概要

近年, Web 上のテキストデータの要約や, 情報抽出に関する様々な手法が提案されている. その一環として, 本研究では関連した文章から項目ごとに情報を整理することで表の作成を行う. 一般的に要約は, 長文から情報の取捨選択をし, それ以外の情報を捨てることになる. しかし, 捨てられた情報の中にも場合によっては必要な情報となる場合もある. そこで, 情報の取捨選択ではなく項目ごとに整理することで, もとの文章の情報量を維持しつつ, 読み手は項目に注目することで素早く必要な情報を取得できる. また, 近年 Web 上のデータを収集し, 要約・構造化することでデータベースを構築するプロジェクトがいくつか存在する. 文章を表へ整理することはこのようなデータベースを構築する際に役立つと考えられる.

岡崎の研究 [?] では, 同じ事柄について書かれた複数の記事に対して, 文を項目ごとに整理していた. この際, 文章を文ごとに区切り, 文内に含まれる一部の名詞の埋め込み表現を平均したものを文ベクトルとし, これを基準にクラスタリングしていた. しかし, 抽出されるべき情報が抜け落ちることなどで精度に問題があった. この問題は, 表の生成に大きく関わる埋め込み表現とクラスタリングの精度によるものと考え, これらの改善することで表の精度向上につながると考えた. クラスタリングの精度に関しては, すでに岡崎の研究で取り組まれていた問題で, 表の精度が向上している. そこで, 本研究では岡崎の手法をベースラインとし文ベクトルの生成を SentenceBERT[?] に置き換えるというアプローチを用いた. 従来の手法では既存の表を学習し, 文ベクトルを調整することが出来なかった. 一方, SentenceBERT はファインチューニングを使用することで既存の表を学習することができ, 表生成へ最適化することが出来ると考えた. 実験をしたところ, 先行研究の F 値は 0.57 であったが, ファインチューニングを行った SentenceBERT では F 値が 0.48 と性能が下がる結果となった. 下がった原因として, 第一に学習に使用したデータが 8 個の表と少なく, 各表でドメインも異なっていた. このことにより, 文ベクトルの生成を悪化させるような学習となったと考えられる.

目 次

第 1 章　はじめに **1**

第 2 章　先行研究 **3**

2.1　表に整理する手順 3

2.2　手順 2 文ベクトルの作成手順 5

2.3　手順 3 階層化クラスタリング 6

2.4　手順 5 項目名の付与 7

2.5　先行研究の問題点 7

第 3 章　提案手法 **8**

3.1　表に整理する手順 8

3.2　BERT 8

3.3　SentenceBERT 9

3.4　Batch All Triplet Loss 9

第 4 章　実験 **11**

4.1　実験条件 11

4.1.1　学習用データ 11

4.1.2　テストデータ 13

4.2　実験内容 15

4.2.1　先行研究 15

4.2.2　ファインチューニングなし 15

4.2.3　ファインチューニングあり 15

4.3　評価方法 16

4.4　結果 16

第 5 章　考察 **20**
5.1　従来手法とファインチューニングなしとの比較 20
5.2　従来手法とファインチューニングありとの比較 20
5.3　一部の表での大幅な適合率の低下 20
5.4　SentenceBERT による手法での精度向上 21

第 6 章　おわりに **22**

謝辞 **23**

図 目 次

1.1 入力した文章の例 2

2.1 表に整理する手順 4
2.2 表に整理する手順 5
2.3 階層化クラスタリングの流れ 6

3.1 BERT の処理の流れ 9
3.2 BERT の処理の流れ 10

5.1 一つのセルに文がまとまった例 (カメラ) 21

表 目 次

1.1 出力された表の例 2

3.1 Triplet の例 10

4.1 学習で用いる各複数文書の詳細 12
4.2 実験で用いる各複数文書の詳細 14
4.3 実験結果 17
4.4 先行研究結果 17
4.5 ファインチューニングなしの結果 17
4.6 ファインチューニングありの結果 17
4.7 最も F 値が高い結果 (力士) 18
4.8 最も F 値が低い結果 (カメラ) 19

第1章　はじめに

近年, Web 上のテキストデータの要約や, 情報抽出に関する様々な手法が提案されている. その一環として, 図 1.1, 表 1.1 のように, 本研究では関連した文章から項目ごとに情報を整理することで表の作成を行う. 一般的に要約は, 長文から情報の取捨選択をし, それ以外の情報を捨てることになる. しかし, 捨てられた情報の中にも場合によっては必要な情報となる場合もある. そこで, 情報の取捨選択ではなく項目ごとに整理することで, もとの文章の情報量を維持しつつ, 読み手は項目に注目することで素早く必要な情報を取得できる. また, 近年 Web 上のデータを収集し, 要約・構造化することでデータベースを構築するプロジェクトがいくつか存在する. 文章を表へ整理することはこのようなデータベースを構築する際に役立つと考えられる.

