単語クラスタリングによって文書情報を整理する手法の改良

符 家俊 [*1]　村田 真樹 [*1,2]　馬 青 [*3]
[*1] 鳥取大学大学院 工学研究科 情報エレクトロニクス専攻
[*2] 鳥取大学工学部付属クロス情報科学研究センター
[*3] 龍谷大学 先端理工学部 数理・情報科学課程
[*1,2]{m20j4046h@edu. ,murata@}tottori-u.ac.jp
[*3] qma@math.ryukoku.ac.jp

1　はじめに

近年, インターネットの発展と伴い, ネット上の情報が急速的に増えている. これらの情報を上手く利用するために, ネット上の文書の情報を利用して, 重要な情報を表に整理する研究が考えられている. このような研究では重要な情報を表に整理して, 重要な情報を見やすいようにする. 作文の際に書き漏らしの部分を検出することもできる. 赤野ら [1] の研究では, 文書を単語単位で整理する. まずは人手でクラスタ数を 1000 と決定して, Wikipedia でのデータと Word2vec を用いて, クラスタリングする. そして, このクラスタリングの結果を用いて, 電子文書に出現するクラスタの単語を該当する行と列の箇所に埋めて, 表を作る. 岡崎ら [2] の研究では, 情報を文単位で抽出して, これらの情報をクラスタリングする際, 自動的に最適なクラスタ数や列の重要度を決めて, 情報を表に整理する.

岡崎ら [2] の研究では文で情報を整理するので, 単語のように細かく情報を整理することはできない. 一方, 赤野ら [1] の研究では重要な列が見つからない場合がある. 本研究ではこの欠点を改善するため, 単語レベルで情報を抽出しつつ, 岡崎らの研究を使って, 表の埋まり具合と情報の密集度を用いて, クラスタリングの際の最適なクラスタ数を推定して, 重要な列を自動的に選択する. 本研究での主張点は以下の 3 点である.

新規性

赤野ら [1] の研究は人手でクラスタ数 1000 で設定して, 人手で重要な列を選択する. 本研究では自動でクラスタ数を決定して, 重要度の順に列を自動的に並べ替える.

有用性

赤野ら [1] の研究では人手で重要な列を選択する必要がある. 本研究では手作業がなくても, 自動的に重要な列を選択することができる.

性能

本研究で整理した表の性能を F 値で評価すると, 平均値は 0.82 である. 異なるデータでの実験であるが F 値で赤野ら [1] の研究を評価すると, その平均値は 0.75 である.

2　従来手法

赤野ら [1] の研究では文書に含まれる情報を単語ごとに分割し, 種類ごとに分類し, 表に整理する手法を提案した. 表に整理する手順を以下に示す.

2.1　従来手法の手順

手順 1 抽出したい事柄を決定する. Wikipedia から, 抽出したい事柄を含む文書を抽出する.

手順 2 Wikipedia の全データを用いて, Word2vec で単語をクラスタリングする. 各クラスタにはクラスタ番号をふる.

手順 3 クラスタリング結果に基づく単語のクラスタを表の列とし, 文章を表の行とし, 文書に出現するクラスタの単語を該当する行と列の箇所に埋める.

手順 4 表の各列にある単語の頻度を求める. この頻度を用いて, 頻度の大きい列を左の方にするようにソートする. 頻度の少ないクラスタ番号の列を削除する.

手順 5 表のソート結果を基に人手で表の列を選択する.

2.2　従来手法の問題点

従来手法では Wikipedia の全データを使って, 人手でクラスタ数を 1000 に設定してクラスタリングし, このクラスタリングの結果と整理したい文書を比較し表を作るので, 列の数が多すぎて単語の頻度でソートしても重要な列を選択するのが難しいという問題がある.

3 提案手法

従来手法では Wikipedia 全データを使って, 人手でクラスタ数を 1000 で設定してクラスタリングする. 本研究では岡崎ら [2] の研究を使って, 表の埋まり具合と情報の密集度のバランスを最適にする方法でクラスタ数を推定し, クラスタリングする. 赤野ら [1] の研究では人手で重要な列を選択する. 本研究では岡崎ら [2] が提案した情報を自動的に重要度で整理する方法を使って, 文書中の重要な単語レベルの情報を自動的に選択し, 表に整理する. 提案手法の手順を以下に示す.

3.1 提案手法の手順

複数文書に含まれる単語の情報を表に整理する手順を以下に示す.

