知能情報実験 III（データマイニング班）
word2vec による感情分析

215135K 玉城 浩輔
215402B 内藤 一
215721H Luong Van Bao
215760J Hyun Juwon

提出日：2023 年 8 月 14 日

目次

1　はじめに

1.1　概要

本実験は文章から書き手の感情を分析することである．つまりどのような感情がそのくらいの強さで文章に込められているかを調べたい．具体的な手法を説明する．追加学習した word2vec モデルを用いて文章をベクトルに変換する．その後，分類器 svm を使用して含まれる感情の強度を分析する．svm のハイパーパラメーターを調節して適切な値を見つけてモデルを作成した．

本実験での未解決問題は二つである．一つ目に word2vec モデルに対して追加学習を実行し特定の感情に強い単語の分散表現が得られるかを調査する．次に，分類器に使用する svm のハイパーパラメーターを調査し適切な評価方法で評価を行う．

本実験によって自然言語処理における文章の前処理の方法および word2vec モデルの追加学習や svm について理解することができた．

1.2　テーマについて

本実験のテーマは，文章から書き手の感情を分析することである．つまり文章に込められた感情とその強さを分析する．分析にはプルチックの感情の輪 [1] を使用する．

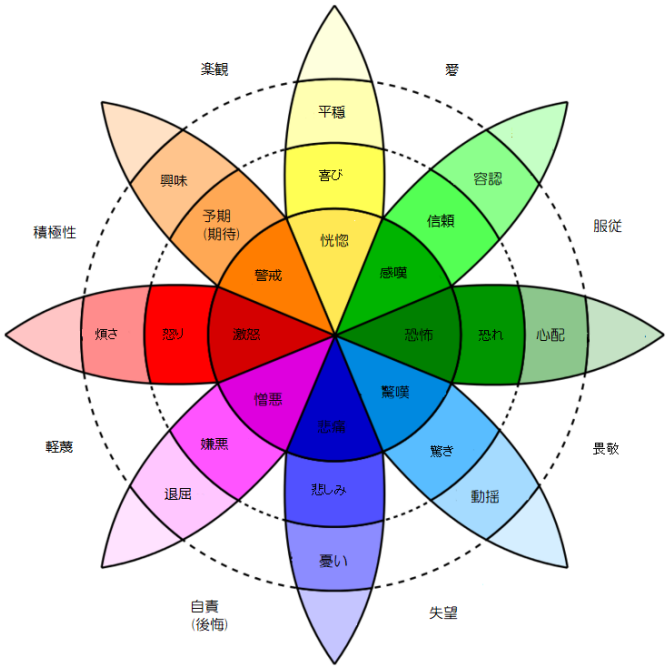

図 1　プルチックの感情の輪 [1]

上図 1 はロバート・プルチックが 1980 年に提唱した感情の輪である．8 つの基本的な感情，‘喜び‘，‘信頼‘，‘不安‘，‘驚き‘，‘悲しみ‘，‘嫌悪‘，‘怒り‘，‘期待‘を一次感情としてそこから強弱に派生し二次感情，三次感情の応用感情を定めて理論化した．

本実験ではこの一次感情の強さを 0: 無，1: 弱，2: 中，3: 強の強度で分析することを目的としている．つまり文章に含まれる一次感情の強さを 4 段階評価で示したい．通常，感情分析は単純にポジティブ，ネガティブ，中立の判定に留まるが，この実験では感情をより豊かな一次感情で評価する．感情は単にポジティブかネガティブかだけでなく，具体的な種類によっても評価されるべきであると考えたからである．したがって，ここでは 8 つの異なる感情状態についての評価を行う．この手法は，レビューやソーシャルメディアの投稿，顧客フィードバックなど，豊富なテキストデータから感情を多次元的に抽出し，その結果をビジネスや研究で活用する新たな道を開く可能性がある．今回は word2vec を使用して文章ベクトルを得て特徴量とする．次節で word2vec について説明する．

1.3 word2vec とは

Word2Vec は単語の分散表現を得ることでベクトル空間上に表現できる．この実験では感情の強さを表現する特徴量として使用される．単語をベクトル空間に表現することで，単語同士の意味を類似度の指標としてコサイン類似度を求めることができる．コサイン類似度とは，ベクトル空間上の 2 つのベクトルの類似度を計算するための指標の一つである．ベクトルの内積をベクトルのノルムの積で割ることで計算される．

ベクトル $\mathbf{A}$ とベクトル $\mathbf{B}$ のコサイン類似度を求めるとき以下のようになる．

$$\text{コサイン類似度} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|A\| \cdot \|B\|}$$

