ニューラルネットワークを用いた SNS における感情分析手法の提案

麦倉柊太

目次

1. はじめに

1.1 研究の背景

昨今のインターネットや SNS を用いることで、世界中の人々と瞬時に連絡を取ることが可能になり、人々の生活は豊かになった。一方で、近年 SNS における炎上や一方的な誹謗中傷というものが数多く発生するようになり、非常に注目を浴びている。炎上とは、 “Twitter や Facebook などの SNS での不用意な投稿が原因となって投稿者本人が非難に晒されたり、これらの SNS での消費者の投稿を契機として企業が予期せぬ非難に晒されたりする（現象）”[1]というものである。この炎上や誹謗中傷が社会や人に与えている影響は、大きい。例えば、炎上や誹謗中傷によって、人々の生活が従来のものとは大きく変化してしまったり、鬱を患ってしまったりすることが多々起こっている。

そのため、炎上や誹謗中傷に対する対処は SNS における重大な問題であるがゆえに、工学的にも多くの研究が取り組まれている。しかし、SNS を含む自然言語処理には数多くの問題点がある。

その中で大きいものが感情推定問題である。吉野ら[2]によると、炎上や誹謗中傷が起こる際にはその特性上、その対象者を中心に批判の情報が伝搬することが多い。そのため、SNS における炎上や誹謗中傷をコンピュータに認識させるために、投稿された文章の感情を推定することが非常に有用である。そのための代表的なものとして Lei Zhang ら [3]によって”感情分析を機械学習で行う”という手法が提案されている。

しかし SNS においては、人々が自由に発信できるという特性上、他の文章媒体と比べスラングなどのノイズとなる情報が多くなる。炎上分析においては、この SNS においても用いることのできる感情推定手法が必要であるため、SNS においてノイズを除去し、適切に感情分析を行う手法が必要である。

1.2 研究の目的

本研究においては、Deep Neural Network(DNN)という手法を用いることで、SNS において感情分析を行うことを最終目標とする。そのために本研究では、公開されている Twitter データに感情情報を付与したデータセットを用いた学習データ利用する。そしてそれを用いることで DNN を学習させる。そして、その際に用いた手法に対し評価を行うことで、SNS に対してもより良い感情分析を行うことのできる手法を検討する。

1.3　本論文の構成

本論文は以下のような構成になっている。

第 1 章は本章であり、研究の背景や目的について述べた。

第 2 章では、本論文中で用いた提案手法の説明や、それに関連した従来手法の説明を行う。

第 3 章では、提案手法を用いて実装を行い、それに対して評価を行う。

第 4 章では、第 3 章の評価を元にして考察を行う。

第 5 章では、本研究のまとめを行い、今後の展望を述べる。

2. 提案手法とそれに関する従来手法

2.1　提案手法の概要

本研究においては、下記のような情報を取得済みであることを想定する。

DNN の学習時

英語で書かれた文章。また、その文章が Positive な文章であるか Negative な文章であるかの情報。

DNN を実用するとき(テスト時)

入力として、英語で書かれた文章

今回の目的は DNN や次元削減手法の構築であるため、その前段階である形態素解析(詳細は後述する)を精度高くかつ容易にするため、入力を英語の文章に限定した。

本研究ではテスト時に、入力した文章が Positive な文章か Negative な文章かを出力として取得することを目標とする。このような入出力を学習するための DNN を検討する。

