Insight Factory R&Dランチタイム・ミーティング

テキスト分析と分散表現

2026.02

小野 滋

INSIGHT FACTORY

目次

1. イントロダクション

1.1 テキストの分散表現とは

- 単語や文章を、数字を並べたもの (ベクトル)として表現すること
  - 「埋め込み」ということも多い (後述)
  - テキスト分析において広く用いられている

ChatGPTくんが作ってくれたイメージ図。
日本語がなんだか変

1.2 本日の目的

- 目的1. リサーチャーのみなさまに、テキストの分散表現についてわかりやすく説明すること
  - リサーチの企画にあたっては、「なにができるか」「どのくらい大変か」についてイメージを持っておくことが必要
  - マーケティング・リサーチにとって、テキスト分析は重要な手段のひとつ
    - テキスト分析の可能性と工数は、分散表現の普及によって大きく変わった
    - しかし、一般的なテキスト分析の教科書では、分散表現についての解説はなかなか出てこない

金 (2021) の目次:

1. テキストアナリティクス
2. テキストのクリーニングと関連技法
3. テキスト処理のツール
4. テキストの基本統計と視覚化
5. 共起とbigramのネットワーク分析
6. テキストの特徴分析
7. トピック分析
8. テキストのクラスタリング
9. アソシエーション分析法による共起分析
10. テキストの分類分析
11. テキストデータを用いた予測
12. 特徴量選択
13. **分散表現**

- 目的2. 本日のトークを機に、私が勉強すること
  - この分野について知識が欠けている。すごく欠けている
  - こういう機会にキャッチアップしたい

1.3 おわび

- つまり、みなさまは私の勉強に付き合わされているわけです。すいません
- たくさんの誤りを含んでいると思います。すいません

スライディング土下座

1.4 分析例

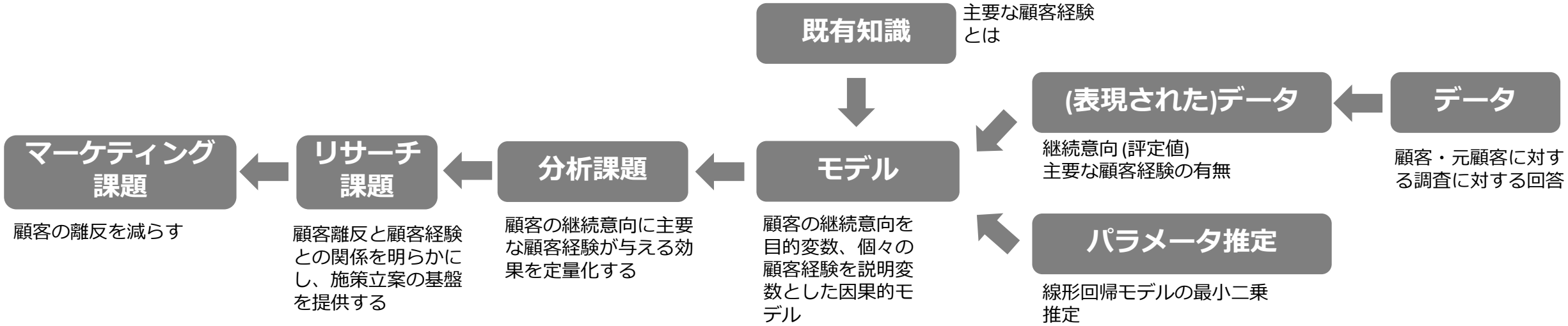
- 本資料で紹介した分析例について、詳細を以下で公開しています:
  - https://elsur.jpn.org/202602Embedding/

2. テキスト分析における分散表現の意義

2.1 データ分析プロセス

- データ分析とは
  - データから有用な知見を得るための取り組み
  - マーケティング・リサーチでは... リサーチ課題を解決するための方法のひとつ

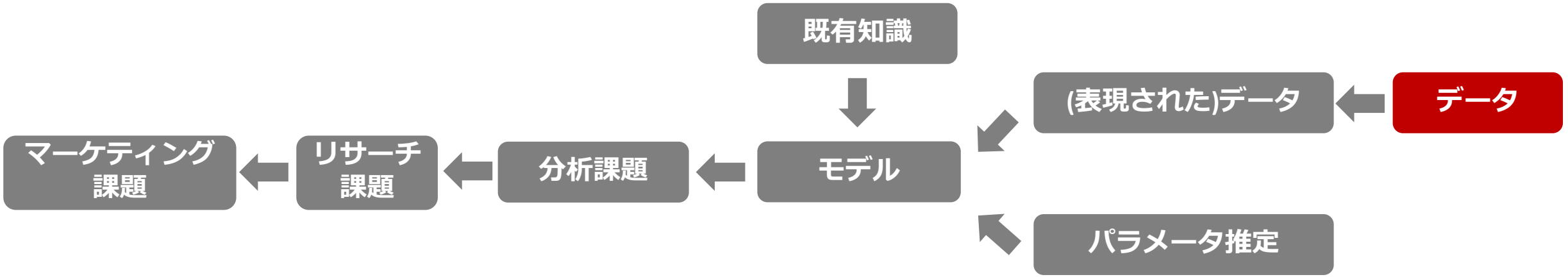
- データ分析プロセスの構成要素
  - 既有知識に基づき、モデルを設計する
  - 必要なデータを収集し、それを表現する
  - モデルのパラメータを推定する
  - 推定結果に基づき、分析課題への解を提供する



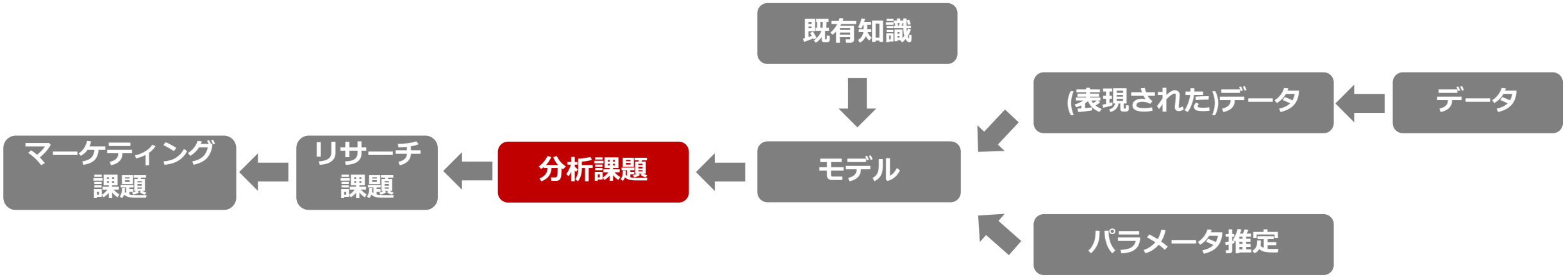
2.2 マーケティング・リサーチとテキスト分析

- テキスト分析
  - データ分析のうち、データが自然言語(日本語、英語など)で書かれたものである場合を指す

- マーケティング・リサーチにおけるテキスト・データ
  - 調査対象者が生成したテキスト
    - 調査で得た**自由記述・発言記録**など
  - ユーザが生成したテキスト(**UGC**)
    - SNSへの投稿、クチコミサイトへの投稿など
  - 製品・サービスについての**マーケティング素材**
    - 製品テストで用いたコンセプト文、広告など

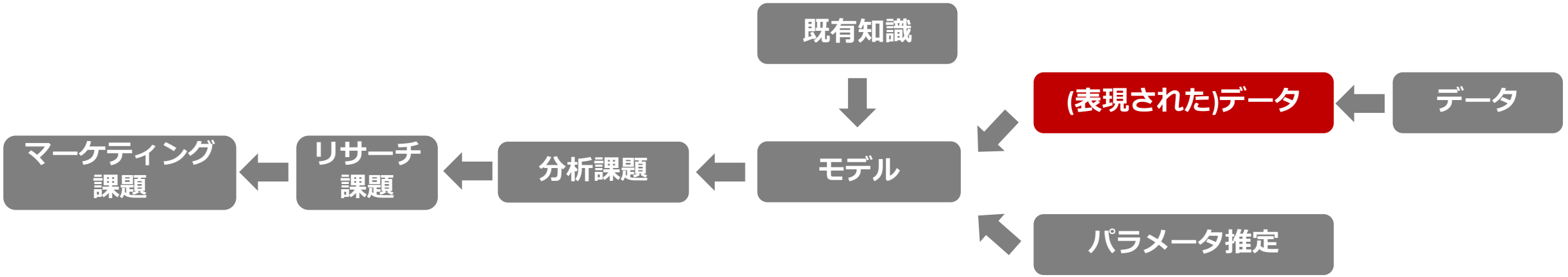


- マーケティング・リサーチにおけるテキスト分析の課題
  - 典型的には、次の3つ
  - **視覚化**
    - さまざまなテキストの内容の分布を、視覚的に捉えたい
  - **分類・クラスタリング**
    - さまざまなテキストを、なんらかの基準に基づいて分類したい
    - さまざまなテキストを、基準なしでクラスタリングしたい
  - **説明・予測**
    - さまざまなテキストの内容を、それを生み出した人や状況に基づいて説明したい
    - さまざまなテキストの内容から、それに対応する行動を予測したい



2.3 テキスト・データの表現

- テキスト・データをなんらかの形で表現したい
  - モデルに基づいて有用な知見を引き出せるような表現が望ましい



- なんだか役に立ちそうにない表現の例

テキスト

これは、私が小さいときに、村の茂平というおじいさんから聞いた話です。

表現

位置	文字
1	こ
2	れ
3	は
..	…

文の文字数
34

文字	出現頻度
あ	0
い	0
…	…
お	1
..	…

- 広く用いられている表現の例

テキスト

これは、私が小さいときに、村の茂平というおじいさんから聞いた話です。

形態素解析

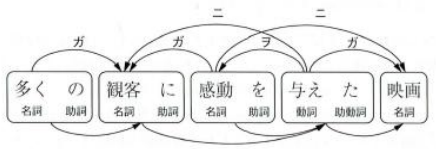
表現

単語	(品詞)	出現頻度
が	(格助詞)	1
から	(格助詞)	1
これ	(代名詞)	1
さん	(接尾辞)	1
た	(助動詞)	1
…	…	…
聞く	(動詞)	1
話	(名詞)	1
..	…	…

Bag of Words (BoW) 表現

- テキストに含まれる単語とその出現頻度のみに注目し、テキストを頻度のベクトルとして表現している
- 形態、語順、構文、意味的依存性を無視している

- もっと豊かな表現を目指して
  - 理論駆動的アプローチ

言語学の部門	自然言語処理における分野	
形態論	形態素解析	例)「外国人参政権について議論した」 → 「外国人」(名詞)「参政権」(名詞)「に」(格助詞)「つく」(動詞)「て」(助詞)… → 「外国」(名詞)「人参」(名詞)「政権」(名詞)「に」(格助詞)「つく」(動詞)「て」(助詞)… どっちが適切？
統語論	構文解析	例) 依存構造解析 (係り受け解析)  図 1.4 【例文 2】「新聞で汚れた窓を掃除した」の係り受け木の例
意味論	意味解析	例) 述語項構造解析 (格解析)  図 1.6 【例文 3】「多くの観客に感動を与えた映画」の述語項構造解析結果
語用論	計算語用論	例) 発話行為タグ付け 「窓を開けてください」 → 要求 「必ずやります」 → 約束 「話を戻しますが」 → トピック制御 …

形態素解析の文例と図の出典: 岡崎ほか(2022)

- データ駆動的アプローチ
  - テキストや大量の言語資料(コーパス)から、単語や文章の意味的関係性を学習する
  - 単語や文章を実数ベクトルとして表現する (**分散表現**)
    - 意味が近い単語・文章を、近いベクトルとして表現する
  - ベクトルの各要素がなにを表しているのかははっきりしない
    - opp. 局所表現 (例, BoW表現)

  - 自然言語処理のさまざまな分野で、謎の大成功を収めている

2.4 テキスト分析における分散表現の意義

- マーケティング・リサーチにおけるテキスト分析で、なぜ分散表現が有用なのか？

- → **一般的な言語知識を簡単に取り入れることができるから**
  - 大量の言語資料に基づいて学習されたモデル (**事前学習済みモデル**)を使って、手元のテキストに含まれる文章・単語を分散表現する
  - 得られた分散表現は、文章・単語の背後にある一般的な言語知識を取り込んでいる

BoW表現

	…	…	…	…	青空	晴れ	…	…	…	…	…
文章1	1	1	0	1	0	0	0	0	1	0	…
文章2	1	0	1	0	**1**	0	1	0	0	0	…
文章3	0	0	0	0	0	**1**	0	1	1	1	…
…	…	…	…	…	…	…	…	…	…	…	…

「青空」と「晴れ」は似た意味を持つ単語だが、そのことが表現されていない

事前学習済みモデルによる分散表現

文章1	+1.27	-2.09	+0.01	+0.05	-3.21	…
文章2	…	…	…	…	…	…
文章3	…	…	…	…	…	…
…	…	…	…	…	…	…

「青空」が出てくる文章と「晴れ」が出てくる文章は近いベクトルになっている (と期待される)

2.5 次章からの内容

- 以下では、テキストの分散表現について、次の3つにわけて紹介する：

(説明のための準備)

- **特異値分解に基づくテキストの分散表現** (4章)
  - 手元のテキスト・データに基づき、そこに含まれている単語・文章の分散表現を得て、その後の分析に用いる

← **特異値分解** (3章)

- **ニューラルネットワークに基づく単語の埋め込み表現** (6章)
  - 大量の言語資料を用いて、単語の分散表現を得る
  - 手元のテキスト・データについて、そこに含まれている単語を分散表現に換え、その後の分析に用いる

← **ニューラル・ネットワーク** (5章)

- **ニューラル言語モデルに基づく文章の埋め込み表現** (8章)
  - 大量の言語資料を用いて、文章を分散表現に換えるためのモデルを構築しておく
  - 手元のテキスト・データについて、そこに含まれている文章を分散表現に換え、その後の分析に用いる

← **ニューラル言語モデル** (7章)

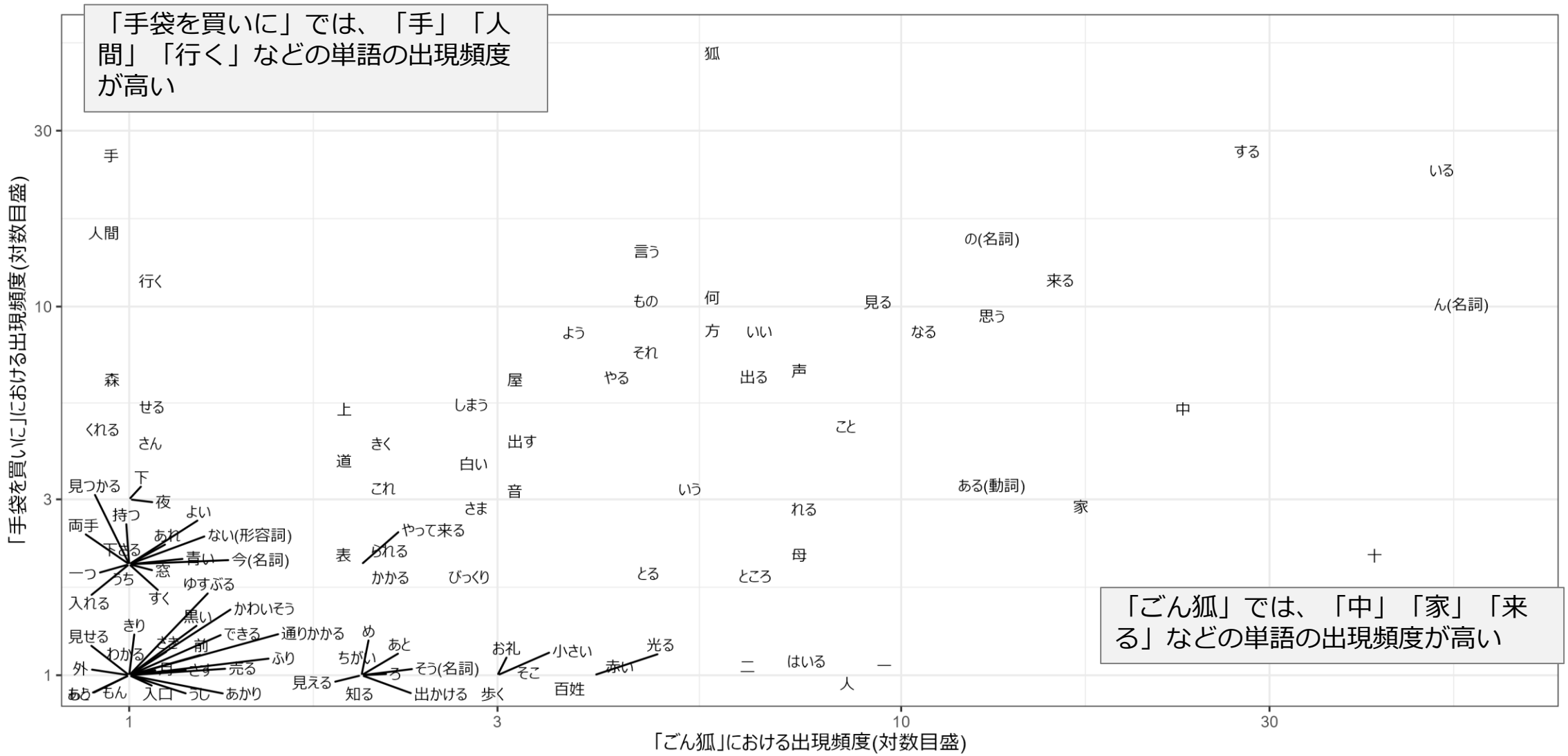
2.6 説明に用いるテキスト

- テキストの例として、新美南吉「ごん狐」「手袋を買いに」を使用する
  - 「ごん狐」には180文,「手袋を買いに」は112文が含まれている

文番号	物語番号	物語内の文番号	文
1	1	1	これは、私が小さいときに、村の茂平というおじいさんからきいたお話です。
2	1	2	むかしは、私たちの村のちかくの、中山というところに小さなお城があって、中山さまというおとのさまが、おられたそうです。
3	1	3	その中山から、少しはなれた山の中に、
4	1	4	「ごん狐」
5	1	5	という狐がいました。
	…	…	
176	1	176	「ごん、お前だったのか。
177	1	177	いつも栗をくれたのは」
178	1	178	ごんは、ぐったりと目をつぶったまま、うなずきました。
179	1	179	兵十は火縄銃をばたりと、とり落しました。
180	1	180	青い煙が、まだ筒口から細く出ていました。
181	2	1	寒い冬が北方から、狐の親子の棲んでいる森へもやって来ました。
182	2	2	或朝洞穴から子供の狐が出ようとしましたが、
183	2	3	「あっ」
184	2	4	と叫んで眼を抑えながら母さん狐のところへころげて来ました。
185	2	5	「母ちゃん、眼に何か刺さった、ぬいて頂戴早く早く」
	…	…	
288	2	108	「まあ！」
289	2	109	とあきれましたが、
290	2	110	「ほんとうに人間はいいものかしら。
291	2	111	ほんとうに人間はいいものかしら」
292	2	112	とつぶやきました。

「ごん狐」のあらすじ：
いたずら好きな狐のごんは、兵十の大切なうなぎを盗んだ罪を悔い、こっそり償いを続ける。しかしそれを知らぬ兵十は… 宿命の悲劇がいま幕を開ける！

「手袋を買いに」のあらすじ：
雪の夜、子狐は愛する母狐を残し、ひとり町へと旅立つ。子狐は手袋を手に入れることができるのか？ 信じ合う心は生物種を超えるのか？ 感動の巨編！



名詞・動詞・形容詞のうち、いずれの物語でも出現している単語を示す

3. 特異値分解

(3-4章の内容)

(説明のための準備)

- **特異値分解に基づくテキストの分散表現** (4章) ← **特異値分解** (3章)
  - 手元のテキスト・データに基づき、そこに含まれている単語・文章の分散表現を得て、その後の分析に用いる

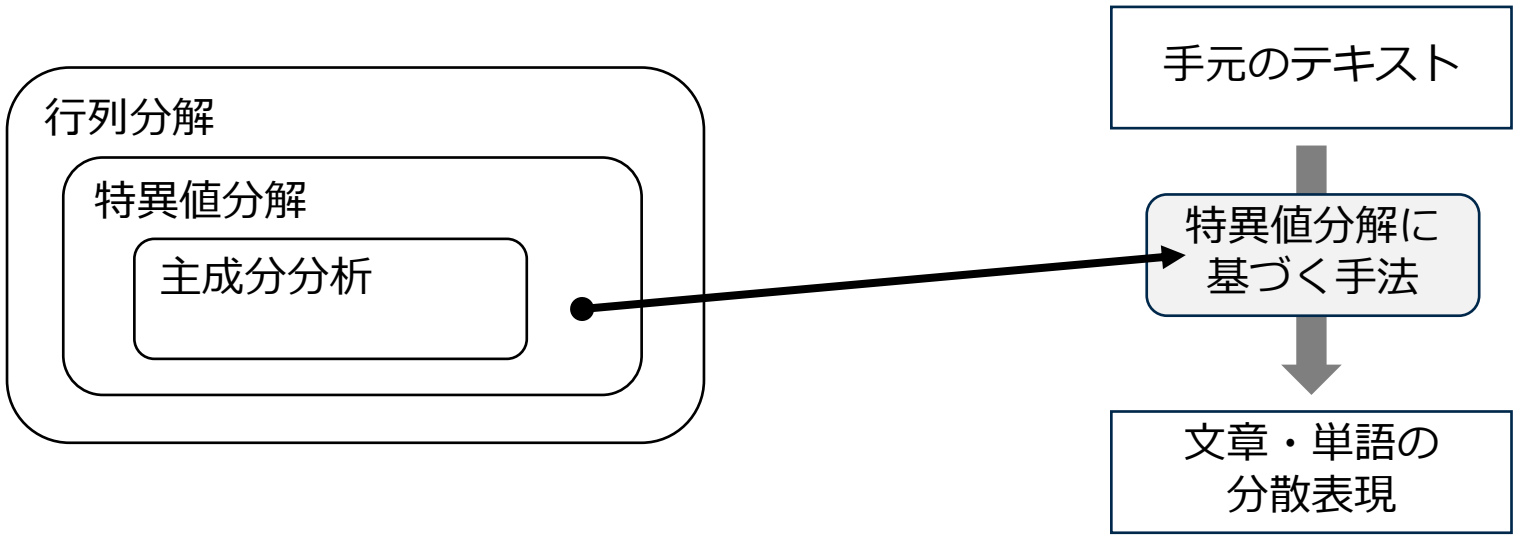
- **ニューラルネットワークに基づく単語の埋め込み表現** (6章) ← **ニューラル・ネットワーク** (5章)
  - 大量の言語資料を用いて、単語の分散表現を得る
  - 手元のテキスト・データについて、そこに含まれている単語を分散表現に換え、その後の分析に用いる

- **ニューラル言語モデルに基づく文章の埋め込み表現** (8章) ← **ニューラル言語モデル** (7章)
  - 大量の言語資料を用いて、文章を分散表現に換えるためのモデルを構築しておく
  - 手元のテキスト・データについて、そこに含まれている文章を分散表現に換え、その後の分析に用いる

- 3章では、4章の準備として…
  - 特異値分解について紹介する
  - 主成分分析についておさらいし、特異値分解との関係を示す
- 4章では
  - 特異値分解を利用して、手元のテキストから文章・単語の分散表現を得る方法を紹介する

3章. 特異値分解

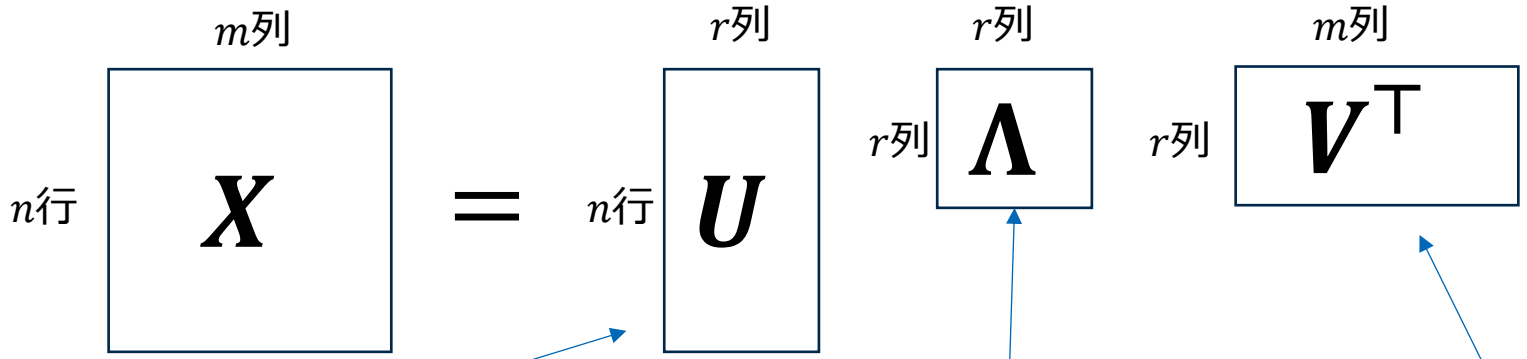
4章. 特異値分解に基づく分散表現



3.1 特異値分解

- 数の分解
  - 「18を2つの数の積として表現してください」
    - 無数の分解方法がある
  - 「18を2つの数 a, b の積として表現してください。ただし、bはaの2倍になるようにしてください」
    - 分解方法は2つに絞られる

- 行列の分解
  - 「行列$\boldsymbol{X}$を3つの行列の積として表現してください」
    - 無数の分解方法がある
  - 「行列$\boldsymbol{X}$を、次のページで説明する性質を持っている3つの行列の積 $\boldsymbol{U\Lambda V}^{\mathsf{T}}$ として表現してください」
    - この分解を**特異値分解** (singular value decomposition; SVD) という
    - あらゆる$\boldsymbol{X}$について解が存在する
    - しかも、解はほぼひとつ
      - $\boldsymbol{\Lambda}$は一意に決まる
      - $\boldsymbol{U}, \boldsymbol{V}$は、特異値が重複しない限り、符号を除いて一意に決まる



- 直交行列
  - それぞれの列において、要素の二乗和は1
  - 列同士を掛けた和は0
- 各列を**左特異ベクトル**という

- 対角行列
  - 対角要素だけに0でない値が入っている正方行列
- rは行列$\boldsymbol{X}$の階数(ランク)
  - m,nの小さいほうの値、ないしそれより小さい値
- 対角要素に入っているr個の値を**特異値**という
  - すべて0より大きい
  - 左上から大きい順に並べる

