自然言語処理

目　次

教科書更新に伴う補足

講義リリース時点では Yahoo!ショッピング「商品レビュー検索API」というサービスがありました。

しかし、当サービスが2021年9月30日をもって終了となったため、急遽、教科書のアップデートを実施しました。

教科書は更新したものの、講義ビデオは更新しておりません。影響は以下の通りとなります。ご留意ください。

① 「データ取得」の章についてはビデオと教科書の内容が異なる。

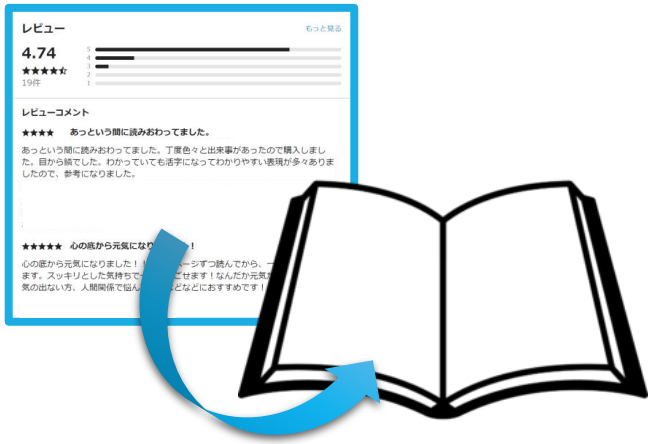
② 教科書P19以降、ビデオと教科書のページ数が異なる。

1．この章で扱うケース

出版社での企画検討

あなたは出版社に勤めているデータ解析者です。現在、あなたの会社では自己啓発系の本の企画をしていますが、既に世の中に数多くの書籍が存在しており、むやみに本を作ってもヒットさせるのは難しい状況です。

ある程度、売れる見込みがあるものを企画していきたいということで、
インターネットサイト（今回はYahoo!ショッピングを選定）の
レビューを見て、人気のある本の傾向をつかむ事にしました。
しかし、無数のレビューがあるため、
なかなか全容がつかみにくい…
という事がわかりました。



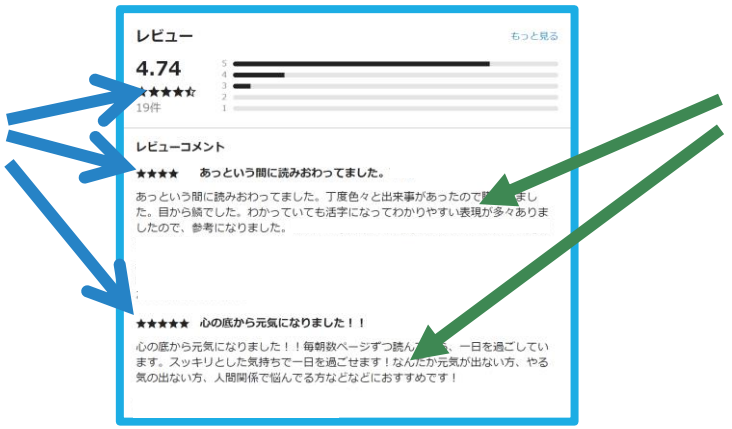
そこでデータ解析を実施し、何か
できないか？という事になりました。

2．自然言語処理概要

ケースに対応するために：自然言語処理とは

本ケースにおいて、対象となるデータはインターネットサイトの口コミデータとなります。今まで学んできた分析との大きな違いは、その対象が量的データ＋通常の質的データではなく、人間が普段使っている文章（自然言語データ）であるということです。

「星の数」は
量的データ



「コメント」は自然言語

自然言語データを機械（コンピュータ）で処理するための技術を一般に自然言語処理といいます。自然言語処理を行う目的は、概ね「自然言語データを処理して、有益な情報/示唆を導出すること」となります。

通常のデータ解析のプロセス

誤解を恐れず「通常の」という言葉を使いますが、通常のデータ解析では
対象データは「量的データ、数値データ」「質的データ、カテゴリデータ」
の２種類に分かれます。
質的データはダミー変数化などを行い、全てを量的に扱えるデータに
直してから利用します。

氏名	性別
佐藤太郎	男性
鈴木花子	女性
高橋幸子	女性

性別をダミー変数化

性別_女	性別_男
0	1
1	0
1	0

細かい事を一旦無視すると、分析は以下のプロセスで分析が進みます。

自然言語データの数値化について

自然言語データは質的データ・名義尺度の一種として捉える事も可能ですが、
「純粋な」名義尺度にはない特徴があります。**１データ内に要素が存在する**事です。
純粋な名義尺度と同じ方法（ダミー変数化）で数値化しても、嬉しくありません。
コメント間の関係性が現れないからです。（この状態を「互いに直交してしまう」と言います。）

コメント
イケメンだけど冷たい
イケメンで優しい
ブサイクなのに冷たい

コメントを
ダミー変数化

イケメンだけど冷たい	イケメンで優しい	ブサイクなのに冷たい
1	0	0
0	1	0
0	0	1

要素に配慮して、以下のように数値化できるとコメント間の関連性が見えてきます。

コメント
イケメンだけど冷たい
イケメンで優しい
ブサイクなのに冷たい

コメントを
ダミー変数化

イケメン	ブサイク	優しい	冷たい
1	0	0	1
1	0	1	0
0	1	0	1

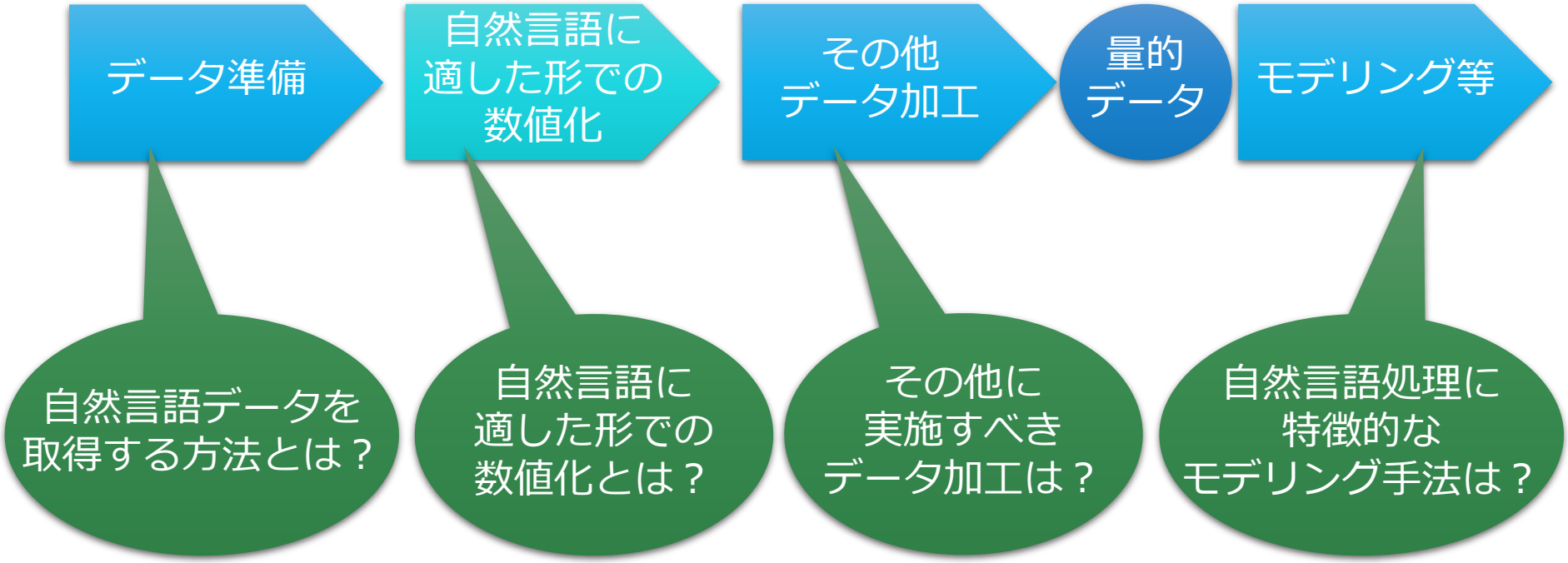
通常のデータ解析 / 自然言語処理のプロセスの違い

通常のデータ解析でも自然言語処理でも、まずは質的データを量的データに加工しなければ使えないという意味においては共通しています。
そういう意味では、データ解析のプロセスも共通していると言えます。

ただ、前頁の通り、単純に「質的データのダミー変数化」をしても有意義な数値化ができない事がわかっています。すなわち、「**データ間の関係性に関する情報を失わない形での量的データへの変換**」が望まれます。
そこが大きな特徴であり、通常のデータ解析との違いになっています。

ケースに対応するために学ぶべき事

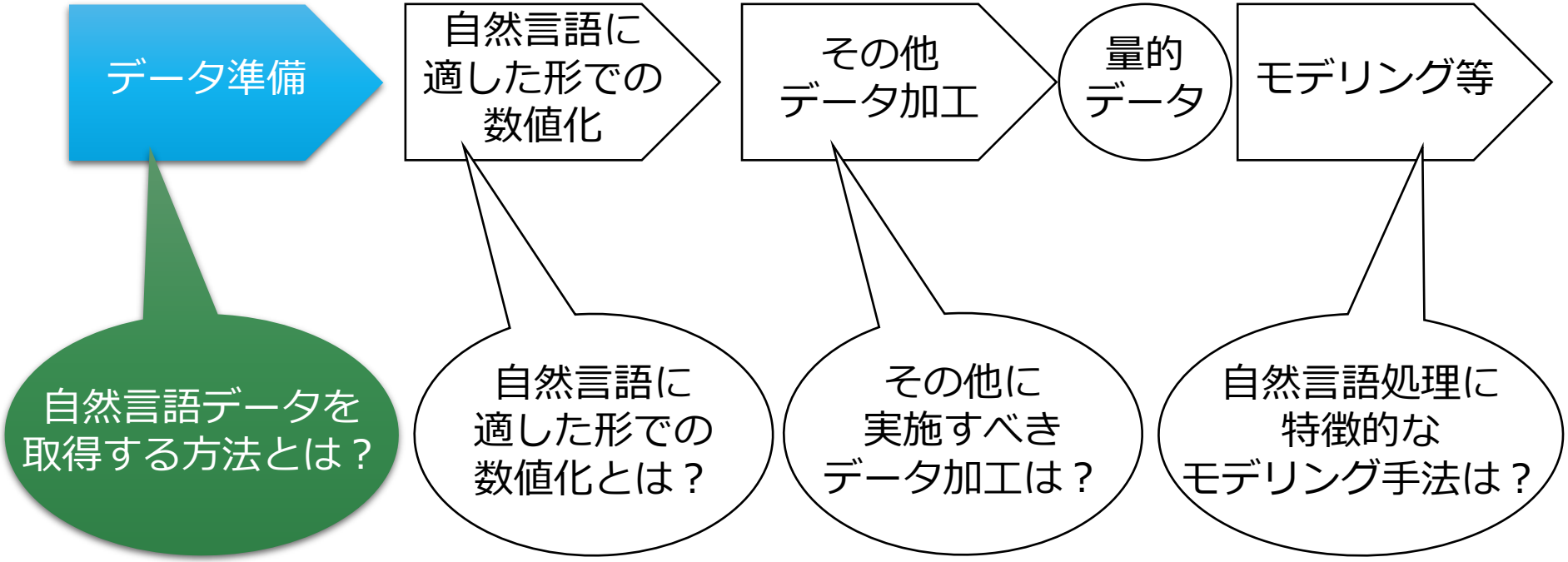
前頁までに学んだ事に配慮すると、本ケースに対応するためには、
以下の事を学ばなければならない…という事がわかります。



