文書分類タスクにおけるディープラーニングの学習プロセス可視化手法の提案

Visualizing Learning Process of Convolutional Neural Networks for Sentence Classification

井上　達郎†　　佐藤　哲司††

† 筑波大学情報学群知識情報・図書館学類　〒 305-8550 茨城県つくば市春日 1-2
†† 筑波大学図書館情報メディア研究科　〒 305-8550 茨城県つくば市春日 1-2
E-mail: †{inoue,satoh}@ce.slis.tsukuba.ac.jp

あらまし　近年, 深層学習において, モデルの出力の根拠を説明しようという研究が盛んに行われている. これまでは, 学習済みのモデルが, 入力のどの部位に着目して出力しているか可視化する研究が多く行われている.　本研究では, 畳み込みニューラルネットワークを用いた文書分類タスクにおいて, モデルの学習過程での入力に対する注目度変化を分析する手法を提案する. これにより, 学習による注目度が不安定な単語, すなわち, 誤分類の原因と考えられる単語を特定し, より少ない学習データセットで高い精度を達成することを目指す.　実験の結果, 注目度変化の不安定な単語を多く含むデータを使って学習したモデルが, より高い精度を達成した.

キーワード　ディープラーニング, 可視化, テキスト分類, 畳み込みニューラルネットワーク, 学習プロセス

1. はじめに

1.1 研究背景

深層学習と呼ばれる機械学習の手法が, 画像分類やテキスト分類などの様々な分類タスクにおいて高いパフォーマンスを発揮している. 例えば, 畳み込みニューラルネットワークを用いたテキスト分類では, シンプルなモデルで高い精度を達成している [1].

しかし, 入力から限られた特徴量を抽出し, それを用いて分類を行うこれまでの手法とは異なり, 深層学習は, 大量の入力を直接学習し, その上で分類を行う手法である. このため, 高い精度で分類を行うことができても, なぜそのように分類したのかや, なぜ分類を間違えたかといった要因を分析するのは困難である. 実運用において, 分類の根拠を示したり, 誤分類の原因を解析し, モデルの性能を改善することは, 非常に重要である. このため, モデルがどのように分類を行っているか分析することは, 重要な研究課題となっている.

このような背景から, 学習済みのモデルが入力のどの部位に着目しているか可視化することにより, モデルの出力の根拠を説明しようという研究が盛んに行われている. しかし, 学習過程でどのようにモデルが, 入力の注目部位を変化させたか分析する研究は, 著者の知る限り行われていない.

本研究では, 学習過程でモデルがどのように入力の注目度を変化させたかに着目する. 具体的には, モデルが学習によって入力の注目部位をどのように変化させたか分析し, 正しく分類されたデータセットと, 誤って分類されたデータセットの違いを特定する. これにより, 誤分類の原因を特定するとともに, その情報を利用して, よりモデルの性能を向上させることのできる学習データの特徴を明らかにすることを目指す.

本論文の構成は次の通りである. まず第 2 章で, 関連研究について概観し, 本研究の位置付けを明らかにする. 第 3 章では, 単語注目度の算出手法について説明する. 第 4 章で, 提案手法について詳細を述べる. 第 5 章では, 提案手法による実験の結果を示すとともに, 結果を踏まえた考察を述べる. 第 6 章で本論文のまとめ, 今後の課題について述べる.

2. 関連研究

深層学習では, モデルがなぜその出力をしたのかわからないという問題がある. 本章では, モデルの動作の可視化に関連した研究を, 学習済みのモデルに注目した研究と, モデルの学習過程に注目した研究に分けて概観し, 本研究の立ち位置を明らかにする.

2.1 学習済みモデルに着目した研究

2.1.1 出力を入力方向にたどる

出力から入力方向へと逆に辿ることで, 出力に貢献した入力部位を特定する手法が提案されている. 例えば, Jost ら [2] は画像の分類タスクにおいて, 逆伝搬時に制約を設けることで, 出力に及ぼす影響が高い入力部位を可視化する手法を提案している. この時, ある層の出力のうち, 調べたい出力箇所以外を 0 とする制約を設け, 逆伝搬する. また, Activation を減衰させている値を除去するため, 負の値となっている箇所を 0 として伝搬・逆伝搬を行う. これにより, 入力において重要なピクセルを可視化することに成功した.

