情報処理学会論文誌 Vol.777 No.1 1–6 (Jan. 2736)

分散表現を用いた話題変化判定

芳野 魁[1,a)]

受付日 2015年3月4日, 採録日 2015年8月1日

概要：大規模意見集約システム COLLAGREE ではファシリテーターと呼ばれる人物が議論のマネジメントを行っているが，長時間に渡って大人数での議論の動向をマネジメントし続けるのは困難である．ファシリテーターが画面を見なければならない時間を減らし，負担を軽減する工夫があることが望ましい．ファシリテーターが画面を見るべきタイミングは議論の話題が変化したときであると考える．すなわち，ファシリテーターの代わりに自動的に議論中の話題の変化を観測することが必要である．本研究ではファシリテーターに話題の変化を伝えるために，分散表現を用いて発言の繋がりの度合いを数値化し，繋がりの度合いが小さいもの，即ち話題の変化を起こしうる発言を検出することを目指す．

キーワード：自然言語処理，議論支援，話題遷移，COLLAGREE，分散表現

A Topic Change Judgment Method based on Distributed Representation

Kai Yoshino[1,a)]

Received: March 4, 2015, Accepted: August 1, 2015

Abstract:

Keywords: NLP，discussion support，topic shift，COLLAGREE，word embedding

1. はじめに

近年，Web 上での大規模な議論活動が活発になっているが，現在一般的に使われている ”2 ちゃんねる” や ”Twitter” といったシステムでは整理や収束を行うことが困難である．困難である原因として議論の管理を行う者がいないことを挙げることができ，「炎上」と呼ばれる議論の無秩序の状態が頻繁に観測されている．つまり，議論を整理・収束させるには議論のマネジメントを行う人物が必要である．大規模な意見集約を目的とした大規模意見集約システム COLLAGREE?が開発された．COLLAGREE では，掲示板のような議論プラットフォームをベースにしており，自由に意見を投稿することができる．COLLAGREE では議論を秩序的に進行し，収束させるためにファシリテーターと呼ばれる人物が議論のマネジメントを行っている．しかし，ファシリテーターは人間であり，長時間に渡って大人数での議論の動向をマネジメントし続けるのは大きな負担がかかり困難である．

COLLAGREE で大規模かつ長時間の議論を収束させるためには，ファシリテーターが必要な時には画面を見るようにして，他の時は見なくても済むようにすることで画面に向き合う時間を減らす工夫があることが望ましい．ファシリテーターが画面を見るべきタイミングは議論の話題が変化する，または話題の変化を起こしえる発言が投稿されたときである．以前の議論の内容から外れた発言が投稿された時，ファシリテーターが適切な発言をすることで，脱線や炎上を避けて議論を発展させながら収束させることができる．すなわち，ファシリテーターの代わりに自動的に議論中の話題の変化を判定することが求められている．

現在，COLLAGREE 上で使用されている議論支援機能で

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，名古屋工業大学
Presently with Nagoya Institute of Technology
[a)] joho.taro@ipsj.or.jp

はファシリテーターの作業量の減少には繋がりにくい．

近年，単語を実数ベクトルで表現する技術である分散表現は自然言語処理の分野において多くの研究で使われており，機械翻訳を始めとする単語の意味が重要となる分野で精度の向上が確認されている．分散表現を用いることで，従来の手法より単語の意味を考慮した処理が可能になり，内積計算を用いることで単語間の類似度を計算できる．

本論文では，新たに投稿された発言を調査し，分散表現を用いてファシリテーターの代わりに自動的に話題の変化を判定する手法を提案する．具体的には新しく投稿された発言を過去に投稿された発言と 1 つ 1 つ比較して類似度を計算する．類似度の計算は三段階で行われる．全ての過去の発言との総合類似度を計算し類似した発言があるかどうかで話題の変化の判定を行う．COLLAGREE 上で実際に行われた議論データを対象に評価実験を行う．提案手法がファシリテーターの代わりの自動的な話題の変化判定において有用であることを示す．

