BERTによる感情分析

明治大学 総合数理学部
現象数理学科
4年2組42番
村田龍也

2022年2月13日

目 次

第 1 章　序論 **3**
1.1　研究の背景・動機 3
1.2　研究の目的 3
1.3　本論文の構成 3

第 2 章　自然言語処理における感情分析 **5**
2.1　はじめに 5
2.2　感情分析とは 5
2.3　感情分析の手法 5
2.3.1　感情辞書による手法 5
2.3.2　深層学習による手法 6
2.3.3　それぞれの手法のメリット・デメリット 6
2.4　BERT(深層学習モデル) を採用した理由 6

第 3 章　深層学習による自然言語処理 **7**
3.1　はじめに 7
3.2　ニューラルネットワーク 7
3.2.1　ユニットと活性化関数 7
3.2.2　順伝播型ニューラルネットワーク 8
3.2.3　学習の概要 9
3.2.4　損失関数 10
3.2.5　ミニバッチの利用 10
3.2.6　Adam 11
3.2.7　ソフトマックス関数 12
3.2.8　残差接続 12
3.2.9　Layer Normalization 13
3.2.10　Attention 13
3.3　深層学習 14
3.4　Transformer 14

3.4.1 Transformer の構成 14
3.4.2 Multi-Head Attention 16
3.5 BERT 17
3.5.1 BERT の構成 17
3.5.2 事前学習 17

第 4 章 BERT によるクラス分類 19
4.1 はじめに 19
4.2 トークン化 19
4.3 BERT の入出力 20
4.4 [CLS] によるクラス分類 20

第 5 章 実験 21
5.1 はじめに 21
5.2 実験の目的 21
5.3 実験に用いるデータセット 21
5.4 ハイパーパラメータの設定 22
5.5 実験 1 の概要 23
5.6 実験 1 の結果 23
5.7 実験 2 の概要 24
5.8 実験 2 の結果 24
5.9 実験の考察 25

第 6 章 結論と今後の課題 27
6.1 結論 27
6.2 今後の課題 27

第1章　序論

1.1　研究の背景・動機

近年、深層学習の発展により従来よりも高い精度での感情分析が可能になっている. また、インターネットを通じて様々な種類のデータを大量に入手できるようになったため、感情分析の適用分野は広がりつつある. ビジネスに活かす試みも増えており、感情解析ビジネスの市場予測を公表している調査会社の多くが今後3〜5年で3兆円規模の巨大市場になると予測している [1].

感情分析は音声、テキスト、映像の解析などがあるが、本論文では扱うのはテキストの感情分析である. テキストの感情分析は、これから流行しそうなトピックの特定や自社製品に対するフィードバックの収集、ネガティブコメントを特定して事前に炎上を防いだりと様々な方法で社会の役に立っている. テキストの感情分析では感情極性辞書による手法か深層学習による手法が用いられることが多い.

自然言語処理における感情分析を研究テーマに選んだ動機は、感情を数値化してコンピューターに解かせるという一連の流れが新鮮で興味深く感じたためである.

1.2　研究の目的

宿泊先のレビューに感情極性のラベル付けをしたコーパスを訓練データとして用いてBERTを学習させ、テストデータに対して推論を行い感情極性辞書による手法の精度との比較を行うことを目的とした.

1.3　本論文の構成

本論文の構成は以下の通りである.
第1章は本章であり、本研究の背景や目的について述べる.
第2章では、感情分析の概要と手法について述べる.
第3章では、深層学習に関する用語や定義について述べる.
第4章では、BERTによるクラス分類の手法について述べる.

第 5 章では、実験概要、結果および考察について述べる.
第 6 章では、本論文の結論と今後の課題について述べる.

第2章　自然言語処理における感情分析

2.1　はじめに

本章では、自然言語処理における感情分析で用いられている手法、そしてそれらの手法のメリットとデメリットについて述べる.

2.2　感情分析とは

感情分析とは、文章に含まれる感情を推定するタスクのことである. 入力の文章を有限個のクラス (感情) に分類するためクラス分類に属する. 特に、三種類以上のクラスを扱う場合は多クラス分類と呼ばれる. 推定する感情はモデルによって異なり、ポジティブかネガティブかの二種類のみを扱うモデルもあれば、十種類の感情を扱うモデルもある. 本論文ではポジティブ、ネガティブの二種類とポジティブ、ネガティブ、ニュートラルの三種類の感情を扱うものとする.

2.3　感情分析の手法

感情分析の手法は大きく分けて二種類ある. 一つは感情極性辞書による手法、もう一つは深層学習による手法である。

2.3.1　感情辞書による手法

この小節は、[2] による. 感情極性辞書による違いはあるが、基本的な流れは以下の通りである.

1. 対象の文章を形態素解析、係り受け解析により単語に分割
2. 分割した単語を辞書に基づいてポジティブ、ネガティブを振り分ける

3. 各単語の情報を用いて文章の感情を推定

形態素とは、言葉が意味を持つまとまりの単語の最小単位のことである. 形態素解析で文章を形態素に分割し、係受け解析により各形態素の係り先を決定する.

日本語の感情極性辞書では東北大の乾研究室が公開している日本語評価極性辞書 [3][4] と東工大の高村教授が公開している単語感情極性対応表 [5] が広く使われている.

2.3.2 深層学習による手法

深層学習による手法では、文章と感情極性に関するラベルが対になった訓練データを用いる. ラベルはコンピュータで処理しやすいように感情カテゴリーを整数値で置き換える. どのような言い回しや表現があればポジティブ、ネガティブ、ニュートラルになるかを事前に定義する. このようなデータセットを用いて、定めた損失関数が最小となるようにモデルの重みとバイアスを更新する.

2.3.3 それぞれの手法のメリット・デメリット

感情極性辞書のメリットは判定の過程が解釈しやすいことである. また、深層学習のように大量のデータを必要とすることなく文章を辞書に沿って判別することが可能である. デメリットとしては、多くのルールを作成する必要があること、感情極性辞書に載っていない単語があると対応できないこと、ニュートラルの判別が難しいことが挙げられる.

