平成29年度　　卒 業 論 文

分散表現を用いた
話題変化判定

提出日　　平成29年9月19日

所属　　名古屋工業大学 情報工学科

指導教員　伊藤 孝行 教授

入学年度　平成29年度入学

学籍番号　26115162

氏名　　芳野 魁

論文要旨

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議論の整理や収束を行うことが困難である．Web 上での大規模合意形成を実現するために，本研究では過去に大規模意見集約システム COLLAGREE を開発した．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．また，他のシステムで議論の整理や収束を行うことが困難であった原因として議論の管理を行う者がいないことが挙げられるが，COLLAGREE ではファシリテーターによる適切な議論プロセスの進行を行っている．他にもインセンティブ機構による議論の活性化等により，大規模意見集約の実現を目指す．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困難である．

COLLAGREE で大規模な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だけ画面を見るようにして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したときである．以前の議論の内容から外れた発言がされた時，ファシリテーターが適切に発言することで，脱線や炎上を避けて議論を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事前に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システムは投稿支援システムと議論可視化システムの 2 つに大別できる．しかし，既存の支援システムではファシリテーターの負担を軽減するものがまだない．また，既存の COLLAGREE 以外での議論支援研究においても議論の活性化に焦点を当てたものはあるが，ファシリテーターの負担を軽減するものは筆者の知る限りない．同様に話題遷移検出でも本研究と似た手法を用いているものはあるものの議論データを対象にしたものや分散表現を使用した研究はなく，分散表現を話題に対して使用した研究では文の生成や分類等が目的となっており話題遷移を扱っているものは筆者の知る限りない．

近年，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る．

以上のような背景を踏まえて，本研究では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ことを目標とする．話題の変化の判定は，新しく投稿された発言と過去に投稿された発言との類似度を計算してどれか類似しているものがあるかどうかで判定する．発言の類似度は発言文中に現れる単語の類似度と見なすことができる．発言の中から発言をよく表す重要な単語を抜き出し，単語を分散表現に変換して類似度を内積計算によって類似度を求める．発言文から単語を選ぶ際には自動要約を用いる．発言文から重要な単語だけを取り出すことで類似度の計算における精度を高めることが可能となる．

具体的な提案手法は，既存の抽出的要約手法である okapiBM25 と LexRank を組み合わせて重要な文の中の重要な単語を抜き出し，選ばれた単語の類似度を計算するという手法である．提案手法と分散表現を用いない比較手法により，議論中

の話題変化判定の評価実験を行う．評価実験によって，提案手法を用いることで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できることを確認する．

目 次

図 目 次

表 目 次

第1章

序論

本章では， 本研究を行うに至った背景と目的について述べた後，本論文の構成について述べる．

1.1 研究の背景

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．そこで本研究では，過去に Web 上での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[?] を開発した．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システムは「(1) 投稿支援システム」と「(2) 議論可視化システム」の 2 つに大別できる．投稿支援システムはポイント機能やファシリテーションフレーズ簡易投稿機能のように，ユーザーが投稿をする際に何らかの補助やリアクションを行う．現行の機能では選択肢の提示に留まっており，作業量を減らすことには繋がりにくい．一方，議論可視化システムは議論ツリーやキーワード抽出のように，ユーザーにスレッドとは異なる議論の見方を提供する．現行の機能では議論を見やすくすることに重点が置かれており，議論の把握の助けにはなるが画面に向き合う時間を減らすことにはなりにくい．むしろ，作業量を増やすことになり得ることもある．従って，現行の支援機能で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.

1.2 研究の目的

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．第一段階では既存の抽出的要約手法である okapiBM25 と LexRank を組み合わせて発言文中から重要な単語を抜き出す．重要な単語のみを抜き出すことで第二段階での類似度計算の精度の上昇を目指す．第二段階では抜き出された単語を分散表現を用いて実数ベクトルに変換し，内積計算によって発言の内容の類似度を計算する．第三段階では第二段階で計算した発言内容の類似度と発言間の時間差や返信関係等の他の要素を組み合わせて総合類似度を計算する．時間差等の要素を考慮することで，基本的に少し前の発言に関連して行われることが多いという議論の特性を考慮できるようになる．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.

1.3 本論文の構成

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，3 章では話題変化判定システムの全体モデル説明を行い，4 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 6 章で本論文のまとめと考察を示す．

第2章

関連研究

2.1 序言

本章では，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，既存の議論支援研究や話題遷移を扱う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.

本章の構成を以下に示す．まず，2.2 節では，COLLAGREE の概要について簡単に述べる．2.3 節では，既存の議論支援に関する研究について述べる．2.4 節では話題に関する本研究の見方や既存の話題に関する研究について述べる．次に，2.5 節では本研究の重要な要素である重み付けについて述べる．2.6 節では本研究のもう 1 つの重要な要素である分散表現及び分散表現を使用した話題関連研究について述べる．最後に，2.7 節で本章のまとめを示す.

2.2 COLLAGREE

2.2.1 COLLAGREE 開発の背景

本研究で言う議論は最も近しいものとして加藤ら [?] の定義した”ある特定の問題を解決することを目的とし，２者以上の間で言語的・非言語的媒介を通して意思疎通をすること”を用いる．近年，Web 上での大規模な意見集約や議論を実現する研究に注目が集まっている．Web 上で話をする場として Twitter や Facebook などの SNS，ブログ，および掲示板といったプラットフォームがあるが，大規模な人数での意見集約をしたり，合意形成を行うのは困難である．なぜなら，議論の管理や整理を行う人物がおらず,「炎上」と呼ばれる議論の無秩序の状態がよく観測されているためである．そこで本研究では，過去に Web 上での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 を開発した [?]．図 2.1 に COLLAGREE のトップ画面を示す．

図 2.1: COLLAGREE のトップ画面

2.2.2 COLLAGREEの概要

COLLAGREE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，各ユーザーが自由なタイミングで意見を投稿，返信できる．しかし，2.2.1節で述べたように多様な価値観を持つ大規模な人数での，自由な議論では「炎上」が発生しやすい．従って，COLLAGREEでは，議論のマネジメント役として人間のファシリテーターが導入されている．COLLAGREEのような議論掲示板は基本的には１つの議論テーマに対して関連するテーマを扱った複数のスレッドから構成される．スレッドとはある特定の話題・論点に関する1つのまとまりを指す．例えば，図2.2のようにスレッドを立てたユーザーの発言が親意見となり，他のユーザーが子意見として親意見に返信し，子意見に対して孫意見が存在する場合もある．

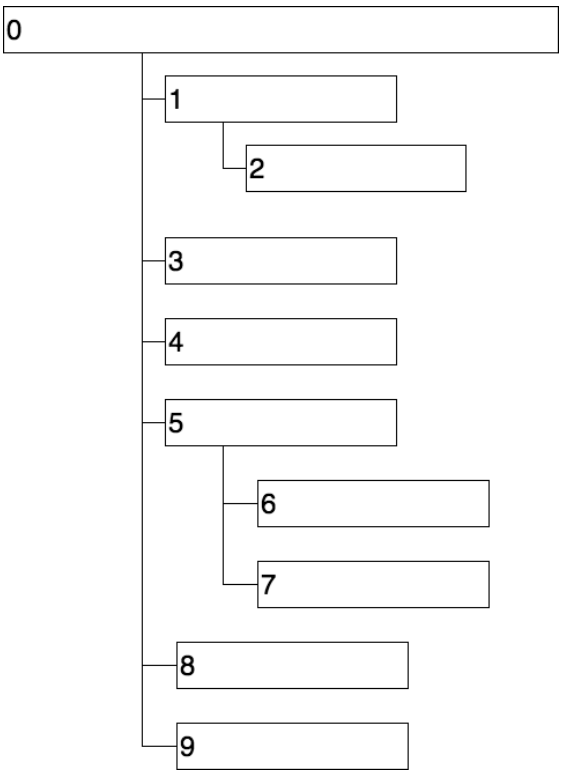


図 2.2: COLLAGREEのスレッドの例

しかし，スレッドや返信は必ずしも正しく使われるとは限らず，単なるチャットと同様に扱われてしまいスレッドが乱立してしまうこともある．図 2.3 に COLLAGREE での実際の議論画面を示す．

図 2.3 の上部の投稿フォームより，参加者は意見を自由に投稿することができる．投稿された意見は図 2.3 の下部に表示され，参加者は各意見に対して返信や賛同を行うことができる．返信を繰り返すことでスレッドが構成され，図 2.3 では 3 層のスレッドが構成されている．COLLAGREE 上での議論を支援する先行研究として，Ito ら [?] は COLLAGREE 上インセンティブとして参加者の活動の活発化や，重要投稿の把握の支援を目的に議論ポイントと呼ばれるシステムを導入した．システムでは参加者の投稿や投稿に対する返信や賛同に沿ってポイントを参加者に付与する．また，高橋ら [?] は質の高い投稿を促すために，議論ポイントを発展させて投稿された意見の質を評価する手法を提案した．投稿された意見中の名詞と以前に出現したキーワードとの一致度を計算し，一致していれば議論を収束させるものとして，一致していなければ議論を発散させるものとしてポイントを与える．他に議論の可視化支援として，仙石ら [?] は議論内容把握支援，合意形成支援，およびファシリテーション支援を目的として議論ツリーの実装を行った．ツリー図を COLLAGREE に応用したものを仙石らが呼んだもので，ツリー図とはファシリテーターが参加者に議論のポイントと各意見の関係を一致させるために，参加者の意見を文字や図形を用いて分かりやすく描く図の 1 つである．図 2.4 に議論ツリーの例を示す．議論ツリーは投稿をノードとし，エッジは基本的に投稿間の返信関係を基に自動的に作られる．そして，より正確な議論ツリーを作成するためにファシリテーターが手動で議論ツリー編集するというハイブリッド方式が導入されている．

図 2.3: COLLAGREE の議論画面

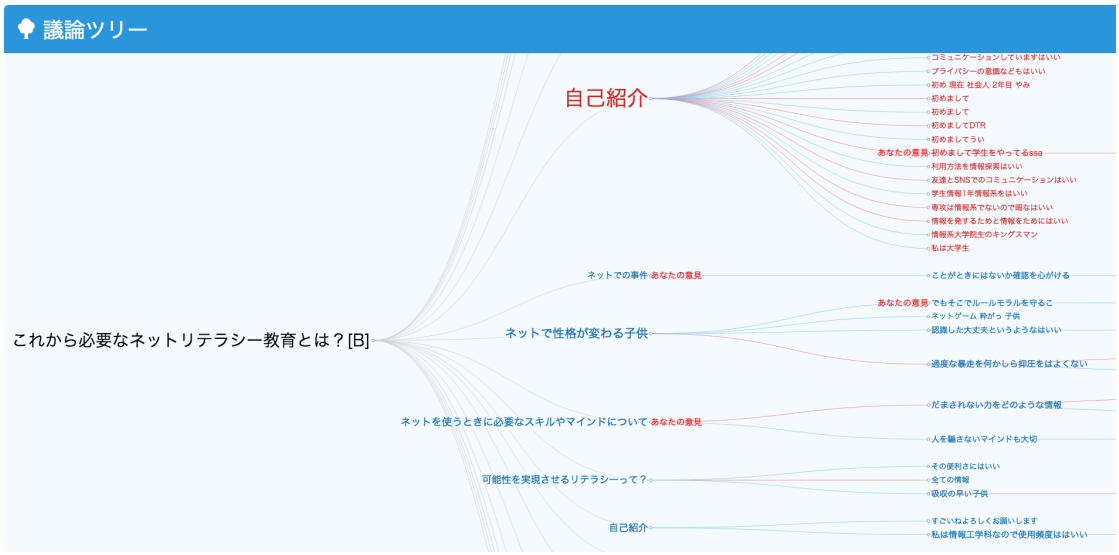


図 2.4: 議論ツリー

2.2.3 ファシリテーター

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．ファシリテーターは”促進者”を意味し，議論そのものには参加せず、あくまで中立的な立場から活動の支援を行うようにし，自分の意見を述べたり自ら意思決定をすることはない．ファシリテーターの基本的な役割として”議論の内容の整理”，”議論の脱線防止”，”意見の促し”等が挙げられる．伊藤ら [?] や伊美ら [?] によってファシリテーターが様々な論点に対して発言を促し議論を発散させることで十分な議論が行われ，ファシリテーターの存在や手腕が合意形成に強く影響を与えることや Web 上での議論においてファシリテーターが有用であることが示されている．

