テキストマイニングを用いた認知機能研究の分析効率化
Leveraging NLP techniques for cognitive studies

岸山 健†，田 子健†，広瀬 友紀†，幕内 充‡
Takeshi Kishiyama, Zijian Tian, Yuki Hirose, Michiru Makuuchi
† 東京大学大学院総合文化研究科，‡ 国立障害者リハビリテーションセンター研究所
The University of Tokyo, National Rehabilitation Center for Persons with Disabilities
kishiyama.t@gmail.com, zijian-tian@ardbeg.c.u-tokyo.ac.jp, hirose@boz.c.u-tokyo.ac.jp,
makuuchi-michiru@rehab.go.jp

概要

認知機能の発達を検証する「描画課題」のデータを例に，コーパスなどのテキストから有用な情報を抽出する「テキストマイニング」を適用する手法を本稿では報告する．行動データを Python ライブラリである scikit-learn に対応した形式に変換すると，コードを簡潔に保ちつつ効率的に分析できる．サンプルコードを公開し，前処理から結果解釈の方法まで述べる．

キーワード： NLP, BoW, ロジスティック回帰

1. はじめに

本稿では，自然言語処理 (natural language processing: NLP) で用いられる手法を認知機能研究に対して適用するデータ分析手法を報告する．認知機能研究において被験者の属性 (母語や年齢など) と行動データの関連を質的に分析する場合には，作業効率や属人性が課題となる．以下，実験実施者が指定したイラストを被検者に呈示し線画で模写させ，イラストを構成するパーツの階層性や再帰性を被験者がどう認知するかを検証する「描画課題」 [3, 4, 9] を例に，分析時の課題を挙げる．

田他 (2021) では日本語話者成人 36 名 (19〜26 歳) とタイ語話者成人 35 名 (18〜20 歳)，タイ語話者幼児 40 名 (4〜5 歳) を対象として，描画課題でイラストの模写を要求した．イラストは成人に対し 30 種類，幼児に対し 15 種類であり，描画時の被験者のペン先の座標や筆圧，時刻はアノトペンと Seldage 社開発の専用紙で記録した．実験で得た座標と時刻のデータを用いて描画されたパーツにアノテーターがラベルを与えた結果，模写の対象が人の顔であれば「髪 輪郭 眉 目 …」のような系列データとなる．しかしながら，得られたアノテーションの数に対して分析対象は 2 件と限定的であった．仮に分析すべき特徴的な描画パターンを機械的に検出し分析を効率化できれば分析対象を増やせる．またその際，可能な限り既存のライブラリーを利用し，分析者が記述するコードを短くできれば作業の属人化を避けられる．

そこで本稿では特徴的なパターンを抽出するため，テキストマイニングを用いた統計分析を採用した．テキストマイニングは目的に応じて情報をテキストから抽出する手法であり，対象テキストは自然言語の文書や文章だけでなく，文字列ならば遺伝情報やアクセスログなども含まれる [7]．したがって，認知機能研究においても行動データをテキストに変換すれば同手法が適用可能になる．また，変換の際には `scikit-learn` の枠組みを使いコードを簡潔にした．そこで本稿では描画課題を例にしつつ，前処理と統計分析，可視化の 3 点の手順をサンプルコードとともに述べる．

なお，環境構築には国立情報学研究所 (NII) による「オンライン分析システム」を利用する．このシステムは日本学術振興会 (JSPS) 人文学・社会科学データインフラストラクチャー構築推進事業の一環として作成され，Jupyter Notebook と RStudio の実行環境を NII がクラウド上で提供するものである．環境構築の不一致による問題を避けるため，本稿の環境やデータは第一著者の GitHub に配置し (https://github.com/kishiyamat/jcss-2022-repl)，具体的な手順も同レポジトリで記述する．

2. 前処理

本稿では，処理の多くを産業や研究で広く使われる機械学習ライブラリである `scikit-learn` に準拠させる．そこで，前処理を (i) `scikit-learn` を利用できる形へのアノテーションデータ整形と (ii) `scikit-learn` によるテキスト特徴量の抽出の 2 段階に分ける．前処理から `scikit-learn` の枠組みを利用することで，自身で書かなければならないコードの量を削減できる．またその後の分析でも `scikit-learn` の枠組みを使うため，一貫した記法を利用できるため可読性が向上する．