一方, Bach ら [3] は, ある入力部位の出力に対する貢献度を,

各層の出力と重みをもとに算出する手法を提案している．この手法を用いて, 画像分類タスクにおいて, 出力に寄与したピクセルをヒートマップで可視化できることを示している．

Leila ら [4] は, [3] と同様の手法をテキスト分類に適用することで, 個々の単語がモデルの分類決定にどの程度貢献したか可視化できることを示している．

2.1.2 入力を変形させて出力根拠を探る

Marco ら [5] は, もともとの入力を幾つかのパターンに変形させ, それぞれの出力を見ることで, モデルが入力中で重要視している特徴を発見するという手法を提案している．入力である画像を, パーツごとに切り分けて, 幾つかのパターンを作成する．その後, パターンごとの出力結果を, 学習済みのモデルを用いて得る．そうしてできたそれぞれのパーツごとの入力, 出力のペアを, 本体のモデルとは別に用意した単純な分類モデルの学習データとして使用し, 学習させる．その後, 学習させた単純なモデルの偏回帰係数から特徴量の寄与度を表示する．この手法により, テキスト分類や画像分類において, モデルが入力のどの部位に着目しているかを可視化できることを示している．

2.1.3 入力に対する感度を分析する

Li ら [6] は, モデルがラベル y を出力した際に, その確率推定値に大きな影響を与えた入力 (単語) を特定する手法を提案している．単語ベクトルの値を微小に変化させた時に, モデルの確率推定値に与える影響度を, 誤差逆伝搬法を用いて損失関数の勾配値を入力まで逆伝搬させることにより算出した．同様の手法を用いて Daniel ら [7] は, 画像分類タスクにおいて, 出力に貢献している入力部位を可視化した．

2.2 モデルの学習過程に着目した研究

Chung ら [8] は, CNN のトレーニングプロセスの解釈性を改善するため, 各層の重みの変化量や活性化率が, 学習過程でどのように変化したか可視化する手法を提案している．

また, Google の研究チームが発表した TensorBoard と呼ばれるサービスでは, 学習過程での各層の重みやバイアスの変化に加えて, 計算時間やメモリ使用率, モデルの構造といった様々な情報を可視化することができる．

これらの手法では, モデルが重みやバイアスをどのように変化させたか可視化することで, モデルの性能向上に役立つ情報を抽出しようとしている．しかし, モデルが入力に対する注目部位をどのように変化させたか分析することで, モデルの性能を向上させようとする試みは行われていない．

2.3 本研究の位置付け

本研究では, Li らの手法と同様に, 誤差逆伝搬法による勾配値を用いて, モデルの出力に貢献している入力部位, すなわち, モデルがどの単語に注目して分類を行っているか, その注目度を算出する．この注目度を用いて, 学習過程でどのようにモデルが入力に対する注目部位を変化させたか可視化し, 誤分類の原因を分析する．また, 可視化結果を踏まえ, 誤分類の原因となるような単語を特定することで, モデルの性能をより向上させる学習データを特定することを目指す．

3. 単語注目度の算出手法

本章では, テキスト分類タスクを対象に, モデルが出力時にどの単語に注目しているかを示す度合い (以下, 単語注目度) を算出する手法を説明する．

3.1 ニューラルネットワークの学習

単語注目度算出手法を説明する前に, ニューラルネットワークの学習の仕組みについて, 簡単に説明する．

ニューラルネットワークでは損失関数の勾配を計算し, 勾配値を手がかりにパラメータを更新する．損失関数とは, ニューラルネットワークが教師データに対してどれだけ適合しているか算出する関数である．ニューラルネットワークでは, 損失関数の値を 0 に近づけるように, 重みを調整することで学習を行う．

例えば, 図 1 のような構造を考える．この時, w_1^1 の重みを更新する場合, 損失関数 L に対する重み w_1^1 の勾配値を利用して, 以下のように更新する．

$$w_1^1 = w_1^1 - \eta \frac{\partial L}{\partial w_1^1} \tag{1}$$

ここで, w_1^1 は, 1 図で表されている, 第 1 層の 1 番目の重みを表す．また, η は, 学習率を表し, 正の数で表される．$\frac{\partial L}{\partial w_1^1}$ は, 損失関数 L に対する重み w_1^1 の勾配値を表し, 誤差逆伝搬法により算出される．

本研究では, これと同様に, 誤差逆伝搬法を利用して, 入力 x の損失関数に対する勾配値を算出することで, 単語注目度を算出する．ここで定義される単語注目度とは, 入力 x を微小に変化させた時に, 出力に及ぼす影響度のことを指す．

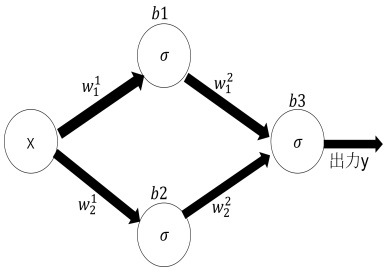

図 1 簡易なニューラルネットワーク構造

3.2 勾配を利用した単語注目度算出手法

本節では, 本研究で用いる注目度の算出手法を説明する．

3.1 節で述べたように, 入力 x の損失関数に対する勾配値を算出することで, 各単語がモデルの出力に及ぼす影響の大きさを, 単語注目度として算出する．

ある入力文 $\boldsymbol{S}$ を, モデル f に入力して, 分類結果 $\boldsymbol{R}$ を得る場合を考える．入力文 $\boldsymbol{S}$ は, 以下のように, 単語列 $\boldsymbol{w}_0$ から $\boldsymbol{w}_i$ で構成される．

$$\boldsymbol{S} = (\boldsymbol{w}_0, \boldsymbol{w}_1, ..., \boldsymbol{w}_i) \tag{2}$$

