* 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．



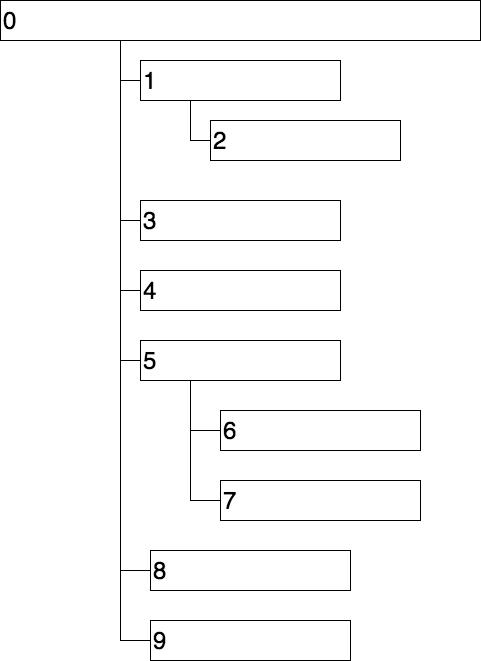
* 2.1: COLLAGREE のトップ画面

空白

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，

* 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．



* 2.2: COLLAGREE のスレッドの例

3

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 COLLA-

GREE での実際の議論画面を示す．

* 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．

4

* 2.3: COLLAGREE の議論画面



5

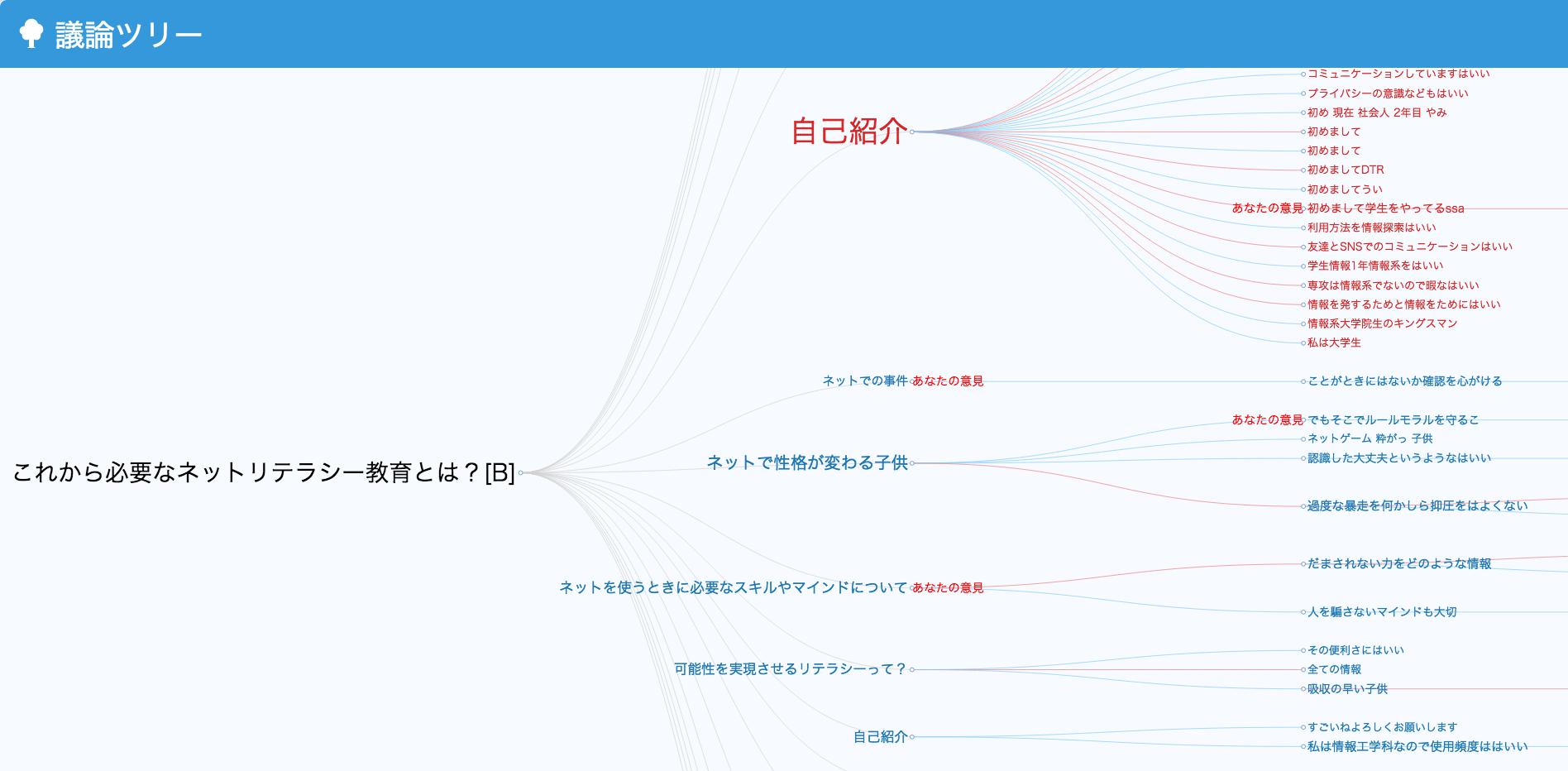


図 2.4: 議論ツリー

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

* [?] 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．

6

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，COL-

