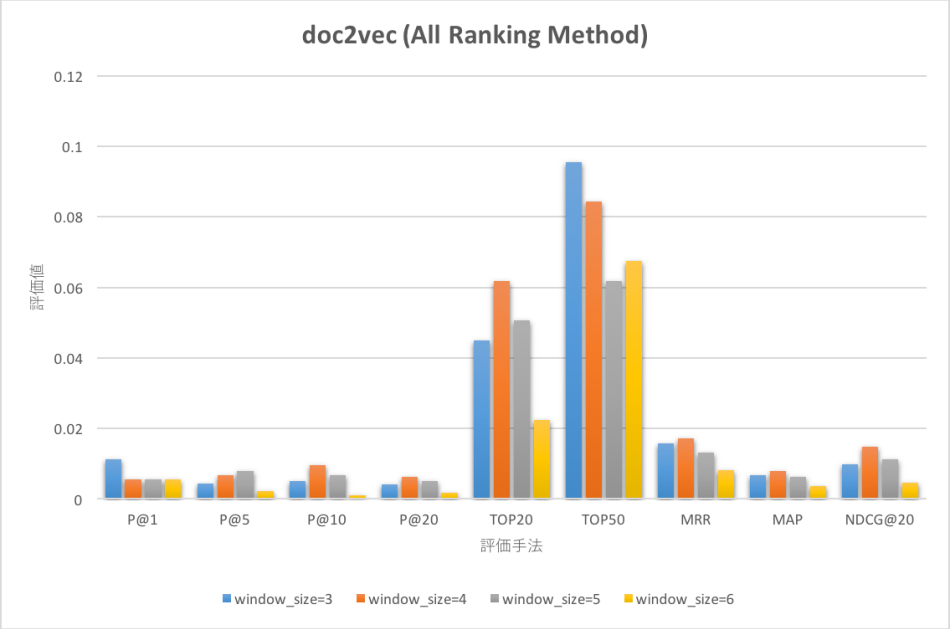

図 8.5: doc2vec (All Ranking Method) チューニング結果

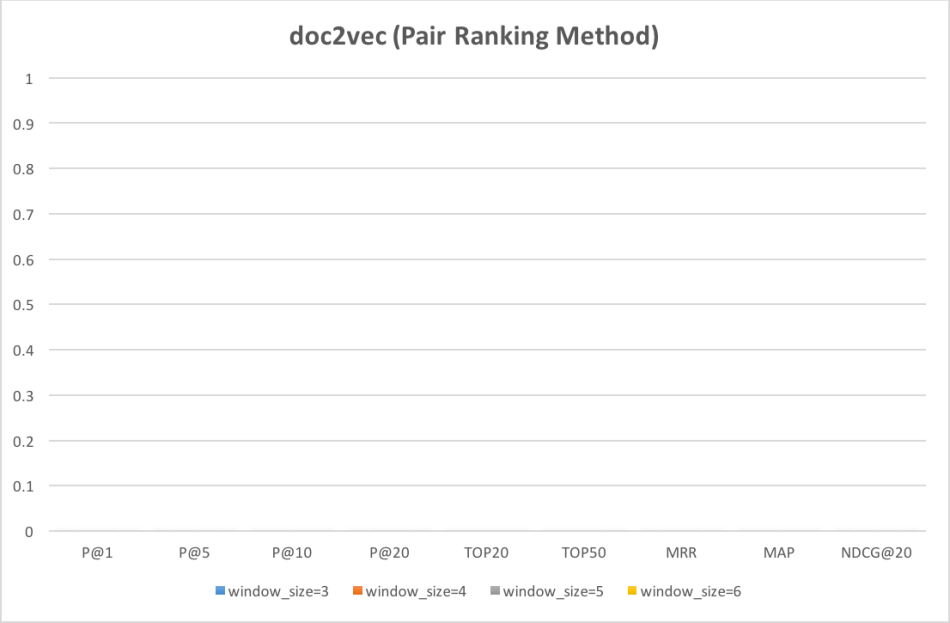

図 8.6: doc2vec (Pair Ranking Method) チューニング結果