手順 1 文書に含まれる単語を MeCab で分割して, 品詞が名詞かつ連続している単語を一つの単語として扱う. このような単語は連続単語と呼ぶ. 名詞ではない単語を削除する.

手順 2 各文について, 文中の単語のベクトルを求め, 連続単語のベクトルはこの連続単語に含まれる単一の単語のベクトルの和をこの連続単語のベクトルとする.

手順 3 単語ベクトルを基に単語を Ward 法による階層クラスタリングでクラスタリングする.

手順 4 階層クラスタリングによって得られた各クラスタ数でのクラスタリング結果を基に, 表を作る. 表の埋まり具合と情報の密集度を使って, 最適なクラスタ数を推定し, 最適な表を選択する.

手順 5 手順 4 で採用されたクラスタ数でのクラスタリングの結果を, 行を文書, 列をクラスタとする, 表に整理する.

手順 6 表の各列について, 重要度を計算して, 重要度で並べ替える.

3.2 文から単語への分割方法

手順 1 における複数文書を単語単位に分割する方法を説明する. 文書を MeCab で単語ごとに分割する. 例えば「本体の前後に 400 万画素 CMOS センサー内蔵カメラをそれぞれ搭載し、全方位撮影できるモデル。」のような文が存在する. この文を MeCab で分割すると「本体 の 前後 に 400 万 画素 CMOS センサー 内蔵 カメラ を それぞれ 搭載 し 、全 方位撮影 できる モデル。」になるが, 分割しすぎる場合がある. 例えば「400」と「万」である. この問題を解決するため連続した名詞を一つの名詞として扱う. この方法で文を処理すると「本体 の 前後 に 400 万画素 CMOS センサー内蔵カメラ を それぞれ 搭載 し 、全 方位撮影 できる モデル。」となる. その後で, 名詞ではない単語を全部削除する. 以下に手順を示す. 図 1 に例を示す.

手順 1 文を MeCab を用いて分割する.
手順 2 連続の名詞を一つの名詞として扱う.
手順 3 名詞ではない単語を削除する.

処理前

本体の前後に 400 万画素 CMOS センサー内蔵カメラをそれぞれ搭載し、全方位撮影できるモデル。

分割して名詞のみ抽出

本体 前後 400 万画素 CMOS センサー内蔵カメラ それぞれ 搭載 方位撮影 モデル

図 1 処理結果の例

3.3 単語ベクトルの計算

連続の名詞を一つの名詞として扱うので, 連続の名詞ではない単語はそのまま Fasttext[3] を用いて, 単語ベクトルを計算する. 連続名詞の場合では以下の手順で単語ベクトルを計算する.

手順 1 連続の単語を MeCab で分割する.
手順 2 分割した単語を Fasttext[3] で単語ベクトルを求める. 連続単語に含まれる単語のベクトルの和をその連続単語のベクトルとして扱う.

3.4 単語ベクトルモデル

単語ベクトルを求めるとき使ったモデルは Fasttext[3] によって Wikipedia 全データを学習させたものである. Fasttext[3] は隠れ層と出力層からなる 2 層のニューラルネットワークで, CBOW や Skip-gram を用いて, 単語をベクトル化する.

3.5 Ward 法による階層クラスタリング

階層クラスタリング [4] は最初一つのデータを一つのクラスタとして扱う. そして距離が最も近いクラスタを統合する. この統合をそのまま設定したクラスタ数になるまで繰り返す. Ward 法は二つのクラスタ間の距離を推定する方法の一つである.

3.6 クラスタリング結果の score の計算方法

クラスタ数 k での表の埋まり具合 ($cover_k$) を式 (1) から求める. 情報の密集度 ($density_k$) を式 (2) からそれぞれ求める. クラスタ数 k での $score_k$ はこの 2 つの値を掛け算して求めた値である. これらの数式に現れる変数について説明する. C_k はクラスタ数 k での表で列の総数である. C_{ki} はクラスタ数 k での表に i 番目の列に含まれる単語の数である. W_{kij} はクラスタ数 k での表の i 番目の列で j 番目の単語のベクトルである. 単語の間の類似度は cos 類似度で求める.

$$cover_k = \frac{\text{クラスタ数 } k \text{ での表の空でないセルの数}}{\text{クラスタ数 } k \text{ での表のセルの総数}} \tag{1}$$

$$density_k = min\left(cos\left(W_{kij}, W_{kih}\right)\right) \tag{2}$$

$$i = 1, 2, ..., C_k \quad j, h = 1, ..., C_{ki}$$

ここで, cover と density のスケールが異なるため, クラスタ数 k での表の score を計算するとき影響を与えないように, この二つの値の正規化した値を掛け算したものをクラスタ数 k での表の score として扱う. そして score が最も大きいクラスタリング結果を選択する. 数式にある max(x),min(x) は x の最大値, 最小値を意味する.