2.3　議論支援

2.2.1 節で述べたように近年大人数による議論には注目が集まっており，COLLAGREE 以外でも議論内容の把握支援を行い，議論の進行を支援するための研究が行われている．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．以上を踏まえて，本研究と小谷らの開発した議論支援システムとの関連性と相違点について述べる．両研究とも発言が議論に与える影響を扱っている点で関連している．しかし，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．

2.4　話題遷移検出

話題に関連する研究は 1960 年代に「会話分析」という学問から始まったとされる．会話分析は会話を始めとする相互行為の組織 (構造) を明らかにしようとする社会学の研究分野であり，相互行為の分析法を取り扱う．1960 年代に Harvey Sacks と Emanuel A.schegloff によって創案された [?]．話題は本来流れの一時点で簡単に区切ることのできるものではない．しかし，筒井 [?] は会話内容を分析する上で，連鎖組織及び言語形式との関連に基づいて分析を行っている．以下に筒井が立てた話題を区切る基準を 1 から 5 に示す．

1. それまで話題となっていた対象や事態とは異なる，新しい対象や事態への

言及

2. すでに言及された対象や事態の異なる側面への言及

3. すでに言及された対象や事態の異なる時間における様相への言及

4. すでに言及された対象や事態について，それと同種の対象や事態への言及

5. すでに言及された個別の対象や事態の一般化

本研究では上記の基準を参考に話題変化の判定の評価の基準を設けている．

話題の遷移に基づいた文章の分割は人間によるテキスト全体の内容の把握の容易化や複数のテキストに対する自動分類や検索の精度向上のために研究されている．以下では話題の遷移に関する関連研究について述べる．

2.4.1 テキストセグメンテーション

テキストセグメンテーションは複数のトピックが混合的に書かれている非構造的である文書をトピックに応じて分割する手法である．

TextTiling

Hearst ら [?] は複数の単語を連結した 2 つのブロックをテキストの初めから末尾まで動かしていきブロック間の類似度を計算する手法 (TextTiling，Hearst 法とも呼ばれる) を提案した．類似度はブロック間で共通して使われる単語数などで計算する．図 2.5 に手法の簡単な流れを示す．

計算された類似度をグラフにすると図 2.6 のようになる．縦軸はブロック間の類似度，横軸はブロック間の番号を表し，グラフ中の番号は段落番号，垂直線は選出された文の境界位置を表す．グラフは各ブロック間の類似度を繋いだものと，

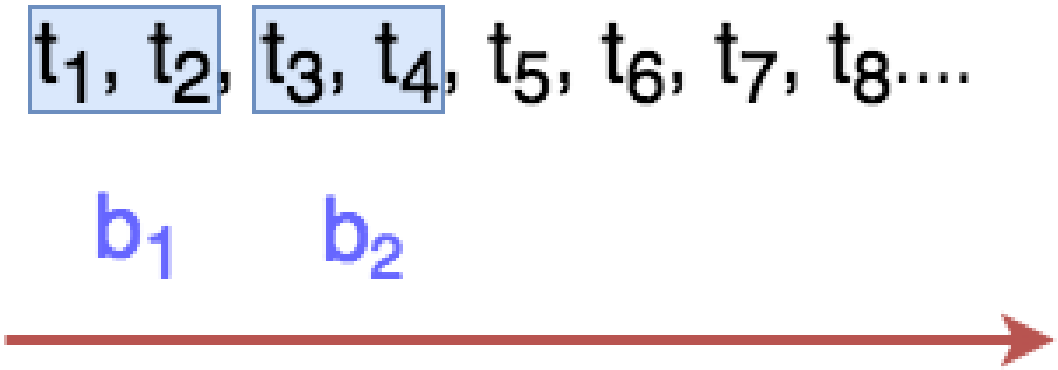


図 2.5: TextTiling の流れ

更に平滑化したものの 2 つが描かれている．グラフの谷のような場所は左右のブロック間の類似度が下がる場所を表している．類似度が下がることは使われる単語が変わったこと，すなわちトピックが変わったことを意味しており，谷の付近に境界を設けることで文書をトピックごとに区切ることを可能としている．

テキストセグメンテーションを使用した関連研究

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．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．

以上を踏まえて，本研究とテキストセグメンテーションを用いた研究との相違点について述べる．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，

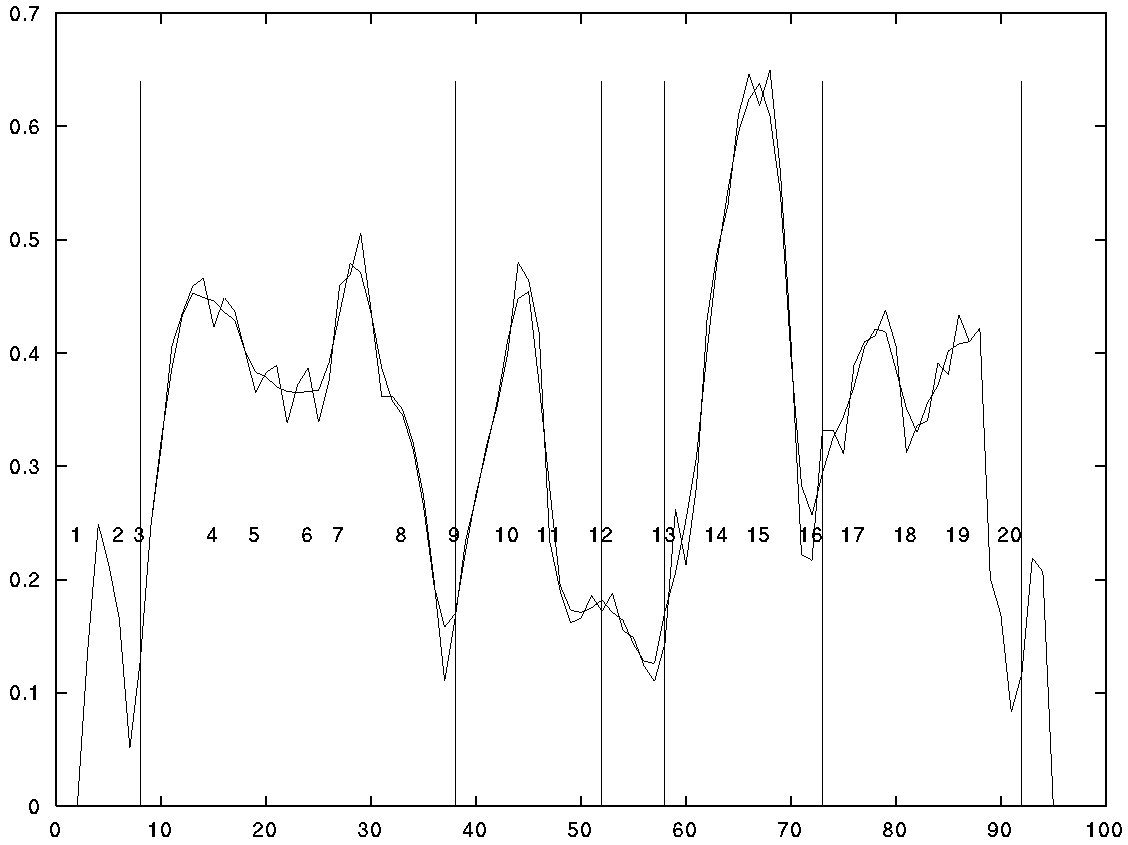


図 2.6: TextTiling のグラフ

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．

2.4.2　トピックモデル

トピックモデル [?] は確率モデルの一種にあたり，文章中の「単語が出現する確率」を推定している．単語が出現する確率をうまく推定することができれば，似たような単語が出てくる文章が把握できる．すなわち，トピックモデルとは「文書における単語の出現確率」を推定するモデルといえる．

ユニグラムモデル

ユニグラムモデルは，全ての文書の単語は，一つのトピックから生成されたものと仮定するモデルである．以下の図 2.7，図 2.8，図 2.9 は四角の箱が文章を表し，中身の色がトピックを表しているとしてユニグラムモデルについて述べる．図 2.7 で示されているユニグラムモデルは文章であるすべてのボックスが同じ色の状況を示している．すなわち，全ての文書の単語は 1 つのトピックから生成されたと仮定するモデルである．

図 2.7: ユニグラムモデル

混合ユニグラムモデル

図 2.8 で示されている，混合ユニグラムモデルは各ボックスの色が異なっている．

つまり，各文書に一つのトピックがあり，該当するトピックから文書の単語が生成されると仮定するモデルである．

図 2.8: 混合ユニグラムモデル

LDA

図 2.9 で示されている，LDA(Latent Dirichlet Allocation)[?] では，ボックスの中で色が異なっている．つまり，各文書は複数のトピックで構成されていて，各トピックの単語分布を合算した形で単語が生成されていると仮定を行うモデルである．

図 2.9: LDA

LDA では文書 d 中のトピック t の単語の割合 $p(t|d)$ とトピック t で単語 w が生成される確率 $p(w|t)$ を学習データとの誤差が小さくなるように学習していく．

トピックモデルを使用した関連研究

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．中村らが行ったトピック変化点を検出する手順を図 2.10 に示す．