3．データ取得

ケースに対応するために、この章で学ぶべき事

この章では、データの取得方法について学びましょう。（あまり本質的ではありませんが）Yahoo!ショッピングからのデータ取得方法も学びます。



自然言語処理におけるデータの準備

自然言語データは、世の中のいろんな場所に存在します。

- 企業が保有しているデータ（例：コールセンターの応対履歴）、
- ＥＣサイト等の口コミデータ
- ＳＮＳへの投稿
- 国立情報学研究所のデータレポジトリ（研究用にしか使えない）
  https://www.nii.ac.jp/dsc/idr/

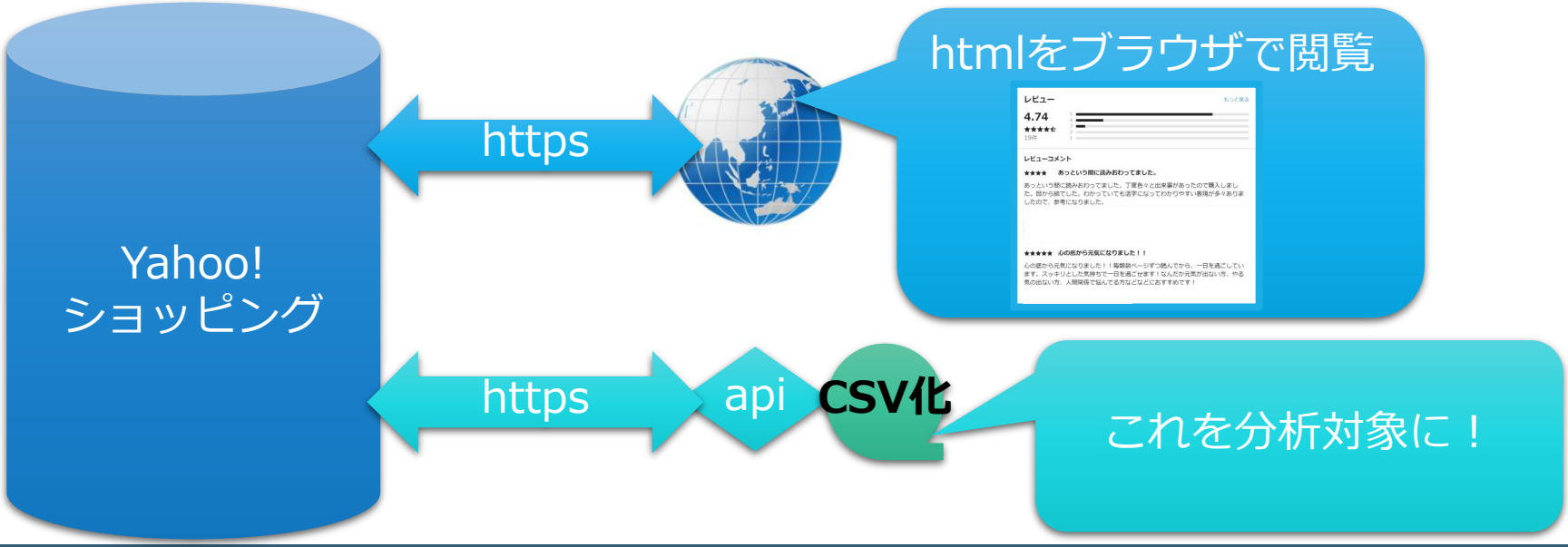
※ 尚、外部データを利用する際は著作権法、及び各サイトの利用規程に注意しましょう。

取得方法も様々です。

- 既にデータベースに格納されているデータをSQLで取得
- Web上にHTML形式で公開されているデータをスクレイピングで取得
- Web-APIが用意されていて各種形式（CSV、json等）で取得

Yahoo!ショッピングの口コミデータ

今回は Yahoo!ショッピング の口コミデータを分析を進めます。
元々 Yahoo!ショッピング は開発者向けに「商品レビュー検索API」という API を公開していました（2021年9月30日にAPIサービス終了）。
サービス終了までは API を利用する事で、JSON形式にて口コミデータを取得できました。今回はサービス提供時に取得したデータをCSV化したものを分析対象としていただきます。



Yahoo!ショッピング データについて

以下のデータを連携します。

1. categoryMaster.csv

- ✓ カテゴリ大分類ID（category_L_id）：カテゴリ大分類のID
- ✓ カテゴリ大分類名（category_L_name）：カテゴリ大分類の名称
- ✓ カテゴリ中分類ID（category_M_id）：カテゴリ中分類のID
- ✓ カテゴリ中分類名（category_M_name）：カテゴリ中分類の名称
- ✓ カテゴリ小分類ID（category_S_id）：カテゴリ小分類のID
- ✓ カテゴリ小分類名（category_S_name）：カテゴリ小分類の名称
- ✓ レビュー数（reviewCount）：reviews.csv の中に何件のレビューが格納されているかを記録（最大1,000件）

1. reviews.csv

- ✓ カテゴリ大分類ID（category_L_id）：カテゴリ大分類のID
- ✓ カテゴリ大分類名（category_L_name）：カテゴリ大分類の名称
- ✓ カテゴリ中分類ID（category_M_id）：カテゴリ中分類のID
- ✓ カテゴリ中分類名（category_M_name）：カテゴリ中分類の名称
- ✓ カテゴリ小分類ID（category_S_id）：カテゴリ小分類のID
- ✓ カテゴリ小分類名（category_S_name）：カテゴリ小分類の名称
- ✓ 評価（rate）：☆1～☆5 の評価値
- ✓ 商品名（name）：ショッピングサイト内で表示される商品名
- ✓ 口コミ（description）：口コミデータ

CSVから調査対象の商品データを抽出する

CSVから調査対象のデータを抽出するためのサンプルコードを
NLP_1_getDataFromReviews.ipynb として提供します。
このコードを確認しながら、必要なデータを取得してみてください。

ポイントは３つです。

① まず categoryMaster.csv を見て調査したいカテゴリを把握する
② ①の上で reviews.csv から対象カテゴリのデータを抽出する
③ （何度も大データをロードするのは大変なので）抽出後、保存する。

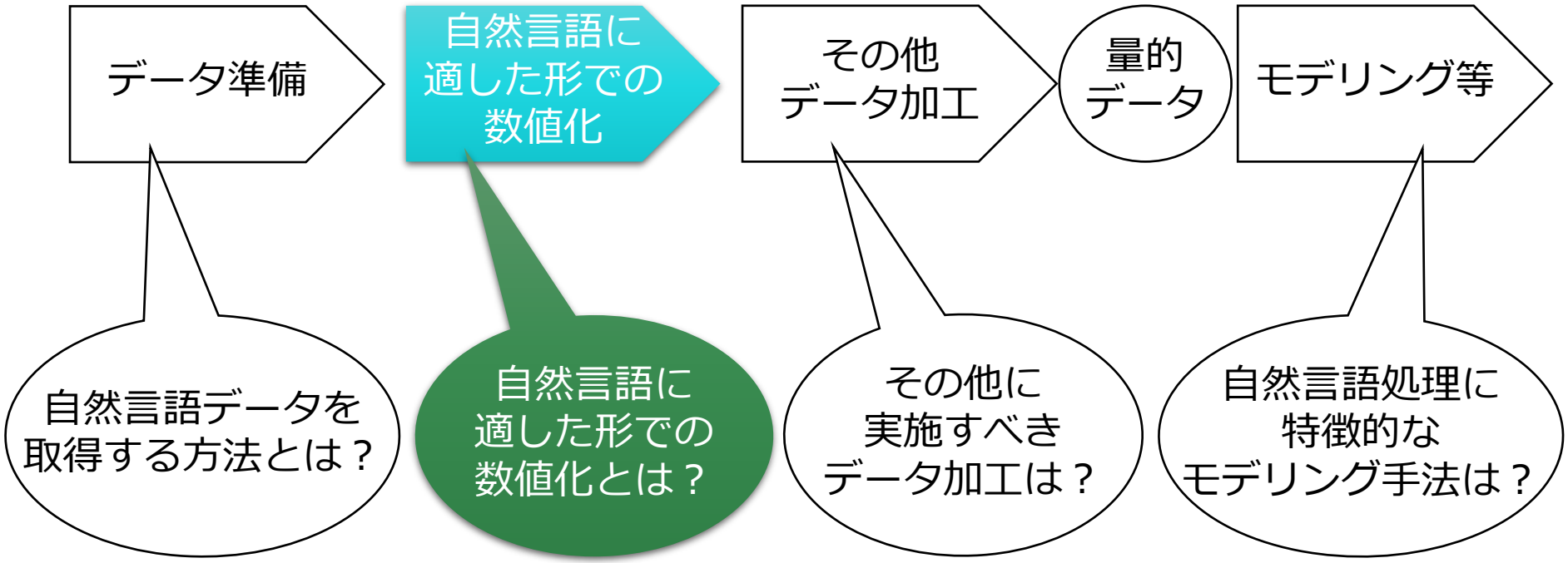
演習問題①：自分の好きな商品カテゴリについて収集

Yahoo!ショッピングから提供されている「商品レビュー検索API」を使い、自分の好きなカテゴリについてデータを集めてみましょう。

４．テキストデータの数値化①

ケースに対応するために、この章で学ぶべき事

この章では自然言語処理データの数値化について学びましょう。
特に日本語データに対応するための技法について学んでいく事にします。



自然言語データの数値化と要素分解

先立って説明した通り、自然言語データは要素分解が可能であり、コレに配慮せずそのままダミー変数化してもコメント間の関係性について全く配慮できないデータができあがってしまいます。要素に配慮する事で、この関係性を表現できます。

コメント
イケメンだけど冷たい
イケメンで優しい
ブサイクなのに冷たい

コメントを
ダミー変数化

イケメン	ブサイク	優しい	冷たい
1	0	0	1
1	0	1	0
0	1	0	1

これをコンピュータに実行させようとする場合、**「何をもって要素とするか？」**という課題に配慮する必要があります。次頁から分解と数値化について学びます。

例）「ブサイクなのに冷たい」を要素に分解するいくつかの方法

- 2-gram：ブサ　サイ　イク　クな　なの　のに　に冷　冷た　たい
- 3-gram：ブサイ　サイク　イクな　クなの　なのに　のに冷　に冷た　冷たい
- 形態素解析：ブサイク　なのに　冷たい

N-gram と Bag of Words（BoW） による数値化

自然言語データを、**連続するN文字（あるいはN単語）を N-gram** といい
これを要素とした集合に分ける事が可能です。
①ブサイクなのに冷たい、②ブサイクだけど優しい は文字 2-gram を
つかって以下のように要素分解できます。

① ： ブサ　サイ　イク　クな　なの　のに　に冷　冷た　たい
② ： ブサ　サイ　イク　クだ　だけ　けど　ど優　優し　しい

要素化してから、要素の出現回数をカウントすると以下のように表現できます。

番号	ブサ	サイ	イク	クな	なの	のに	に冷	冷た	たい	クだ	だけ	けど	ど優	優し	しい
①	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
②	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1

2-gram のようなシンプルな分解でも共通要素が表現できていることが見てとれます。
この**出現回数カウント表をBoW（Bag of Words）** と言います。

