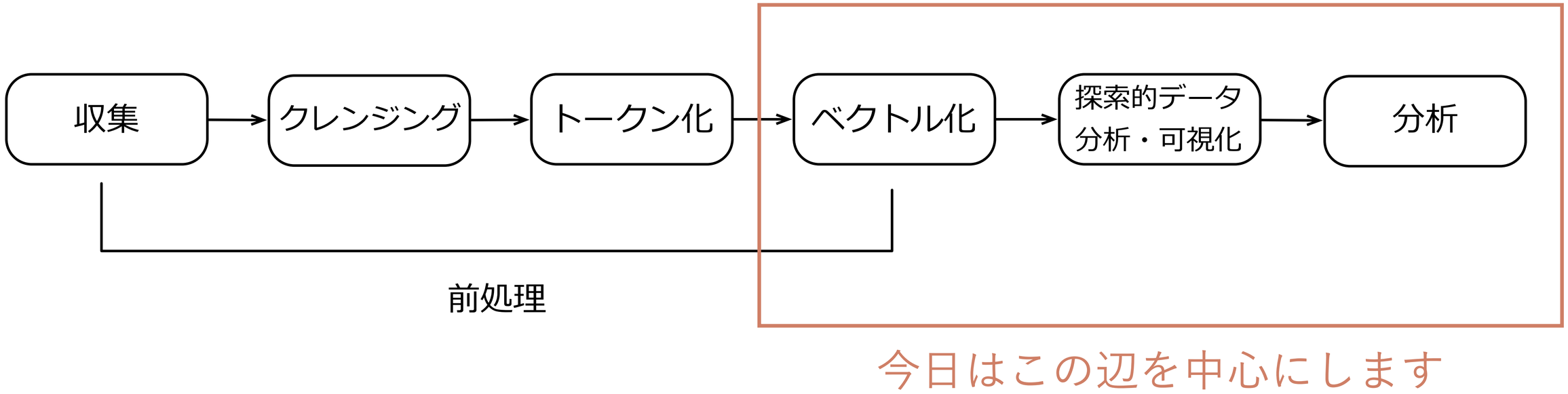
テキスト分析(2)

社会システム科学

テキスト分析の手順（後半）

テキスト分析の手順



ベクトル化

- トークン単位のベクトル化
  - One-hot表現
  - Word2Vec
- テキストデータ単位のベクトル化
  - 特徴量ベクトル
  - Bag-of-Words (BoW)
  - TF-IDF
  - 潜在トピックモデル (LDA)
  - Doc2Vec

特徴量に基づくベクトル化

- 先頭50行の特徴量ベクトルの例

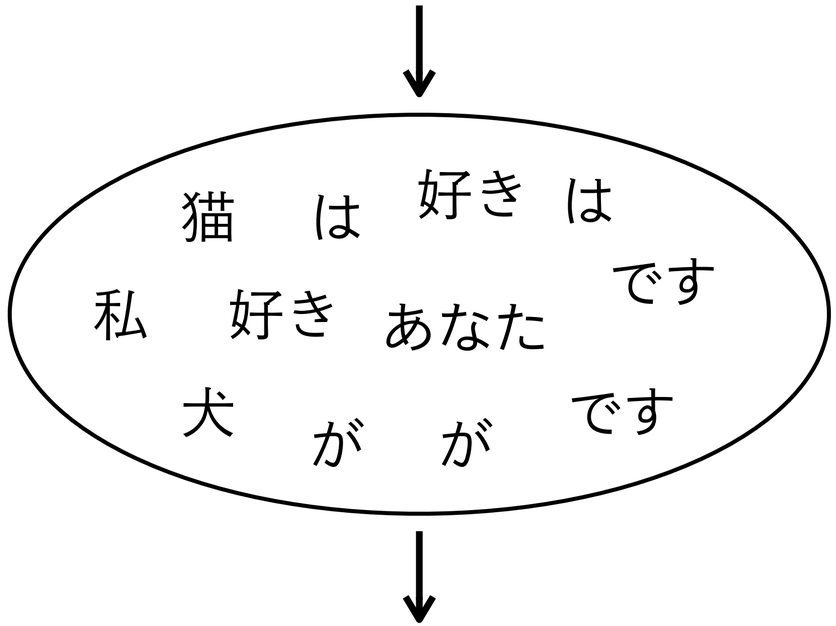
文書	文の数	トークン数	名詞率	接続詞率	係り受け距離	
					最大	平均
坊っちゃん	100	985	0.298	0.003	24	2.276
吾輩は猫である	100	808	0.261	0.007	29	2.331
羅生門	100	1,300	0.279	0.015	66	3.037
蜘蛛の糸	100	1,432	0.251	0.013	55	3.068
細雪	72	927	0.319	0.004	53	2.623
卍	103	2,942	0.282	0.009	124	3.227

※青空文庫のテキストデータより計算

Bag-of-Words (BoW)