岡崎の研究 [?] では, 同じ事柄について書かれた複数の記事に対して, 文を項目ごとに整理していた. この際, 文章を文ごとに区切り, 文内に含まれる一部の名詞の埋め込み表現を平均したものを文ベクトルとし, これを基準にクラスタリングしていた. しかし, 抽出されるべき情報が抜け落ちることなどで精度に問題があった. この問題は, 表の生成に大きく関わる埋め込み表現とクラスタリングの精度によるものと考え, これらの改善することで表の精度向上につながると考えた. クラスタリングの精度に関しては, すでに岡崎の研究で取り組まれていた問題で, 表の精度が向上している. そこで, 本研究では岡崎の手法をベースラインとし文ベクトルの生成を SentenceBERT に置き換えるというアプローチを用いた. 従来の手法では既存の表を学習し, 文ベクトルを調整することが出来なかった. 一方, SentenceBERT はファインチューニングを使用することで既存の表を学習することができ, 表生成へ最適化することが期待される.

本研究の主な主張を以下に整理する.

- 本研究は SentenceBERT を用いて複数文章から表を作成した. 岡崎の手法では F 値が 0.57 であったが, ファインチューニングなしの SentenceBERT では F 値が 0.54, ファインチューニングありでは F 値が 0.48 と共に下がる結果となった.

- 本研究において, ファインチューニングをしたところ, 適合率が 0.30 以下と大きく下がった表が確認された. 適合率が大きく下がった表では, 一つのセルに異なる項目の文が多く出力されていた.
- ファインチューニングをし, 適合率が大きく下がった原因として, 学習に使用したデータが 8 個の表と少なく, 各表でドメインも異なっていた. このことにより, 性能を悪化させるような学習となったと考えられる.

本論文の構成は以下のとおりである. 第 2 章では, 本研究に関する過去に行われた研究についての説明と問題点を記述する. 第 3 章では, 本論文が提唱する手法について過去の研究と比較し記述する. 第 4 章では, 過去の研究と本論文が提唱する手法での実験手法と結果を記述する. 第 5 章では, 本研究の結果について考察をし, 第 6 章ではまとめを行う.

図 1.1: 入力した文章の例

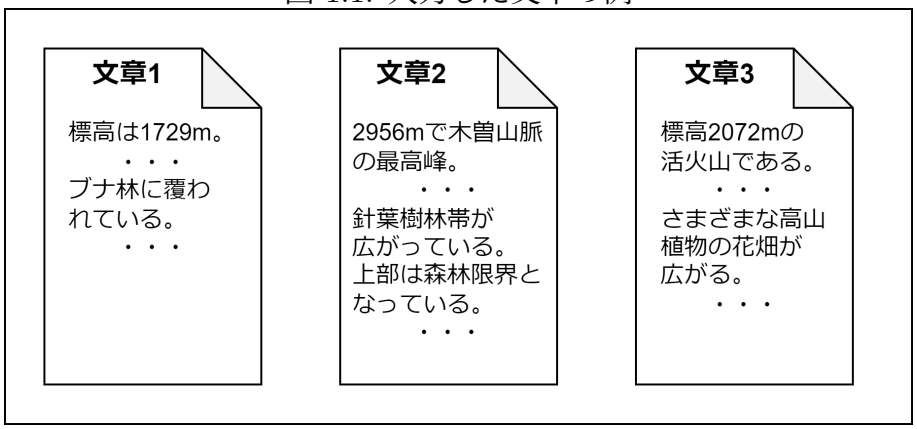


表 1.1: 出力された表の例

	標高	植生	地質
文章 1	標高は 1729m。	ブナ林に覆われている。	
文章 2	2956m で木曽山脈の最高峰。	針葉樹林帯が広がっている。 上部は森林限界となっている。	花崗閃緑岩から成る。
文章 3	標高 2,702m の活火山である。	さまざまな高山植物の花畑が広がる。	山頂付近は白山火山噴出物からなる。

第2章　先行研究

岡崎の研究では, 同じ事柄について書かれた複数の記事に対して, 文を項目ごとに整理していた. この際, 文章を文ごとに区切り, 文内に含まれる一部の名詞の埋め込み表現を平均したものを文ベクトルとし, これを基準にクラスタリングしていた.

2.1　表に整理する手順

文章を表に整理する手順を以下に示す. また, 手順の例を図 2.1 に示す.

手順 1 複数文書に含まれる文を句点区切りで抽出する.