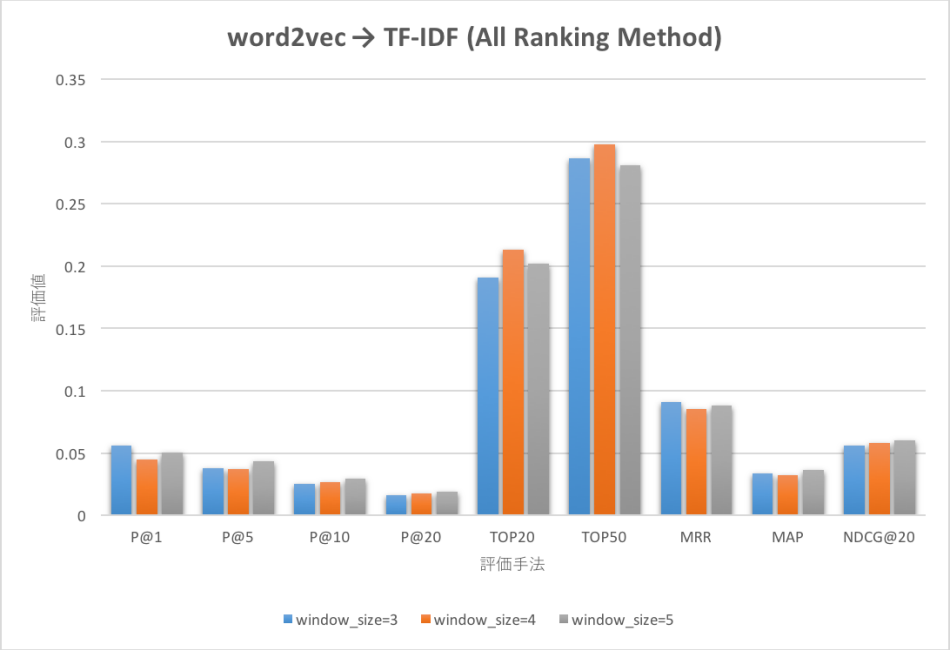

図 8.7: word2vec → TF-IDF (Pair Ranking Method) チューニング結果

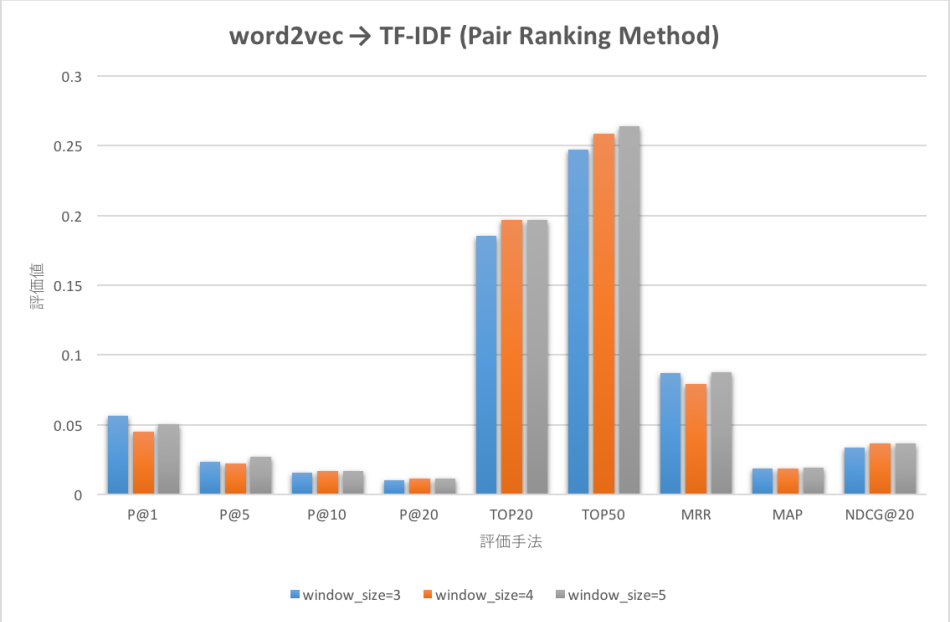

図 8.8: word2vec → TF-IDF (Pair Ranking Method) チューニング結果

