エンジニアリングデザイン演習・G1
アマゾンレビューのサクラ判定

195702D 奥間 涼
195748B 徳里 光陽
195761K 塩月 流星
195769E 佐藤 和周
195770J 栗原 瑠威

2022 年 2 月 1 日

目次

1　テーマ

近年オンラインショップを活用する人々が増加している。一方でサクラレビューによって偽物、欠陥や欠品のある商品が良い商品として扱われ、その結果そのレビューを見た購入者へ本来求めている商品が届かないケースもあり、サイト運営社は対応に追われている [5]。購入者が求めている商品が届かないということは、購入者の金銭や時間の損失だけではなく質の悪い商品が出回ることで社会に悪影響を与え得ることが考えられる。

特に有名なオンラインショッピングサイトである Amazon に着目し、その中からサクラレビューを中心にサクラ判定を行っていこうと考えた。

よって本グループでは『Amazon レビューのサクラ判定』というテーマを設定しそれを元に実験を行った。

2　目的

今回の実験を通して、サクラレビューの特徴を獲得し実際のレビューに適用できるモデルを作成する。

3　目標

サクラレビューの認識率 70% 以上のモデルを作成する。

4　アプローチの全体像

スクレイピングによって集めた Amazon のレビューに 5 段階評価の教師ラベルを各自の評価で付与し、spaCy、BERT でベクトル化したデータセットを用いる (詳細は 6 節データセットの構築方法)。

作成したデータセットを RandomForest で学習し，spaCy，BERT での認識率の違いを調べた。

5　予定していた実験計画

本実験では当初，amazon やサクラチェッカー [3] のサイトを用いて，サクラレビューが多いと判断された商品の URL を取得し、スクレイピングでデータの収集を収集する。そして 5 段階評価のラベルをつけ、データセットを作成。spaCy、BERT によるベクトル化を用いて学習を行い、ハイパーパラメータを調整することで精度を上げるという流れを計画していた。しかし，5 段階評価での分類では思うように精度が上がらないという問題に直面したため，5 段階評価を 3 段階評価にしたパターンでの分類も行った。

6　データセットの構築方法

Amazon サイトからレビュー数が 100 以上ある商品の URL を探しサクラチェッカー [3] で検索をかける。検索すると、レビューの内容やレビューが書き込まれた日付の偏り，評価の分布なども加味して，その商品がサクラ度 (サクラレビューが多い可能性) が分かるため，サクラ度が高いと判定された商品の URL を集めた。

BeautifulSoup4(以下 bs4) と selenium[6] を用いて、収集した URL からレビュー毎に csv ファイルへスクレイピング [2] した。

bs4 の機能を用いることで以下の様にタグからデータを取得した。

- .product-title = 商品の名前
- .averageStarRating = 商品の平均評価
- .review-title-content = レビュータイトル
- .review-rating = レビュー評価
- .review-text = レビュー本文

サクラチェッカーは，全てのレビューや評価から商品が危険かどうかを判定するサイトなので，プログラムにより取得したデータセット (一つ一つのレビュー) はサクラかどうかのラベルが付与されていない状態である。そこでメンバー 3 名がレビューの内容を確認し、各々が 5 段階で評価してその平均値をラベルとして付与した。なお、ラベルは学習に用いる際に int 型にキャストして用いた。

5 段階評価の評価基準 (1〜2:サクラ: 3: どちらとも言えない: 4〜5: 信頼できる):

表 1　5 段階評価基準

ラベル	レビュー内容
1	不自然な日本語で書かれていないか
2	未使用でレビューしている文 (例:”まだ使っていないですが期待を込めて星 5 です”)
3	コピペ文のようなレビュー (例:”すごく良い”，”使いやすい”，”特になし”など)
4	商品についての説明，評価理由などが簡潔に書かれている
5	商品についての説明，評価理由などが詳細に書かれている

ラベル付されたデータを BERT と spaCy の 2 種類でベクトル化したものをデータセットとした。

また，3 値分類で使用する際，1〜2.5 ⇒ 1 : 2.5〜3.5 ⇒ 2 : 3.5〜5 ⇒ 3 にラベルを再付与して分類する。下表 2,3 にラベル毎のデータ数を示す。

表 2　spaCy の 3 値分類で用いたデータセットの各ラベルの個数

ラベル	データ数
1(サクラレビュー)	32
2 (どちらとも言えない)	97
3(信頼できるレビュー)	379

表 3　BERT の 3 値分類で用いたデータセットの各ラベルの個数

ラベル	データ数
1(サクラレビュー)	69
2 (どちらとも言えない)	97
3(信頼できるレビュー)	394

7　機械学習の進め方

まず、自分たちで作成したデータセットの文章の部分を比較用に spicy と bert でベクトル化をし、それをランダムフォレストを用いて学習を進めていく。BERT は、日本語 Wikipedia で学習を進めたモデルを利用した [7]。spaCy のベクトル化の際に用いた学習済みモデルは GiNZA の日本語モデル”ja-ginza”を用いた [1]。https://megagonlabs.github.io/ginza/

8　実験結果

学習を行なった結果を spaCy と BERT の場合で示す。

8.1　spaCy(サンプル数 508 個，ベクトル数 508 個)