一方、深層学習のメリットはルール作成が不要であること、単語間の関係性を持つこと. デメリットは学習するために多くの教師データを用意する必要があること、判定の過程が解釈しづらいことが挙げられる

2.4 BERT(深層学習モデル) を採用した理由

BERT は 2018 年に Google により提案された自然言語における深層学習のモデルである [6]. BERT の特徴として、従来の深層学習モデルと比べて学習に必要な訓練データ数が少なくて済むこと、単語間の関係を考慮できることが挙げられる.

BERT は感情辞書による手法と比べて深層学習による手法が苦手な点を克服しており、長所を強化している. これらの理由より、BERT を用いて感情分析を行うことを決めた.

第3章　深層学習による自然言語処理

3.1　はじめに

本章では、深層学習の基礎知識と Transformer, BERT について述べる.

3.2　ニューラルネットワーク

ニューラルネットワークとは、人間の脳神経をモデルにした情報処理システムである. 本節ではニューラルネットワークの構成要素、そして学習の概要について述べる. この節の内容は、岡谷 [7] による.

3.2.1　ユニットと活性化関数

ユニットはニューラルネットワークを構成する最小の要素である. 複数の入力を受け取り、1 つの出力を計算する. 図 3.1 の場合、4 つの入力 x_1,x_2,x_3,x_4 を受け取り、各入力に異なる重み w_1,w_2,w_3,w_4 を掛けて加算したものにバイアスと呼ばれる値 b を足したものが総入力 u である.

$$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4 + b \tag{3.1}$$

この総入力 u を活性化関数と呼ばれる関数 f に通した値がユニットの出力となる.

$$z = f(u) \tag{3.2}$$

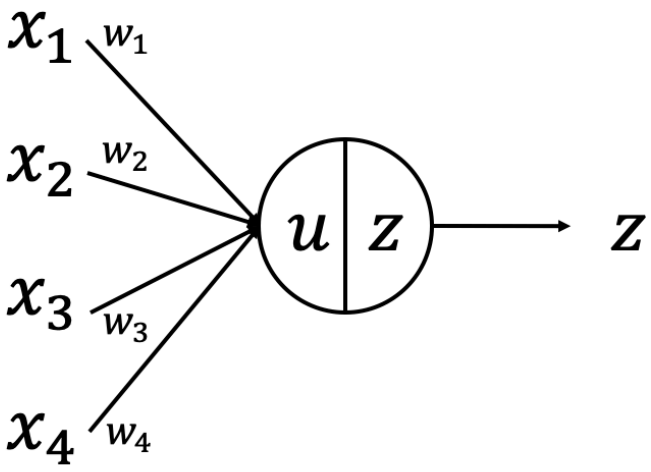

図 3.1: ユニット

線形変換で得られた u を活性化関数 $f(\cdot)$ で非線形変換を行うことでニューラルネットワーク全体としても非線形性を持つことができる.

主に ReLU(Rectified Linear Unit)、シグモイドと呼ばれる活性化関数などが用いられている.

3.2.2 順伝播型ニューラルネットワーク

順伝搬型ニューラルネットワークは層状に並べたユニットが隣接した層間でのみ結合した構造を持ち、 入力 $\boldsymbol{x}$ から各層の計算を順番を実行し、最後に出力 $\boldsymbol{y}$ を得るニューラルネットワークである. 代表的なニューラルネットワークの一つである. この基本構造を図 3.2 に示す. この図の l=1 の層を入力層、l=2 の層を中間層、l=3 の層を出力層と呼ぶ. また、第 l 層の i 番目のユニットに紐づいている重みのうち、j 番目の重みを $w_{i,j}^{(l)}$ と表現する.

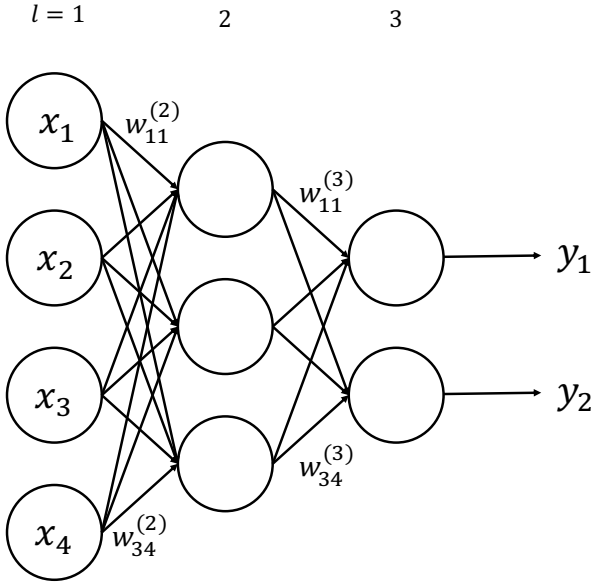

図 3.2: 順伝搬型ニューラルネットワーク

この順伝搬型ニューラルネットワークが入力 $\boldsymbol{x}$ から出力 $\boldsymbol{y}$ を得る計算は、関数 $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ として表現できる, 関数の中身は、各層間の結合の重みとバイアスによって決まる. ネットワークが表現する関数は、このパラメータを変えることで様々な関数に変化させることができる.