本研究で用いる提案手法の流れを図 1 に記す。

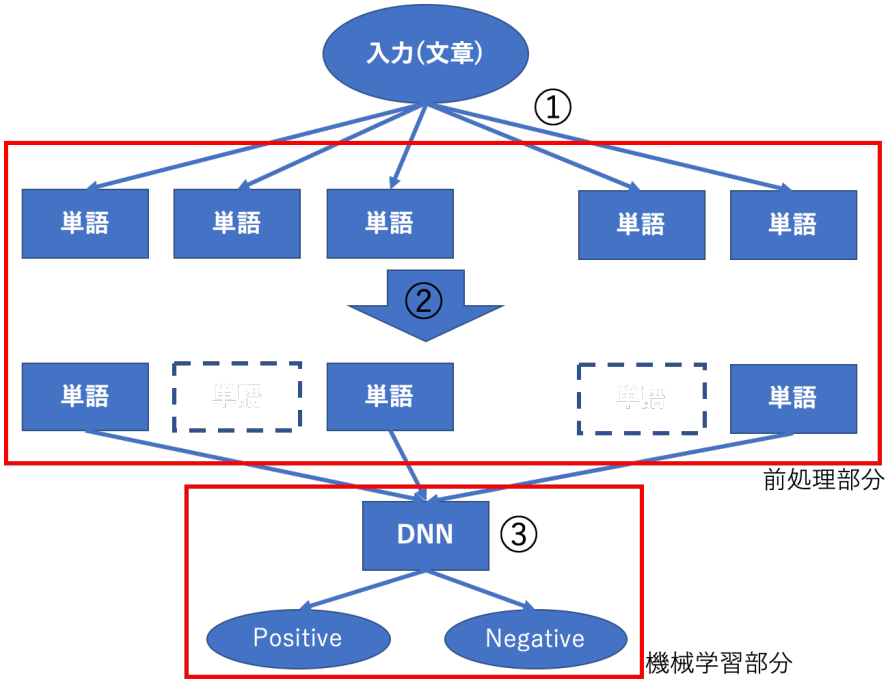


図 1　提案手法の流れ

図 1 に示したように、提案手法は主に下記の 3 つに分けられる。
① 入力から文章から単語を抽出する
② ①で抽出した単語から、機械学習に使用する単語を選定する
③ 機械学習を行い、DNN を学習させる

今回は、①と②を前処理部分、③を機械学習部分と呼ぶことにする。次節より、それぞれの具体的手順や役割について紹介する。

2.2 前処理部分

先に述べたとおり、この前処理部分は、主に下記の 2 手順に分けられる。
① 入力した文章から単語を抽出する
② ①で学習した単語から、機械学習に称する単語を選定する。

まず、①に関して説明する。ここでは、主にコーパスを用いて形態素解析が行われる。形態素解析とは、工藤[4]によると、文の中で最小の意味単位である形態素をもとにして、文章を形態素に分割するプロセスことである。
例えば、”I have a pen”という文章があった場合にその文章を構成する最小単位要素(今回の場合は”I” “have” “a” “pen”)に分解する過程のことである。英語であれば基本的にスペースで各単語が区切られるため、今回は英語の文章を対象にした。

次に、②に関して説明する。ここでは、DNN に入力する単語に成約を設けることで、自由度の削減や、ノイズ情報の除去を行う。手順としては、下記の 5 個の手順に分けられる。

(1) 冠詞や前置詞など、感情に関わりづらい情報の除去を行う
(2) それぞれのトークンに対し、正規化を行う
(3) トークンから記号を除去する
(4) 長過ぎる単語や短すぎる単語の除去を行う
(5) 登場頻度の少ない単語を除去する

これを行うことによって、感情に直接寄与しにくい要素(一般的に冠詞や前置詞、また、”I”や”me”など短い単語はストップワードと呼ばれ、感情表しにくいと想定されている)を除去することができる。また、登場頻度が極端に低いものは除去することによって、誤字や SNS に登場するスラングなどを除去しやすくなる。このようなスラングはノイズ源となりがちであり、このような処理を行うことで、ノイズを少なくした上で機械学習を行うことができる。またこの処理によって、DNN の入力に用いる単語の種

類を減らすことができる。

本来であれば、出現する単語が未知である SNS の文章においては、人間が日常的に使用することのある全ての単語が探索対象である。佐藤ら[4]によると、日本人大学生の入学時における日本語語彙数は約 40,000 万語前後であると述べられている。これは、英語でも同等であると思われ、入力単語種類の自由度は、最低でも 40,000 個ということになる。しかし、先に述べたような次元圧縮の処理を行うことによって、この自由度を大幅に下げることが可能になるのである。これによって、ノイズによる出力の影響を減らすことが出来る。また自由度を下げることにより、2.3 で説明する過学習を減らすことが出来ると考えられる。