- $\boldsymbol{V}$は直交行列
  - $\boldsymbol{V}^{\mathsf{T}}$のそれぞれの行において、要素の二乗和は1
  - $\boldsymbol{V}^{\mathsf{T}}$の行同士を掛けた和は0
- $\boldsymbol{V}$ の各列($\boldsymbol{V}^{\mathsf{T}}$の各行)を**右特異ベクトル**という

- 数値例

$$\underset{(n,m)}{\boldsymbol{X}} = \underset{(n,r)}{\boldsymbol{U}} \quad \underset{(r,r)}{\boldsymbol{\Lambda}} \quad \underset{(r,m)}{\mathbf{V}^{\top}}$$

$$\begin{bmatrix} 1 & 10 & 6 \\ 7 & 9 & 11 \\ 4 & 8 & 12 \\ 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 & -.27 \\ +.63 & +.35 & +.44 \\ +.59 & +.34 & -.52 \\ +.22 & -.29 & +.66 \end{bmatrix} \begin{bmatrix} 24.78 & 0 & 0 \\ 0 & 4.98 & 0 \\ 0 & 0 & 3.26 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \\ +.84 & +.13 & -.51 \end{bmatrix}$$

- 直交行列
  - それぞれの列において、要素の二乗和は1
  - 列同士を掛けた和は0
- 各列を**左特異ベクトル**という

- 対角行列
  - 対角要素だけに0でない値が入っている正方行列
- rは行列$\boldsymbol{X}$の階数(ランク)
  - m,nの小さいほうの値、ないしそれより小さい値
- 対角要素に入っているr個の値を**特異値**という
  - すべて0より大きい
  - 左上から大きい順に並べる

- $\boldsymbol{V}$は直交行列
  - $\boldsymbol{V}^{\top}$のそれぞれの行において、要素の二乗和は1
  - $\boldsymbol{V}^{\top}$の行同士を掛けた和は0
- $\boldsymbol{V}$ の各列($\boldsymbol{V}^{\top}$の各行)を**右特異ベクトル**という

- おさらい: 行列の積の求め方

$$\begin{bmatrix} 1 & 10 & 6 \\ 7 & 9 & 11 \\ 4 & 8 & 12 \\ 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 & -.27 \\ +.63 & +.35 & +.44 \\ +.59 & +.34 & -.52 \\ +.22 & -.29 & +.66 \end{bmatrix} \begin{bmatrix} 24.78 & 0 & 0 \\ 0 & 4.98 & 0 \\ 0 & 0 & 3.26 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \\ +.84 & +.13 & -.51 \end{bmatrix}$$

$$\begin{array}{lllll} +0.63 & \times 24.78 & \times +.64 & = & +10.1 \\ +0.35 & \times 4.98 & \times -.75 & = & -1.3 \\ +0.44 & \times 3.26 & \times +.13 & = & +0.2 \\ \hline & & & \text{計} & 9 \end{array}$$

- 行列$\boldsymbol{U}$の行は、元の行列$\boldsymbol{X}$の行を表している

$$\begin{bmatrix} 1 & 10 & 6 \\ 7 & 9 & 11 \\ 4 & 8 & 12 \\ 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 & -.27 \\ +.63 & +.35 & +.44 \\ +.59 & +.34 & -.52 \\ +.22 & -.29 & +.66 \end{bmatrix} \begin{bmatrix} 24.78 & 0 & 0 \\ 0 & 4.98 & 0 \\ 0 & 0 & 3.26 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \\ +.84 & +.13 & -.51 \end{bmatrix}$$

元の行列の2行目の情報を含んでいる箇所

- 行列$\boldsymbol{V}^{\mathsf{T}}$の列 ($=\boldsymbol{V}$の行)は、元の行列$\boldsymbol{X}$の列を表している

$$\begin{bmatrix} 1 & 10 & 6 \\ 7 & 9 & 11 \\ 4 & 8 & 12 \\ 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 & -.27 \\ +.63 & +.35 & +.44 \\ +.59 & +.34 & -.52 \\ +.22 & -.29 & +.66 \end{bmatrix} \begin{bmatrix} 24.78 & 0 & 0 \\ 0 & 4.98 & 0 \\ 0 & 0 & 3.26 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \\ +.84 & +.13 & -.51 \end{bmatrix}$$

元の行列の2列目の情報を含んでいる箇所

- 特異値分解と低ランク近似
  - 行列$\boldsymbol{U}$の左からk列、行列$\boldsymbol{\Lambda}$の左上からk行k列, 行列$\boldsymbol{V}^{\mathsf{T}}$の上から$k$行を取り出し、それぞれ$\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Lambda}}, \tilde{\boldsymbol{V}}^{\mathsf{T}}$としよう
  - $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{V}}^{\mathsf{T}}$は、$X$の良い近似となる
    - 近似の良さを差の二乗和で評価した時、 $\tilde{\boldsymbol{X}}$はサイズ$n \times m$、階数kの行列のなかでもっとも良い近似である

$$\boldsymbol{X} = \boldsymbol{U} \quad \boldsymbol{\Lambda} \quad \boldsymbol{V}^{\mathsf{T}}$$

$$\begin{bmatrix} 1 & 10 & 6 \\ 7 & 9 & 11 \\ 4 & 8 & 12 \\ 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 & -.27 \\ +.63 & +.35 & +.44 \\ +.59 & +.34 & -.52 \\ +.22 & -.29 & +.66 \end{bmatrix} \begin{bmatrix} 24.78 & 0 & 0 \\ 0 & 4.98 & 0 \\ 0 & 0 & 3.26 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \\ +.84 & +.13 & -.51 \end{bmatrix}$$

近似 ⇑

$$\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{U}} \quad \tilde{\boldsymbol{\Lambda}} \quad \tilde{\boldsymbol{V}}^{\mathsf{T}}$$

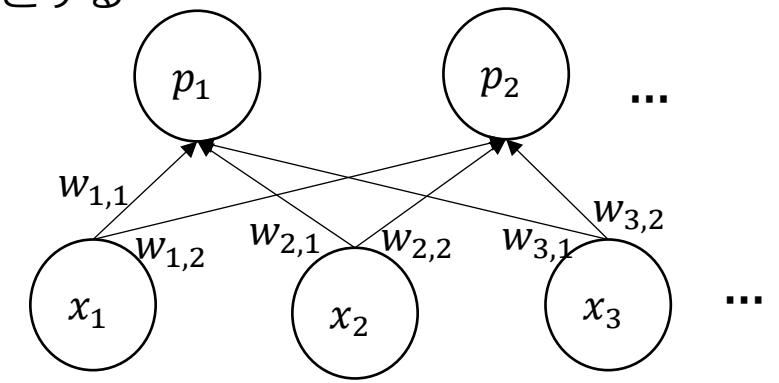
⇓ 灰色部分を捨てる

$$\begin{bmatrix} 1.7 & 10.1 & 5.5 \\ 5.8 & 8.8 & 11.8 \\ 5.5 & 8.2 & 11.1 \\ 1.2 & 4.7 & 3.1 \end{bmatrix} = \begin{bmatrix} +.44 & -.81 \\ +.63 & +.35 \\ +.59 & +.34 \\ +.22 & -.29 \end{bmatrix} \begin{bmatrix} 24.78 & 0 \\ 0 & 4.98 \end{bmatrix} \begin{bmatrix} +.31 & +.64 & +.69 \\ +.42 & -.75 & +.50 \end{bmatrix}$$

3.2 主成分分析

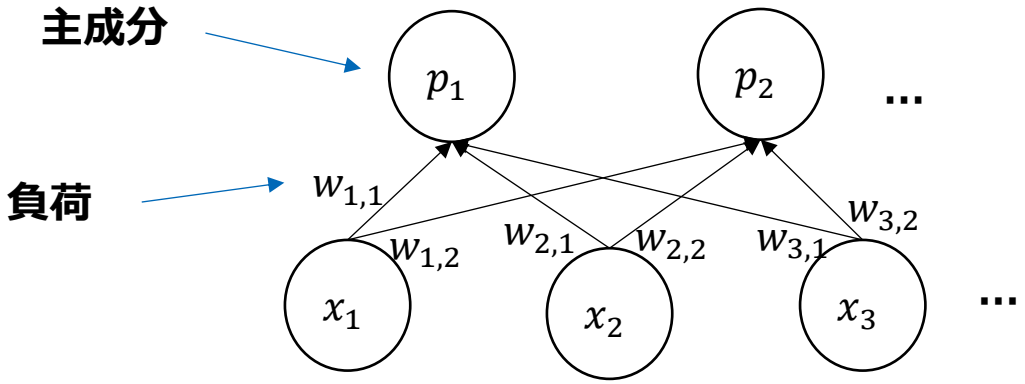
- データ分析のための重要な考え方

- 基本的な発想
  - たくさんの変数を、より少数の変数に縮約したい
  - それぞれの変数を中心化した変数(個々の値から平均を引いた変数)を、 $x_1, \ldots, x_m$とする
  - 元の変数$x_1, \ldots, x_m$の線形和(重みづけ和)によって、変数p_1, p_2 …をつくろう
    - $p_1 = w_{1,1}x_1 + w_{2,1}\,x_2 + \cdots + w_{m,1}x_m$
    - $p_2 = w_{1,2}x_1 + w_{2,2}\,x_2 + \cdots + w_{m,2}x_m$
    - …
  - ただし、次の条件を満たしたい



- 変数p_1の分散をできるだけ大きくしたい
  - 重み$w_{1,1}, \ldots, w_{m,1}$は二乗和1とする。以下同様
- 変数p_2は、p_1 との相関が0である変数とし、かつ分散をできるだけ大きくしたい
- 変数p_3は、p_1, p_2 との相関が0である変数とし、かつ分散をできるだけ大きくしたい
- …

- この条件を満たす変数$p_1, p_2, \ldots$を**主成分**という
  - 重み$w_{j,1}, \ldots, w_{j,m}$を、第j主成分の**負荷**という
  - r個の主成分を求めることができる
    - rは元のデータ行列の階数
    - (行数と列数の小さいほうの値、ないしそれより小さい値となる)



- 数値例
  - 10人の子どものテストの成績を用いる
  - 中心化して主成分分析を行い、3つの主成分を得た

成績

国語	英語	音楽
86	79	45
71	75	95
42	43	40
62	58	80
96	97	75
39	33	50
50	53	65
78	66	40
21	44	90
89	92	50

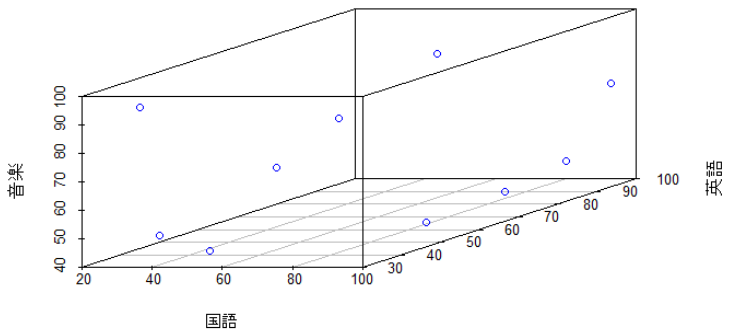
成績 (中心化)

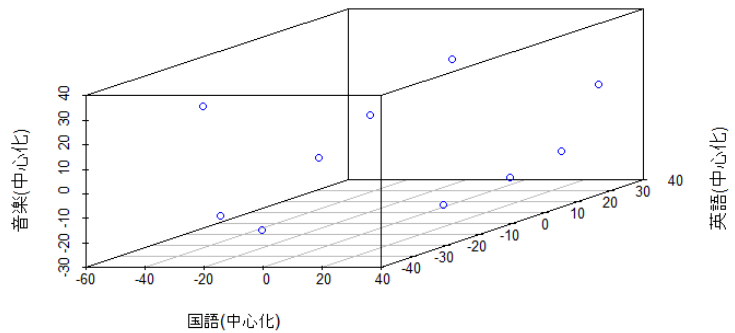
国語	英語	音楽
22.6	15	-18
7.6	11	32
-21.4	-21	-23
-1.4	-6	17
32.6	33	12
-24.4	-31	-13
-13.4	-11	2
14.6	2	-23
-42.4	-20	27
25.6	28	-13

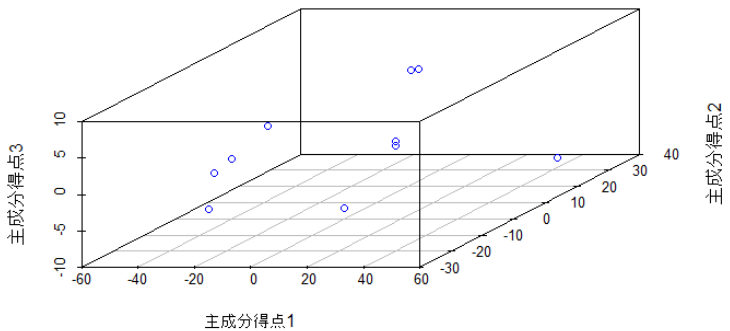
主成分得点

主成分1	主成分2	主成分3
-28.30	-16.09	-0.07
-10.29	32.96	3.25
27.96	-25.25	-2.99
6.30	15.46	6.94
-45.09	16.18	-0.83
37.49	-17.33	4.45
17.44	0.64	-0.15
-14.24	-23.07	3.33
47.30	25.25	-7.22
-38.56	-8.76	-6.70

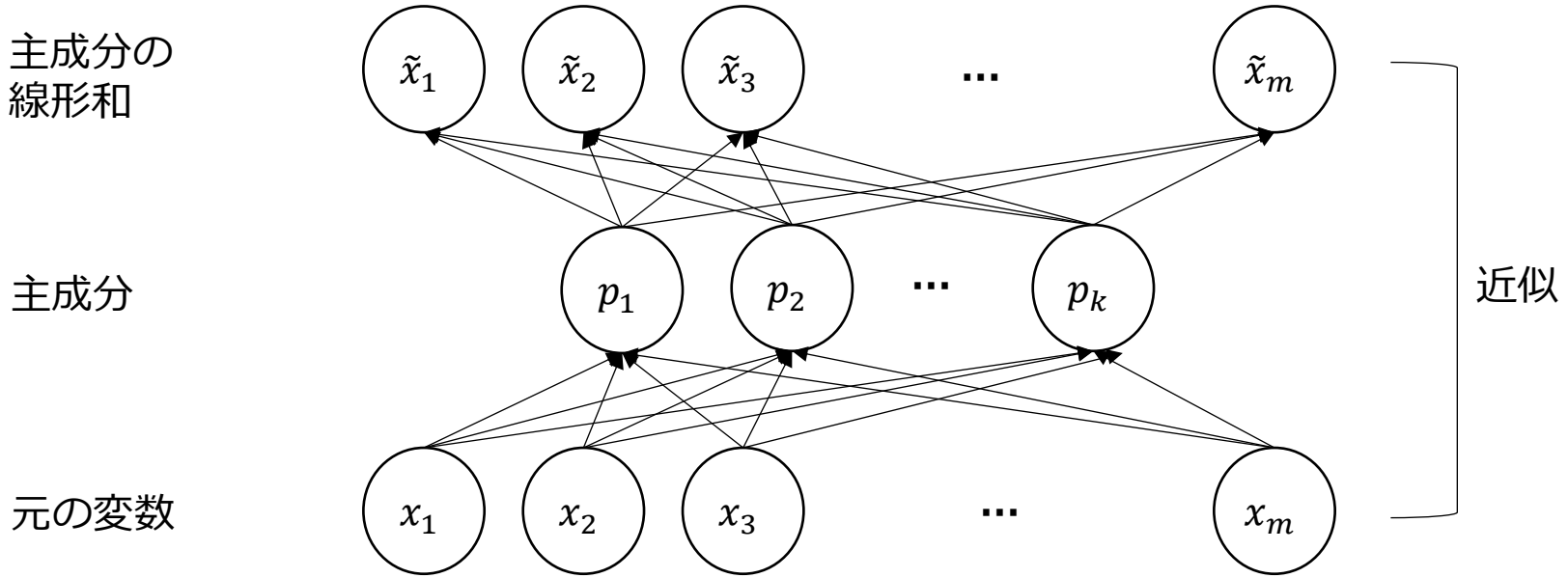
- ここで主成分分析は、元のデータの行が布置する3次元空間を、データ点の重心を原点として回転し、(左図の空間を上から見て)もっとも奥行きがない布置へと回転している、と捉えることもできる







- 主成分の線形和によって、元の中心化済み変数を近似できる
  - いま、元の変数$x_1, \ldots, x_m$の線形和によってなんらかの$k(< r)$個の変数$p_1, \ldots, p_k$をつくり、さらに、元の変数を$p_1, \ldots, p_k$の線形和によって近似しよう
    - 近似の良さは誤差二乗和で評価しよう
  - 近似が最も良くなるのは、 $p_1, \ldots, p_k$として第1, ..., k主成分を使ったときである
  - 主成分をできるだけたくさん作れば ($k = r$ にすれば)、元の変数とぴったり一致する



- 数値例

成績 (中心化)

国語	英語	音楽
22.6	15	-18
7.6	11	32
-21.4	-21	-23
-1.4	-6	17
32.6	33	12
-24.4	-31	-13
-13.4	-11	2
14.6	2	-23
-42.4	-20	27
25.6	28	-13

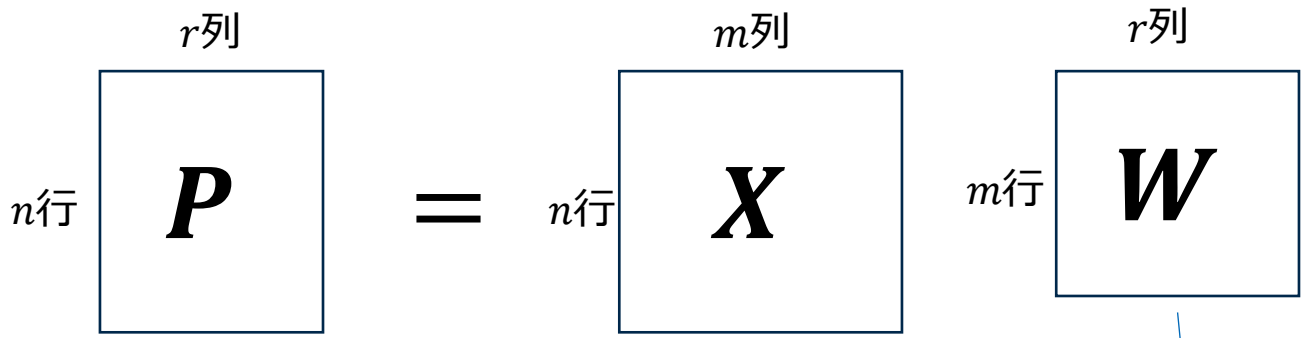
主成分1,2による近似

国語	英語	音楽
22.6	14.9	-18.0
5.5	13.4	31.3
-19.5	-23.2	-22.4
-5.9	-0.9	15.6
33.1	32.4	12.2
-27.3	-27.7	-13.9
-13.3	-11.1	2.0
12.5	4.4	-23.7
-37.7	-25.3	28.5
29.9	23.1	-11.7

3.3 主成分分析と特異値分解

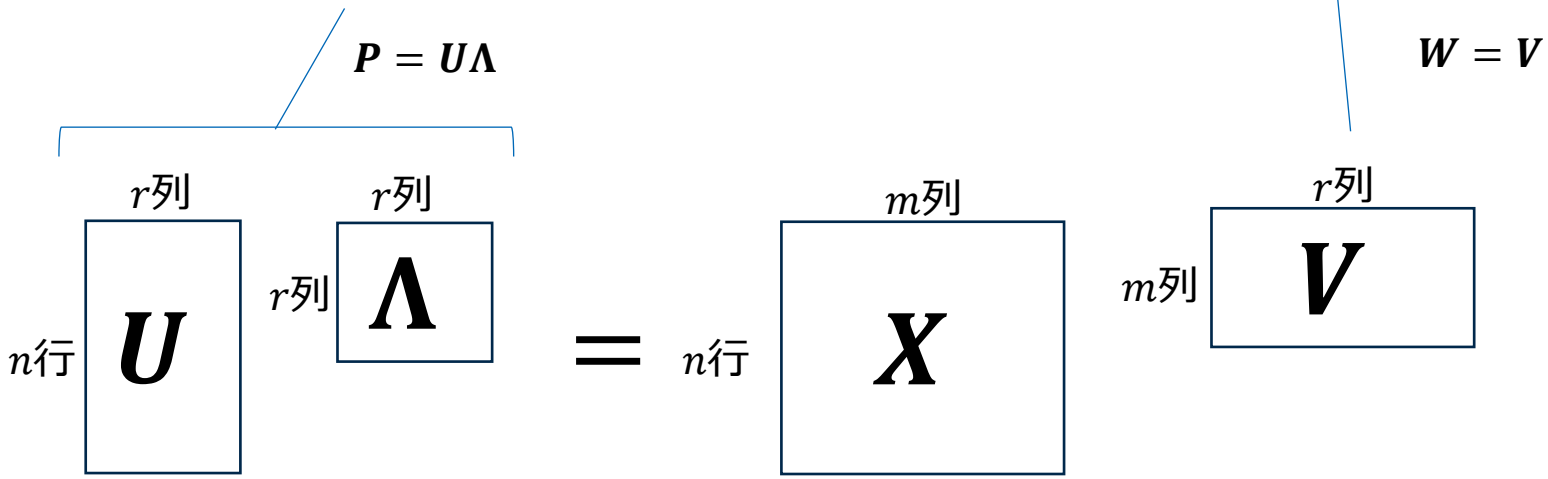
- 主成分分析は中心化済みデータ行列の特異値分解と同値
  - 中心化済みデータ行列$\boldsymbol{X}$の主成分行列を$\boldsymbol{P}$, 負荷行列を$\boldsymbol{W}$とする。すなわち $\boldsymbol{P} = \boldsymbol{XW}$
  - $\boldsymbol{X}$の特異値分解を $\boldsymbol{X} = \boldsymbol{U\Lambda V}^{\mathrm{T}}$ とする
  - $\boldsymbol{P} = \boldsymbol{U\Lambda}, \boldsymbol{W} = \boldsymbol{V}$ であることをかんたんに証明できる

主成分分析

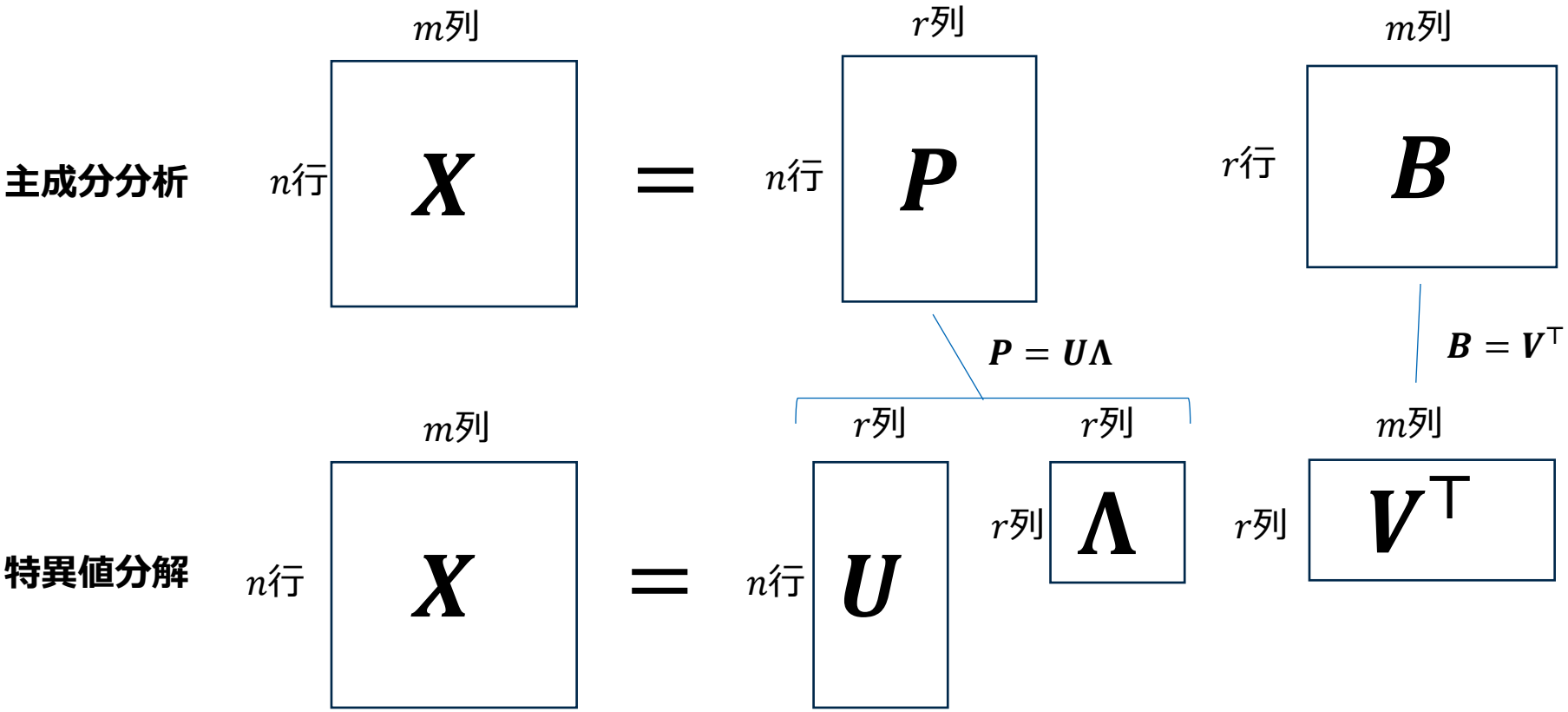


特異値分解

$\boldsymbol{X} = \boldsymbol{U\Lambda V}^{\mathrm{T}}$
$\boldsymbol{V}$は直交行列なので
$\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}$
よって
$\boldsymbol{U\Lambda} = \boldsymbol{XV}$



- 元の変数は、すべての主成分の線形和となる
  - 主成分分析の記号を使うと: 主成分に対する重み行列を$\boldsymbol{B}$として $\boldsymbol{X} = \boldsymbol{PB}$
  - 特異値分解の記号を使うと: $\boldsymbol{X} = \boldsymbol{U\Lambda V}^{\top}$
  - $\boldsymbol{P} = \boldsymbol{U\Lambda}$ より、$\boldsymbol{B} = \boldsymbol{V}^{\top}$