LAGREE 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

* 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

7

1. すでに言及された対象や事態の異なる側面への言及
2. 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
3. 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
4. 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

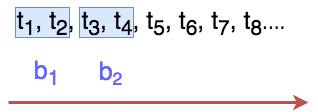
る．図 2.5 に手法の簡単な流れを示す．

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，

8



* 2.5: TextTiling の流れ

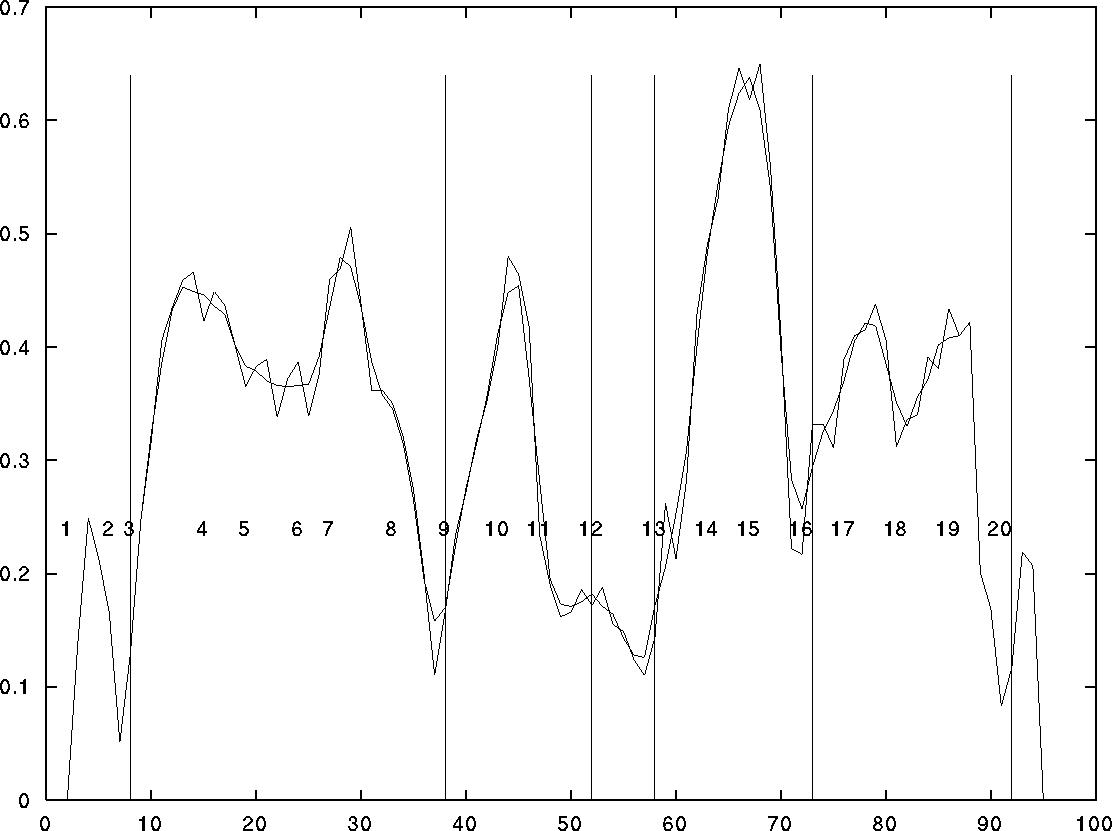
更に平滑化したものの２つが描かれている．グラフの谷のような場所は左右のブロック間の類似度が下がる場所を表している．類似度が下がることは使われる単語が変わったこと，すなわちトピックが変わったことを意味しており，谷の付近に境界を設けることで文書をトピックごとに区切ることを可能としている．