コサイン類似度は，-1 から 1 までの値を取る．類似した方向を向いているほど値は 1 に近くなる．Word2Vec では，単語をベクトル空間上に埋め込むことで，単語間の意味的な関係をベクトルの類似度として表現する．したがって，2 つの単語のベクトルを取得し，それらのベクトルのコサイン類似度を計算することで，単語間の意味的な類似度を推定することが可能である．

次に word2vec の利用方法を実際にコードで説明してみる．

初めに gensim ライブラリをインストールする．そして今回は学習済み word2vec モデルの chiVe[3] を使用する．

```
1 !pip install gensim
2 import gensim
3 # のモデルを選択するword2vecchiVe
4 vectors = gensim.models.KeyedVectors.load("./chiVe-1.2-mc5_gensim/chiVe-1.2-mc5
      .kv")
5 # 「嬉しい」に最も類似した単語（最大個）を検索する5
6 vectors.most_similar("嬉しい", topn=5)
```

Listing 1　word2vec を使って類似した単語を検索する

以上の Lstlisting1 は学習済み word2vec モデル chiVe を使用してコサイン類似度の高い，つまり意味的に類似している単語を上から 5 つ検索するコードである．結果は以下のようになった．

```
[('ウレシイ', 0.8493819832801819),
 ('喜ぶ', 0.7328100204467773),
 ('有り難い', 0.714138925075531),
 ('感激', 0.6744762659072876),
 ('(＊ˆ＿ˆ＊)', 0.667408279205322)]
```

Word2Vec を使用すると，単語の意味が類似しているかどうかを判別できる．具体的には，単語のベクトル表現を通じて，単語間の意味的な関連性を数値化し，それに 基づいて類似性を測定することが可能である．以上の特性を利用し文章中の各単語をベクトル化しそれらを合計したものを文章ベクトルとし特徴量として利用する．

2　実験方法

1. 実験目的
2. 実験計画
3. データセット構築
4. モデルの選定
5. 文章の前処理について
   （a）正規化について
   （b）ストップワードの除去について
   （c）分ち書きについて
6. パラメータ調整
   （a）word2vec の追加学習
   （b）文章ベクトルの生成方法
   （c）主成分分析について
   （d）SVM のハイパーパラメーター
7. 評価方法

2.1　実験目的

本研究の実験目的は，word2vec を活用し，‘喜び‘ ‘信頼‘ ‘不安‘ ‘驚き‘ ‘悲しみ‘ ‘嫌悪‘ ‘怒り‘ ‘期待‘ といった各種感情を 0〜3 の段階で評価するためのモデル，特にこれら各感情に特化したモデルを開発し，その有効性を検証することである．
本実験を通して，以下の点を明らかにしたい．

- 1. word2vec による単語のベクトル表現が感情表現にどの程度対応しているか.
- 2. 各感情に特化したモデルが一般的な感情分析と比較してどの程度の精度で感情を評価できるか.

これらの実験を通じて，多次元感情分析とそのための特化型モデルの開発の有効性と応用可能性を評価し，人間の感情表現の理解を深めることを目指している.

2.2 データセット構築

データセットには WRIME(`https://github.com/ids-cv/wrime`) を使用した．WRIME とは文章の一次感情 8 つ (‘喜び‘ ‘信頼‘ ‘不安‘ ‘驚き‘ ‘悲しみ‘ ‘嫌悪‘ ‘怒り‘ ‘期待‘) について主観（テキストの筆者 1 人）と客観（クラウドワーカ 3 人）の両方の立場から 4 段階（0: 無，1: 弱，2: 中，3: 強）の感情ラベルが付与されている．文章は 43200 件あり文章の平均文字数は 39 文字である.

2.3 モデル選定

学習済みモデルとして chiVe を選択した理由は，chiVe が日本語の自然言語処理タスクに特化して設計されているからである．chiVe は SudachiPy という日本語の形態素解析器によって得られた単語を元に訓練された Word2Vec モデルである．これにより，日本語の単語をより適切に理解できるベクトル表現を得ることが可能となる．chiVe は，日本語 Wikipedia と日本語ウェブコーパス（NWJC）という大規模な日本語コーパスを用いて訓練されている．これにより，広範かつ現代的な日本語表現の理解を反映した単語ベクトルが得られる．このため，本実験の目的である日本語テキストからの感情分析において，高い性能を発揮すると期待できる.

