令和 6 年度 卒業研究

ソーシャルメディアデータの感情分析とネガティブな投稿を非表示にできるツールの作成

Analyze social media data for sentiment and develop tools to hide negative posts

函館工業高等専門学校
生産システム工学科 情報コース
10 番 金田 堅志郎
指導教員 東海林 智也

目次

1.　序章

1.1　目的

1.1.1 和文

本研究の目的は、SNS のテキストデータを感情分析することで、テキストを「ポジティブ」「ネガティブ」「中立」の三つの感情に分類し、ネガティブな投稿を非表示にするツールを作成することである。

1.1.2 英文

The purpose of this study is to analyze SNS text data for sentiment, categorizing it into three categories: "positive," "negative," and "neutral," and to develop a tool that hides negative posts.

1.2　研究背景

近年、SNS ではインプレッション数の多さが収益に直結する仕組みが普及している。この影響により、過激な投稿や真偽不明な情報の増加が目立つようになった。その結果、SNS 利用中に不快感を覚えたり、精神的に傷ついたりするユーザーが増加していると考えられる。

そのため、ネガティブな投稿を非表示にするツールを作成することで、SNS 利用中に精神的な負担を軽減し、傷つくユーザーを減らすことを目的とする。本研究テーマは、このような問題を解決するために選定した。

2.　アプリケーション概要

2.1　開発環境

本研究における開発環境は以下のとおりである

OS: Windows 11
開発環境: Anaconda, Cloud Run functions
実装場所: Bluesky
使用言語: Python, JavaScript
使用プラットフォーム: Chrome 拡張機能, Cloud Run functions

2.2　実装場所と使用プラットフォームについて

本研究では、Chrome 拡張機能と Cloud Run functions を使用し、Bluesky を実装場所として選定した。

Bluesky は、X（旧 Twitter）と同様のテキストベースのソーシャルメディアである。ユーザー数は X に比べて少ないものの、API 利用料金の観点から今回は Bluesky を選択した。

Chrome 拡張機能は Google Chrome ブラウザの機能を追加・拡張するためのツールであり、作業の効率化や生産性向上を目的として広く利用されている。拡張機能は JavaScript、HTML、CSS を用いて開発される軽量なプログラムであり、Web ページの内容やブラウザの動作を柔軟にカスタマイズできる。この特徴を活用し、本研究では Web ページ上のネガティブな投稿を検出して非表示にする機能を実現するために Chrome 拡張機能を採用した。

Cloud Run functions は、HTTP リクエストや特定のイベントをトリガーとしてプログラムを起動できるサーバーレスの実行環境である。Python、Java、JavaScript、Go など複数のプログラミング言語をサポートし、大容量ファイルの取り扱いにも適している。この特性を活用し、本研究では Bluesky の API 呼び出し、テキストデータの取得、そして感情極性辞書を用いた感情分析のために Cloud Run functions を使用した。

これらの技術を組み合わせたことで、投稿データの取得から感情分析、ブラウザ上での非表示処理に至る一連のプロセスを効率的かつ効果的に実現できた。

2.3　感情分析の手法

感情分析を行う手法には、大きく「感情極性辞書（図 1）を用いた手法」と「機械学習を用いた手法」の 2 種類がある。本研究では、感情極性辞書を採用した。その理由は主に、Cloud Run functions におけるストレージ容量の制限である。

当初は TensorFlow を利用した機械学習モデルの実装を検討していた。しかし、Cloud Run functions 環境で TensorFlow パッケージをインストールすると、ストレージ容量が制限を超える問題が発生した。この問題により、Chrome 拡張機能側のプログラムと正常な通信ができなくなるなど、実装に支障をきたした。

一方、感情極性辞書は必要なパッケージ数が少なく、ファイル容量も小さいため、環境への負荷を軽減できるという利点がある。また、Cloud Run functions との接続もスムーズに行える。