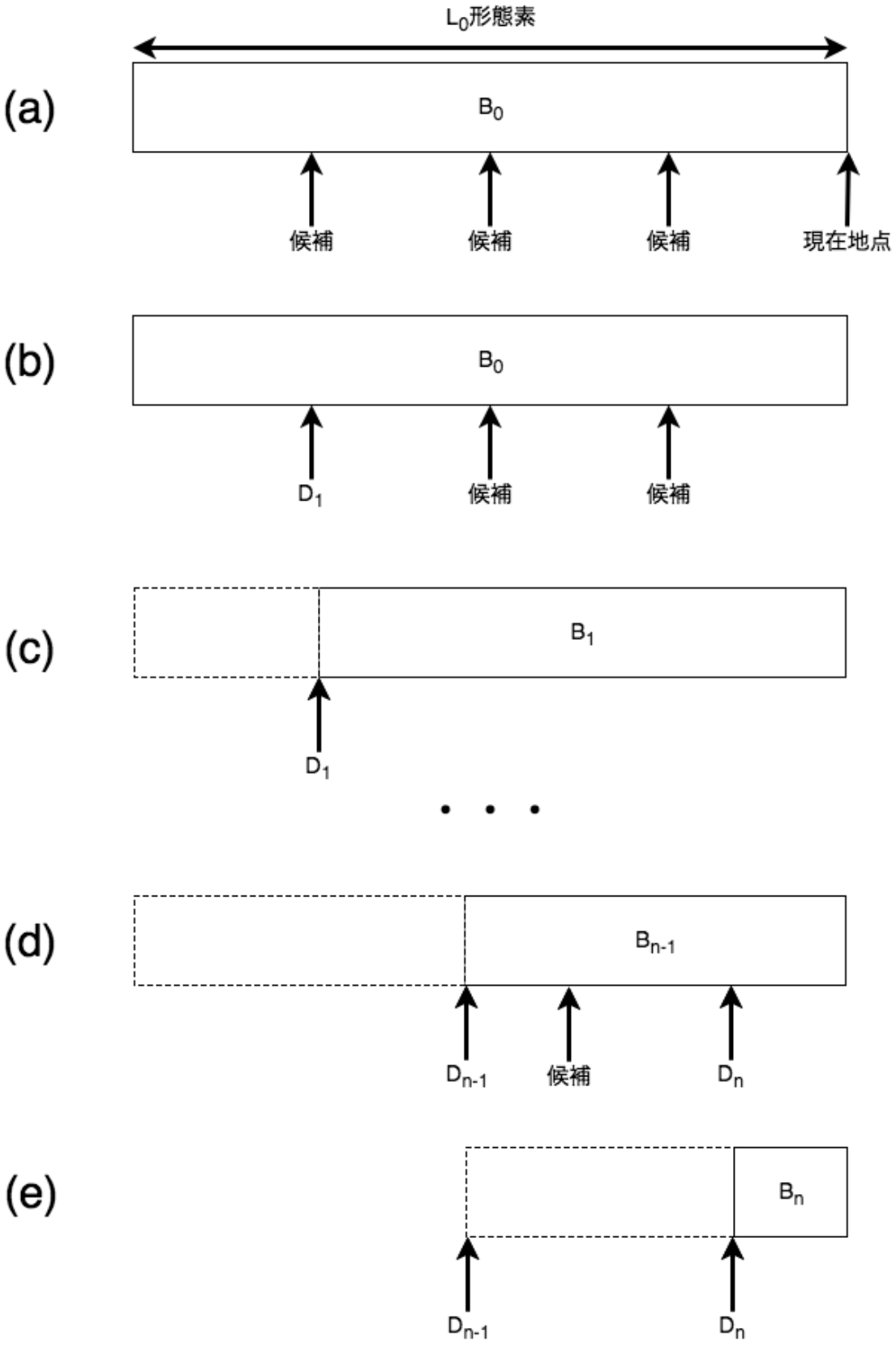


図 2.10: トピック変化点検出手順

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 とし ($L_0 > L; L =$ 文章長の下限)，B_0 から複数のトピック変化点候補箇所を抽出する．図 2.10(a) に該当する．

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．図 2.10(b) に該当する．

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．図 2.10(c) に該当する．

   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．図 2.10(d) に該当する．

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．図 2.10(e) に該当する．

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．

以上を踏まえて，本研究とトピックモデルを用いた研究との相違点について述べる．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．

2.5　重み付け

重み付けは情報検索を主とする分野で使われる方法で，蓄積された情報中の語の索引語，すなわち特定の情報の特徴を表し検索の手掛かりとなる語，としての重要度を数値的に表現し，それぞれの語の重要度に応じて重みを付け，集計して総合評価を出す手法である．出された総合評価はスコア等と呼ばれる．一般的に語の重要度は整数または実数値で与えられる．自動的な重み付けにおいては語の出現頻度情報や出現位置を利用して数値を与える．本研究で行う重み付けは下記の 2 種類の重み付けを基にしている．

2.5.1　Okapi BM25

Okapi BM25[?] は出現頻度情報を用いる重み付け手法の代表の 1 つである．Okapi BM25 では TF(Term Frequency 単語の出現頻度)[?]，IDF(Inverse Document Frequency 逆文書頻度)[?]，DL(Document Length 文書長) の 3 つを用いて重要度の計

算を行う．各用語について説明を行う．

TF

TF は単語の出現頻度を表し，文書中において出現頻度の高い単語は重要であるという考え方に基づく．ある単語 t_i の文書 D_j 中における出現頻度重み $tf_{i,j}$ は式 (2.1) のようにして求められる．

$$tf_{i,j} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{2.1}$$

ここで n_{ij} は文書 d_j における単語 t_i の出現回数，$\sum_k n_{kj}$ は文書 d_j におけるすべての単語の出現回数の和である．

IDF

IDF は逆文書頻度を表し，多くの文書において出現頻度の高い単語は重要ではないという考え方に基づく．IDF は多くの文書に出現する語，すなわち一般的な語の重要度を下げ，特定の文書にしか出現しない単語の重要度を上げる役割を果たす．ある単語 t_i の逆文書頻度重み idf_i は式 (2.2) のようにして求められる．

$$idf_i = \log \frac{|D|}{|\{d : d \ni t_i\}|} \tag{2.2}$$

ここで $|D|$ は総文書数，$|\{d : d \ni t_i\}|$ は単語 t_i を含む文書数である．

DL

DL は文書長を表し，ある単語の出現回数が同じ 2 つの文書について，総単語数の少ない文書と多い文書では，前者のほうがより価値があるという考え方に基づく．

ある文書 d_j の文書長重み ndl_j は式 (2.3) のようにして求められる．

$$ndl_j = \frac{dl_j}{ave(dl)} \tag{2.3}$$

ここで dl_j は文書 d_j の総単語数，$ave(dl)$ はすべての文書の平均 dl を表す．

上記の 3 つの重みを用いて Okapi BM25 は (2.4) のように単語 t_i の文書 D_j 中における統合重み cw_{ij} を求める．

$$cw_{ij} = \frac{tf_{i,j} \cdot idf_i \cdot (k_1 + 1)}{k_1 \cdot (1 - b + b \cdot ndl_j) + tf_{i,j}} \tag{2.4}$$

ここで定数 k_1 と b について説明する．２つの定数はどちらもチューニングの役割を果すもので k_1 は単語の出現頻度による影響を，b は文書の長さによる影響を調節する．

2.5.2　LexRank

Erkan ら [?] によって考案された LexRank は Google の PageRank[?] を使用した文章要約アルゴリズムで，文の類似度を計算して次の 2 つの基準に基いて文の重要度を計算する．

1. 多くの文に類似する文は重要な文である．

2. 重要な文に類似する文は重要な文である．

LexRank でいう類似度は簡単にいえば 2 文がどれだけ共通の単語を持つかということを表し，文を TF-IDF を用いてベクトル化して Cosine を求めることで類似度

としており，式 2.5 に沿ってベクトル x と y の Cosine が計算される．

$$\text{idf} - \text{modified} - \text{cosine}(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}} \tag{2.5}$$

文の間の Cosine 類似度をグラフとして可視化すると図 2.11 のようになる．各エッジは文の間の Cosine 類似度を表し，$dXsY$ は文書 X の Y 番目の文を示す．

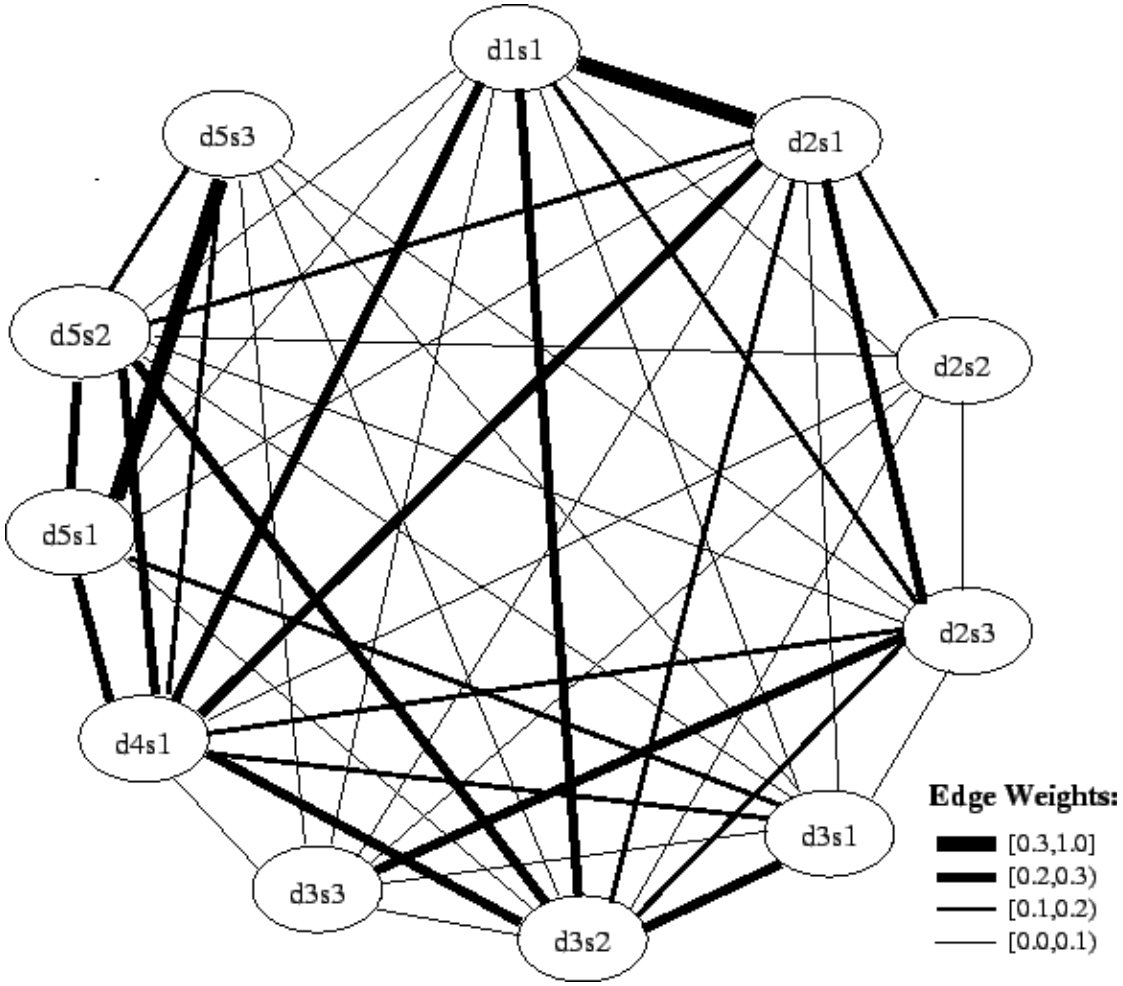


図 2.11: 類似度グラフの例

その後，Cosine 類似度が閾値を超えたかどうかを基に隣接行列が作成される．隣接行列はグラフを表現するために用いられる行列で，あるノード v と w の間にエッ

ジの有無が行列の (v, w) 成分に割り当てられる．隣接行列の各要素を類似している文の数で割り，確率行列に変換した後，**Algorithm1** に従って行列の固有ベクトル $\boldsymbol{p}$ が計算される．求められた固有ベクトルが LexRank スコアとなる．