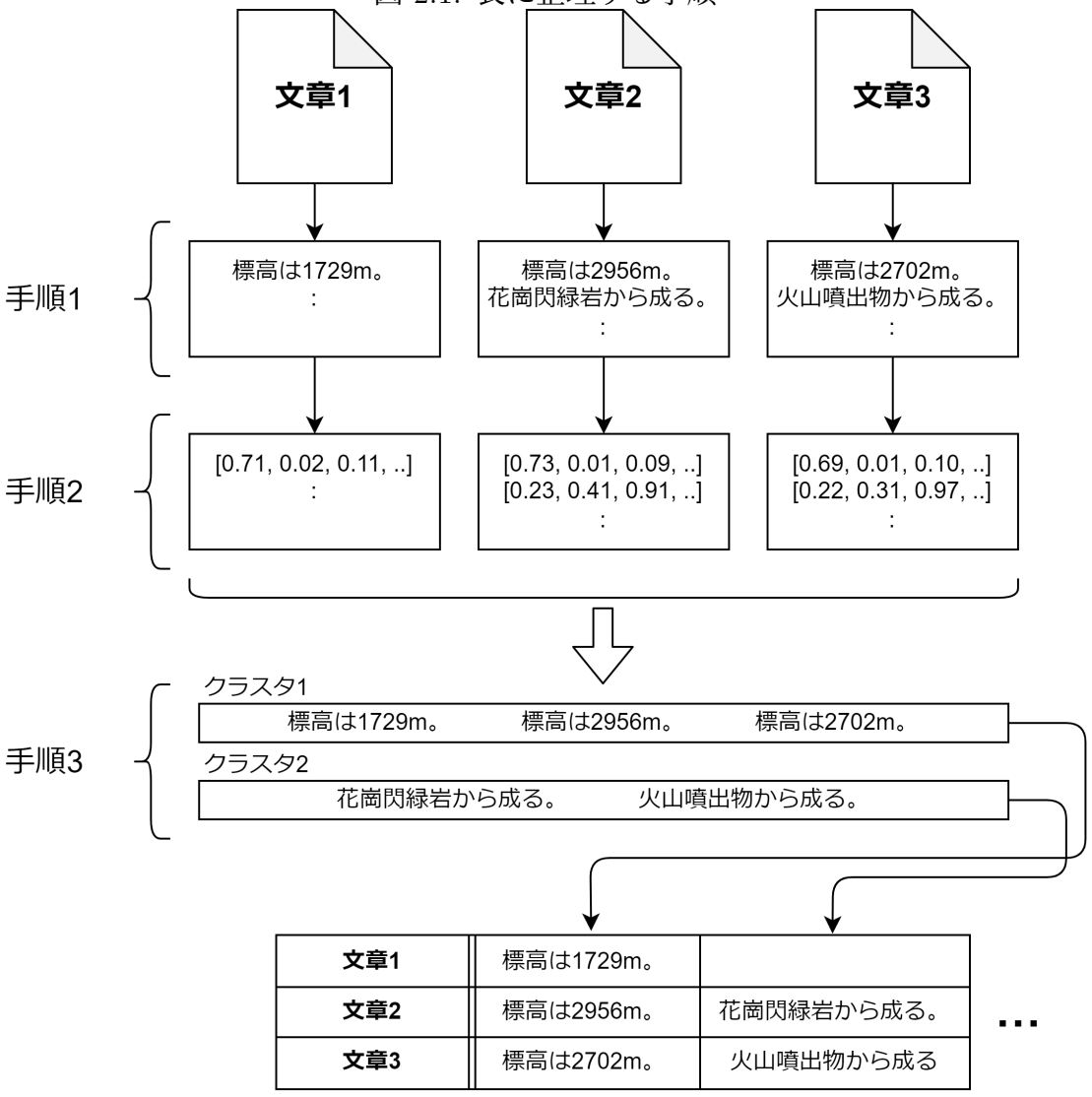
手順 2 抽出された文の文ベクトルを計算する.

手順 3 文ベクトルを基に文を 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 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 5 表の各列について, 項目名を付与する.

図 2.1: 表に整理する手順



2.2　手順2 文ベクトルの作成手順

2.1 節の手順 2 に関して, 文ベクトルの作成手順を以下に示す.

事前準備 Wikipedia から fastText[?] を用いて単語ベクトルのデータベースを作成する.

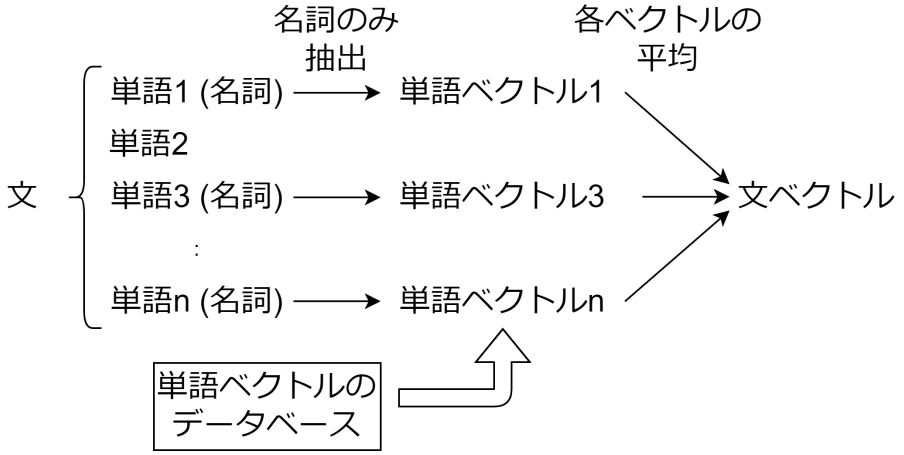
1. 文を MeCab[1] を用いて形態素解析を行う.

2. 形態素解析の結果, 各文から名詞かつ, MeCab が定義している, 代名詞，数，非自立, 副詞可能でない単語を抽出する.

3. 事前準備で用意した単語ベクトルのデータベースを用いて抽出した単語をベクトル化する.

4. 単語ベクトルを平均し, 文ベクトルとする.

図 2.2: 表に整理する手順



[1]MeCab https://taku910.github.io/mecab/

2.3　手順3 階層化クラスタリング

階層クラスタリングは，距離の最も近いクラスタ同士の統合を再帰的に繰り返すクラスタリング手法である. 階層クラスタリングには, クラスタ間の距離に関していくつか種類がある. 今回は Word 法を用いた.