- 元の変数を、第$k(< r)$主成分までの線形和によって近似できる
  - 主成分分析の記号を使うと: 第k主成分までを$\boldsymbol{P}$から取り出して$\tilde{\boldsymbol{P}}$、$\boldsymbol{B}$から取り出して$\tilde{\boldsymbol{B}}$とし、$\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{P}}\tilde{\boldsymbol{B}}$
  - 特異値分解の記号を使うと: 第k列までを$\boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{V}$から取り出して$\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Lambda}}, \tilde{\boldsymbol{V}}$とし、 $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{V}}^{\mathsf{T}}$
  - 特異値分解の性質より、$\tilde{\boldsymbol{X}}$は階数kの行列のなかで$\boldsymbol{X}$のもっともよい近似である

主成分分析

$$\underset{n\text{行}}{\tilde{\boldsymbol{X}}}\ (\underset{m\text{列}}{}) = \underset{n\text{行}}{\tilde{\boldsymbol{P}}}\ (\underset{k\text{列}}{}) \quad \underset{k\text{行}}{\tilde{\boldsymbol{B}}}\ (\underset{m\text{列}}{})$$

特異値分解

$$\underset{n\text{行}}{\tilde{\boldsymbol{X}}}\ (\underset{m\text{列}}{}) = \underset{n\text{行}}{\tilde{\boldsymbol{U}}}\ (\underset{k\text{列}}{}) \quad \underset{k\text{行}}{\tilde{\boldsymbol{\Lambda}}}\ (\underset{k\text{列}}{}) \quad \underset{k\text{行}}{\tilde{\boldsymbol{V}}^{\mathsf{T}}}\ (\underset{m\text{列}}{})$$

(3章のまとめ)

- 特異値分解
  - 行列$\boldsymbol{X}$を、ある特徴を持つ3つの行列の積 $\boldsymbol{U\Lambda V}^{\mathrm{T}}$ に分解すること
  - $\boldsymbol{U}$の行は$\boldsymbol{X}$の行を表し、$\boldsymbol{V}$の行は$\boldsymbol{X}$の列を表している
  - $\boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{V}$から第$k$特異値までを取り出した行列を$\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Lambda}}, \tilde{\boldsymbol{V}}$とすると、 $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{V}}^{\mathrm{T}}$ は $\boldsymbol{X}$ の良い近似となる

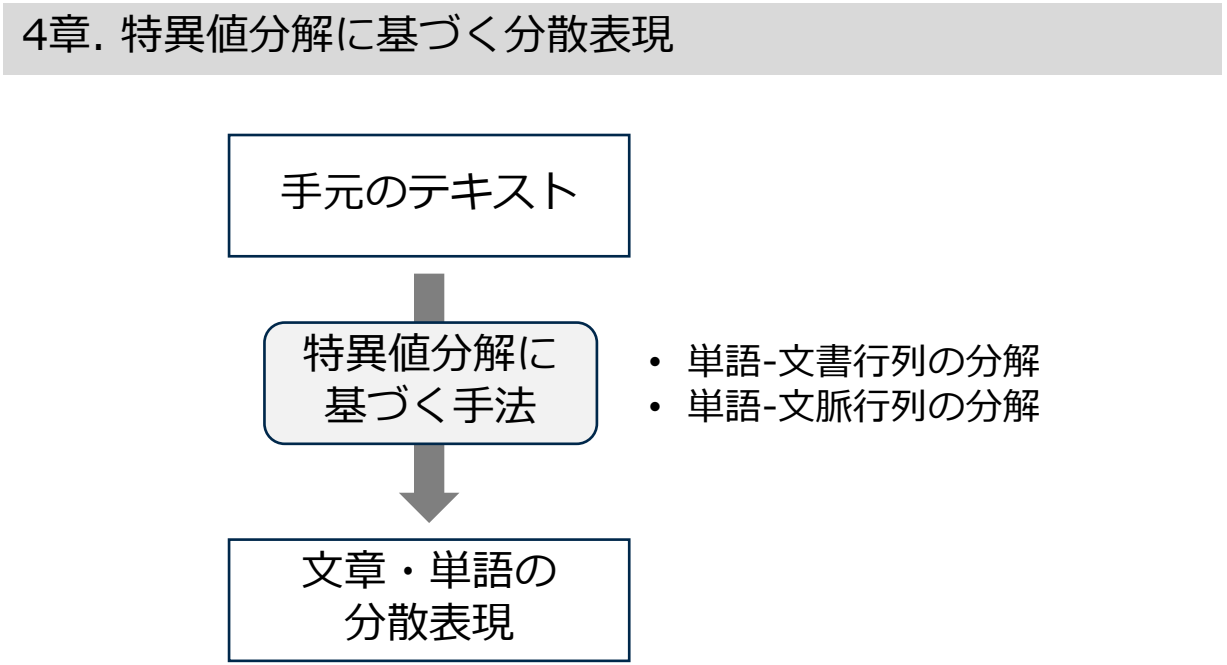
- 主成分分析
  - データ行列$\boldsymbol{X}$の特異値分解
    - ふつう、$\boldsymbol{X}$は中心化済みデータ行列
  - $\boldsymbol{U\Lambda}$を主成分、$\boldsymbol{V}$を負荷と呼ぶ

4. 特異値分解に基づく分散表現

(4章の内容)

- 特異値分解を利用して、手元のテキストから文章・単語の分散表現を得る方法を紹介する

- 次の2つを紹介する
  - 単語-文書行列の特異値分解による分散表現
  - 単語-文脈行列の特異値分解による分散表現



4.1 単語-文書行列の分解による分散表現

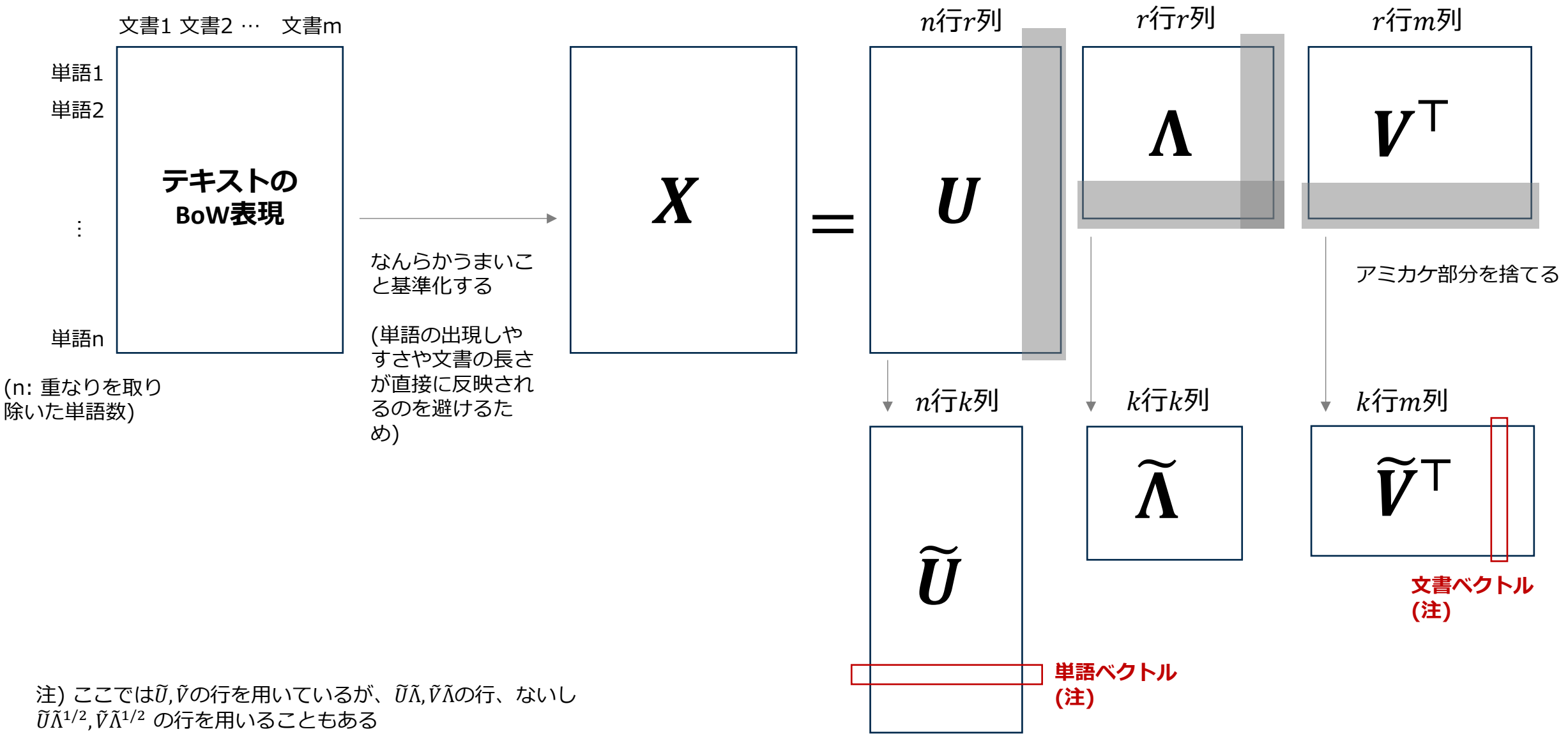
- 分析対象となるテキストが n 個の文書からなるとしよう
  - ここで「文書」は、単一の文でも、複数の文からなる文章でもよい

- 形態素解析によって、各文書に含まれる単語の頻度を数え、BoW表現を得たとしよう
  - (以下では単語を行、文書を列とする。逆でもよい)
  - この行列の各行は、単語を「その単語がどの文書で出てきたか」という観点から表現している
  - この行列の各列は、文書を「その文書にどの単語が出てきたか」という観点から表現している
  - ゼロが非常に多い巨大な行列となり、扱いにくい

	1	2	3	4	5	…	176	177	178	179	180	181	182	184	185	…	287	289	290	291	292
ア	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あきれる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	1	0	0	0
…																					
おじいさん	1	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おと	0	1	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おる	0	1	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おれ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
…																					
嗅ぐ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
樅	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
檜	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0

「ごん狐」「手袋を買いに」に含まれる文(292文)のそれぞれを文書とみなし、BoW表現を得た。単語のうち名詞・動詞・形容詞を抜き出して行列とした。単語数746, 文書数279となった

- それぞれの単語・文書の分散表現を得たい
  - 特異値分解すればいいんじゃない？
    - $\boldsymbol{U}$の行は、$\boldsymbol{X}$の行 (=単語) に対応するベクトル
    - $\boldsymbol{V}$の行は、$\boldsymbol{X}$の列 (=文書) に対応するベクトル
    - 適当な第k列までを抜き出した $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Lambda}}, \tilde{\boldsymbol{V}}$ を使えば、ベクトルも短くなり扱いやすい！



注) ここでは$\tilde{U},\tilde{V}$の行を用いているが、$\tilde{U}\tilde{\Lambda},\tilde{V}\tilde{\Lambda}$の行、ないし$\tilde{U}\tilde{\Lambda}^{1/2},\tilde{V}\tilde{\Lambda}^{1/2}$ の行を用いることもある

- 計算例：「ごん狐」「手袋を買いに」

単語-文書行列 (BoW) (再掲)

	1	2	3	4	5	…	176	177	178	179	180	181	182	184	185	…	287	289	290	291	292
ア	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あきれる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	1	0	0	0
…																					
おじいさん	1	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おと	0	1	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おる	0	1	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おれ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
…																					
嗅ぐ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
樅	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
樫	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0

単語-文書行列 (基準化)

	1	2	3	4	5		176	177	178	179	180	181	182	184	185		287	289	290	291	292
ア	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
あきれる	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	2.45	0	0	0
おじいさん	2.45	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おと	0	2.45	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おる	0	2.14	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
おれ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
嗅ぐ	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
樅	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
樫	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0

TF-IDFによって基準化した。単語 $i\,(=1,\ldots,n)$ の文書 $j(=1,\ldots,m)$ における出現頻度を TF_{ij}, 単語 i が出現する文書の数をDF_i として

$$(TF-IDF)_{ij} = TF_{ij} \times \log\left(\frac{m}{DF_i}\right)$$

n行r列

$$\boldsymbol{U}$$

ア	0.02	0.01	0.03	0.02	0.00	0.00	-0.01	0.07	-0.03	0.09	…
あう	0.04	0.03	0.00	-0.02	-0.01	0.02	0.01	0.01	-0.08	0.03	…
あかり	0.01	0.00	0.00	0.00	-0.01	0.00	0.00	-0.01	-0.01	0.00	…
あがる	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	…
あきれる	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	…
…	…	…	…	…	…	…	…	…	…	…	…
おじいさん	0.00	0.00	0.00	0.00	0.01	0.00	0.00	-0.01	-0.01	0.00	…
おと	0.01	0.02	0.00	0.00	0.00	0.00	0.01	0.00	-0.05	-0.02	…
おもう	0.01	0.00	0.01	0.00	-0.01	0.00	0.00	-0.01	0.00	0.02	…
おる	0.01	0.02	0.00	0.00	0.00	0.00	0.01	0.00	-0.05	-0.02	…
おれ	0.06	0.05	0.09	0.06	0.00	-0.02	-0.05	0.20	-0.14	0.35	…
…	…	…	…	…	…	…	…	…	…	…	…
嗅ぐ	0.01	-0.01	0.00	-0.01	0.01	-0.01	0.01	0.00	0.02	0.00	…
抓る	0.01	-0.01	0.00	-0.01	0.01	-0.01	0.01	0.00	0.02	0.00	…
椛	0.01	0.01	0.00	-0.01	0.01	0.00	0.00	0.00	-0.01	-0.01	…
樗	0.02	-0.01	0.00	0.00	-0.04	0.02	-0.01	0.00	-0.01	0.02	…
眩しい	0.01	0.02	0.01	-0.02	0.01	0.00	0.02	-0.01	-0.03	-0.03	…

r行r列

$$\boldsymbol{\Lambda}$$

14.9	0	0	0	0	0	0	0	0	…
0	12.4	0	0	0	0	0	0	0	…
0	0	11.7	0	0	0	0	0	0	…
0	0	0	11.4	0	0	0	0	0	…
0	0	0	0	11.2	0	0	0	0	…
0	0	0	0	0	10.8	0	0	0	…
0	0	0	0	0	0	10.5	0	0	…
0	0	0	0	0	0	0	10.3	0	…
0	0	0	0	0	0	0	0	10.3	…
0	0	0	0	0	0	0	0	0	10.1	…
…	…	…	…	…	…	…	…	…	…	…

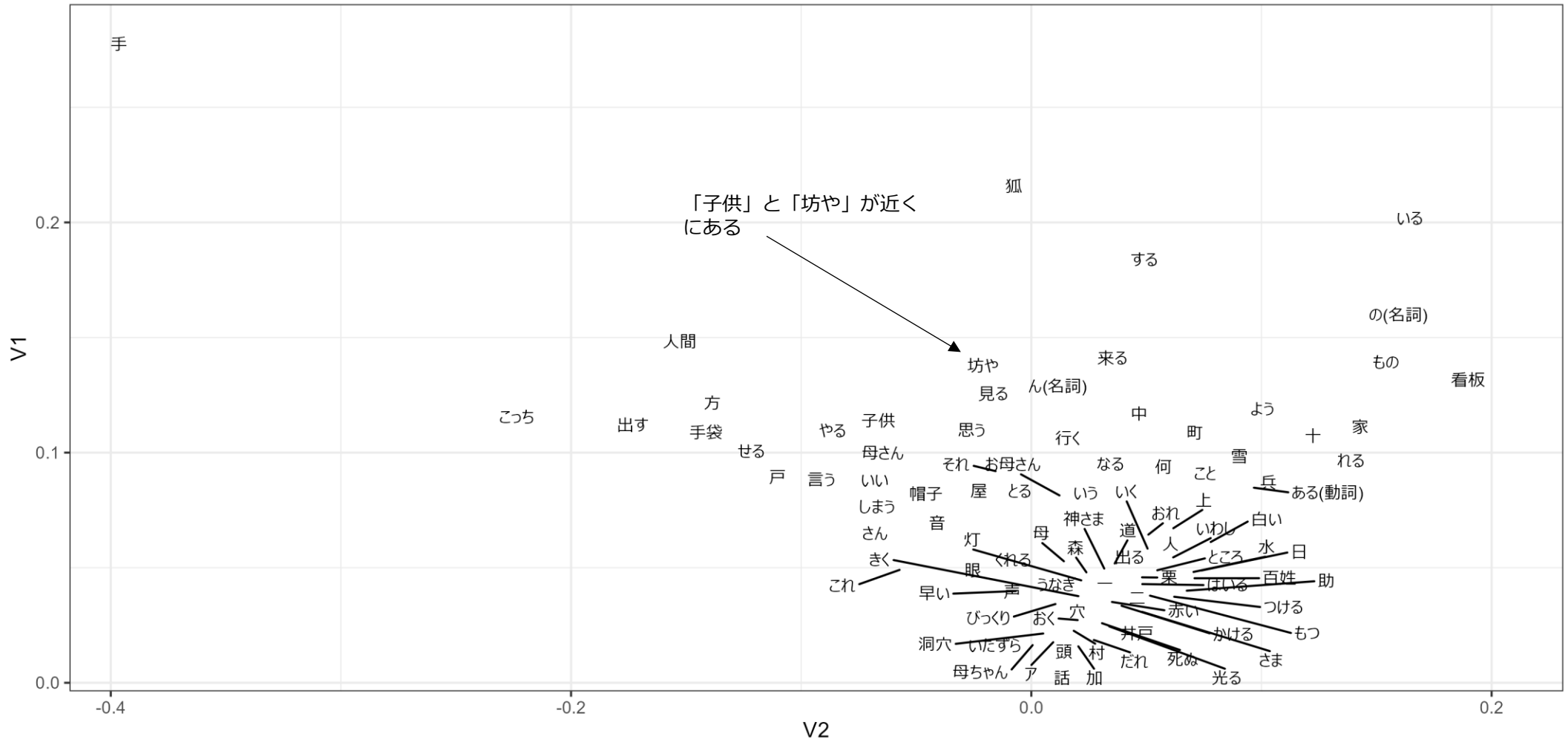
r行m列

$$\boldsymbol{V}^{\top}$$

1	2	3	4	5	…	176	177	178	179	180	181	182	184	185	…	287	289	290	291	292
0.02	0.05	0.01	0.02	0.02	…	0.02	0.02	0.01	0.02	0.02	0.03	0.04	0.04	0.04	…	0.02	0.00	0.03	0.03	0.00
0.01	0.08	0.01	0.00	0.01	…	0.02	0.02	0.01	0.02	0.02	0.02	0.00	0.00	-0.01	…	0.00	0.00	-0.01	-0.01	0.00
0.01	0.01	0.02	0.00	-0.01	…	0.00	0.01	0.02	0.04	-0.02	-0.03	-0.02	-0.01	-0.02	…	-0.01	0.00	-0.03	-0.03	0.00
0.01	0.00	-0.01	0.00	0.00	…	0.00	0.01	0.00	0.01	0.00	-0.07	0.01	0.00	-0.02	…	0.00	0.00	0.01	0.01	0.00
0.03	0.01	-0.01	0.00	0.00	…	0.00	0.00	-0.01	0.02	0.00	0.04	0.02	0.02	-0.01	…	0.00	0.00	-0.06	-0.06	0.00
0.00	0.02	0.01	0.00	0.00	…	0.00	0.00	0.00	-0.01	0.00	-0.01	-0.01	-0.01	0.00	…	-0.01	0.00	0.02	0.02	0.00
0.01	0.05	0.00	0.01	0.02	…	0.01	0.00	-0.01	-0.01	0.00	0.02	0.03	0.04	0.07	…	0.03	0.00	-0.02	-0.02	0.00
-0.03	0.01	-0.02	0.00	-0.01	…	0.01	0.02	0.00	-0.02	-0.02	0.00	-0.01	-0.03	-0.01	…	-0.01	0.00	0.02	0.02	0.00
-0.05	-0.19	-0.02	0.01	0.01	…	-0.02	-0.01	0.00	-0.01	0.00	0.06	0.03	0.03	0.05	…	0.03	0.00	-0.01	-0.01	0.00
-0.01	-0.10	-0.01	0.01	0.00	…	0.01	0.01	0.02	-0.01	-0.01	0.03	-0.01	-0.01	-0.01	…	0.00	0.00	0.02	0.02	0.00
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

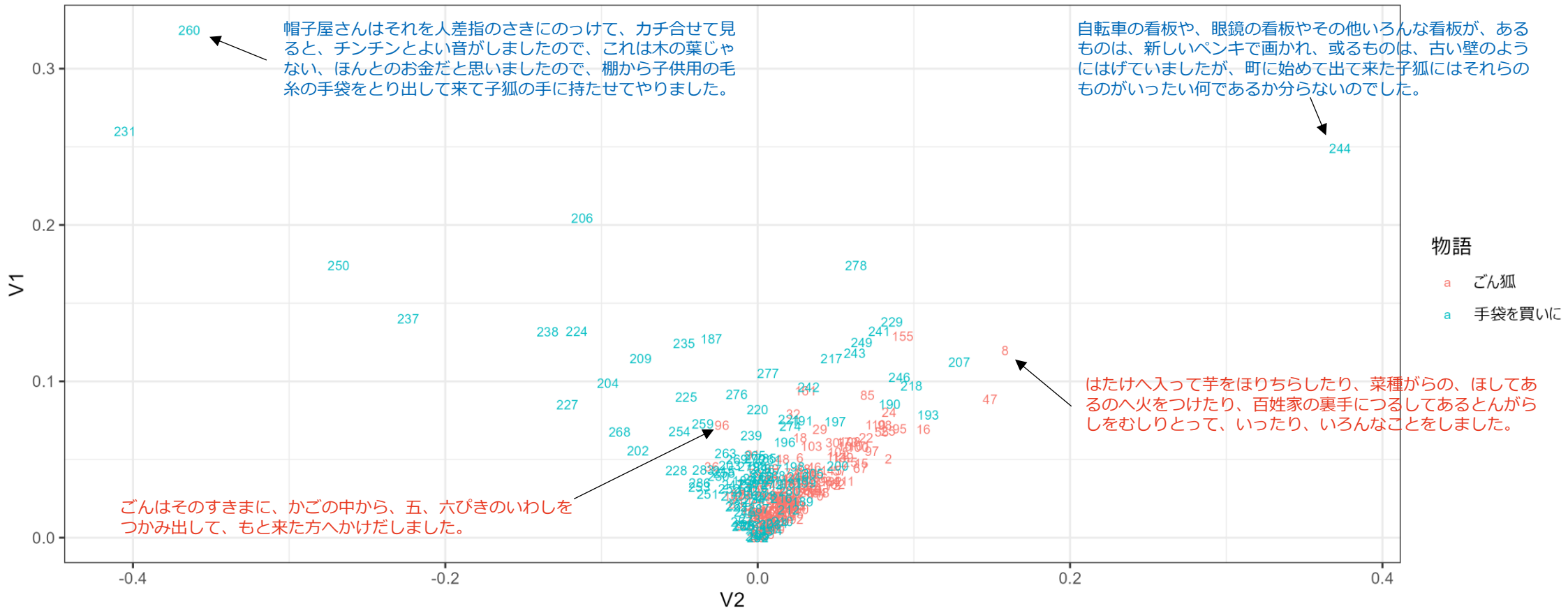
次頁以降で用いる部分を赤枠で示す

- 最初の2次元を用いて、単語ベクトルを図示する



頻度5以上の単語のみ図示

- 最初の2次元を用いて、文ベクトルを図示する
  - 物語の間での単語のちがいが、空間上でもある程度まで表現されている (「ごん狐」は右側)



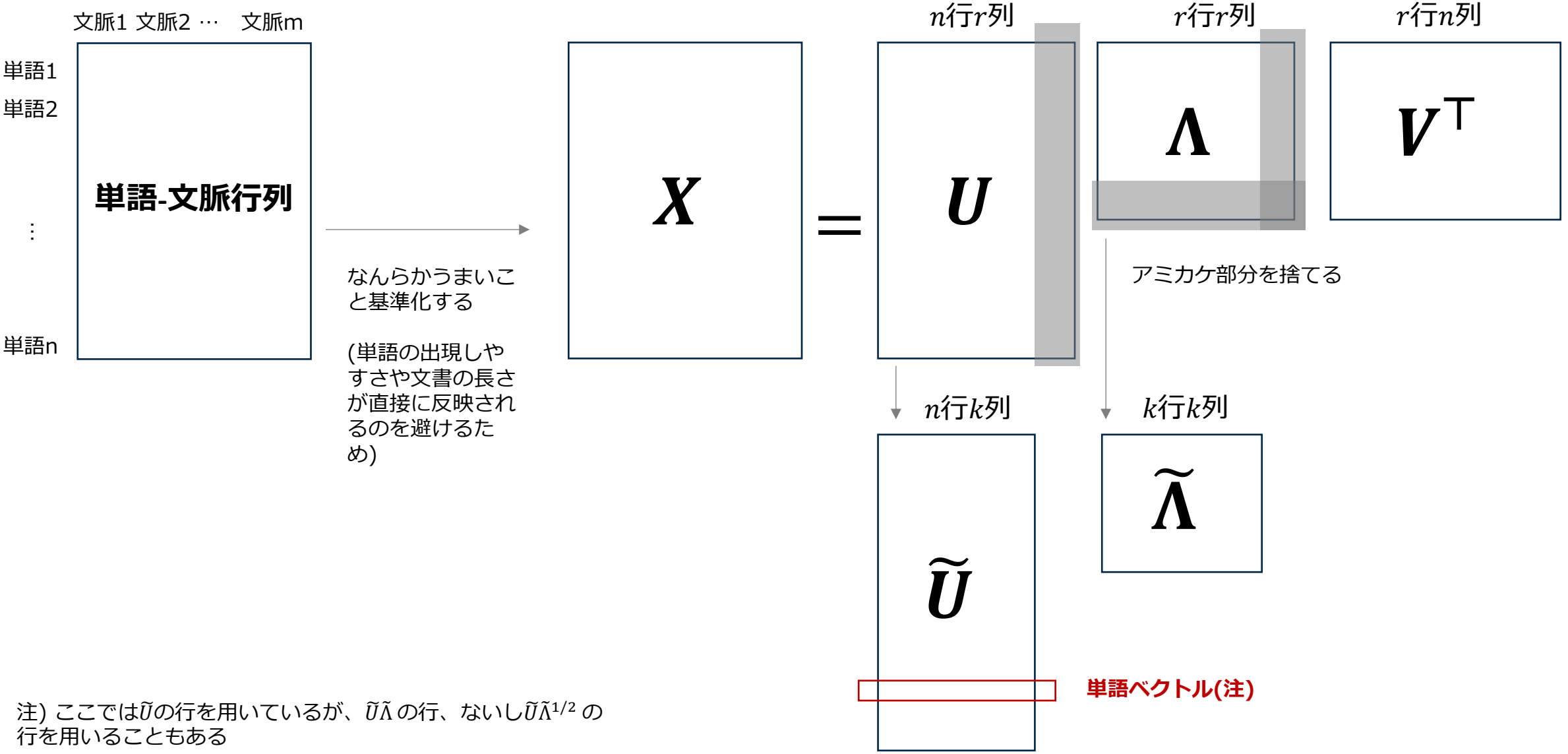
4.2 単語-文脈行列の分解による分散表現 (岡崎ほか(2022), 3.3-3.4節)