また, 単語 $\boldsymbol{w}_i$ は, 以下のように多次元ベクトルで表現される．

$$\boldsymbol{w_i} = (d_0, d_1, ..., d_j) \tag{3}$$

この時, 入力文 $\boldsymbol{S}$ 中のある単語ベクトル $\boldsymbol{w}_i$ の j 次元目 $\boldsymbol{w}_{i,j}$ の注目度は, 以下の式で表される.

$$Grad_S(i,j) = \left| \frac{\partial L(f(\boldsymbol{S}), R)}{\partial w_{i,j}} \right| \tag{4}$$

ここで, $L(f(\boldsymbol{S}), R)$ は, 損失関数に出力層からの出力である, 各ラベルの確率推定値と, 確率推定値を最大とする出力ラベルを渡した時の損失を表す. 式 (4) を, より直感的に解釈するため, 損失関数 L を一次までテイラー展開する.

$$L(R, f(\boldsymbol{w}_0, ..., \boldsymbol{w}_i)) \approx \quad L'(\boldsymbol{w}_{i,j})\delta k + L(\boldsymbol{w}_{i,j}) \tag{5}$$

(5) 式より, 本研究における単語注目度は, 単語ベクトルの値を微小に (δk) 変化させる時, 各ラベルの確率推定値に大きな影響を与えるものを求めていると言える.

ここで, 具体的に, 単語数 i，各単語の次元数 j の文章 $\boldsymbol{S}$ が与えられた時の単語 $\boldsymbol{w}_i$ の注目度, wa_i の計算手順について説明する．はじめに，誤差逆伝搬法を用いて, (4) 式を計算し, 文 $\boldsymbol{S}$ を構成する各単語について, 各次元のの注目度 $\boldsymbol{wa}_{i,j}$ を求める. 次に, 以下のように各次元の単語注目度の平均値を求めることで, 単語 $\boldsymbol{w}_i$ の注目度 wa_i 算出する.

$$wa_i = \frac{1}{j} \sum_{j=1}^{n} Grad_S(i,j) \tag{6}$$

4. 学習プロセス可視化手法の提案

4.1 概　　要

本研究では, 学習過程の単語の注目度変化を分析することで, 誤分類の原因となっている単語を特定することを目指す. 具体的には, 学習過程で注目度が大きく変化している単語を, 学習がうまくいっていない単語, すなわち, 誤分類の原因となっている単語であると仮定する. このような, 注目度変化の大きい単語を特定することにより, モデルの性能向上に大きく寄与する, 質の高いデータセットの特徴を発見することが, 本研究の目的である.

提案手法では, 前半学習用データセットによる学習において, 注目度変化量の多い単語を特定する. その後, 後半学習用データセットを. 注目度変化の多い単語を多く含むデータセットと, 含まないデータセットに分割する. 後半学習では, それぞれのデータセットを用いた場合に, どのようにモデルの性能が変化するか分析し, 提案手法の有効性を検証する.

4.2 学習データの分割

本研究では, データセットを以下のように分割する.

- 前半学習用データセット (以下, T1)
- 後半学習用データセット (以下, T2)
- 評価用データセット (以下, T3)

なお, 分割の割合は, それぞれ「4.5:4.5:1」の割合で分割する. 本研究では, T1 による学習を前半学習フェーズ, T2 による学習を後半学習フェーズとして学習を行う.

4.3 前半学習フェーズ

4.3.1 単語注目度変化の記録

前半学習フェーズでは, T1 を用いてモデルの学習を行いながら, 1 epoch ごとに学習中のモデルを使って T1 を分類し, 分類時の単語注目度を記録する. なお, 単語注目度の算出は, 3. 章で述べた手法を用いる. この時, 損失関数に与える入力は, モデルの出力層である, 各ラベルの確率推定値と, 確率値が最大となっているラベルである. これにより, 単語ごとに, モデルの出力への影響度を求める.

4.3.2 単語ごとの注目度変化量

注目度変化の激しい単語を見つけるため, 分散を利用して単語ごとに注目度変化量を定義する. 例えば, w という単語の注目度変化量 wac を定義する場合を考える.

T1 内で w が v 回出現しており, epoch 数が k 回の時, w の注目度を格納する配列 $waarray$ は, 以下のような 2 次元配列で表せる.

$$waarray = (wa_{v,k}) \tag{7}$$

本研究では, epoch の進行に伴って, 注目度が上がり続けている単語よりも, 上がったり下がったりしている単語の注目度変化量を高くする. このため, epoch ごとの差分を求めた配列を以下のように定義する.

$$wadarray_{v,k} = (wa_{v,k+1} - wa_{v,k}) \tag{8}$$

ここで, $(wa_{v,k+1} - wa_{v,k})$ は, 単語 w_v における, k+1 回目の epoch における単語注目度と, k 回目の epoch における単語注目度を減算することを表す. この時, wac は以下のように, $wadarray_v$ の分散値の平均により算出される.

$$wac = \frac{1}{v} \sum_{i=1}^{v} V(wadarray_i) \tag{9}$$

ここで, $V(wadarray_i)$ は, $wadarray_i$ の分散, すなわち, T1 内で i 番目に出現した単語 w_i における, 各 epoch の注目度差分の分散を表す. そして, T1 内での単語 w の分散の平均値を求めることで, 単語 w の注目度変化量 wac が定義される.

