Researcher2Vec: ニューラル線形モデルによる自然言語処理研究者の可視化と推薦

持橋 大地
統計数理研究所 数理・推論研究系 ／
日本学術振興会 学術情報分析センター
daichi@ism.ac.jp

1 はじめに

科学技術の高度化に伴い, 論文数や研究者の数は増える一方であり, その全貌を直感的に把握することはますます難しくなっている. この傾向は自然言語処理を含む計算機科学分野では特に顕著であり, 言語処理学会 2020 年大会の論文数は 338 本, ACL 2020 の採択論文数は 779 本, NeurIPS 2020 では 1,900 本を超えている. これらをタイトルだけであっても, すべて読んで関連研究を見つけたり, 興味の近い教員や研究者を探すことは, 今やほぼ不可能であると言ってよい.

この問題は論文の査読に際しても重要であり, 近年では国際会議を含む論文の査読割り当ては LDA [1] を内部的に用いた Toronto Paper Matching System (TPMS)[2] による適切な候補者の自動スコア付けなしでは成り立たなくなっている. しかし, TPMS はあくまで個々の論文に査読者をマッチさせるものであり, 各研究者が全体としてどのような専門性を持っているかはわからないため, 最終的な割り当てには個人の発表文献などの情報を人手で確認する必要がある. 特に, 査読者やエリアチェアのリクルート, ワークショップの発表者などで「人を探す」必要がある場合は TPMS は使えず, 人的ネットワークに頼らざるを得ないという現状がある. このため, たとえ興味や実力がマッチしていても, 地方にいたり, 無名研究室の場合に研究者のネットワークに入れないという重大な問題があった.

また, 大学の学生や企業の担当者が, 興味を持ったテーマに詳しい教員や研究者を見つけたい, という場合も, 多様な専門性を客観的にデータベース化したシステムがないため, 学外も含めた適切な先生が見つけられないという問題点がある.

こうした研究者情報の把握のために, researchmap[1] のような公式情報の載ったデータベースに加えて, 日本の研究.com[2], JDream Expert Finder[3] といったサービスが提供されている. しかし, これらはほとんど, 研究者の実際の専門性を示す論文の内容を用いていない.[4] 代わりに論文に付与されたキーワードなどのメタ情報や論文の引用情報が用いられているが, これらは本来は補助的な情報であり, 人手で付与されたキーワードには漏れが非常に大きい. たとえば筆者が (自然言語処理の分野では珍しく) 逐次モンテカルロ法やガウス過程に詳しいことや, ベイジアンであることは, 人手による大雑把なキーワードや引用情報から推測するのは不可能である.

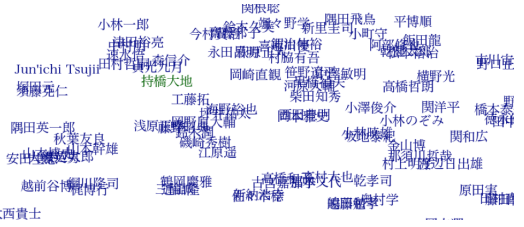


図 1: Researcher2Vec による日本の自然言語処理研究者の専門性の埋め込み. (t-SNE による可視化, 一部). 全体像は付録の図 6 を参照されたい. 論文の内容に基づいているため, 共著関係がなくとも, 専門性が近ければ近い位置に研究者が配置されている. 参考のため, 筆者を緑色で示した.

そこで本論文では, Word2vec が自己相互情報量 (PMI) 行列の行列分解と等価であることを示した Levy らの考察 [3] を応用し, 各論文のニューラル文書ベクトルを SVD によって効率的に計算し, これから「研究者ベクトル」を求める Researcher2Vec を提案する. これにより, 各研究者の専門性を低次元空間で客観的に把握して可視化することが可能になる. また, 単語をいくつか入力することで, 専門性の近い

2) https://research-er.jp/search/researchers
3) https://jdream3.com/service/expert-finder/
4) TPMS はこの情報を用いているが, 研究者別の情報は公開されておらず, 対象分野も限定的である.