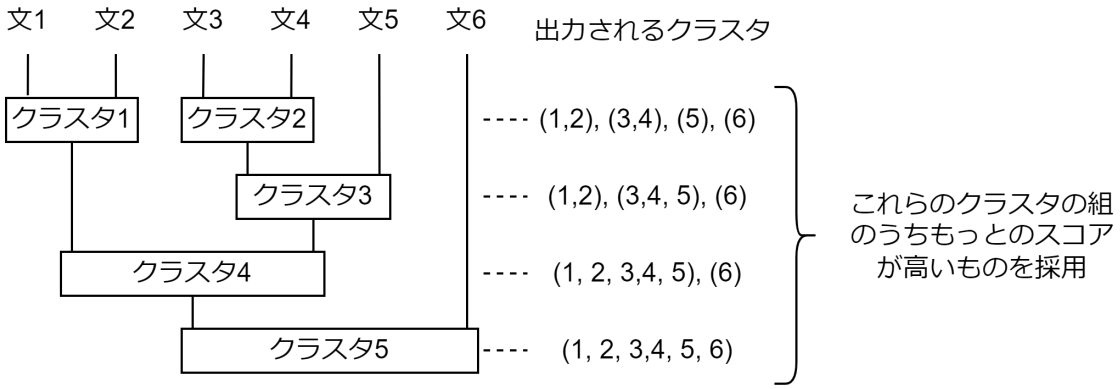
また, 図 2.3 のように階層化クラスタリングは, どの程度クラスタに分割するかで, 複数のクラスタリング結果が得られる. クラスタリングの結果に対して $score$ を定義し, 最も $score$ が高いクラスタリング結果を最終的な出力とした. $cosine(x, y)$ は x，y のコサイン類似度を求める関数, C をクラスタの集合, S_c をクラスタ c の文ベクトルの集合としたとき, スコアは以下のように計算する.

$$cover = \frac{\text{表の埋まっているセルの数}}{\text{表のセルの総数}} \tag{2.1}$$

$$density = \min(cosine(s1, s2)), (s1, s2) \in S_c \times S_c, c \subset C \tag{2.2}$$

$$score = cover \times density \tag{2.3}$$

図 2.3: 階層化クラスタリングの流れ



2.4 手順5 項目名の付与

2.1 節の項目名の付与の手順を以下に示す.

1. 列に含まれる各文について，文に含まれる品詞が名詞の単語を抽出する.
2. 1 で抽出した各単語について，列ごとに出現頻度を求める.
3. 各列について, 出現頻度が最大の単語をクラスタの項目名として付与する.
4. 出現頻度が最大の単語が複数ある場合は，読点で区切って全て付与する.

2.5 先行研究の問題点

岡崎の研究では, 十分な精度が出なかった. 先行研究は文のクラスタリングの際に, 単語のベクトルを fastText[?] で取得して, そのベクトルの平均としている. そのベクトルをクラスタリングすることで, 表を作成している. この方法では表の精度は文ベクトル及び, クラスタリングに大きく依存するため, この二つの精度が十分ではなかったといえる.

第3章　提案手法

提案手法では, 岡崎の手法をベースラインとして, 文ベクトルの作成を SentenceBERT に置き換えるというアプローチで精度向上をはかる. 従来の手法では既存の表を学習し, 文ベクトルを調整するということが出来なかった. 一方, SentenceBERT はファインチューニングを使用することで既存の表を学習することができ, 表生成を最適化することができるため, 提案手法で用いることにした.

3.1　表に整理する手順

文章を表に整理する手順を以下に示す.

手順 1 複数文書に含まれる文を句点区切りで抽出する.

手順 2 文ベクトルを計算する.

手順 3 文ベクトルを基に文を 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 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 5 表の各列について，項目名を付与する.

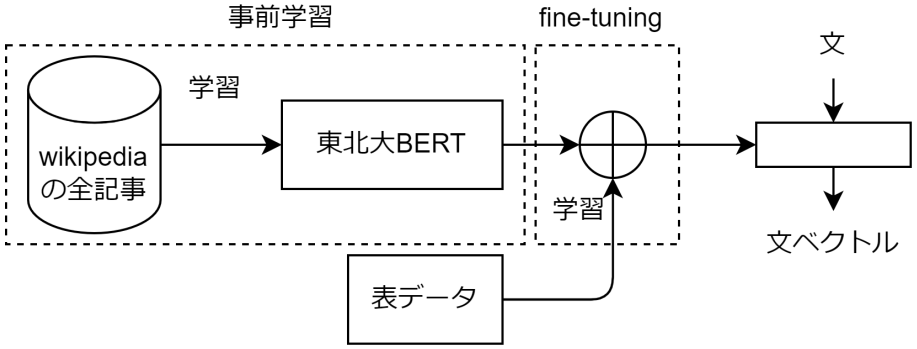
3.2　BERT

BERT[?] は言語そのものを学習する事前学習と各タスクに特化した学習を行うファインチューニングに分かれる. このように分割することで, 事前学習の学習データを使いまわすことができ, また少量のデータで, ファインチューニングをすることができ, 計算リソース, 学習データの節約が可能となるうえ, 大幅な精度向上が見込める.