以上の 2 点を、事前処理として DNN に入力する前に行う。

2.3 機械学習部分

今回用いる DNN とは、一般的に図 2 のように記述できる。

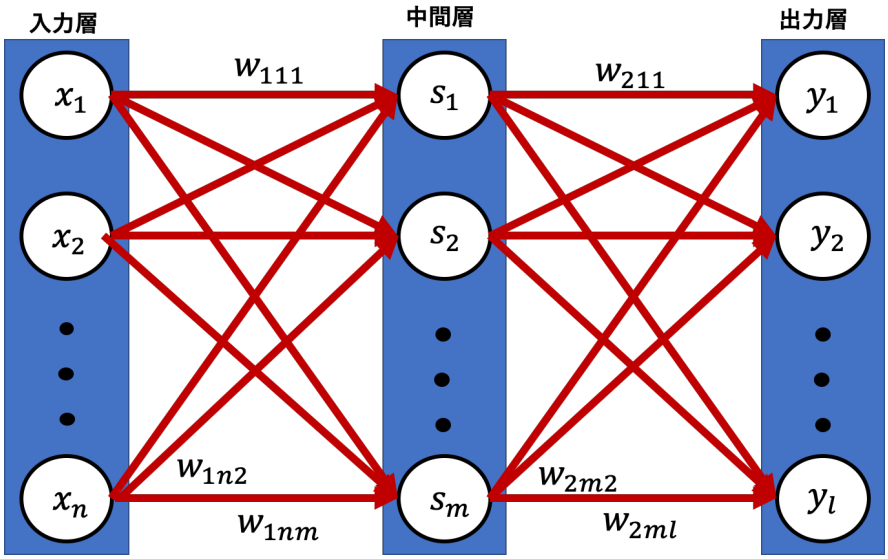


図 2　DNN の概要図

なお、中間層は複数存在する場合もある。DNN のステップは、学習ステップと推定ステップに分けられる。

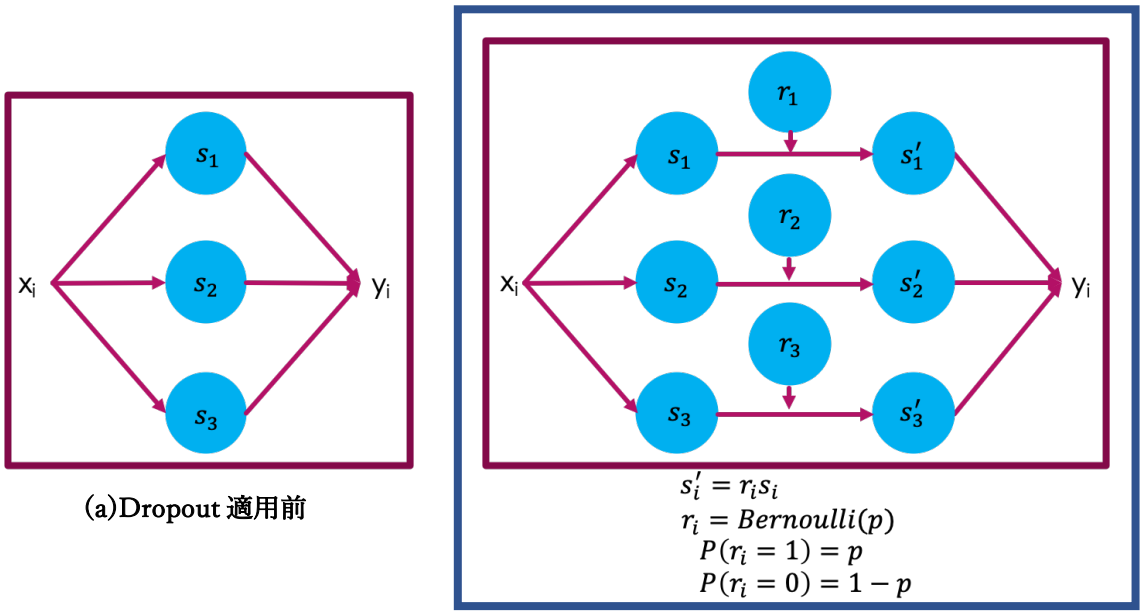
学習ステップでは、入出力データが学習データとして与えられ、それを適切に表現するように各ノードの学習が行われる。各ノード学習時には、誤差伝搬法を用いて学習を行う。この手法では、入力層に与えたデータによって出力層に生じた出力の値と、その入力によって本来出力されるはずであった出力との誤差を取る。そして、それを各ノードに伝搬することによって、各ノードの値を修正していく手法である。この手法では、

各ノード間のパラメータは式(1)のようにして修正される。

$$w_{ijk}^{k+1} \leftarrow w_{ijk}^{k} - \sigma \frac{\partial E}{\partial w_{ijk}} \quad \text{但し } E = \sum (y_i - u_i)^2 \quad u_i：\text{理想の出力} \qquad \text{式(1)}$$

このとき、中間層の数が多いほど過学習という現象が起きやすいことに、注意が必要である。過学習とは、ネットワークが学習データに対して学習されている一方で、未知のデータ(テストデータ)に対して適合しなくなるという現象である。今回の場合、目的は未知のデータを扱うことにあるため、この現象による影響を減らす必要がある。

そのため、今回の研究においては、Srivastava ら[6]によって提案された Dropout という手法を採用する。図 3 に Dropout の概要図を示す。



(a)Dropout 適用前

(b)Dropout 適用後

図 3　Dropout の説明

Dropout とは、過学習を防ぐための手法の 1 つである。過学習は、入力の数に対しノードの数が多いことで、ノードの自由度が高いために生じる。そのため、学習時においてノードの活動をある確率 p で止めることで過学習を防ぐことが、Dropout の目的である。

つまり、本来図 3(a)のように全てのノードが活性化しているものに対し、確率 p にしたがったベルヌーイ分布をかけることで、ノードを不活性化するということである。これによって、学習時のノードの自由度をさげることが可能になる。

一方で推定ステップにおいては、学習ステップで得られたネットワークに対し入力の

みが与えられて、それに対する出力が計算される。一般には、その出力と本来出力されるはずであった出力とを比較することにより、ネットワークの精度などを計算することで、ネットワークの評価が行われる。

また、本研究については他にも注意しなければならない点が存在する。それは、言語は、単語の出現する順番によって意味が変わってくるという点だ。例えば

Toru kissed Tama

Tama kissed Toru

この 2 つの英文においては、 同一の単語が登場してくるが、意味は異なってくる。このように、文章においては単語の順番に情報があるため、文章を用いた感情分析を行う際には単語の順序を配慮する必要がある。