$$norm\left(cover_k\right) = \frac{cover_k - min\left(Cover\right)}{max\left(Cover\right) - min\left(Cover\right)} \tag{3}$$

$$norm\left(density_k\right) = \frac{density_k - min\left(Density\right)}{max\left(Density\right) - min\left(Density\right)} \tag{4}$$

$$score_k = norm\left(cover_k\right) * norm\left(density_k\right) \tag{5}$$

3.7 列に重要度をつける方法

列の重要度の計算は score の計算と似ている. 列の密集度と列のカバー率を掛け算した値をその列の重要度として扱う. 列の密集度は実際その列にある単語間の最小類似度である. この値が高いほど, 列にある単語の関連性が高い. この列の関連性と列のカバー率を掛け算した結果の値が大きいと, この列の重要度も高いと思われる. 数式は以下で示す.

$$cover_i = \frac{\text{表の } i \text{ 番目の列の空でないセルの数}}{\text{表の } i \text{ 番目の列のセルの総数}} \tag{6}$$

$$density_i = min\left(cos\left(W_{ij}, W_{ih}\right)\right) \tag{7}$$

$$i = 1, 2, ..., C \quad j, h = 1, ..., C_i$$

$$norm\left(cover_i\right) = \frac{cover_i - min\left(Cover\right)}{max\left(Cover\right) - min\left(Cover\right)} \tag{8}$$

$$norm\left(density_i\right) = \frac{density_i - min\left(Density\right)}{max\left(Density\right) - min\left(Density\right)} \tag{9}$$

$$score_i = norm\left(cover_i\right) * norm\left(density_i\right) \tag{10}$$

4 実験

4.1 実験データ

実験で用いるデータは毎日新聞で抽出した強盗に関する記事 20 件と Wikipedia から抽出した城に関する記事 20 件である. データに関する詳細は表 1 に示す.

表 1 複数文書の詳細

	文書数	1 文の平均文字数
城に関する記事	20	32.45
強盗に関する記事	20	43.75

4.2 実験結果

強盗に関するデータを用いて, 出力された表のうち重要度が高い部分を表 2 に示す. ここで, 列はクラスタを示している. 行は記事を示している.

表 2 出力された表の一部

記事番号	クラスタ 1	クラスタ 2	クラスタ 3
記事 1	男, 男たち, 男 4	24 日午前 1 時 5 分ごろ	
記事 2	男, 男, 男	23 日午前 6 時半ごろ	
記事 3	男, 男, 男	29 日午後 9 時 15 分ごろ	25 万円
記事 4	男, 男, 男	9 日午前 2 時 45 分ごろ	40 万円
記事 5	男 4 人	13 日午前 4 時 5 分ごろ	1 万円
...	...	...	...

4.3 実験精度の評価

F 値でクラスタリングの性能を評価する. 実験で得られた表で人が重要と思う列のうち重要度が最も高い上位 10 列を使って, 人手で正解の表を作って, 再現率と適合率を計算して, F 値でクラスタリングの結果を評価する. 適合率, 再現率と F 値を以下の数式で示す. 実験結果を用いて, 計算した F 値を表 3, 表 4 に示す. 表 3, 表 4 には 3.7 節の方法で求める列の重要度も示している. 表 3, 表 4 に人手で正しい列名を括弧に書いている. 計算した F 値の平均結果は 0.82 である.

$$\text{適合率} = \frac{\text{正解の表の列と実験結果の表の列に共通する文の数}}{\text{実験結果の表の列に含まれる文の数}} \tag{11}$$

$$再現率=\frac{正解の表の列と実験結果の表の列に共通する文の数}{正解の表の列に含まれる文の数} \quad (12)$$

$$F\ 値=\frac{2\times適合率\times再現率}{適合率\ 再現率} \quad (13)$$

列の重要度を評価することも重要なことである. 強盗に関する記事で作成した表の最初の 48 列を人手で重要かどうかを評価した結果を表 5 に示す. 列名は列に出現する頻度が最も高い単語とする.