本論文の構成を以下に示す．2 章では COLLAGREE の概要と議論支援及び本研究で支援することを目標とするファシリテーターについて説明し，関連研究についても説明するとともに本研究との違いを述べる．更に本研究の重要な要素である重み付けと分散表現についても詳しく述べる．次に，**??**章では話題変化判定システムの全体モデル説明を行い，**??**章では分散表現を用いた発言内容の類似度計算について説明する．そして，5 章では話題の変化判定の評価実験について説明する．最後に**??**章で本論文のまとめと考察を示す．

2. 関連研究

議論支援を行う研究では小谷ら**?**は好意的発言影響度を取り入れた議論支援システムを開発した．システムは議論中の発言の意図や内容に加えて，発言に対するリアクション (同意、非同意、意見) などから議論進行をモニタリングしている．モニタリングの結果を基にして議論の活性化や深化に対して参加者が果たしている役割を”好意的発言影響度”として定量化して表示する．小谷らは一般参加者や学習者の議論活性化及び収束に向けた支援を目的としているが，本研究はファシリテーターに対する支援を目的としている点で異なる．

話題遷移を扱う研究では別所**?**は単語の共起頻度行列を特異値分解で次元を削減して作成した単語の概念ベクトルを用いて，連結された新聞記事を元の形になるように分割をする実験を行っている．テキストセグメンテーションの 1 手法である TextTiling を用いてブロック間の類似度を計算する際にブロック中の単語の中で自立語にのみ概念ベクトルを付与し，左右のブロックの和ベクトル (または重心ベクトル) の余弦測度を求め，類似度 (または結束度) としている．テキストセグメンテーションは基本的にある地点 A で話題に沿って分割をするか考える際に，地点 A より先の情報を使うことができる．すなわち，ある程度の文章が現れてから話題が変わったかどうかの判定をしており，リアルタイムでの動作を想定していない．本研究はリアルタイムでの動作を想定し，ある発言 B が話題の変化を起こすかどうか判定する際に B より先の情報を使うことなく判定している点で異なる．中村ら**?**は連結された新聞記事を LDA を用いてトピック変化点を検出して分割する実験を行った．中村らは次の手順でトピック変化点を検出した．

(1) 現在地点より前の L_0 個の形態素を処理対象範囲 $B_0(L_0$¿L：L=文章長の下限) とし，B_0 から複数のトピック変化点候補箇所を抽出する．

(2) B_0 を各トピック変化点候補箇所で 2 ブロックに分割し，2 ブロック間の類似度を計算する．そして類似度が最も小さくなるトピック変化点候補箇所 D_1 とその類似度 S_1 を求める．文書ブロック間の類似度は LDA によって求められるブロックのトピック混合比ベクトルを用いて算出する．

(3) B_0 を D_1 で分割し，現在地点に近い側のブロックを B_1 とする．
以降は以下 4. と 5. の処理を繰り返す ($n \geq 2$)．

(4) ブロック B_{n-1} に対し 2. と同様の手順でトピック変化点候補箇所 D_n とその類似度 S_n を求める．

(5) $S_n \geq S_{n-1}$ の場合，D_{n-1} をトピック変化点として処理を終了する．$S_n < S_{n-1}$ の場合，B_{n-1} を D_n で分割し現在地点に近い側のブロックを B_n として，処理を継続する．

なお，上記 (1) におけるトピック変化点候補は 1 文中でトピックが変化することは考えにくいことと計算コストを考慮し，文境界（句点出現位置）をトピック変化点候補としている．

トピックモデルはトピックに基づいて学習を行っていて，基本的に学習の際にトピックの数を指定する必要があり，指定に伴いトピック数が制限されてしまう．また，教師なし学習であるので必ずしも話題を捉えた学習がされるとは限らない．すなわち，トピックの数によっては変化に対応できない話題が存在することがあり得る．本研究はトピックの指定が必要な LDA によるトピックモデルとは違い，トピックに関係なく単語の共起頻度を学習する fastText を用いている点で異なる．更に，本研究は中村らの研究とは話題の変化点を判定する際に複数候補から類似度が最小である候補点を選ぶのではなく，候補点を逐一調べ閾値を下回るかどうかで変化点を判定している点や対象とするデータが新聞記事ではなく議論である点でも異なる．