3.2.3 学習の概要

入力 $\boldsymbol{x}$ に対する望ましい出力 $\boldsymbol{d}$ のペアが複数与えられているとする. これらのペア 1 つ 1 つを訓練サンプルと呼び、その集合を訓練データと呼ぶ. また、入力 $\boldsymbol{x}$ の要素を特徴量と呼ぶ. $\boldsymbol{d}$ はラベルと呼ばれる.

$$\{(\boldsymbol{x}_n, \boldsymbol{d}_n)\}_{n=1,...,N} = \{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), ..., (\boldsymbol{x}_N, \boldsymbol{d}_N)\} \tag{3.3}$$

ニューラルネットワークにおける学習とは、与えられた訓練データのすべての訓練サンプル $(\boldsymbol{x}_n, \boldsymbol{d}_n)$ $(n = 1, \ldots, N)$ について、入力 $\boldsymbol{x}_n$ を与えたときのニューラルネットワークの出力 $\boldsymbol{y}(\boldsymbol{x}_n)$ が、なるべく $\boldsymbol{d}_n$ に近くなるように重みとバイアスを調整することである. 以降、モデル内の全ての重みとバイアスを成分に持つベクトルを $\boldsymbol{w}$ と定義する.

$$\boldsymbol{w} = [w_1, \ldots, w_M, b_1, \ldots, b_N] \tag{3.4}$$

M は重み、N はバイアスの成分数である.

3.2.4 損失関数

損失関数 $E_n(\boldsymbol{w})$ は $\boldsymbol{x}_n$ に対するニューラルネットワークの出力 $\boldsymbol{y}(\boldsymbol{x}_n)$ と訓練データ $\boldsymbol{d}_n$ との差を測る関数である. 損失関数を最小化するように重みとバイアスの更新を行う. 更新には損失関数の勾配を用いる. 勾配とは以下のベクトルである.

$$\frac{\partial E}{\partial \boldsymbol{w}} = \left[\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_M}, \frac{\partial E}{\partial b_1}, \ldots, \frac{\partial E}{\partial b_N}\right] \tag{3.5}$$

更新は現在の重みとバイアスを $\boldsymbol{w}_t$, 更新した後の重みとバイアスを $\boldsymbol{w}_{t+1}$, 更新量の大きさを定める学習率を η とすると更新は以下の式で行われる.

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \tag{3.6a}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \frac{\partial E}{\partial \boldsymbol{w}_t} \tag{3.6b}$$

このように、すべての訓練サンプルを使って $\boldsymbol{w}$ を更新する方法をバッチ学習と呼ぶ. 損失関数は問題に応じて選定する. 回帰問題では二乗誤差、分類問題では交差エントロピーといった損失関数がよく使われている.

3.2.5 ミニバッチの利用

バッチ学習は最小化する目的関数が常に同じなので、局所的な極小解から抜け出すことができない. そこで一定数の訓練サンプルの集合単位で重みを更新する方法が用いられることが一般的である. この訓練サンプルの集合のことをミニバッチと呼ぶ. また、学習で t 回目の更新に用いるミニバッチを D_t とするとき t 回目の更新は以下の式で行われる.

$$E_t(\boldsymbol{w}) = \frac{1}{N_t}\sum_{n\in D_t} E_n(\boldsymbol{w}) \tag{3.7a}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\frac{\partial E_t}{\partial \boldsymbol{w}_t} \tag{3.7b}$$

ミニバッチを D_1,D_2,... と取り出して使い、一巡したら再度、最初から D_1,D_2,... の順に取り出して使う. ミニバッチ学習による $\boldsymbol{w}$ の更新回数をステップ数と呼ぶ. また、ミニバッチが一巡することをエポックと呼ぶ.

3.2.6 Adam

勾配降下法による学習の成否は学習率の選定に左右される. そのため学習率の選定の重要度を下げるべく、重みの更新の幅を適応的に調整することを目的として最適化アルゴリズムが作られている. 現在、最も広く使われているのが Adam という手法である.

$$\boldsymbol{v}_t = \beta_1\boldsymbol{v}_{t-1} + (1-\beta_1)\frac{\partial E_t}{\partial \boldsymbol{w}_t} \tag{3.8a}$$

$$\boldsymbol{s}_t = \beta_2\boldsymbol{s}_{t-1} + (1-\beta_2)\frac{\partial E_t}{\partial \boldsymbol{w}_t}\odot\frac{\partial E_t}{\partial \boldsymbol{w}_t} \tag{3.8b}$$

$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_1^t} \tag{3.8c}$$

$$\hat{\boldsymbol{s}}_t = \frac{\boldsymbol{s}_t}{1-\beta_2^t} \tag{3.8d}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\frac{\hat{\boldsymbol{v}}_t}{\sqrt{\hat{\boldsymbol{s}}_t + \varepsilon}} \tag{3.8e}$$

パラメータの更新は式 (3.8e) によって行う. 勾配とその二乗の移動平均はモーメントの推定値としては偏差を含むため、式 (3.8c),(3.8d) によって補正する. 勾配の移動平均を取り振動方向の学習率を下げることで勾配降下法より効率よく更新が可能なため、安定して良い結果が得られやすい.

3.2.7　ソフトマックス関数

ソフトマックス関数は多クラス分類を対象とする場合に、出力層の活性化関数として用いられる関数である．分類したクラス数 K と同数のユニットの総入力 $u_j(j = 1, \ldots, K)$ に対して以下の式で計算される．

$$y_k = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)} \tag{3.9}$$

こうして決まる出力 y_1,...,y_K は総和が 1 になる．このとき y_k は与えられた入力 $\boldsymbol{x}$ がクラス k に属する確率を表している．

3.2.8　残差接続

残差接続とは、図のように層の出力 $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ に入力 $\boldsymbol{x}$ を足し合わせる構造のことである．残差接続の直前から加算後のひとまとまりを指して残差ブロックと呼ぶ. 多くの場合、残差ブロックを複数積み重ねて 1 つのニューラルネットワークを構築する．

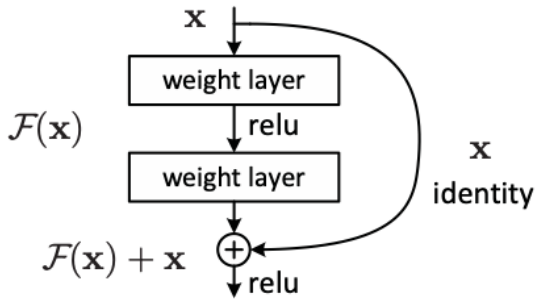

図 3.3: 残差接続

残差接続を持つニューラルネットワークでは、多層であっても学習の難しさが大きく緩和されるため、残差接続によって一般に推論の性能向上につながるネットワークの多層化が可能になる．