N-gram と BoW の作成例

NLP_2_N-gram_try.ipynb というサンプルプログラムにて、
文章を文字についての 1-gram、2-gram、3-gram で要素分解した上で、
BoW を作成してみる処理に挑戦してみて下さい。

ユーザー定義関数 to_n_gram、make_BoW を実装すれば可能です。

【サンプル・レビュー文章】（解析対象）

"気づけばもうこんな時間…そんなことを無くしたい。
コロナ禍にあって、もっと有意義な時間の使い方を
したい、そう思いこの本を購入しました。"

"コロナ禍で皆それぞれに様々な気付きがあった
と思う。雇用される脆さ、自分の必要性、組織の
妥当性…。今後も繰り返されるであろうパンデミック
の中でも生き残る答えが散りばめられている。"

"とても為になり、面白かったです。ゴリラの表紙では
少し激しい感じの本かと思いますが、とても有意義で
今後の自分に生かせる答えがもらえた本でした。"

※ 答えは次のページにて…

N-gram と BoW の作成例

NLP_2_N-gram.ipynb というサンプルプログラムにて、
文章を文字についての 1-gram、2-gram、3-gram で要素分解した上で、
BoW を作成してみる例を示してみました。

ポイントはユーザー定義関数 to_n_gram、make_BoW と、適用結果です。

```
def to_n_gram(text, n):
  # 返り値用のリストを用意する
  ret_list = []
  # 開始位置を１文字ずつ動かしながら、n で指定した文字分を抜き出してリストに追加
  for i in range(len(text)-n+1):
    ret_list.append( text[i:i+n] )
  return ret_list
```

← **N-gram で分割するためのアルゴリズム**

```
# N-gramデータについて前から1つずつずらして検索をして、
# キーワードと一致するものをカウントする関数を作成
def make_BoW( text, keyword ):
  # 返り値の初期値（リストの中に単語が無ければゼロ）を設定する
  ret_num = 0
  # N-gram化された文字が、keyword と一致していれば 1 を加える
  for word in text:
    if word == keyword:
      ret_num += 1
  return ret_num
```

← **BoW を作成するためのアルゴリズム**

【サンプル・レビュー文章】（解析対象）

"気づけばもうこんな時間…そんなことを無くしたい。コロナ禍にあって、もっと有意義な時間の使い方をしたい、そう思いこの本を購入しました。"

"コロナ禍で皆それぞれに様々な気付きがあったと思う。雇用される脆さ、自分の必要性、組織の妥当性…。今後も繰り返されるであろうパンデミックの中でも生き残る答えが散りばめられている。"

"とても為になり、面白かったです。ゴリラの表紙では少し激しい感じの本かと思いますが、とても有意義で今後の自分に生かせる答えがもらえた本でした。"

5．テキストデータの数値化②

N-gram の難点

N-gram は自然言語データを機械的に分解できる強力な方法ではありますが、必ずしも**共通要素の数と、意味上の類似度が一緒にならない**という弱点もあります（共通要素の数が単語長に依存する）。以下の例を見てみましょう。

例：①ブサイクなのに冷たい、②ブサイクだけど優しい、③ハンサムなのに優しい

①：ブサ　サイ　イク　クな　なの　のに　に冷　冷た　たい

②：ブサ　サイ　イク　クだ　だけ　けど　ど優　優し　しい

③：ハン　ンサ　サム　ムな　なの　のに　に優　優し　しい

番号	ブサ	サイ	イク	クな	なの	のに	に冷	冷た	たい	クだ	だけ	けど	ど優	優し	しい	ハン	ンサ	サム	ムな	に優
①	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
②	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
③	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1

①-②の共通要素の数は３つ、①-③の共通要素の数は２つ、②-③の共通要素の数は２つです。

しかし意味を考えると①-②が、①-③、②-③と比べて類似度が高いとは言いきれません。

形態素解析

自然言語データを意味の最小単位（単語、品詞）＝形態素に分解し、形態素１つ１つについて品詞など文法的な意味を判別していく処理を形態素解析といいます。要は、**形態素解析＝意味の最小単位への分解**です。

前頁の例について形態素解析⇒BoWの作成を行い、N-gramとの違いを確認してみましょう。

①：ブサイク　な（だ）　のに　冷たい

②：ブサイク　だ　けど　優しい

③：ハンサム　な（だ）　のに　優しい

番号	ブサイク	ハンサム	だ	のに	冷たい	優しい
①	1	0	1	1	1	0
②	1	0	1	0	0	1
③	0	1	1	1	0	1

①-②、②-③、①-③ともに共通要素の数は２つであり、それぞれ類似度は同じとなります。

（類似度が同じである事が「正」とも言い切れませんが、差をつける有意な理由がない以上、異なる事には違和感があります。）

形態素解析器による処理実行のポイント

形態素解析は形態素解析器と呼ばれるソフトウェアを用いて実行します。形態素解析器が実行してくれる処理について、有名な例である「すももももももものうち」を通して学びましょう。大きく、以下の通りです。

① **辞書をつかって文章を単語に分割**する（可能な分割を全て列挙）。

1. すもも（名詞）/も（助詞）/もも（名詞）/も（助詞）/もも（名詞）/の（助詞）/うち（名詞）
2. すもも（名詞）/も（助詞）/もも（名詞）/もも（名詞）/も（助詞）/ の（助詞）/うち（名詞）
3. すもも（名詞）/もも（名詞）/も（助詞）/ もも（名詞）/も（助詞）/の（助詞）/うち（名詞）

② 候補の中から**最も日本語らしい分割方法を選択**する。

具体的には、前後関係の発生確率 / 単語の出現確率をモデル化し（言語モデルと呼ばれる）もっとも発生しやすそうな分割方法を選択してくれる。（上記なら１）

形態素解析の性能を決めるのは「利用する辞書」「言語モデル」である事を理解しておきましょう。

形態素解析器、辞書のいろいろ

日本語は形態素間に区切りが無い言語なので、形態素解析は難しく、
そのため先人が多くの形態素解析器を残してきました。

- **MeCab**：解析速度が速い。
- **Janome**：Pythonで実装された解析器。
- **JUMAN**：歴史ある形態素解析器。

形態素解析の際に使う辞書も、様々な用途向けに整備されています。

- **IPA辞書**：小学校～高校で習うものに似た品詞体系をしています。
  MeCab、janome は標準で IPA 辞書を採用しています。
- **Unidic**（ユニディク）：国立国語研究所が作成している辞書。
  検索システム向けの分析を行う際に有用。
- **NEologd**（ネオログディー）：新語に強い巨大な辞書です。
  Twitter等、比較的、新しい言葉が
  頻出するものを対象にする際に有用。

形態素解析の実行例

NLP_3_MorphologicalAnalysis_IPADic.ipynb というサンプルにて形態素解析の実行例を見てみましょう。形態素解析器には janome を用いています。辞書は IPA辞書 を用います。

ポイントは以下の通りです。

```
# janome ライブラリのインストール
!pip install janome==0.4.1
import pandas as pd
from janome.tokenizer import Tokenizer
```

最初に必要なモジュールをインストール

形態素解析を実行するプログラム

```
# IPADic を辞書として用いる
t=Tokenizer()

# janome を使って形態素解析を行う関数を実装する
def doMorphologicalAnalysis(text, tokenizer):
    t_list=[]
    for token in tokenizer.tokenize(text):
      res = []
      # 表層形
      res = res + [token.surface]
      # 品詞 (＝品詞,品詞細分類1,品詞細分類2,品詞細分類3が出力される)
      res = res + [token.part_of_speech.split(",")[0]]
      res = res + [token.part_of_speech.split(",")[1]]
      res = res + [token.part_of_speech.split(",")[2]]
      res = res + [token.part_of_speech.split(",")[3]]
      # 活用型
      res = res + [token.infl_type]
      # 活用形
      res = res + [token.infl_form]
      # 基本形
      res = res + [token.base_form]
      # 読み
      res = res + [token.reading]
      # 発音
      res = res + [token.phonetic]
      print(res)

dataset["description"].apply( doMorphologicalAnalysis, tokenizer=t )
```

形態素毎に品詞に関する情報が並ぶ。

【出力内容】
表層形、品詞、品詞細分類1、
品詞細分類2、品詞細分類3、
活用型、活用形、原形、読み、発音

```
['まじ', '名詞', '形容動詞語幹', '*', '*', '*', '*', 'まじ', 'マジ', 'マジ']
['卍', '名詞', '一般', '*', '*', '*', '*', '卍', 'マンジ', 'マンジ']
['あざ', '名詞', '一般', '*', '*', '*', '*', 'あざ', 'アザ', 'アザ']
['まる', '接頭詞', '数接続', '*', '*', '*', '*', 'まる', 'マル', 'マル']
['水産', '名詞', '一般', '*', '*', '*', '*', '水産', 'スイサン', 'スイサン']
['気づけ', '動詞', '自立', '*', '*', '五段・カ行イ音便', '仮定形', '気づく', 'キヅケ', 'キズケ']
['ば', '助詞', '接続助詞', '*', '*', '*', '*', 'ば', 'バ', 'バ']
['もう', '副詞', '一般', '*', '*', '*', '*', 'もう', 'モウ', 'モー']
['こんな', '連体詞', '*', '*', '*', '*', '*', 'こんな', 'コンナ', 'コンナ']
['時間', '名詞', '副詞可能', '*', '*', '*', '*', '時間', 'ジカン', 'ジカン']
['…', '記号', '一般', '*', '*', '*', '*', '…', '…', '…']
```

形態素解析の結果が表示

形態素解析を行う上での工夫

形態素解析の性能を決めるのは辞書とモデルである事は前述の通りです。
ここで **NLP_4_MorphologicalAnalysis_NEologd.ipynb** というサンプルを用意しました。こちらは前頁の例と同様、形態素解析器には janome を用いていますが、辞書には NEologd を用いるようにしました。

実行方法の違いはサンプルをご確認いただきたいですが、ここで重要なのは **辞書を変える事で出現する単語が変わる** という事です。

IPA辞書

```
['まじ', '名詞', '形容動詞語幹', '*', '*', '*', '*', 'まじ', 'マジ', 'マジ']
['卍', '名詞', '一般', '*', '*', '*', '*', '卍', 'マンジ', 'マンジ']
['あざ', '名詞', '一般', '*', '*', '*', '*', 'あざ', 'アザ', 'アザ']
['まる', '接頭詞', '数接続', '*', '*', '*', '*', 'まる', 'マル', 'マル']
['水産', '名詞', '一般', '*', '*', '*', '*', '水産', 'スイサン', 'スイサン']
['気づけ', '動詞', '自立', '*', '*', '五段・カ行イ音便', '仮定形', '気づく', 'キヅケ', 'キズケ']
['ば', '助詞', '接続助詞', '*', '*', '*', '*', 'ば', 'バ', 'バ']
['もう', '副詞', '一般', '*', '*', '*', '*', 'もう', 'モウ', 'モー']
['こんな', '連体詞', '*', '*', '*', '*', '*', 'こんな', 'コンナ', 'コンナ']
['時間', '名詞', '副詞可能', '*', '*', '*', '*', '時間', 'ジカン', 'ジカン']
['…', '記号', '一般', '*', '*', '*', '*', '…', '…', '…']
['そんな', '連体詞', '*', '*', '*', '*', '*', 'そんな', 'ソンナ', 'ソンナ']
['こと', '名詞', '非自立', '一般', '*', '*', '*', 'こと', 'コト', 'コト']
['を', '助詞', '格助詞', '一般', '*', '*', '*', 'を', 'ヲ', 'ヲ']
['無くし', '動詞', '自立', '*', '*', '五段・サ行', '連用形', '無くす', 'ナクシ', 'ナクシ']
['たい', '助動詞', '*', '*', '*', '特殊・タイ', '基本形', 'たい', 'タイ', 'タイ']
['。', '記号', '句点', '*', '*', '*', '*', '。', '。', '。']
```