2.1 アノテーションデータ整形

後の統計分析の準備として，行動データをスペース区切りで記述しなおす前処理が必要になる．アノテーションデータは前処理なく分析可能である場合は少なく，大抵は前処理が必要になる．これはアノテーションのしやすさと計算機での扱いやすさが一致しないことが原因となる．表 1 には例として，描画課題におけるアノテーションデータを 3 行のみ示す．表の一列目 `stroke_id` は描き順の番号を整数値で示し，その他の列は描かれたパーツが被験者ごとに示されている．表 1 に示す形式は，アノテーション時に複数のファイルを開く必要がなく編集が効率的である．しかしながら，この形式では各列が異なる意味をもっているため，一般に列名を変数（要因）とする R や Python での分析が難しい．

表 1 アノテーションされたデータの例

stroke_id	1_Adult	2_Adult	3_Child
1	round3	round3	round1_1
2	round3	round2_1	round1_2
3	round2_2	round2_1	round1_3

上記のデータ形式に対し，計算機が扱いやすいデータ形式は「整然データ」と呼ばれ，以下の 3 つの条件を満たす [5].

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

表 1 に示したデータは要因が列を形成していないので条件 1 を満たさない．また，一つの行に複数の被験者の反応が混ざっているため，条件 2 も満たさない．これに対して，条件を満たす整然データを表 2 に示す．各列は被験者の観測ごとの要因を記録しており，各行はトライアルや一つの単位の観測を持つ．

表 2 整然データの例

subject_id	age	strokes
1	Adult	round3 round3 round2_2…
2	Adult	round3 round2_1 round2_1…
3	Child	round1_1 round1_2 round1_3…

なお表 2 では，行動データは `strokes` という列にスペース区切りで格納されている．整然データの枠組みにおいて一つのセルにスペースやカンマで区切ったデータを格納することは推奨されていないが，この形式にすることで Python の `scikit-learn` の枠組みを利用して分析しやすい形に変形できるようになる．

2.2 テキスト特徴量の抽出

次に整然データに整形した行動データをテキストとして扱い，被験者の属性に固有のパターンを抽出する準備をする．最終的に特定の属性の確率を上げるパターンを見つけることを目的とするため，パターンの有無を表現する数値に変換したい．この場合，各観測を文書 (document)，全体のデータをコーパス (corpus) として自然言語処理の手法が使える．

単純で効率の良い方法の一つに BoW(bag-of-words) 表現への変換が挙げられる．描画データの例で言えば，表 2 の `strokes`を `round3`や `round2_1`というパーツ (個々の意味は後述する) が描画された回数で表現し直す変換が該当する．さらに「`round3` から `round2_2` への遷移」が出現した回数を表現する場合，2 つの組み合わせをみるので “bi-gram” と呼ばれる．このような n 個の連続に着目する分析を n-gram と呼ぶ．

上記の n-gram や無視したい語 (stop-words) を指定する実装はすでに `scikit-learn` にあるため，これらも `scikit-learn` の枠組みを使うと容易に調整できる．まずは n-gram の幅や stop-words を指定して，`CountVectorizer`を `vectorizer`として初期化する (下記コード 1)．次に strokes 列にあった行動データを `corpus` に格納し (2)，コーパス全体で発生するパターンを `fit`で記録する (3)．手順 (3) により，`vectorizer`はテキストを BoW に変換可能になる．なお，ここでは表 2 のデータが `df`という変数に格納されていることを前提としている．コードは analysis.ipynb として先述したレポジトリにも配置した．

```
from sklearn.feature_extraction.text \
    import CountVectorizer
n_gram = 2

# (1)
vectorizer = CountVectorizer(
    stop_words=["nan", "extra", "point"],
    ngram_range=(1, n_gram),
)
# (2)
corpus = df.strokes
# (3)
vectorizer.fit(corpus)
```

上記のコードにより，テキストに対して bi-gram を算出する準備が整った．あとは `transform`と `toarray` の実行により行列に変換し，`X`に格納する．このベク

トルが bi-gram のカウントデータであり，列方向には get_feature_names_outで参照できるパターン名があり，行方向には各トライアルおけるパターンの出現数が与えられる．その後の可視化のため，bi-gram は - でつないだ文字列とする．例として，一行目には最初の被験者の描画で round1_1が現れた回数や round1_1-round1_1 が現れた回数などが整数値で表される．

```
X = vectorizer.transform(corpus).toarray()
cols = vectorizer.get_feature_names_out()
cols = list(map(
    lambda s: s.replace(" ", "-"), cols))
df_bow = pd.DataFrame(X, columns=cols)
```