3.3 SentenceBERT

SentenceBERT[?] は BERT の派生版であり, 意味的類似性検索や、クラスタリングといったタスクに適している. 通常の BERT はこれらのタスクに関して, 学習の際, 文の組み合わせを考える必要があり, 組み合わせの数が莫大となり計算に多くの時間がかかった. SnentenceBERT は Siamese Network という手法を用いることで計算時間を短縮した. 事前学習モデルは従来の BERT のものが利用できるので, 本研究では図 3.1 にある通り, 事前学習のデータはに訓練済みである東北大 BERT [1] をダウンロードしたものを用い, ファインチューニングには人手で作成した表を用いる.

図 3.1: BERT の処理の流れ



3.4 Batch All Triplet Loss

教師ありの機械学習では一般的に, 学習データとモデルの出力の差を少なくするようにモデルを調整していく. 学習データとモデルの出力の差を損失 (Loss) といい, この損失の定義によって学習効率が異なる. 本研究では, ファインチューニングを行う際 Batch All Triplet Loss[?] を用いて学習を行う.

Batch All Triplet Loss[?] は 表 3.1 のように基準となる文 A(Anchor), 文 A と同じラベルの文 P(Positive), 文 A と異なる文 N(Negative) からなる三つの組を利用する誤差関数である. 誤差関数 $Tl(x_a, x_p, x_n)$ は 図 3.2 のように文 A に対して, 文 P を近づけ, 文 N を異なるように動作する. この誤差は次の式で計算できる. ただし, x_a, x_p, x_n はそれぞれ文 A, P, N の文ベクトルであり, α はハイパーパラメータである.

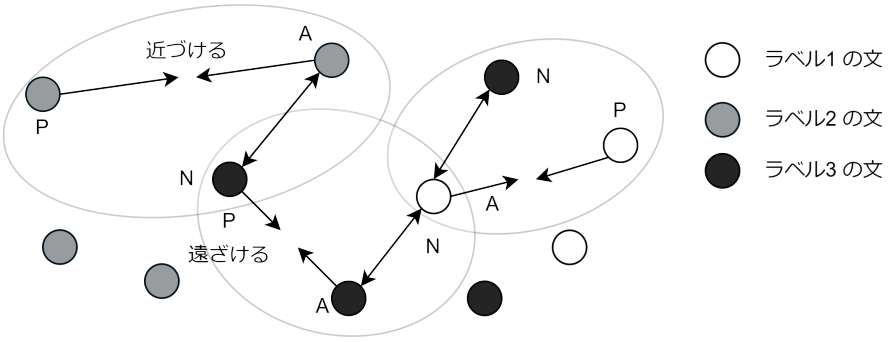
[1] 東北大学 乾研究室が公開している訓練済みモデル,
https://www.nlp.ecei.tohoku.ac.jp/news-release/3284/

$$Tl(x_a, x_p, x_n) = \max(\text{distance}(x_a, x_p) - \text{distance}(x_a, x_n) + \alpha, 0) \tag{3.1}$$

表 3.1: Triplet の例

	ラベル	文
Anchor	標高	標高は 1729m。
Positive	標高	2956m で木曽山脈の最高峰。
Negative	地質	山頂付近は白山火山噴出物からなる。

図 3.2: BERT の処理の流れ



第4章　実験

本研究ではベースラインと提案手法である SentenceBERT を使用したモデルを比較する. 提案手法では ファインチューニング を行い最適化されたものと, してないものを比較し, ファインチューニング がどの程度性能に影響するかを検証する.

4.1　実験条件

4.1.1　学習用データ

本研究では岡崎の研究で使用されたデータのうち 8 つを SentenceBERT のファインチューニング用として用いる. 具体的なデータの詳細は以下に示す.

- エアコンに関する新製品記事 20 件
  2018 年 1 月 15 日時点での「価格.com」のエアコン・クラーカテゴリーにおける最新の新製品ニュース記事 20 件を抽出したデータ
- 恐竜に関する Wikipedia の記事 20 件
  2017 年 6 月 1 日時点での Wikiedia のカテゴリー「ジュラ紀の恐竜」に含まれる全ページのうち，ランダムに抽出した 20 記事の要約部を抽出したデータ
- 地震に関する新聞記事 20 件
  2016 年度の毎日新聞から見出しに「地震」と「震度」を含む記事をランダムに 20 件抽出したデータ
- 野球チームに関する Wikipedia の記事 20 件
  2017 年 6 月 1 日時点での Wikiedia のカテゴリー「アメリカ合衆国の野球チーム」に含まれる全ページのうち，ランダムに抽出した 20 記事の要約部を抽出したデータ
- 山に関する Wikipedia の記事 20 件
  2017 年 6 月 1 日時点での Wikiedia のカテゴリー「日本百名山」に含まれる全ページのうち，ランダムに抽出した 20 記事の要約部を抽出したデータ

- 城に関する Wikipedia の記事 20 件
  2017 年 6 月 1 日時点での Wikiedia のカテゴリー「日本の 100 名城」に含まれる全ページのうち，ランダムに抽出した 20 記事の要約部を抽出したデータ

- 交通事故に関する新聞記事 20 件
  2016 年度の毎日新聞から見出しに「交通事故：」を含む記事をランダムに 20 件抽出したデータ

- テレビ関する新製品記事 20 件
  2018 年 1 月 15 日時点での「価格.com」の薄型テレビ液晶テレビカテゴリーにおける最新の新製品ニュース記事 20 件を抽出したデータ

表 4.1: 学習で用いる各複数文書の詳細

	文書数	総文数	1 文あたりの平均文字数
新製品記事 (エアコン)	20	255	62.3
Wikipedia(恐竜)	20	77	49.9
新聞記事 (地震)	20	91	37.2
Wikipedia(野球チーム)	20	68	46.9
Wikipedia(山)	20	76	31.5
Wikipedia(城)	20	94	31.2
新聞記事 (交通事故)	20	143	41.7
新製品記事 (テレビ)	20	273	49.0

4.1.2 テストデータ

本研究では岡崎の研究で使用されたデータのうち 8 つを SentenceBERT のファインチューニング用として用いる. 具体的なデータの詳細は以下に示す.