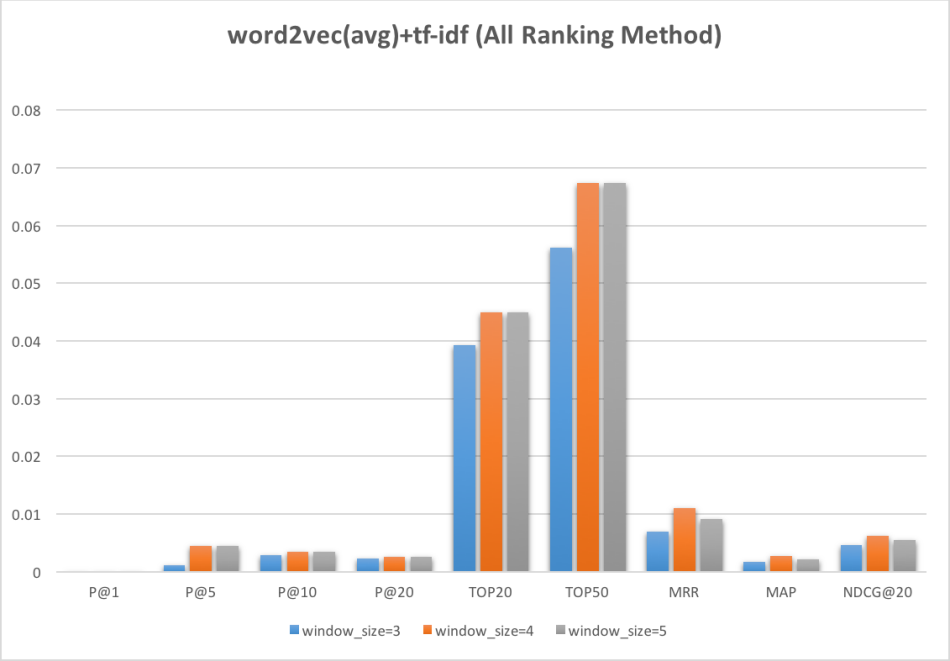

図 8.9: word2vec(avg) + tfidf (All Ranking Method) チューニング結果

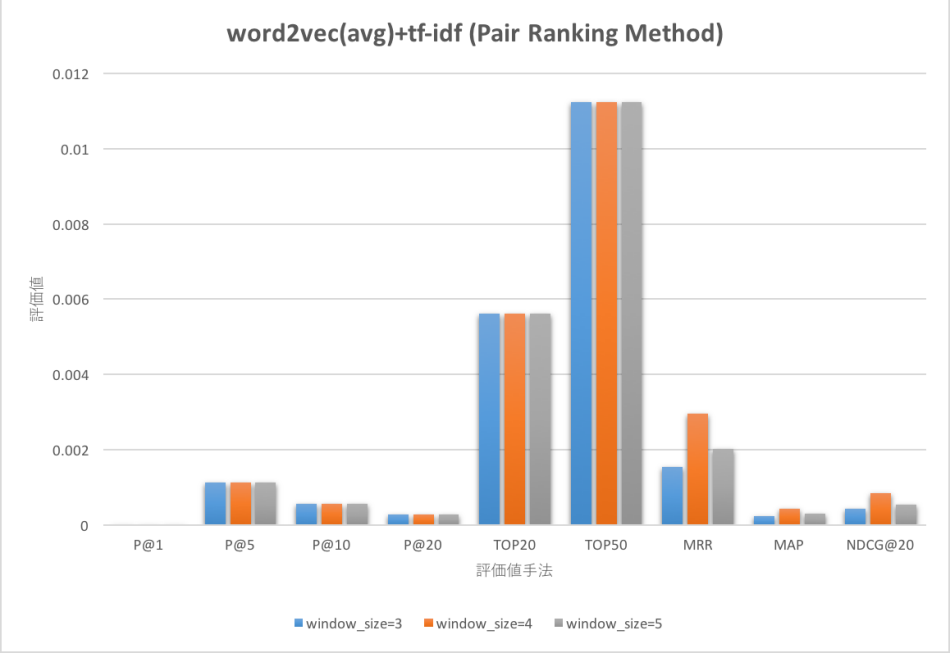

図 8.10: word2vec(avg) + tfidf (Pair Ranking Method) チューニング結果