- 単語-文脈行列
  - テキストの各単語について、その「文脈」を定義する
    - 例) 単語の前後2個の範囲にある単語を、その単語の「文脈」と呼ぶ
  - 行に単語、列に「文脈」、セルに「すべての文書を通じた、単語と文脈の結びつきの強さ」を格納した行列
    - 例) セルの値は「(行側の)単語の前後2単語以内に、(列側の)文脈単語が含まれた数」とする
  - この行列の各行は、「その単語がどんな文脈で出てきたか」を表している
  - ゼロが非常に多い巨大な行列となり、扱いにくい

	ア	あう	あかり	あがる	あくる日		おじいさん	おと	おもう	おる	おれ		嗅ぐ	抓る	楜	樫	眩しい
ア	0	0	0	0	0		0	0	0	0	1		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あくる日	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おじいさん	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おと	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おれ	1	0	0	0	0		0	0	0	0	0		0	0	0	0	0
嗅ぐ	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
楜	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
樫	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0

「ごん狐」「手袋を買いに」について、ある単語からみて同一の文の中で前後3つの範囲にある単語を、その単語の文脈単語と定義した。名詞・動詞・形容詞に注目し、単語と文脈単語のペアが出現する回数を数えて行列とした

- それぞれの単語の分散表現を得たい
  - 特異値分解すればいいんじゃない？
    - $\boldsymbol{U}$の行は、$\boldsymbol{X}$の行 (=単語) に対応するベクトルとなる
    - 適当な第k特異値までを抜き出した $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Lambda}}$ を使えば、ベクトルも短くなり扱いやすい



注) ここでは$\tilde{U}$の行を用いているが、$\tilde{U}\tilde{\Lambda}$ の行、ないし$\tilde{U}\tilde{\Lambda}^{1/2}$ の行を用いることもある

- 計算例：「ごん狐」「手袋を買いに」

単語-文脈行列 (頻度) (再掲)

	ア	あう	あかり	あがる	あくる日		おじいさん	おと	おもう	おる	おれ		嗅ぐ	抓る	樅	檻	眩しい
ア	0	0	0	0	0		0	0	0	0	1		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あくる日	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おじいさん	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おと	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おれ	1	0	0	0	0		0	0	0	0	0		0	0	0	0	0
嗅ぐ	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
樅	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
檻	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0

単語-文脈行列 (基準化)

	ア	あう	あかり	あがる	あくる日		おじいさん	おと	おもう	おる	おれ		嗅ぐ	抓る	樅	檻	眩しい
ア	0	0	0	0	0		0	0	0	0	3.42		0	0	0	0	0
あう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あかり	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あがる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
あくる日	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おじいさん	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おと	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おもう	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おる	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
おれ	3.42	0	0	0	0		0	0	0	0	0		0	0	0	0	0
嗅ぐ	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
抓る	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
樅	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
檻	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
眩しい	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0

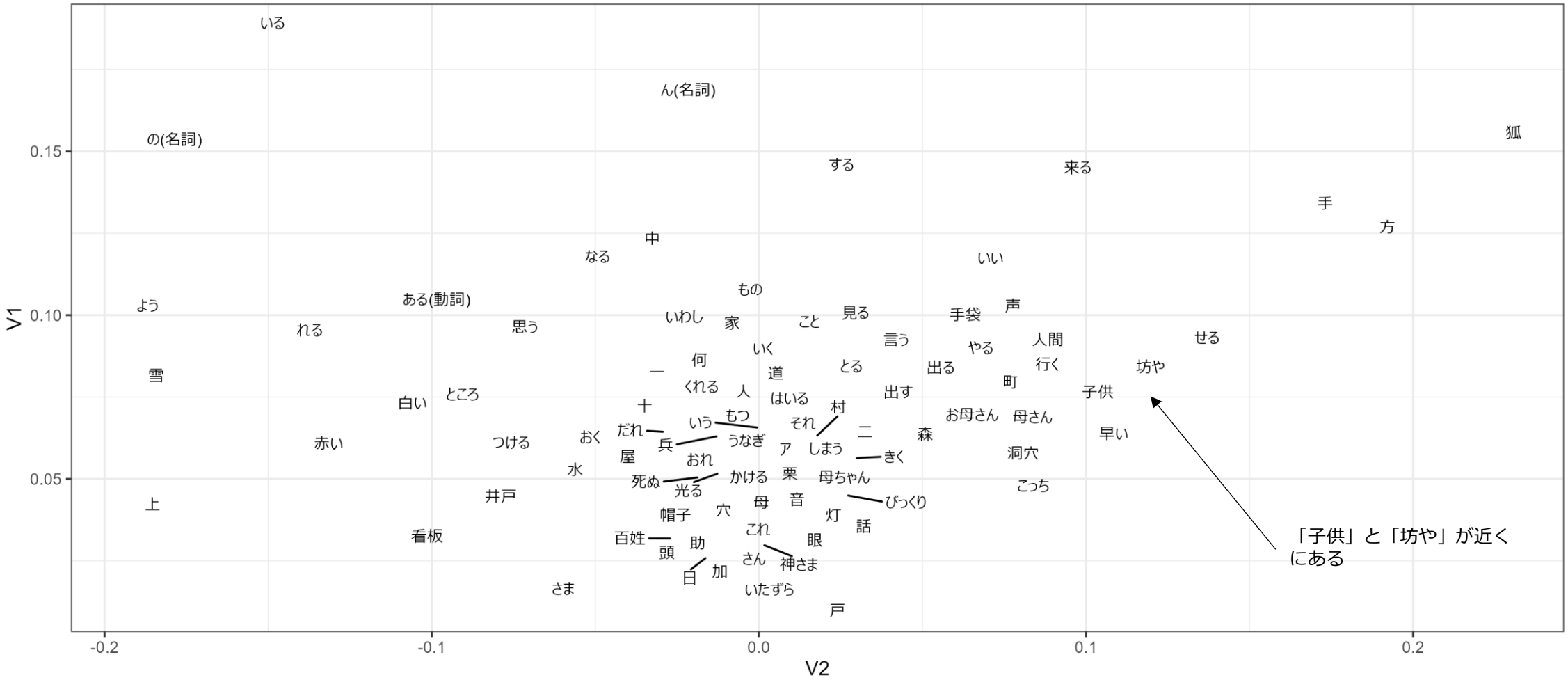
基準化のために自己相互情報量(PMI)を用いた。単語$i(=1,\dots,n)$, 文脈$j(=1,\dots,n)$の共起頻度をx_{ij}として、単語-文脈行列の要素m_{ij}は

$$m_{ij} = \max(0, PMI_{ij})$$

$$PMI_{ij} = \log(Z) + \log(x_{ij}) - \log(x_i) - \log(c_j)$$

$$Z = \sum_i^n \sum_j^n x_{ij}\,, \qquad x_i = \sum_j^n x_{ij}\,, \quad c_j = \sum_i^n x_{ij}$$

- 最初の2次元を用いて、単語ベクトルを図示する



頻度5以上の単語のみ図示

4.3 行列分解による分散表現

- 1990年代以降、広く用いられてきた
  - 単語-文書行列のSVDは **潜在意味解析**(LSA)と呼ばれていた
    - 心理学では、知識表象と獲得を表現するための心理学的モデルとして捉えられた
    - 情報検索の分野では潜在意味インデキシング (LSI)と呼ばれている
  - テキスト分析の一般向け解説書でも紹介されていることが多い

- 悪く言えば…
  - 場当たり的
    - BoW表現をどう基準化するか？ 単語と文脈との関連の強さをどう表すか？ 特異値の数をいくつ採用するか？
    - 明確な判断基準がなく、場当たり的に決めている
  - 生成モデルではない
    - 「ことばがどのように生成されているか」をモデル化しようとしていない
    - 拡張が難しい

Psychological Review
1997, Vol. 104, No. 2, 211–240

Copyright 1997 by the American Psychological Association, Inc.
0033-295X/97/$3.00

A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge

Thomas K Landauer
University of Colorado at Boulder

Susan T. Dumais
Bellcore

How do people know as much as they do with as little information as they get? The problem takes many forms; learning vocabulary from text is an especially dramatic and convenient one for research. A new general theory of acquired similarity and knowledge representation, latent semantic analysis (LSA), is presented and used to successfully simulate such learning and several other psycholinguistic phenomena. By inducing global knowledge indirectly from local co-occurrence data in a large body of representative text, LSA acquired knowledge about the full vocabulary of English at a comparable rate to schoolchildren. LSA uses no prior linguistic or perceptual similarity knowledge; it is based solely on a general mathematical learning method that achieves powerful inductive effects by extracting the right number of dimensions (e.g., 300) to represent objects and contexts. Relations to other theories, phenomena, and problems are sketched.

Prologue

"How much do we know at any time? Much more, or so I believe, than we know we know!"
—Agatha Christie, *The Moving Finger*

A typical American seventh grader knows the meaning of 10–15 words today that she did not know yesterday. She must have acquired most of them as a result of reading because (a) the majority of English words are used only in print, (b) she already knew well almost all the words she would have encountered in speech, and (c) she learned less than one word by direct instruction. Studies of children reading grade-school text find that about one word in every 20 paragraphs goes from wrong to right on a vocabulary test. The typical seventh grader would have read less than 50 paragraphs since yesterday, from which she should have learned less than three new words. Apparently, she mastered the meanings of many words that she did not encounter. Evidence for all these assertions is given in detail later.

This phenomenon offers an ideal case in which to study a problem that has plagued philosophy and science since Plato 24 centuries ago, the fact that people have much more knowledge than appears to be present in the information to which they have been exposed. Plato's solution, of course, was that people must come equipped with most of their knowledge and need only hints and contemplation to complete it.

In this article we suggest a very different hypothesis to explain the mystery of excessive learning. It rests on the simple notion that some domains of knowledge contain vast numbers of weak interrelations that, if properly exploited, can greatly amplify learning by a process of inference. We have discovered that a very simple mechanism of induction, the choice of the correct dimensionality in which to represent similarity between objects and events, can sometimes, in particular in learning about the similarity of the meanings of words, produce sufficient enhancement of knowledge to bridge the gap between the information available in local contiguity and what people know after large amounts of experience.

Overview

In this article we report the results of using latent semantic analysis (LSA), a high-dimensional linear associative model that embodies no human knowledge beyond its general learning mechanism, to analyze a large corpus of natural text and generate a representation that captures the similarity of words and text passages. The model's resulting knowledge was tested with a standard multiple-choice synonym test, and its learning power compared to the rate at which school-aged children improve their performance on similar tests as a result of reading. The model's improvement per paragraph of encountered text approximated the natural rate for schoolchildren, and most of its acquired knowledge was attributable to indirect inference rather than direct co-occurrence relations. This result can be interpreted in at least two ways. The more conservative interpretation is that it shows that with the right analysis a substantial portion of the information needed to answer common vocabulary test questions can be inferred from the contextual statistics of usage alone. This is not a trivial conclusion. As we alluded to earlier

Thomas K Landauer, Institute of Cognitive Science, University of Colorado at Boulder; Susan T. Dumais, Information Science Research Department, Bellcore, Morristown, New Jersey.

We thank Karen Lochbaum for valuable help in analysis; George Furnas for early ideas and inspiration; Peter Foltz, Walter Kintsch, and Ernie Mross for unpublished data; and for helpful comments on the ideas and drafts, we thank, in alphabetic order, Richard Anderson, Doug Carroll, Peter Foltz, George Furnas, Walter Kintsch, Lise Menn, and Lynn Streeter.

Correspondence concerning this article should be addressed to Thomas K Landauer, Campus Box 345, University of Colorado, Boulder, Colorado 80309. Electronic mail may be sent via Internet to landauer@psych.colorado.edu.

Launder & Dumais (1997 *Psychological Review*)
心理学の超一流誌に掲載された長大な論文。
LSAを心理学的モデルとして捉えていた。
そんな時代もあったねと… (by 中島みゆき)

4.4 分散表現の背景

- 分散表現が表している「意味」とは？
  - ＜意味が近い文章・単語は近いベクトルとして表現される＞というとき、その「意味」ってなに？

- 分布意味論 (distributional semantics)
  - 「単語の意味は、それが出現した文脈によって決まる」(分布仮説)という考え方に基づく意味論
    - 1950年代の言語学に源流を持つ
  - データ駆動的テキスト表現の背後にある考え方
    - 現代の自然言語処理の発展の基盤となった
  - 「意味」をテキストの世界の内側で説明しようとしている

- 分散表現は、分布意味論がいう意味での「意味」を表そうとしている

John Ruper Firth (1890-1960)
英国の言語学者。
分布仮説を表す格言 “You shall know a word by the company it keeps” で知られる

図の出典: Scott (1961)

(4章のまとめ)

- テキストにおける単語の頻度を行列で表現し、それを分解することで、単語・文書の分散表現を得ることができる

5. ニューラルネットワーク

(4-5章の内容)

(説明のための準備)

- **特異値分解に基づくテキストの分散表現** (4章) ← **特異値分解** (3章)
  - 手元のテキスト・データに基づき、そこに含まれている単語・文章の分散表現を得て、その後の分析に用いる

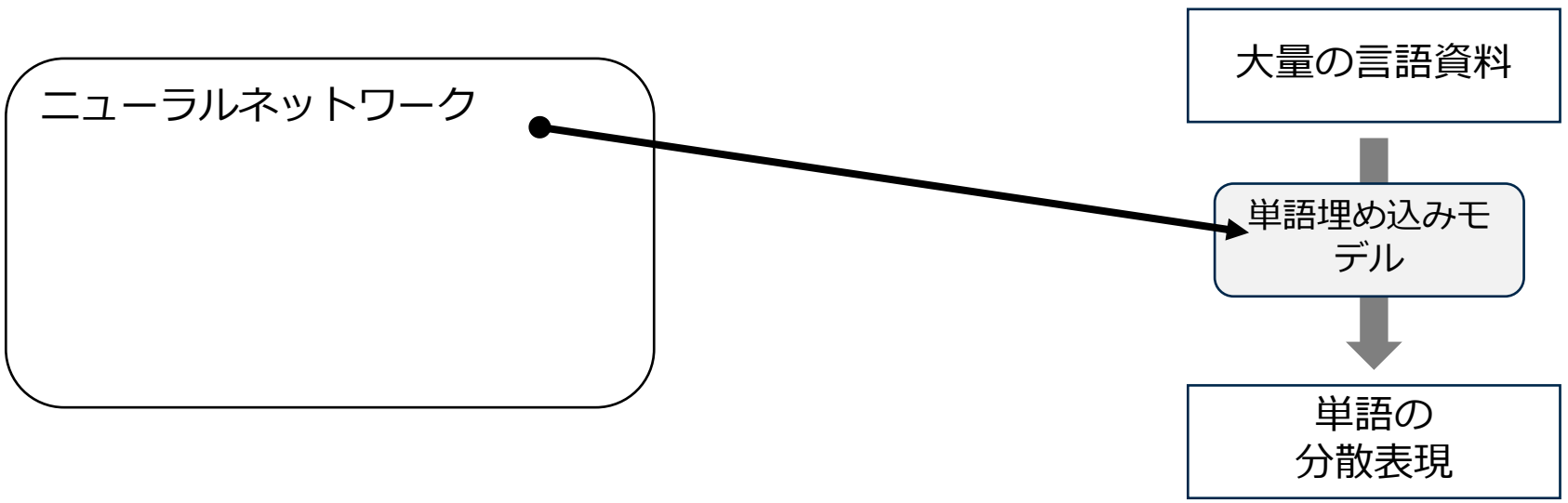
- **ニューラルネットワークに基づく単語の埋め込み表現** (6章) ← **ニューラル・ネットワーク** (5章)
  - 大量の言語資料を用いて、単語の分散表現を得る
  - 手元のテキスト・データについて、そこに含まれている単語を分散表現に換え、その後の分析に用いる

- **ニューラル言語モデルに基づく文章の埋め込み表現** (8章) ← **ニューラル言語モデル** (7章)
  - 大量の言語資料を用いて、文章を分散表現に換えるためのモデルを構築しておく
  - 手元のテキスト・データについて、そこに含まれている文章を分散表現に換え、その後の分析に用いる

- 5章では、6章の準備として…
  - ニューラルネットワークについて紹介する
- 6章では
  - 大量の言語資料を用いて単語の分散表現を得る方法を紹介する

5章. ニューラルネットワーク

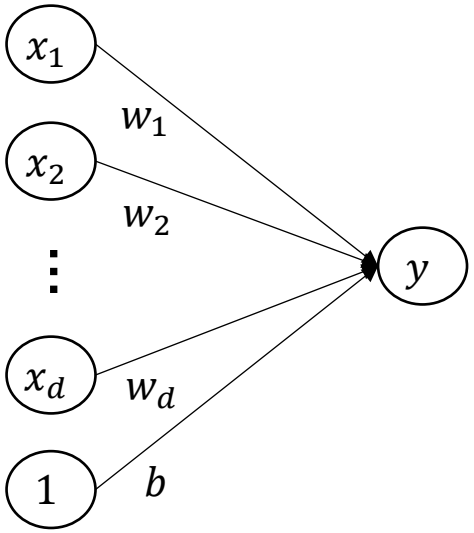
6章. ニューラルネットワークによる単語の分散表現



5.1 ニューラルネットワーク

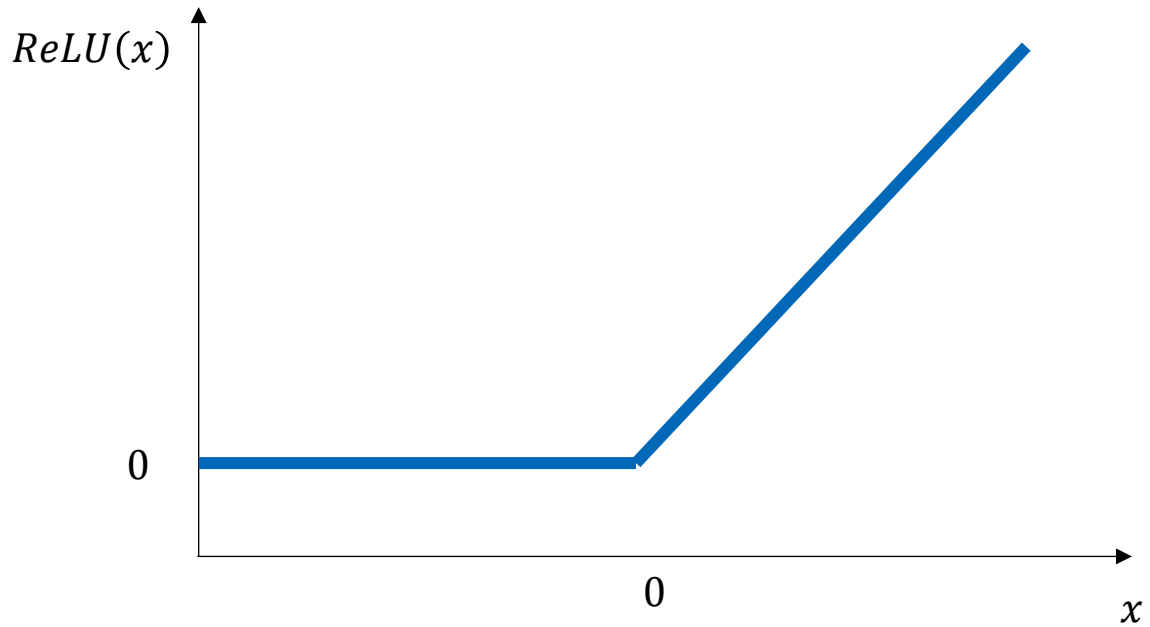
- ニューラルネットワークとは (岡崎ほか(2022) 2.9節)
  - 「ニューロン」と呼ばれる要素をたくさん並べて作った数理モデル
  - ニューロン
    - 複数の実数値を受け取り、重みづけて足し上げ、なんらかの関数で変換して、ひとつの実数値を返す
      - 生物のニューロン (神経細胞) を模している
    - 入力を$\boldsymbol{x} = (x_1, \ldots, x_d)$, 出力を y として

$$y = g\left(\sum_i^d w_i x_i + b\right) = g(\boldsymbol{w}^\top \boldsymbol{x} + b) \longleftarrow \boldsymbol{w} = (w_1, \ldots, w_d)$$



- 活性化関数
  - ニューロンが値を変換するために使う関数
  - いろいろな関数が用いられている
  - 代表例: 正規化線形関数 (ReLU)

$$ReLU(x) = \max(0, x)$$



- とても簡単な例: 2層のニューラルネットワーク
  - 入力層、第1層(隠れ層)、第2層(出力層)からなるニューラルネットワーク
  - 隠れ層 の i 番目の要素 h_i は、入力層の要素数をd_0として

$$h_i = g\left(\sum_j^{d_0} w_{ij}x_j + b_i\right) = g(\boldsymbol{w_i}^\top \boldsymbol{x} + b_i)$$

  - 隠れ層のベクトルを$\boldsymbol{h} = (h_1, \ldots, h_{d_1})$として

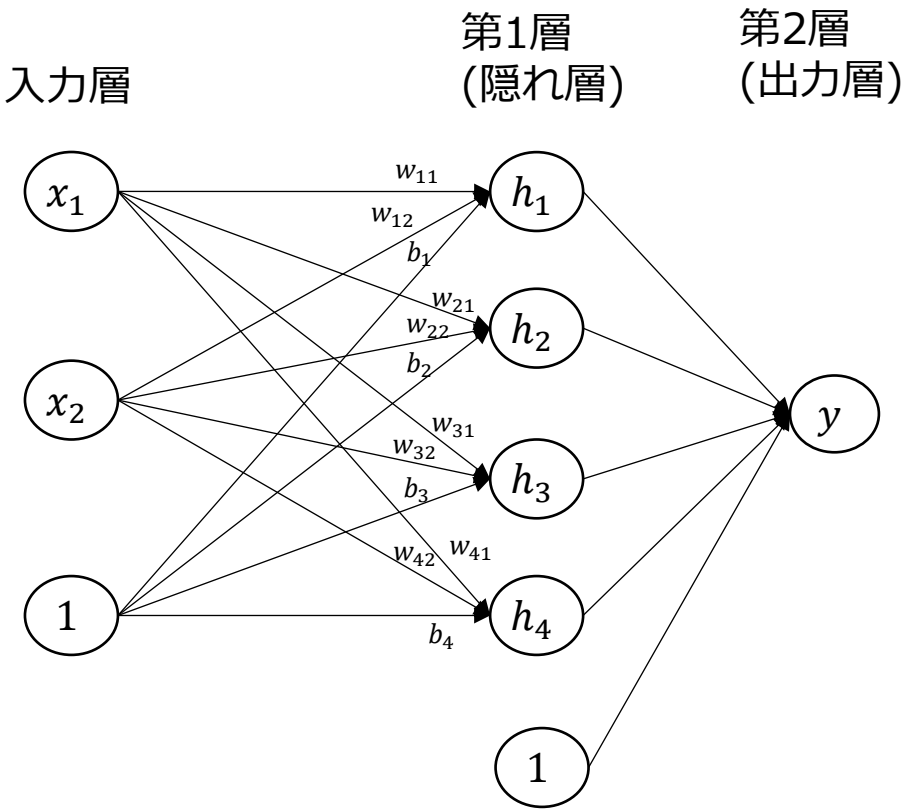
$$\boldsymbol{h} = g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

    - 例. 入力層の要素数2, 隠れ層の要素数4のとき

$$\boldsymbol{h} \quad = g(\quad \boldsymbol{W} \quad \boldsymbol{x} \quad + \quad \boldsymbol{b} \quad)$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = g\left(\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}\right)$$

    - ここで、$\boldsymbol{W}$は入力層から隠れ層への重みを表す行列
      - 行が隠れ層の要素、列が入力層の要素を表している

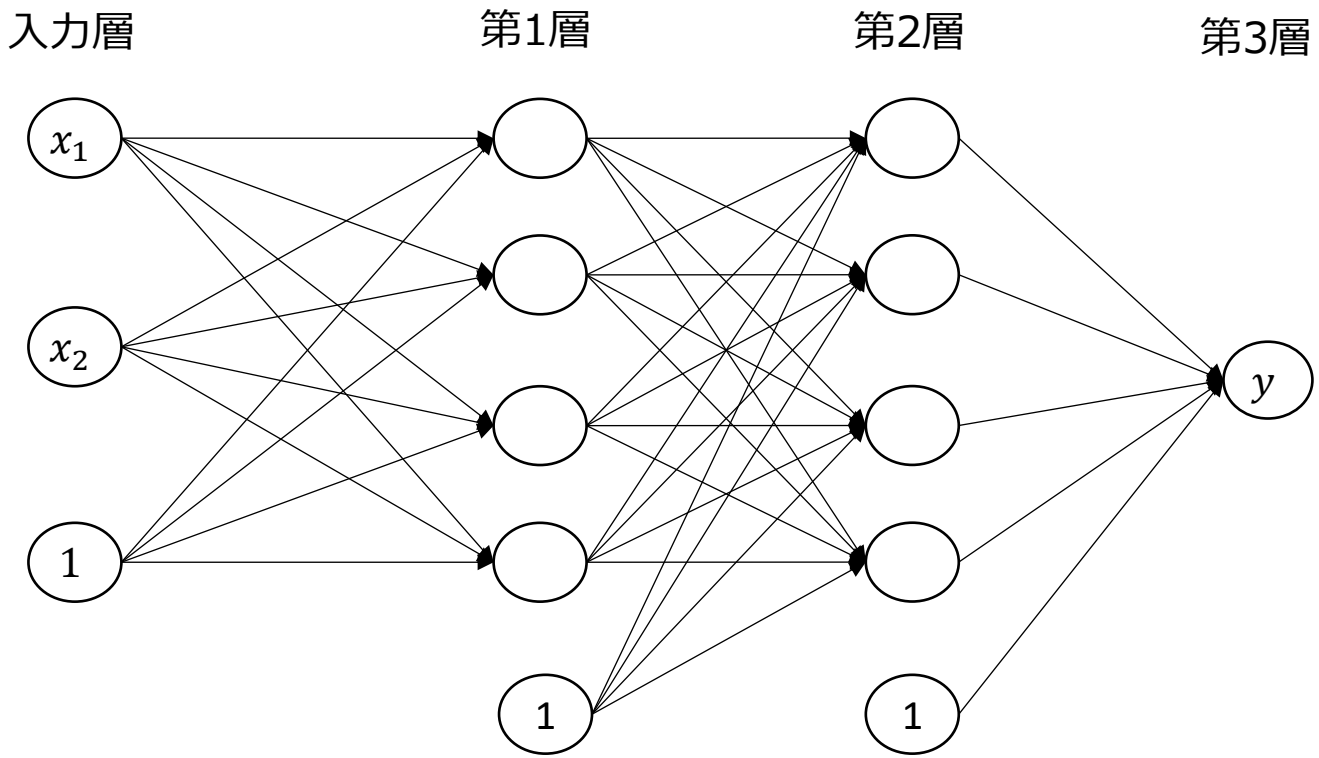


- パラメータ
  - 前頁では $\boldsymbol{W}$ と $\boldsymbol{b}$
  - データに基づいて学習する
    - 出力と学習データのずれを表す、損失関数を決めておく
    - 損失関数の値が最小になるような重みを決める
  - 本日は、パラメータを学習する方法については一切省略します