2.4　文章の前処理について

文章をベクトルかする前に単語に分割して必要のない単語を取り除いたり，表現を統一しなければならない．この作業を前処理と呼称する．前処理の手順は以下の図 2 の通りである．

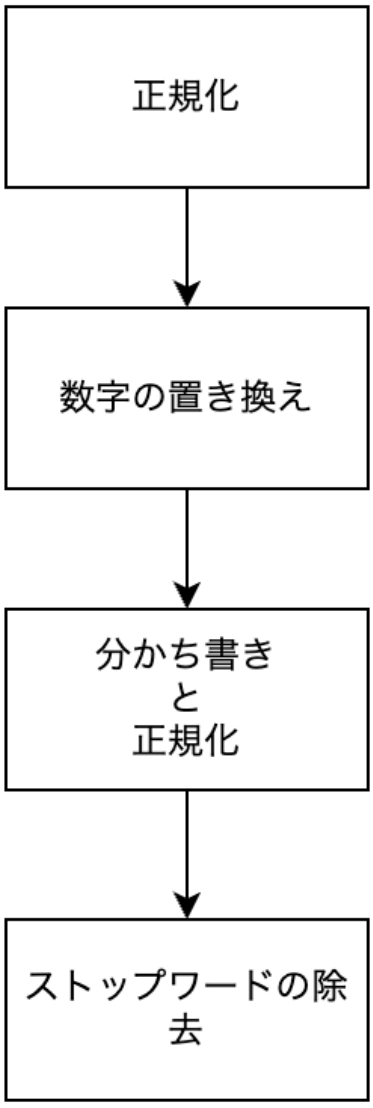

図 2　前処理のフローチャート

以下に各手順について詳しく説明する．

2.4.1 テキストの正規化

テキスト内の異なる表記や形式を統一し，綴りや表記ゆれの吸収といった単語を置き換える処理をした．この処理を行うことで，全角と半角とひらがなの単語を同じ単語として処理できるようになる．

具体的は今回 neologdn ライブラリを使用して，テキスト内の文字を正規化した．例えば全角英数字を半角に変換するなどの処理を行なった．

2.4.2 数字の置き換え

数値表現は多様で出現頻度が高く，今回のタスクには影響が少ないと考えたため,neologdn ライブラリで正規化されたテキストの小数点やカンマなどを削除し，正規表現で数字を 0 に置き換えた数字を 0 に置き換えした．

数字の置き換え後のリストのテキストデータを改行区切りで結合し，ファイルに書き込んだ．各テキストデータが改行で区切られた形式でファイルに保存するようにした．

2.4.3 Sudachi で分ち書きと NFKC 正規化 [8]

正規化や数字の置き換えを行ったファイルからテキストを読み込み，SudachiPy を使用して分ち書きを行った．この際，SudachiPy ではデフォルトで NFKC をつかった Unicode 正規化が行われている．

2.4.4 ストップワード除去

自然言語処理では出現回数が比較的多く，タスクに影響を与えない単語を削除する手法がある．今回は [7] のテキストファイルの中にある単語を削除対象にした．

2.5 パラメータ調整

精度向上を目指して以下の 3 つの要素のパラメータについて調整を行った．

2.5.1 word2vec の追加学習

gensim の Word2Vec モデルでは事前学習を引き継ぎつつ，ドメインに合わせたデータで追加学習を行うことができる．今回の実験では特定の感情が強い文章を追加学習させることにより，その感情に特化した単語の分散表現を得ることを目指した．‘特定の感情が強い文章 ‘としてデータセット [4] の中でその感情の強度が 1 以上の文章を用い，追加学習を行う際は chiVe のドキュメント [6] を参考にして進めた．

2.5.2　文章ベクトルの生成方法

文章ベクトルの生成は，文章中の全ての単語のベクトルを組み合わせることで作成した．具体的には，各単語のベクトルを合計して文章全体の感情ベクトルを作成した．この文章ベクトルを特徴量として svm を用いた教師あり学習によってラベルを分類するモデルを作成する．

