# 深層学習に基づく Publisher2vec による出版者のグルーピング
## ～TRC MARC を利用して～

岸田和明† 門脇夏紀‡

†慶應義塾大学文学部 kz_kishida@keio.jp

‡慶應義塾大学大学院文学研究科 kadowaki.72@keio.jp

抄録

深層学習においては，テキスト中の単語は Word2vec のようなアルゴリズムで事前に数値ベクトル化されてニューラルネットワークに組み込まれることが多い。本研究では，この線に沿って，MARC レコード中に出現する出版者（出版社）の数値ベクトル化を試みる（Publisher2vec）。具体的には，LSTM を使って，各図書の書名と内容説明からその出版者を予測する分類器を構築し，その中から各出版者に対応するベクトルを取り出した。それに基づいて，計量的な多次元尺度構成法を試みたところ，経験的に妥当なグルーピング結果が得られた。

## 1. はじめに

最近，ニューラルネットワークでの深層学習の技術が急速に発達し，さまざまな問題に応用されている。深層学習では，その内部で分析対象が数値ベクトル化され，柔軟性の高い回帰・分類が実現される。そのため，テキスト処理への応用では，Word2vec や GloVe などのアルゴリズムにより，テキスト中の単語の数値ベクトルを別コーパスで事前学習することが標準となっている。

本研究では，この線に沿って，MARC レコード中に出現する出版者（出版社）の数値ベクトル化を試みる。具体的には，LSTM（Long Short Term Memory）を使って，各図書の書名と内容説明からその出版者を予測する分類器を構築し，その中から各出版者に対応するベクトルを抽出する。

想定しているのは，図書間の類似度に基づく情報推薦への応用である。例えば，書名や目次等を数値ベクトル化してこの類似度を求める際に，出版者のベクトルを加えることで，より有用な推薦を実現できるかもしれない。出版者によって出版物のスタイルやジャンルが異なるので，その情報を追加することは推薦にとって有用であると予想される。

ただし本稿では，推薦への応用の前段階として，抽出したベクトルを使って多次元尺度構成法（MDS）を実行し，出版者を 2 次元空間上に布置した結果までを報告する。すなわち，ベクトル化の妥当性を確認することまでが本稿の目的である。

## 2. 関連研究

発表者らが調べた限りでは，MARC レコードに基づいて出版者の数値ベクトルを推計した試みは見つけられなかった。出版者に関する計量書誌学的な分析として，その評判・名声を反映した出版者の順位を求める研究がいくつか行われている[1)]。また，出版物の売上予測モデルにおいて，売上の時間変化についてのパターンのクラスタリングを活用する試みもある[2)]。これらはいずれも出版者に直接的あるいは間接的に関連した計量分析であるものの，本稿の先行研究というわけではない。

一方，出版者以外の何らかの実体を数値ベクトルに変換する試みは，上記の Word2vec をはじめとして枚挙にいとまがない。これはもちろん，深層学習をそれぞれの問題に応用する場合に，必要となるためである。

Word2vec の発表は 2013 年であり，その直後，Doc2vec が提案された。後者は「単語からベクトル」ではなく，「語の集合（文書）からベクトル」への対応付けである。一方，Seq2vec は「語の並び（sequence）」の変換であり，深層学習に基づくテキスト分類の基本となっている。そして，自然言語処理の主要な応用である機械翻訳は，Seq2seq の実現として捉えることができる。例えば日英翻訳は日本語文（seq）から英文（seq）への変換である。

このような流れからさまざまな「○○2vec」が考案されており，検索エンジンで「2vec」と投入すると，数多くのこの種のアルゴリズムのページがヒットする。これを模倣すれば，本稿での数値ベクトルの算出はいわば「Publisher2vec」と呼べるかもしれない。ただし，技術的には，後述するように，

Seq2vec の一種に過ぎない。

## 3. 出版者の数値ベクトル化のための方法

### 3.1 Word2vec での単語のベクトル化

Word2vec の実体は，1 つの隠れ層をもつニューラルネットワークであり，ある単語の周囲に生起する文脈語を入力層として，その単語を出力層で予測する場合，特に CBOW モデルと呼ばれる。例えば「Yesterday, I visited Tokyo by train with my friend.」で，「Tokyo」をその前後 2 語ずつ（I, visit, by, train）で予測すると仮定する。文脈語の数を$m$と表記すれば$m = 4$であり，また，文脈語の$i$番目の位置（$i = 1,2,3,4$）に$j$番目の語が出現していることを$x_{ij} = 1$で表す（そうでなければ 0）。ここで「$j$番目」とは，対象コーパス中での単語のいわば「通し番号」に相当し，単語の異なり数を$d$とすれば，1 から$d$までのいずれかの番号が$j$の実際の値となる。