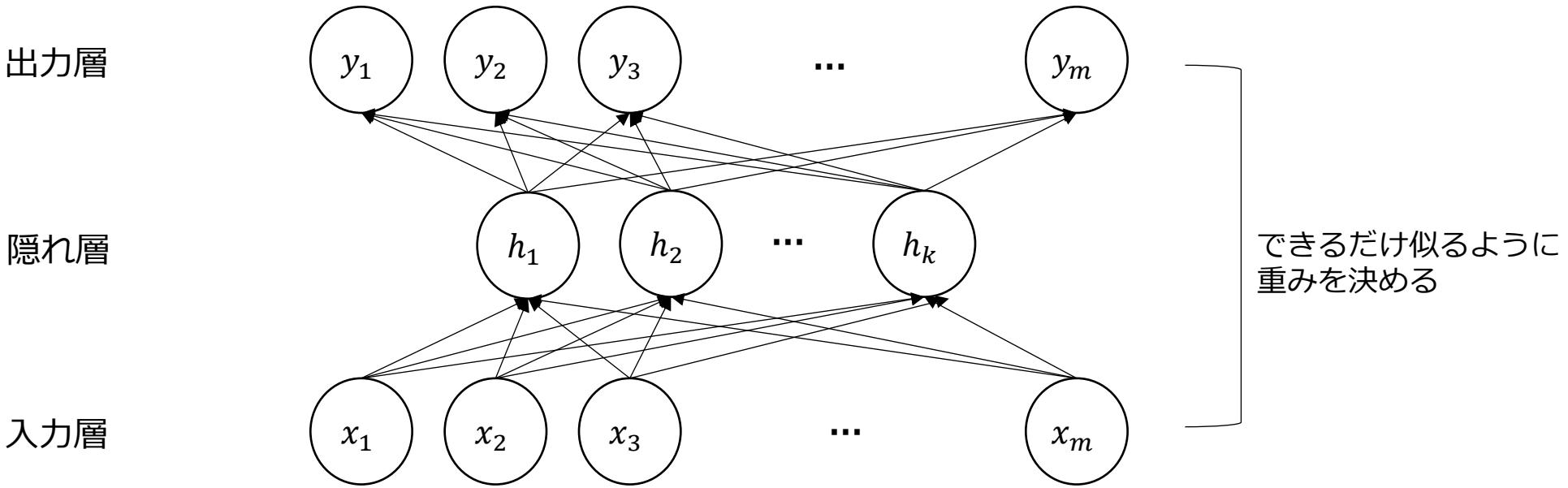
- アーキテクチャの設計
  - ほとんどの場合、ニューラルネットワークは層の連鎖構造として設計される
  - 活性化関数のタイプ、層の数、各層における要素数を、あらかじめ決める必要がある
  - 層の数を増やすと、データにおける複雑な関係を表現できるが、パラメータの学習は難しくなる

5.2 さまざまなニューラルネットワーク

- 順伝播型ニューラルネットワーク (feedforward neural network)
  - 第1層, 第2層, ..., という順に、入力から出力に向かって順に計算を行うネットワーク



- オートエンコーダー (autoencoder)
  - 出力が入力の近似になるように学習する順伝播型ニューラルネットワーク
  - 出力自体には関心がない。ある入力に対応する隠れ層の値や、新しいデータに対するあてはまりの良さに関心がある
  - 非線形な主成分分析と捉えることができる



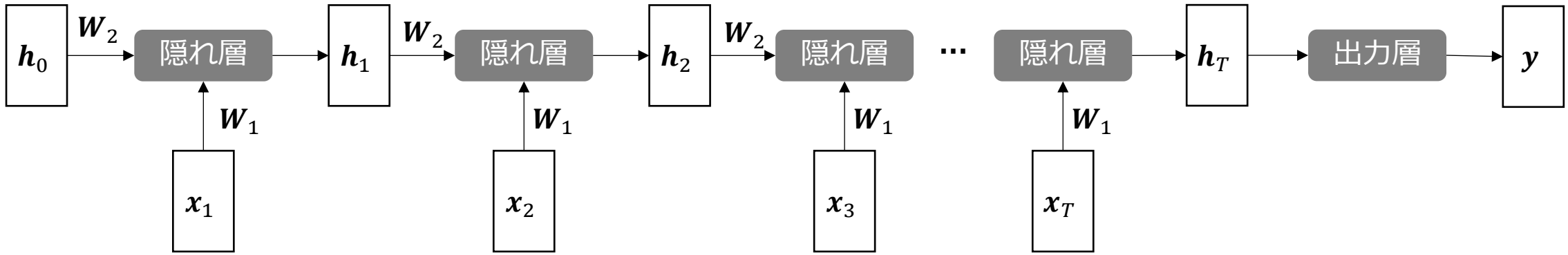
二層のオートエンコーダー

- 再帰型ニューラルネットワーク (recurrent neural network; RNN) (岡崎ほか(2022) 4.2節)
  - 入力としてベクトルの系列を逐次的に受け取り、隠れ状態を変えていくネットワーク
    - 同じニューラルネットワーク(入力層と隠れ層)を、パラメータを変えずに繰り返し用いる
  - もっともシンプルな例：エルマン・ネットワーク
    - t番目に受け取る入力ベクトルを$\boldsymbol{x}_t = (x_{t1}, x_{t2}, \ldots, x_{tm})$ とする
    - $\boldsymbol{x}_t$を受け取る直前の隠れ層の状態を表すベクトル(隠れ状態ベクトル) を$\boldsymbol{h}_{t-1} = (h_{t-1,1}, h_{t-1,2}, \ldots, h_{t-1,d})$とする

$$\boldsymbol{h}_t = g(\boldsymbol{W}_1 \boldsymbol{x}_t + \boldsymbol{W}_2 \boldsymbol{h}_{t-1} + \boldsymbol{b})$$

$\boldsymbol{W}_1$は入力層から隠れ層への重み
$\boldsymbol{W}_2$は直前の隠れ状態ベクトルから隠れ層への重み

    - 入力系列が終わったら、出力層の値を求め、出力する
    - 入力系列の長さがTならば、$T+1$ 層の順伝播型ニューラルネットワークとなる



(5章のまとめ)

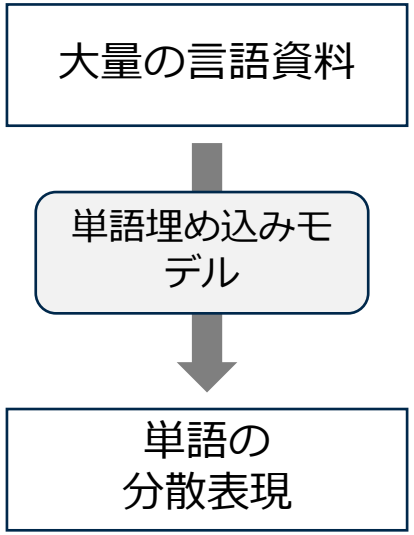
- ニューラルネットワークとは
- 順伝播型ニューラルネットワークとは
- 再帰型ニューラルネットワークとは

6. ニューラルネットワークによる単語の分散表現

(6章の内容)

- 大量の言語資料を用いて単語の分散表現を得る方法を紹介する
  - その代表例である word2vec に注目する

6章. ニューラルネットワークによる単語の分散表現



- word2vec

6.1 word2vec

- word2vecとは (岡崎ほか(2022), 3.5節)
  - 大量の言語資料を学習し、単語の分散表現を得る方法
    - 2013年、Googleの研究者らによって発表された
    - 自然言語処理の世界に大きな影響を与えた
  - 分布仮説に基づく
    - 単語の意味は、その単語の周辺に現れる単語によって決まる
  - 次の2つのモデルがある
    - Continuous Bag-of-Words (CBoW)モデル
    - Skip-Gramモデル
  - 以下では、CBoWモデルを紹介する

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

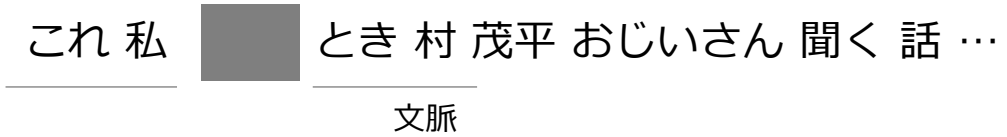
1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

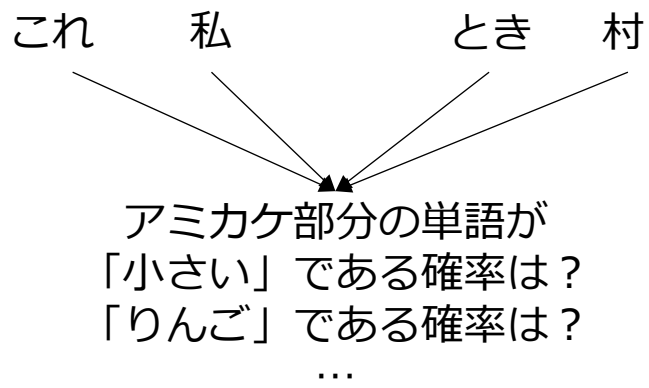
1

Mikolov, Chen, Corrado, & Dean (2013 *Proc. Workshop at ICLR*)
Google Scholarによれば被引用件数51139件。まじか

- CBoWモデルの発想
  - ある位置の単語の前にあるω個の単語と、後にあるω個の単語を、その位置の単語の「文脈」と呼ぶ



  - ある単語を、その文脈から予測する
    - 「文脈と似ている単語が出現しやすい」と想定する



- 表記
  - 位置 t にある単語を、**ワンホットベクトル $\boldsymbol{x_t}$** で表す
    - 語彙サイズの長さを持つベクトル。1か所だけ1, あとは全部0

位置	単語	ワンホットベクトル
1	これ	$x_1 = (0, 0, \dots, 0, 0, 0, \dots, 0, 1, 0, \dots 0, 0, 0, \dots, 0, 0, 0, \dots, 0, 0)$
2	は	$x_2 = (0, 0, \dots, 0, 0, 0, \dots, 0, 0, 0, \dots 0, 1, 0, \dots, 0, 0, 0, \dots, 0, 0)$
3	私	$x_3 = (0, 0, \dots, 0, 0, 0, \dots, 0, 0, 0, \dots 0, 0, 0, \dots, 0, 1, 0, \dots, 0, 0)$
4	が	$x_4 = (0, 0, \dots, 0, 1, 0, \dots, 0, 0, 0, \dots 0, 0, 0, \dots, 0, 0, 0, \dots, 0, 0)$
…		

- CBoWモデル
  - 位置 t の単語 $\boldsymbol{x}_t$ の隠れ層ベクトル を得る
    - ただし、活性化関数は用いない
    - 従って、隠れ層ベクトル$\boldsymbol{w}_{x_t}$はパラメータを $\boldsymbol{W}, \boldsymbol{b}$ として $\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b}$となる。以下、$\boldsymbol{b}$は略記する

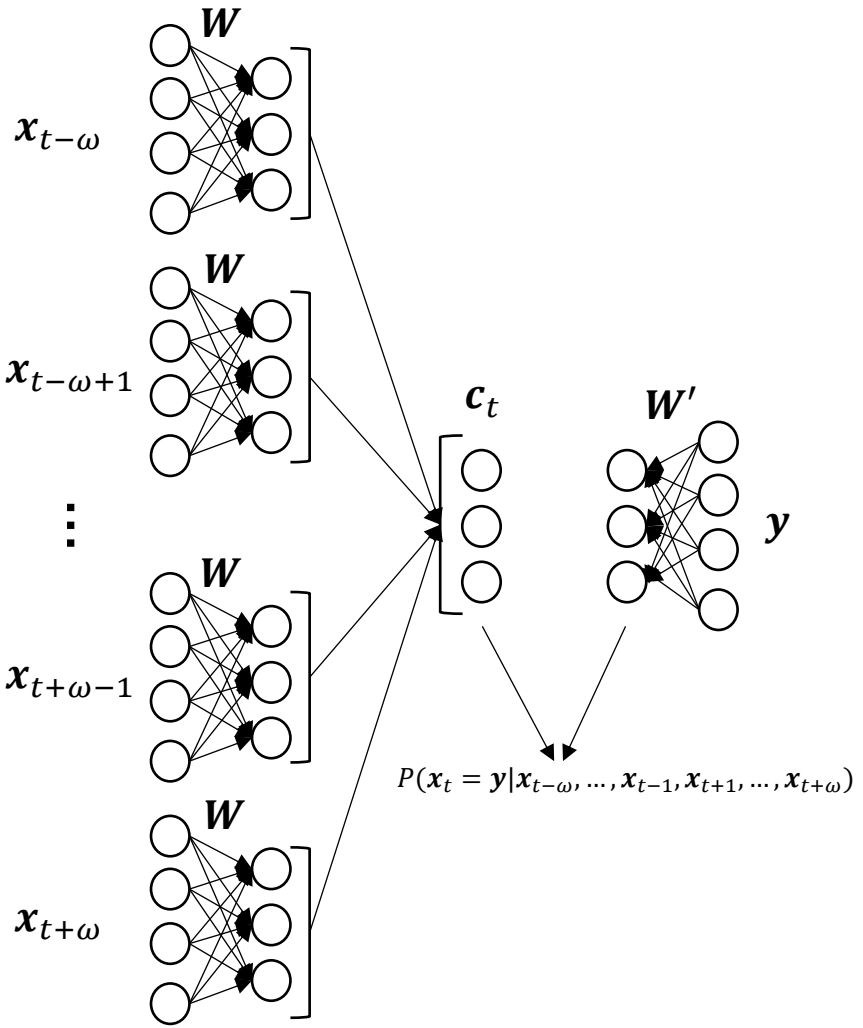
$$\boldsymbol{w}_{x_t} = \boldsymbol{W}\boldsymbol{x}_t$$

  - 位置 t の文脈$(\boldsymbol{x}_{t-\omega}, \dots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_{t+\omega})$について、隠れ層ベクトルを合計し $\boldsymbol{c}_t$ を得る
  - 語彙集合に含まれる語彙 $\boldsymbol{y}$ について隠れ層ベクトル $\boldsymbol{w}'_{\mathrm{y}}$ を得る

$$\boldsymbol{w}'_y = \boldsymbol{W}'\boldsymbol{y}$$

  - 位置 t において任意の語彙 $\boldsymbol{y}$ が出現する確率を求める
    - 語彙集合を$\mathbb{V}$として

$$P(\boldsymbol{x}_t = \boldsymbol{y} | \boldsymbol{x}_{t-\omega}, \dots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_{t+\omega}) = \frac{\exp(\boldsymbol{c}_t^{\top} \boldsymbol{w}'_{\mathrm{y}})}{\sum_{\boldsymbol{v} \in \mathbb{V}} \exp(\boldsymbol{c}_t^{\top} \boldsymbol{w}'_v)}$$



$\boldsymbol{c}_t^{\top} \boldsymbol{w}'_{\mathrm{y}}$を下限0, 合計1となるように調整している。これを**ソフトマックス関数**と呼ぶ。以下では $\mathrm{softmax}(\boldsymbol{c}_t^{\top} \boldsymbol{w}'_y)$ と略記する

6.2 単語埋め込みベクトル

- word2vecが提供するもの
  - 重み行列 $\boldsymbol{W}$
    - 列ベクトルは、その単語に対応する隠れ層ベクトル

  - この隠れ層ベクトルのことを**単語埋め込み** (word embedding) ベクトルという
    - ニューラルネットワークの隠れ層に、単語の意味が「埋め込まれて」いるというイメージ

  - 現代の自然言語処理では、「分散表現」と「埋め込み」はほぼ同じ意味で用いられている
    - たいていの場合、単語・文章の分散表現はニューラルネットワークにおける埋め込みとして得られているから

- 単語埋め込みベクトルの入手方法
  - 手元のテキストを学習して得ることもできるけれど…
  - 大量の言語資料を学習して得られた単語埋め込みベクトルをありがたく使わせていただく
    - データとして公開されている (右図)

3.1 日本語学習済み word2vec まとめ

上記モデルを含め、公開されているword2vecについて、個人的に見つけた結果を以下にまとめました。

Name	Model	Data	Dim	Tokenizer	Dict
WikiEntVec	Skip-gram?	Wikipedia	100,200,300	mecab	mecab-ipadic-NEologd
白ヤギ	CBOW?	Wikipedia	50	mecab	mecab-ipadic-NEologd
chiVe	Skip-gram	NWJC	300	Sudachi	
bizreach	Skip-gram	求人データ	100, 200	mecab	ipadic
dependency-based-japanese-word-embeddings	Dependency-Based Word Embeddings	Wikipedia	100, 200, 300	Ginza	
hottoSNS-w2v (※要問い合わせ)	CBOW	ブログ, Twitter	200	Juman, mecab	mecab-ipadic-NEologd
朝日新聞単語ベクトル (※要問い合わせ)	Skip-gram, CBOW, Glove	朝日新聞	300	mecab	ipadic
fastText	CBOW	Common Crawl, Wikipedia	300	mecab	?
wikipedia2vec	Skip-gram	Wikipedia	100, 300	mecab	?
wordvectors	Skip-gram?, fastText	Wikipedia	300	mecab	?

学習方法 (Skip-gram or CBOW) について、READMEなどのドキュメントに明記されておらず、学習コードのパラメーターから判断したモデルに関しては？をつけています。 (具体的には gensim.models.word2vec.Word2Vecのパラメータ `sg` で判断しています)

図の出典: 白井(2020)

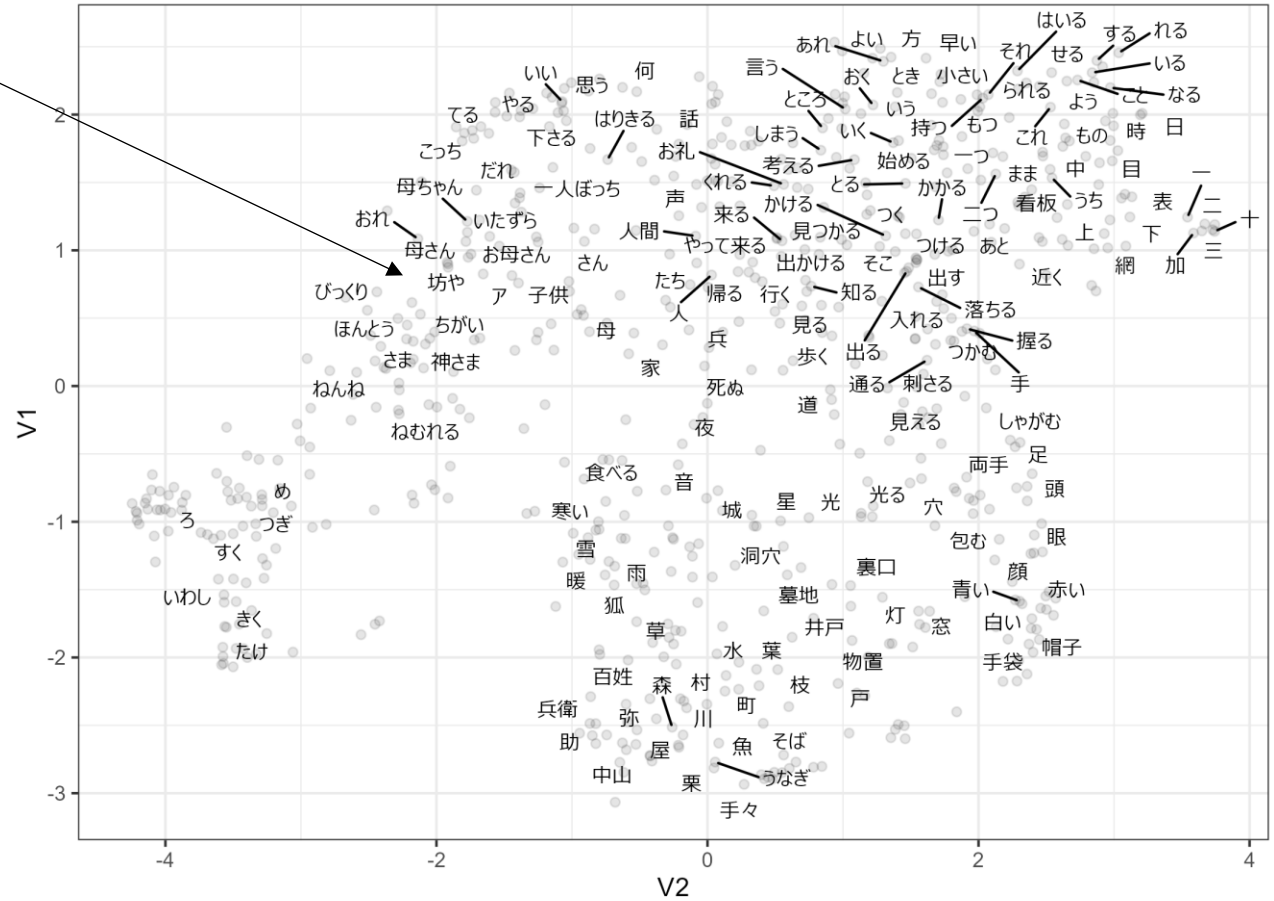
6.3 分析例

- 計算例：「ごん狐」「手袋を買いに」
  - 出現する単語について、単語埋め込みベクトルを得た
    - Wikipedia Entity Vectors (https://github.com/singletongue/WikiEntVec/) を利用させていただいた
    - 説明用データに含まれている名詞・動詞・形容詞のうち688語について、100次元のベクトルを得た

	1	2	3	4	5 …	97	98	99	100
あう	0.17	0.10	0.07	-0.43	-0.08	0.09	0.07	-0.05	0.03
あかり	0.51	0.08	0.24	-0.11	0.08	0.34	-0.12	-0.71	-0.19
あがる	-0.16	-0.11	-0.15	-0.24	-0.33	-0.08	-0.14	0.06	-0.21
あきれる	0.38	-0.47	-0.70	-0.79	-0.24	-0.22	0.47	0.25	-0.26
あくる日	0.19	-0.39	-0.18	-0.09	-0.24	0.05	0.64	0.37	-0.19
…									
おじいさん	0.12	0.06	-0.49	-0.31	-0.36	0.00	0.22	0.00	-0.44
おと	0.32	-0.05	-0.22	-0.30	-0.01	0.34	-0.05	-0.19	-0.20
おもう	0.28	-0.67	-0.67	-0.51	-0.10	0.42	0.26	-0.08	-0.02
おる	0.59	-0.30	-0.68	-0.31	-0.19	-0.01	-0.02	-0.14	-0.12
おれ	0.29	-0.13	-0.72	-0.35	-0.30	-0.01	-0.08	0.04	-0.22
…									
鳴る	-0.01	-0.05	-0.19	-0.01	-0.14	0.19	0.03	0.44	0.24
麦	0.11	-0.18	-0.09	-0.07	-0.31	-0.04	0.12	0.26	0.22
黄	0.00	0.35	-0.18	-0.04	0.11	-0.32	0.40	0.51	-0.05
黒い	0.08	0.49	-0.19	-0.17	-0.01	0.14	0.36	0.21	-0.05
黒子	0.26	0.22	0.24	-0.27	0.26	0.33	0.10	0.04	-0.20

- 得られた単語ベクトルを、UMAP によって二次元空間にマッピングした
  - 関連した意味を持つ単語は、ベクトルも似ている … ような気がする

「子供」と「坊や」が近くにある



頻度3以上の単語のみラベルを表示

UMAP: 非線形的な次元縮約手法のひとつ。上の例では、近いベクトルが近い位置に表示されている

6.4 単語埋め込みの性質と課題

- 単語埋め込みとアナロジー (岡崎ほか(2022), 3.6節)
  - アナロジー課題
    - 例) フランスにおけるパリは、イタリアにおいてなにか？
  - word2vecで得た単語埋め込みベクトルの足し算・引き算で、アナロジーが解ける
    - 次の性質が成り立っている
      - (パリの埋め込みベクトル) – (フランスの埋め込みベクトル)
        ≒ (ローマの埋め込みベクトル) – (イタリアの埋め込みベクトル)
  - → 単語埋め込みが爆発的に普及するきっかけになった

- word2vecと行列分解
  - word2vec は活性化関数を使っていない
  - おおまかにみれば、単語-文脈行列を$\boldsymbol{W}$と$\boldsymbol{W}'$に分解しているだけ
  - 実際、単語-文脈行列の特異値分解で単語の分散表現を得る方法でも、工夫すれば word2vecと類似した結果を得ることができる (Levy & Goldberg, 2014)

- 意味の類似性と関連性
  - word2vecで得られた単語埋め込みベクトルは、「文脈による単語の予測」という観点から構築されている
  - 人間の観点から見ると、単語埋め込みベクトル間の類似性は次の2つの側面を表している
    - 単語の意味の類似性
      - 例) 「緑茶」と「煎茶」
    - 文脈から見た、単語の関連性
      - 例) 「コーヒー」と「カップ」
      - 例) 「明るい」と「暗い」 (意味は正反対！)

- 新語
  - 単語列を入力とするモデル (例, word2vec) は、コーパスに含まれていなかった語を扱えない
  - → テキストを単語よりも細かい単位 (サブワード, 文字, バイト列) に分けて入力するモデルも用いられている

- 多義語・曖昧性
  - word2vec モデルの学習が終わると、単語から埋め込みベクトルへの変換は固定的になり、文脈は無視される
  - 単語をその文脈に応じて、異なる埋め込みベクトルに変換するには？
  - → 「文脈化された単語埋め込みモデル」も用いられている

(6章のまとめ)

- 単語埋め込みとは
- word2vecとは

7. ニューラル言語モデル

(7-8章の内容)