2.6　主成分分析について

また，主成分分析について pca で成分分析を行なってところ以下図 3 のようになった．以上図 3

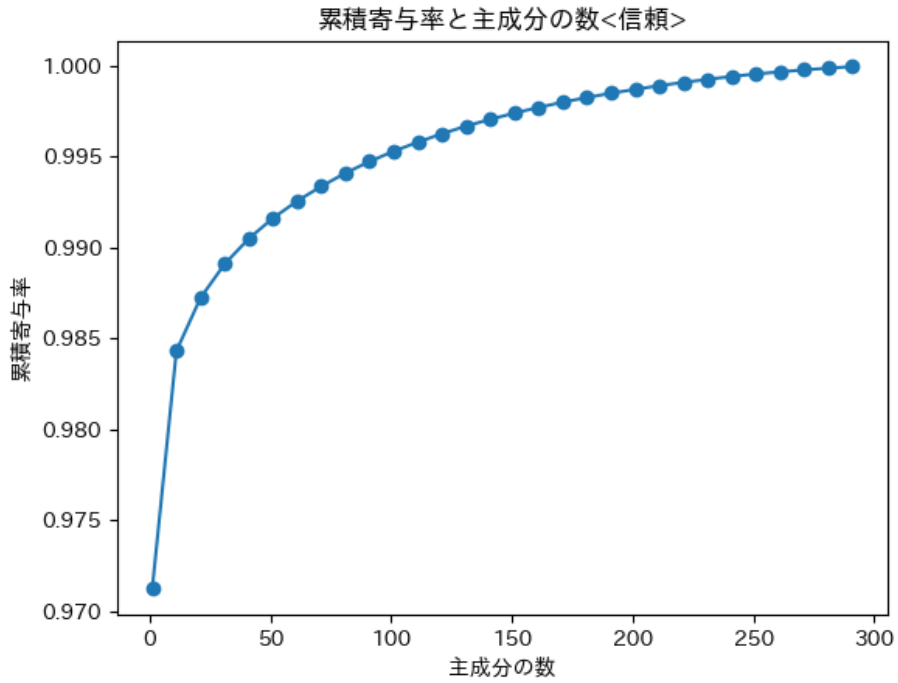

図 3

はエルボー法を用いて次元を削減した時の累積寄与率を調査した．しかし結果からもわかる通り 300 次元と 1 次元のベクトルの保っている情報の差がほとんどないことがわかる．主成分の次元削減を行わなかった．

2.6.1　SVM のハイパーパラメーター

SVM のハイパーパラメータは，ランダムサーチを用いて探索範囲を絞り込み，その後，その範囲内でグリッドサーチを行った．この二段階の探索によりパラメータ C とガンマパラメータ，カーネルの最適化を行った．

パラメータ C は，SVM の誤分類に対するペナルティを決定する．C の値が大きいと誤分類への

ペナルティが強くなり，訓練データに対してより厳密にフィットするモデルが生成される．逆に C の値が小さいと，モデルはより柔軟になる．

ガンマパラメータは，RBF カーネルなどで使用され，決定境界の形状に影響する．ガンマが大きい場合，決定境界はより複雑になり，小さい値ではより単純な境界となる．

カーネルは，非線形データを高次元空間に変換する関数である．RBF，多項式，シグモイドなど，異なるカーネル関数が存在し，各カーネルがデータを異なる形で変換を行う．カーネルの選択と調整は，モデルがデータの非線形特性をどのように捉えるかに直結している．

文章ベクトルとこれらのパラメータの関係においては，文章ベクトルは文章全体の意味や感情を捉えるための表現であり，SVM の入力として使用されます．C，ガンマ，カーネルといったハイパーパラメータは，この文章ベクトルをどのように解釈し，分離境界をどう形成するかを決定する．したがって，これらのハイパーパラメータの適切な調整により，文章ベクトルが反映する文章の感情や特性を，より精緻に分析できるモデルを構築が可能である．これらのパラメータ調節により，感情分析の精度を最大化するようなモデル設定をした．

具体的にランダムサーチとグリッドサーチについてコードで説明する．

```
from sklearn.model_selection import RandomizedSearchCV
from thundersvm import SVC
import numpy as np

# ハイパーパラメータの探索範囲
param_grid = {
    'C': [0.1,1,10],
    'gamma': [0.01, 0.1, 1],
    'kernel': ['linear','polynomial' , 'rbf']
}

# ランダムサーチの設定
random_search = RandomizedSearchCV(
    estimator=SVC(),
    param_distributions=param_grid,
    random_state=42,
    n_iter=20,
    verbose=2,
    n_jobs=-1
)

# ランダムサーチの実行
random_search.fit(train_data, train_labels)

# 最適なモデルとハイパーパラメータの取得
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Best␣Model:", best_model)
print("Best␣Parameters:", best_params)
print("Best␣Score:", random_search.best_score_)
print("Grid␣Scores:", random_search.cv_results_['mean_test_score'])
```

Listing 2　ランダムサーチのコード

```
from sklearn.model_selection import GridSearchCV
from thundersvm import SVC
import numpy as np
best_params = {'kernel': 'rbf', 'gamma': 0.1, 'C': 1}

# グリッドサーチの設定
param_grid = {
    'C': np.linspace(max(0.1, best_params['C']-0.3), best_params['C']+0.3, 10),
          # ベストパラメータの前後を探索
        0.5
    'gamma': np.logspace(np.log10(max(0.01, best_params['gamma']/2)), np.log10(
        best_params['gamma']*2), 10),# ベストパラメータのから倍の範囲を探索
        1/210
    'kernel': [best_params['kernel']]
}

# グリッドサーチの実行
grid_search = GridSearchCV(
    estimator=SVC(),
    param_grid=param_grid,
    scoring='accuracy',
    cv=15,
    n_jobs=-1
)

# グリッドサーチの実行
grid_search.fit(train_data, train_labels)

# 最適なモデルとハイパーパラメータの取得
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_

print("Best␣Model:", best_model)
print("Best␣Parameters:", best_params)
print("Best␣Score:", grid_search.best_score_)
```

Listing 3　グリッドサーチのコード

具体的にはランダムサーチで C:[0.1,1,10]，ガンマ [0.1,1,10]，カーネル [rbf,poly,sigmod] からベストな値を見つけその後グリッドサーチで C はベストな値の± 5 をガンマは± $\frac{1}{2}$ の範囲をベストなカーネルの元で探索した.