- 強盗事件に関する新聞記事 20 件
  2016 年度の毎日新聞から見出しに「強盗：」を含む記事をランダムに 20 件抽出したデータ

- 外為・株式に関する新聞記事 20 件
  2016 年度の毎日新聞から見出しに「外為・株式：」を含む記事をランダムに 20 件抽出したデータ

- リコールに関する新聞記事 20 件
  2016 年度の毎日新聞から見出しに「リコール：」を含む記事をランダムに 20 件抽出したデータ

- スマートフォンに関する新製品記事 20 件
  2018 年 1 月 15 日時点での「価格.com」のスマートフォンカテゴリーにおける最新の新製品ニュース記事 20 件を抽出したデータ

- デジタルカメラに関する新製品記事 20 件
  2018 年 1 月 15 日時点での「価格.com」のデジタルカメラカテゴリーにおける最新の新製品ニュース記事 20 件を抽出したデータ

- ロボット掃除機に関する新製品記事 20 件
  2018 年 1 月 15 日時点での「価格.com」の掃除機カテゴリーにおけるロボット掃除機に関する最新の新製品ニュース記事 20 件を抽出したデータ

- 力士に関する Wikipedia の記事 20 件
  2017 年 6 月 1 日時点での Wikiedia のカテゴリー「高校相撲部出身の大相撲力士」に含まれる全ページのうち，ランダムに抽出した 20 記事の要約部を抽出したデータ

表 4.2: 実験で用いる各複数文書の詳細

	文書数	総文数	1 文あたりの平均文字数
新聞記事 (強盗)	20	128	39.3
新聞記事 (外為・株式)	20	124	49.2
新聞記事 (リコール)	20	89	56.7
新製品記事 (スマホ)	20	313	46.3
新製品記事 (カメラ)	20	340	52.0
新製品記事 (ロボット掃除機)	20	235	47.7
Wikipedia(力士)	20	103	28.7

4.2 実験内容

4.2.1 先行研究

先行研究 [?] の実験ではテストデータに 4.1.2 節に示すものを使用していた.

4.2.2 ファインチューニングなし

ファインチューニングなしの BERT ではどの程度の性能が出るかを検証する. 事前学習はあらかじめ学習されたデータである東北大 BERT を使用する. また, テストデータに 4.1.2 節に示すものを使用する.

4.2.3 ファインチューニングあり

BERT の派生版である SentenceBERT を利用しファインチューニングを行う. 学習に使用するデータは 4.1.1 節に示し, テストデータに 4.1.2 節に示すものを使用する. また事前学習のデータは 4.2.2 節同様, 東北大 BERT を使用する.

4.3 評価方法

実験によって得られた表を, 正解と比較し評価する. この評価に基づいて, 再現率 (Recall), 適合率 (Precision), F 値を算出することで情報抽出の精度を求める. 生成された表の精度を以下の手順で算出する.

1. あらかじめ作成した正解の表の各列に注目する.
2. 注目している列に含まれるデータを最も多く含む列を抽出する.
3. 式 4.2，式 4.1，式 4.3 から適合率，再現率，F 値を求める.
   ただし, A を正解の列に含まれる文の集合, R を結果の列に含まれる文の集合であり, これらの絶対値は集合に含まれる要素数とする.
4. 2，3 をすべての正解の列に対して行い，各列の F 値の平均を求め，これを実験の表の評価結果とする.

$$\text{Recall} = \frac{|A \cap R|}{|A|} \tag{4.1}$$

$$\text{Precision} = \frac{|A \cap R|}{|R|} \tag{4.2}$$

$$\text{F} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \tag{4.3}$$

4.4 結果

先行研究と手案手法について, Web から持ってきた 15 の記事群を SentenceBERT 学習用に 8 つ, テスト用に 7 つ用いて実験を行った. 表 4.3 に結果を示す. この実験結果ではファインチューニングありの提案手法では, 適合率の低下により精度を落とす結果となり, ファインチューニングに問題があると推測できる. 各記事の評価について, 先行研究, ファインチューニングなし, ファインチューニングありの結果をそれぞれ表 4.4, 4.5, 4.6 に示す. また, ファインチューニングありの実験で最も精度が良かった結果「相撲」, 最も悪かった結果「カメラ」をそれぞれ表 4.7, 4.8 に示す.