隠れ層のベクトルにおける$q$番目の要素を$h_q$と書き，この要素と$j$番目の語の関係の程度（重み）を$u_{jq}$と表記すれば，

$$h_q = \sum_{i=1}^{m}\left[\sum_{j=1}^{d} u_{jq}x_{ij}\right]$$

となる [3]。本稿では，隠れ層の次元は$p$で表すことにする（$q = 1,2,\dots,p$）。

出力層についても同様に，例えば「Tokyo」を$j$番目の語として$y_j = 1$（それ以外の語では 0）とする。ここで，$h_q$と$y_j$との関係の程度（重み）を$v_{qj}$と書けば（$q = 1,\dots,p; j = 1,\dots,d$），softmax 関数を利用して，

$$\hat{y}_j = \frac{\exp\left(\sum_{q=1}^{p} h_q v_{qj}\right)}{\sum_{k=1}^{d}\exp\left(\sum_{q=1}^{p} h_q v_{qk}\right)}$$

のように予測できるので，実際の$y_j$との誤差に基づいた損失関数により，$u_{jq}$と$v_{qj}$の値をデータから学習することが可能になる [3]。

その上で，特定の$j$番目の語について，

$$\vec{v}_j = [v_{1j}, v_{2j}, \dots, v_{qj}, \dots, v_{pj}] \qquad (1)$$

として$p$次元ベクトルを構成すれば，これがすなわち，$j$番目の語に対する数値ベクトルとなる。

### 3.2 出版者を予測するための分類器の構成

Word2vec の枠組みに従い，本稿でも，隠れ層ベクトルの要素の値$h_q$に基づいて最終的な softmax 関数を計算するための重みである$v_{qj}$を取り出し，(1)式を出版者の数値ベクトルとして採用する。この場合，$v_{qj}$は「$j$番目の出版者」についての$q$番目の要素の値ということになる（$q = 1,\dots,p$）。

実際には，MARC レコードから各図書の書名と内容説明のテキストを抽出した上で，それに基づいて出版者を予測する分類器を学習し，その結果から(1)式のベクトルを取り出すことを試みる。そのため，前提は「刊行した図書の書名や内容説明のテキストが似ているほど，出版者もまた類似している」となる。MARC レコードを使えば，著者や NDC 番号，件名などの手がかりも利用可能であるものの，今回は，これらは使用しない。さらには，組織としての沿革やその他の企業情報など，MARC レコードの「外側」の何らかのデータも有用かもしれないが，本稿の範囲外である。

### 3.3 LSTM の学習と数値ベクトルの取り出し

テキストデータの場合，「I visited Tokyo.」のように語が時系列的に連続する。この種の入力に適したアルゴリズムとして RNN（Recurrent Neural Network）がある。この場合，「I」「visit」「Tokyo」についてのそれぞれの数値ベクトルが個別の入力となり，その都度，隠れ層の状態が変化する。これを模式的に書けば，図 1 のようになる。

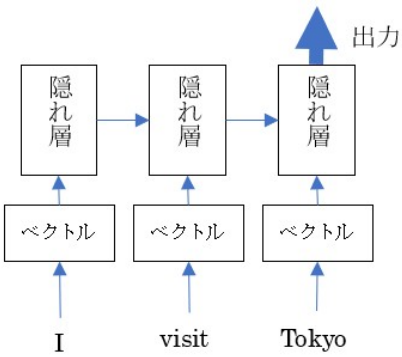


**図 1　RNN の概念図**

より最近提案された Transformers もまたテキストデータに適したモデルであり，RNN とは異なり，attention の仕組みを有効活用している。現在ではこのためのソフトウェアが充実しているものの [4]，本稿では，RNN の仕組みに基づき，図 1 の最後の「出力」の部分から出版者のベクトルを取り出すことをまずは試みる。ただし，RNN ではなく，それに対して改良が加えられた LSTM を利用する。

本稿の実験では LSTM の実行に PyTorch における torch.nn の LSTM モジュールを利用した（nn.LSTM）。(1)式の$v_{qj}$は，その都度 nn.Linear で計算し（これは標準的な手続きである），その結果を最終的に取り出した。

## 4. 出版者ベクトルの推計に関する実験

### 4.1 使用したデータ

今回の実験では，2019 年 12 月から 2020 年 12 月までに作成された TRC MARC を利用した。レコードの総数は 53,658 件で，そこに含まれる出版者数は合計 3,241 であった。最多出版点数は KADOKAWA の 2,352 点で，それに講談社の 1,273 点が続いている。

出版者を出版点数の降順で並べ，縦軸を出版点数の対数，横軸を順位の対数とした分布を図 2 に示す。これはすなわち Zipf の法則に相当し，出版者とその点数の場合でも，この経験則が成立していることが図から読み取れる。