さらにグリッドサーチでは For integer/None inputs, if the estimator is a classifier and y is either binary or multiclass, StratifiedKFold is used.[5] より svm はマルチクラスの分類であるため交差検証時に各ラベルの割合を保ったまま分割を行う StratifiedKFold を使用して Lstlisting3 において 15 分割して StratifiedKFold を使用した.

2.7 評価方法

本実験の評価方法は，文章から感情強度を 0, 1, 2, 3 の 4 つのカテゴリーで分類することである．具体的には，開発したモデルを用いて文章から各感情を分析し，その感情強度を 0, 1, 2, 3 のいずれかに分類する．その分類結果とデータセットの既存ラベルとの一致度を計算し，正解率を求める．正解率が高いほど，開発したモデルの精度が高いと評価される．さらに各ラベルの正解率を混同行列によって評価を行った．これら評価方法を通じて，モデルの精度評価と改良を行い，感情分析の精度向上を目指す．

```
# テストデータでモデルを評価
test_score = best_model.score(test_data, test_labels)

print("Test␣Score:", test_score)

from sklearn.metrics import confusion_matrix

pred_labels = best_model.predict(test_data)

# 混同行列の作成
cm = confusion_matrix(test_labels, pred_labels)
print("Confusion␣Matrix:")
print(cm)
```

Listing 4　モデルの評価を行うコード

以上の Lstlisting4 は作成したモデルを事前に分割しておいたテストデータと比較することで評価を行っている．正解率と混同行列を求める．

3 実験結果

各感情について，ベストなハイパーパラメーターを求めてそれに対するテストデータのラベルとの正解率を求めた．さらに各ラベルについての制度も見るために混同行列も求めた．