表 4　テストデータと訓練データの割合によるモデルの精度の変化 (5 値分類)
train_test_split(random_state=1),forest(max_depth=None,random_state=1)

test_size(割合)	精度
0.3	0.5228
0.4	0.5196
0.5	0.5157

ランダムフォレストの最大深さ max_depth を変化した際にモデルの精度がどの程度変化するか調べた [4]。その結果を表 4 に示す。

表 5　ランダムフォレストのハイパーパラメータ (深さの最大値) の変化に対するモデルの精度の変化 (5 値分類)
train_test_split(test_size=0.4, random_state=1),forest(random_state=1)

max_depth	精度
None(default)	0.5229
3	0.5294
4	0.5098
5	0.5294
6	0.5228

表 4,5 より、spacy を用いてベクトル化したデータセットを用いて学習を行なったが、思うような結果が得られなかった。そこで、教師ラベルを 5 個から 3 個に減らして再度学習を試みた。その結果を表 6,7 に示す。

表 6　テストデータと訓練データの割合によるモデルの精度の変化 (3 値分類)
train_test_split(random_state=1),forest(max_depth=None,random_state=1)

test_size(割合)	精度
0.3	0.7516
0.4	0.7647
0.5	0.7519

表 7　ランダムフォレストのハイパーパラメータ (深さの最大値) の変化に対するモデルの精度の変化 (3 値分類)
train_test_split(test_size=0.4, random_state=1),forest(random_state=1)

max_depth	精度
None(default)	0.7647
3	0.7549
4	0.7500
5	0.7647
6	0.7402

表 4,5,6,7 より，5 値分類より 3 値分類の方が精度が高かった。

8.2　BERT(サンプル数 560 個，ベクトル数 560 個)

＊修正前は、768 と書いてあったが、確認してみたところ、同一のベクトルが誤って複数回書き込まれてしまっていたため、そのようになってしまっていた。確認のため、もう一度レビューの本

文をベクトル化してみたところ、サンプル数とベクトル数の数は一致していた。

表 8　テストデータと訓練データの割合によるモデルの精度の変化 (5 値分類)
train_test_split(random_state=1),forest(max_depth=None,random_state=1)

test_size(割合)	精度
0.3	0.5357
0.4	0.5893
0.5	0.5214

表 9　ランダムフォレストのハイパーパラメータ (深さの最大値) の変化に対するモデルの精度の変化 (5 値分類)
train_test_split(test_size=0.4, random_state=1),forest(random_state=1)

max_depth	精度
None(default)	0.5893
3	0.5580
4	0.5625
5	0.5714
6	0.5625

表 8,9 より，spaCy の結果と比べると精度が 5% 以上良くなっていることがわかる。しかし、5 値分類では spaCy と同様に思うような結果が得られなかった。

表 10　テストデータと訓練データの割合によるモデルの精度の変化 (3 値分類)
train_test_split(random_state=1),forest(max_depth=None,random_state=1)

test_size(割合)	精度
0.3	0.7976
0.4	0.7991
0.5	0.7893

表 11　ランダムフォレストのハイパーパラメータ (深さの最大値) の変化に対するモデルの精度の変化 (3 値分類)
train_test_split(test_size=0.4, random_state=1),forest(random_state=1)

max_depth	精度
None(default)	0.7991
3	0.7857
4	0.7857
5	0.7946
6	0.7901

9　考察

表より、5 値分類よりラベル数を少なくした 3 値分類の方が認識率が高くなっていることがわかる。これは、ラベル間の特徴の違いが曖昧になっていたことが原因として考えられる。例えば 5 値分類の場合、評価基準の 1 と 2 はどちらも怪しいレビュー (サクラレビュー) であることを示しているが、その間の区別がつきにくく、認識率の低下に影響していると考えられる。

今回のデータセットを作成するときにサクラレビューの基準として、単語ではなく文脈や流れなどに着目して判定を行った。spaCy は固有表現に重きを置いている。逆に BERT の特徴は、Transformer という構造が組み込まれており、その構造により文頭と文末の双方向から学習を行うので文脈を読むというような形になっている。ということより、spaCy より BERT の方が精度が高くなったと考えられる。

10　役割

195702D 奥間涼	サクラレビューのある商品の URL 収集、レビューの評価
195748B 徳里 光陽	スクレイピングプログラムの作成。
195761K 塩月 流星	サクラレビューのある商品の URL 収集、レビューの評価
195769E 佐藤 和周	spacy によるベクトル化、ランダムフォレストを用いた機械学習
195770J 栗原 瑠威	BERT によるベクトル化、ランダムフォレストを用いた機械学習

11　振り返り

スクレイピングのコードの完成が早く、レビューはすぐに多数集めることができた。だが、ラベル付けに時間がかかってしまった。

ラベル付け後は、各自が役割を見つけて情報も共有しながらうまく進めることができたと思います。

改善点として、最初は作業の役割分担などをせずに、各自で進めていた部分があるので、最初から役割をはっきり決めていたら作業効率を上げることができたと考えます。

12　時間の都合上省いた項目

いろんな商品のデータセットを用意
他の機械学習の手法との比較

参考文献

[1]]”GiNZA - Japanese NLP Library”
https://megagonlabs.github.io/ginza/

[2]]”スクレイピング禁止の Amazon からレビューを抜き出す”
https://self-development.info/スクレイピング禁止の amazon からレビューを抜き出す/

[3] ”サクラチェッカー”
https://sakura-checker.jp/

[4] ”データ科学便覧”
https://data-science.gr.jp/implementation/iml_sklearn_random_forest.html

[5] ”Amazon 激怒？中国企業アカウントを凍結　サクラレビュー対策に本腰か”
https://news.livedoor.com/article/detail/20667140/

[6] ”初心者でも簡単にできる Selenium のインストール【Python】”
https://self-development.info/初心者でも簡単にできる selenium のインストール【python】/

[7] ”BERT with SentencePiece を日本語 Wikipedia で学習してモデルを公開しました”
https://yoheikikuta.github.io/bert-japanese/