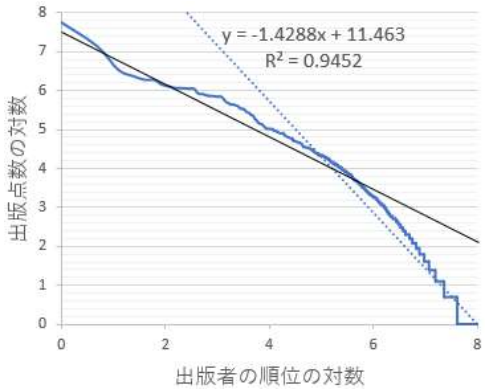


**図 2　MARC データにおける出版点数の分布**

この実験では，150 点以上の図書を刊行している出版者に限定して，その数値ベクトルを求めることとした。その結果，対象となる出版者の総数は 57，その合計出版点数は 19,778 となった（全体の約 37%）。この 19,778 レコードが分類器の学習のための標本サイズに相当する。また，分類の目的変数は「出版者」なので，今回の実験では 57 カテゴリのマルチクラス分類を実行することになる。

表 1 は NDC の最上位 10 区分別での出版点数である。「その他」には絵本などが含まれる。

### 4.2 語の埋め込み

TRC MARC の書名フィールドと内容説明フィールドから，図書ごとにテキストデータを取り出した。そこから janome で名詞と形容詞のみを抽出し，そのままの出現順序どおりに（出現回数のカウントはせずに），語の並びとして LSTM へ投入した。なお，その際に不要語をいくつか除去している[5]。この入力から図 1 に書かれているような出力を得た後で，(1)式のベクトルを使って 57 カテゴリへの分類を試みることから，この仕組みは基本的には Seq2vec である。

**表 1　NDC10 区分別での出版点数**

| NDC | 今回の分析対象 | | 全体 | |
|---|---|---|---|---|
| 0 類 | 650 | 3.3% | 1882 | 3.5% |
| 1 類 | 918 | 4.6% | 3299 | 6.1% |
| 2 類 | 1360 | 6.9% | 3527 | 6.6% |
| 3 類 | 3086 | 15.6% | 10816 | 20.2% |
| 4 類 | 1355 | 6.9% | 5987 | 11.2% |
| 5 類 | 1686 | 8.5% | 5260 | 9.8% |
| 6 類 | 566 | 2.9% | 2314 | 4.3% |
| 7 類 | 2313 | 11.7% | 6734 | 12.5% |
| 8 類 | 227 | 1.1% | 938 | 1.7% |
| 9 類 | 7215 | 36.5% | 11347 | 21.1% |
| その他 | 402 | 2.0% | 1554 | 2.9% |
| 合計 | 19778 | 100.0% | 53658 | 100.0% |

ただし，図 1 に示すように，LSTM 内部では，各語は数値ベクトルとして埋め込まれる。このベクトルを MARC レコードのテキストで LSTM 自体に学習させることはもちろん可能であるが，そうすると，各語の本来的な意味に関係なく，単に出版者の分類を成功させるという基準の下に過学習がなされてしまう。そこで，この実験では，東北大学にて Wikipedia データで算出された Word2vec のベクトル[6]を埋め込み，それを LSTM 内部で更新しないように固定した。

### 4.3 LSTM 分類器の学習

LSTM の実験では通常，データを「訓練用」「検証用」「評価用」に分割する。この実験では，未知の新規データに対する分類性能を確認する必要はないので，評価用データは不要である。一方，過学習の程度を観察して最適なエポック数を決めるために検証用データは必須となる。

本実験では，19,778 レコードを 18,778 件と 1,000 件に 2 分割して前者を訓練用データ，後者を検証用データとした。そして，これらを使った学習によってエポック数を決定した後，今度は 19,778 レコード全体で学習を再度試み，そのエポック数で停止した時点で，(1)式の数値ベクトルを分類器から取り出した。これが最終的な出版者ベクトルということになる。

なお，LSTM への投入の直前に，1 つの図書にしか出現しない語は落とした。LSTM の隠れ層の次元は 200 に設定し，Word2vec による語の埋め込みの次元（ベクトルの次元）も同様に 200 とした。そのほか，ミニバッチ数は 32，学習率は 0.01 とし

て，最終段階の分類には対数 softmax 関数を利用して，実験を進めた（すべて Python で実装）。

### 4.4 実験結果

名詞と形容詞の抽出・不要語の除去・出現図書数 1 の語の除去の結果，19,778 レコード全体での異なり語数は 40,358，平均文書長は約 23.2 語であった。検証用データでの損失関数の変化を視認し，エポック数を 5 とした。最終的に，このエポック数で 19,778 レコード全体を使って分類器を学習したところ，正解率は約 32%であった。ここから各出版者の数値ベクトルを取り出した（(1)式参照）。