3.1 喜び

C	2.06
gamma	0.020
kernel	rbf
正解率	61.0%

図 4

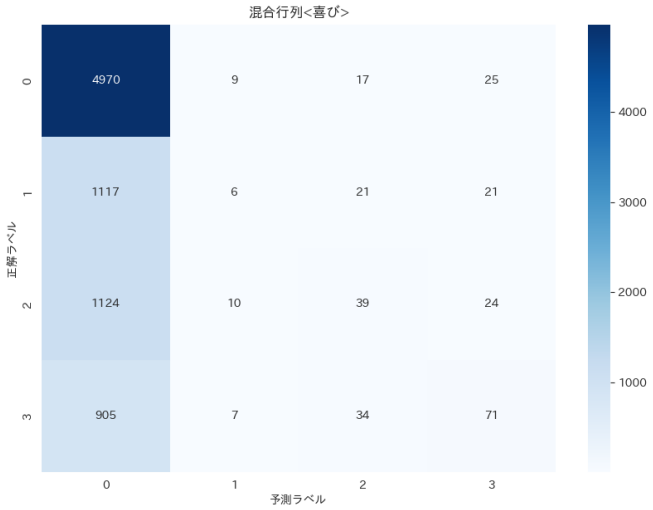

図 5

3.2　信頼

C	1.70
gamma	0.058
kernel	rbf
正解率	78.6%

図 6

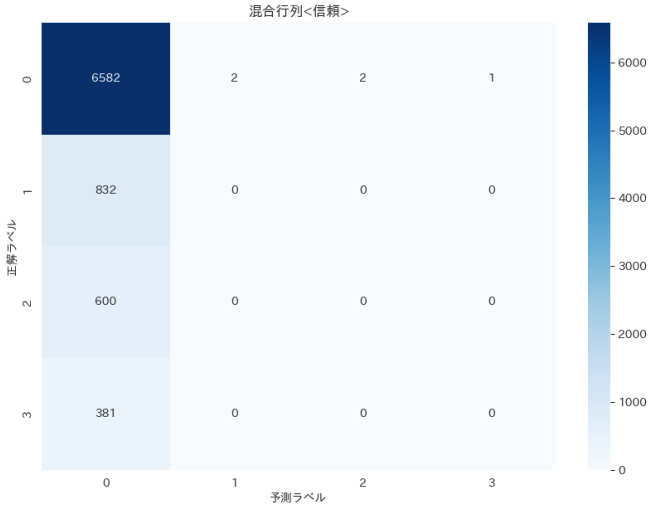

図 7

3.3　不安

C	1.70
gamma	0.185
kernel	rbf
正解率	78.0%

図 8

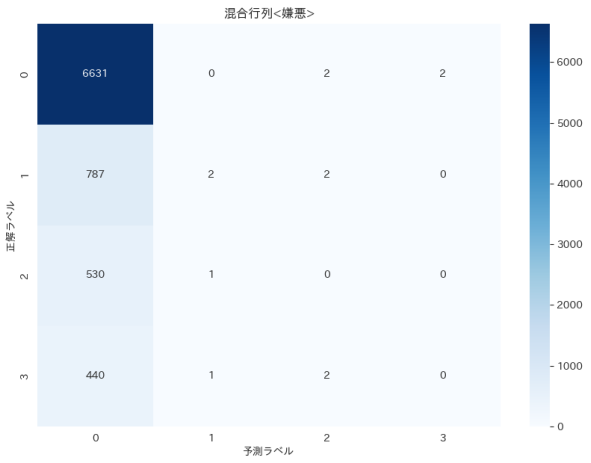

図 9

3.4　驚き

C	3.70
gamma	0.299
kernel	rbf
正解率	71.7%

図 10

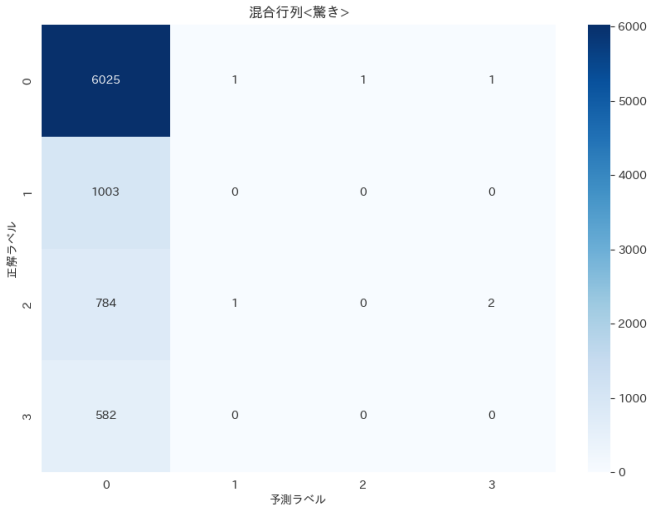

図 11

3.5　悲しみ

C	3.70
gamma	0.345
kernel	rbf
正解率	65.4%

図 12

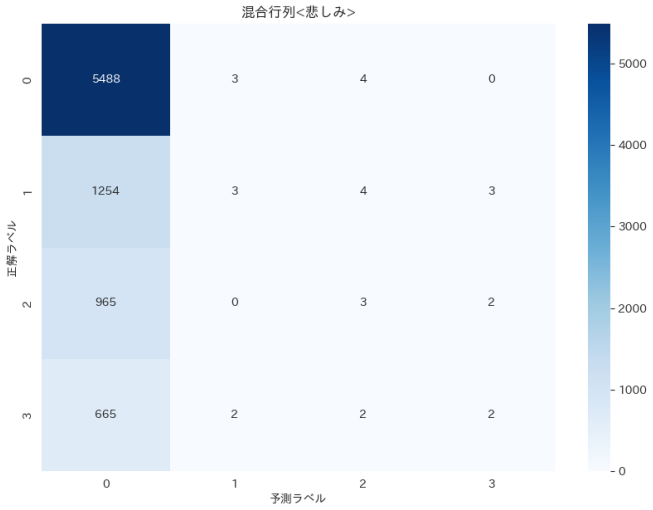

図 13

3.6　恐怖

C	1.70
gamma	0.185
kernel	rbf
正解率	79%

図 14

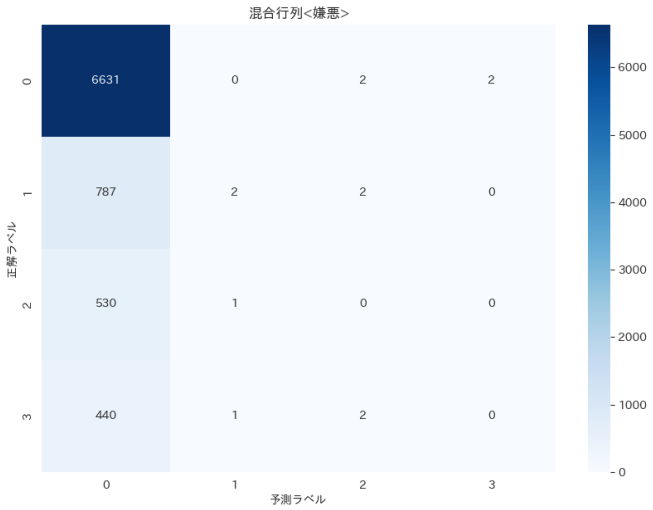

図 15

3.7　怒り

C	1.23
gamma	0.08
kernel	rbf
正解率	78.3 %

図 16

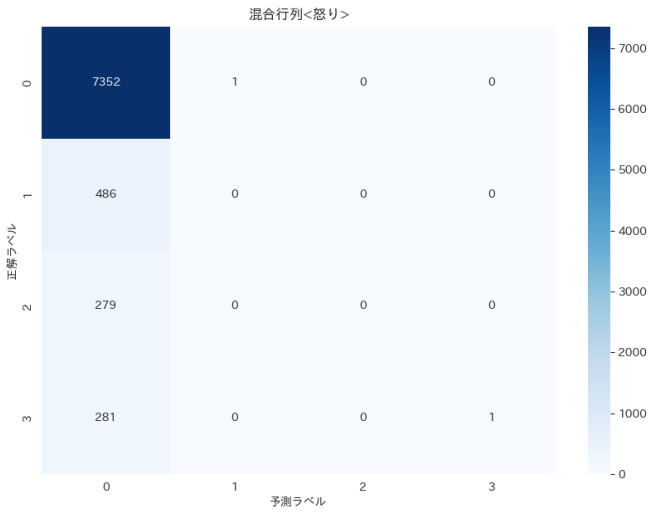

図 17

3.8 期待

C	1.03