Algorithm 1 べき乗法の計算アルゴリズム

```
1: Input : 確率的かつ既約かつ非周期的な行列 M
2: Input : 行列サイズ N, 誤差許容値 ε
3: Output: 固有ベクトル p
4: procedure powerMethod(M, N,ε)
5:     p_0 = (1/N)1;
6:     t = 0;
7:     repeat
8:         t = t + 1;
9:         p_t = M^T p_{t-1};
10:        δ = ||p_t − p_{t-1}||;
11:    until δ < ε
12:    return p_t;
```

LexRank スコアを計算する一連のアルゴリズムを **Algorithm2** に示す．

7∼14 行目では隣接行列が作成されており，15∼18 行目では LexRank が計算されている．

2.6　分散表現

自然言語処理では単語を低次元または高次元の実数ベクトルで表現する技術で Harris[?] 及び Firth[?] が提唱した”分布仮説”(”同じ文脈で出現する単語は類似した意味を持つ傾向があり，単語はその単語とともに出現する単語等によって特徴づけられる．”という考え方) に基づいている．

Algorithm 2 LexRank スコアの計算アルゴリズム

```
1: Input : n 個の文からなる配列 S, コサイン類似度の閾値 threshold
2: Output : 各文の LexRank スコアを格納した配列 L
3: Array  CosineMatrix[n][n];
4: Array  Degree[n];
5: Array  L[n];
6: procedure LexRank(S, t)
7:     for i ← 1 to n do
8:         for j ← 1 to n do
9:             CosineMatrix[i][j] = idf-modified-cosine(S[i], S[j]);
10:            if CosineMatrix[i][j] > threshold then
11:                CosineMatrix[i][j] = 1;
12:                Degree[i] + +;
13:            else
14:                CosineMatrix[i][j] = 0;
15:    for i ← 1 to n do
16:        for j ← 1 to n do
17:            CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
18:    L = PowerMethod(CosineMatrix, n, ε );
19:    return L
```

2.6.1　単語文脈行列

単語文脈行列は分散表現の最も基本的な形式で図 2.12 のような形の行列で表される．

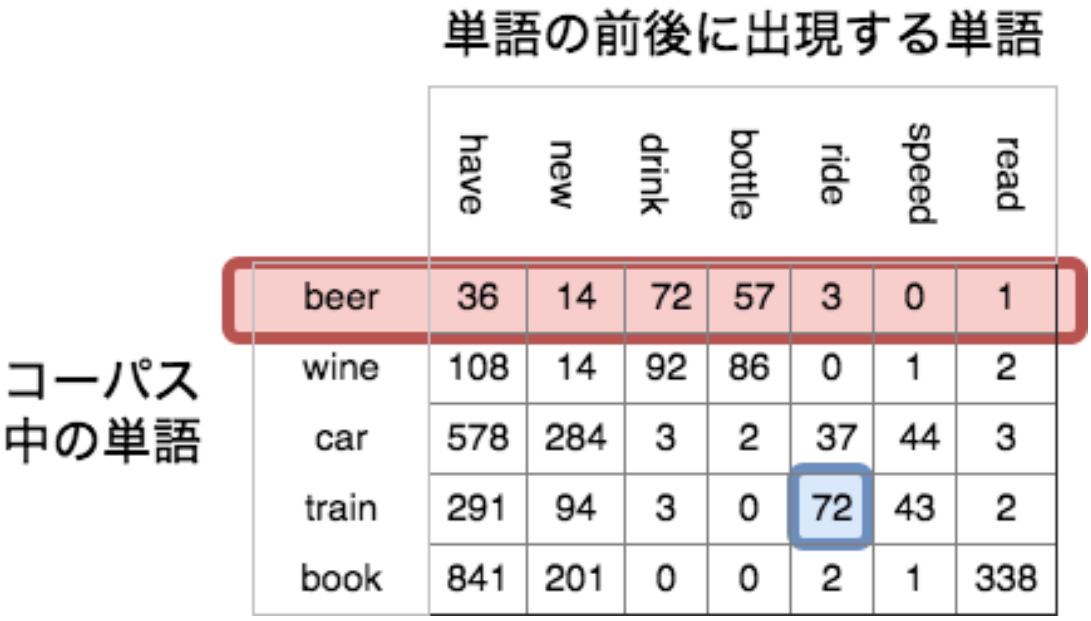


図 2.12:　単語文脈行列 (　https://www.slideshare.net/naoakiokazaki/20150530-jsai2015 の表を筆者修正)

行列中の各要素 M_{ij} は単語 i と文脈 j の共起頻度を表しており，例として青い四角は train と ride が 72 回共起したことを表している．各行 M_i は単語 i の意味ベクトルを表し，例として赤い四角は”beer”の単語ベクトルを表している．また，単語の類似度を式 2.6 のように単語の意味ベクトルのコサイン類似度で求めることができる．

$$cos\theta = \frac{M_i \cdot M_j}{|M_i||M_j|} \tag{2.6}$$

式 2.6 を図 2.12 の行列に適応させると beer と wine のコサイン類似度は約 0.941，beer と train のコサイン類似度は約 0.387 となり，train よりも wine の方が beer に

類似していることが分かる．

2.6.2 word2vec

2.6.1 節で説明した単語文脈行列から得られる単語ベクトルは単語の類似度を求めることは出来たが，他の数学的処理には対応していなかった．Mikolov ら [?] が開発した word2vec はニューラルネットワークを用いて分散表現の生成を行う手法で，文脈または単語を予測するようにニューラルネットワークで学習を行い，隠れ層をベクトルとする．具体的には word2vec で使用される学習方法には CBOW と skip-gram の 2 種類が存在する．図 2.13，図 2.14 はそれぞれ CBOW と skip-gram の構造を表す．

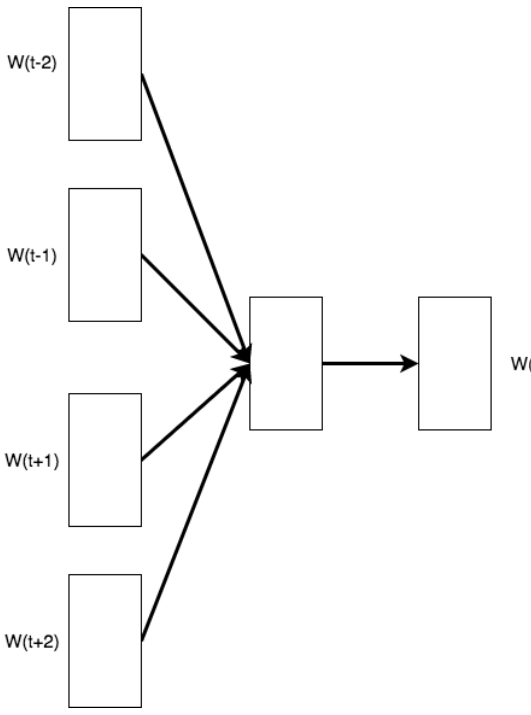


図 2.13: CBOW

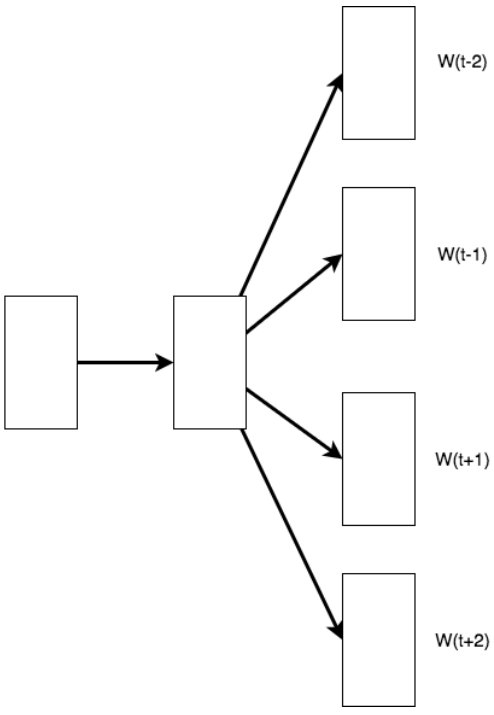


図 2.14: skip-gram

CBOW は周辺の単語 $W(t-2)\cdots W(t+2)$ を入力として，現在の単語 $W(t)$ を

予測することを目指して学習する．逆に skip-gram は現在の単語を入力として，周辺の単語を予測することを目指して学習する．どちらの手法でも中間層と単語の変換処理が行われており，学習によって得られる単語と中間層での数値が対応した行列が単語の分散表現モデルとなる．

word2vec が従来の手法と比べて大きく異なる点は　ニューラルネットによる学習で単語の類似度の計算に加えて，ベクトルの加減算が単語の意味の加減算に対応しているということである．加減算に対応できる意味を学習していることから異なる言語においても類似した単語の関係性が学習でき，機械翻訳において高い性能を示すことが Mikolov ら [?] が示している．また，従来の手法を用いた単語ベクトルよりも類推精度が高いことが Levy ら [?] によって示されている．例えば，$X = vector(\text{"}biggest\text{"}) - vector(\text{"}big\text{"}) + vector(\text{"}small\text{"})$ として計算されたベクトル X に最も類似したベクトルを探すことで biggest が big に類似しているのと同じ意味で small に類似している単語 smallest を見つけることができる．ただし，分散表現モデルが十分に訓練されていることが前提である．

2.6.3　fastText

fastText[?][?] は word2vec を発展させた手法でより大きな語彙や多くの稀な単語に対応することができ，学習の速度を上昇させることにも成功している．

fastText は skip-gram モデルを採用しており，学習の際に単語だけでなく部分語 (単語を構成する文字のまとまり) についても考慮する．以下に単語 w_t が与えられた時に予測した文脈単語 w_c の間のスコア関数を示す．

$$s(w_t, w_c) = \boldsymbol{u}_{w_t}^{\mathrm{T}} \boldsymbol{v}_{w_c} \tag{2.7}$$

$$s(w_t, w_c) = \sum_{g \in \mathcal{G}_w} \boldsymbol{z}_g^{\mathrm{T}} \boldsymbol{v}_{w_c} \tag{2.8}$$

式 2.7 は fastText 以前の手法でのスコア関数を表し，式 2.8 は fastText でのスコア関数を表す．$\boldsymbol{u}_{w_t}$，$\boldsymbol{v}_{w_c}$ はそれぞれ単語 w_t，w_c を実数ベクトルで表したもので，式 2.8 において $\mathcal{G}_w$，$\boldsymbol{z}_g$ はそれぞれ単語 w の n-gram の集合と n-gram g を実数ベクトルで表現したものを表す．式 2.7 では単語と文脈単語の間のスカラー積をスコアとしているが，式 2.8 では単語の n-gram と文脈単語の間のスカラー積の合計をスコアとしている．式 2.8 の手法を用いることで従来のモデルでは考慮されていなかった”活用形”を考慮できるようになった．例として，単語 go と goes と going は全て go の活用形であるが字面は異なるので従来のモデルでは異なる単語として学習されていたが，fastText では部分語である”go”を 3 つ全てで学習することで意味の近しい単語として学習することが可能となることが挙げられる．

本研究では分散表現の手法として fastText を使用している．

2.6.4 分散表現を使用した話題関連研究

分散表現は関連語を導出できることから分散表現を話題関連に用いる研究が行われている．中野ら [?] は分散表現を用いて雑談対話システムでのシステム側の応答生成を行っている．図 2.15 に話題展開システムの構造を示す．