表 4.3: 実験結果

	先行研究	ファインチューニングなし	ファインチューニングあり
適合率	0.62	0.66	0.51
再現率	0.74	0.54	0.56
F 値	0.57	0.54	0.48

表 4.4: 先行研究結果

	適合率	再現率	F 値
カメラ	0.98	0.57	0.65
掃除機	0.91	0.62	0.69
強盗	0.22	0.85	0.30
為替	0.61	0.76	0.61
リコール	0.35	0.84	0.43
スマートフォン	0.53	0.84	0.59
相撲	0.75	0.71	0.70

表 4.5: ファインチューニングなしの結果

	適合率	再現率	F 値
カメラ	0.89	0.17	0.29
掃除機	0.61	0.20	0.54
強盗	0.55	0.80	0.63
為替	0.82	0.39	0.52
リコール	0.33	0.72	0.41
スマートフォン	0.59	0.78	0.63
相撲	0.82	0.70	0.74

表 4.6: ファインチューニングありの結果

	適合率	再現率	F 値
カメラ	0.26	0.54	0.30
掃除機	0.30	0.57	0.35
強盗	0.53	0.59	0.54
為替	0.54	0.55	0.50
リコール	0.47	0.52	0.45
スマートフォン	0.69	0.52	0.56
相撲	0.75	0.66	0.66

表 4.7: 最も F 値が高い結果 (力士)

	出身	最高	身長
琴藤沢和穂	琴藤沢和穂は、高知県高知市出身で佐渡ヶ嶽部屋に所属した元大相撲力士。	本名は藤沢和穂。	
逆鉾昭廣	逆鉾昭廣は鹿児島県姶良市出身で井筒部屋所属の元大相撲力士。	本名は、福薗好昭。	身長 183 ｃｍ、体重 191 ｋｇ。
政風基嗣	政風基嗣は、長崎県長崎市出身で尾車部屋所属の現役大相撲力士。	本名は北園基嗣。	
栃の山博士	栃の山博士は、東京都立川市出身で、千賀ノ浦部屋の元大相撲力士で、現世話人。	本名は山田博士。	
栃乃里隆光	栃乃里隆光は、石川県石川郡野々市町出身で春日野部屋に所属していた元大相撲力士。	本名は矢鋪光太郎。	
誉富士歓之	誉富士歓之は、青森県西津軽郡鰺ヶ沢町出身で伊勢ヶ濱部屋所属の現役大相撲力士。	本名は三浦歓之。	身長は 180 ｃｍ、体重は 160 ｋｇ。
朝陽丸勝人	朝陽丸勝人は、大阪府枚方市出身で高砂部屋所属の元大相撲力士。	本名は三浦歓之。	
吉井山朋一郎	吉井山朋一郎は、福岡県田川郡糸田町出身で、出羽海部屋に所属した大相撲力士。	本名は吉井朋一郎。	現役時代の体格は 182 ｃｍ、92 ｋｇ。
朝乃翔嚆矢	朝乃翔嚆矢は、神奈川県小田原市出身で若松部屋所属の元大相撲力士。		本名は小塚一、身長 186 ｃｍ、体重 147 ｋｇ。
大岳宗正	大岳宗正は滋賀県草津市出身の元大相撲力士。	本名は横江英樹。	身長 185 ｃｍ、体重 148 ｋｇ。
大翔鳳昌巳	大翔鳳昌巳は、北海道札幌市豊平区平岸出身で立浪部屋所属の元大相撲力士。	本名は村田昌巳。	全盛期の体格は 187 ｃｍ、144 ｋｇ。
玉大輝剛志	玉大輝剛志は、石川県鳳珠郡能登町出身で片男波部屋に所属していた元大相撲力士。	本名は久山毅。	
開隆山勘之亟	開隆山勘之亟は、秋田県南秋田郡昭和町出身で、1960 年代に活躍した大相撲力士。		現役時代の体格は 179 ｃｍ、116 ｋｇ。
大翔鵬清洋	大翔鵬清洋は、モンゴル・ウランバートル市出身で、追手風部屋所属の大相撲力士。		身長 182 ｃｍ、体重 146 ｋｇ。
木村山守	木村山守は、和歌山県御坊市出身で春日野部屋所属だった元大相撲力士。		

表 4.8: 最もＦ値が低い結果 (カメラ)

	画素	撮影	SD
チーズのような超小型トイデジタルカメラ「DSC Pieni Cheese」、3,510円。	このほか主な仕様は、有効131万画素の１／10型ＣＭＯＳセンサーを装備。	シャッタースピードは１／100秒、ＩＳＯ感度はＩＳＯ100。	外部メモリーは、ｍｉｃｒｏＳＤＨＣメモリーカードに対応。
キヤノン、重量399gのAPS-Cコンデジ「PowerShot G1 X Mark III」	有効約2420万画素、約22．3×14．9mmのＡＰＳ－ＣサイズＣＭＯＳセンサーを、キヤノンのコンパクトカメラとして初めて搭載した。ＣＭＯＳセンサーの画素が撮像と位相差ＡＦの機能を併せ備えた「デュアルピクセルＣＭＯＳＡＦ」も採用。	動画機能は、1920×1080記録に対応する。	このほか主な仕様は、外部記録媒体がＳＤ／ＳＤＨＣ／ＳＤＸＣメモリーカードをサポート。
カシオ、G-SHOCK級のタフカメラ「G'z EYE GZE-1」	性能面では、撮像素子に有効690万画素の１／2．3型ＣＭＯＳセンサーを搭載する。記録画素数は、静止画が600万画素、動画がフルＨＤ／ＳＴＤ／ＨＳ240／ＨＳ120。		記録メディアは、内蔵メモリーが約14．8ＭＢ、外部メモリーがｍｉｃｒｏＳＤＸＣメモリーカードに対応する。
ソニー、0.03秒高速AF対応の超望遠600mmズームカメラ「RX10 IV」	ソニーは、デジタルカメラ「サイバーショット」シリーズより、メモリー一体1．0型積層型ＣＭＯＳイメージセンサーＥｘｍｏｒＲＳを搭載した「ＲＸ10ＩＶＤＳＣ－ＲＸ10Ｍ4」を発表。	画素加算のない全画素読み出しによる高解像度４Ｋ動画機能、撮影時間が増えた最大40倍のスーパースローモーション機能によって、プロの映像制作のニーズにも応える。このほか、静止画の記録フォーマットがＪＰＥＧ、ＲＡＷ、動画の記録フォーマットがＸＡＶＣＳ、ＡＶＣＨＤＶｅｒ．2．0に対応。ファインダーは0．39型の電子式有機ＥＬを採用し、背面モニターには、タッチパネル対応の３型チルト液晶を装備。	