以上の理由から、本研究では感情極性辞書を用いた手法を採用した。

```
優れる:すぐれる:動詞:1
良い:よい:形容詞:0.999995
喜ぶ:よろこぶ:動詞:0.999979
褒める:ほめる:動詞:0.999979
めでたい:めでたい:形容詞:0.999645
賢い:かしこい:形容詞:0.999486
善い:いい:形容詞:0.999314
適す:てきす:動詞:0.999295
天晴:あっぱれ:名詞:0.999267
祝う:いわう:動詞:0.999122
⋮
物寂しい:ものさびしい:形容詞:-0.999457
辛い:つらい:形容詞:-0.999466
苦しめる:くるしめる:動詞:-0.999484
嘲る:あざける:動詞:-0.999499
背く:そむく:動詞:-0.999531
敵:てき:名詞:-0.999579
責める:せめる:動詞:-0.999615
惨い:むごい:形容詞:-0.999657
荒荒しい:あらあらしい:形容詞:-0.999661
```

図 1 感情極性辞書(一部)

3.　成果物

以下で、今回作成したツールについて詳しく説明する。図 2 は今回作成したツールの構成図である。

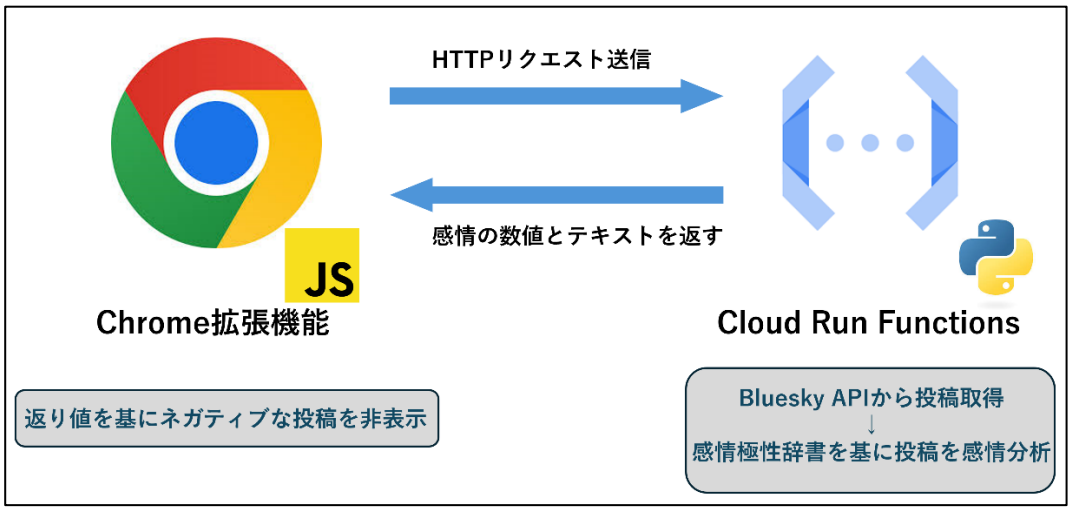

図 2　ツールの構成

3.1　Bluesky との接続

Chrome 拡張機能側のプログラムは、HTTP リクエストを送信することで Cloud Run functions 上の関数をトリガーする。本システムでは、そのトリガーに対応する関数として hello_http 関数（図 3）を実行する。この関数では、最初に create_session 関数（図 4）を呼び出して Bluesky へのログイン処理を行う。具体的には、Bluesky の API エンドポイント（url に格納されたリンク）に対して、ユーザー識別情報（identifier）およびパスワード情報を含むリクエストを送信する。ログインに成功した場合、レスポンスからアクセストークン（accessJwt）を取得し、このトークンを後続の API 呼び出しに利用する。

次に、取得したアクセストークンを引数として fetch_feed_data 関数（図 5）を実行する。この関数は、Bluesky の投稿データを取得するためにエンドポイント（url に格納されたリンク）にリクエストを送信するものである。リクエストパラメータとして、特定の投稿フィードを指定するための feed パラメータ（今回は日本語投稿専用のフィードを指定）および取得件数を制限するための limit=5 を設定する。また、認証情報であるアクセストークンを含めた HTTP ヘッダーを設定し、認証された状態でデータを取得する。