3.2.9 Layer Normalization

Layer Normalization はデータの正規化の一種である. 各層の出力を強制的に正規化することによって、出力の分布を整える. 活性化関数を適用する前のユニットの出力 $\boldsymbol{u} = [u_1, u_2, ..., u_J]^T$ (J はこの層のユニットの数) に対し、同じ層の全ユニットにわたる平均 $\mu = \frac{1}{J}\sum_{j=1}^{J} u_j$ と分散 $\sigma^2 = \frac{1}{J}\sum_{j=1}^{J}(u_j - \mu)^2$ を計算し、以下の式で正規化を行う. $\hat{u_j}$ は正規化した u_j である.

$$\hat{u_j} = \gamma_j \frac{u_j - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta_j \tag{3.10}$$

γ_j と β_j は重みとバイアスと同様に学習によって決定する. ϵ は 0 で除算することを防ぐために導入された整数である. 正規化を行うことで学習の安定と推論精度の向上が見込める.

3.2.10 Attention

Attention(注意) とは、複数の要素からなる集合 $\boldsymbol{z}_i (i = 1, ..., N)$ に対し、クエリ $\boldsymbol{q}$ に応じた重要度に従って、各要素を重み付けすることである. クエリはタスクにおける入力である.

$\boldsymbol{z}_i, \boldsymbol{q} \in \mathbb{R}^D$ とするとき要素 $\boldsymbol{z}_i$ とクエリ $\boldsymbol{q}$ の関連性を内積によって計算する. $\sqrt{D}$ で割っているのは、このあと適用するソフトマックス関数による出力が 0 と 1 に二極化する傾向を緩和するためである.

$$r_i = \frac{\boldsymbol{z}_i^T \boldsymbol{q}}{\sqrt{D}} \tag{3.11}$$

各要素に対して得られた $r_1, r_2, ..., r_N$ を総和が 1 になるようにソフトマックス関数 (3.9) で正規化する.

$$a_i = \frac{\exp(r_i)}{\sum_{j=1}^{N} \exp(r_j)} \tag{3.12}$$

そして、a_i を重みとする要素 $\boldsymbol{z_i}$ の加重平均を求める.

$$\boldsymbol{z} = \sum_{i=1}^{N} a_i \boldsymbol{z}_i \tag{3.13}$$

こうして得られた $\boldsymbol{z}$ は、$\boldsymbol{q}$ と関連の深い要素を 1 つのベクトルでコンパクトに表現したものと考えられる. 以上の計算において、特徴の集合 $\boldsymbol{z}_i(i = 1, ..., N)$ をソース、クエリ $\boldsymbol{q}$ をターゲットと呼ぶ. そして式 (3.11) から式 (3.13) に至る計算をソース・ターゲット注意 (source-to-target attention) と呼ぶ. ソースとターゲットを同じにする場合、自己注意 (self attention) と呼ぶ. 自己注意を用いたモデルは従来のモデルと比べて、文章内の単語の関係性が把握できるようになった.

3.3 深層学習

深層学習は中間層を複数個持つニューラルネットワーク (ディープニューラルネットワーク) を用いた機械学習手法の総称である.

3.4 Transformer

Transformer[8] は 2017 年に発表された深層学習モデルであり、自然言語などの時系列データを扱ってタスクを行うように設計されている. 従来、自然言語処理で広く用いられていた RNN（リカレントニューラルネットワーク）構造を使用せず、Attention を核にした構造を持つ. 元々は機械翻訳を目的に考案されたが、言語を扱うほぼすべてのタスクに対して有効であることが確かめられている.

3.4.1 Transformer の構成

Transformer の構成は以下の図 3.4 で表される. 図の左側のブロックが Encoder、右側のブロックが Decoder と呼ばれる. BERT は Transformer を使ったモデルであるが、使っているのは Transformer の Encoder だけであるため、Decoder についての内容は省略する.