研究者を線形モデルにより高速に検索することができるシステムを構築し, 一般公開した.

2　研究者の専門性の把握

研究者の専門性を把握するためには, 先に述べた (1) 引用情報やキーワードなどのメタ情報を用いるほかに, (2) LDA を代表とするトピックモデルによるアプローチがこれまでに提案されている.

(1) の方法は 1 節で述べた欠点のほかに, 最終的な可視化が多くの場合グラフに基づいており, 距離に客観的な意味がないことや, グラフに枝を張る基準や枝の重みの定義によって結果が変わってしまい, 客観性に欠けるという大きな問題点がある.

(2) のアプローチは実際に TPMS の内部で基準の一つとして用いられており, また桂井ら [4] は CiNII に登録された約 10 万件, 約 300 万語の論文概要に対してトピック数 $K=500$ の LDA を実行し, 同姓同名の共著者の曖昧性解消タスクで評価することで, 次元圧縮を行わないベクトル空間モデルより精度が高いことを示している. 筆者も日本学術振興会において, 約 11 万件, 約 3 億語の科研費文書について $K=2000$ の巨大な LDA を計算し[5], 特定の大型種目の各申請について, 審査員候補者のリストを参考情報として出力するシステムを構築して実運用を行っている [6].

ただし, LDA には計算量が大きいという以外に,

- トピック分布 θ の総和が 1 になるという制約があるため, 複数のトピックを同様に強く持つ論文や研究者をうまく扱うことができない
- θ の要素は常に正であるため, 負の相関も含めた細かい制御ができない

という問題点がある [7]. 端的にいえば, トピックモデルは文書や論文の「大雑把なジャンル」は表現できるが, その中での細かい差異を表現できないことが欠点となっている.

3　ニューラル線形モデルによる文書・研究者ベクトル

そこで本論文では, より柔軟な実数ベクトル空間への埋め込みによって, 文書 (論文) およびそれを書いた研究者を表現することを試みる. Doc2Vec[8] のようなニューラルネットと異なり, 提案法はニューラル手法と等価でありながら, 線形代数によって高速かつ効率的に学習・推薦を行うことが可能である.

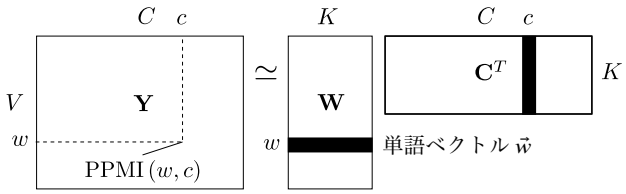


図 2: PPMI 行列の SVD による単語ベクトルの計算.

Levy ら [3] は, 単語埋め込み word2vec の計算が, 自己相互情報量を要素とする行列の行列分解と等価であることを示した. $n(w,c)$ を単語 w と周辺の文脈語 c が一定の窓内で共起した回数とし, w と c の頻度を $n(w), n(c)$ とすると, 次の Shifted Positive PMI (PPMI)

$$\mathrm{PPMI}(w,c) = \max\left(\log\frac{n(w,c)\cdot N}{n(w)n(c)} - \log k, 0\right) \quad (1)$$

を要素とする行列 $\mathbf{Y}$ を図 2 のように $\mathbf{Y} \simeq \mathbf{W}\mathbf{C}^T$ と特異値分解したときの $\mathbf{W}$ の各行は, word2vec で得られる単語ベクトルと等価となる. またこのとき, $\mathbf{C}$ の各行は文脈語の単語ベクトルとなる. ここで $N=\sum_w\sum_c n(w,c)$ とおいた. なお, k は word2vec における負例数に相当するが, [9] によって SVD を使う場合は $k=1$ (シフトを行わない) が最適であることが実験的に示されており, 以下では $k=1$ を用いた.