NEologd

```
['まじ卍', '名詞', '固有名詞', '一般', '*', '*', '*', 'まじ卍', 'マジマンジ', 'マジマンジ']
['あざまる水産', '名詞', '固有名詞', '一般', '*', '*', '*', 'あざまる水産', 'アザマルスイサン', 'アザマルスイサン']
['気づけ', '動詞', '自立', '*', '*', '五段・カ行イ音便', '仮定形', '気づく', 'キヅケ', 'キズケ']
['ば', '助詞', '接続助詞', '*', '*', '*', '*', 'ば', 'バ', 'バ']
['もうこん', '名詞', '一般', '*', '*', '*', '*', '毛根', 'モウコン', 'モーコン']
['な', '助動詞', '*', '*', '*', '特殊・ダ', '体言接続', 'だ', 'ナ', 'ナ']
['時間', '名詞', '副詞可能', '*', '*', '*', '*', '時間', 'ジカン', 'ジカン']
['…', '記号', '一般', '*', '*', '*', '*', '…', '…', '…']
['そんな', '連体詞', '*', '*', '*', '*', '*', 'そんな', 'ソンナ', 'ソンナ']
['こと', '名詞', '非自立', '一般', '*', '*', '*', 'こと', 'コト', 'コト']
['を', '助詞', '格助詞', '一般', '*', '*', '*', 'を', 'ヲ', 'ヲ']
['無くし', '動詞', '自立', '*', '*', '五段・サ行', '連用形', '無くす', 'ナクシ', 'ナクシ']
['たい', '助動詞', '*', '*', '*', '特殊・タイ', '基本形', 'たい', 'タイ', 'タイ']
['。', '記号', '句点', '*', '*', '*', '*', '。', '。', '。']
```

IPA辞書は「まじ卍」を「まじ」「卍」に分解。
IPA辞書は「あざまる水産」を「あざ」「まる」「水産」に分解。
NEologdは「もうこんな」を「毛根」「な」として解釈。

自分で単語登録すれば出力を変えられる 事も覚えておきましょう。

※ 登録方法は NLP_3_MorphologicalAnalysis_IPADic.ipynb

6．テキストデータの整備

自然言語処理データを「より良く」量的データ化するために

前頁までにお伝えした自然言語データを量的データにするための流れをまとめますと、以下のようになります。

少なくとも上記の２ステップがあれば、自然言語データを量的データに変換する事ができます。しかし、その後の分析で良い結果を出そうとするならば、**量的データにするまで＋した後で工夫を加える必要**があります。

次のページからは、量的データにするまでに可能な工夫について紹介します。

自然言語データを扱う際にできる工夫

前頁の通り、大きく分けると BoW化する前と後にできる工夫があります。

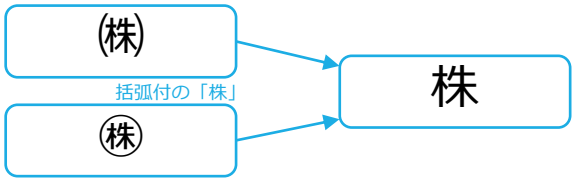
- BoW化前
  - 辞書の変更および単語登録（既に説明済）
  - 単語の正規化
    - ✓ 文字種の統一
    - ✓ 半角全角の統一
    - ✓ 大文字小文字の統一
    - ✓ 原型への統一
    - ✓ 間違いによる表記ゆれの統一
  - 数値表現の無意味化
  - 品詞 / 頻度カットオフ
  - リストによるストップワード除去

- BoW化後
  - TF-IDF変換

単語の正規化

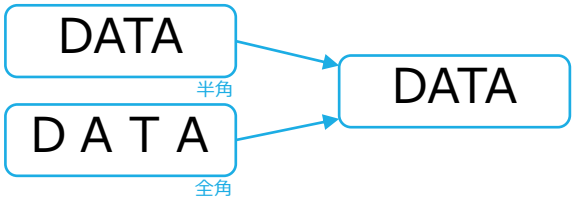
様々な原因によって発生する**単語の表記ゆれを解消**するために行う処理を単語の正規化と呼びます。正規化の対象となる表記ゆれには以下のようなものがあります。
統一方法としては辞書を使うもの、変換リストを作るものなどがあります。

文字種の統一



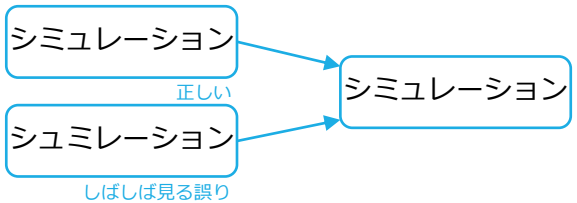
※ できれば辞書で統一したい

半角全角の統一



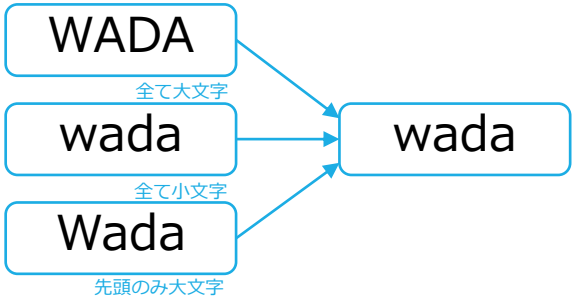
※ 変換リストで対応可能

間違いによる表記ゆれの統一



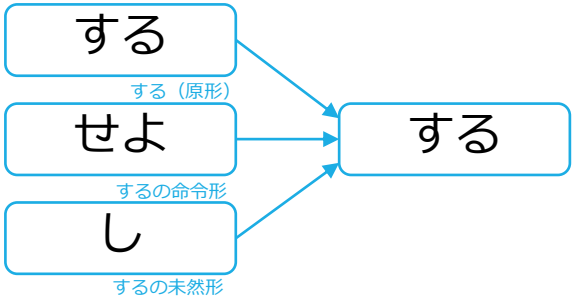
※ できれば辞書で統一したい

大文字小文字の統一



※ 変換リストで対応可能

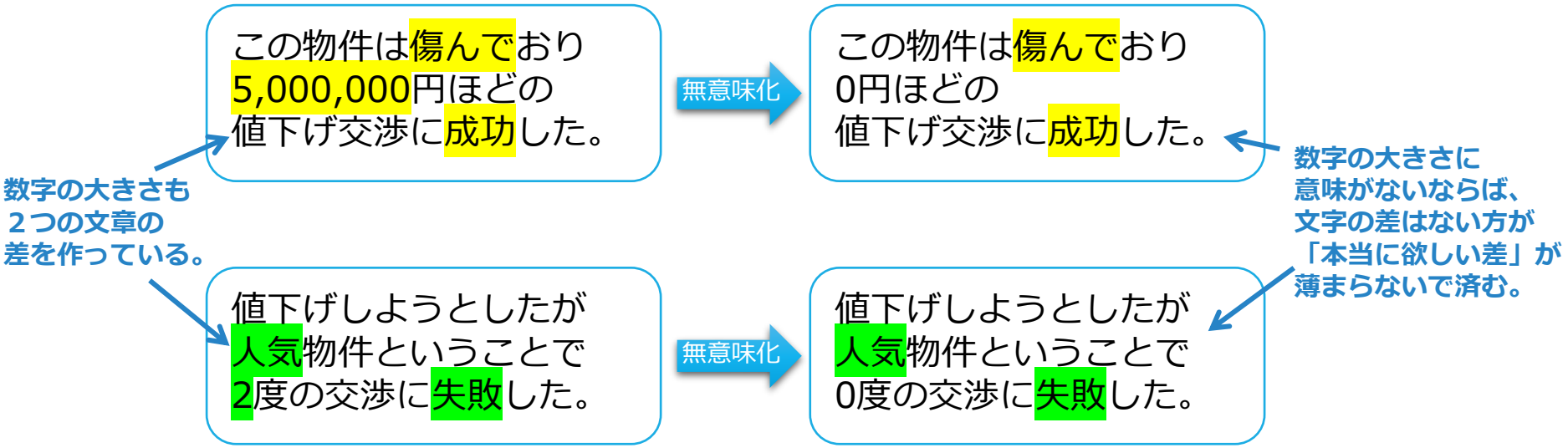
原型への統一



※ できれば辞書で統一したい

数値表現の無意味化

自然言語処理においては、しばしば文章中の具体的な数値に意味を
持たせても意義がないというケースが発生します。
このようなケースを防ぐため、**数値を全て 0 に変えてしまう（無意味化）**
を行うというテクニックがあります。



※ あるいは数字はストップワード（後述）にしてしまう…というテクニックもあります。

品詞 / 頻度カットオフ

あまりに一般的な言葉であるため文章の特徴となり得ない単語について、予め消してしまうというテクニックがあります。（ストップワード除去）

具体的には以下のような方法があります

- 品詞カットオフ
  - ✓ もちろん対象データにもよるのですが、**情報量の少ない品詞の単語は形態素解析の時点で捨ててしまう**という手があります。
  - ✓ 比較的情報量が多いと考えられている：名詞、動詞、形容詞、感動詞、連体詞
  - ✓ 比較的情報量が少ないと考えられている：副詞、助詞、助動詞、接続詞

- 頻度カットオフ
  - ✓ **出現頻度が高い単語（例：上位50位）、及び、出現頻度が少ない単語（例：40000位以下）はカットしてしまう**というのもアリです。

リストによるストップワード除去

前頁の通りですが、あまりに一般的な言葉であるため文章の特徴と
なり得ない単語について、予め消してしまうというテクニックを
ストップワード除去といいます。

「よくストップワードとして採用される単語」リストがWeb上で公開されています。Slothlib もその一つです。以下のサイトにリストがありますので、これをそのままストップワードとして採用してしまうのも手です。

http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt

実務上は前頁＋本頁のテクニックを組み合わせて使ったりします。
例えば「品詞カットオフをかけた上で、Slothlib の Japanese.txt を使い、ストップワード除去を実施する」などは有効です。

TF-IDF 変換 (TF:Term Frequency、IDF : Inverse Document Frequency)

「一般的な言葉であるため文章の特徴となり得ない」という発想の元、予め単語を消してしまうのがストップワード除去というテクニックでした。これをもう少し発展させて**「特定の文章にしか現れない単語に対して、重みをつけて量的データ化する」という手法が TF-IDF 変換**になります。TF-IDF は BoW から出発して、以下のように作成されます。

BoW

No	ラーメン	カレー	寿司
1	1	2	0
2	0	1	1
3	1	1	0

総単語数

計
3
2
2

TF = BoW ÷ 総単語数

No	ラーメン	カレー	寿司
1	0.3333	0.6667	0.0000
2	0.0000	0.5000	0.5000
3	0.5000	0.5000	0.0000