このリクエストの結果として、最新の投稿データがレスポンスとして返される。このレスポンスは JSON 形式であり、必要な投稿情報は feed_data 変数に格納する。その後、for 文を用いて各投稿データを個別に取り出し、感情分析関数へ渡して分析処理を実行する。

```python
# メイン処理
def hello_http(request):
    access_jwt = create_session()
    #省略
    feed_data = fetch_feed_data(access_jwt)
    #省略
    results = []
    for feed in feed_data[:5]:  # 最初の5つの投稿を処理
        post_text = feed["post"]["record"]["text"]
        sentiment_score = sentiment_analyze(post_text)
        results.append({
            "text": post_text,
            "score": sentiment_score
        })
    return jsonify({"posts": results})
```

図 3 hello_http 関数

```python
def create_session():
    url = "https://bsky.social/xrpc/com.atproto.server.createSession"
    data = {
        "identifier": "",  # 正しいidentifierをここに入力
        "password": ""  # 正しいpasswordをここに入力
    }
    headers = {
        "Content-Type": "application/json; charset=UTF-8"
    }

    try:
        response = requests.post(url, headers=headers, json=data)
        #省略
        json_response = response.json()
        return json_response.get("accessJwt")  # アクセストークンを返す
    #省略
```

図 4 create_session 関数

```python
def fetch_feed_data(access_jwt):
    url = "https://bsky.social/xrpc/app.bsky.feed.getFeed"
    params = {
        "feed": "at://did:plc:q6gjnaw2blty4crticxkmujt/app.bsky.feed.generator/cl-japanese",
        "limit": 5
    }
    headers = {
        "Authorization": f"Bearer {access_jwt}"
    }
    try:
        response = requests.get(url, params=params, headers=headers)
        #省略
        response_json = response.json()
        return response_json.get("feed")
    #省略
```

図 5　fetch_feed_data 関数

3.2　感情分析の実行

図 6 に示す sentiment_analyze 関数は、感情極性辞書（図 1）を用いて投稿データの感情分析を行う処理である。この関数の動作は以下の通りである。まず、感情極性辞書ファイルを開き、単語とそれに対応する感情スコアを辞書型データ構造に格納する。この際、ファイル内の各行をコロン（:）で区切り、それぞれの項目から必要な情報を抽出して辞書に追加する。

次に、Janome ライブラリを使用して Bluesky から取得した投稿データを単語単位に分割する。この分割処理によって、各単語が感情極性辞書からスコアを参照できる形式に変換される。分割された各単語について、感情極性辞書にその単語が存在する場合には、その感情スコアを取得し、累積スコアに加算する。同時にスコア計算の対象となった単語数もカウントする。

この一連の処理によって、テキスト全体の感情スコアが累積される。最後に、累積スコアを単語数で除算することで、投稿データの平均的な感情スコアを算出する。この感情スコアは、正の値であればあるほどポジティブ、負の値であればあるほどネガティブを示す指標となる。

求めた感情スコアは hello_http 関数(図 3)へ返される。同関数では、Bluesky から取得した投稿データに対して感情スコアを対応付け、これらを辞書型データとして results に格納する。たとえば、投稿データの本文とそれに対応する感情スコアが一つの辞書オブジェクトとして保存される。

この辞書は最終的に Chrome 拡張機能側のプログラムへレスポンスとして返される。これにより、Chrome 拡張機能側では投稿データとその感情スコアを基に適切な処理を行うことが可能になる。

```
def sentiment_analyze(text):
    sentiment_dic = {}
    with open('/tmp/pn_ja.dic.txt', 'r') as f:
        lines = f.readlines()
        for line in lines:
            line_components = line.split(':')
            sentiment_dic[line_components[0]] = line_components[3]
    t = Tokenizer()
    tokens = t.tokenize(text)
    sentiment_val = 0
    word_cnt = 1e-6  #割り算で使うので0に近い値
    for token in tokens:
        #単語の基本形を取り出し
        word = token.surface
        if word in sentiment_dic:
            sentiment_val = sentiment_val + float(sentiment_dic[word])
            word_cnt += 1

    return round(sentiment_val/word_cnt, 5)
```