この時点で，もともと 16 だったパターン数は遷移を含め 47 種類となった．したがって，X は被験者数を n_sample，パターン数を n_feature としたとき，(n_sample, n_feature) の行列となる．次に，この 47 列からなる BoW ベクトルを独立変数とし，年齢を従属変数とするロジスティック回帰分析を作成する．そこで変数となる y を作成する．ここではタイ語母語話者のうち，成人だった場合に 0，幼児だった場合に 1 とする．

```
y = df.age == "Child"
y = y.astype(int)
```

以上で被験者の属性で特徴的なパターンを取得する前処理は完了する．

3. 可視化と解釈

次に，前処理の段階で 47 となったパターンの中から，被験者の属性を予測するパターンを抽出する．前処理において，被験者が幼児である場合は 1，成人である場合は 0 としてコーディングした．ここでロジスティック回帰モデルを作成すると，従属変数は幼児か否かの 1/0，独立変数は 47 パターンそれぞれの出現回数となる．各パターンの出現回数に対し，それぞれで傾き β が推定される．推定された β は，β と対応するパターンが現れる回数が 1 増えると「従属変数が 1 となる (つまり該当するトライアルが幼児の行動データである) 確率」に対する「従属変数が 0 となる確率」の割合，つまりオッズ比が $\exp(\beta)$ 倍になると解釈できる．

このロジスティック回帰を単に適用すると 47 の β が得られるが，目的は情報抽出であるため有効なパターンに数を制限したい．このような場合には「正則化」が有効であり，ここでは複数の β を 0 とする L1 正則化を行う．これにより，β が 0 でないパターンが被験者の属性に役に立つと解釈できる．以下では正則化のパラメータ C にデフォルトの 1 を与えて L1 正則化したロジスティック回帰を作成した．その結果，47 あったパターンの中で 0 以外の値を与えられたのは 12 個であった．正則化のパラメータ C を小さくすると，より多くの β が 0 となる．

```
from sklearn.linear_model \
    import LogisticRegression

C = 1
lr_l1 = LogisticRegression(
    C=C,
    penalty="l1",
    solver="liblinear")
lr_l1.fit(X, y)  # estimate beta
coef = lr_l1.coef_.flatten()
non_zero = coef != 0
sum(non_zero)  # 12
```

ここで係数が 0 でないパターンは以下のように抽出，描画できる．係数とパターンの列から 0 となった index を省いたデータフレームを作成して res に格納し，R の ggplot2 を移植したパッケージ plotnine を利用して描画，保存する．

```
from plotnine import *

res = pd.DataFrame(
    {"coef": coef[non_zero],
     "feature":np.array(cols)[non_zero]})
p = (ggplot(res, aes(x="coef", y="feature"))
     + geom_point()
     )
p.save(f"coef_{C}.pdf",
       height=7, width=7,
       units="cm")
```

その結果，図 1 の散布図が描画できる．図の x 軸には係数，y 軸には後述するパターンを配置しているため，分析者はどのパターンがどの属性の予測に役立つかを特定できる．例として，round1_3 から round2_2 への遷移は，幼児データであるオッズ比 (幼児データである確率/成人データである確率) が $\exp(0.6)$，つまり約 1.63 となる．反対に，round1_3から round1_4への遷移は，幼児データでオッズ比が約 0.37 となる．

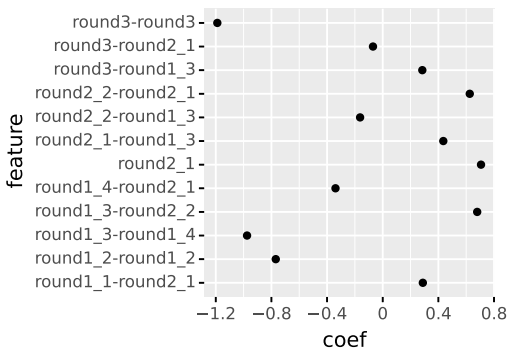

図 1 ロジスティック回帰の係数とパターン

これらの結果の解釈を図を用いて説明する．上で述べた round1_3や round2_2などは図 2 に示す「トーナメント」と呼ばれる階層的な構造のパーツ名である．それぞれを．先程の解釈より，round1_3-round2_2 が現れると幼児のデータである確率が約 1.63 倍となるのに対し，round1_3-round1_4 は約 0.37 倍に下げた．遷移を比較する際，このように起点を揃えると，「round1_3から round2_2という下から上へ描き進める手順は round1_4へと描き進める手順より幼児に特徴的だ」という解釈できる．