1÷2

出現頻度（1以上なら出現）

ラーメン	カレー	寿司
2	3	1

IDF = log(文章数÷出現頻度) ＋1

ラーメン	カレー	寿司
1.1761	1.0000	1.4771

log(3÷1)+1

TF-IDF = TF × IDF

No	ラーメン	カレー	寿司
1	0.3920	0.6667	0.0000
2	0.0000	0.5000	0.7386
3	0.5880	0.5000	0.0000

TF-IDF 変換 の意味

TF は 文章毎に合計値が1になるように正規化された出現頻度のことです。
IDF は出現のレア度を表す指標です。 すなわち TF-IDF は正規化された出現頻度に出現のレア度が掛かったもの＝「特定の文章にしか現れない単語に対して、重みをつけて量的データ化」したものとして解釈できます。

BoW

No	ラーメン	カレー	寿司
1	1	2	0
2	0	1	1
3	1	1	0

IDF：出現のレア度

ラーメン	カレー	寿司
1.1761	1.0000	1.4771

TF：出現頻度（文章毎に合計1になるよう正規化）

No	ラーメン	カレー	寿司
1	0.3333	0.6667	0.0000
2	0.0000	0.5000	0.5000
3	0.5000	0.5000	0.0000

TF-IDF：出現頻度×出現のレア度

No	ラーメン	カレー	寿司
1	0.3920	0.6667	0.0000
2	0.0000	0.5000	0.7386
3	0.5880	0.5000	0.0000

No=2 のカレーと寿司をみたとき、両者、出現回数は1回だが、寿司の方がレアなのでより大きい数字で評価されている。

TF-IDF 変換 のいろいろ

ひとくちにTF、IDFといっても計算方法にいくつかの選択肢があります。

- TF
  - BoWの値を構成比（＝L1ノルムで正規化）に変換して投入する事も可能です。
    - ✓ 前述までの方法はこれにあたります。
    - ✓ gensim.models.TfidfModel で計算する場合は、TF計算式を定義しwlocalオプションに入力、normalize オプションを Falseにする必要があります。
  - BoWの値（＝単純に出現頻度）をそのまま投入する事も可能です。（正規化しない）
    - ✓ この場合、TFの値が文章が長短にも依存してしまうので、本当にそれでいいのか？事前に考えておく必要があります。
    - ✓ gensim.models.TfidfModel で計算する場合は、TFの計算式を定義せずに、normalize オプションを False にすれば実行できます。
  - BoWの値を構成比（＝L2ノルムで正規化）に変換して投入する事も可能です。
    - ✓ gensim.models.TfidfModel で計算する場合は、何のオプションも指定せず入力すればOKです。

- IDF
  - LOG の底を低くすればするほど「滅多に出ないワード」が重要視されます。
    - ✓ だいたい 2 あたりから実行します（サンプルも2）が、1.5付近がいい…とする方もいらっしゃいます。

自然言語データを扱う際にできる工夫の実施例

NLP_5_TF-IDF_and_others.ipynb というサンプルを用意しました。
こちらはこの章で学んだテクニックを使った例になっております。

- 形態素解析（辞書としてNeologdを利用）
- 単語の正規化実施
- 品詞によるカットオフ
- slothlib の Japanese.TXT をつかったストップワード除去
- TF-IDF変換

学んだ内容について、具体的な実行例を参照してみましょう。

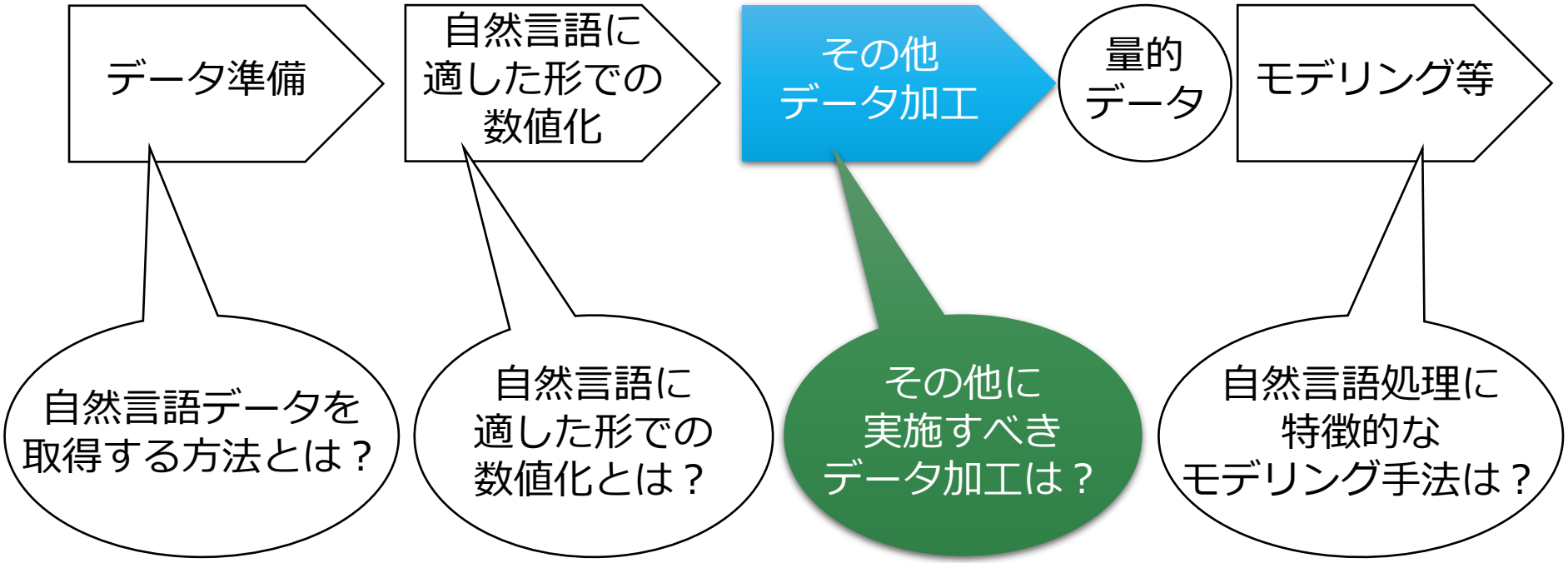
演習問題②：演習問題①で集めたデータを量的データ化

演習問題①で集めたデータを、この章で学んだテクニックを使って
量的データ化してみましょう。

7．データの俯瞰と加工

ケースに対応するために、この章で学ぶべき事

この章では自然言語データに特有のデータ加工について学びましょう。
自然言語処理では泥臭いデータ加工をする事で、得られる示唆の量が
決まってくると言えます。そのあたり、ポイントを学んでいきましょう。



データの俯瞰

テーブル型のデータならば、まずデータを前にしたときに、見える化（ヒストグラムを描いて分布を見る、散布図を描いて変数間の関係を把握する…等）や記述統計学に基づいた要約統計量の算出をおこなうでしょう。

しかしながら**自然言語データの場合、そのような俯瞰が容易でありません**。要約しようとする処理そのものが大きなバイアスになってしまい、本来、データが持つ情報を歪めてしまう可能性があります。そこで、自然言語データを前にしたとき、**実はまずやるべき事は「実際に生データを読んでみる」という事**になります。

（自分の主観がバイアスになってしまうリスクに注意を払いながら）実際に読んでみる事で、やるべき事が見えてくる可能性があります。

データを読む事で得られる情報

今回の口コミデータを読んでみましょう。

description
はじめて利用させていただきました。書店になく、こちらで購入させていただきましたが、梱包も丁寧で安心できるストアさんです。また機会があれば利用させていただこうと思います。
すぐ届きました。梱包も丁寧です。内容も興味深く参考になりそうです。
急いで購入する必要があり、慌てて買いました。発送や梱包は全く問題なくてよかったです。届けも素早く手配してくれました。本の商品自体も内容も良かったです。
普段はつい因果論で考えがちで、自分にとって都合の悪いこと、嫌なことが起こるとあの時こういうことを言われたからだ、とか考えてしまいますが、この本では真逆のことが書いてあります。この本の内容を素直に受け止めて、プラスになるように考える癖をつけたい。
この本は本来非常に長いもので読むに時間がかかり忙しい人にはなかなかハードルが高いけれど、図解されコンパクトに要点がまとめられ、重要な部分は繰り返し出てくるので、ナポレオン　ヒルの言わんとする事があますくことなく吸収できて便利です!?

実は本の書評ではなく、ストアの評価も入っている事がわかるでしょう。

これらのコメントは今回のケースには不要なコメントです（あくまでどの本が売れるか？その方向性が知りたい）。　読むことで「梱包」「発送」「届く」のようなキーワードが入っているコメントは削除するべきという事がわかります。

自然言語データを扱う際にできる工夫の実施例

NLP_6_preprocessing.ipynb というサンプルを用意しました。
こちらでは、自然言語データを実際に読むことで得た示唆（評価すべきでないデータが入っている）を基にデータを加工した結果が入っています。

NLP_5_TF-IDF_and_others.ipynb とほぼ同じ処理ですが、
最後のランキングの結果にどのような変化があったかを確認してみましょう。

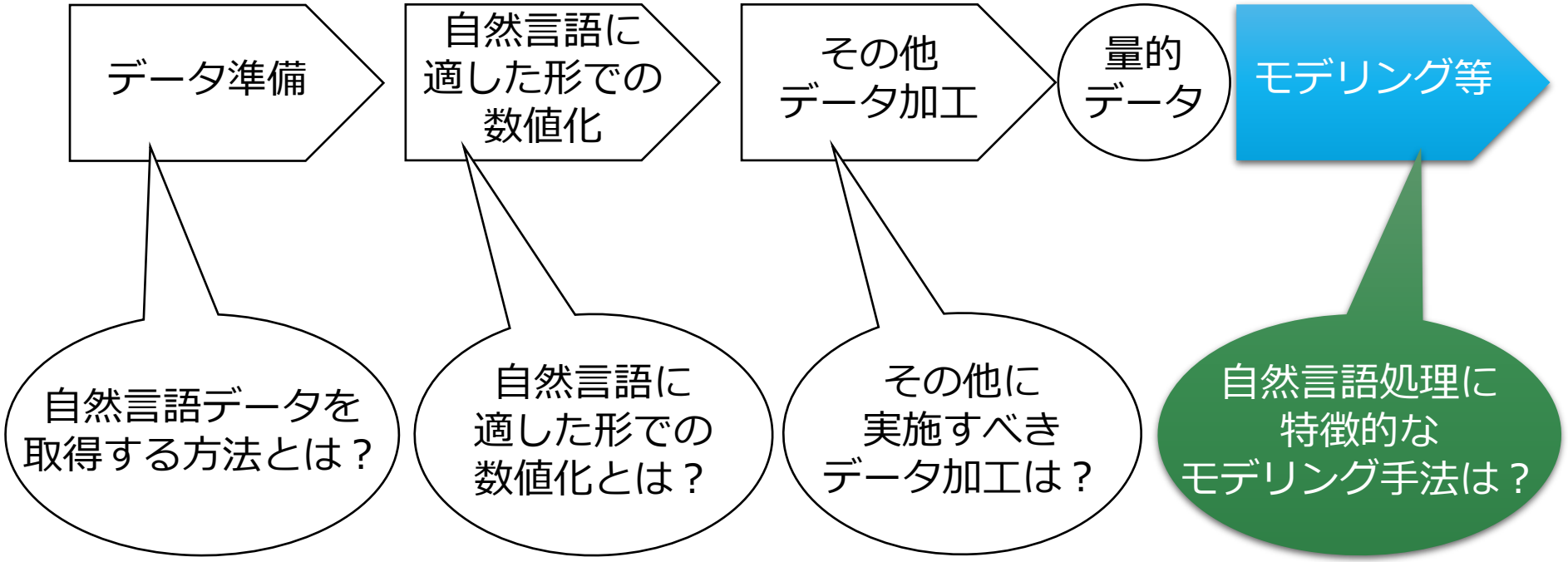
演習問題③：データを読んだうえでデータ加工を実施

演習問題①で集めたデータについて、実際にデータを読んでみて、
その上で必要なデータ加工も行い、最後に量的データ化してみましょう。

８．自然言語処理でよく使う分析手法

ケースに対応するために、この章で学ぶべき事

自然言語処理データを数値化すれば、通常の分析と同じような手法を適用する事も可能になってきます。それらは適宜、自身でやってもらう事とし、この章では自然言語処理において特徴的な分析手法について学びます。