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，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．

9



* 2.6: TextTiling のグラフ

10

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．

11

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

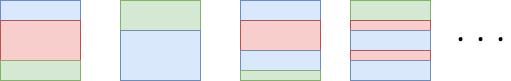
されると仮定するモデルである．



図 2.8: 混合ユニグラムモデル

LDA

* 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．



* 2.9: LDA

LDA では文書 *d* 中のトピック *t* の単語の割合 *p*(*tjd*) とトピック *t* で単語 *w* が生

成される確率 *p*(*wjt*)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．

12

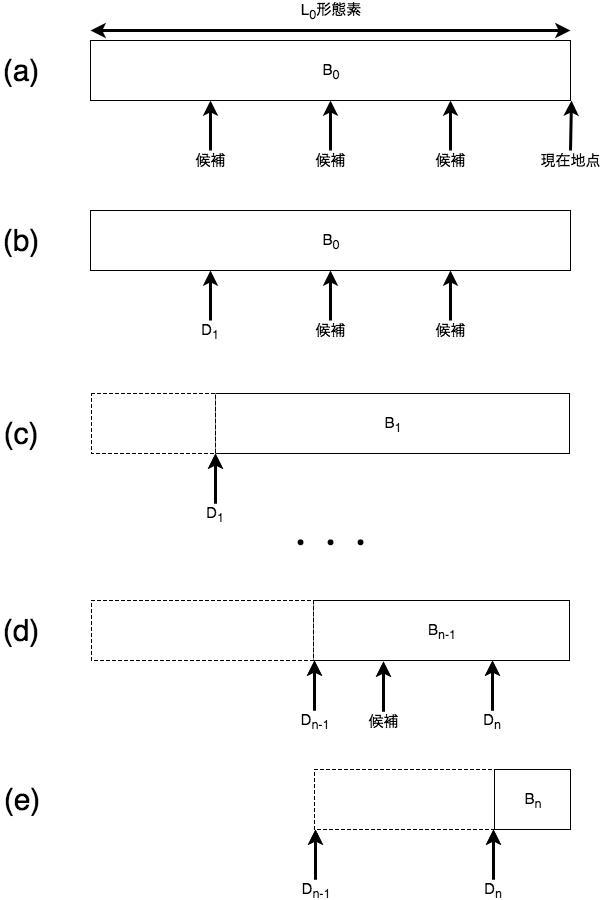


図 2.10: トピック変化点検出手順

13

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．

1. *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．
2. *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* 2)．

1. ブロック *Bn* 1に対し2.と同様の手順でトピック変化点候補箇所 *Dn* とその類似度 *Sn* を求める．図2.10(d)に該当する．
2. *Sn Sn* 1の場合，*Dn* 1をトピック変化点として処理を終了する．*Sn < Sn* 1の場合，*Bn* 1を *Dn* で分割し現在地点に近い側のブロックを *Bn* 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

14

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 Fre-

quency 逆文書頻度)[?]，DL(Document Length 文書長) の３つを用いて重要度の計

15

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 *ti* の文書 *Dj* 中における出現頻度重み *tfi;j* は式(2.1)

のようにして求められる．

|  |  |
| --- | --- |
| *nij* |  |
| *tfi;j* =∑*k nkj* | (2.1) |

∑

ここで *nij* は文書 *dj* における単語 *ti* の出現回数， *k nkj* は文書 *dj* 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 *ti* の逆文書頻度重み *idfi* は式(2.2)のようにして求められる．

|  |  |  |
| --- | --- | --- |
| *idfi* = log | *jDj* | (2.2) |
| *jfd* : *d ∋ tigj* |
|  |  |

ここで *jDj* は総文書数，*jfd* : *d ∋ tigj* は単語 *ti* 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．

16

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *dlj* | | (2.3) |
|  |  |  |
| *ndlj* = *ave*(*dl*) | | |
|  |

ここで *dlj* は文書 *dj* 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 *ti* の文書 *Dj* 中にお

ける統合重み *cwij* を求める．

|  |  |  |
| --- | --- | --- |
| *cwij* = | *tfi;j idfi* (*k*1+ 1) | (2.4) |
|
| *k*1(1 *b* + *b ndlj* ) + *tfi;j* |
|  |  |

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

17

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．

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| idf modi ed cosine(*x; y*) = |  |  |  |  | *w2x;y* | *tfw;xtfw;y*(*idfw*)2 | | | | | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | ( | *tf idf* |  | ) | 2 |  |  |  | ( | *tf* | | *idf* |  | ) | 2 |
|  |  |  |  |  |  |  |  |  |
|  | √∑ | | *xi2x* | ∑ | *xi* |  |  | √∑*yi2y* | |  | *yi;y* |  | *yi* |  |
|  |  | *xi;x* |  |  |  |  |  |  |  |