同様にして, 論文 d に単語 w が現れた頻度を $n(d,w)$ とし, $n(d)$ を d の単語数とするとき, $\mathrm{PPMI}(d,w)$ を要素とする行列 $\mathbf{Y}$ を図 3 のように

$$\mathbf{Y} \simeq \mathbf{D}\mathbf{W}^T \quad (2)$$

と特異値分解することで, $\mathbf{D}$ および $\mathbf{W}$ の各行として「文書ベクトル」$\vec{d}$ と「単語ベクトル」$\vec{w}$ を同時に得ることができる. これは Doc2Vec と異なり, 理論的に構成されるため学習率やミニバッチ等のハイパーパラメータの調整を必要とせず, 検索に以下で述べるような解析解が存在する. 大量のデータに対しても, 必要があれば redsvd [10] のような高速な疎行列分解を用いることで, 非常に効率的かつ一意に埋め込みベクトルを得ることができる.

3.1　研究者の専門性の可視化

こうして各論文の文書ベクトル $\vec{d}$ が得られると, それらを書いた研究者の埋め込みベクトルを計算することができる. 著者に研究者 r が含まれている論文の集合を Ω_r とすると, 研究者ベクトル $\vec{r}$ は, Ω_r に含まれる論文の文書ベクトルの平均として

$$\vec{r} = \frac{1}{|\Omega_r|}\sum_{d\in\Omega_r}\vec{d} \quad (3)$$

5)　この計算をナイーブに行うと数ヶ月を要するため, K に関わらず $O(1)$ で単語のトピックをサンプリングできる LightLDA[5] を用いて計算を行っている. この計算には数日を要する.

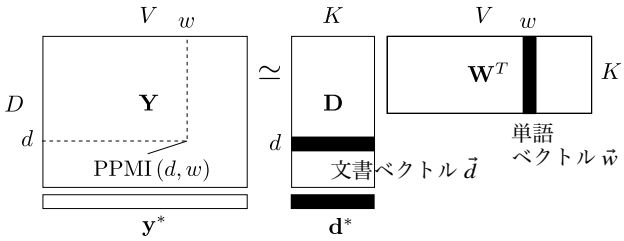


図 3: 文書–単語行列の SVD による文書ベクトルと単語ベクトルの同時計算. このとき検索クエリは仮想的な文書 $\mathbf{y}^*$ とみなせ, 対応する潜在的な文書ベクトル $\mathbf{d}^*$ が存在する.

と計算できる. これは, 埋め込み空間において各研究者を高次元ガウス分布で表現し, その平均を求めたことに相当している.[6)]

図 1 に, 言語処理学会年次大会の 20 年分の論文コーパスから計算した $K=100$ 次元の研究者ベクトル $\vec{r}$ を t-SNE によって 2 次元に可視化した例の一部を示した. 全体像は, 付録の図 6 を参照されたい. 英語の論文は含まれていないため, これはあくまで言語処理学会の論文の内容から計算された埋め込みとなっている. 処理の詳細については, 4 節で説明する.

3.2 言葉による研究者の検索

こうして研究者の埋め込みベクトルが得られると, 関連の深い研究者を言葉によって検索することが可能になる. 提案手法は従来と異なり, キーワードではなく書いた論文の内容を潜在空間で直接モデル化しているため, たとえ専門性の高い言葉でも, それに関連した研究者を検索することができる.

いま, 「談話 強化学習」のようなクエリを q とおくと, これは図 3 のように仮想的な「文書」$\mathbf{y}^*$ とみなすことができ, 対応する埋め込みベクトル $\mathbf{d}^*$ があると考えられる. 文書ベクトルと研究者ベクトルは式 (3) によって同じ埋め込み空間に存在しているから, $\mathbf{d}^*$ がわかればそれに近い研究者ベクトルを求めることで, 研究者の検索が可能になる.