自然言語処理でよく使う分析手法

以下について説明していきます。

- ✓ ランキング把握：**NLP_7_ranking.ipynb**
- ✓ 共起分析：**NLP_8_colocation.ipynb**
- ✓ LDA（Latent Dirichlet Allocation）：**NLP_9_LDA.ipynb**
- ✓ 簡単なポジネガ分析：**NLP_10_SimplePositiveNegativeAnalysis.ipynb**
- ✓ Word2Vec：**NLP_11_word2vec_wikipedia.ipynb**
- ✓ Doc2Vec：**NLP_12_doc2vec.ipynb**

8-1. 単語ランキング

単語ランキングによる概要把握

自然言語データを量的データ化（BoW or TF-IDF）してから、単語毎に値の合計値を計算し、ランキング化する事で概要を把握する事が可能です。
例えばケースの場合、評価毎にランキングをとると、傾向の違いを把握可能です。ランキングを見て頻度カットオフなどの調整をする事もできます。

評価毎ランキング（頻度カットオフ前）

	Rate-5	tfidf-5	Rate-4	tfidf-4	Rate-3	tfidf-3	Rate-2	tfidf-2	Rate-1	tfidf-1
0	する	42.768889	読む	18.384799	内容	7.842789	の	2.993562	レベル	2.958846
1	読む	34.619275	する	13.890785	する	6.534194	内容	2.162914	肝心	1.899968
2	本	33.679648	内容	13.777103	ある	6.438025	向き	1.973350	番号	1.899968
3	良い	33.471313	思う	11.959191	思う	6.308733	微妙	1.899968	使用済み	1.726681
4	ありがとう	32.268292	やすい	11.554421	読む	5.687886	暇つぶし	1.899968	タイトル	1.639636
5	なる	27.685661	良い	10.093003	良い	5.516022	ない	1.844760	使える	1.553394
6	いる	25.998987	いる	8.669203	なる	5.499814	する	1.841396	買う	1.546552
7	やすい	22.946879	ある	8.415704	本	4.954622	ある	1.786889	する	1.500741
8	購入	20.331083	なる	8.079136	参考	4.758366	読み切る	1.625315	いる	1.414377
9	思う	20.114297	本	7.573813	購入	4.633346	たいした	1.519974	知る	1.368658
10	ある	16.542560	購入	6.603911	こと	4.374030	いる	1.519205	残念	1.274913
11	*	16.102219	楽しみ	6.404972	の	4.324937	人	1.416238	う	1.266645
12	こと	15.276536	読める	6.291030	期待外れ	3.792401	読む	1.208186	誇大広告	1.266645
13	綺麗	14.885656	頂く	5.962998	いる	3.670653	期待	1.200588	催促	1.266645
14	できる	14.560432	せる	5.907118	役に立つ	3.551471	中古品	1.161622	レビュー	1.266645
15	参考	14.248105	できる	5.273696	ところ	3.526842	所	1.130792	つまらない	1.266645

共通する部分（黄色）をカットオフ

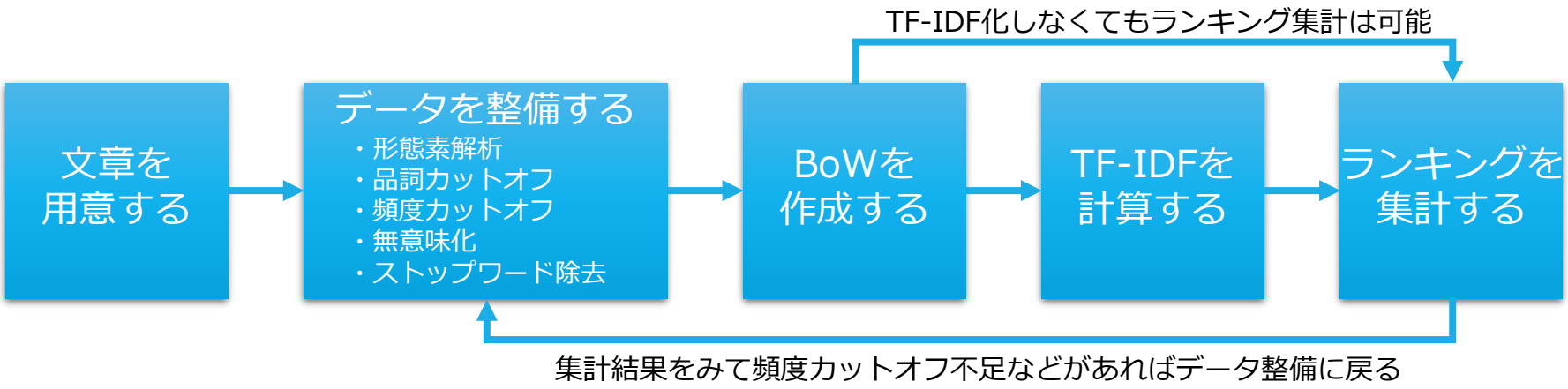
評価毎ランキング（頻度カットオフ後）

	Rate-5	tfidf-5	Rate-4	tfidf-4	Rate-3	tfidf-3	Rate-2	tfidf-2	Rate-1	tfidf-1
0	ありがとう	41.411666	やすい	18.709227	コメント	6.906723	たいした	3.799935	レベル	3.313907
1	やすい	34.711093	楽しみ	15.321398	期待外れ	6.501258	ない	3.653947	番号	2.533290
2	できる	20.736177	せる	13.202405	違う	6.379432	微妙	2.533290	肝心	2.533290
3	綺麗	20.375228	読める	8.589618	買う	5.683789	暇つぶし	2.533290	使用済み	2.302241
4	お願い	18.792743	勉強	8.064248	ところ	5.366678	論	2.302241	タイトル	1.942338
5	買う	18.196206	状態	7.748217	当たり前	5.088564	読み切る	2.167086	レビュー	1.899968
6	いい	18.173276	楽しむ	7.599870	役に立つ	4.992027	実行	2.167086	催促	1.899968
7	よい	17.192956	できる	7.337299	ない	4.388170	向き	1.973350	う	1.899968
8	中古	16.133563	この	6.651678	的	4.131168	中古品	1.936037	買う	1.767249
9	れる	16.091226	楽しい	6.646922	勉強	4.069168	所	1.884653	知る	1.621333
10	大変	16.075636	わかる	6.558636	られる	3.937219	共感	1.800882	誇大広告	1.519974
11	状態	15.895589	…。	6.501258	てる	3.822818	止まる	1.726681	残念	1.413490
12	商品	15.307908	ため	6.352268	インバスケット	3.799935	人	1.587755	評価	1.413490
13	この	14.830938	的	6.343707	トラブル	3.799935	0分	1.553394	見出し	1.266645
14	面白い	14.749036	よい	5.828955	練習	3.799935	受ける	1.553394	新聞	1.266645
15	勉強	14.728907	面白い	5.782171	想像	3.799935	刺激	1.553394	つまらない	1.266645
	さん	14.344776	早い	5.648888	やすい	3.768437	思える	1.534713	負け	1.266645
17	助かる	14.166354	実践	5.600708	れる	3.671819	読める	1.527491	最低	1.266645
18	満足	14.134803	そう	5.598921	有難う	3.494419	自殺率	1.519974	立ち読み	1.151121
19	みる	12.841385	文章	5.490974	終える	3.365687	スカスカ	1.519974	ない	1.037930
20	わかる	12.822439	商品	5.316708	できる	3.328044	失業率	1.519974	無駄	1.035596

評価3～5に特徴的な単語と
評価1～2に特徴的な単語の
違いを確認する。
（色付けしてみました）

単語ランキングによる概要把握するための手順

以下のプロセスでランキングを確認します。



NLP_7_ranking.ipynb を用意しましたのでご確認ください。

8-2. 共起分析

共起分析：共起 とJaccard係数

ある単語のペアが、ある文書の中で共に出現することを共起するといいます。

① 非常に長いもので、読むに時間がかかり、忙しい人にはなかなかハードルが高い。

② 長いものには巻かれろの言葉通り、権力側に従うことで成長してきた氏の考えがわかる。

③ ハードル走は単純な短距離走とは異なる魅力がある。

- 最初の文章では「長い」「ハードル」は共起している。
- 次の２つの文章「長い」「ハードル」は共起していない。

共起の発生度合いを示す指標として、しばしば**Jaccard 係数**が使われます。
例えば単語ペア A、B が存在するとき、A、B の Jaccard 係数 $J(A,B)$ は以下の式で計算されます。

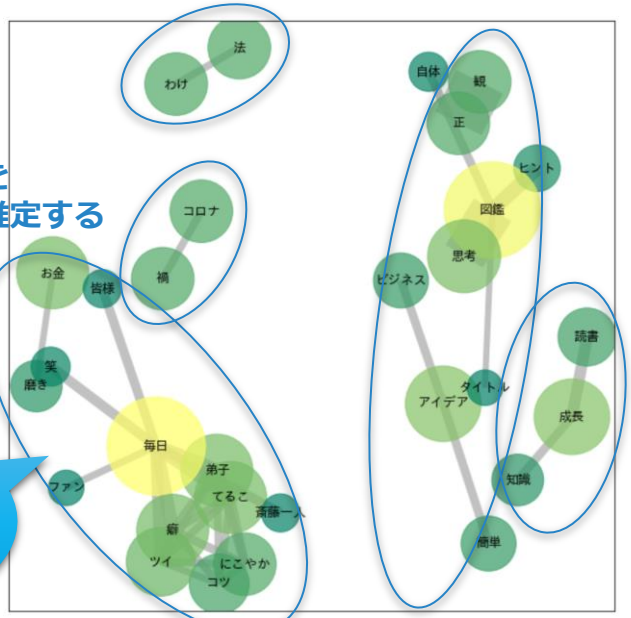
$$J(A,B) = \frac{A \cap B}{A \cup B} = \frac{A \cap B}{A + B - A \cap B} = \frac{A\text{とB}\text{が共起する文章数}}{A\text{が出現する文書数} + B\text{が出現する文書数} - A\text{とB}\text{が共起する文章数}}$$