(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．

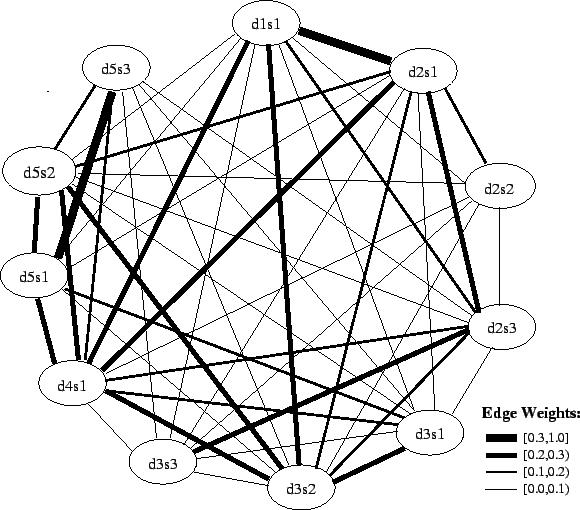


図 2.11: 類似度グラフの例

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

18

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 *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= *N*11;
6. *t* = 0;
7. repeat
8. *t* = *t* + 1;
9. *pt* = *M*T*pt* 1;

10: = *∥pt* *pt* 1*∥*;

1. until *< ϵ*
2. return *pt*;

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．

19

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 CosineM atrix*[*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. *CosineM atrix*[*i*][*j*] = idf-modi ed-cosine(*S*[*i*]*; S*[*j*]);
10. if *CosineM atrix*[*i*][*j*] *> threshold* then

|  |  |
| --- | --- |
| 11: | *CosineM atrix*[*i*][*j*] = 1; |
| 12: | *Degree*[*i*] + +; |

1. else

14: *CosineM atrix*[*i*][*j*] = 0;

1. for *i* 1 *to n* do
2. for *j* 1 *to n* do
3. *CosineM atrix*[*i*][*j*] = *CosineM atrix*[*i*][*j*]*=Degree*[*i*];
4. L = PowerMethod(*CosineM atrix; n;* ε);
5. return L

20

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．



* 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 *Mij* 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 *Mi* 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* = | *Mi Mj* | (2.6) |
|
| *jMijjMj j* | |  |