システムではユーザ発話を形態素解析して検出された単語を単語分散表現による類似語検索から得られた結果と入れ替えることでシステム応答の生成を行っている．

また，Li ら [?] は分散表現を用いて Twitter のツイートをトピックカテゴリに分

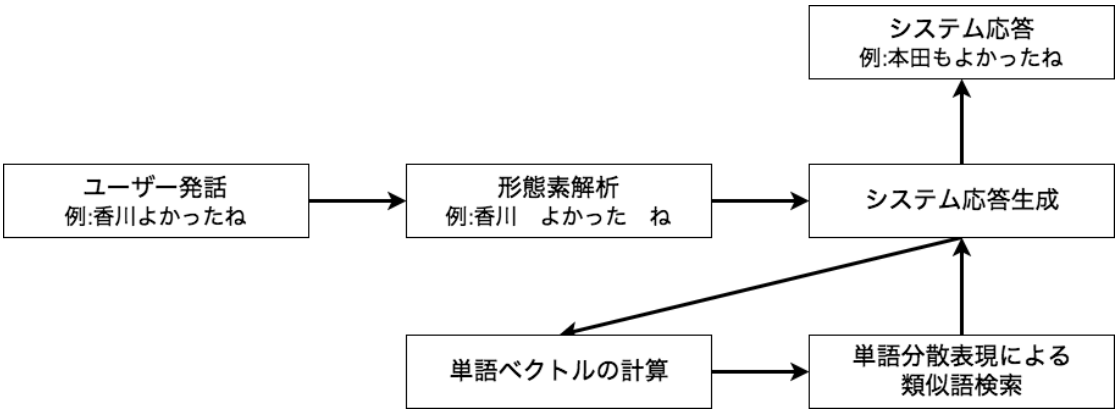


図 2.15: 話題展開システムの構造

類する分類器 TweetSift を提案している．図 2.15 にトピックカテゴリの予測のワークフローを示す．

図 2.15 の右のフローではツイートデータとスクレイピングで作成された知識ベースを用いて知識ベース特徴を生成している．図 2.15 の左のフローでは前処理を行ったツイートと作成済み分散表現モデルを用いて分散表現特徴を生成している． 2 つの特徴を用いて SMO(Sequential Minimal Optimization)[?] によってトピックが予測されている．Li らの研究で用いられる分散表現は学習の際に単語だけでなく，LDA を用いて予測した単語のトピックも使用されている．

以上を踏まえて，本研究と分散表現を用いた研究との相違点について述べる．本研究は Web 上での議論を対象としており，対話や SNS を対象としていない点で異なる．また，中野らの研究とは文の生成を行わない点でも異なり，Li らの研究とは分散表現の学習の際にトピックを用いていない点でも異なる．

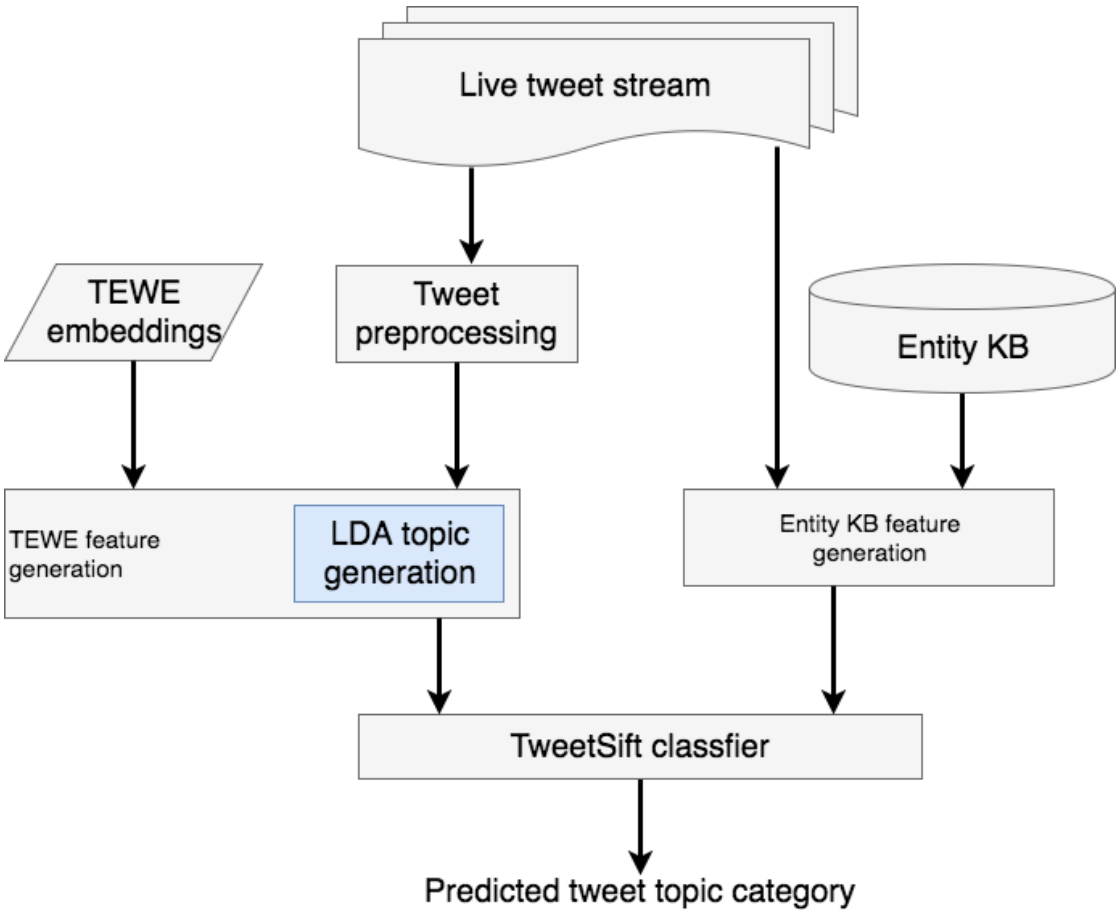


図 2.16: TweetSift によるトピック予測のワークフロー

2.7 結言

本章では，COLLAGREE の開発の背景及び概要と COLLAGREE 上で既存の議論支援について説明し，本研究で支援することを目的とするファシリテーターについても説明した．次に，既存の議論支援研究を紹介して本研究との相違点を説明した．また，話題に関連する研究の概要と話題遷移を研究している点で本研究と関連している研究についても述べ，本研究との相違点を説明した．そして，本研究での重要な要素である重み付けと分散表現についてと，分散表現を使用した関連研究について説明し，本研究との相違点を説明した．

第3章

話題変化判定システム

3.1 序言

本章では話題変化判定システムの概要について説明する.

以下に本章の構成を示す．まず 3.2 節でシステム全体の動作の流れを示し, アルゴリズムについても説明を行う．次に 3.3 節ではシステムの詳細を説明すると共にシステムで扱う発言データの形式や発言間の類似度計算について述べる．最後に，3.4 節で本章のまとめを示す.

3.2 システムの動作の流れ

擬似コードを **Algorithm3** に示し，図示したものを図 3.1 に示す.

Algorithm3 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 3 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
```

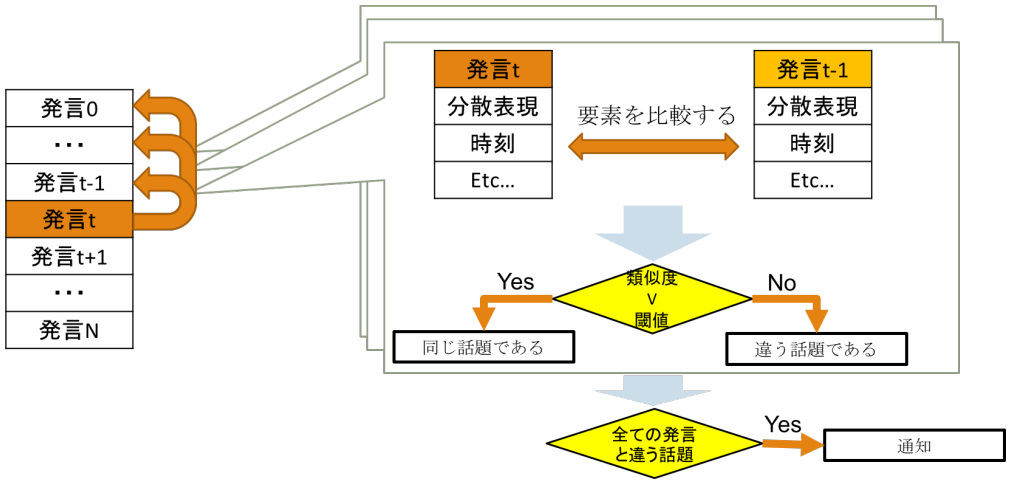


図 3.1: システムの流れ

3.3　システム詳細

3.3.1　発言データ

本システムで扱う”発言データ”は単なる文字列ではなく，タイムスタンプ等の他のデータを持つリスト形式のデータである．表 3.1 にデータの一部を示す．また，以下で本研究で使用する発言データの要素について説明する．

① id

発言データを識別するための番号で，全て整数値で表される．

② title

スレッドのタイトル名を表す文字列で，発言がスレッドの先頭でない限りは NULL となる．

③ body

発言の内容を表す文字列．

④ parent-id

発言の返信先，すなわち親発言の id を表す番号で，親発言がある場合は親発言の id と同じ番号になり，ない場合は NULL となる．

⑤ crreated-at

発言が投稿された時間を示すタイムスタンプ．

id	title	body	parent-id	created-at	(以降は省略する)
18	オンラインでの議論に関する実験	参加者の皆様が集まるまでお待ち下さい。	NULL	2017/03/14 16:28:00	(以降は省略する)

表 3.1: 発言データ

3.3.2 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 4 章で述べる．

② 発言内容の類似度計算

①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 4 章で詳しく述べる．

③ 総合類似度計算

上記の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．

時間差評価値

発言 new と以前の発言 old 間の時間差を式 3.1 に基いて正規化された評価値として求める．

$$tValue = 1 - \frac{epoch(new.created) - epoch(old.created)}{maxTime} \tag{3.1}$$

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．

返信距離

発言 new と以前の発言 old 間の返信距離を Algorithm4 に基いて再帰的に求める．

Algorithm 4 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．

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 3.2 のように計算される．

$$tSim = tValue * tWeight + sim * (1 - tWeight) \tag{3.2}$$

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．

3.4 結言

本章では話題変化判定システムの動作の流れやアルゴリズムについて説明し，扱うデータの形式や内容を示した．4 章にて説明する発言内容の類似度の計算手法の

詳細を除いて，発言間の類似度計算の手法についても説明した．また，時間的に近しい議論ほど類似度が上昇するように時間差評価値と発言内容の類似度の２つを用いて総合類似度計算することを示した．

第4章

発言内容の類似度計算

4.1 序言

本章では発言内容，すなわち文字列の意味的類似度を計算する手法を提案する．以下に本章の構成を示す．まず，4.2 節では類似度の精度を上昇させるために行う前処理について説明する．4.3 節では発言内容の類似度計算の手法について述べる．4.4 節では本章のまとめを示す．

4.2 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 2.5 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．

4.2.1 MeCab

MeCab(めかぶ)[?] は京都大学情報学研究科-日本電信電話株式会社コミュニケーション科学基礎研究所共同研究ユニットプロジェクトを通じて開発されたオープンソース形態素解析エンジンである．MeCab に対して「MeCab はオープンソース形態素解析エンジンである。」と入力した際の結果を図 4.1 に示す．