そのため本研究では、DNN のアーキテクチャとして、LSTM(Long short-term memory)を採用する。LSTM とは図 4 に示されたようなネットワークである。この図において、最終的な出力は h_m である。

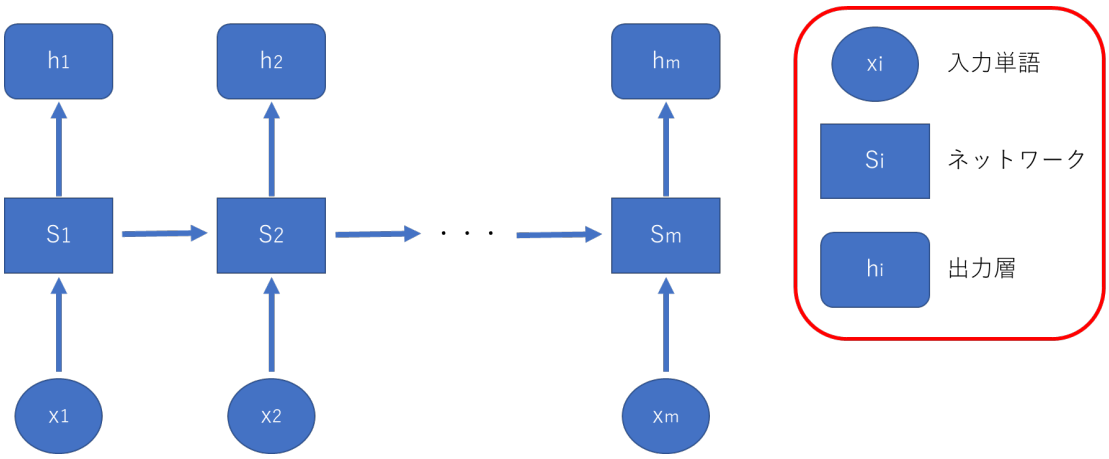


図 4　LSTM

このネットワークでは、その時点で入力されたデータだけでなく、それより前の段階で入力されたデータも考慮することができる。これによって、過去の入力データを出力に反映させることができるため、今回の研究には最適な DNN だと考える。

3. 提案手法の実装と評価

3.1　実装環境

今回の研究は、表 1 の環境で実装を行った。

表 1　実装環境構成

コンピュータ	ASUS Comupter inc. X541UAK
プロセッサ	Intel Core i7-7500U
メモリ	4GB
グラフィックス	Intel HD Graphics 620
OS	Windows 10 Home (Build 18362)
開発環境	Python 3.7.7 Tensorflow 2.2.0 Anaconda 4.8.2

またデータセットとして、Twtter のデータに感情情報を付与したものとして” Sentiment140 dataset with 1.6 million tweets”[8]を、SNS 以外の環境で取得されたデータセットとして、映画のレビューに対し感情情報を付与した” IMDb データセット“[9]を使用した。