$waarray$ ではなく, $wadarray$ の分散を利用するのは, 線形な変化をしている場合の注目度変化量を小さくし, 非線形に変化している場合の注目度変化量を大きくするためである. $wadarray$ では, epoch ごとの注目度差分を求めているため, 線形に変化している場合の分散値は小さくなる. 一方, 非線形に変化している場合, 注目度差分の値が一定にならないため, 分散値が大きくなる. これにより, より不安定な注目度変化をしている単語を特定することを目指す.

4.3.3 後半学習データセットの分割

後半学習データセットは, 4.3.2 節で求めた単語ごとの注目度変化量を利用して, 注目度変化量が大きい単語で構成された

データセット $T2_h$ と, 注目度変化量が小さい単語で構成されたデータセット $T2_l$ に分割して学習を行う.

具体的に, T2 のあるレビュー文 S を, どちらのデータセットに割り振るか考える. $\boldsymbol{S}$ は, 単語列 w_1 から w_n までの単語列で構成されている. この時, S の注目度変化具合 A_S を以下のように, 各単語の注目度変化量の平均値として定義する.

$$A_S = \frac{1}{n}\sum_{i=1}^{n} wac_i \tag{10}$$

これを T2 中のすべてのレビュー文で算出する. これにより, T2 のそれぞれのレビュー文について, 注目度変化量が定義される. この時, あるレビュー文 S の注目度変化量 A_S が, T2 における注目度変化量の中央値より上ならば, $T2_h$, 下ならば, $T2_l$ に割り振る. 中央値を用いて分割するため, $T2_h$ と $T2_l$ のデータセットの量はそれぞれ等しくなる.

4.4 後半学習フェーズ

後半学習フェーズでは, それぞれ以下の組み合わせで学習を行う.

- $T2_h$: 前半学習の重みを引き継いで, $T2_h$ を用いてモデルを構築する
- $T2_l$: 前半学習の重みを引き継いで, $T2_l$ を用いてモデルを構築する
- $T1 and T2_h$: 前半学習の重みを引き継がず, T1 と $T2_h$ を用いて, モデルを構築する
- $T1 and T2_h$: 前半学習の重みを引き継がず, T1 と $T2_l$ ををを用いて, モデルを構築する
- $T1 and T2$: 前半学習の重みを引き継がず, T1 と T2 を用いて, モデルを構築する
- $T1 and T2_{ran}$: 前半学習の重みを引き継がず, T1 と, T2 からランダムに $T2_h$ ないし $T2_l$ と同数のデータを抽出したものを用いて, モデルを構築する.

後半学習を $T2$ を分割したデータだけで行う場合は, T1 による前半学習で設定された重みを引き継いで学習を行った. これは, T1 で学習した結果を引き継ぎつつ, 注目度が不安定な単語を, T2 による学習で安定させることを期待したためである.

一方, T1 と T2 を分割したデータを組み合わせて学習する場合は, T1 による学習の重みを引き継がずにモデルを構築した. これは, T1 による学習の重みを引き継ぐと, T1 を 2 回学習することになり, T1 にモデルが適合しすぎるのを避けるためである.

また本研究では, T1 と T2 を組み合わせて (学習データを全て利用して) 学習した場合のモデルも構築する. この場合は, 先ほどの場合と同様の理由で, T1 による学習の重みを引き継がずにモデルを構築した. 学習モデルを全て利用した場合のモデルの性能とも比較することで, 提案手法によって単語注目度変化を特定し, 注目度変化の大きい単語を多く含むデータセットを利用することの有効性を検証する.

4.5 評価フェーズ

評価フェーズでは, 4.4 節で述べた手法で構築したモデルを用いて, 評価セットを分類し, その結果を比較することで, 提案手法の有効性を検証する. なお, 比較は, モデルが評価セットをどれだけ正しく分類できたか (精度) を指標として用いる. また, 評価では, 学習済みモデルの精度だけでなく, 学習過程での精度の変化も比較する.

5. 実験・考察

本章では, 提案手法による実験の環境について述べたのち, 各実験の結果を示す. その後, 実験の結果を踏まえた考察を示す.

5.1 実験環境

実験は, 以下の環境で行った. なお, データセットの規模は, ポジティブラベル 5331 文, ネガティブラベル 5331 文, 計 10662 文である.