例）前述の３文章において「長い」「ハードル」の Jaccard 係数は 1/3

共起分析：共起ネットワーク図

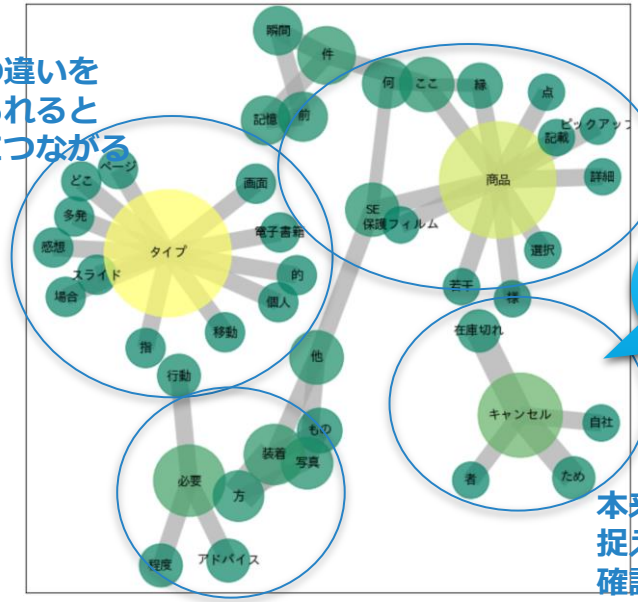
文章に発生する様々な単語のペアについて、出現回数・共起回数を調べ、Jaccard 係数を計算したうえで単語間の関係性を可視化したものが**共起ネットワーク図**です。　以下のような図を描くことができます。

星3～5の文書（左）、星1～2の文章（右）の共起ネットワーク



単語のカタマリを
とらえて意味を推定する

毎日コツコツ
笑顔を磨くことで
お金が貯まる？



左右の違いを
捉えられると
示唆につながる

在庫切れのため
キャンセル？
（書評じゃない？）

本来捉えたいものが
捉えられているかも
確認していく

※ Jaccard係数が大きい程、単語同士が近くに来ます。共起回数が多い程、単語を結ぶ線を太くしてあります。

※ ノードはページランク（後述）の大きさに従って色付けされています。（黄色が高ランク、緑が低ランク）

共起分析：ページランク

ページランクとは本来、Webページの重要度を測るために定義された指標。
前ページの場合はそれを**「単語の重要度」を測る指標**として転用している。

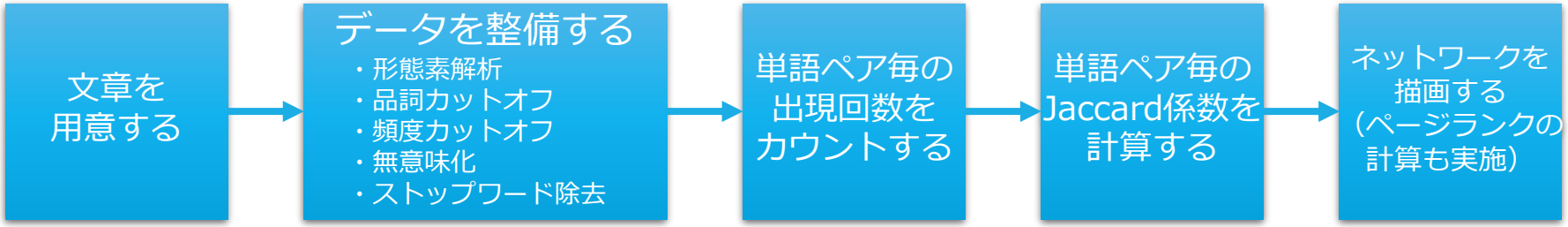
今回の場合、ある単語 A のランクは、A が単語ペアの中に含まれるほど、
また、A とペアをつくる単語が重要となるほど、ランクが高くなる。
定義式は以下の通りとなる。

$$PageRank(A) = (1-d) + d\sum_{i=1}^{n}\frac{Pagerank(W_i)}{C(W_i)}$$

W_i は単語、i は A 以外の単語を掃引する。$C(W_i)$ は W_i とペアになっている単語の総数となる。
d はダンピングファクターと呼ばれるパラメータで、適当な値（例：0.8～0.9）等に設定されている。

共起分析するための手順（ネットワーク図を出力するまで）

以下のプロセスで分析を実行します。



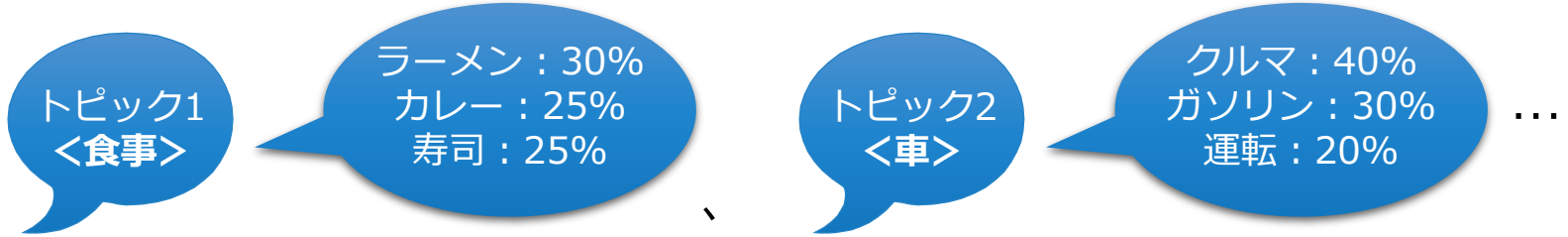
共起分析については **NLP_8_colocation.ipynb** にソースを用意しましたのでご確認ください。

8-3. LDA（Latent Dirichlet Allocation）
（潜在的ディリクレ配分法）

LDA（Latent Dirichlet Allocation）

LDA は文章を仲間分けするための分析です。テキストデータにおける
クラスター分析だと捉えていただいてよいです。
この分析を行うと文章ごとに、その文章がどのクラスターに所属するか？
という確率が出力されます。この分析ではクラスターをトピックと呼びます。

トピック毎に、そのトピックで発生する単語が異なっており、単語の発生
確率も算出されます。

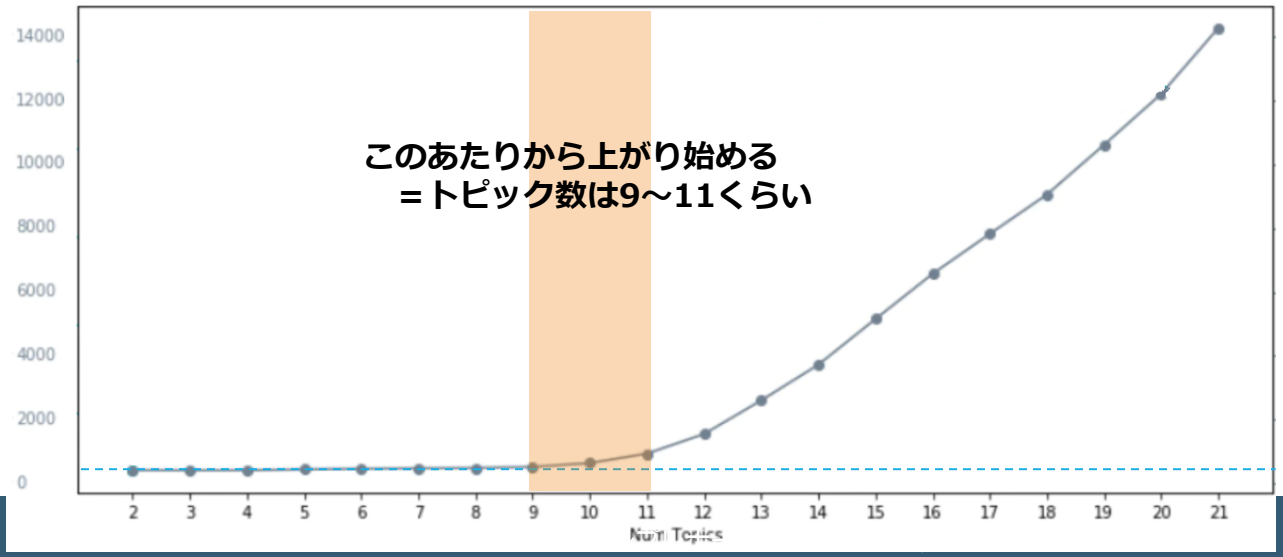


（具体的な算出方法を理解するためには、階層ベイズモデルを理解する必要があります。こちらは一旦スコープ外とします。）

トピック数の決定

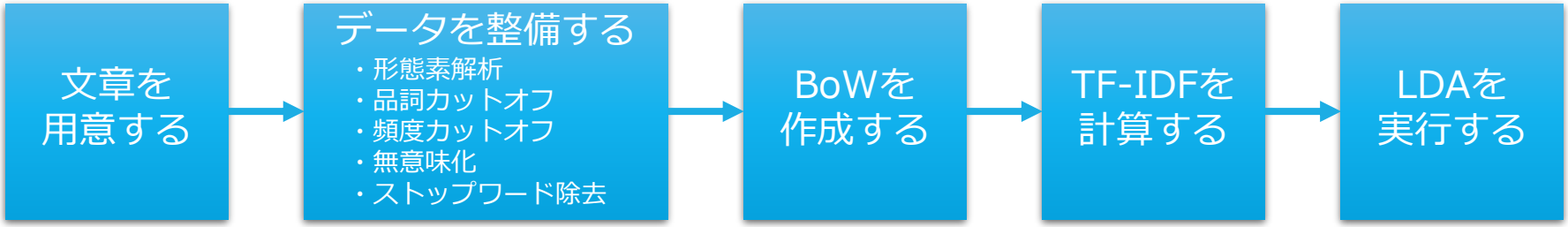
LDA を実行するためには、予め、いくつのトピック数にするのかを決定する必要があります。実務的には「解釈しやすい数にする」という方法を取る事が多いですが、理論上は **Perplexity という量をつかってトピック数を決定**することもできます。Perplexity は、例えばあるトピックにおいて『〇〇は美味しいです。』という文章があった場合に、〇〇に入る言葉の選択肢の数を表します。小さい方が嬉しいです。原理的に**トピックが多くなると Perplexity は増えますので、大きく増加し始める直前のトピック数を選択**するようにします。（現実はなかなかここまでキレイにならない…）

トピック数毎のPerplexityのプロット



LDAを実行するための手順

以下のプロセスで分析を実行します。



NLP_9_LDA.ipynb を用意しましたのでご確認ください。

具体的に以下がどのように出力されているかをチェックしましょう。

- トピック毎の単語出現確率
- 文章毎のトピック所属確率

8-4. 極性辞書とポジネガ分析

極性辞書と簡単なポジネガ分析

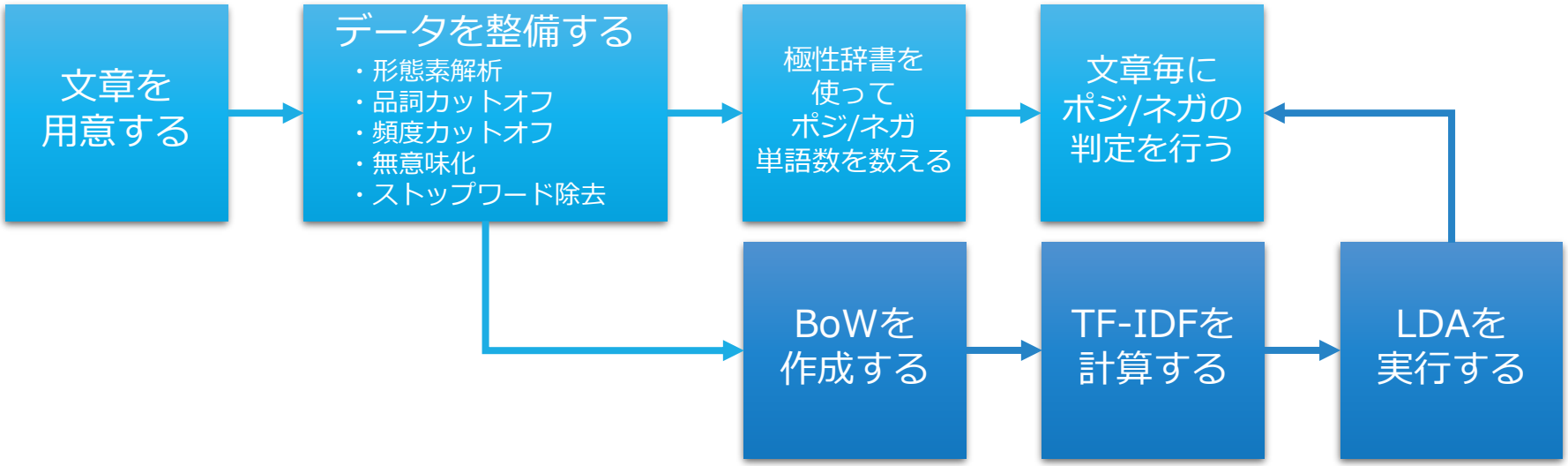
極性辞書とは、単語毎に「ポジティブ」「ネガティブ」「ニュートラル」といった分類がなされた辞書のことを言います。
日本語については、以下の２つあたりが有名です。