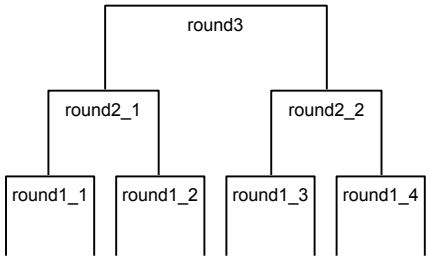

図 2 描画課題に用いた「トーナメント」とアノテーションされたパーツ名

この可視化により，どの行動データがどの属性に特徴的なのかを定量的な値によって判断できるようになる．その際，同じコードを使いまわせるのに加え，注目すべき値が β に集約されるため分析者にとって効率的となる．また，整然データと scikit-learn の枠組みにより，コードは簡潔に保てるため属人化が避けられる．

4. まとめ

最後に派生する分析方法を 3 点述べる．まず，以上の分析は bi-gram に制限したが，前処理や分析の段階で変更ができる．例として，tri-gram やより長い系列をパターンとみなしたい場合，CountVectorizer を初期化する際のパラメータ ngram_range で調整できる．また，同じパターンの重複をなくしたい場合，前処理の際にテキストから重複を除外して対応できる．

次に，時間軸に沿った分析方法を述べる．今回の分析では BoW に基づき時間情報を全て捨てている．しかしながら，特定のパターンが行動の最初にでてくるか，最後にでてくるかといった時間情報に興味がある場合もある．そのような場合は，テキストに系列全体に対する出現位置を分析の目的に沿った形で整数値に丸め込み，単語に加える対応が必要となる．

最後に，今回は BoW を用いたため言語における「語」相当の分析方法となったが，より階層的な句構造の分析には文脈自由文法を用いた方法が考えられる．作業は増えるものの，まずはアノテーションデータに対して句構造をさらにアノテーションし，そのアノテーションに対して今回と同じ手法を用いれば特徴的な「句」や「節」の分析が可能となる．その際は Penn Treebank と同様の形式でアノテーションし，nltk パッケージの Treeクラスを使うと可視化や句構造の推測もできる．

なお，個々のプロジェクトに依存する前処理に関しては専門の書籍 [8] を参考にすると良い．また今回用いたロジスティック回帰の解釈に関しては，統計の入門的な書籍 [6] が詳しい．なお，精度を求める場合はモデルの種類をロジスティック回帰から他のモデル，近年では LightGBM [2] などへ変更すると，予測精度を上げつつ解釈がしやすいモデルが作成できる．その際，ngram_rangeはハイパーパラメータとなるので，optuna [1] などを用いて調整するとより精度が向上するが，過学習に注意が必要となる．

文献

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[2] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, Vol. 30, , 2017.

[3] Michiru Makuuchi. Hierarchical composition in complex object drawing. 2019.

[4] Michiru Makuuchi, Tatsuro Kaminaga, and Morihiro Sugishita. Both parietal lobes are involved in drawing: a functional mri study and implications for constructional apraxia. *Cognitive brain research*, Vol. 16, No. 3, pp. 338–347, 2003.

[5] Hadley Wickham. Tidy data. *Journal of Statistical Software*, Vol. 59, No. 10, p. 123, 2014.

[6] 久保拓弥. データ解析のための統計モデリング入門. 岩波書店, 2012.

[7] 石田基広, 金明哲. コーパスとテキストマイニング. コーパスとテキストマイニング. 共立出版, pp. 1–14, 2012.

[8] 中村大城. 前処理大全 [データ分析のための sql/r/python 実践テクニック]. システム/制御/情報, Vol. 63, No. 7, pp. 296–296, 2019.

[9] 田子健, 岸山健, 陳姿因, 広瀬友紀, 幕内充. 幼児のアイテム描画課題におけるチャンキング傾向の考察. 電子情報通信学会技術研究報告; 信学技報, 2021.