$\mathbf{y}^*$ の要素は PPMI すなわち $\max\left(\log\frac{p(w|q)}{p(w)}, 0\right)$ であるから, $\mathbf{y}^*$ は q に含まれる単語に 1 が立ち[7)], 他はすべて 0 の疎ベクトルである. このとき図 3 の下段のように, 最小二乗誤差の意味で $\mathbf{y}^* \simeq \mathbf{d}^*\mathbf{W}^T$ が成り立つ $\mathbf{d}^*$ を求めたい. この式の両辺を転置して $\mathbf{y}^*, \mathbf{d}^*$

6) 分散も同様に計算することができるが, 次節で説明する推薦に分散も考慮することは今後の課題としたい.

7) この値は「仮想的な文書 q において, 各単語が平均より何倍多く出現したか」の対数であり, 1 とすることは exp(1) = 2.72 倍多く出現したと指定することを意味する. 後の実装では, クエリに「音声*」「構文解析**」のように*を付けることで, exp(2) 倍, exp(3) 倍, …が指定できるようにした.

(a) 論文内容からの研究者の検索　(b) 類似研究者の表示

図 4: 研究者ベクトルに基づく Web 検索インターフェース. 研究者情報は名前の文字列で絞り込むことができる.

の転置をそれ自身で再定義すると $\mathbf{y}^* \simeq \mathbf{W}\mathbf{d}^*$ であり, これは通常の線形回帰モデル (OLS) であることがわかる. よって, $\mathbf{d}^*$ の最適解は

$$\mathbf{d}^* = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{y}^* \tag{4}$$

であり [11] [8)], 事前に回帰行列 $\mathbf{R} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$ を計算しておけば, クエリ q に対して最適な $\mathbf{d}^*$ は

$$\mathbf{d}^* = \mathbf{R}\mathbf{y}^* \tag{5}$$

として, 一瞬で求めることができる. なお, $\mathbf{R}$ の計算は明示的に逆行列を使わず, Python では `R=numpy.linalg.solve(numpy.dot(W.T,W),W.T)` とすればよい. 語彙数を V とすると W は $V\times K$ の行列であり, よって $\mathbf{R}$ は $K\times V$ の行列となる. V や K が大きい場合にはメモリ使用量が大きくなるが[9)], 式 (5) より, $\mathbf{d}^*$ は $\mathbf{y}^*$ のうちごく少数の非零の要素に対応する $\mathbf{R}$ の列ベクトルの和であるから, $\mathbf{R}$ は事前に計算してディスクに `mmap()` しておけば, 空間計算量も最小に抑えることができる.

クエリに対応する $\mathbf{d}^*$ が得られれば, 研究者ベクトルとのコサイン距離が近い順に研究者を表示することで検索を行うことができる.

4 実装と実験

言語処理学会年次大会の 1995-2013 年の約 20 年分の発表論文データ [12] を使って実装および実験を行った. 今回は直感が働く言語処理分野で実験を行ったが, 提案法は自然言語処理に限らず, あらゆる分野の論文・言語に適用することができる.

4.1 データ

論文のタイトルおよび全文を MeCab で形態素解析し, 句読点や括弧を除いた単語の原形を使用した.

8) これに対し, Doc2Vec は出力層に softmax の形で非線形性が含まれているため, こうした解析解を得ることはできず, 検索の度に繰り返し最適化が必要になる. また, その際に巨大なパラメータもすべてメモリ上にロードしておく必要がある.

9) 日本学術振興会では筆者は現在 $V=180,000$, $K=4,000$ で計算しており, $\mathbf{R}$ は要素数 7 億 2000 万個, 4GB の行列になる.

表 1: 著者予測タスクにおける平均適合率 (MAP) の平均. 提案法は Doc2Vec の性能を上回り, 圧倒的に高速かつ省メモリである. 計算時間は $K=1000$ の場合を表している.