gamma	0.05
kernel	rbf
正解率	62.4%

図 18

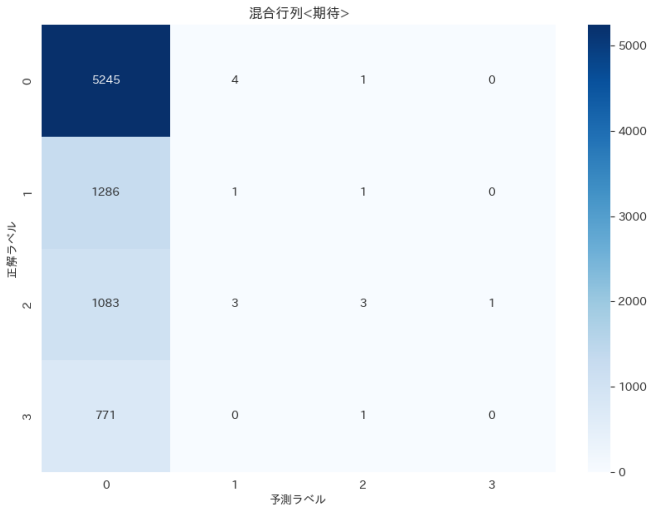

図 19

以上が 8 感情分の svm の適切なハイパーパラメーターと混同行列である．混同行列に着目すると，いずれもほとんどをラベル 0 つまり感情強度：無であると分類していることがわかる．しかしラベル 0 のサンプル数が他のラベルに比べて多いため見かけの正解率は高くなっている．

顕著な例として，怒りの結果について，図 16 と図 17 の混同行列を見てみる．正解率は 78% と高いが，混同行列を見ると 2 サンプルを除いて全てをラベル 0 に分類している．

4 考察

4.1 正解ラベルの偏りについて

結果より，いずれの感情に対応したモデルも正解ラベル 0 に分類してしまっている．これは訓練データ (データセットの 8 割 [4]) の正解ラベルの偏りが原因と考えた．図 5 の喜びの混同行列が他の感情に比べてラベルの偏りが小さいと共に，各ラベルの正解率も高かったからである．

よって喜びの感情の訓練データを最もサンプル数の少ないラベル 3 の数に揃えた上で学習を行った．

C	0.10
gamma	2.00
kernel	rbf
正解率	27.3%

図 20

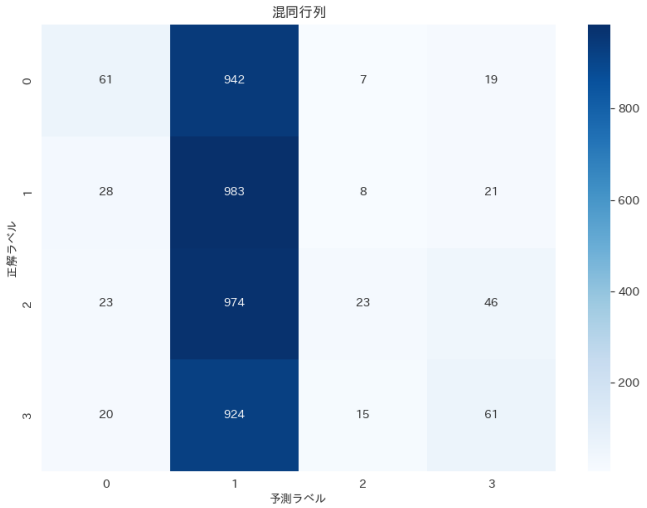

図 21

以上図 20 と図 21 の混同行列が正解ラベルの偏りを無くした結果である．正解率を見ると 27.3% に低下している．そして混同行列よりほとんどのラベルを 1 つまり感情強度：弱に分類していることがわかる．