- テキストデータをトークンの出現頻度ベクトルで表現
  - テキストの構造は無視
  - 固定次元ベクトルで表現

私は猫が好きです。あなたは犬が好きです。



私	あなた	…	猫	犬	鳥	…	好き	嫌い
1	1	…	1	1	0	…	2	0

TF-IDF

- トークンの出現頻度に基づいたテキストデータの特徴付け手法
- 仮定：以下のようなトークンはテキストデータの特定に重要
  - あるテキストデータにおける出現頻度が高い（TF）
  - 他のテキストデータにはあまり出現しない（IDF）

TF-IDFの定義

$$\text{TF-IDF}(D, d, t) = \text{TF}(d, t) \times \text{IDF}(D, t)$$

$\text{TF}(d, t)$: テキストデータ d におけるトークン t の出現頻度（BoW）

$$\text{TF}(d, t) = \frac{n_{d,t}}{\sum_{t'} n_{d,t'}}$$

$\text{IDF}(D, t)$: テキストデータ群 D におけるトークン t の出現頻度（文書単位）

$$\text{IDF}(D, t) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

探索的データ分析

- EDA：Exploratory Data Analysis
- データの様子を把握するための予備的なデータ分析

［代表的な方法］

- ワードクラウド
- 主成分分析
- 共起ネットワーク

分析

- テキスト分析の代表的なタスク：テキスト分類・テキスト変換

［目的］

- テキストの分類や変換を行う。
- 分類や変換を行うモデルを獲得する。
- 分類や変換を通してテキストからの特徴を抽出する。

テキスト分類の代表的な手法

- 汎用的な手法（数値ベクトルに対しては多くの分類手法が利用可能）
  - SVM（Support Vector Machine）
  - 決定木 / Random Forest
  - ベイジアンネットワーク / 単純ベイズ推定
- テキスト処理向けの手法
  - BERT
  - HMM（Hidden Markov Model）

テキスト変換の代表的な手法

- 統計的手法
  - HMM（Hidden Markov Model）
- ニューラルネットワーク（深層学習）
  - Seq2seq
  - Transformer / GPT

［演習］テキスト分析をやってみる（後半）

［演習］テキスト分析の後半をやってみる

- ここからは Google Colaboratory で作業します。
- ノートブックの説明を見ながら解説します。
- ノートブックの指示に応じてこちらの資料に戻って参照します。

BoW：Bag-of-Words

- BoWの出力の見方

```
array([[1, 0, 0, 0, 0, 0, 0, 0,...1, 1, 1, 0, 0],
       [0, 1, 1, 1, 0, 0, 1, 0,...0, 0, 1, 1, 0],
       [1, 0, 0, 0, 0, 1, 0, 0,...0, 1, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 1, 0,...0, 0, 1, 1, 0],
       [1, 0, 0, 0, 0, 1, 0, 0,...0, 0, 0, 0, 1]])
```

0番目の文書

1番目の単語の頻度

TF-IDF

- TF-IDFの出力の見方

```
array([[ 0.29021887, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
                          .....
         0.        , 0.34962323, 0.24414126, 0.        , 0.        ],

       [ 0.        , 0.34780952, 0.34780952, 0.28061053, 0.        ,
         0.        , 0.28061053, 0.28061053, 0.        , 0.28061053,
                          .....
         0.        , 0.        , 0.19594981, 0.28061053, 0.        ],
                          ...
         0.        , 0.        , 0.        , 0.        , 0.46482379]])
```

1番目の文書

0番目の単語のTF-IDF

WordCloud

- WordCloudオブジェクト作成の際のオプション指定

```
wc = WordCloud(
    font_path="/usr/share/fonts/opentype/noto/NotoSansCJK-Regular.ttc",
    background_color="white",
    width=1600, height=900,
    mask = mask_image)
```

日本語フォントファイル

画像の大きさ

マスク

- WordCloudの生成

```
stopwords = ["ある", "する", "いる", "ホームズ"]
words = wc_data.loc[0].to_dict()
for sw in stopwords:
  words.pop(sw, None)
img = wordcloud.fit_words(words)
plt.imshow(img)
plt.axis("off")
```

ストップワード（Word Cloudに含めない単語）

1番目のテキストデータ（赤毛連盟）のWordCloud
（0が1番目，1が2番目…となる点に注意）

主成分分析

- 主成分空間でのテキストデータのプロット

```
import matplotlib.pyplot as plt
import japanize_matplotlib

name = ["赤毛連合", "空き家の冒険", "踊る人形", "まだらの紐", "ボヘミアの醜聞", "最後の事件"]
for i in range(len(name)):
  plt.scatter(x[i, 0], x[i, 1], label=name[i])
  plt.text(x[i, 0], x[i, 1], name[i])
```

この数と順番が「4. ベクトル化」の「4-1. テキストデータの結合」でまとめた
テキストデータの数と順番に一致すること。