[1]波線部は不適切な出力

第5章　考察

5.1　従来手法とファインチューニングなしとの比較

本研究によりファインチューニングなしの SentenceBERT を用いた表生成の精度は岡崎の手法と比べ精度が悪いことが分かった. 特に再現率が先行研究では 0.74 であったが, 0.54 と大きく低下していた. 再現率が低下した明確な理由は不明であるが, 本研究においてファインチューニングなしの SentenceBERT ではより多くのデータの取りこぼしが発生しているといえる.

5.2　従来手法とファインチューニングありとの比較

本研究ではファインチューニング前では F 値が 0.54 だったが, ファインチューニング後では 0.48 となり, 性能が低下することが分かった. 表 4.6 より, 特に「カメラ」と「掃除機」の F 値が悪いことが分かる. この二つの結果は共通して適合率が悪いということが分かる.

5.3　一部の表での大幅な適合率の低下

ファインチューニングをした SentenceBERT では, 一部の表で大幅に適合率が下がることが分かった. このことについて考察を行う. まず, 大幅に適合率が低下した結果である表 4.8 において「撮影」の列や図 5.1 のような異なる意味を持つ文が同じセルにまとまることが複数発生した. このことにより適合率が低下したと思われる. また, このようなセルが発生した理由として, ファインチューニングに原因があると推測する. 本研究ではファインチューニングに使用したデータが少なく, 複数のドメインが混ざっていた. そのため他のラベルに当てはまると判断された可能性がある.

高速性と追従性にすぐれた像面位相差...
315 点の像面位相差検出ＡＦセンサーを...
バッファメモリーの大容量化などにより、249 枚まで連続撮影できる
さらに、進化した画像処理エンジン...
動きのあるポートレート撮影で、正面から...
連写した静止画のグループ表示や、...
このほか、サイレント撮影が可能なため、...
加えて、ファストハイブリッドＡＦが...
計 8 枚のＥＤガラスを採用し、望遠撮影時に...
ズーム全域で、高コントラストかつシャープな...
超望遠や暗所撮影時に役立つ 4．5 段分の高い...
さらに、600 ｍｍで約 72 ｃｍまで被写体に寄れる、...
バッテリー使用時間は、静止画撮影時が約 400 枚、...

図 5.1: 一つのセルに文がまとまった例 (カメラ)

5.4 SentenceBERT による手法での精度向上

本研究では, ファインチューニングした SentenceBERT の性能が悪い原因を, 学習データが少なく, 複数のドメインが混ざっていたことが原因だと考察した. 本研究の手法で性能を上げるにはドメイン特化にし, 学習データを増やす必要があると思われる.

第6章　おわりに

本研究では, 岡崎の手法を教師あり学習が行えるように改良し表の生成を行った. その一環として, 本研究では関連した文章から項目ごとに情報を整理することで表の作成を行った岡崎の研究 [?] では, 同じ事柄について書かれた複数の記事に対して, 文を項目ごとに整理していた. この際, 文章を文ごとに区切り, 文内に含まれる一部の名詞の埋め込み表現を平均したものを文ベクトルとし, これを基準にクラスタリング行っていた. しかし, 抽出されるべき情報が抜け落ちることなどで精度に問題があった. そこで, 文ベクトルの生成を BERT に置き換え, 表のデータを学習させ表の精度向上を目指した. 実験をしたところ, 先行研究の F 値は 0.57 であったが, ファインチューニングを行った SentenceBERT では F 値が 0.48 と性能が下がる結果となった. 下がった原因として, 第一に学習に使用したデータが 8 個の表と少なく, 各表でドメインも異なっていた. このことにより, 文ベクトルの生成を悪化させるような学習となったと考えられる. 精度向上をするためには学習用のデータを増やす必要がある. 本研究の手法で性能を上げるにはドメイン特化にし, 学習データを増やす必要があると思われる.

謝辞

最後に本研究を行うにあたり, ご指導を賜った, 鳥取大学工学部知能情報工学科，自然言語処理研究室の村田真樹教授, 多くの助言を賜った村上仁一准教授, 同研究室学生に深く感謝致します. また, 先行研究を遂行してくださった健介岡崎氏, ならびに参考にさせていただいた論文の著者の方々に対して深く感謝申し上げます.