- 実行言語 : Python 3.5.4
- フレームワーク : Chainer 2.0.2
- データセット : 映画レビューデータセット (注1)

また本研究では, kim ら [1] の手法に基づいて, 畳み込みニューラルネットワークで, 入力となるレビュー文が, ポジティブな文かネガティブな文かに分類するタスクを設定し, 提案手法を実装した. 具体的には, 「This movie is good」というような肯定的な文書には positive のラベルを, 「This movie is bad」というような否定的な文書には negative のラベルを出力するような, 2 値分類のタスクである. なお, データセットとして利用するのは, kim ら [1] の研究でも使用されている, Cornell 大学が公開している映画のレビューデータセットを利用する. このレビューデータセットは, すべてのレビュー文に positive か negative かというラベルが付いているため, これを正解ラベルとして利用する.

5.2 データセットの分割

T1, $T2_h$, $T2_l$, T3 の各データセットの文量を, 表 1 に示す. 4.2 節で示したように, 全データセット 10662 文を, 「4.5:4.5:1」に T1, T2, T3 にそれぞれ分割する。その後, T1 学習の結果, 単語注目度変化量の大きいデータセットを $T2_h$, 小さいデータセットを $T2_l$ に, それぞれ「1:1」の割合で T2 を分割する.

表 1 各データセットの文量

T1	$T2_h$	$T2_l$	T3
4797	2399	2399	1067

5.3 構築した CNN モデル

本研究では, 図 2 のように, シンプルな畳み込みニューラルネットワークによって, 与えられた文書をポジティブかネガティブに分類するモデルを構築する.

本モデルでは, 以下のようにして, 入力がどちらのラベルか出力する.

（1） 入力文を, Google が公開している, 訓練済みの word2vec モデル (GoogleNews-vectors-negative300.bin) を利用してベクトル表現にする. これは, 単語ごとに 1 意に決まっている数字列によって表される.

(注1) : https://www.cs.cornell.edu/people/pabo/movie-review-data/

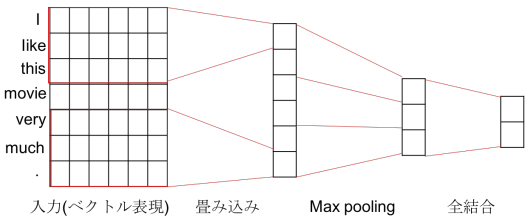

図 2　構築した CNN モデル

（2） 単語ベクトルの列で表現された入力に対して, 幅を単語ベクトルの次元数, 高さを 3 としたフィルタによる畳み込み処理を行う. 畳み込み処理により, 入力に対して 3 単語ずつのフィルターがかかり, 非線形の活性化関数を通る. この結果, 特徴ベクトル $\boldsymbol{c}$ を得る. 特徴ベクトル $\boldsymbol{c}$ は, それぞれのフィルターによる畳み込み処理によって得られる, 複数のベクトルによって構成される.

（3） 畳み込み層の出力である, 特徴ベクトル $\boldsymbol{c}$ を構成する各ベクトルに対して, それぞれ最大プーリング処理を行う. これは, 各ベクトルにおいて最大のものを取る処理である. これにより, プーリング層の出力 $\boldsymbol{p}$ を得る.

（4） 最後に, 出力層で, $\boldsymbol{p}$ を受け取り, Soft max 関数による各ラベル確率推定値の出力を行う. これにより, $[0.7, 0.3]$ のような各ラベルの確率推定値を得る, この場合, ラベル 0 の確率が 0.7, ラベル 1 の確率が 0.3 であるため, モデルの出力はラベル 1 となる.

5.4 実 験 結 果

5.5 前半学習の可視化結果

本節では, T1 を用いた前半学習について, 正しく分類されたデータセット (以下, 正分類), 誤って分類されたデータセット (以下, 誤分類) の学習過程を比較して可視化した結果を示す.

5.5.1 単語注目度変化の可視化

正分類データと誤分類データから, それぞれ抽出し図 3, 図 4 に可視化する. これらの図は, データを構成する各単語が, どのように学習過程で注目度を変化させたか可視化している. なお, 横軸は epoch 数, 縦軸はその epoch 数での単語注目度を表す. また, ここで可視化したデータは, 各 epoch ごとに比較できるように, epoch ごとに各単語の注目度を標準化したデータを可視化している. 図中の各グラフは, 抜き出したデータセット中で使われている各単語の注目度変化を表す.

図 3, 図 4 から, 誤分類と正分類の結果を比較する. 0 から 20 epoch をみると, どちらのグラフでも注目度が大きく変化していることがわかる. しかし, 20 epoch 以降を見てみると, 正分類データは注目度があまり変化していないのに対して, 誤分類データは注目度が細かく変化している. すなわち, 正分類データは比較的早い段階で注目度が収束しているのに対して, 誤分類データは注目度が学習の後半に入っても収束していないと言える.