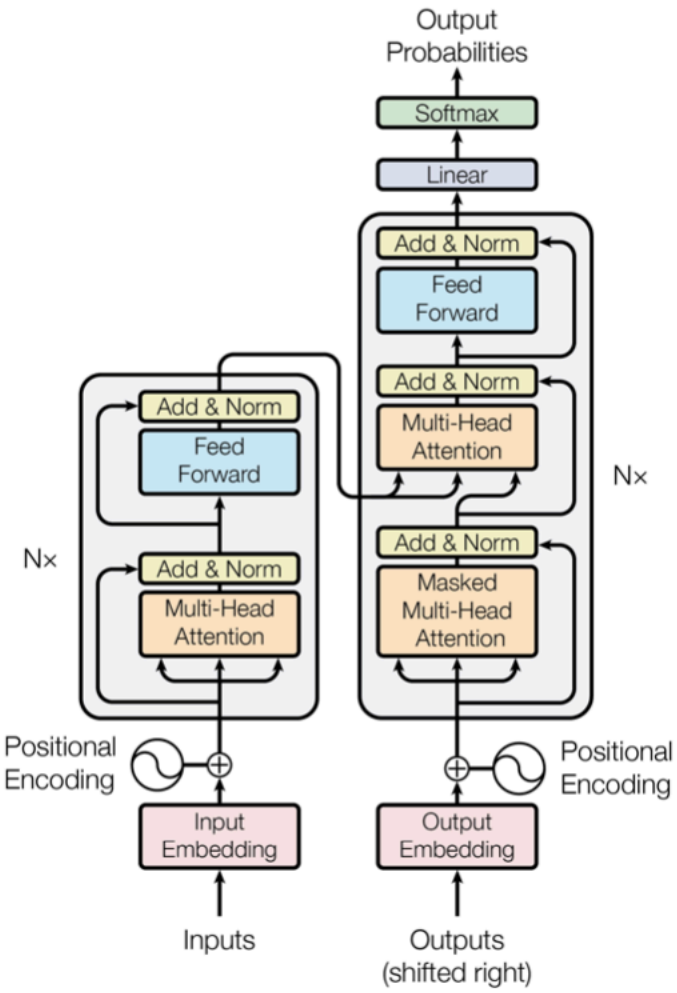

図 3.4: Transformer

Transformer の Encoder では Multi-Head Attention, Add & Norm, Feed Forward, Add & Norm, これらの層を順に計算を行なっていく．ここでは残差接続 (3.2.8) と Layer Normalization(3.2.9) を合わせて Add & Norm, 順伝搬ニューラルネットワーク (3.2.2) のことを Feed Forward と呼んでいる．活性化関数には ReLU を採用している．また、ここでの Multi-Head Attention はソースとターゲットを同じにして計算を行う self attention である．

3.4.2 Multi-Head Attention

Transformer では複数のクエリを扱うほか、Attention の重みも平行に複数生成し、ソースのベクトルに適用する．これを Multi-Head Attention と呼ぶ．また、Transformer ではソースを Key と Value に分離する．分離することにより、Key と Value 間の非自明な変換によって高い表現力が得られると考えられている．

$\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M] \in \mathbb{R}^{M \times D}$, $\boldsymbol{K} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N], \boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N] \in \mathbb{R}^{N \times D}$ とするとき Attention 適用後の行列 Attention$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \in \mathbb{R}^{M \times D}$ は以下のように計算される．

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{D}}\right)\boldsymbol{V} \tag{3.14}$$

ただし、ここでの softmax 関数は入力の行列の各行ベクトルに対して独立に適用している．

次のようにして、Atteniton の計算を H 個並列に実行する．各 Attention $h(= 1, \ldots, H)$ では $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ の各行ベクトルを D より小さい D' 次元空間に線形写像する．これを行うため、3 つの行列 $\boldsymbol{W}_h^Q, \boldsymbol{W}_h^K, \boldsymbol{W}_h^V$ を導入し、クエリ、Key、Value それぞれにこの行列を掛け計算した結果を $\text{head}_\text{h} \in \mathbb{R}^{M \times D'}$ とする．

$$\text{head}_\text{h} = \text{Attention}(\boldsymbol{W}_h^Q, \boldsymbol{W}_h^K, \boldsymbol{W}_h^V) \tag{3.15}$$

こうして得られた H 個の $\text{head}_\text{h}(h = 1, \ldots, H)$ を行方向に連結し、$\boldsymbol{W}^O \in \mathbb{R}^{D'H \times D}$ による線型写像を行い出力 MultiHead$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \in \mathbb{R}^{M \times D}$ とする．

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \mathit{Concat}(\text{head}_1, ..., \text{head}_\text{H})\boldsymbol{W}^O \\ \boldsymbol{where} \quad \text{head}_\text{h} &= \text{Attention}(\boldsymbol{W}_h^Q, \boldsymbol{W}_h^K, \boldsymbol{W}_h^V) \end{aligned} \tag{3.16}$$

Multi-Head Attention は単一の Attention より性能が高いことが示されている [8].

3.5 BERT

この節の内容は、. BERT は Bidirectional Encoder Representations from Transformers の略称で、Transformer の Encoder を使っているモデルである. BERT はラベルのついていない文章から言語表現を事前学習するように作られたモデルで、タスク内容に応じて BERT に新しい分類器などを接続するなどして、タスクに特化したモデルを作る. 当時、自然言語処理のタスク 11 個で従来のモデルの記録を更新した.

3.5.1 BERT の構成

BERT は Transformer の Encoder を使っているモデルである. BERT にはモデルサイズの異なる 2 つのタイプがあり、Encoder を 12 層繋げたモデルを BERT-base, 24 層繋げたモデルを BERT-large と呼ぶ. Transformer の Encoder では、Multi-Head Attention, Add & Norm, Feed Forward, Add & Norm, これらの層を順に計算を行なっていく. この BERT の出力をタスクに合わせた分類器に入力し、タスクに応じた出力を得る.

3.5.2 事前学習

事前学習 (Pre-training) は大規模な文章コーパスを用いて汎用的な言語のパターンを学習するために行われる. このとき、用いるのはラベル付けされていない生の文章データである. ラベルなしデータを用いるメリットは、比較的容易に大量のデータを収集できる点である. BERT の事前学習では以下の二つのタスクを行うことによってモデルのパラメータを更新する.

1. マスク付き言語モデル
   あるトークンを文章中の他のトークンから予測するタスク.

   文章の中でランダムに選ばれた 15%のトークンを [MASK] という特殊トークンに置き換え、[MASK] の位置に元々あったトークンを予測する.

2. Next Sentence Prediction
   入力された二つの文章が連続したものであるか判定するタスク.

このとき入力された二つの文章は、50%で二つ目の文章が一つ目の文章に連続する文であり、残りの 50%は二つ目の文章がランダムに選ばれた文である. 入力された二つの文が連続したものである.