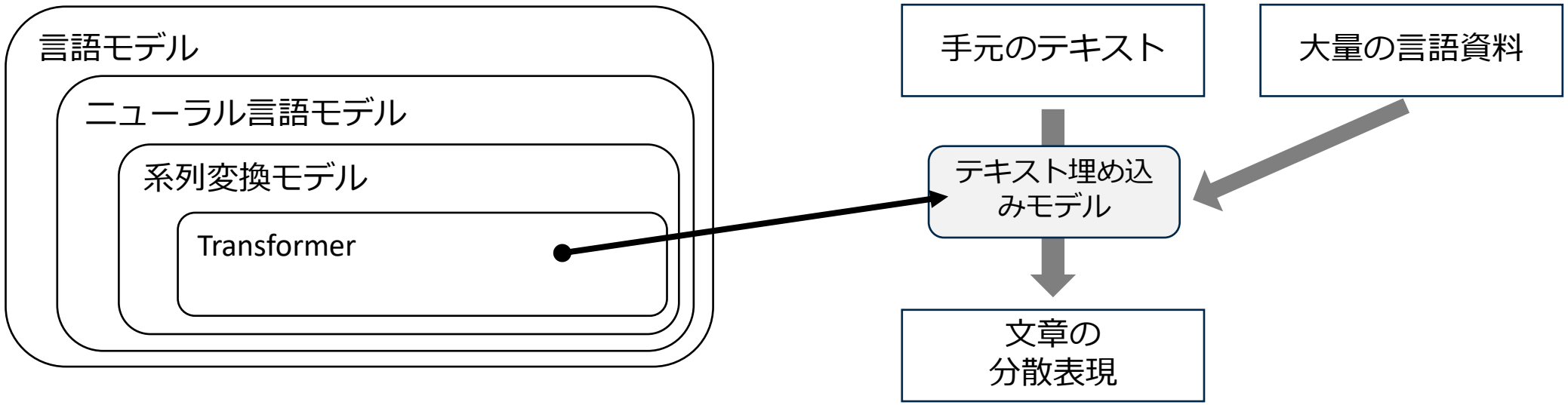
(説明のための準備)

- **特異値分解に基づくテキストの分散表現** (4章) ← **特異値分解** (3章)
  - 手元のテキスト・データに基づき、そこに含まれている単語・文章の分散表現を得て、その後の分析に用いる
- **ニューラルネットワークに基づく単語の埋め込み表現** (6章) ← **ニューラル・ネットワーク** (5章)
  - 大量の言語資料を用いて、単語の分散表現を得る
  - 手元のテキスト・データについて、そこに含まれている単語を分散表現に換え、その後の分析に用いる
- **ニューラル言語モデルに基づく文章の埋め込み表現** (8章) ← **ニューラル言語モデル** (7章)
  - 大量の言語資料を用いて、文章を分散表現に換えるためのモデルを構築しておく
  - 手元のテキスト・データについて、そこに含まれている文章を分散表現に換え、その後の分析に用いる

- 7章では、8章の準備として…
  - ニューラル言語モデルについて紹介する
    - 特に、Transformer アーキテクチャにおける**自己注意機構**というアイデアに焦点を当てる
    - 自己注意機構の説明に必要な内容に注目し、他の内容はできる限り省略する
- 8章では
  - 大量の言語資料を用いて、文章を分散表現に換えるためのモデル(テキスト埋め込みモデル)を構築する方法を紹介する

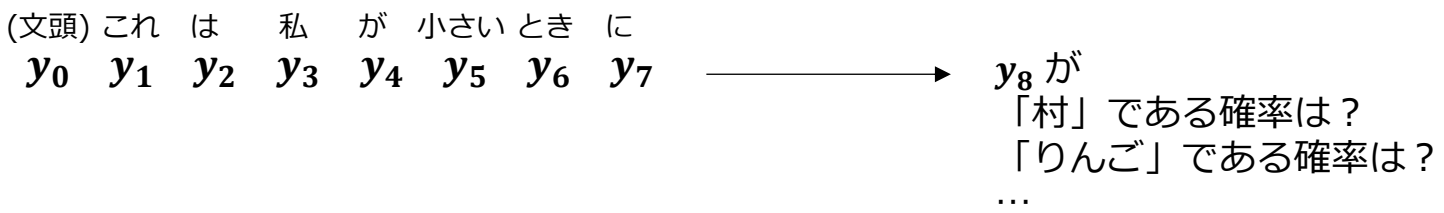
7章. ニューラル言語モデル

8章. ニューラル言語モデルによる文章の分散表現



7.1 言語モデル

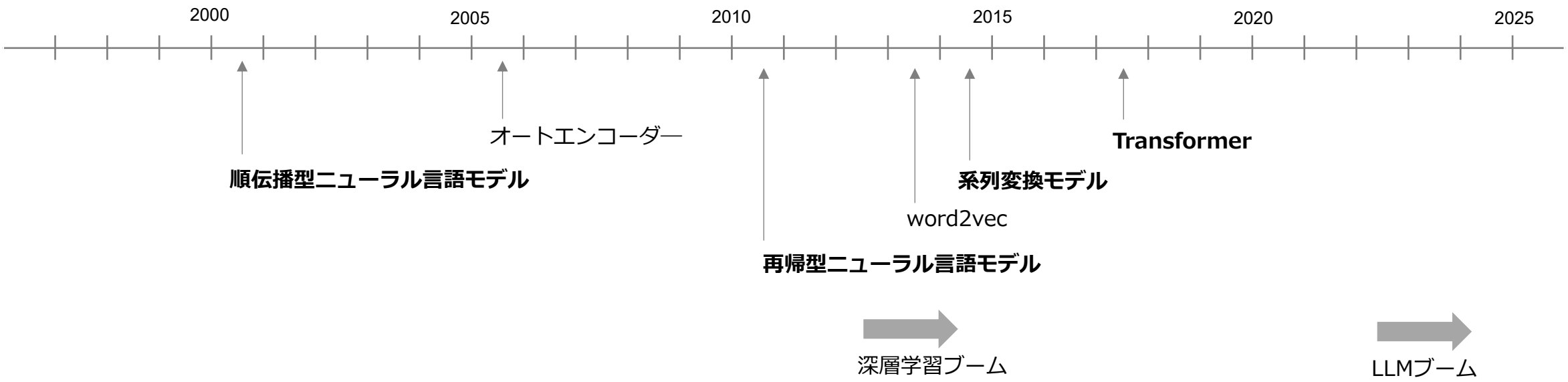
- 言語モデルとは (岡崎ほか(2022) 5.1-5.2節)
  - 文章の$t-1$番目までの単語に基づき、t番目の単語を予測するモデル
  - t 番目の単語が、ベクトル $\boldsymbol{y}_t$ で表現できているとしよう
    - 典型的にはワンホットベクトル
    - 文頭を表すベクトルを $\boldsymbol{y}_0$ とする
  - 入力:
    - $t-1$番目までの単語を表しているベクトル の系列 $\boldsymbol{Y}_{0:t-1} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{t-1})$
  - 出力:
    - t番目の単語の条件付き確率分布。すなわち、任意の語彙$\boldsymbol{y}$について $P(\boldsymbol{y}_t = \boldsymbol{y} \mid \boldsymbol{Y}_{0:t-1})$



7.2 ニューラル言語モデル

- ニューラル言語モデルとは (岡崎ほか(2022) 5.5節)
  - ニューラルネットワークを用いた言語モデル
  - 現代の言語モデルの主流

- ニューラル言語モデルの歴史
  - 岡崎 (2023) より抜粋



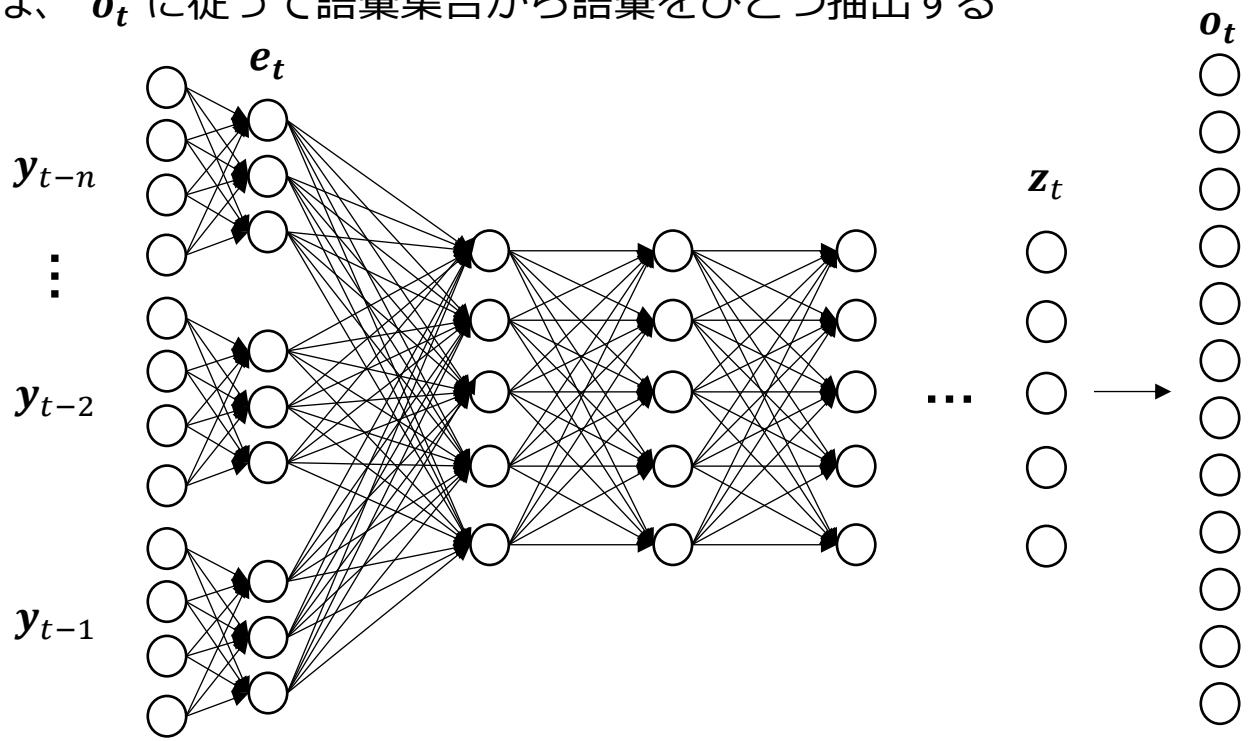
7.3 テキストのトークン化

- トークン化
  - テキストをモデルに入力するために、最小単位 (トークン) へと分割すること
    - 形態素解析はトークン化の方法のひとつとみなせる
  - ニューラル言語モデルでは、トークンとして単語よりも細かい単位が用いられることが多い
    - サブワード, 文字, バイト列...

- 本章では
  - 話をわかりやすくするため、トークンを単語とみなして説明します

7.4 順伝播型ニューラル言語モデル

- 順伝播型ニューラルネットワークに基づく言語モデル
  - 直前のn個の単語$\boldsymbol{y}_{t-n}, \ldots, \boldsymbol{y}_{t-1}$ を、それぞれ埋め込みベクトルに変換し、縦に並べて $\boldsymbol{e}_t$ とする
  - $\boldsymbol{e}_t$を入力する順伝播型ニューラルネットワークをつくる。その出力層ベクトルを$\boldsymbol{z}_t$とする
  - 確率ベクトル $\boldsymbol{o}_t$ を求める
    - i番目の要素は、語彙集合のi番目の語彙が出現する確率を表す。全要素の和は1になる
    - $\boldsymbol{W}, \boldsymbol{b}$ をパラメータとし、ソフトマックス関数を使って $\boldsymbol{o}_t = \mathrm{softmax}(\boldsymbol{W}\boldsymbol{z}_t + \boldsymbol{b})$
- $\boldsymbol{y}_t$を生成する場合は、 $\boldsymbol{o}_t$ に従って語彙集合から語彙をひとつ抽出する

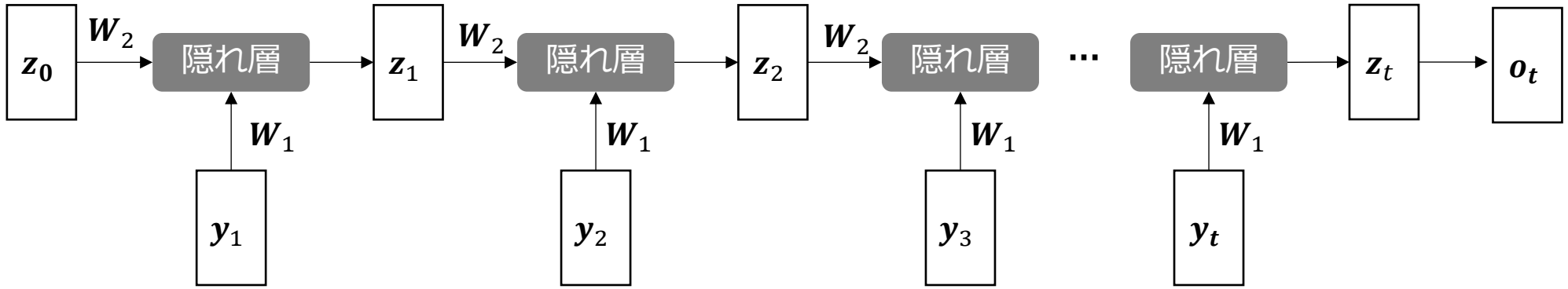


7.5 再帰型ニューラル言語モデル

- 再帰型ニューラルネットワークに基づく言語モデル
  - 隠れ状態ベクトル $\boldsymbol{z}_{t-1}$ と前の単語 $\boldsymbol{y}_{t-1}$を受け取り、更新した隠れ状態ベクトル $\boldsymbol{z}_t$を出力するニューラルネットワークをつくり、これを繰り返し使う

$$\boldsymbol{z}_t = g(\boldsymbol{W}_1\boldsymbol{y}_{t-1} + \boldsymbol{W}_2\boldsymbol{z}_{t-1} + \boldsymbol{b})$$

    - 原理的には、文章の先頭から直前までのすべての単語を合成している
    - 実際には、 $\boldsymbol{z}_t$は直近の単語をより強く反映する
  - 隠れ状態ベクトル$\boldsymbol{z}_t$を確率ベクトル $\boldsymbol{o}_t$ に変換する (順伝播型ニューラルネットワークと同じ)
- $\boldsymbol{y}_t$を生成する場合は、 $\boldsymbol{o}_t$ に従って語彙集合から語彙をひとつ抽出する



7.6 系列変換モデル

- さまざまな自然言語処理課題は、系列から系列への変換とみなすことができる

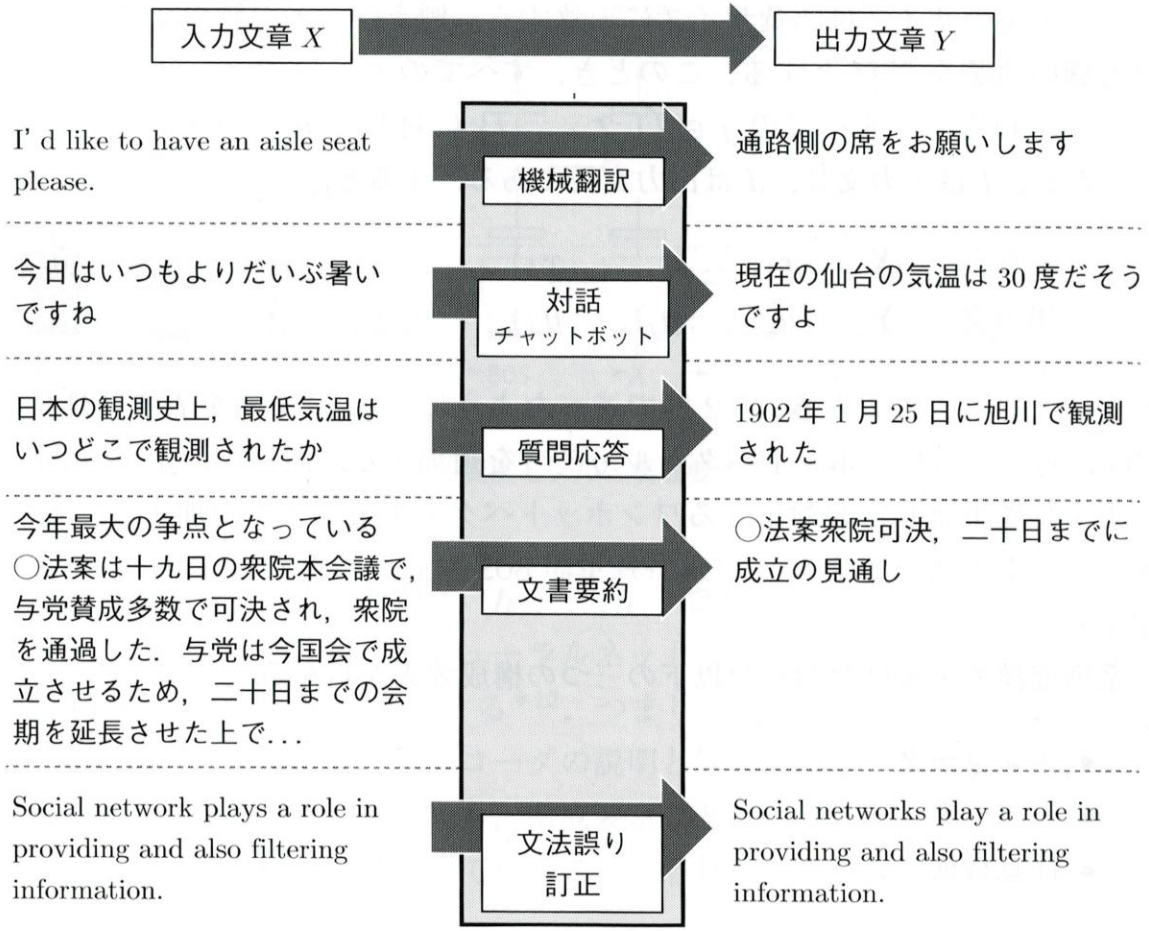
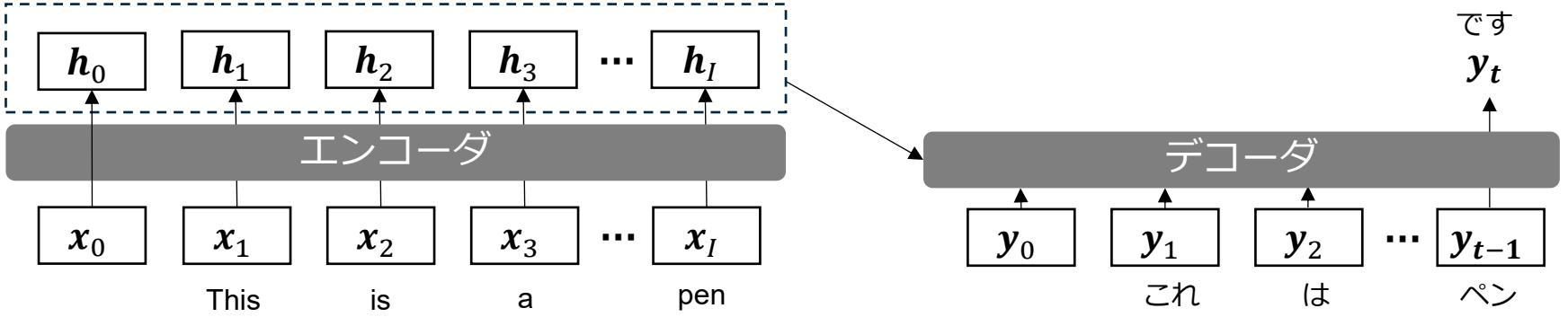


図 5.8　系列変換モデルを適用できるタスクの例

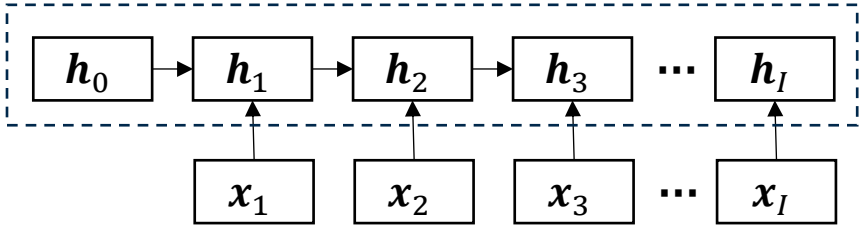
図の出典: 岡崎ほか (2022)

- 系列変換モデルとは (岡崎ほか(2022) 5.6節)
  - 系列から系列への変換を行うタイプの言語モデル
- 表記
  - 入力系列 $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$
  - 出力系列 $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_J)$
  - $\boldsymbol{x}_i, \boldsymbol{y}_j$は単語を表すワンホットベクトル、$\boldsymbol{x}_0, \boldsymbol{y}_0$は文頭を表すワンホットベクトルとする
- エンコーダ
  - $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$ を受けとり、位置 i における隠れ状態ベクトル$\boldsymbol{h}_i$を出力する
- デコーダ
  - $\boldsymbol{H} = (\boldsymbol{h}_0, \ldots, \boldsymbol{h}_I)$ と、それまでに出力した単語列 $\boldsymbol{Y}_{0:j-1} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{j-1})$ を受け取り、位置 j の単語 $\boldsymbol{y}_j$ を生成する

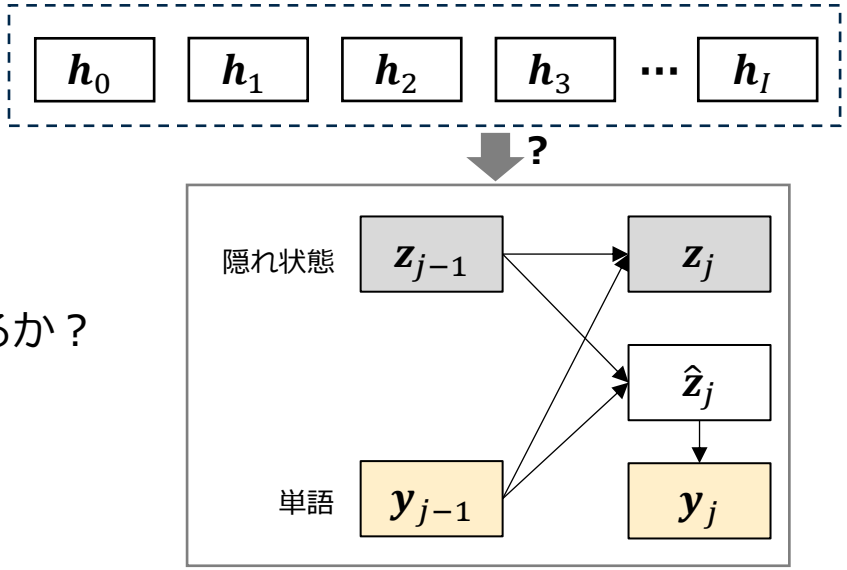


- 再帰型ニューラルネットワークを用いる場合 …

- エンコーダの仕組み
  - $\boldsymbol{h}_{i-1}$を$\boldsymbol{x}_i$ の埋め込みベクトルによって更新し、 $\boldsymbol{h}_i$とする



- デコーダの仕組み
  - 隠れ状態ベクトル $\boldsymbol{z}_{j-1}$ を、 $\boldsymbol{y}_{j-1}$を使ってどうにかして更新し$\boldsymbol{z}_j$とする
  - また、どうにかして単語生成のためのベクトル$\hat{\boldsymbol{z}}_j$をつくる
  - $\hat{\boldsymbol{z}}_j$ を確率ベクトル $\boldsymbol{o}_t$ に変換し、$\boldsymbol{y}_t$を生成する
    - 順伝播型ニューラル言語モデルと同じ
  - → エンコーダが作成した$\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_I)$を、デコーダはどのように利用するか？



- **注意機構** (attention mechanism)
  - エンコーダが作成した$\boldsymbol{H}$を、デコーダが利用する方法のひとつ
  - 現状ではこれ一択だそうです

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

* CIFAR Senior Fellow

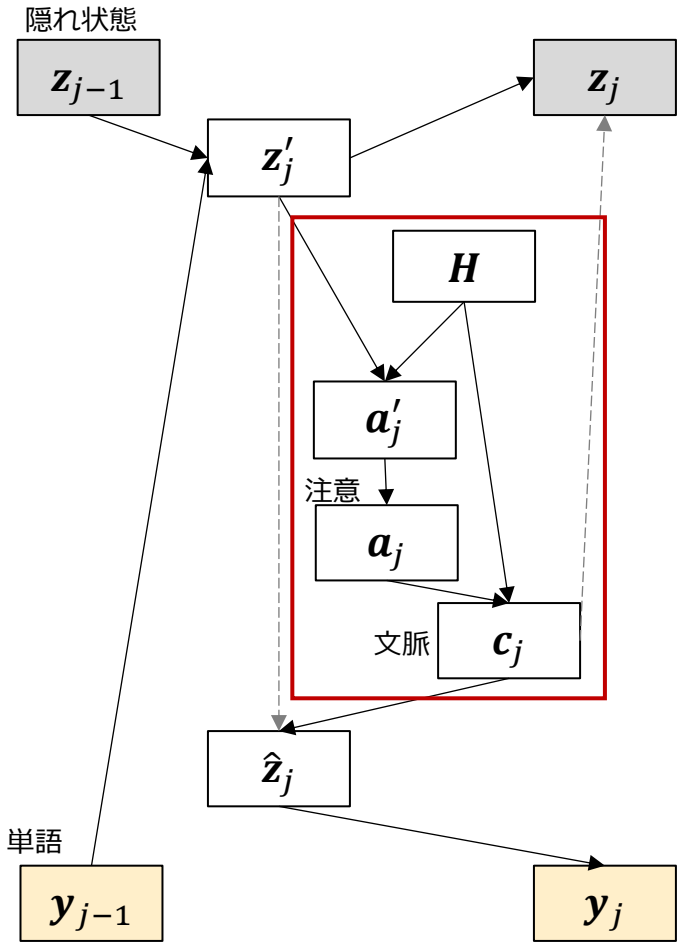
1

Bahdanau, Cho, & Bengio (2015 *ICLR*)
機械翻訳の文脈での研究。注意機構を提案した論文としてよく引用される。
Google Scholarによれば被引用回数41645件

- 直前の隠れ状態ベクトル $\boldsymbol{z}_{j-1}$を、直前に生成した単語$\boldsymbol{y}_{j-1}$によって更新し、$\boldsymbol{z}'_j$を得ておく
  - たとえば再帰型ニューラルネットワークを用いる