```
Mecabはオープンソース形態素解析エンジンである。
Mecab    名詞,固有名詞,組織,*,*,*,*
は       助詞,係助詞,*,*,*,*,は,ハ,ワ
オープン         名詞,サ変接続,*,*,*,*,オープン,オープン,オープン
ソース   名詞,一般,*,*,*,*,ソース,ソース,ソース
形態素   名詞,一般,*,*,*,*,形態素,ケイタイソ,ケイタイソ
解析     名詞,サ変接続,*,*,*,*,解析,カイセキ,カイセキ
エンジン         名詞,一般,*,*,*,*,エンジン,エンジン,エンジン
で       助動詞,*,*,*,特殊・ダ,連用形,だ,デ,デ
ある     助動詞,*,*,*,五段・ラ行アル,基本形,ある,アル,アル
。       記号,句点,*,*,*,*,。,。,。
EOS
```

図 4.1: 解析結果

出力フォーマットは次の形式となっている．

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

「読み」と「発音」は図 4.1 の”MeCab”のように不明であるものには付与されない．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 4.2.2 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．

4.2.2 重み付け

提案手法では 2.5 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．アルゴリズムを **Algorithm5** に示す．

Algorithm 5 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; ▷ 以前に重み付けを行った最大 n 個前までの文章のリスト
4: procedure calcCombinedWeight(reamrk)
5:     bm25Weight = calcBM25Weight(remark)          ▷ 単語と重みの連想配列
6:     for Each sent ∈ remark do      ▷ remark を句点，改行コードで分割する
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
```

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．

4.3 類似度計算

4.3.1 単語抽出

4.2 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．

4.3.2 分散表現による類似度計算

4.3.1 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，2.6 の図 2.6 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．

4.4 結言

本章では発言内容の類似度を計算する手法について説明した．文章を単語に分割する手法と使用したツールと単語を除外する前処理についても述べた．また，okapiBM25 と LexRank を組み合わせた発言中の単語の重み付け手法，及び重み付けによって抽出した単語集合の類似度計算手法についても説明した．

第5章

評価実験

5.1 序言

本章では，COLLAGREE で行われた議論のデータを対象にした提案手法の評価実験について述べる．以下に本章の構成を述べる．まず，5.2 節では実験に用いたデータについて説明する．5.3 節では実験設定について述べ，5.4 節では評価実験の結果を示す．5.5 節では実験結果に対する考察を行い，最後に 5.6 節で本章のまとめを示す．

5.2 対象データ

5.2.1 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．データの概要を以下に示す．

【実験概要】

グループ人数 ：2~3名

議論時間 ：90分前後

議論テーマ ：外国人観光客向けの日本旅行プランの決定

議論テーマ説明文 ：みなさまに、外国人観光客向けの日本旅行プランを立てていただきます。 想定される旅行者の条件は以下の通りです。

- 英語は話せるが、日本語は話せない
- 初めての日本旅行である
- 日程は6泊7日
- ホテルは自分たちで手配できる
- 旅行のために貯金したので、金銭的には余裕があり、国内をいろいろとまわることが可能である
- 来日、帰国の際の空港は、どこでもかまわない
- ２つのプランを比較したいと考えている（プランは２つ用意してください）

ファシリテータ ：あり

5.2.2 評価データ

5.2.1 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．

① それまで話題となっていた対象や事態とは異なる，新しい対象や事態への言及する発言

話されている内容が，以前と全く異なる対象や事態へと移行する位置でデータを区切る．

例 1:

(今までの話題:パック旅行はなぜ安いのかについて)

- A:ホテルが宿泊費の一部を出しているから安いのかな？
- B:おそらく。
- A:なるほど。
- B:沖縄行きも安いね。(今まで沖縄の話はされておらず，この後“ 沖縄行きのパック旅行 ”に話題が変わる (かもしれない))

例 2:

(今までの話題:外国人のツアー旅行の行き先について)

- A:他は寄らなくてもよいですか？(新しい行き先が出るように仕向けている)

② 既に言及された対象や事態の異なる側面への言及する発言

既に話題として取り上げられることについて，以前とは異なる側面から言及がなされる位置で区切る．

例 3:

(今までの話題:外国人のツアー旅行の行き先について)

- A:広島、長崎はどう？
- B:外国人観光客とか広島、長崎で見かけた覚えないな。
- A:ツアーに英語を話せるスタッフとか付けられるかな？(“ツアー旅行のスタッフ”に話題が変わる (かもしれない))

③議論のフェーズを移行させる (かもしれない) 発言

議論のフェーズを今までから移行させる (と思われる) 発言の位置で区切る．

例 4:

(今までの話題:外国人のツアー旅行の行き先について)

- A:八坂神社や清水寺など有名どころがたくさんありますし、魅力的だと思います
- B:京都周辺ツアー清水寺、金閣寺、銀閣寺、伏見稲荷大社、嵐山、など日本の建物や食べ物など広島長崎ツアー広島、長崎の戦争の地を見る事と、それぞれの場所で食べ物建造物を見るツアー (地名を挙げる段階から，各地点を結ぶツアープランへの作成段階に話題が変わる (かもしれない)．)

例 5:

(今までの話題:外国人のツアー旅行のプランについて)

- A:京都周辺ツアー清水寺、清水焼体験、抹茶・和菓子など体験、きもの体験、金閣寺、嵐山、伏見稲荷大社その中で乗れそうなら屋形船などはどうでしょうか?
- B:屋形船、風情があって良いと思います。
- (途中省略)
- C: まとめると、・京都周辺ツアー京都周辺（八坂神社、清水寺、金閣寺、銀閣寺、伏見稲荷大社、嵐山、有馬温泉)、おいしい料理（豆腐など)、温泉、6 泊 7 日ツアー

  ・広島長崎ツアー広島（3 日）: 広島の原爆ドーム、平和記念公園、厳島神社、もみじまんじゅう、牡蠣、広島の筆（メイクや書道なので使用する)、お好み焼き、呉の戦艦、アナゴ（移動 1 日）長崎（3 日）: ハウステンボス、グラバー園、眼鏡橋、大浦天主堂、軍艦島、長崎ちゃんぽん、佐世保バーガー

  この 2 プランで問題ないでしょうか？(初めて，２つのツアーの内容をまとめ，議論の収束に近づけた．)

また，ファシリテーターによる議論をコントロールするような発言も含む.

例 6:

- F:もし現在の旅先候補でよろしければ、具体的なプランづくりに移行したいと思います。よろしいでしょうか？
- F:残り 20 分を切りました。皆様、いかがでしょうか？

以上の基準に沿ってタグを付けてもらい，”1”のタグが過半数より多く付けられた発言を正解値=1，他を正解値=0 とした．

5.3 実験設定

5.3.1 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．

5.3.2 比較手法

① 常時通知

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．

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 5.1: パラメーターの設定

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．**Algorithm5** の 5 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．

5.3.3 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 $TP(True\ Positive)$ または $hits$(的中数)，予測値=0 かつ正解値=1 であるものの個数を $FP(False\ Negative)$ または

$misses$(見逃し数)，予測値=1 かつ正解値=0 であるものの個数を $FP(False\ Positive)$ または $falseAlarms$(誤警報数) 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．そして，式 5.1，式 5.2 及び式 5.3 に従って適合率，再現率，F 値を計算する．

$$Precision = \frac{hits}{hits + falseAlarms} \tag{5.1}$$

$$Recall = \frac{hits}{hits + misses} \tag{5.2}$$

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.3}$$

３つの値はどれも値が高いほど判定精度が高いことを示す．

5.4 実験結果

実験結果を表 5.2 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 5.2: 実験結果

また，各手法の TP,TN,FP,FN の割合の平均，及び TP と FP の平均割合の和である P-SUM と TN と FN の平均割合の和である N-SUM を表 5.3 に示す．

手法	平均割合					
	TP	TN	FP	FN	P-SUM	N-SUM
比較手法 1	0.251993	0	0.748006	0	1.0	0
比較手法 2	0.0239234	0.730462	0.0079744	0.2280701	0.0414673	0.9585326
提案手法	0.1419457	0.5980861	0.1499202	0.1100478	0.2918660	0.7081339

表 5.3: 実験結果 2

5.5 考察

実験結果から次のことが言える．

考察 1 提案手法はバランス良く判定をすることができる

考察 2 提案手法の精度は重み付けに依存する

考察① 提案手法はバランス良く判定をすることができる

表 5.2 が示すように，提案手法は比較手法よりも高い F 値を出している．原因を究明するために，他 2 つの比較手法の問題点を考える．比較手法 1 は常に話題が変化したと判定するため見逃しが無く再現率が高いが，対価として何も除外しないので適合率は低くなってしまう．極端さが F 値の低下に繋がったと言える．
一方，比較手法 2 では TF-IDF による発言ベクトルを用いて発言内容の類似度を計算しているが，TF-IDF では文字の出現頻度のみを使用していることと全ての単語を発言ベクトルに含んでいることから新しく投稿された発言の内容文と過去の発言の内容文の両方に同じ単語が含まれている程，類似度が大きくなる．すなわち，過去の発言に全く登場していない新しい単語を多く含んだ内容文を持つ発言

でない限り，過去の発言と同じ話題であると判定されやすい．結果的によほど顕著なものでない限り，話題が変化したと判定しなくなるため適合率は高くなるが，対価として見逃しが増え再現率が低くなる．事実，表 5.3 が示すように比較手法 2 では比較手法 1 とは逆に発言の殆どが話題の変化を起こさないものとして判定されている．図 5.1 に比較手法 2 で発言の繋がりを図示したものの拡大図を示す．

図 5.1 で示されるように殆どの発言がかなりの数の発言と同じ話題であると判定

図 5.1: 比較手法 2 による繋がりグラフ-拡大図

されていることが確認できる．

以上の２つの比較手法の問題点に対する考察を踏まえて提案手法が比較手法と異なる点を考える．異なる点として，上位の単語のみを類似度計算に用いている点と分散表現を用いている点が挙げられる．比較手法2と違って，上位の単語のみが用いられることで同じ単語が含まれているだけでは類似度が上がるとは限らなくなり，結果話題が変化したと判定する回数が増え比較手法2よりも高い再現率を示したと考えられる．図5.2に提案手法で発言の繋がりを図示したものを示す．