図 6　sentiment_analyze 関数

3.3　表示制御の実装

Chrome 拡張機能側のプログラムは、hello_http 関数(図 3)から受け取った感情スコアを基に judgeSentiment 関数（図 8）で投稿の感情を判定する。この関数では、感情スコアが 0.5 以上の場合は「positive」、-0.5 以下の場合は「negative」と判定し、それらの間に収まる場合は「neutral」とする仕組みである。

その後、querySelectorAll メソッドを用いて、data-testid=”postText”属性を持つ要素をすべて取得する。この属性には、Bluesky 上で表示される各投稿のテキストデータが格納されている。取得された要素について、Chrome 拡張機能は投稿データとレスポンス内の投稿データを比較する。この比較において、レスポンスで「negative」と判定された投稿と一致するテキストデータが見つかった場合、その投稿要素の表示を div.style.display = ”none”によって非表示にする。この処理により、ユーザーはネガティブな投稿を視覚的に見えない状態で Bluesky を利用できるようになる。

このような一連の処理によって、Cloud Run functions と Chrome 拡張機能の連携を通じ、Bluesky 上の投稿データに対する感情フィルタリング機能が実現される。

```
if (response.ok) {
  const data = await response.json();

  // データの形式確認
  if (data.posts && Array.isArray(data.posts)) {
    data.posts.forEach((post, index) => {
      const sentiment = judgeSentiment(post.score); //スコアを基に感情を分類

      // Cloud Run から取得した各テキストを利用してページ内をチェック
      document.querySelectorAll('div[data-testid="postText"]').forEach(div =>
        if (div.textContent.trim() === post.text && sentiment === "negative")
          // テキストの感情がnegativeだった場合は非表示
          div.style.display = "none";
        }
      });
```

図 7　Chrome 拡張機能側のプログラム(一部)

```
function judgeSentiment(score) {
  if (score === undefined) {
    return;
  }
  // PN評価値を用いてテキストの感情を取得
  if (score >= 0.5) {
    return "positive";
  } else if (score <= -0.5){
    return "negative";
  } else {return "neutral"}
}
```

図 8 感情を判別する関数

3.4　実行結果

最後に、ツールを使用した際の処理の流れについて説明する。

Bluesky にアクセスしてページのロードが完了すると、プログラムが開始される。今回の実装では、結果をわかりやすく提示するため、プログラム終了後にアラートとして実行結果が表示される（図 9）。このプログラムでは、感情スコアが-0.5 以下の投稿をネガティブと判定する。

アラートには、図 9 および図 10 に示すように、投稿テキストとその感情スコアが表示される。図 10 の例では、5 つの投稿のうち、スコアが-0.5 以下の 1～4 番目の投稿がネガティブと判定され、5 番目の投稿はニュートラルと判定される。

アラートの「OK」ボタンを押すと非表示プログラムが実行される。図 11 を見ると、この処理によって、ニュートラルと判定された 5 番目の投稿のみが表示され続ける一方、ネガティブと判定された最初の 4 つの投稿は非表示となることが確認できる。