- $\boldsymbol{z}'_j$ の観点からみた、$\boldsymbol{H}$の各列の重要度 $\boldsymbol{a}'_j$を求める
  - 典型的には内積 $\boldsymbol{a}_j' = \boldsymbol{H}^\top \boldsymbol{z}'_j$
  - $\boldsymbol{z}'_j$ と$\boldsymbol{H}$の各列との間の類似性が高いとき、重要度が高くなる
- $\boldsymbol{H}$の各列にどのくらい「**注意**」しなければならないかを決める
  - $\boldsymbol{a}_j = \mathrm{softmax}(\boldsymbol{a}_j')$
  - 重要度を、下限0, 合計が1になるように調整している
- 文脈ベクトルをつくる
  - $\boldsymbol{c}_j = \boldsymbol{H}\boldsymbol{a}_j$
  - 行列$\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_I)$の各列を、注意で重みづけて足し上げている

注意機構



- 単語生成のためのベクトル$\hat{\boldsymbol{z}}_j$ をつくる
  - 文脈ベクトル $\boldsymbol{c}_j$そのもの、ないし$\boldsymbol{c}_j$ と $\boldsymbol{z}'_j$を結合したベクトルを、$\hat{\boldsymbol{z}}_j$とする
- 新しい隠れ状態ベクトル $\boldsymbol{z}_j$をつくる
  - $\boldsymbol{z}'_j$ そのものを$\boldsymbol{z}_j$とする。ないし、$\boldsymbol{z}'_j$を文脈ベクトル$\boldsymbol{c}_j$で更新し$\boldsymbol{z}_j$とする

7.7 Transformer

- Transformerとは (岡崎ほか(2022) 6.1節)
  - ニューラル言語モデル・系列変換モデルのひとつ
  - 2017年、Googleの研究者らにより提案
  - 2018年、Googleが発表した言語モデル BERT で採用され、爆発的に普及

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Vaswani, et al. (2017 *NIPS*)
Transformerを提案した論文。題名”Attention Is All You Need”はビートルズへのオマージュ。
Google Scholarによれば被引用回数209837件。
えええええ？

- たくさんのアイデアが幕の内弁当のように詰め込まれている
  - 自己注意、QKV注意、マルチヘッド注意、位置符号、残差結合、層正規化、フィードフォワード層、…

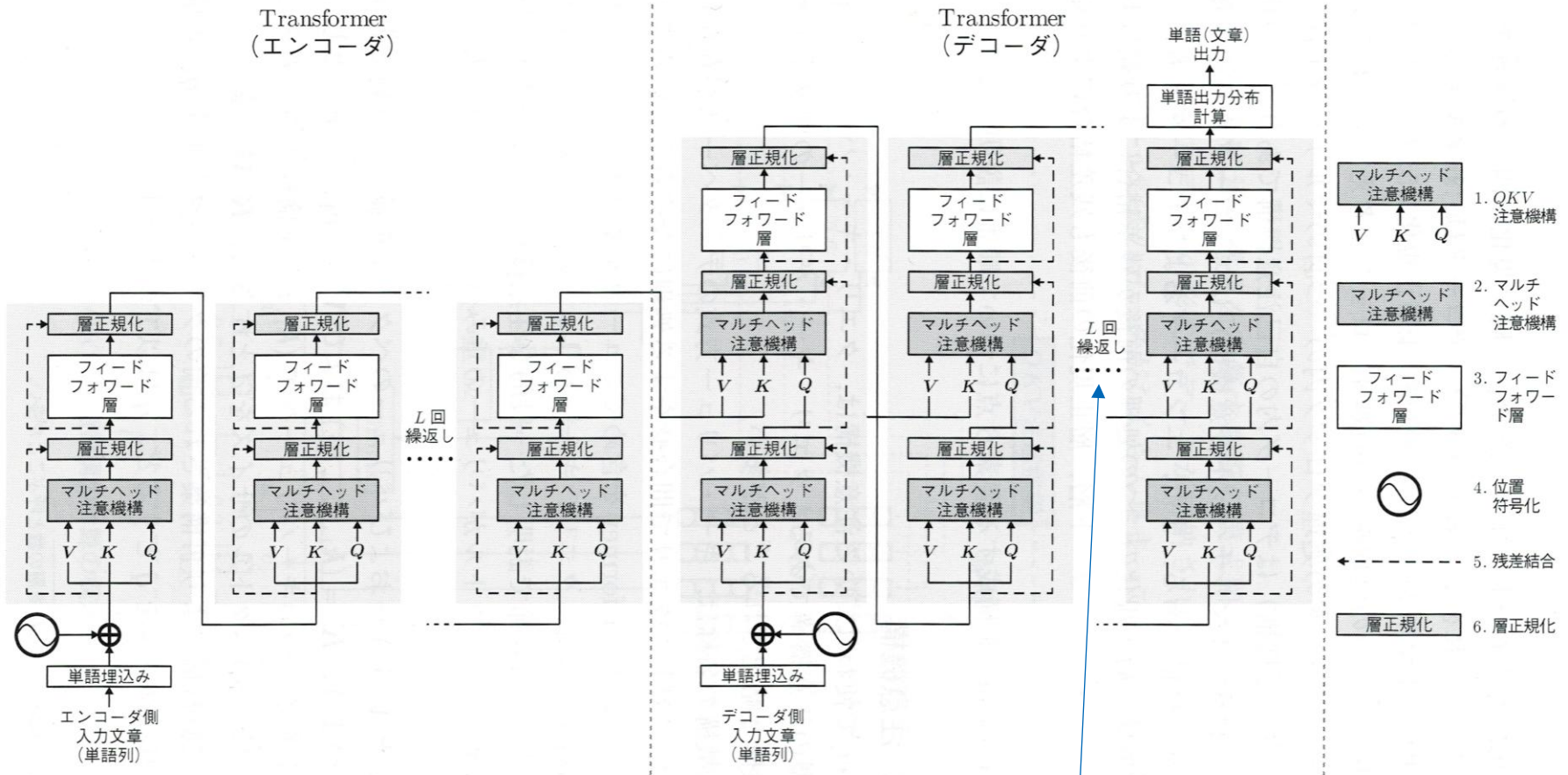


図 6.1　Transformer の全体構成図

元論文では6層。その後の大規模言語モデルではより多くの層が用いられている模様。たとえばGPT-3では96層だそうです

図の出典：岡崎ほか (2022)

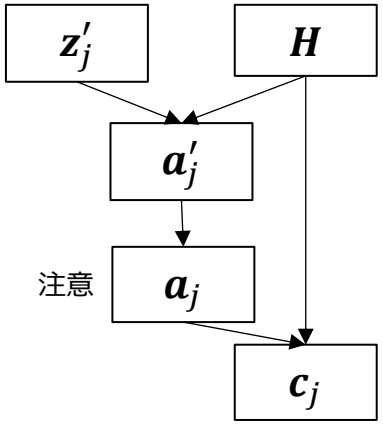
- 次の2つの部分に分かれる

- Transformer エンコーダ
  - 抽象的にいえば、従来の系列言語モデルにおけるエンコーダと同じ
    - 単語 $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_I$ を受けとり、隠れ状態ベクトル$\boldsymbol{h}_0, \ldots, \boldsymbol{h}_I$を出力する
  - たくさんの層の積み重ねからなる
  - 再帰型ニューラルネットワークではなく、**自己注意機構**を使う

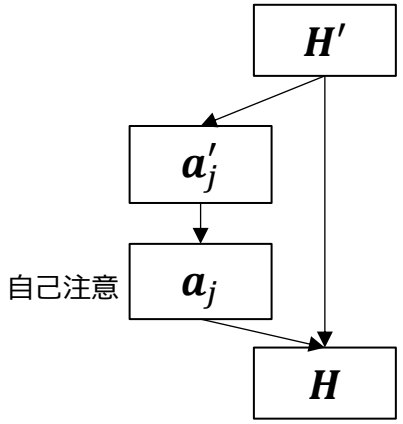
- Transformer デコーダ
  - 抽象的にいえば、従来の系列言語モデルにおけるデコーダと同じ
    - 位置 j における単語 $\boldsymbol{y}_j$ を生成する
    - 入力は、それまでに出力した単語列 $\boldsymbol{Y}_{0:j-1} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{j-1})$と、エンコーダから受け取った $\boldsymbol{H} = (\boldsymbol{h}_0, \ldots, \boldsymbol{h}_I)$
  - たくさんの層の積み重ねからなる
  - 再帰型ニューラルネットワークではなく、自己注意機構を使う
    - その際、エンコーダが作成した$\boldsymbol{H}$を注意機構によって利用する

- 自己注意機構とは (岡崎ほか(2022) 6.2節)
  - 入力された単語系列からそれぞれの位置における隠れ状態ベクトルをつくる際の仕組み
  - 従来の注意機構の変形と捉えることができる
    - 従来の系列位置モデルにおける注意機構:
      - エンコーダで作った隠れ状態ベクトル列$\boldsymbol{H}$を、デコーダの現在の位置での隠れ状態ベクトルの観点から合成し、文脈ベクトルをつくる
    - Transformer エンコーダにおける自己注意機構:
      - 入力単語系列を表す埋め込みベクトル列を、ある位置の埋め込みベクトルの観点から合成し、隠れ状態ベクトルをつくる

従来の系列変換モデルにおける注意機構



tranformer エンコーダにおける自己注意機構



従来の系列変換モデルにおける注意機構

- 入力
  - エンコーダからもらった、単語系列 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I$の各位置における隠れ状態ベクトル列 $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_I)$
  - デコーダ側の現在の隠れ状態ベクトル $\boldsymbol{z}'_j$
- 手続き
  - $\boldsymbol{z}'_j$ の観点からみた、$\boldsymbol{H}$の各列の重要度 $\boldsymbol{a}'_j$を求める
    - 典型的には内積 $\boldsymbol{a}_j{}' = \boldsymbol{H}^\top \boldsymbol{z}'_j$
  - $\boldsymbol{H}$の各列にどのくらい「**注意**」しなければならないかを決める
    - $\boldsymbol{a}_j = \mathrm{softmax}(\boldsymbol{a}_j{}')$
    - 重要度を、下限0, 合計が1になるように調整している
  - 位置 j における文脈ベクトルをつくる
    - $\boldsymbol{c}_j = \boldsymbol{H}\boldsymbol{a}_j$
    - エンコーダの隠れ状態ベクトル列$\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_I)$を、現在の隠れ状態の観点からみた注意で重みづけて足し上げている

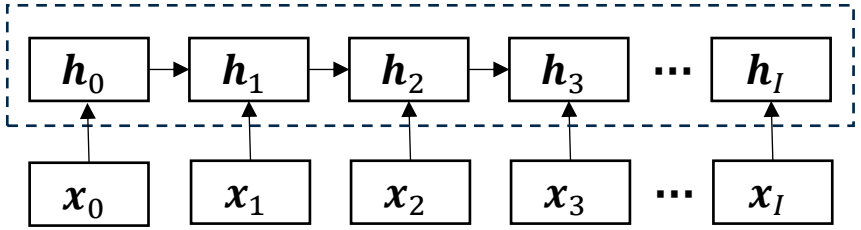
tranformer エンコーダにおける自己注意機構

- 入力
  - 単語系列 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I$のそれぞれの埋め込みベクトル列 $\boldsymbol{H}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_I)$
- 手続き
  - $\boldsymbol{h}'_i$ の観点からみた、$\boldsymbol{H}'$の各列の重要度 $\boldsymbol{a}'_i$を求める
    - 典型的には内積 $\boldsymbol{a}_i{}' = \boldsymbol{H}'^\top \boldsymbol{h}'_i$
  - $\boldsymbol{H}'$の各列にどのくらい「**注意**」しなければならないかを決める
    - $\boldsymbol{a}_i = \mathrm{softmax}(\boldsymbol{a}_i{}')$
    - 重要度を、下限0, 合計が1になるように調整している
  - 位置 i における隠れ状態ベクトルをつくる
    - $\boldsymbol{h}_i = \boldsymbol{H}'\boldsymbol{a}_i$
    - 埋め込みベクトル列$\boldsymbol{H}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_I)$を、各列の観点からみた注意で重みづけて足し上げている

- 再帰型ニューラルネットワークと比べると
  - 再帰型ニューラルネットワークの場合:
    - 位置 $i, \ldots, I$ における隠れ状態ベクトルを**順に**つくる
    - → 直前の位置の単語に影響されやすい
  - 自己注意機構の場合:
    - すべての位置における隠れ状態ベクトルを**一気に**作る
    - → 離れた位置にある単語も参照できる

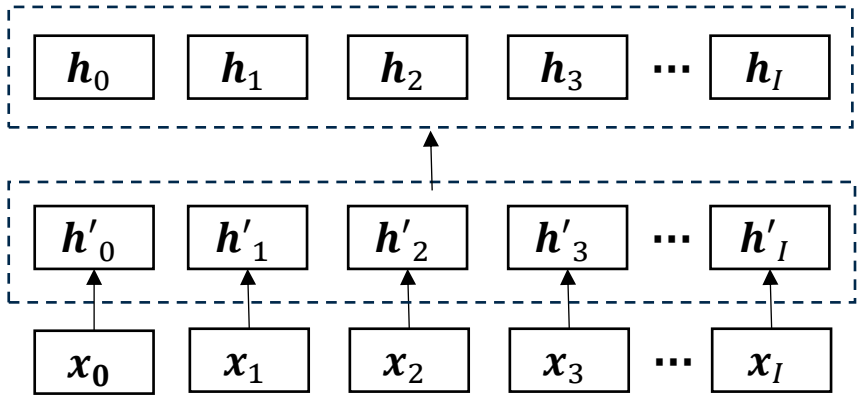
再帰型ニューラルネットワーク

- 入力
  - 単語系列 $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_I$
- 手続き
  - 位置 $1, \ldots, I$ について、**順に**隠れ状態ベクトル $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_I$ を作る
    - $\boldsymbol{h_i} = g(\boldsymbol{W}_1 \boldsymbol{x_{i-1}} + \boldsymbol{W}_2 \boldsymbol{h_{i-1}} + \boldsymbol{b})$



tranformer エンコーダにおける自己注意機構

- 入力
  - 単語系列 $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_I$ のそれぞれの埋め込みベクトル列 $\boldsymbol{H}' = (\boldsymbol{h}'_0, \ldots, \boldsymbol{h}'_I)$
- 手続き
  - 隠れ状態ベクトルを**一気に**つくる
    - $\boldsymbol{h}_i = \boldsymbol{H}' \mathrm{softmax}(\boldsymbol{H}'^{\top} \boldsymbol{h}'_i)$



7.8 事前学習済み言語モデル

- **事前学習済み言語モデル**とは
  - 大量の言語資料に基づいてパラメータを学習した言語モデル
  - t番目の単語の予測だけでなく、さまざまな用途に用いられている
  - パラメータ数がある程度以上大きいとき、**大規模言語モデル** (LLM) と呼ばれる

- 現在実用化されている事前学習済み言語モデルのほぼすべてが、Transformerアーキテクチャに基づいている

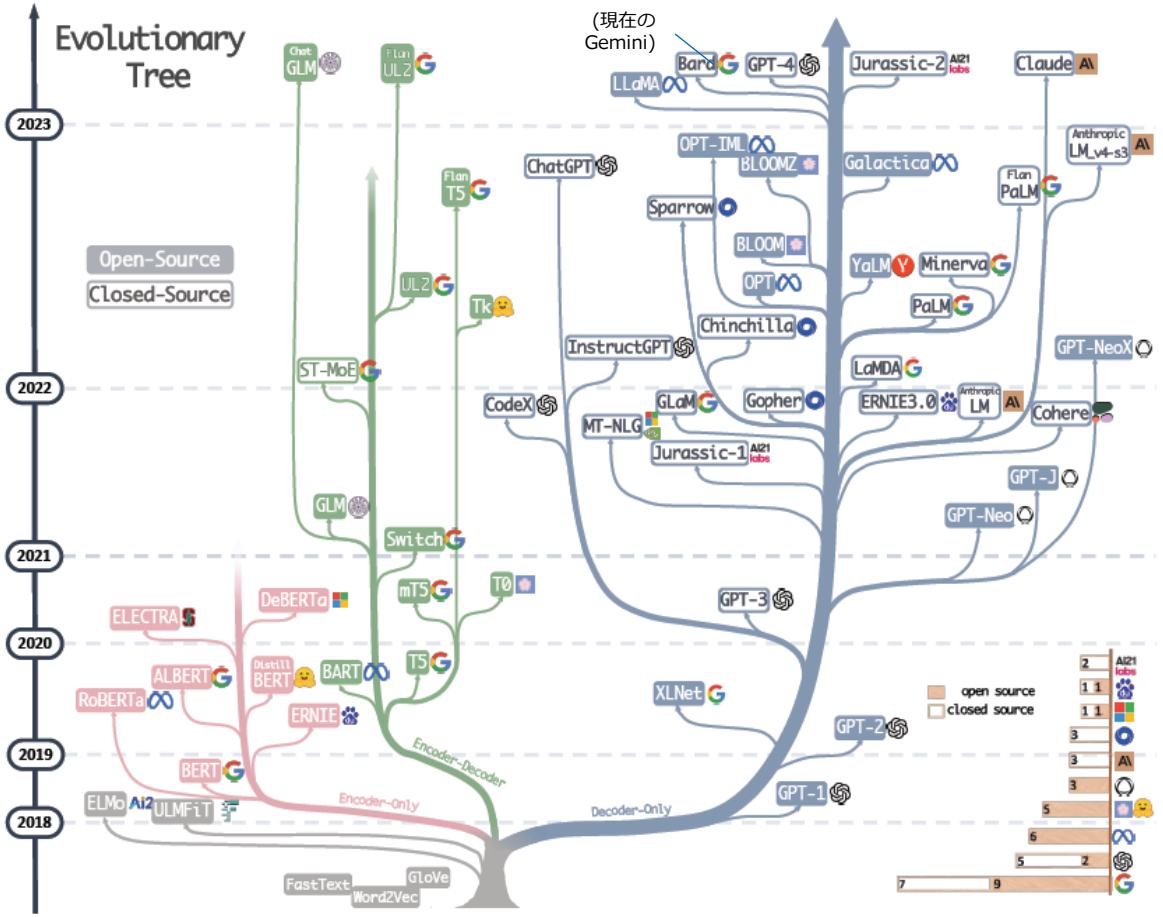


Fig. 1. The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models. Models on the same branch have closer relationships. Transformer-based models are shown in non-grey colors: decoder-only models in the blue branch, encoder-only models in the pink branch, and encoder-decoder models in the green branch. The vertical position of the models on the timeline represents their release dates. Open-source models are represented by solid squares, while closed-source models are represented by hollow ones. The stacked bar plot in the bottom right corner shows the number of models from various companies and institutions.

大規模言語モデルの進化を示す図。
左下のグレーで書かれたモデルを除き、すべてTransformerベース
(グレーのモデルはほぼすべて単語埋め込みモデル)

図の出典: Yang et al. (2023)

(7章のまとめ)

- ニューラル言語モデルとは

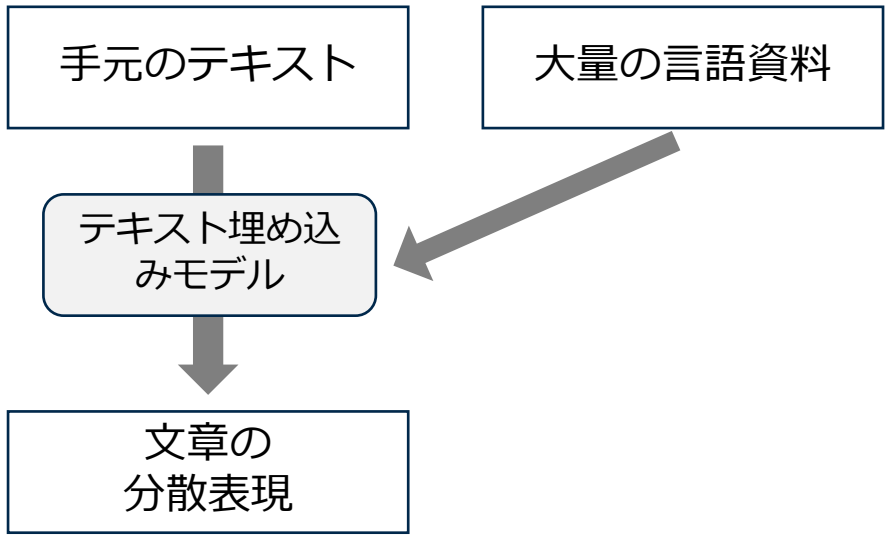
- ニューラル言語モデルは、単語系列のある位置において、どの単語を合成しているか？
  - 順伝播型ニューラル言語モデル
    - 直前の n 個の単語を合成している
  - 再帰型ニューラル言語モデル
    - 原理的には、文章の先頭から直前までのすべての単語を合成している。ただし直前の単語の影響を受けやすい
  - Transformer エンコーダ
    - その文章に含まれるすべての単語を合成している

8. ニューラル言語モデルによる文章の分散表現

(8章の内容)

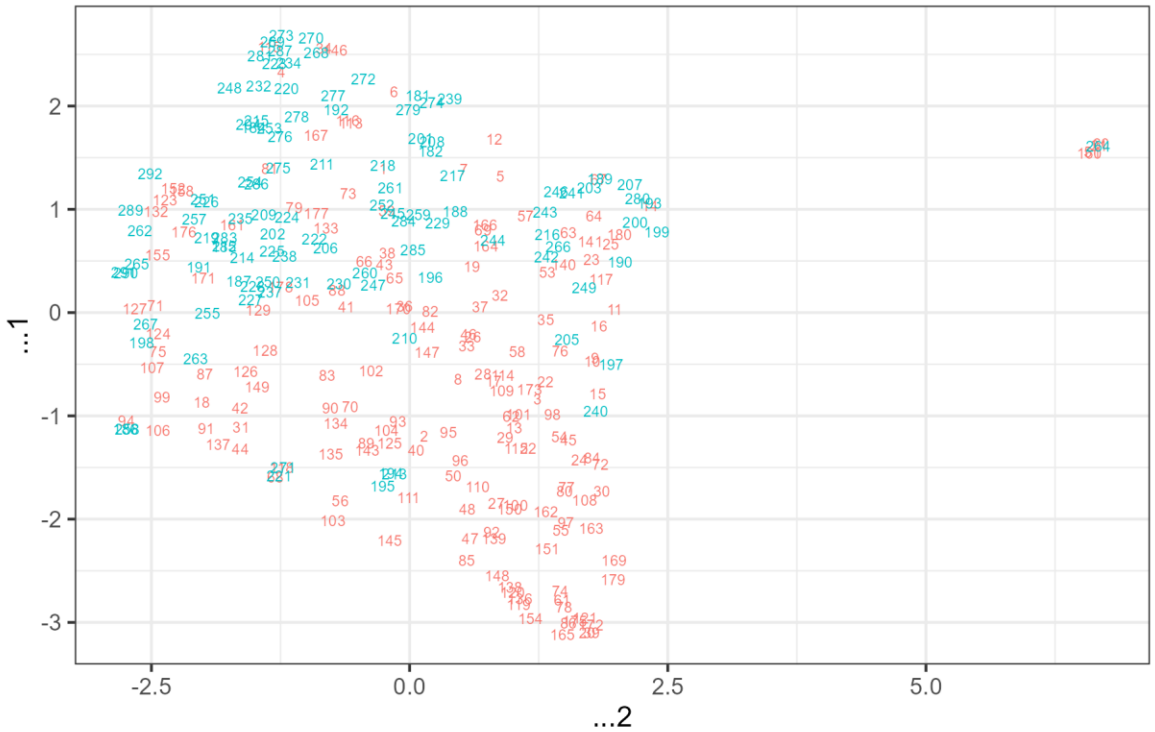
- 大量の言語資料を用いて、文章を分散表現に換えるためのモデル(テキスト埋め込みモデル)を構築する方法を紹介する
- 手元のテキスト・データについての、分散表現を用いた分析方法を紹介する

8章. ニューラル言語モデルによる文章の分散表現



8.1 文章の埋め込み表現

- 文章の埋め込み表現をどのように得るか？
  - その文章に含まれている単語の埋め込みベクトルを平均すればよいのでは？
    - → 単語の出現順序を無視している



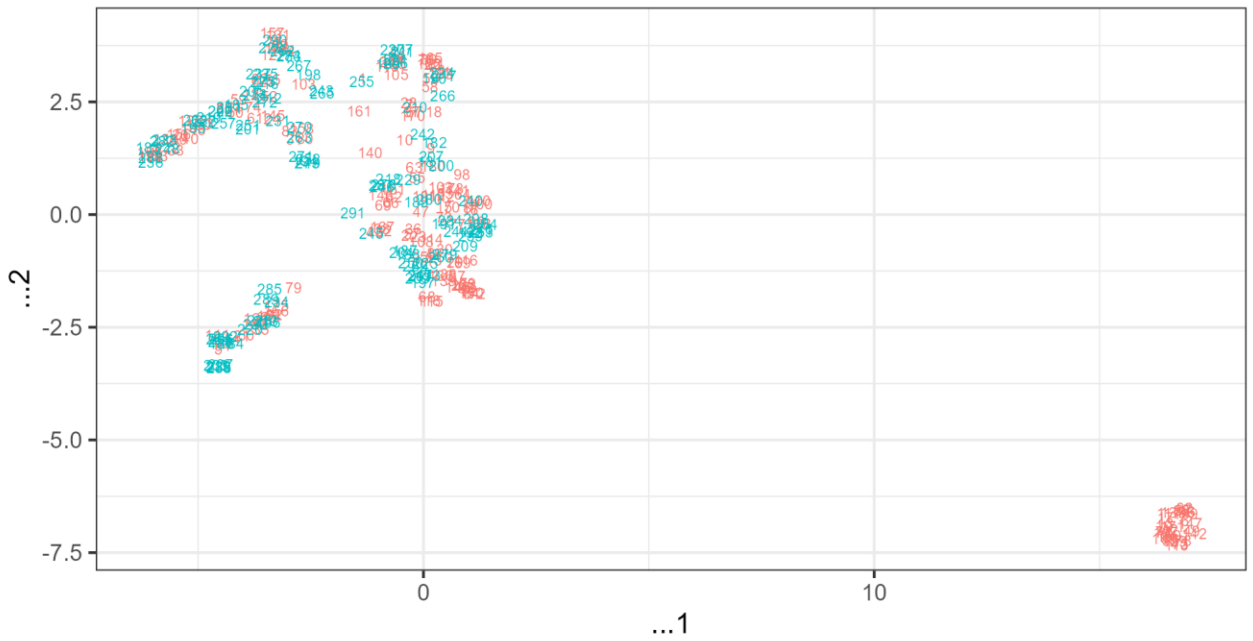
物語

a ごん狐

a 手袋を買いに

「ごん狐」「手袋を買いに」に含まれる文(292文)について、各文に含まれる単語のword2vec埋め込みベクトルを平均し、各文のベクトルをえた。それらをUMAPで二次元空間にマッピングした。
右側には「と思いました」を含む文が集まっている