モデル＼K	500	1000	2000	計算時間
Researcher2Vec	**61.3%**	**66.6%**	**70.2%**	1m 8s
Doc2Vec [8]	57.4%	60.5%	62.0%	21m 9s
LDA KLdiv	56.3%	56.2%	51.2%	14h 48m
LDA cosine [4]	44.2%	48.0%	32.7%	14h 48m

語彙は頻度 10 以上の 18,135 語, 論文数は 4,082 本である. 全文を用いているため, 総単語数は 13,654,061 となった. 著者の総数 3,660 人のうち, 5 本以上の論文があった著者 499 人を実験の対象とした. 疎行列の SVD による文書ベクトル・単語ベクトルの計算は高速であり, $K=1{,}000$ のとき 1 分程度で終了した.

4.2 実装

Python の Flask モジュールを用いて図 4 のように検索サーバを実装し, 公開した. サーバは http://clml.ism.ac.jp/nlp2vec/ でアクセスすることができる. 検索結果の著者をクリックすることで, 研究者ベクトルの類似する研究者も同時に表示できる.

4.3 評価実験

論文の内容から著者を予測する実験で評価を行った. ランダムな 500 文書をテストデータ, 残りの 3,582 文書を学習データとし, 学習データの中で 5 本以上の論文を持つ研究者がテストデータの著者に含まれる場合 (410/500 文書) の各文書について, 研究者をスコア順に並べた中での真の著者集合の平均適合率 (MAP) を計算した. スコアは文書の Bag of words 表現から 3.2 節の方法で求めた文書ベクトルと, 研究者ベクトルの余弦類似度で計算する[10]. MAP は高いほど良い指標であり, MAP=1 とは, スコア順に並び換えた研究者リストの最上位がすべて真の著者で占められていることを意味する. 比較として, Doc2Vec [8] および LDA を用いた実験も行った.

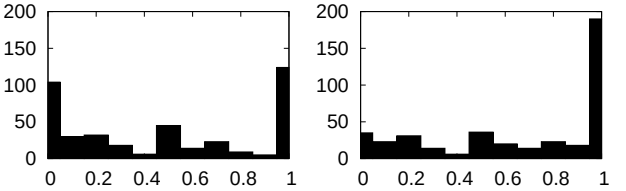
Doc2Vec 標準的に使われている gensim の Doc2Vec 実装を使用した. 条件を揃えるため分散 BoW のモデルを使用し, 学習時のパラメータは epochs=100, hs=1, alpha=0.025,sample=1e-5 である. 学習したモデルを使用し, テストデータの各文書について Doc2Vec の `infer_vector()` で繰り返し計算により文書ベクトルを求める他は, 提案手法と同様に計算する.[11]

LDA 公開している Cython 版の LDA 実装[12]を使用し, モデルが充分収束するまで 1,000 回の MCMC を行った. 学習したモデルを用い, テストデータの各文書に対して変分ベイズ EM 法でトピック分布の事後期待値 θ を求め, 式 (3) で計算した著者のトピック分布 $\vec{r}$ と比較してスコアを計算する.[13] [4] では単純に θ と $\vec{r}$ の余弦距離を計算しているが (LDA cosine), 確率分布に対してコサイン距離を適用するのは適切ではない. そこで同時に, θ と $\vec{r}$ の Kullback-Leibler ダイバージェンス

$$\mathrm{KL}(\theta||\vec{r}) = \sum_{k=1}^{K} \theta_k \log \frac{\theta_k}{r_k} \propto -\sum_{k=1}^{K} \theta_k \log r_k \qquad (6)$$

をスコアとする実験 (LDA KLdiv) も行って比較した.

結果 表 1 に, 各モデルの平均適合率の結果とモデルの計算時間を示した. 提案手法は Doc2Vec や LDA を超えて最高性能を達成し, モデルの計算もきわめて高速である. また図 5 のように, 平均適合率は多くの論文で 1 に集中し, それ以外は英語論文や, 教員の専門と異なる学生の論文の場合となっていた. 提案手法での K の最大値は SVD の性質から D(=3582) 以下であり, $K=3000$ での精度は 71.6%であった.