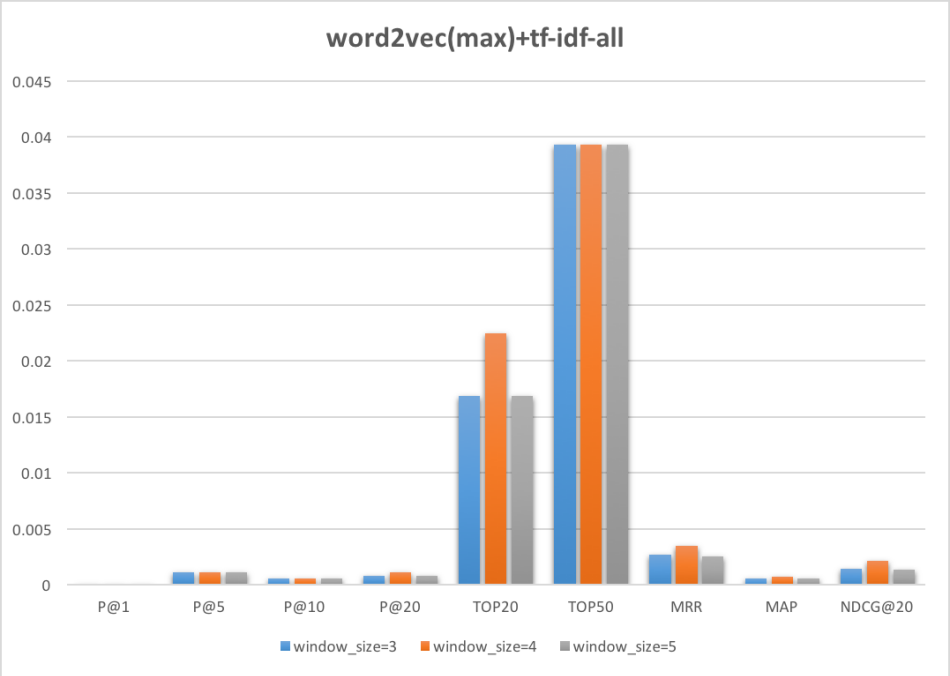

図 8.11: word2vec(max) + tfidf (All Ranking Method) チューニング結果

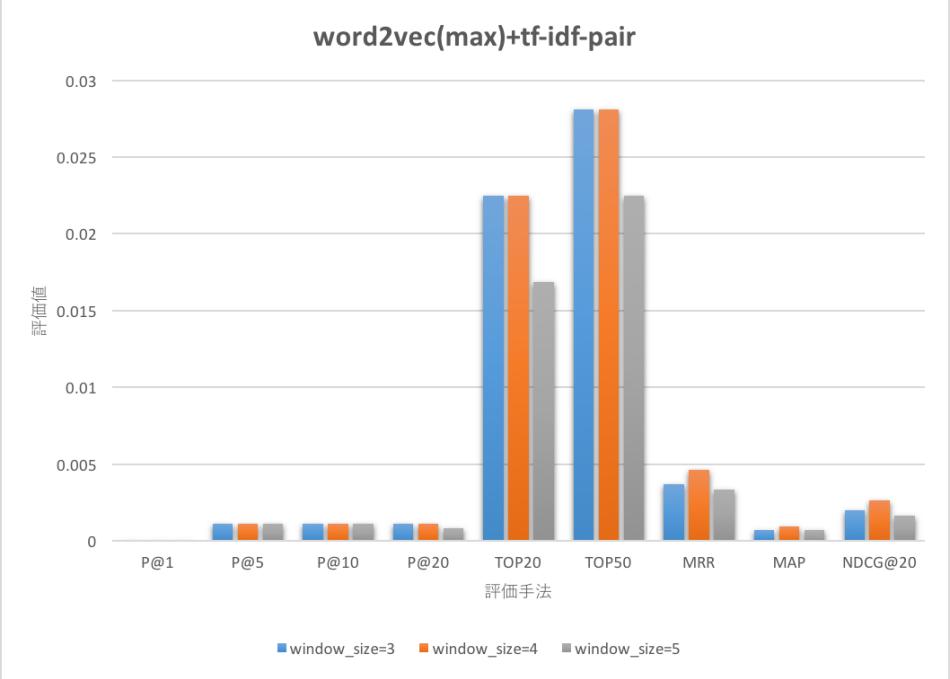

図 8.12: word2vec(max) + tfidf (Pair Ranking Method) チューニング結果