3.2　実験の構成

本研究においては、図 5 のような構成を考える。

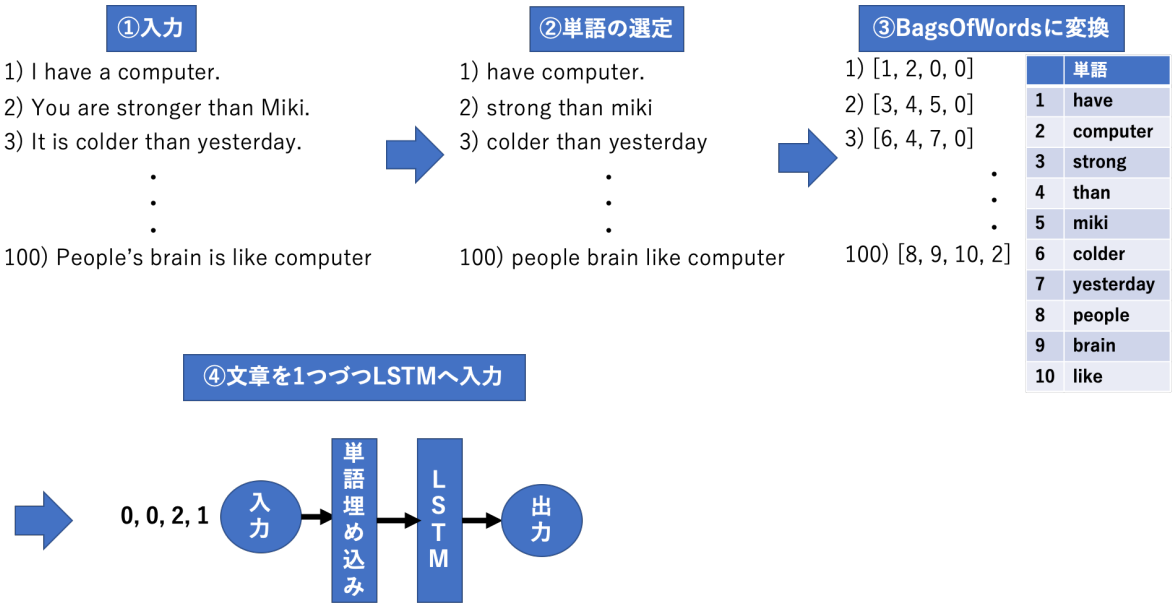


図 5　本実験における構成

まず、入力した文章から単語の選定を行う。次に、選定された単語を BagsOfWords の形式に変換を行う。これによって、単語を数値の形にして表すことができ、かつ、共通の単語を同じ番号で、コンピュータが認識することができるようになる。なおこの際に、各文章を N 個からなるシーケンスデータに変換している。例えば、図 5 においては N=5 である。そして最終的に、LSTM に対しこのシーケンスデータが入力されることで、結果が出力される。

この際 N の値の設定には注意が必要である。LSTM の特性上、単語は必ず N 個入力しなければいけないため、文章の単語数が N よりも少なかった場合、不足分は 0 でパディングされる。しかし 0 でパディングする部分は本来不要な部分であるため、ノイズになりうる。一方で、文章の単語数が N よりも多かった場合、N 個を越した部分は切り捨てられる。そのため、その分情報が抜け落ちることになる。

後者においては、その切り捨てられた単語に感情推定を行う上での重要なデータが含まれていた場合、そのデータが切り捨てられてしまうことになる。そのため一般的には、パディングによるノイズより、切り捨てることによる情報損失の方が出力結果に与える影響は大きい。

よって今回は、学習データとして与える文書群の中で、最大の単語数を持つ文章の単語数を N とすることによって、学習時にはすべての単語を入力する。なおテストデータにおいては、N 個より多い単語は切り捨てられ、一方で単語数が N 個未満の文章においては 0 がパディングされることで、N 個の配列となる。

このような構成を用いて、本研究を進める。

3.3　本研究の評価軸

本研究では本研究の目的に沿って、次の 2 個の評価軸に沿って、SNS における感情分析を行うために提案したネットワークの評価を行う。

① ノイズ除去や次元圧縮のために行った事前処理が出力結果に与える影響

本研究においては、誰でも自由に投稿できるが故に SNS において生じるノイズを減らすために、単語の選定を行った。しかし、単語の選定を行うとノイズ以外の文章の情報も減少するので、それによって出力結果に大きな影響が生じないかを検討する。

② 他の媒体をターゲットとしたデータセットとの比較

今回の研究では、SNS の文章は感情分析を行う上ではノイズが多いため、感情分析が較的難しいということから、行われている。一方で、DNN を用いて映画レビュー(IMDb データセット)に対して感情分析を行った際の評価は、Yoon Kim[5]によってなされており、高い正解率を示している。

そのため今回は、IMDb データセットを用いた際の正答率と Twitter データに対する正答率を比較することで、ノイズの多い Twitter に対し感情分析を行ったときにおける、システムの評価を行う。

3.4　評価を行う手法

① ノイズ除去や次元圧縮のために行った事前処理が出力結果に与える影響

今回は、全文章中で登場した回数が少ない単語を文章から除去することによって、単語種類の削減を行った。その際のパラメータとして、全文章中で何回以上登場した単語を残すかというものがある。それを変化させたときに正答率にどのような影響が出るかを検討する。

② 他の媒体をターゲットとしたデータセットとの比較

今回は、SNS 以外からの媒体によるデータセットとして、映画レビューに対して感情データが割り当てられたものを用いる。DNN の構成など条件を同一にして、Twitter のデータセットと映画レビューのデータセットそれぞれに対し学習や出力の取得を行い、評価を行う。

なお、映画レビューのデータセットでは単語数が 300 近い文章が多い一方、Twitter レビューのデータセットでは単語数が 30 程度の文章が多い。これは、Twitter において 280 文字までしか入力できないことが起因している。

この条件による差異を減らすために、映画レビューのデータセットではそれぞれの文章において、頭 50 単語のみを DNN に入力し、以下の単語を切り捨てる。

これは、同一の文章内はどの部分を切り抜いても、同一の感情を示している(文章の途中で感情が変化していない)ことを仮定している。例えば、頭 50 単語では Positive の感情が現れていたとしたら、以下の切り捨てられた単語でも同様に Positive の感情が現れているということだ。

この仮定の妥当性は、学習データに対する DNN の正答率によって検証する。

3.5 評価結果

本章では、実際にネットワークを組み学習させた結果、テストデータに対して成功率がどのようになったかについて述べる。なお、実際に用いた Python プログラムは付録に記載した。

今回ネットワークにおけるエポック数(同一の学習データを何回学習させるかを表す)は 20 とした。また、Dropout の活性確率である p は 0.5 と設定した。これは、Srivastava ら[6]によって、「ほとんどの場合において 0.5 が最適である」と示されているためである。