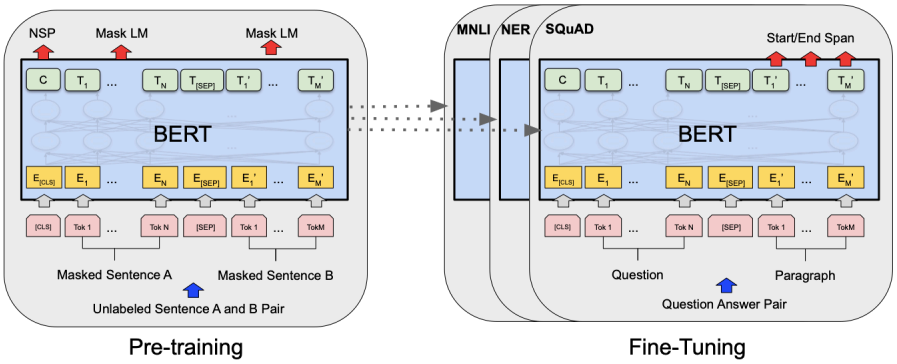

図 3.5: BERT の学習

事前学習したモデルは行いたいタスクに応じた訓練データを用いて学習を行う. この学習方法は、BERT では Fine-Tuning と呼ばれている. 事前学習で得たパラメータを初期値とすることで効率よくパラメータの更新が可能になり、少ない訓練データでモデルの性能を向上させることができる.

第4章　BERTによるクラス分類

4.1　はじめに

本章では、BERT によるクラス分類の手法について述べる. 本論文で扱う感情分析もクラス分類の一種である. 本章は、近江、金田、森永、江間見 [9] による.

4.2　トークン化

文章をニューラルネットワークに入力するためには、文章を数値化する必要がある. まず、文章を適当な単位に分割する. これをトークン化と呼び、トークン化によって得られた文章のそれぞれの構成要素をトークン、すべての構成要素をまとめてトークン列と呼ぶ.

トークン化には何種類かあるが、BERT で用いられているのはサブワード分割という手法である. サブワード分割は単語単位で分割した後、さらに部分文字列に分割する. この手法には、語彙に含まれない未知語に対応できるという特徴がある. また、BERT ではトークン列の先頭に [CLS] を、末尾に [SEP] という特殊トークンを加える. [CLS] は分類問題を解く際に使い、[SEP] は入力の終わりを示す. このトークン列が、ニューラルネットワークへの入力に変換される手順は以下の通りである.

1. 事前にトークンの集合 (語彙) を作成し
   これに含まれる各トークンに対して順番に ID を割り当てる

2. ニューラルネットワークに渡されたトークンを語彙に従い ID に変換する

以上の変換を行うことでトークン列を語彙による ID 列にする.

4.3 BERTの入出力

BERTの入力は文章をサブワード分割して得られたID列である、このID列の長さは全ての文章で揃える必要がある. BERTでは入力のID列の長さは512個となっており、512個より短いトークン列に対しては末尾に特殊トークン[PAD]を必要な数だけ追加し、512個より長いトークン列に対しては必要な数だけ末尾のトークンを取り除く.

その後、このID列を各単語のベクトル表現の行列(512×768)に変換する. この行列の各要素は学習するパラメータである. BERTでは各単語の次元は768次元と定められている. そして得た行列をTransformerのEncoderで12回繰り返して出力する. この出力の行列の形状も(512×768)である. この出力は、文章内の各単語の関係性を考慮した単語ベクトルの行列である.

4.4 [CLS]によるクラス分類

BERTの出力層の[CLS]の特徴量ベクトル(1×768)に対して線形変換を行い、ラベルの数Nと同じ次元を持つベクトル$(1 \times N)$を出力する. このとき、出力のベクトルの各要素の値はそれぞれラベルへの予測の確度を表すものと考え、分類スコアと呼ぶ. j番目の分類スコアとj番目のラベルが対応している(j = 1,...,N). 分類スコアが最も高いラベルを予測値とすることで文章のクラス分類を行う. このラベルに感情カテゴリーを整数値に置き換えたものを用いることで感情分析を行うことができる.

第5章　実験

5.1　はじめに

本章では、BERT による感情分析の評価実験、考察を行う.

5.2　実験の目的

宿泊先のレビューに感情極性のラベル付けをしたコーパスを訓練データとして用いて BERT を学習させ、テストデータに対して推論を行い感情極性辞書による手法の精度との比較を行うことを目的とした. この比較はポジティブ、ネガティブの 2 種類のラベルのデータセットで行った. この実験を実験 1 とする. また、感情極性辞書による手法では難しいポジティブ、ネガティブ、ニュートラルの 3 種類のラベルのデータセットでも同様に学習と推論を行いモデルの精度を確かめた. この実験を実験 2 とする. モデルには東北大乾研が日本語 Wikipedia により事前学習を行なったモデル[1]を扱える、huggingface 社が提供している BertForSequenseClasification[2]を使用した. このモデルは BERT-base である. また、クラス分類は BERT の出力層の [CLS] の特徴量ベクトル (1×768) に対して線形変換、活性化関数、線形変換を順に行いラベルの数 N と同じ次元を持つベクトル $(1 \times N)$ を出力する.

5.3　実験に用いるデータセット

実験には、株式会社リクルートの提供している Japanese Realistic Textual Entailment Corpus[11] を用いた[3]. このデータセットは、じゃらんのクチコミデータから抽出した文、それらを加工した文、アノテーション作業者が付与した判定ラベルが含まれる. 判定ラベルはポジティブ、ネガティブ、ニュートラルの三つのうちいずれかである. ポジティブが 1、ネガティブが −1、ニュートラルが 0 に対応

[1]https://github.com/cl-tohoku/bert-japanese
[2]https://github.com/huggingface/transformers
[3]https://github.com/megagonlabs/jrte-corpus

している. 文がどの判定ラベルに対応するかはアノテーションを行った 3 名による多数決によって決定している. このデータセットは 5553 件の訓練サンプルで構成されている. また、訓練データ、検証データ、テストデータにどの訓練サンプルを使うか予め決められている. 今回はこのデータセットを訓練データとして用いて BERT の学習を行なった.