また, 正分類データの可視化結果を見ると, 各単語の注目度変化は, 概ね線形である. 一方, 誤分類データの可視化結果を見ると, 注目度が上がったり下がったりして, 非線形に変化している

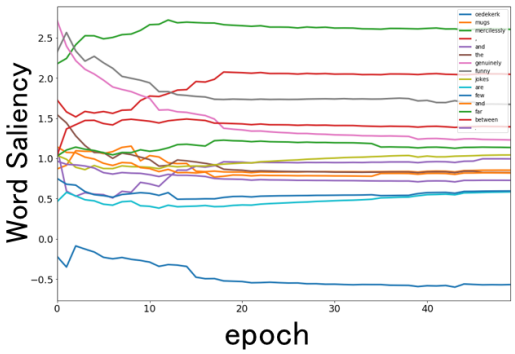

図 3　正分類データの学習記録可視化結果

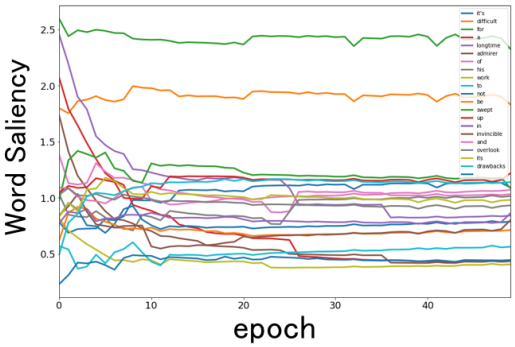

図 4　誤分類データの学習記録可視化結果

単語が多くある. この結果から, 誤分類データセットは, 正分類データセットに比べて, 注目度が不安定な単語が多いといえる.

5.5.2 注目度変化量の比較

正分類データセットと誤分類データセットから, それぞれ 100 件抽出し, 各 epoch での単語注目度変化量を足しあわせた結果を図 5 に示す. 横軸は epoch 数, 縦軸は注目度変化量を表す. ここでいう注目度変化量とは, n epoch から n+1 epoch にかけての注目度変化差分の絶対値を求め, すべての単語でその変化量を足しあわせたものである. なお, 注目度変化量の算出には, 処理を加えていない, そのままの勾配値を利用した.

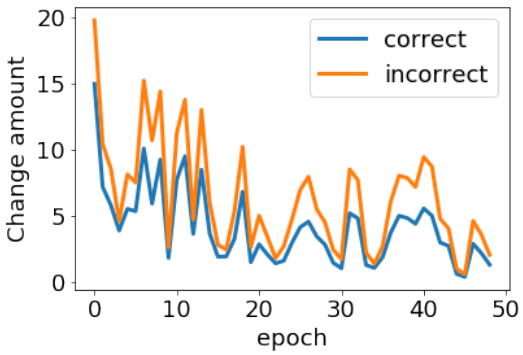

図 5　各 epoch の単語注目度変化量

図 5 を見ると, 各 epoch ごとの単語注目度変化量は, 正分類・誤分類ともに, 全般的に似たような変化を見せている. 例えば, 0 から 10 epoch に注目すると, 正分類・誤分類ともに同じ epoch

で注目度変化量が大きくなったり，小さくなったりしている．しかしながら，その変化量に注目すると，ほとんどの epoch で誤分類の注目度変化量の方が大きいことがわかる．例えば，40 epoch での注目度変化量を比較すると，正分類のグラフに比べて，誤分類のグラフの方が，変化量が大きいことがわかる．

5.6 後半学習による各モデルの性能比較

後半学習データセットを利用した学習結果について，4.4 節で述べた各モデルの性能を比較した結果を図 6 に示す．図中のラベルは，4.4 節で述べたモデルとそれぞれ対応している．

なお，横軸は epoch，縦軸は精度を表す．また，精度は各 epoch でモデルが評価データセットを分類して算出する．

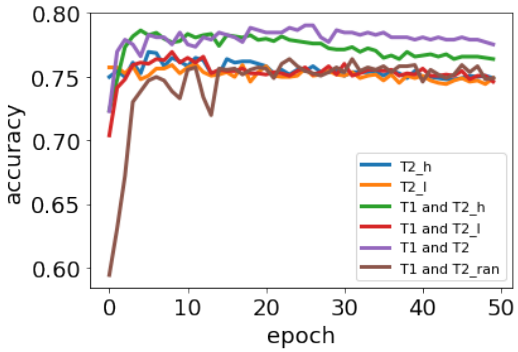

図 6　後半学習による各モデルの性能比較

図 6 の 40 epoch から 50 epoch を見ると，すべてのデータセットを使った学習がもっとも高い精度を示した．次に，T1 と $T2_h$ を組み合わせて，重みを引き継がずに学習したモデルが高い精度を示した．その他のモデルは同程度の精度となった．一方，0 epoch から 10 epoch に着目すると，すべてのデータセットを使った場合と，T1 と $T2_h$ を組み合わせて使った場合の精度はほぼ同じであるのに対して，20 から 30 epoch にかけてすべてのデータセットを使ったモデルの性能のほうが，T1 と $T2_h$ を使ったモデルに比べて高い精度を示した．