取り出したベクトルを使って計量的 MDS を実行した結果を図 3 に示す（Python の sklearn を利用）。なお，実行前に各ベクトルを単位ベクトルに変換した。類似していると経験的に思われる出版者どうしがそれなりに近い場所に布置されている。このことから，出版者の数値ベクトルの事前学習結果としては十分であると判断される。

## 5. おわりに

本稿では，TRC MARC レコードに対して LSTM を実行することにより，出版者に対する数値ベクトルを生成した。その結果から描画された図 3 はそれなりに出版者の特徴を反映していると思われるものの，図 3 を描くだけならそのほかの方法もありうる（各出版者の図書の書名・内容説明から文書ベクトルを構成して余弦係数を計算するなど）。本研究の焦点はあくまで，Publisher2vec であり，出版者を固定長の数値ベクトルで表す点に意味がある。情報推薦などの問題に対して，このベクトルをいかに活用していくかが今後の課題である。

**謝辞**　株式会社図書館流通センターから，研究用として TRC MARC データを提供していただきました。感謝申し上げます。

### 引用文献

1) Zuccala, A. et al. “Can we rank scholarly book publishers? A bibliometric experiment with the field of history,” *Journal of the Association for Information Science and Technology*, vol.66, 2015, p.1333-1347.
2) Tanaka K., et al. “The cluster grouping approach of NM Forecasting Model for book publishing,” Frey D. et al. eds. *Improving Complex Systems Today: Proceedings of the 18th ISPE International Conference on Concurrent Engineering*, Springer, 2011, p.233-240.
3) Aggarwal, C. C. “2.6 Word2vec: An application of simple neural architectures,” *Neural Networks and Deep Learning: A Textbook*. Springer, 2018.
4) URL は，https://huggingface.co/transformers/
5) http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt を利用した。
6) URL は，http://www.cl.ecei.tohoku.ac.jp/~m-suzuki /jawiki_vector/

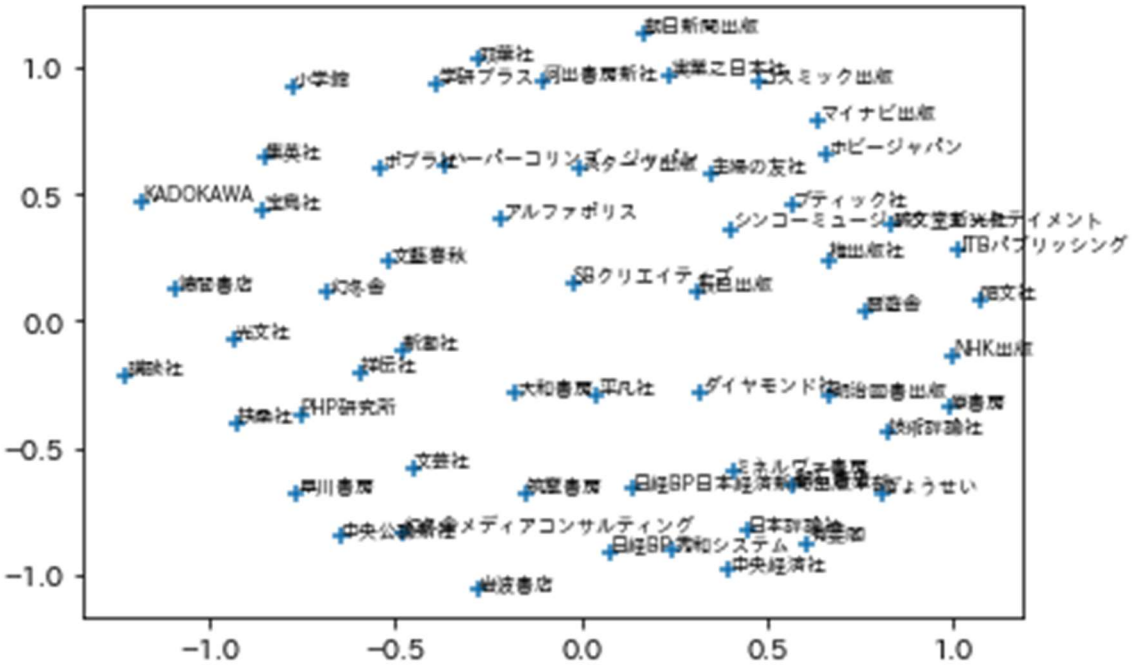


図 3　出版者ベクトルによる計量的 MDS の結果