3. 全体モデル

3.1 システム動作の流れ

擬似コードを **Algorithm1** に示し，図示したものを図 1 に示す．

Algorithm1 を用いてシステムの動作の流れについて説明する．発言 R が投稿された時，過去に投稿された発言と類似度の計算を行い類似度が閾値を超えていれば 2 つの発言が同じ話題であるとみなし，発言 R と同じ話題である発言の集合 SG に登録する．作業を繰り返し全ての発言との計算が終了した後，SG が空集合である，すなわち発言 R と同じ話題である発言がない場合に話題を変化させる発言であると判定して通知を行う．

Algorithm 1 システムの流れ

```
1: Input : 発言 R
2: Output : 通知判定 Notify
3: PG = 過去の発言の集合;
4: procedure TOPICCHANGE(R)
5:     SG = {};
6:     for Each pastR ∈ PG do
7:         sim = similarity(R,pastR)
8:         if sim > threshold then
9:             SG.append(pastR)
10:    Notify = False
11:    if SG == {} then
12:        Notify = True
13:    return Notify
```

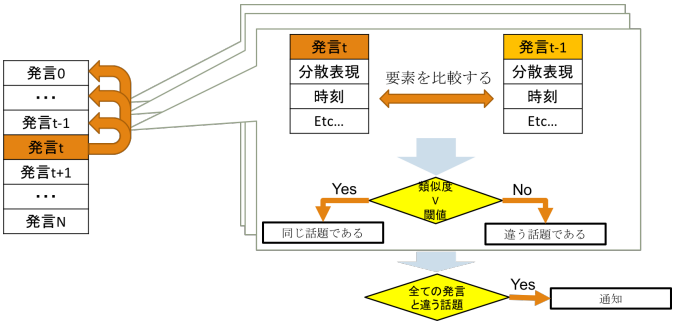


図 1　システムの流れ

3.2 発言間の類似度計算

発言間の類似度計算は次の 3 段階で行われる．

1 前処理

発言データ中の title と body，すなわち発言の内容文を対象に refmodel:simRemark:2 で行われる発言内容の類似度計算の精度を上昇させるためにストップワード (役立たないことから処理対象外とされる単語) の除去や単語の重み付けを行う．また，title が NULL でない場合は title と body を改行コードで繋いで 1 つの文章とする．前処理の詳細については**??**章で述べる．

2 発言内容の類似度計算

3.2 で行われた前処理の情報や分散表現を用いて発言内容文の類似度計算を行う．文章間の類似度計算の手法については**??**章で詳しく述べる．

3 総合類似度計算

上記の 3.2 で計算された発言の文章間の類似度に発言間の時間差と返信関係を組み合わせることで総合類似度を求める．

3.2.0.1 時間差評価値

発言 new と以前の発言 old 間の時間差を式 1 に基いて正規化された評価値として求める．

$$tValue = 1 - \frac{epoch(new.created) - epoch(old.created)}{maxTime} \tag{1}$$

ここで関数 $epoch$ 及び定数 $maxTime$，$x.created$ について説明する．$epoch$ は与えられたタイムスタンプをエポック秒に変換する．$maxTime$ は最大時間差を表し，基本的には議論の制限時間を用いる．$x.created$ は発言 x が投稿された時間を表す．時間差評価値は 2 発言間の時間差が小さいほど関連が強いとみなし，0 から 1 に近づく．

3.2.0.2 返信距離

発言 new と以前の発言 old 間の返信距離を Algorithm2 に基いて再帰的に求める．

Algorithm 2 返信距離

```
1: Input : 発言 new, 発言 old, 返信距離 dist        ▷ 初期値 dist=1
2: Output : 返信距離 dist
3: PG = ID に対応づけられた過去の発言の集合;
4: procedure REPLYDIST(new, old, dist)
5:     if new.parent-id == NULL then
6:         return 0
7:     else if new.parent-id==old.id then
8:         return dist
9:     else
10:        parent = PG[new.parent-id]
11:        dist+ = 1
12:        return REPLYDIST(parent, old, dist)
```