① ノイズ除去や次元圧縮のために行った事前処理が出力結果に与える影響

単語の最低限登場数(min と記す)をそれぞれ変化させた場合の正答率の結果を、表 2 に記した。

表 2　入力単語の次元削減が出力結果に与える影響

min	削減後の 単語の種類数	学習データの 正答率	テストデータの 正答率
0	53,468	95.10	72.79
5	10,400	87.47	73.44
10	6,293	84.77	73.76
50	1,854	79.04	75.06
100	1,024	76.12	74.22
500	237	68.86	69.01
1,000	94	60.85	65.38

② 他の媒体をターゲットとしたデータセットとの比較

Twitter のデータセットを認識させた場合の正答率と、映画データセットを認識させた場合の正答率を比較する。結果を表 3 に記す。

表 3　データセットによる正答率の違い

データの種類	削減後の 単語の種類数	学習データの 正答率	テストデータの 正答率
Twitter	1,854	79.04	75.06
映画	3,131	98.21	77.34

なお、映画のデータセットでは学習データに対する正答率が 98.21%であり、高い精度となった。このことよりこのデータセットにおいては、2.2 で仮定した「同一の文章内はどの部分を切り抜いても、同一の感情を示している(文章の途中で感情が変化していない)」ということに正当性があることを示せた。

4. 考察

実験①より、単語の種類数を適切に削減した場合において、成功率がやや増加したことが分かる。これはつまり、単語数を削減することによって、SNS に登場している、自然言語処理を行う上でノイズとなっていた単語が除去されたということであろう。一方で、入力単語の種類数を過度に削減した場合には、誤差率がより大きくなった。これは、削減した単語の中に、感情を分析する上で重要な情報をもつ単語が含まれていたことを示している。

そのため単語の削減を行う際には、感情分析をする上で重要な情報を残しつつも、ノイズが除去できるような適切な値を選定する必要がある。

また実験②より、今回提案した手法を用いることで、Twitter などのノイズが多い情報源においても、映画レビューなどノイズの少ないデータセットを用いた場合と、ほぼ同等の精度が出ることが確認できた。これによって SNS における負の情報をより精度良く認識することができる。そのため、SNS における炎上分析の分野へ精度良く応用できることが期待できる。

しかし、映画レビューを用いた方が依然として高い精度が出るのも事実である。また映画レビューでは、1 文あたりの単語数が多いため、Twitter のデータベースよりもより多くのデータを得ることができる。そのため、より効率的に学習することができる。

そのため、今後は転移学習を用いて学習することも検討する必要があるだろう。

転移学習とは、”ある領域で学習された知識を適用することによって、新規タスクの効果的な仮説を導く手法”[10]である。つまり今回の事例においては、まず映画デビューなど多くの情報量があるデータを用いてニューラルネットワークの学習を行う。そしてそのデータをもとにして、Twitter のデータセットの解析を行う。これによって、より効率的かつ精度良く学習することができると考えられる。

5. おわりに

今回の手法を用いることで、入力される単語にスラングなどが多いため、自然言語処理においてはノイズとなる可能性の高い情報を持つ SNS においても、他の低ノイズ媒体における文章と同じ程度の精度で、感情分析が行えることが確認できた。これを利用することで、SNS においても精度よく感情分析を行えることが期待できる。

一方で、より精度よく SNS で感情分析を行うためには、感情データも含んだ学習データの取得が大きな問題になってくる。例えば Twitter は仕様上 1 文あたりの文字数が 280 以下であり、複数文に分かれて感情が表現されていることも多いため、一見すると感情を推定しづらいためだ。またそれによって、SNS への投稿に対し感情情報を入れたデータセットが少ないこともある。

この問題に対し、今後は転移学習を用いることを検討したい。転移学習を用いることで、他の媒体におけるデータセットで学習させたネットワーク情報を応用し、SNS に対する感情分析ネットワークを構築できる。これによって、SNS の投稿に対して感情情報を入れたデータセットが少量であった場合でも、より精度の良いネットワークを構築することができる。

また、より精度を上げるために、今回提案したノイズ除去手法を用いた上で、BERT というアーキテクチャを用いることを検討したい。

BERT とは、Jacob ら[11]によって提案された、自然言語処理のために用いられる機械学習アーキテクチャの 1 種である。BERT を用いて一般的な文章に対し感情分析を行った結果、精度が 94%であることが示されている。

BERT に入力する Twitter の情報もノイズが多いことが想定されるが、今回の手法を用いることで、精度高く感情分析を行えることが期待できる。

6. 参考文献

[1] 総務省 (2015) 『平成 27 年版 情報通信白書』 https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/html/nc242210.html (閲覧日：2020 年 6 月 24 日)