pn17q04429	-1	ワンルームタイプの部屋にベッドが置いて	{"-1": 3}	train
pn17q04430	1	」と驚くこと必至で男性の私でも食べきれ	{"1": 3}	dev
pn17q04431	1	明日からの仕事頑張れます。	{"1": 3}	train
pn17q04433	-1	ただ、夜に観光から戻り二階のエレベーク	{"-1": 3}	dev
pn17q04434	1	客室露天風呂の他温水洗浄付トイレ、ボ	{"0": 1, "1": 2}	dev
pn17q04435	1	ホテルのかたの素早い対応も気持ちよか	{"1": 3}	train
pn17q04437	1	部屋からは瀬戸内の海が目の前に広がり、	{"1": 3}	dev
pn17q04438	1	115kの私もゆったりなベッド&浴そう!	{"1": 3}	train
pn17q04439	-1	お部屋は清潔に保たれていましたが、寝る	{"-1": 2, "0": 1}	train
pn17q04440	-1	朝食は他のグループホテルと比べるとかな	{"-1": 2, "0": 1}	train
pn17q04443	1	選択肢も比較的多く満足できた。	{"1": 3}	train
pn17q04447	0	朝食はハーフバイキングみたいな感じでし	{"0": 2, "1": 1}	dev
pn17q04448	1	今回もこちらのホテルにして良かったと実	{"1": 3}	train
pn17q04449	0	今回の旅行は知人を東京観光に案内するス	{"0": 3}	train

図 5.1: Japanese Realistic Textual Entailment Corpus

5.4 ハイパーパラメータの設定

ハイパーパラメータとは、モデルが学習を行う前に人間が予め設定する必要のあるパラメータである. 筆者が設定したハイパーパラメータとその値は以下の通りである.

- バッチサイズ : 32
- エポック : 10
- 学習率 (実験 1) : 1×10^{-6}
- 学習率 (実験 2) : 5×10^{-6}

学習率を実験 1 と実験 2 で変えた理由として、2 種類のラベルの推論を行う実験 1 は実験 2 と比べて過学習しやすいためである. 過学習とは訓練データに過剰に適合して汎化性能が低くなることである. 学習率を小さくすると過学習を抑えられるため、実験 1 の学習率を実験 2 の学習率より小さくした.

5.5 実験 1 の概要

データセットをポジティブ、ネガティブの 2 種類のラベルを限定して、BERT で学習と推論を行なった. 目的は、推論の精度を辞書による手法と比較することである. 精度は以下の式より算出した. そのため、実験 1 の学習率を実験 2 の学習率より小さくした.

$$\text{精度} = \frac{\text{推論でラベルを当てたテストデータの数}}{\text{テストデータの総数}} \tag{5.1}$$

5.6 実験 1 の結果

訓練データで BERT の学習を行なったところ、訓練データと検証データでの学習曲線は下記の図の通りになった. この図の縦軸はステップ数、横軸は損失関数の値である. 検証データは訓練データで学習した BERT が過学習しているかどうかを確認するために使うデータである.

図 5.2: 訓練データの学習曲線

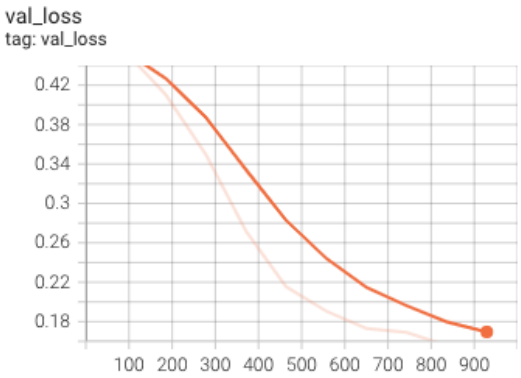

図 5.3: 検証データの学習曲線

図 5.3 の検証データの学習曲線はステップが進むにつれ損失関数の値が下がっている. これは BERT が過学習せずに汎化性能を高められていることを示している.

学習した BERT の評価をテストデータを用いて行なった．その結果、96%の精度で推論を行うことができた．

```
test = trainer.test(test_dataloaders=dataloader_test)
print(f'Accuracy: {test[0]["accuracy"]:.2f}')
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing: 100%
--------------------------------------------------------
DATALOADER:0 TEST RESULTS
{'accuracy': 0.9613526463508606}
--------------------------------------------------------
Accuracy: 0.96
```

図 5.4: テストデータに対する精度

Kaggle で行われた RKcup#1 [4]にて東北大乾研究室が公開している日本語評価極性辞書を用いた手法 (ベースライン) での精度は 80%だった．データセットは本研究と同じ Japanese Realistic Textual Entailment Corpus である．BERT による精度は 96%と辞書による手法の精度より高いことが分かった．

5.7 実験 2 の概要

ポジティブ、ネガティブ、ニュートラルの 3 種類のラベルのデータセットを用いた．感情極性辞書による手法では扱うことが難しいニュートラルを含んだデータセットでの BERT の推論の精度を確かめることを目的とした．

5.8 実験 2 の結果

実験 1 と同様に BERT の学習と推論を行なった．学習曲線は下記の図の通りになった．この図から BERT が過学習せずに汎化性能を高められていることが分かる．

[4]https://www.kaggle.com/c/rkcup-1

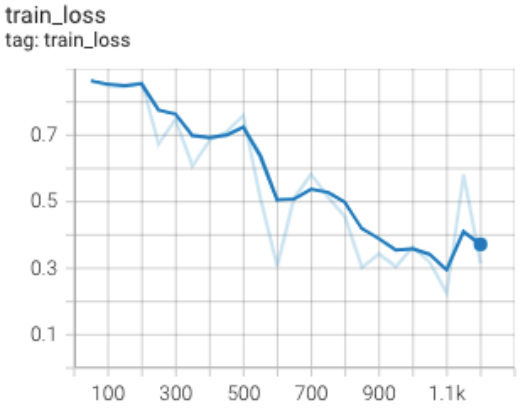

図 5.5: 訓練データの学習曲線

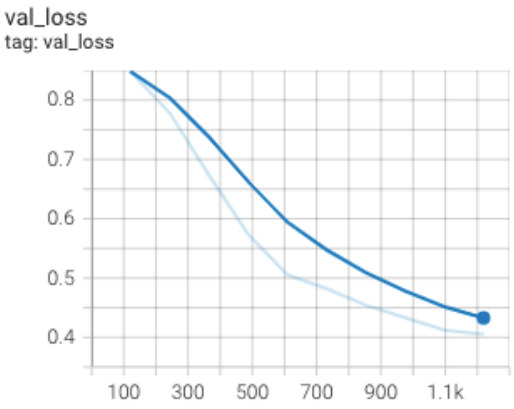

図 5.6: 検証データの学習曲線

学習した BERT でテストデータに対して推論を行なったところ精度は 85%だった．精度は実験 1 と同様に式 (5.1) で求めた．この精度は 2 種類のラベルの感情極性辞書による手法の精度より高いことが分かった．