7 ～ 8 行目， 9 ～ 12 行目で示すように発言の id が一致した場合は現在の返信距離を返し，一致しなかった場合は返信距離を 1 増やして new の親発言 $parent$ と old の返信関係を再帰呼び出しで求め，返り値を返す．また，5 ～ 6 行目で示すように 2 発言間が返信関係になかった場合は 0 を返す．

3.2.0.3 総合類似度

総合類似度は前述の発言内容の類似度，時間差評価値，返信距離によって求められる．返信距離が 0 である時，すなわち 2 発言が異なるスレッドに属している場合は類似度と

時間差評価値から総合類似度を計算する．類似度だけでなく時間差評価値を使用するのは，総合類似度だけで判断してしまうと議論の終盤になって発言数が多くなってきた時に新しく投稿された発言が多くの古い発言と類似していると判断されてしまうことがあり得るからである．議論は基本的に少し前の発言に関連して行われることが多いことから時間差評価値を使用して時間的に近しいものほど総合類似度が上昇するようにする．具体的には式 2 のように計算される．

$$tSim = tValue * tWeight + sim * (1 - tWeight) \tag{2}$$

ここで変数 sim，定数 $tWeight$ について説明する．sim は発言内容の類似度を表し，0 から 1 の値を取る．$tWeight$ は時間差評価値の総合類似度の計算における重要性を表し，0 から 1 の値を取る．また，返信距離が 0 でない，すなわち 2 発言が同じスレッドに属している場合は何らかの関連があると考えられることから，時間差評価値を無視して発言内容の類似度を直接，総合類似度として用いる．

4. 発言内容の類似度計算

4.1 前処理

分散表現による類似度計算で精度を上昇させるためには発言内容から余分な単語を取り除きいて重要な単語を抽出する，または極めて短く要約することが重要である．提案手法では形態素解析エンジン MeCab と**??**節で説明した okapiBM25 と LexRank を用いて発言の文章から重要単語を抽出し，抽出した単語の類似度を分散表現を用いて計算する．MeCab による形態素解析の結果，次の条件を満たす単語を除外している．

(1) 品詞細分類に「数」を含む

(2) 「読み」,「発音」が不明である

(3) 品詞が「助詞」,「助動詞」,「記号」,「連体詞」のどれかである．

(4) 1 文字のひらがなである

(5) 品詞細分類に「接尾」または「非自立」を含む

上記の条件を満たす単語を除外したのは 4.1.1 節で説明する重み付けにおいて重要な単語であると判定されやすいが，**??**節で説明する類似度計算において精度を下げてしまうからである．単語の除外は重み付けにおいて文章を単語に分割する際に行われる．

4.1.1 重み付け

提案手法では**??**節で説明した okapiBM25 と LexRank の 2 種類の重み付け手法をを統合して発言の内容の文字列 $remark$ 中の単語に対して重み付けを行う．アルゴリズム

Algorithm 3 統合重みの計算アルゴリズム

```
1: Input : remark 発言内容の文字列
2: Output : combinedWeight remark 中の単語と重みを対応付けた連想配列
3: Array  sentList;
4: procedure calcCombinedWeight(reamrk)
5:     bm25Weight = calcBM25Weight(remark)
6:     for Each sent ∈ remark do
7:         sentList.append(sent)
8:     lexWeight = calcLexRank(sentList)
9:     for Each word ∈ bm25Weight.keys() do
10:        wordWeight = bm25Weight[word]
11:        if word is 固有名詞 then
12:            wordWeight *=2
13:        sentWeight = 0
14:        for Each sent ∈ remark do
15:            if word in sent then
16:                sentWeight+=lexWeight[sent]
17:        combinedWeight[word] = wordWeight*sentWeight
18:    return combinedWeight
```

を **Algorithm3** に示す．固有名詞は文章の中で重要な役割を果たす可能性が大きいと考え，12 行目では固有名詞の単語重みを倍にしている．そして，14 ～ 17 行目では word を含む全文章の重みの合計を求め，okapiBM25 による単語重みを掛け合わせたものを word の統合重みとしている．単語重みに単語を含む文章の重みを掛け合わせることで感嘆文のような文章そのものは重要でないが頻度の少ない単語を使用する文章中の単語が選ばれる可能性を下げている．