6. 考　　察

本節では，それぞれの実験における結果について，考察を述べる．

6.1 前半学習における可視化結果

図 4 の，誤分類データセットの可視化結果を見ると，誤分類データセットの単語の多くは，学習の後半に入っても注目度が収束していないことがわかった．また，正分類データセットは，注目度が線形に変化しているのに対して，誤分類データセットは注目度が不規則に変化していることがわかった．このことから，誤分類データの中には，注目度が収束せず，不安定な単語が多いと考えられる．

また，図 5 の可視化結果から，学習過程での，各単語の単語注目度変化量の総量は，誤分類データの方が正分類データより大きいことがわかった．これは，各単語の n epoch から n+1 epoch にかけての注目度差分を，それぞれのデータセットの全単語で足しあわせた時，誤分類データセットの総量の方が大きかったということを表している．このことから，誤分類データセットで出現している単語は，正分類データセットで出現している単語に比べて，学習過程で注目度を大きく変化させていると考えられる．すなわち，誤分類データセットの単語は，モデルが適切な重みを設定できていない可能性が示唆される．

このような，注目度が大きく変化している単語が，誤分類の原因であると仮定すると，このような単語をうまく学習させ，注目度を収束させることにより，モデルの性能が向上するのではないかと考えられる．

では，どのようにすれば注目度を収束させることができるか考える．結論としては，注目度を収束させるには，注目度変化の大きい単語を含む学習データが必要なのではないかと考えた．

前述したように，注目度変化が収束していないということは，モデルがその単語に対する適切な重みを決められていない可能性が考えられる．ニューラルネットワークは，入力から出力までの各層の重みを適切に調整することにより，モデルを構築する．適切な重みを設定するために必要不可欠なものの一つに，学習データがある．例えば，「good」という単語が出現した時，それがポジティブラベル文の中で多く使われ，ネガティブラベル文の中でほとんど使われていなければ，モデルは「good」と言う単語が，ポジティブな意味で使われている，分類に重要な単語であるとして，重みを大きくすると考えられる．しかし，例えば「marvelous」と言う単語が，データセット中に少ない場合，その単語が分類に重要な単語であるか判断できないため，適切な重みを設定できないのではないかと考えた．

そこで本研究では，後半学習で注目度変化量の大きい単語を多く含むデータセット $T2_h$ と，注目度変化の小さい単語を多く含むデータセット $T2_l$ をそれぞれ学習に用いてモデルの性能を比較することで，モデルの性能を向上させることができるか検証した．注目度変化量の大きい単語を多く含むデータセット $T2_h$ を用いて学習したモデルの性能の方が，注目度変化の小さい単語を多く含むデータセット $T2_l$ を用いて学習したモデルよりも高ければ，単語注目度変化の大きい単語を特定することにより，質の高い (モデルの性能をより向上させられる) 学習データの特徴を特定できるのではないかと考えた．

質の高い学習データの特徴を特定できれば，データを収集する際の指針にもなるため，有用な知見になりうると考える．

6.2 後半学習における各モデルの性能比較

図 6 の 40 から 50 epoch に注目すると，T1 と T2 のデータをすべて使ったモデルがもっとも高い精度を示し，約 0.78 の精度であった．次に，T1 と $T2_h$ を学習データとしたモデルが高い精度を示し，約 0.76 の精度だった．その他のモデルは同程度の精度となり，約 0.75 の精度だった．

まず，T1 よる前半学習の重みを引き継いで，$T2_h$ を用いて後半学習をしたモデルと，$T2_l$ を用いて後半学習をしたモデルの結果を比較する．0 epoch から 20 epoch に注目すると，$T2_h$ の方が $T2_l$ よりも高い精度となっている．しかし，30 epoch から 50 epoch に注目すると，$T2_h$ の精度が低くなり，どちらのモデルも同程度の精度となっている．このことから，学習の前半では $T2_h$ が高い精度を示すものの，学習を進めるにつれて，$T1_h$

と $T2_l$ のどちらも同じ精度となると言える．$T2_h$ は，学習の前半にかけて注目度が不安定な単語の特徴を学習するものの，後半では少ないデータセットから重みを調整してしまうために，学習データセットに適合しすぎて評価セットを正しく分類できなくなる，いわゆる過学習を起こしているのではないかと考えられる．

次に，$T1andT2_h$，$T1andT2_l$ の結果を比較する．まず $T1andT2_h$，$T1andT2_l$ のグラフに着目すると，学習を通して，$T1andT2_h$ の方が，$T1andT2_l$ に比べて高い精度を示していることがわかる．また，$T1andT2_{ran}$ と比較した場合も，同様に $T1andT2_h$ の方が，高い精度を示している．このことから，注目度変化の大きい単語を多く含むデータを利用することで，より精度の高いモデルを構築できる可能性があることが示唆された．

これは，既に構築したモデルの性能をより向上させるために，どのようなデータを収集すればいいか考える際に有用な情報となると考えられる．構築済みのモデルの性能を向上させる際，注目度変化の大きい単語を特定し，そのような単語を多く含むデータセットを収集して，再度モデルを構築することで，効率的にモデルの性能を向上させることができると考えられる．