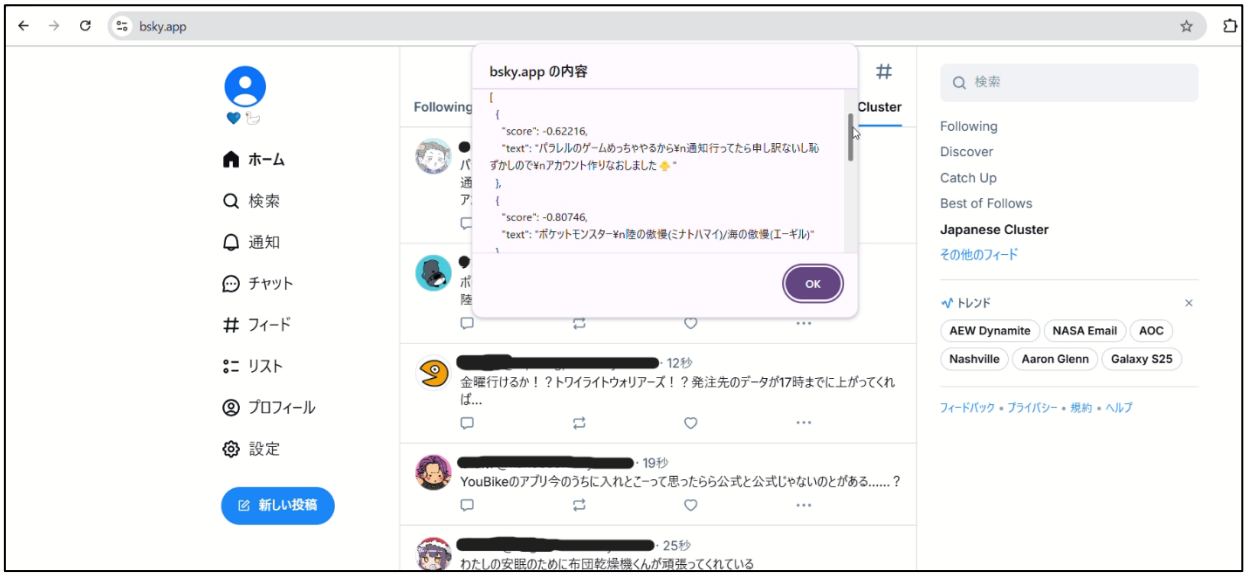

図 9 実行画面

```
  "score": -0.62216,
  "text": "パラレルのゲームめっちゃやるから¥n通知行ってたら申し訳ないし恥ずかしので¥nアカウント作りなおしました🙇"
 },
 {
  "score": -0.80746,
  "text": "ポケットモンスター¥n陸の傲慢(ミナトハマイ)/海の傲慢(エーギル)"

  "score": -0.55161,
  "text": "金曜行けるか！？トワイライトウォリアーズ！？発注先のデータが17時までに上がってくれば..."
 },
 {
  "score": -0.57081,
  "text": "YouBikeのアプリ今のうちに入れとこーって思ったらら公式と公式じゃないのとがある.....？"

  "score": -0.26945,
  "text": "わたしの安眠のために布団乾燥機くんが頑張ってくれている"
```


図 10 アラートの中身

図 11 実行結果

4.　現状の課題

本研究では、SNS のネガティブな投稿を非表示にするツールを開発したが、現時点でいくつかの課題が存在する。

第一に、Bluesky へのアクセス後、ツールが反映されるまでに時間がかかる点が挙げられる。

第二に、ネガティブな投稿を非表示にするために毎回リロード操作が必要であり、利便性に欠ける点も問題である。

第三に、一度のプログラム実行で取得可能なテキストデータが最大 5 件と限られており、ツールの適用範囲が狭まることも問題である。

最後に、本プログラムの感情分析では形態素解析を使用していないため、分析精度が低いという課題も存在する。当初は MeCab ライブラリを使用して文章を形態素解析する予定であったが、Cloud Run functions 上で MeCab を使用するとエラーが発生し、その原因を解明できなかったため、代替として Janome ライブラリを用いた。その結果、文脈や文法の正確な解析が難しくなり、分析精度が低下した。

これらの課題を克服することで、ツールの実用性と分析精度を向上させ、より多くのユーザーに満足して利用されるシステムを構築できると考える。

謝辞

指導教員の函館工業高等専門学校生産システム工学科 東海林教授には終始適切なご指導を賜りました。ここに深謝の意を表します。

参考文献

[1] 高村大也, 乾孝司, 奥村学
"スピンモデルによる単語の感情極性抽出", *情報処理学会論文誌ジャーナル*, Vol.47 No.02 pp. 627--637, 2006.
http://www.lr.pi.titech.ac.jp/~takamura/pndic_ja.html

[2] Bluesky の特定のフィードの投稿のみを JSON ファイルに抽出するプログラムを作った
https://qiita.com/taxi13245/items/4fe52c356af4a5a6d145

[3] Python で感情極性辞書を用いたテキスト感情分析をつくってみた
https://qiita.com/miso_taku/items/c8e48401346eca23bae8

[4] 自然言語処理の力で世の中から暗いサイトを駆逐する Chrome 拡張を作った
https://qiita.com/tomoino/items/859c34823367aa9cbd13