[2] 吉野ヒロ子, 小山晋一, 高田倫子(2018). 「ネット「炎上」における情報・感情拡散の特徴 : Twitter への投稿データの内容分析から」 『広報研究』第 22 号 pp. 60-78

[3] Lei Zhang, Shuai Wang, Bing Liu (2018), “Deep Learning for Sentiment Analysis: A Survey”, arXiv:1801.07883v2

[4] 佐藤尚子, 田島ますみ, 橋本美香, 松下達彦, 笹尾洋介(2017)「使用頻度に基づく日本語語彙サイズテストの開発 —50,000 語レベルまでの測定の試み—」『千葉大学国際教養学研究』 pp.15-25
[5] 西條由貴男(2017)『起業のツボとコツがゼッタイにわかる本』 秀和システム, p.84
[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014), "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15 (2014)
[7] Steven L. Brunton, J. Nathan Kutz (2019), "Data-Driven Science and Engineering Machine Learning. Dynamical Systems, and Control", Cambridge University Press, p.195-226
[8] Go, A., Bhayani, R. and Huang, L. (2009)," Twitter sentiment classification using distant supervision.", CS224N Project Report, Stanford, 1(2009), p.12.
[9] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts (2011), "Learning word vectors for sentiment analysis.", ACL (2011)
[10] 神嶌敏弘(2010)『転移学習』 人工知能学会誌 25(4), p.572-580
[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee Kristina,Toutanov(2018). "BERT: Pre-training of deep bidirectional transformers for language understanding". arXiv preprint arXiv:1810.04805, 2018.