4.2 類似度計算

4.3 単語抽出

4.1 節で計算された単語重みの値が大きいものの上位 n 個までの単語を発言文章 remark において重要度の高い単語であるとして抽出する．単語重みが等しいものが複数あった場合は単語を昇順に並び替えて順序を付けている．また，使用する分散表現モデルに登録されていない単語は除外している．

4.3.1 分散表現による類似度計算

4.3 節で述べた手法を用いて 2 発言それぞれから抽出した単語集合の類似度を分散表現を用いて求める．それぞれの単語集合の単語ベクトルの平均を求め，**??**の図**??**で述べたように Cosine 類似度を 2 平均ベクトル間で取っている．

5. 評価実験

5.1 対象データ

5.1.1 議論データ

議論データは COLLAGREE 上で行われた別の実験での議論のものを使用する．

5.1.2 評価データ

5.1.1 節で説明した議論データに対し，次に述べる基準でアノテーションを行ってもらった．基準を満たすと思われる発言に”1”のタグを，満たすと思われない発言に”0”のタグを付ける．

5.2 実験設定

5.2.1 パラメーター

本実験ではパラメーターは次の通りに設定した．前処理にて用いる okapiBM25 のパラメーターは k1=2, b=0.75 とし，LexRank のパラメーターは n=50，threshold=0.7 とした．また，重み付けを用いて文章から抽出する単語の数は 5 個とした．分散表現として用いる fastText は次元数を 100 次元とし，学習データには wikipedia ダンプデータを用いた．総合類似度の計算に用いるパラメーターは maxTime=5400(90 分)，tWeight = 0.5 とし，総合類似度の閾値は 0.8 とした．表 5.1 に実験の設定をまとめる．

5.2.2 比較手法

5.2.2.1 1 常時通知

okapiBM25	k1	2
	b	0.75
LexRank	n	50
	threshold	0.7
抽出単語数		5
fastText	次元	100
	学習データ	wikipedia ダンプデータ
maxTime		5400
tWeight		0.5
類似度閾値		0.8

表 1 パラメーターの設定

最も単純かつ分かりやすい比較手法として，発言の内容に関係なく常に通知を行う手法を用いる．

5.2.2.2 2 TF-IDF ベクトル

単語の意味は考慮せず出現頻度に基づく比較手法として，分散表現の代わりに TF-IDF で発言をベクトル化する手法を用いる．**Algorithm3** の**??**行目で okapiBM25 の代わりに TF-IDF を用いて連想配列を求め，重みのベクトルに変換する．発言内容の類似度計算は提案手法と同じで Cosine 類似度を用い，以降の総合類似度も提案手法と同じである．

5.2.3 評価指標

本実験では評価指標として適合率 (Precision)，再現率 (Recall)，F 値 (F-measure) の 3 種類の指標を用いる．
適合率，再現率，F 値はそれぞれ次のようにして求める．まず，発言の通知を行うと判定した時を予測値=1，通知を行わないと判定した時を予測値=0 とおく．次に，予測値=1 かつ正解値=1 であるものの個数を $TP(True\ Positive)$，予測値=0 かつ正解値=1 であるものの個数を $FP(False\ Negative)$，予測値=1 かつ正解値=0 であるものの個数を $FP(False\ Positive)$ として数える．また，予測値=0 かつ正解値=0 であるものの個数を $TN(True\ Negative)$ として数える．

5.3 実験結果

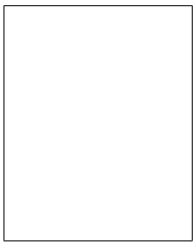
実験結果を表 2 に示す．

手法	平均評価指標		
	Precision	Recall	F-measure
比較手法 1	0.256579335	1	0.404074977
比較手法 2	0.75	0.105429708	0.180947229
提案手法	0.517148273	0.558192262	0.509950195

表 2 実験結果

謝辞

参考文献



情報 太郎 （正会員）

1970 年生．1992 年情報処理大学理学部情報科学科卒業．1994 年同大学大学院修士課程修了．同年情報処理学会入社．オンライン出版の研究に従事．電子情報通信学会，IEEE，ACM 各会員．本会シニア会員．