表 3　強盗に関する記事の評価結果の精度

列名	F 値	重要度
男（加害者）	0.80	0.91
午前 1 時（時間）	1.00	0.88
6000 円（場所）	0.75	0.73
170 センチ（身長）	0.87	0.68
20 歳（年齢）	0.75	0.66
府警（警察署）	0.75	0.64
強盗事件（事件種類）	0.91	0.54
逃走（逃走した？）	0.93	0.36
男性 (被害者性別)	1.00	0.32
刃物 (凶器)	0.71	0.30
平均値	0.84	

表 4　城に関する評価結果の精度

列名	F 値	重要度
城（城名）	1.00	0.67
愛媛県松山市（場所）	0.76	0.49
江戸時代（時代）	0.87	0.43
城跡（城郭形態）	0.70	0.38
山城（山や平地への築城）	0.70	0.18
日本 100 名城（日本 100 名城）	0.91	0.17
1933 年（時間）	1.00	0.09
整備（改築した？）	0.72	0.07
重要文化財（重要文化財？）	0.85	0.01
要塞（要塞？）	0.50	0.01
平均値	0.80	

4.4　考察

表 3,4 のように, 強盗に関する記事の中で加害者, 被害者, 犯罪時間, 犯罪場所, 犯罪者の特徴などの重要な情報を抽出できた. これらの情報の重要度が高いと思われる. 城に関する記事の中で城の場所, 城が建てられる時間などを抽出できたが, 有益ではない情報（例えば, 指定, 国など）も多く抽出された. 有益ではない情報が抽出された原因はこれらの単語は何も有用な情報を持っていないが, 出現する頻度か高かったからである. 強盗に関する事件で, 抽出された情報が重要かどうかを人手で評価した結果を表 5 に示す. 人手で重要と判断したものに○を, 重要でないと判断したものに×をつけている . 列の重要度が高いほど, 重要と思われる確率が高いと見える. 表 5 を見ると, 最初の 48 列について 12 列ごとに人が重要と思う列の比率を計算すると, 58.3%,33.3%,33.3%,25.0%である. この計算結果を見ると, 重要度が高いほど, 人が重要と思う比率も高いことが確認できるが, 城に関する記事を使って, 得られた表の列は少ないので, 8 列ごとに計算すると, 0.375,0.375,0.142,0.285 である. 重要順になっていない.

5　終わりに

本研究の提案手法では文を単語レベルで分割して, 名詞だけ残して, Fasttext を用いて, 単語をベクトル化して, 階層クラスタリングを用いて, 表のカバー率と密集度のバランスによって, 最適なクラスタ数を推定して, 最適なクラスタ数で単語をクラスタリングする. クラスタリングした結果の表の列の重要度を計算して, 重要な列を自動的に選択する. 城に関する記事と強盗に関する記事を用いて実験を行った. F 値で提案手法の性能を計算すると, 平均値は 0.82 であった. 提案手法ではクラスタ数を自動的に推定することができる. さらに, 自動的に重要な列を選択することができる. 実験結果では列の重要度が高いほど, 人が重要と思う比率が高いことを強盗に関する記事を使った実験で確認し, 提案手法の有効性を確認できた. 城に関する記事での実験では抽出された列が少なく, 一部の結果は重要順になっていないという問題がある. これは今後の課題である.

表 5　評価結果の精度

列名	重要性	列名	重要性	列名	重要性
男	○	男性	○	もの	×
午前 1 時	○	路上	×	大阪府大東市三箇	○
6000 円	○	タクシー	×	ローソン	○
捜査	×	刃物	○	乗車	×
170 センチ	○	客	×	アルバイト	×
20 歳	○	軽傷	○	紺色	×
府警	○	顔	×	刃渡り	×
調べ	×	ところ	×	パーカ	×
身長	×	レジ	×	防犯カメラ	×
けが	×	大阪市北区中津	○	3 人	○
強盗事件	○	後部座席	×	かばん	×
金	×	現金	○	首	×
いずれ	×	店内	×	車内	×
売上金	×	姿	×	千葉市中央区本千葉町	○
逃走	○	店員	×	野球帽	○
同署	×	ナイフ	×	会社員	×

参考文献

[1]Hokuto Akano, Masaki Murata, and Qing Ma. Detection of inadequate descriptions in Wikipedia using information extraction based on word clustering. *IFSA-SCIS 2017*, pp. 1–6, 2017.

[2]岡崎健介, 村田真樹, 馬青. 複数文書からの文レベルの情報の書き漏らしの検出. 言語処理学会第 25 回年次大会,

pp. 359–362, 2019.

[3]Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *In Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135–146, 2017.

[4]Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.