- 文章をTransformerエンコーダに与え、最終層における隠れ状態ベクトル(たとえば$\boldsymbol{h}_0$)を用いればよいのでは？
  - → 意外にうまくいかない。文章のベクトル間の類似性は、必ずしも人間からみた類似性を表さない



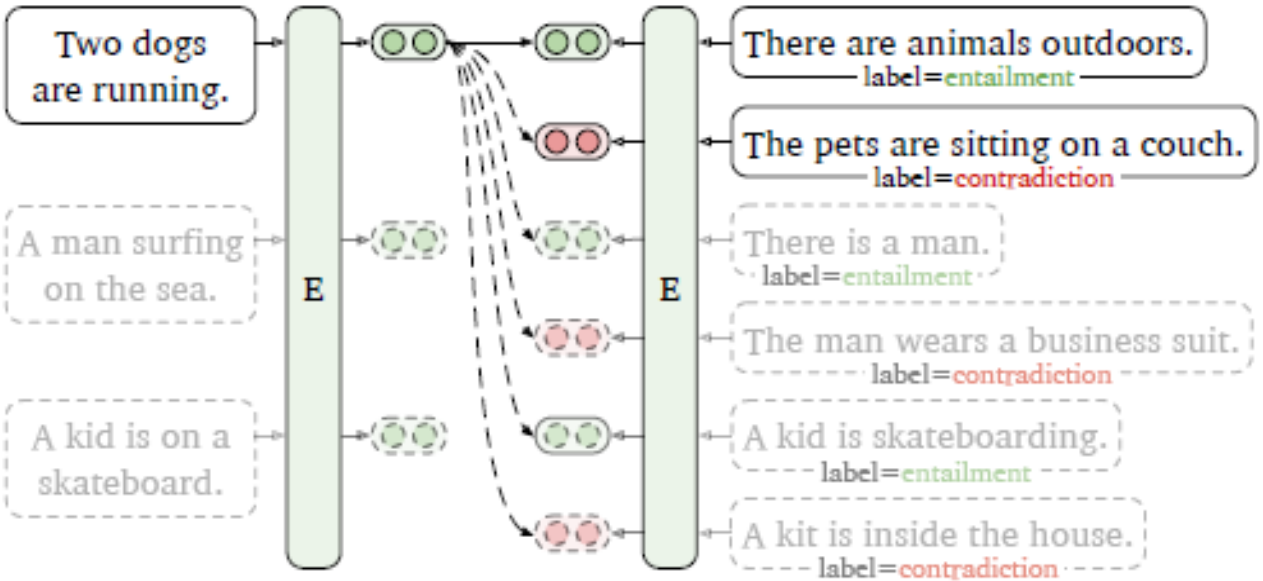
「ごん狐」「手袋を買いに」に含まれる文(292文)を、Transformerエンコーダに基づく言語モデルBERTに与え、各文に対応する最終層の文頭(CLSトークン)に対応する隠れ状態ベクトルを得た。それらをUMAPで二次元空間にマッピングした。
右下には「ごんは…」からはじまる文章が集まっている

8.2 テキスト埋め込みモデル

- 事前学習済み言語モデルは、「事前学習」と「ファインチューニング」という枠組みを採用している
  - 事前学習
    - 大量の言語資料を学習し、t番目の単語を予測するモデルを構築する
  - ファインチューニング
    - 実際の応用場面に近い課題を用いて再度学習を行い、パラメータを調整する

- **テキスト埋め込みモデル**
  - 事前学習済み言語モデルをファインチューニングしたモデル
    - ほとんどのモデルは Transformer エンコーダに基づく
  - 文章を受け取り、人間からみて適切な文章埋め込みベクトルを出力する

- ファインチューニングの方法: **対照学習**
  - 人間からみて近い文は埋め込みベクトルが近くなるように、遠い文は遠くなるように学習する
  - 例) Supervised SimSCE (Gao et al., 2022)
    - {ある文、それが含意している文}のペアや、{ある文、それと矛盾している文} のペアを大量に集めたデータセットを用いる
    - 含意しているペアでは埋め込みベクトルが近くなるように、矛盾しているペアでは遠くなるように学習する



図の出典: Gao et al. (2022)

- 多言語対応のテキスト埋め込みモデル
  - OpenAIのTest Embedding 3, MicrosoftのMultilingual E5 など
- 日本語に特化したテキスト埋め込みモデル (下図)

埋め込み (Embeddings) 作成に特化したモデル [20]

Bi-Encoders

Single-representation bi-encoders

	入力で扱えるトークン数	開発元	ライセンス
Ruri-v3 (v3-30m, v3-70m, v3-130m, v3-310m)	8,192	名大 笹野研	Apache 2.0
PLaMo-Embedding-1B (1b)	4,096	Preferred Networks	Apache 2.0
Sarashina-Embedding-v2 (v2-1b)	8,192	SB Intuitions	Sarashina Model NonCommercial License
sbintuitions/sarashina-embedding-v1-1b	8,192	SB Intuitions	Sarashina Model NonCommercial License
AMBER (base, large)	512	レトリバ	Apache 2.0
RoSEtta (base-ja)	1,024	PKSHA Technology	Apache 2.0
GLuCoSE v2 (base-ja-v2)	512	PKSHA Technology	Apache 2.0
Ruri (small, base, large, small-v2, base-v2, large-v2)	512	名大 笹野研	Apache 2.0
Japanese SimCSE (unsup-simcse-ja-base, unsup-simcse-ja-large, sup-simcse-ja-base, sup-simcse-ja-large)	512	名大 笹野研	CC BY-SA 4.0
GLuCoSE (base-ja)	512	PKSHA Technology	Apache 2.0
colorfulscoop/sbert-base-ja		Colorful Scoop	CC BY-SA 4.0
MU-Kindai/SBERT-JSNLI-base MU-Kindai/SBERT-JSNLI-large		近畿大学 (研究室不明)	?
MU-Kindai/Japanese-SimCSE-BERT-base-unsup MU-Kindai/Japanese-SimCSE-BERT-large-unsup MU-Kindai/Japanese-SimCSE-RoBERTa-base-unsup MU-Kindai/Japanese-SimCSE-BERT-base-sup MU-Kindai/Japanese-SimCSE-BERT-large-sup		近畿大学 (研究室不明)	MIT
pkshatech/simcse-ja-bert-base-clcmlp		PKSHA Technology	CC BY-SA 4.0
MU-Kindai/Japanese-MixCSE-BERT-base MU-Kindai/Japanese-MixCSE-BERT-large		近畿大学 (研究室不明)	MIT
MU-Kindai/Japanese-DiffCSE-BERT-base		近畿大学 (研究室不明)	MIT
bclavie/fio-base-japanese-v0.1		個人 (Benjamin Clavié)	
cl-nagoya/shioriha-large-pt		名大 笹野研	

Multi-representation bi-encoders

	開発元	ライセンス
JaColBERTv2.5 (JaColBERTv2.4, JaColBERTv2.5)	Answer.AI	MIT
JaColBERTv2 (JaColBERTv2)	個人 (Benjamin Clavié)	MIT
JaColBERT (JaColBERT)	個人 (Benjamin Clavié)	MIT

図の出典: LLM勉強会 (2025)

- 実は、テキスト埋め込みモデルは最近のホットトピック… らしい
  - LLMでRAG (検索拡張生成) を利用する際には、あらかじめ外部データを埋め込みベクトルに変換しておくのだそうです
  - ローカルLLMとRAGの普及のせいで、テキスト埋め込みモデルへの注目が高まったそうです。へえー

言語処理学会 第31回年次大会 発表論文集 (2025年3月)

Ruri: 日本語に特化した汎用テキスト埋め込みモデル

塚越 駿　　笹野 遼平
名古屋大学大学院情報学研究科
tsukagoshi.hayato.r2@s.mail.nagoya-u.ac.jp
sasano@i.nagoya-u.ac.jp

概要

近年，英語や多言語の汎用的なテキスト埋め込みモデルの開発が盛んに行われている．しかし，日本語でのモデル開発の取り組みは限定的であり，その理由としてはデータセットの不足やモデル開発のための知見が少ないことが挙げられる．本稿では，日本語汎用テキスト埋め込みモデル Ruri を開発し，その過程について述べる．具体的には，訓練データの不足を補うための大規模言語モデルによる合成データセット構築，対照事前学習によるベースモデルの訓練，そして高品質データを用いた微調整について説明する．構築したテキスト埋め込みモデル Ruri は，日本語テキスト埋め込みのベンチマークにおいて既存のモデルを上回る性能を達成した．

1　はじめに

テキスト埋め込みは，検索拡張生成や類似文書検索などのタスクで広く利用されており [1, 2]，特に近年は多様なデータセットで学習した汎用的なテキスト埋め込みモデルの開発が盛んに行われている [3, 4, 5, 6, 7]．しかし，これらの取り組みは主に英語モデルや多言語モデルの構築を目標としており，日本語の語彙・データは相対的に小さな割合にとどまっている．大規模な日本語データセットを用いて埋め込みモデルを学習すれば，これらの多言語モデルより日本語を対象としたタスクにおいて高性能なモデルの実現が期待できるが，テキスト埋め込みの学習に利用できる日本語資源は限定的である．

本稿では，テキスト埋め込みモデルの学習に利用できるような大規模日本語データセットを構築し，日本語に特化したテキスト埋め込みモデルを開発する．具体的には，既存データセットの収集・整理と，大規模言語モデル (LLM) による合成データセットの構築を行い，構築されたデータセットに基づき，対照学習による埋め込みモデルの事前学習と高品質な人手データによる微調整を行った．構築されたテキスト埋め込みモデル Ruri は，日本語テキスト埋め込みのベンチマークで既存のモデルを大きく上回る性能を達成した．本研究で構築したモデルおよびデータセットは公開している[1]．

2　対照事前学習

近年のテキスト埋め込みモデル開発では，2 段階の学習アプローチを取ることが一般的である [3, 5, 6]．具体的には，まず大規模な弱教師データセットで対照学習損失 [8] に基づく事前学習 (対照事前学習) を行い，その後，人手で作成された高品質なデータを用いてモデルの微調整を行う．本稿ではこの流れにしたがい，対照事前学習に基づいて日本語テキスト埋め込みモデルの強力なベースモデルを構築し，微調整によって性能を高めることを目指す．しかし，英語や多言語モデルと異なり，日本語の汎用テキスト埋め込みモデル構築には，訓練データ量が不足している．そこで，本稿では既存の資源を収集するとともに，LLM を用いてデータ合成を行うことでその不足を補う．本研究では，対照事前学習用のデータセットを v1 と v2 の 2 通り構築する．それぞれのデータセットに含まれるデータの内訳を表 1 と表 2 に示す．既存のデータセットについては付録 A に記載する．

2.1　合成データセット

近年，テキスト埋め込みの学習に合成データを活用する取り組みが盛んであり [9, 10, 11, 12]，特に日本語のような公開データセットの少ない言語では有用と考えられる．本稿でも，LLM を用いてテキスト埋め込みモデル学習用のデータを合成する．

本稿では，検索・QA データセットや自然言語推論 (NLI) データセットなど複数種類のデータを

1) https://huggingface.co/collections/cl-nagoya/ruri-japanese-general-text-embeddings-66cf1f3ee0c8028b89d85b5e

— 1622 —

This work is licensed by the author(s) under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/).

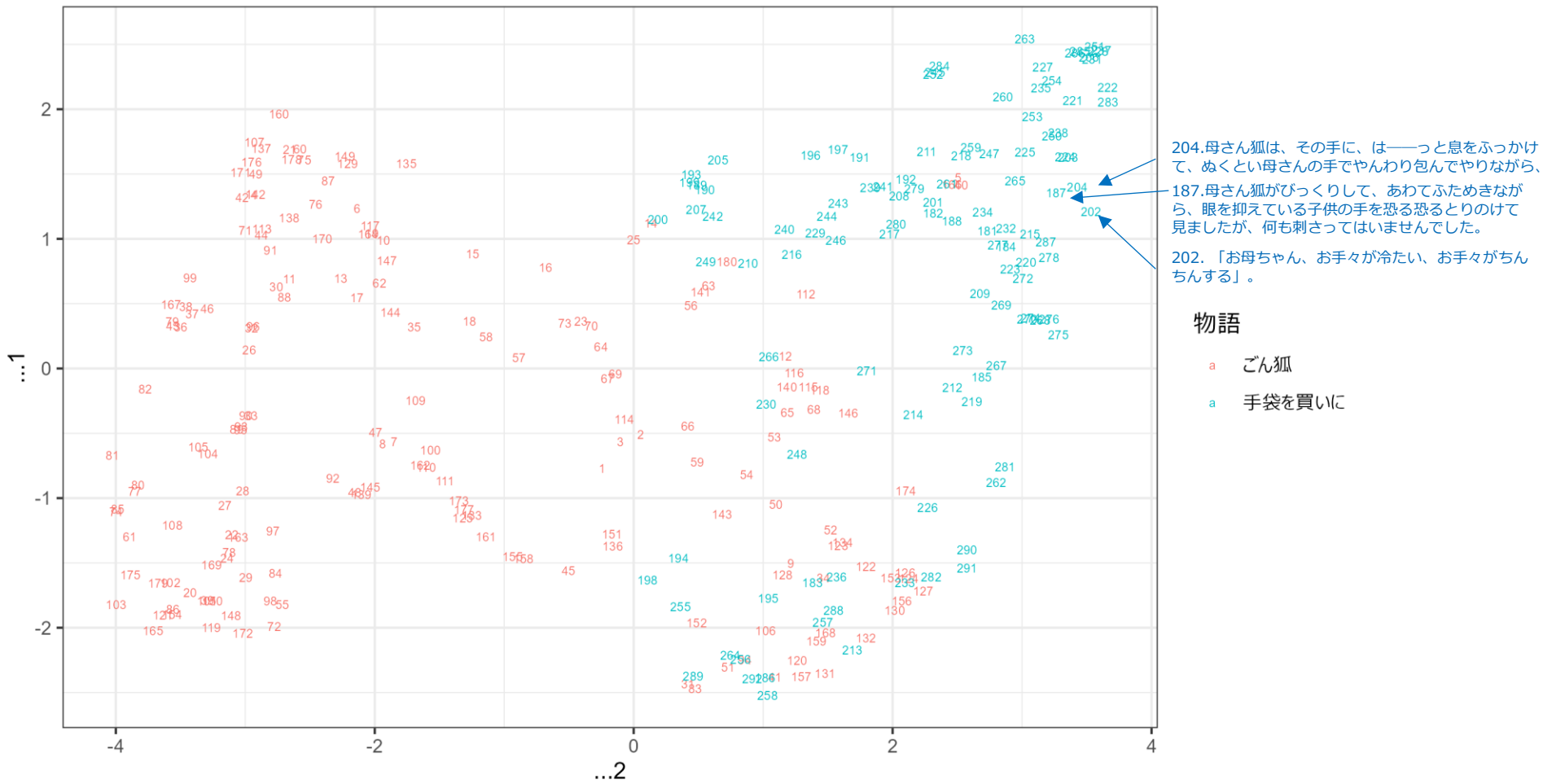
塚越・笹野(2025 言語処理学会)
最近話題となっている、日本語テキスト埋め込みモデル Ruri についての学会発表

8.3 分析例

- 「ごん狐」「手袋を買いに」に含まれている292文について、テキスト埋め込みモデル Ruri によって埋め込みベクトルを求めた

文番号	文	埋め込みベクトル (長さ768)										
1	これは、私が小さいときに、村の茂平というおじいさんからきいたお話です。	0.49	3.70	-0.56	-0.09	1.30	…	-0.34	-0.10	0.26	0.34	0.73
2	むかしは、私たちの村のちかくの、中山というところに小さなお城があって、中山さまというおとのさまが、おられたそうです。	0.57	4.06	0.14	0.80	-0.02	…	0.16	0.45	0.48	0.02	-0.15
3	その中山から、少しはなれた山の中に、	-0.92	2.30	0.00	0.58	-0.33	…	-0.65	0.10	0.17	0.55	0.54
4	「ごん狐」	-0.33	2.64	-0.45	-0.06	1.07	…	-0.51	1.31	0.13	1.01	0.61
5	という狐がいました。	-0.27	2.43	-1.12	-0.29	1.60	…	-0.30	1.10	0.93	1.38	0.87
…	…	…	…	…	…	…	…	…	…	…	…	…

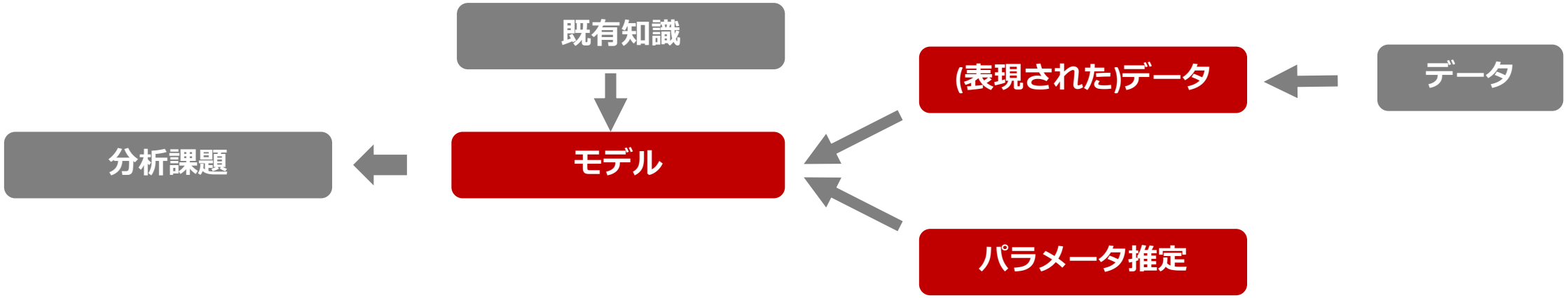
- 意味が近い文は、ベクトルも類似している … ような気がする
- 物語のなかで連続している文は、ベクトルも類似しているようにみえる



埋め込みベクトルをUMAPで二次元空間にマッピングした

8.4 テキスト埋め込みを活用した分析手法

- 定型的な分析課題に対しては、テキスト埋め込みモデルによる埋め込み表現への変換と、モデル構築・パラメータ推定をセットにした分析手法が用意されている場合がある
- ここでは、そのうち2つの手法を紹介する



- 文章のカテゴリ分類
  - 定型的な分析課題
    - なんらかのカテゴリを想定し、たくさんの文章をそれらのカテゴリにわける (**zero-shot分類**)

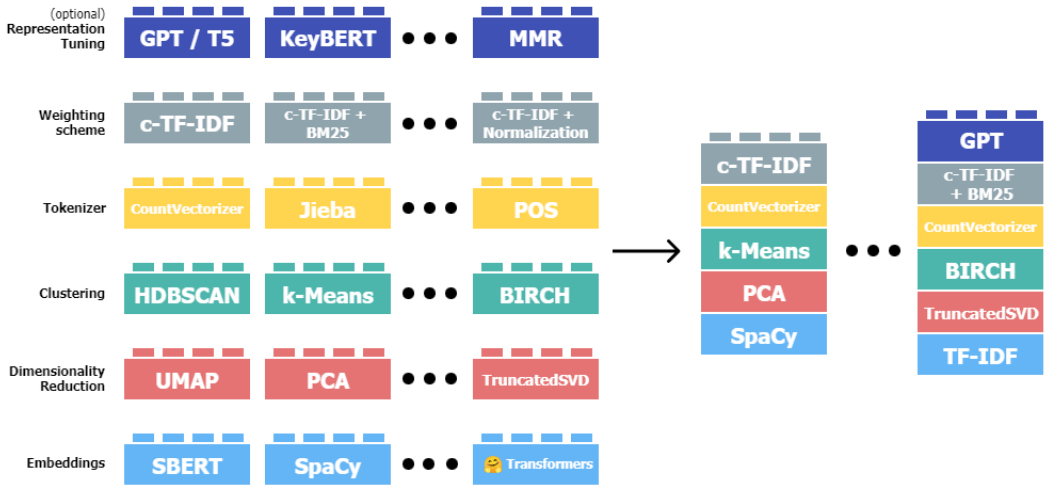
  - 埋め込み表現を用いる場合の典型的な分析手順
    1. データの観察や議論を通じて、カテゴリを定める
    2. 各カテゴリに適切なカテゴリラベルをつける
    3. テキスト埋め込みモデルによってカテゴリラベルと文章の埋め込み表現を得る
    4. 次元を縮約する
    5. なんらかのモデルに基づいて文章をカテゴリへと分類する
    6. 得られた分類の特徴をわかりやすく表現する

  - 自然言語処理用のソフトウェアはzero-shot分類の機能を提供している
    - 内部では上記の3,4,5を行っていることが多い

- 文章のクラスタリング
  - 定型的な分析課題
    - たくさんの文章を未知のクラスタにわけ、クラスタを命名・解釈する

  - 埋め込み表現を用いる場合の典型的な分析手順
    1. テキスト埋め込みモデルによって文章の埋め込み表現を得る
    2. 次元を縮約する
    3. なんらかのモデルに基づいてクラスタリングする
    4. 得られたクラスタの特徴をわかりやすく表現する

  - **BERTopic**
    - 上記の手続きをセットにした分析手法



図の出典: Grootendorst, M. (2024)

(8章のまとめ)

- テキスト埋め込みモデルとは
- テキスト埋め込みを活用した分析手法
  - 文章のクラスタリング
  - 文章のカテゴリ分類

9. マーケティング・リサーチとテキスト分析

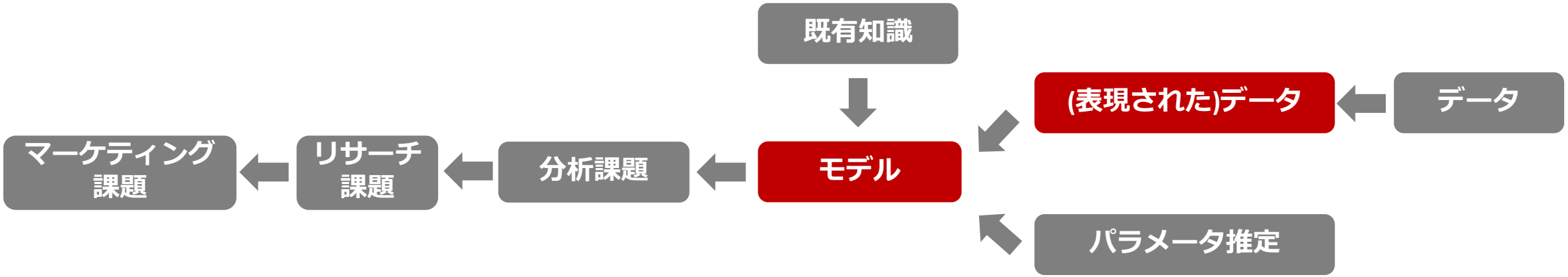
(9章の内容)

- マーケティング・リサーチとテキスト分析に関して、2つのポイントについて論じる

- ポイント1. 事前学習済みモデルを用いた分散表現を用いるべきか
- ポイント2. 埋め込み表現を利用する際、どのようにしてモデルを構築するか

9.1 事前学習済みモデルを用いた分散表現を用いるべきか

- その分析課題・モデルから見て、(表現された)データは一般的な言語知識を反映しているほうがよいだろうか？
  - 多くの場合、答えはYes
    - → 事前学習済みモデルによる分散表現 (埋め込み表現) を利用する
  - しかし…
    - モデルのパラメータの解釈はかえって困難になるかもしれない
      - テキストの分散表現は、その個々の要素が持つ意味がはっきりしない
    - 一般的な言語知識と領域的な言語知識のずれによって、テキストが持つ特徴がかえってわかりにくくなるかもしれない
      - 特定の領域についてのテキストを分析する場合、深刻な問題となる



9.2事前学習済みモデルを用いた分散表現を用いる際のモデル構築

- 路線1. 埋め込み表現を活用した定型的な分析手法を利用する
  - 定型的な分析課題に向いている
    - 例) 文章のクラスタリング(BERTopic); 文章のカテゴリ分類 (zero-shot分類)
  - 欠点
    - モデルの設計に自由度がなく、既有知識を活用しにくい

- 路線2. 自力で埋め込み表現を得たうえで、モデルを構築する
  - 定型的でない分析課題に向いている
    - 例) Arora, et al. (2025)
      - 分析課題: 異なるインタビュー手法の間で、発言記録の類似性を評価する
      - → 発言記録に含まれる発言を埋め込み表現に変換し、次元を縮約し、手法別にマッピングし、マップ間の類似性を求めている
  - 欠点
    - 面倒

Key Takeaway

- 大量の言語資料に基づく事前学習済みモデルを用いて、手元のテキストを分散表現に変換することで、一般的な言語知識を取り入れたテキスト分析を容易に行うことができる

ご清聴ありがとうございました！

参考文献

- 岡崎直観・荒瀬由紀・鈴木潤・鶴岡慶雅・宮尾雄介 (2022) 自然言語処理の基礎. オーム社.
- 岡崎直観(2023) 大規模言語モデル. 統計関連学会連合大会チュートリアルセッション配布資料. 2023/9/3.
- 金明哲 (2021) テキストアナリティクスの基礎と実践. 岩波書店.
- 白井 (2020) 学習済み日本語word2vecとその評価について. 株式会社ホクソエムのブログ. https://blog.hoxo-m.com/entry/2020/02/20/090000. 2025/12/29閲覧.
- LLM勉強会 (2025) 日本語LLMまとめ. https://llm-jp.github.io/awesome-japanese-llm/. 2025/12/29閲覧.
- Arora, N., Chakraborty, I., & Nishimura, Y. (2025) AI-Human Hybrids for Marketing Research: Leveraging Large Language Models (LLMs) as Collaborators. *Journal of Marketing*, 89(2), 43-70.
- Gao, T., Yao, X., & Chen, D. (2021) SimCSE: Simple Contractive Learning of Sentence Embeddings. *arXiv*:2104.08821.
- Grootendorst, M. (2024) BERTopic: The Algorithm. https://maartengr.github.io/BERTopic/algorithm/algorithm.html. 2026/02/14閲覧.
- Levy, O., & Goldberg, Y. (2014) Neural Word Embedding as Implicit Matrix Factorization. *NIPS'14: Proceedings of the 28th International Conference on Neural Information Processing Systems.* 2:2177-2185.
- Scott (1961) John Rupert Firth. *Bulletin of the School of Oriental and African Studies*. 24(3):412-418.
- Yang, J., et al. (2023) Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *arXiv:*2304.13712.