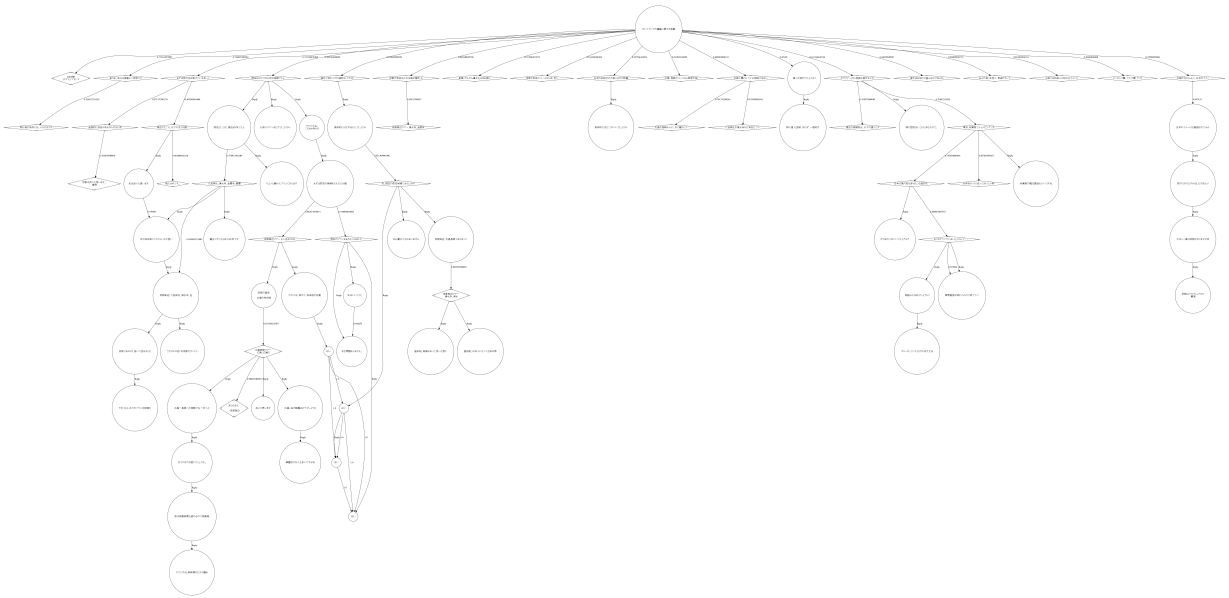


図 5.2: 提案手法による繋がりグラフ

図5.1とは違い，1つの発言に対して同じ話題であると判断された発言の数は少なく，比較手法2に比べ類似度が抑えめになっていることが伺える．また，分散表現を用いることで字面の異なる単語でも類似性を示すことができるので，再現率と共に適合率を上昇させることに繋がったと考えられ，故にバランス良く判定をすることができたと考えられる．

考察② 提案手法の精度は重み付けに依存する

表 5.4，table:MissesRemark にそれぞれ $falseAlarms$，$misses$ となった発言の例を示す． まず，表 5.4 の上段にあるような，議論の最初の方において意見を促

title	body
まずは旅先候補を	まずは旅先候補をどんどんお願いします。
京都周辺ツアー	八坂神社、清水寺、金閣寺、銀閣寺、伏見稲荷大社、嵐山、有馬温泉 今あがっているのは、このぐらいでしょうか？

表 5.4: falseAlarms 発言データ

title	body
NULL	やはり東京は人気ですよね、アキバ行きたいって外国のかたが多いイメージです
NULL	1 日目 奈良 東大寺 鹿がいる公園 2 日目 京都 金閣寺 銀閣寺 清水寺 3 日目 大阪 万博記念公園 たこやき食べ 4 日目 5 日目 長崎 世界遺産巡り 6 日目 長崎 教会群巡り 原爆資料館 7 日目 長崎 食めぐり

表 5.5: misses 発言データ

すようなファシリテーターが行うような発言が falseAlarms として誤判定されることが多かった．原因として過去に投稿された発言が少なく他の発言との差が顕著になってしまったと考えられる．また，まとめを行う発言に弱いことが示された．

5.6 結言

本章では本研究で提案する話題変化の判定手法が有用であることを実験により確認した．COLLAGREE にて行われた議論データに対して基準を満たすと思われるものにタグを付けてもらい評価データとした．評価実験では発言の内容に関係

なく常に通知する手法と分散表現の代わりに TF-IDF を用いて発言をベクトル化する手法を比較手法として用いた.

実験の結果，提案手法は適合率と再現率の両方でバランス良く高い結果を示すことが分かり，比較手法よりも良い結果を出すことがわかった.

第6章

結論

6.1　序言

本章では本論文のまとめを行う．

以下に本章の構成を示す．まず，関連研究の調査，提案手法の実装，および評価実験等，本研究を通して得られた知見を 6.2 節で述べる．次に今後の課題と展望を 6.3 節で述べる．最後に，6.4 にて本研究のまとめを示す．

6.2　得られた知見

本研究を通して，次の知見が得られた．

知見１　分散表現の類似度を発言の話題変化の判定に使うことができる

知見２　リアルタイムな議論での動作を想定した話題の変化の判定システムを提案した

知見1 分散表現の類似度を発言の話題変化の判定に使うことができる

本研究では，分散表現を用いて２つの発言間の類似度を計算する手法を提案した．実験の対象となる議論と類似したデータを事前学習してから実際に適用を行う機械学習的手法とは異なり，汎用的なデータで分散表現を作り時間差や返信関係を組み合わせた手法を使用して実際の議論に適用させた点が，本研究の特色として挙げられる．

知見２ リアルタイムな議論での動作を想定した話題の変化の判定システムを提案した

本研究では，リアルタイムな議論での動作を想定して

6.3　今後の課題と展望

6.4　本研究のまとめ

謝辞

本論文を作成するにあたって，数多くの方々の御支援，御協力を頂きました。ここに，その方々に心から感謝の気持ちを申し上げます．

指導教官である名古屋工業大学大学院産業戦略工学専攻 伊藤孝行教授 に感謝致します．伊藤教授には，日々の研究活動やプレゼンテーションに対する御指導だけでなく，普段の生活に至るまで幅広い御指導を頂き，企業との共同研究という貴重な体験をさせて頂きました。また，学部生にもかかわらず 3 か月間という長い間の研究留学をさせて頂き，海外の研究生の研究に対する姿勢や考え方に関して多くを学べ，自身の研究に対する価値観を変える大変大きな経験をさせて頂きました．さらに，伊藤教授の熱心な御指導により，国際学会を含めた様々な場で発表させて頂く機会を得ることができ，自身の成長につながりました．心より感謝申し上げます．

留学中の指導教官であるシドニー工科大学 Quantum Computation & Intelligent Systems の Ivor Tsang 准教授 に感謝致します．Ivor 准教授には研究活動だけでなく，将来の進路や進学に至るまで幅広い内容に対して熱心な御指導を頂きました．Ivor 准教授から頂いた様々な助言が，私の研究成果につながりました．心より感謝申し上げます．

名古屋工業大学 Rafik Hadfi 特任助教 に感謝致します．Rafik 特任助教には，研

究活動やプレゼンテーションに対する御指導を頂きました．また，英語での発表や論文執筆に対する御指導を頂き，自身の成長につながりました．さらに，将来の進路に関する相談にも親身に乗って頂き，多くの心強い助言を頂きました．ここに感謝の意を表します．

香港科技大学大学院 Department of Computer Science and Engineering の Xingjian Shi 氏に感謝致します．Xingjian 氏には，この論文を書くことになったきっかけを頂けただけでなく，見ず知らずの学部生相手に 40 通近くにも及ぶメールでのやり取りにおいて既存手法に関する詳細な説明を頂き，ソースコードを提供いただいた上に，提案手法に関する助言も頂きました．ここに感謝の意を表します．

シドニー工科大学の Han Bo 氏，Donna Xu 氏には，留学期間中に何度も機械学習に関する相談に乗って頂き，研究に関する助言を頂いただけでなく，論文執筆，学会・ジャーナルへの投稿，さらには将来の進路に至るまで幅広い助言を頂きました．ここに感謝の意を表します．

共同研究先の株式会社ウェザーサービス様には，評価実験のデータ収集に御協力頂いただけでなく，気象学の視点から本研究で提案したモデルや評価実験について参考になる助言を頂きました．ここに感謝の意を表します．

共同研究先の株式会社 NEC ソリューションイノベータ 加藤憲昭氏 には，研究活動に対する御支援をして頂いた上に，本研究の今後の展望について参考になるご意見を頂きました．ここに感謝の意を表します．

名古屋工業大学伊藤孝行研究室の秘書である 杉山順子氏 には，研究室での事務業務など，学生たちがよりよい環境で研究を行えるための御支援を頂きました．

ここに感謝の意を表します．

名古屋工業大学工学部情報工学専攻 伊藤孝行研究室の先輩である 徳田渉先輩には，研究活動に対する御指導を頂き，研究に関して幾度となく相談に乗って頂きました．また，御自身の研究が忙しいにもかかわらず，留学中に論文執筆の指導や研究の協力をしていただき，大変支えられました．ここに感謝の意を表します．

伊藤孝行研究室の卒業生である 佐藤元紀先輩 には，深層学習に関してわからないことがあった際に，幾度となく質問に答えていただきました．また，卒業されているにもかかわらず，深層学習に関する輪講会や勉強会に同伴して頂き，御指導を頂きました．ここに感謝の意を表します．

名古屋工業大学工学部情報工学専攻 伊藤孝行研究室の先輩である 早川浩平先輩には，研究活動に対する御指導を頂いた上に，研究室でより有意義に研究ができるよう取り計い頂きました．早川先輩の御指導，御力添えなくして順当な研究活動は行えなかったと思います．ここに感謝の意を表します．

名古屋工業大学工学部情報工学専攻 伊藤孝行研究室の先輩である 森顕之先輩には，研究活動やプレゼンテーションに対する熱心な御指導を頂き，自身の成長につながりました．また，先輩の研究に対する姿勢や言動から，論理的に話すことの大切さを学びました．ここに感謝の意を表します．

名古屋工業大学工学部情報工学科 伊藤孝行研究室の高橋一将君，仙石晃久君，Gu Wen 君，石田健太君，稲本琢磨君には同じ研究室の仲間として何度も助けられました．そしてこの伊藤研究室での有意義な時間を共に過ごすことができました．ここに感謝の意を表します．

また，友人の皆さんには貴重な時間と数々のご意見を頂きました．皆さんと過ごした時間はこれからも自身の励みとなると思います.

最後に自分の日々の生活を支えて頂いた家族に心より深く感謝いたします.

伊藤孝行研究室にて

2016 年 春

林政行

付 録 A

発表 (予定) 論文一覧

A.1 発表 (予定) 論文一覧

1. 芳野魁, 伊藤孝行 “分散表現を用いた話題変化判定”，情報処理学会第 80 回全国大会 (IPSJ)，2017.

付 録B

第2回市民共創知研究会

投稿論文

第 2 回市民共創知研究会に投稿した論文を示す．本論文は平成 29 年 7 月 1 日に発表された．

分散表現を用いた話題変化判定

A Topic Change Judgment Method based on Distributed Representation

芳野魁 [1] 伊藤孝行 [1]

Kai Yoshino[1] and Takayuki Ito[1]

[1] 名古屋工業大学情報工学科

[1] Nagoya Institute of Technology, Department of Computer Science

Abstract: As discussions on the Web become bigger, it is expected that discussion of multiple people will be necessary and large scale, and the burden on facilitators will increase accordingly. Therefore, in this research we aim to reduce the burden on facilitators by detecting variance of topics under discussion using distributed representation as one of the burden reductions.