以上の結果より，本実験で作成したモデルには特徴量が不足していたと考える．

4.2 追加学習について

追加学習を行うことで特定の感情に強い分散表現を持った word2vec モデル (モデル A と呼称) を作成できたのか考えるために，特定の感情ではなく全ての感情で追加学習を行った word2vec モデル (モデル B と呼称) で学習を行った．

次にそれぞれの混同行列を比較する．

ラベル	モデル A の時の正解率 [%]	モデル B の時の正解率 [%]
ラベル 0	98.9	98.9
ラベル 1	0.51	0.25
ラベル 2	3.26	3.59
ラベル 3	6.98	6.19

表 1

それぞれの混同行列より，それぞれのラベルでの正解率を比較すると，ラベル 2 を除いていずれのラベルも全ての感情において追加学習したモデルの方が正解率が高くなった．

以上の結果より，特定の感情に強い分散表現を持つ word2vec モデルを得ることはできなかったと考える．

C	1.50
gamma	0.002
kernel	rbf
正解率	60.4%

図 22

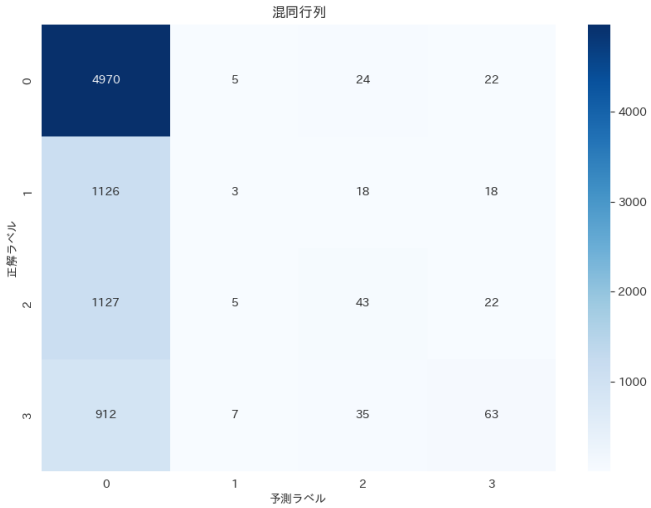

図 23

5　意図していた実験計画との違い

基本的に予定通りに実行できた．word2vec 班と分類器班に別れていたために，互いの進捗と問題点をこまめに共有することに務めた．また，時間を節約するために学習をローカルではなくサーバー上で実行するようにした．

実験の結果については，それぞれのラベルの正解率 65% を目指していたが．達成できなかった．これは実験の評価方法について，混同行列の使用の検討が実験の終盤になってしまったため，実際には精度が得られていないことに気づくのに遅れてしまったことが原因であると考える．これは前処理や特徴量，ハイパーパラメーターの調整を全て同じ感情ラベル (喜び) で行なってしまっていたことが原因である．8 感情のうち同じ処理でも 2 ないし 3 つの感情ラベルを用いて検討すべきであったと考える．

6　まとめ

今回の実験に取り組むにあたってのきっかけは自然言語処理への興味である．そのため理解しやすく，アルゴリズムが他の自然言語処理のモデルよりも比較的簡単な word2vec を使用し word2vec を使用し，同時に機械学習について学ぶために分類器を使用した．そのため，word2vec の使い方や分類器の使用方法，ハイパーパラメーターの探索などを理解することができた．しかし，目標であった正解率のモデルを作成することはできなかった．

今回の実験のタスクは人間でも難易度が高い分類であるため，多値分類ではなく二値分類にするなど，少し難易度の低い分類問題に置き換えたほうが良い結果が見込めたかもしれないと考える．

参考文献

[1] Plutchik’s Wheel of Emotions: Exploring the Emotion Wheel, https://www.6seconds.org/2022/03/13/plutchik-wheel-emotions/,2023/07/31

[2] 自然言語処理の必須知識！ Word2Vec とは？, https://blog.kikagaku.co.jp/word2vec,2023/06/05.

[3] chiVe,https://github.com/WorksApplications/chiVe,2023/06/08.

[4] wrime,https://github.com/ids-cv/wrime,2023/06/08.

[5] sklearn.model_selection.GridSearchCV, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html,2023/07/25.

[6] chiVe の追加学習,https://github.com/WorksApplications/chiVe/blob/master/docs/continue-training.md,2023/08/13.

[7] ストップワード,http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt,2023/08/13.

[8] SudachiPy,https://github.com/WorksApplications/SudachiPy,2023/08/14.