最後に，$T1andT2$ と，$T1andT2_h$ の結果を見る．5.6 節で述べたように，学習の前半ではどちらも同程度の精度だったものの，学習の後半では，$T1andT2$ の精度の方が高くなった．これも先程と同様に，$T1andT2_h$ の方が学習データが少ないために，過学習を起こしているのではないかと考えられる．このことは，$T1andT2$ の精度が学習の後半でもあまり変化していないのに対して，$T1andT2_h$ は精度が 20 epoch 以降，線形に低下していることからも示唆される．

結論として，T2 学習による各モデルの性能比較で，$T1andT2_h$ の方が，$T1andT2_l$ に比べて高い精度を示した．このことから，単語注目度変化の大きい単語を特定することは，質の高い学習データを集めるのに有用であると考えられる．しかし，$T1andT2$ と $T1andT2_h$ の比較において，$T1andT2$ が学習全体で高い精度を維持したのに対して，$T1andT2_h$ が学習の後半で精度を落としたことから，過学習を抑えるためには，より多くのデータが必要であると考えられる．本研究では，比較的小規模のデータセットを使用したために，データセットを減らすことによって，特定のデータに適合しすぎてしまい，過学習の影響が大きくなってしまったと考えられる．このため，$T1andT2$ の方が，$T1andT2_h$ よりも，全体として高い精度となった．今後は，より大規模なデータセットを用いて検証することで，過学習の影響を小さくし，提案手法の有効性をより正確に検証する．また，本研究の検証においては，単語注目度変化の大きい単語を特定し，その単語を追加することによって，単語がうまく学習され，モデルの性能を向上させられるという仮定のもと，検証を行った．しかし，実験では，単語注目度変化の大きい単語を多く含むデータセットを学習に利用することで，モデルの性能がより向上することは確認したが，学習によって注目度が収束したのかは確認しなかった．後半学習によって，注目度変化の大きかった単語の注目度が収束し，うまく学習が行えていたのかも，検証する必要がある．

7. おわりに

本研究では，畳み込みニューラルネットワークによる文書分類タスクにおいて，モデルの学習過程での入力に対する注目度の変化を分析することで，誤分類の原因を分析する手法を提案した．

具体的には，まずデータセットを，前半学習用データセット T1，後半学習用データセット T2，評価用データセット T3 に分割し，T1 による学習において，モデルが入力に対する注目度をどのように変化させているか分析した．

分析の結果，誤分類データセットは，正分類データセットに比べて単語の注目度変化量が大きいことがわかった．また，学習の後半において，正分類データセットの単語は，注目度が線形に変化しているのに対して，誤分類データセットは，注目度が不安定に変化していることがわかった．

このことから，誤分類データセットは注目度が収束せず，注目度が変化している単語が多いと考えられる．すなわち，誤分類データセット中の単語には，モデルが適切な重みを設定できていない単語が多いと考えられる．本研究では，これを誤分類の原因であると仮定して，T2 による学習ではこれをうまく学習することを目指した．

T2 による学習では，T2 を注目度変化量が大きい単語で構成されているデータセット $T2_h$ と，注目度変化量が小さい単語で構成されているデータセット $T2_l$ に分割して後半学習を行い，それぞれのモデルの性能を比較した．

実験の結果，T1 と $T2_h$ を組み合わせたデータセットで学習したモデルが，T1 と $T2_l$ を組み合わせたデータセットで学習したモデルよりも高い精度を示した．このことから，単語注目度の高い単語を特定し，そのような単語を多く含むデータセットを用いて学習することにより，より高い性能のモデルを構築できる可能性が示唆された．

一方，T1 と $T2_h$ を組み合わせたデータセットで学習したモデルは，学習の後半で T3 に対する精度を低下させてしまった．これは，データセットが少ないために，学習データセットに適合しすぎてしまう，いわゆる過学習を起こしてしまったのではないかと考えられる．

今後の課題として，より大規模なデータセットを利用して，提案手法の有効性を検証する．

謝　辞

本研究は JSPS 科研費 JP16H02904 の助成を受けたものです．ここに記して感謝の意を示します．

文　献

[1] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, pp. 1–6, 2014.

[2] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[3] Sebastian Bach, Alexander Binder, Grgoire Montavon, Frederick Klauschen, Klaus-Robert Mller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier

decisions by layer-wise relevance propagation. *PLOS ONE*, Vol. 10, No. 7, pp. 1–46, 07 2015.

[4] Leila Arras, Franziska Horn, Grgoire Montavon, Klaus-Robert Mller, and Wojciech Samek. "what is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE*, Vol. 12, No. 8, pp. 1–23, 08 2017.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.

[6] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[7] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[8] Sunghyo Chung, Cheonbok Park, Sangho Suh, Kyeongpil Kang, Jaegul Choo, and Bum Chul Kwon. Revacnn: Steering convolutional neural network via real-time visual analytics. In *Future of Interactive Learning Machines Workshop at the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.