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

21

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．

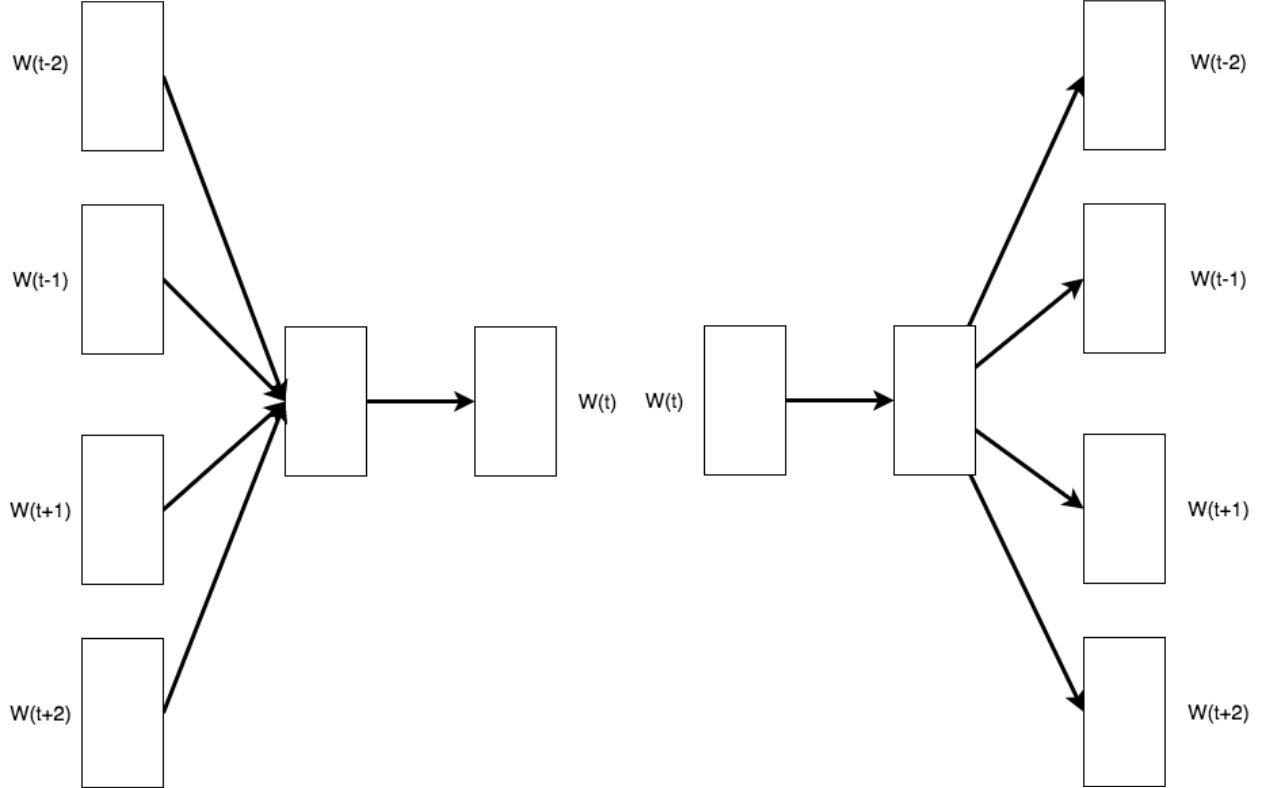


図 2.13: CBOW 図 2.14: skip-gram

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

22

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，

* = *vector*("*biggest*") *vector*("*big*") + *vector*("*small*") 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 *wt* が与えられた時に予測した文脈単語 *wc* の間のスコア関数を示す．

|  |  |
| --- | --- |
| *s*(*wt; wc*) = *uw*T*t vwc* | (2.7) |

23

|  |  |  |
| --- | --- | --- |
|  | ∑ |  |
| *s*(*wt; wc*) = | *zg*T*vwc* | (2.8) |

*g2Gw*

* + 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．*uwt* ，*vwc* はそれぞれ単語 *wt*，*wc* を実数ベクトルで表したもので，
* 2.8 において *Gw*，*zg* 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

24

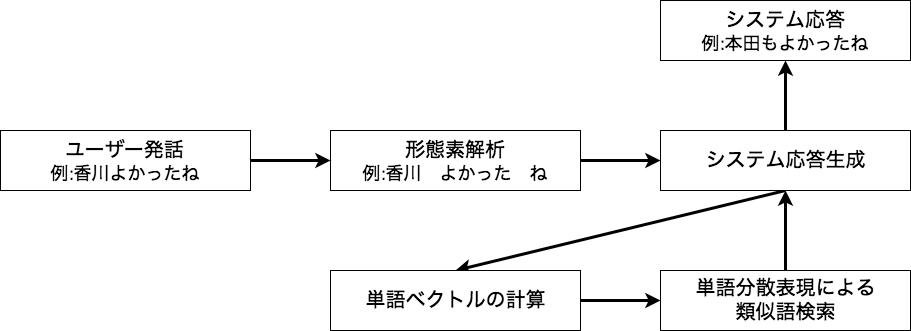


図 2.15: 話題展開システムの構造

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．

* 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．

25

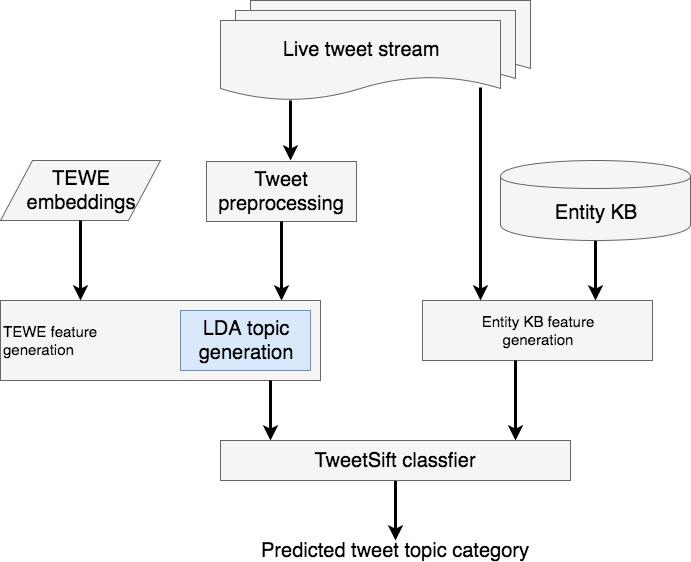


図 2.16: TweetSift によるトピック予測のワークフロー

26

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．

27