1. はじめに

近年, Web 上での大規模な議論活動が活発になり，大規模な人数での議論が期待されている．大規模な議論では意見を共有することは可能であるが，議論を整理させることや収束させることは難しい．以上から大規模意見集約システム COLLAGREE が開発された[1]. 本システムでは Web 上で適切に大規模な議論を行うことができるように議論をマネジメントするファシリテーターを導入した．

過去の実験ではファシリテーターの存在が議論の集約に大きな役割を果たしていることが認識されており，大規模な議論のためにファシリテータは必要である[2][3]．しかし，議論の規模に伴って議論時間が長くなる傾向があり，同時にファシリテーターは常に議論の動向を見続ける必要がある．故に，議論の規模が大きくなればなるほどファシリテーターは長時間かつ大規模な議論の動向の監視によって大きな負担がかかる．大規模な議論が増加する傾向を踏まえるとファシリテーターにかかる負担を軽減する支援が必要となることは明白である．

また，近年自然言語処理の分野において分散表現が多くの研究で使われており，機械翻訳を始めとする複数の分野で精度の向上が確認されている[4]．まだ適応されていない分野でも結果の向上が期待できる．

従って，本研究では負担軽減の１つとして分散表現を用いて議論中での話題の変化を人間の代わりに検知することでファシリテーターの負担を軽減することを目指す．

以下に，本論文の構成を示す．第２章では分散表現を用いた話題変化判定を示す．第３章では評価実験を行い，第４章で本論文のまとめを示す．

2. 分散表現を用いた話題変化判定

COLLAGREE を始めとする議論掲示板では，１つのテーマに対して関連のある複数のテーマを扱う発言が投稿され，場合によってはある投稿者の発言が親意見となり, 他のユーザーが子意見として返信し，更に孫意見が存在する．

上記のような議論掲示板での発言に対して，本論文ではある発言 A と A の子意見，または発言 A と A 直後の発言の間の類似度を計算し，話題が変化したかの判定を行う手法を提案する．処理の流れは以下の通りである．

1. 文章の分解
2. 重要度の計算
3. 単語の重み付け
4. 分散表現
5. 類似度の計算

2.1. 文章の分解

文章から単語へ分割するにあたっては形態素解析エンジン mecab を使用した．

2.2. 重要度の計算

「そうですね」や「はい」のような短く，名詞などの少ない文章は大きな意味は無いが，他の文章との差異が大きくなってしまう傾向があったことから文章中の動詞や名詞の数を集計し，各品詞等の数を基に文章の重要度を求めることを考案した．提案手

法では重要度

$$\hat{\mathrm{Imp}}(s) = \frac{\sum_{i \in PoS} b_i \cdot num(s,\ i)}{a \cdot N}$$

$\mathrm{Imp}(s)$:文章 s の重要度
N:全文書数
Pos:品詞集合

$num(s,\ i)$:文章 s 中に現れた品詞 i の数

a:係数
b_i:品詞 i に対する係数

を求め，値が閾値mを下回ったものは重み付けの前に除外する．

係数aに関しては$a = 2.0$とし，係数b_iに関しては

$b_{\text{名詞}} = 1.0,\ b_{\text{固有名詞}} = 2.0,\ b_{\text{動詞}} = 1.0,\ b_{\text{形容詞}} = 1.0,\ b_{\text{副詞}} = 1.0,\ b_{\text{その他}} = 0.5$

閾値mに関しては$m = 0.45$とした．

2. 3. 単語の重み付け

複数の文書が存在する時，それぞれの文書を特徴付ける単語が特定したくなることがある．単語の特定の基準の 1 つとして TF-IDF という値が使われる．

始めに，TF について説明する．TF は Term Frequency の略で，それぞれの単語の文書内での出現頻度を表し，多く出てくる単語ほど値が大きくなり，重要性が高いことになる[6].

次に，IDF について説明する．IDF は Inverse Document Frequency の略で，それぞれの単語がいくつの文書内で共通して使われているかを表す．いくつもの文書で横断的に使われている単語は値が小さくなり，重要性が高くないことになる[7].

TF-IDF は TF と IDF を掛けたもので，TF-IDF が大きいほどそれぞれの文書を特徴付ける単語であると言える．提案手法では分割された単語の集合の中から TF-IDF が高いものを取り出している．

2. 4. 分散表現

自然言語処理において単語の意味を機械に認識させる時，幾つかの方法がある．認識させる方法の 1 つに単語ごとに人手で意味を付ける方法があるが，人手による手法には幾つか問題点がある．

1. 主観的である．
2. 人間への負担が大きい．
3. 単語間の類似度計算が困難である．

上記の問題を解決するための手法として，単語の言語学的な意味ではなく，文書集合中で周囲に出現している単語の分布を求め，分布を圧縮して密にすることによって単語を低次元の実数値ベクトルで表す方法が考案された[5]．具体的な例を図 1 で示す．単語の分布に基づき分散表現を使用することで客観的かつ機械の手による意味が付属され，実数値ベクトルであることから数学的な処理が可能になり，単語間の類似度を計算することが可能となっている．分散表現を示した図を以下の図 1 に示す．

50~300次元

家	0.9	0.1	0.3	...
犬	0.2	0.8	0.4	...
猫	0.1	0.7	0.4	...
	...	...	...	...

図 1:単語の分散表現

提案手法では分散表現を獲得するにあたり fastText[5] と呼ばれる分散表現への変換手法を使う．学習の際のコーパスには wikipedia の記事データを使用した．

2.5.類似度の計算

類似度計算においては Python 用の自然言語処理ライブラリ Gensim を使用している．Gensim には単語の集合の類似度を計算する関数が実装されているが本研究で新しい手法を提案する．

類似度を計算する際に単語を集合で比べるよりも一対比較を行った方が精度が高くなると判断したことから提案手法では２つの単語の集合同士をまとめて比較するのではなく，２つの集合中の単語全てで一対比較を行って類似度

$sim(w1_i,\ w2_j)$:単語$w1_i,\ w2_j$間の類似度

を求め，その平均を文章同士の類似度

$$similariy\left(s_{1,}\ ,\ s_2\right) = \frac{\sum_{w1_i \in s_1} \frac{\sum_{w2_j \in s_2} sim(w1_i,\ w2_j)}{num(s_2)}}{num(s_1)}$$

$similarity(s_1,\ s_2)$:単語集合 $s_1,\ s_2$間の類似度

$num(s)$:単語集合 s 中の単語数

とする手法を考案した．

3. 実験

3.1.実験概要

話題変化の検出の実験にあたり，COLLAGREE[3]で取られた「外国人旅行客向けの日本旅行プランに関する議論」の議論データを使用した．実験においては２つのデータと基準を使用した．データ 1 は議論の進行を支援するファシリテーターが発言しない議論のデータで，データ２ファシリテーターが積極的に発言する議論のデータである．

個々の発言データに話題が変わったかのタグ付けを行い，とある発言 A と A に対する返信の発言，またはとある発言 A と A 直後の発言のどちらかのペアで提案した手法による比較を行い，類似度が閾値を下回った場合を変化ありと検出し，検出された発言が適切かの判別を行った．

評価の際の指針として下記の３つのものを設けた．

1. 正解率:話題が変化したと判断された発言の内，何%が正しく検知されていたか.
2. 網羅率:話題が変わったとタグ付けされた発言の内，何%を正しく検知できたか.
3. 総検知率:全ての発言の内, 何%を話題が変化したと検知したか.

評価基準として以下の通り，A と B を設けた．

A) ファシリテーターの発言は基本的に話題を変える発言であることが多いので検出した発言がファシリテーターに関するものか否かの判別を行い，正誤率や網羅率を評価する．よって，２つの発言の主の一方がファシリテーターであれば正解とする．

B) 重要であると思われる発言に対してタグ付けを行い，検出した発言にタグがついていた場合正解とする．

また，手法としては下記の３つを使用し，比較した．

- 手法 1: 2.5 での類似度計算のみを使用する．
- 手法 2: 手法 1 に 2.2 での重要性推定を追加し，重要でないと思われる単語は事前に除外してから類似度計算を行った．
- 手法 3: 手法 2 に 2.3 での TF-IDF による単語の重み付けを追加し，２つの発言からそれぞれ重要性が高いと思われる単語を抽出してから，類似度計算を行った．

3.2.実験結果と考察

実験の結果を表 1 に示す.

表 1:実験結果

	手法 1	手法 2	手法 3
評価基準 A (データ 1)	**正:35.3% 網:19.2% 総:10.7%**	**正:35.1% 網:7.6% 総:3.3%**	**正:43.9% 網:22.9% 総:10.4%**
評価基準 A (データ 2)	**正:63.1% 網:23.5% 総:4.92%**	**正:64.6% 網:11.3% 総:4.9%**	**正:71% 網:38.5% 総:15.7%**
評価基準 B (データ 1)	**正:33% 網:19.2% 総:10.7%**	**正:50% 網:12.1% 総:3.3%**	**正:44.5% 網:46.3% 総:10.4%**
評価基準 B (データ 2)	**正:33.2% 網:31.7% 総:15.4%**	**正:66.7% 網:23.5% 総:4.9%**	**正:47.4% 網:60.1% 総:15.7%**

手法 1 から機能を追加していくことで，最終的に多くの場合で総検知率を大きく上げずに正解率，網羅率を上昇させることに成功した．

また，今後の展望としては発言からの単語抽出において更なる工夫が精度を上げるために必要である．

4. まとめ

本研究では分散表現を用いて議論中の話題変化の判定を行った．評価実験により精度が上がってきていることを示した．一方で精度を上げる余地がまだあることも確認した．今後の展望として発言からの単語抽出の改良について現状の TF-IDF を使用した単語抽出ではなく，生成的要約による手法を検証する．

参考文献

[1] Takayuki Ito,Yuma Imi, Eizo Hideshima,,"COLLAGREE: A Faciliatormediated, Large-scale Consensus Support System" ,Collective Intelligence, 2015

[2] 伊藤 孝紀,深町 駿平,田中 恵, 伊藤 孝行, 秀島 栄三, ファシリテータに着目した合意形成支援システムの検証と評価, デザイン学研究, 62, 2015, 4_67-4_76

[3] 伊美裕麻, 伊藤孝行, 伊藤孝紀, 秀島栄三.. 大規模意見集約システム COLLAGREE の開発と名古屋市次期総合計画に関する社会実験. 人工知能学会全国大会論文集, 28, 2014, 1-4

[4] Tomas Mikolov, Quoc V. Le, Ilya Sutskever, "Exploiting

Similarities among Languages for Machine Translation", CoRR, abs/1309.4168, 2013

[5] Piotr Bojanowski , Edouard Grave , Armand Joulin and Tomas Mikolov Enriching Word Vectors with Subword Information, 2016.

[6] Hans Peter Luhn, "A Statistical Approach to Mechanized Encoding and Searching of Literary Information", ournal of research and development. IBM, 1, 1957, 315

[7] Karen Sparck Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", 28, 1972, 11-21

付 録C

IJCAI-16

投稿論文

The 25th International Joint Conference on Artificial Intelligence (IJCAI-16) に投稿した論文を示す.

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
```

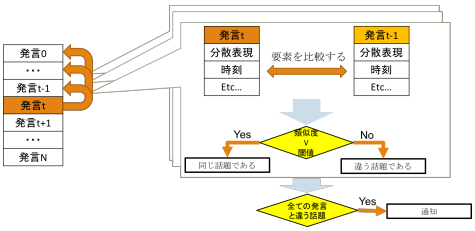


図 1 システムの流れ

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

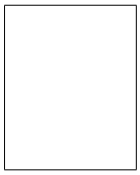
実験結果を表 2 に示す．

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．