(a) LDA KLdiv (b) Researcher2Vec (提案手法)

図 5: テストデータの各論文の著者予測の平均適合率(MAP)のプロット. 提案手法は多くの場合に MAP=1 を達成しており, それ以外は英語論文などのノイズとなっている.

5 まとめ

本研究では, ニューラル手法と等価な文書ベクトルを効率的な SVD によって計算し, 研究者を論文の内容で可視化・検索できるシステムを構築し, 評価した. 実験により, 提案手法はきわめて効率的かつ Doc2Vec およびトピックモデルより高い著者予測性能を見せることがわかった. 今後は arXiv や ACL anthology など, 英語論文への拡張を検討したい.

10) このとき, 文書 d に対する観測値 $\mathbf{y}$ の要素である PPMI (d,w) は $\max\left(\log\frac{p(d,w)}{p(d)p(w)}, 0\right) = \max\left(\log\frac{p(w|d)}{p(w)}, 0\right)$ と書けるから, 訓練データの各単語の確率 $p(w)$ を保存しておけば, $p(w|d)$ は文書 d 内での最尤推定で得られるため, $\mathbf{y}$ は容易に計算できる.

11) 学習時・テスト時の各文書に対する繰り返し回数は 100 としたが, これ以上増やしても性能に変化はみられなかった.

12) http://chasen.org/~daiti-m/dist/lda-python/

13) Ω_r に含まれる論文の θ を生成するディリクレ事前分布を著者毎に Newton 法によって計算し, 式 (6) の KL 距離の期待値を求めるベイズ的な方法も検討したが, あまり高い性能が得られなかった.

参考文献

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, Vol. 3, pp. 993–1022, 2003.

[2] Laurent Charlin and Richard S. Zemel. The Toronto Paper Matching System: An automated paper-reviewer assignment system. In *ICML 2013*, 2013.

[3] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27*, pp. 2177–2185, 2014.

[4] Marie Katsurai, Ikki Ohmukai, and Hideaki Takeda. Topic Representation of Researchers' Interests in a Large-Scale Academic Database and Its Application to Author Disambiguation. *IEICE Trans. Inf. Syst.*, Vol. E99-D, No. 4, pp. 1010–1018, 2016.

[5] Jinhui Yuan et al. LightLDA: Big Topic Models on Modest Computer Clusters. In *WWW 2015*, pp. 1351–1361, 2015.

[6] 日本学術振興会. 学術情報分析センター 平成 30 年度活動報告, 2019. https://www.jsps.go.jp/j-csia/data/h30/JSPS-CSIA_REPORT_2018_4.pdf.

[7] 持橋大地, 吉井和佳, 後藤真孝. ガウス過程に基づく連続空間トピックモデル. 情報処理学会研究報告 2013-NL-213, Vol. 11, pp. 1–8, 2013.

[8] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *ICML 2014*, pp. 1188–1196, 2014.

[9] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, Vol. 3, pp. 211–225, 2015.

[10] Daisuke Okanohara. redsvd: RandomizED Singular Value Decomposition, 2010. https://code.google.com/archive/p/redsvd/.

[11] 持橋大地, 大羽成征. ガウス過程と機械学習. 機械学習プロフェッショナルシリーズ. 講談社, 2019.

[12] 言語処理学会 20 周年記念 年次大会コーパス, 2014. https://www.anlp.jp/anniversary/20th_anniversary.html.

付録

研究者ベクトルの t-SNE による 2 次元への可視化. 高次元の研究者ベクトルは精度が高すぎて「それ自身 (あるいは強い共著者) としか似ていない」状態になり, 全体的な構造を反映しないため, あえて $K = 100$ 次元のベクトルを可視化している. 可視化には Scikit-learn の t-SNE 実装を用いた.[14] 紫は明らかなグループについて筆者が付けた注釈であり, 他にも様々な構造を見て取ることができる.

図 6: 研究者ベクトルの 2 次元への可視化.

14) t-SNE は実行の度に確率的に結果が変わるため, これが唯一の可視化というわけではない.