付録

(1) 使用したプログラム

```
import tensorflow as tf
import numpy as np
import csv
import pandas as pd
import random
import numpy.random as nr
import sys
import h5py
import math
import MeCab
import re
import nltk
```

```
from gensim import corpora
from gensim import corpora, matutils
from keras.optimizers import SGD
import neologdn
nltk.download('stopwords')
nltk.download('punkt')
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

def plot_graphs(history, metric):
  plt.plot(history.history[metric])
  plt.plot(history.history['val_'+metric], '')
  plt.xlabel("Epochs")
  plt.ylabel(metric)
  plt.legend([metric, 'val_'+metric])
  plt.show()

#データ入力
data=pd.read_csv('training.1600000.processed.noemoticon.csv', encoding = "ISO-8859-
1",header=None)
Sentence_before=data[5].values.tolist()
Value_before=data[0].values.tolist()

learn_length=500 #学習データの長さ/2
test_length=500 #テストデータの長さ/2
Sentence_learn=Sentence_before[0:learn_length]
Sentence_learn.extend(Sentence_before[len(Sentence_before)-learn_length-
1:len(Sentence_before)-1])
print(Sentence_learn[0])
Value_learn=Value_before[0:learn_length]
Value_learn.extend(Value_before[len(Value_before)-learn_length-1:len(Value_before)-1])
print(type(Value_learn[0]))
Sentence_test=Sentence_before[learn_length:test_length+learn_length]
Sentence_test.extend(Sentence_before[len(Sentence_before)-learn_length-test_length-
1:len(Sentence_before)-learn_length-1])
Value_test=Value_before[learn_length:learn_length+test_length]
```

```
Value_test.extend(Value_before[len(Value_before)-learn_length-test_length-
1:len(Value_before)-learn_length-1])

#テキストの正規化
Sentence_learn_before=[]
for i in range(len(Sentence_learn)):
    Sentence=[]
    Sentence_learn[i]=neologdn.normalize(Sentence_learn[i])
    Sentence_learn[i]= re.sub(r'https?://[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '',
Sentence_learn[i])
    Sentence_learn[i]= re.sub(r'http?://[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '',
Sentence_learn[i])
    Sentence_learn[i]= re.sub(r'@/[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '',
Sentence_learn[i])
    Sentence_learn[i]= re.sub(r'@[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '', Sentence_learn[i])
    Sentence_learn[i] = re.sub(r'[!-/:-@[-`{-~]', r' ', Sentence_learn[i])
    Sentence_learn[i]=Sentence_learn[i].lower()
    Sentence=nltk.word_tokenize(Sentence_learn[i])
    Sentence_learn_before.append(Sentence)
Sentence_learn=[]
Sentence_learn=Sentence_learn_before
Sentence_learn_before=[]

Sentence_test_before=[]
for i in range(len(Sentence_test)):
 Sentence=[]
 Sentence_test[i]=neologdn.normalize(Sentence_test[i])
 Sentence_test[i]=re.sub(r'https?://[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+','',Sentence_test[i])
  Sentence_test[i]=re.sub(r'http?://[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+','',Sentence_test[i])
 Sentence_test[i]= re.sub(r'@/[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '', Sentence_test[i])
  Sentence_test[i]= re.sub(r'@[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-]+', '', Sentence_test[i])
  Sentence_test[i] = re.sub(r'[!-/:-@[-`{-~]', r' ', Sentence_test[i])
  Sentence_test[i]=Sentence_test[i].lower()
  Sentence=nltk.word_tokenize(Sentence_test[i])
  Sentence_test_before.append(Sentence)
Sentence_test=[]
```

```
Sentence_test=Sentence_test_before
Sentence_test_before=[]

#感情に寄与しにくい単語(ストップワード)を除去
from nltk.corpus import stopwords
en_stops = set(stopwords.words('english'))
Sentence_learn_check=[]
Sentence_test_check=[]

for i in range(len(Sentence_learn)):
    insert=[]
    for j in range(len(Sentence_learn[i])):
        word=Sentence_learn[i][j]
        if  word not  in en_stops:
            insert.append(word)

    Sentence_learn_check.append(insert)

for i in range(len(Sentence_test)):
    insert=[]
    for j in range(len(Sentence_test[i])):
        word=Sentence_test[i][j]
        if  word not  in en_stops:
            insert.append(word)

    Sentence_test_check.append(insert)

#BagOfWords に変換
dictionary = corpora.Dictionary(Sentence_learn_check)
print(len(dictionary.token2id))
#no_below で最低出現文書数を指定(単語削減)
dictionary.filter_extremes(no_below=10,no_above=1)

dict_inside=list(dictionary.token2id.keys())
dict_set= set(dict_inside)
dict_num=len(dictionary.token2id)
```

```
print(len(dictionary.token2id))

dictionary.save_as_text('Language_Processing,txt')


#出力側の値定義
Value_Teach=[]
Value_test_Teach=[]
#0 のときが negative,0 のときが positive
for i in range(len(Value_learn)):
    if Value_learn[i]==0:
        Value_Teach.append(0)
    else:
        Value_Teach.append(1)

for i in range(len(Value_test)):
    if Value_test[i]==0:
        Value_test_Teach.append(0)
    else:
        Value_test_Teach.append(1)

#単語の次元削減
Sentence_Learn_Final=[]
train_dataset=[]
for i in range(len(Sentence_learn_check)):
    insert=[]
    for j in range(len(Sentence_learn_check[i])):
        word=Sentence_learn_check[i][j]
        if word    in dict_set:
            insert.append(word)
    Sentence_Learn_Final.append(insert)

for i in range(len(Sentence_Learn_Final)):
    insert=[]
    for j in range(len(Sentence_Learn_Final[i])):
        word=Sentence_Learn_Final[i][j]
```

```
            insert.append(dictionary.token2id[word])
        train_dataset.append(insert)

Sentence_Test_Final=[]
test_dataset=[]
for i in range(len(Sentence_test_check)):
        insert=[]
        for j in range(len(Sentence_test_check[i])):
            word=Sentence_test_check[i][j]
            if word    in dict_set:
                insert.append(word)
        Sentence_Test_Final.append(insert)

for i in range(len(Sentence_Test_Final)):
        insert=[]
        for j in range(len(Sentence_Test_Final[i])):
            word=Sentence_Test_Final[i][j]
            insert.append(dictionary.token2id[word])
        test_dataset.append(insert)

max=0
for i in range(len(train_dataset)):
        s=len(train_dataset[i])
        if max<s:
            max=s

size=max
#DNN の定義
model_emb=10

model = tf.keras.Sequential([
        tf.keras.layers.Embedding(dict_num, model_emb),
        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(size)),
        tf.keras.layers.Dense(size, activation='relu'),
        tf.keras.layers.Dropout(0.4, noise_shape=None, seed=None),
        tf.keras.layers.Dense(2,activation='softmax')
```

```
])
model.summary()

#1 文章あたりの単語数調整(Pudding など)
for i in range(len(train_dataset)):
    tasu=size-len(train_dataset[i])

    if(tasu<0):
        train_dataset[i].pop()
    tasu=size-len(train_dataset[i])

    for j in range(tasu):
        train_dataset[i].append(0)

for i in range(len(test_dataset)):
    tasu=size-len(test_dataset[i])

    if(tasu<0):
        for j in range((-1)*tasu):
            test_dataset[i].pop()
    tasu=size-len(test_dataset[i])

    for j in range(tasu):
        test_dataset[i].append(0)

#ネットワークに入力するために型変換
inputData_train = np.array(train_dataset)
inputValue_train= np.array(Value_Teach)
inputData_test = np.array(test_dataset)
inputValue_test= np.array(Value_test_Teach)
print(len(Value_Teach))
num=len(test_dataset[0])
for i in range(len(test_dataset)):
    if(len(test_dataset[i])!=num):
        print(num)
        print(len(test_dataset[i]))
```

```
#モデルを学習させる
learning_rate=0.01
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy'])
epoch=3
batch_size = 100
history=model.fit(inputData_train, inputValue_train, epochs=20, batch_size=200)

#学習の評価
history.history

#テストとその評価
test_loss,test_acc=model.evaluate(inputData_test,inputValue_test,verbose=0)
print(test_acc)
```