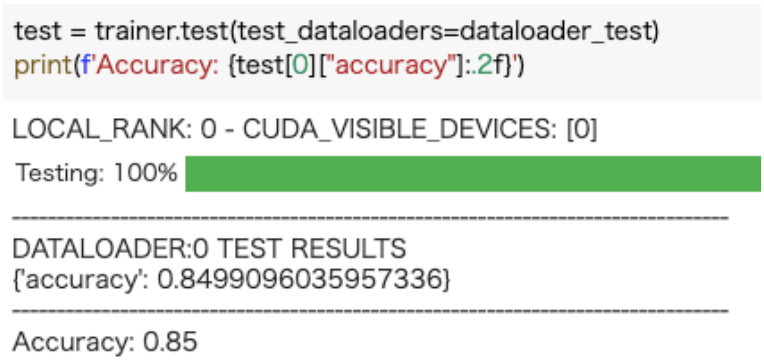

図 5.7: テストデータに対する精度

5.9 実験の考察

実験 1 の結果より、実験で用いたテストデータに対して学習した BERT による推論の精度は 96%であり、感情極性辞書による手法の精度である 80%を大きく上回ることが分かった．また、実験 2 では、感情極性辞書では扱うことが難しいニュートラルのラベルを含んだデータセットに対する推論の精度は 85%であった．

実験 1 で同条件で精度に 16%の差があること、実験 2 でポジティブ、ネガティブ、ニュートラルの 3 種類のラベルのデータセットによる精度も 80%を上回っていることから、Japanese Realistic Textual Entailment Corpus に対して感情極性辞書による手法よりも BERT による手法が有効であることが分かる.

さらに、同条件で精度に大きな差があるため、他のデータセットに対しても、精度において BERT による手法が感情極性辞書による手法を上回ると考えられる. そのため、学習に用いる訓練データがあれば BERT による手法を選択するべきだと考える.

第6章　結論と今後の課題

6.1　結論

本研究では、宿泊先のレビューに感情極性のラベル付けをしたコーパスを訓練データとして用いてBERTを学習させ、テストデータに対して推論を行い感情極性辞書による手法の精度との比較を行うことを目的とした．ポジティブ、ネガティブの2ラベルでの推論の精度は96%であり、感情辞書による手法の精度である80%より高いことが分かった．また、感情極性辞書では難しいポジティブ、ネガティブ、ニュートラルの3ラベルでの推論の精度は85%であった．

6.2　今後の課題

本研究ではhuggingface社が提供しているBertForSequenceClassificationに手を加えずにモデルとして用いたが、より精度の高い手法とするにはデータセットに対して適切なモデルの改良を行うことが必要な課題である．また、少数のデータセットで十分な学習が行えることが分かったので、自作のデータセットを作成して実験を行いたい．

謝辞

本論文の作成にあたり、専門外の研究でありながら適切な助言を賜り、また丁寧に指導して下さった桂田准教授に心から感謝いたします.

また、研究に意欲的な研究室の皆様の存在が、研究を進めていく上で大きな励みとなりました. お礼申しあげます.

参考文献

[1] 感情解析ビジネスの市場規模予測,
https://www.es-jpn.jp/blog/%E3%83%9E%E3%83%BC%E3%82%B1%E3%83%83%E3%83%88/706/

[2] [自然言語処理] 感情分析の進め方&ハマりやすいポイント,
https://qiita.com/toshiyuki_tsutsui/items/604f92dbe6e20a18a17e

[3] 東山昌彦, 乾健太郎, 松本裕治, 述語の選択選好性に着目した名詞評価極性の獲得, 言語処理学会第 14 回年次大会論文集, pp. 584-587, 2008. / Masahiko Higashiyama, Kentaro Inui, Yuji Matsumoto. Learning Sentiment of Nouns from Selectional Preferences of Verbs and Adjectives, Proceedings of the 14th Annual Meeting of the Association for Natural Language Processing, pp. 584-587, 2008.

[4] 小林のぞみ, 乾健太郎, 松本裕治, 立石健二, 福島俊一. 意見抽出のための評価表現の収集. 自然言語処理, Vol.12, No.3, pp. 203-222, 2005. / Nozomi Kobayashi, Kentaro Inui, Yuji Matsumoto, Kenji Tateishi. Collecting Evaluative Expressions for Opinion Extraction, Journal of Natural Language Processing 12(3), pp. 203-222, 2005.

[5] Hiroya Takamura,Takashi Inui,and Manabu Okumura.Extracting semantic orientation of words using spin model. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, 2005.

[6] Devlin, Jacob, et al. “Bert: Pre-training of deep bidirectional transformers for language understanding.” arXiv preprint arXiv:1810.04805, 2018.

[7] 岡谷貴之 機械学習プロフェッショナルシリーズ 深層学習 改訂第 2 版, 講談社, 2022.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In Advances in Neural Information Processing Systems, 2017.

[9] 近江崇宏, 金田健太郎, 森長誠, 江間見亜利, BERT による自然言語処理入門 Transformers を使った実践プログラミング, オーム社, 2021.

[10] 林部祐太. 知識の整理のための根拠付き自然文間含意関係コーパスの構築. 言語処理学会第 26 回年次大会論文集, pp. 820-823. 2020.