- 単語感情極性対応表
  - ✓ 著作者：東京工業大学・高村らが公開
  - ✓ 研究目的に限り利用できる。
  - ✓ http://www.lr.pi.titech.ac.jp/~takamura/pndic_ja.html

- 日本語評価極性辞書
  - ✓ 著作者: 東北大学 乾・岡崎研究室
  - ✓ クレジットを明記していただければ，商用利用も可
  - ✓ http://www.cl.ecei.tohoku.ac.jp/Open_Resources-Japanese_Sentiment_Polarity_Dictionary.html

極性辞書と簡単なポジネガ分析を実行するための手順

極性辞書の内容を確認し、簡単に利用してみましょう。



NLP_10_SimplePositiveNegativeAnalysis.ipynb を用意しましたのでご確認ください。ポジネガ分析は本来、LDA を必要としませんが、サンプルの分析では LDA と組み合わせて使っています。

8-5. ニューラルネットワークによる単語・文書のベクトル化

Word2Vec : BoW の弱点

前章以前では、文章・単語を数値で表現する方法を学んできました。
特に BoW により数値化する方法を学んできたのですが、 BoW にも欠点があります。それは、文脈情報が欠けてしまっている事です。

例えば今まで学んできた BoW では以下の２文は異なるものとして認識されます。（「女王」と「女の王」は全く異なる意味として解釈される。）

文章	私	は	女王	様	です	女	の	王
私は女王様です	1	1	1	1	1	0	0	0
私は女の王様です	1	1	0	1	1	1	1	1

これをある程度解決してくれるアプローチが Word2Vec というニューラルネットワークを基にしたアプローチになります。

Word2Vec：単語のベクトル表現

単語だけ取り出して考えてみましょう。BoW だと以下の表現になります。

文章	女王	女の	王
女王	1	0	0
女の王	0	1	1

これを以下のように表現できれば「女王」と「女の」＋「王」を同じ数値として表現できます。

文章	女王		女の		王	
	王	女の	王	女の	王	女の
女王	1	1	0	0	0	0
女の王	0	0	0	1	1	0

このように１つの単語をいくつかの変数で表現してしまう方法を「単語ベクトル表現」といいます。このような表現を取る事で、単語の表現の違いを吸収することが可能となります。

Word2Vec : Word2Vec による単語のベクトル化

前ページでは「女王」「女の」「王」を以下のベクトル表現にしました。

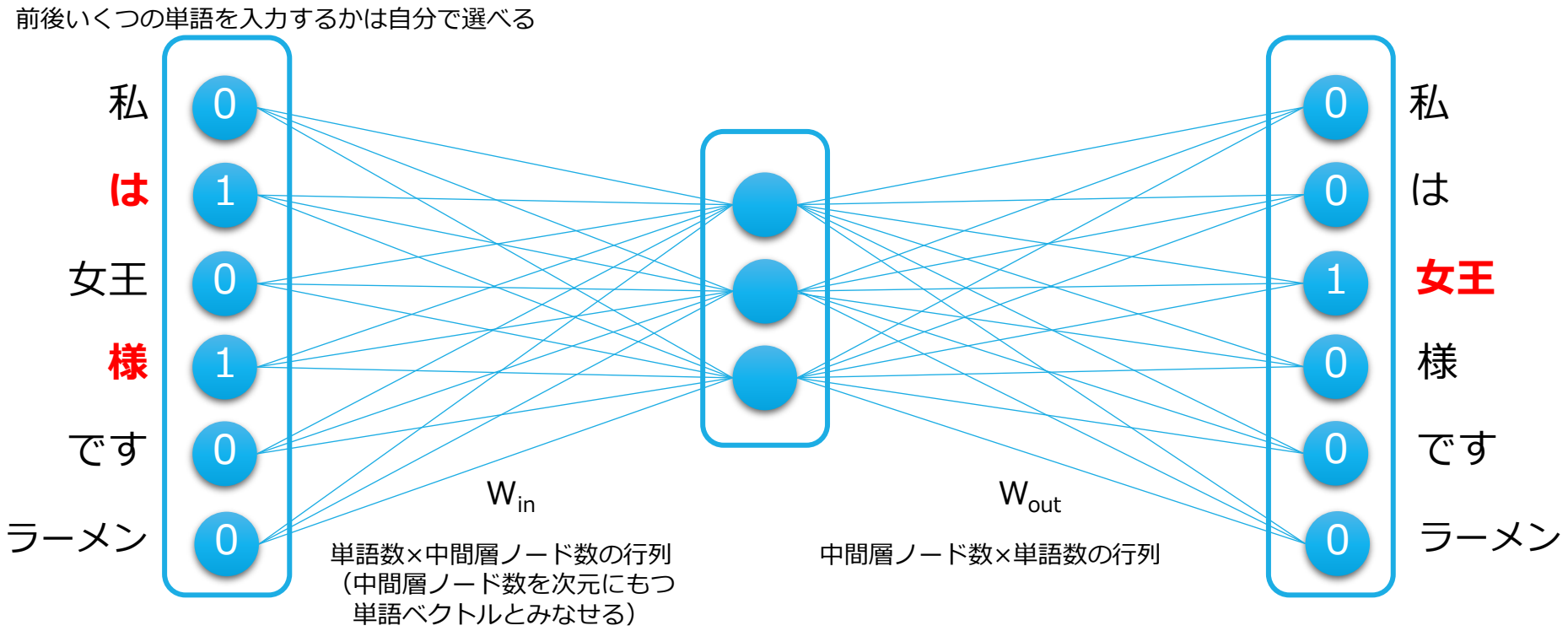
文章	王	女の
女王	1	1
女の	0	1
王	1	0

上記はあくまで意味を理解している人間がベクトル化を行っただけですが、
これを機械に実行可能にしたのが Word2Vec というアルゴリズムです。
Word2Vec は（ベクトル化した要素に意味は持たせられないものの）
各単語の文章における位置を学習させる過程で、単語をベクトル化します。

文章	0	1	2	3	4	5	6	7	8	…
女王	0.1	0.2	0.3	0.4	0.3	0.2	0.4	0.3	0.1	…
女の	0	0	0	0	0	0.2	0.4	0.3	0.1	…
王	0.1	0.2	0.3	0.4	0.3	0	0	0	0	…

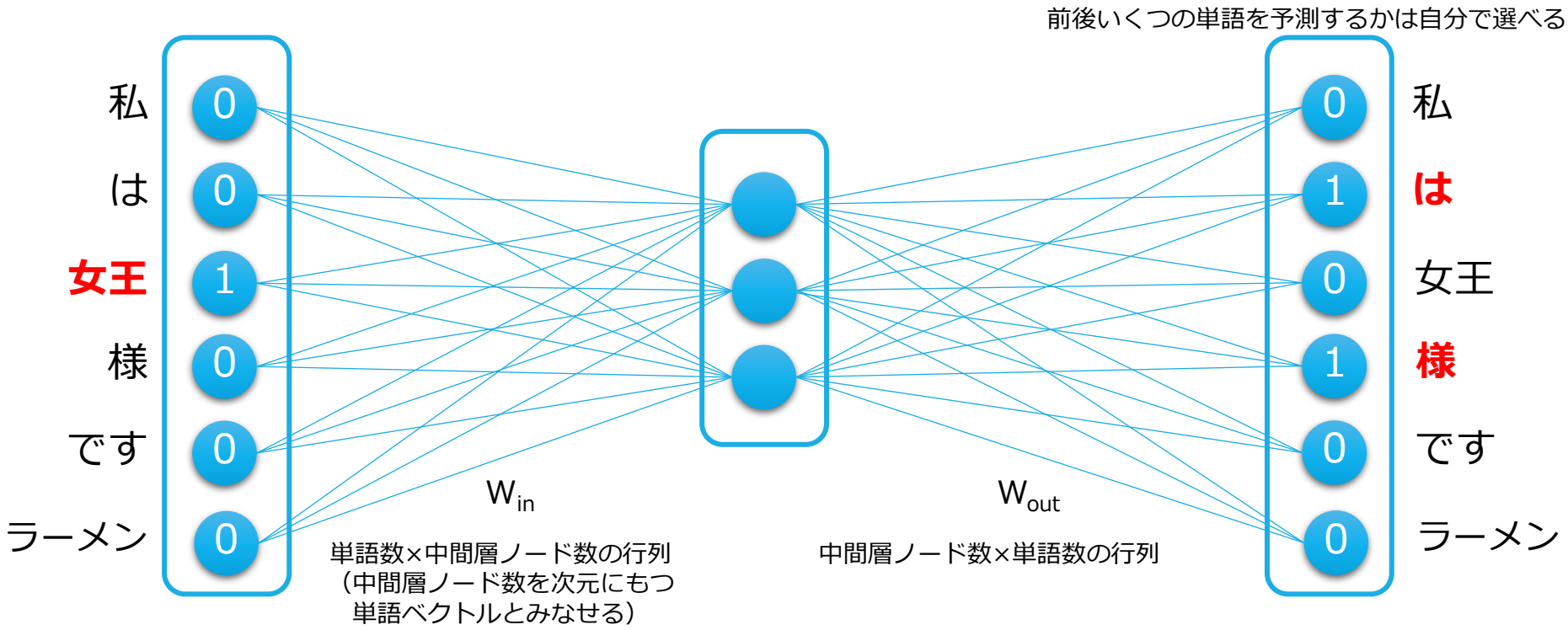
Word2Vec : CBoW （Continuous Bag-of-Words）

単語をベクトル化するための方法がいくつか提案されています。
その１つが CBoWです。CBoW はニューラルネットワークのモデルで、
「私は〇〇様です」の〇〇を予測するというタスクを解くことで作成した
重み（下図の W_{in}）を単語ベクトルとして解釈します。



Word2Vec : Skip-gram

CBoWは「私は〇〇様です」の〇〇を予測するタスクでしたが、逆に「〇〇女王〇〇」として女王様の前後を予測するタスクを設定したのがSkip-gramです。CBoW 同様、このニューラルネットワークを訓練する事で得られた重み（下図の W_{in}）を単語ベクトルとして解釈します。



Word2Vec : 日本語 Wikipedia エンティティベクトルの利用

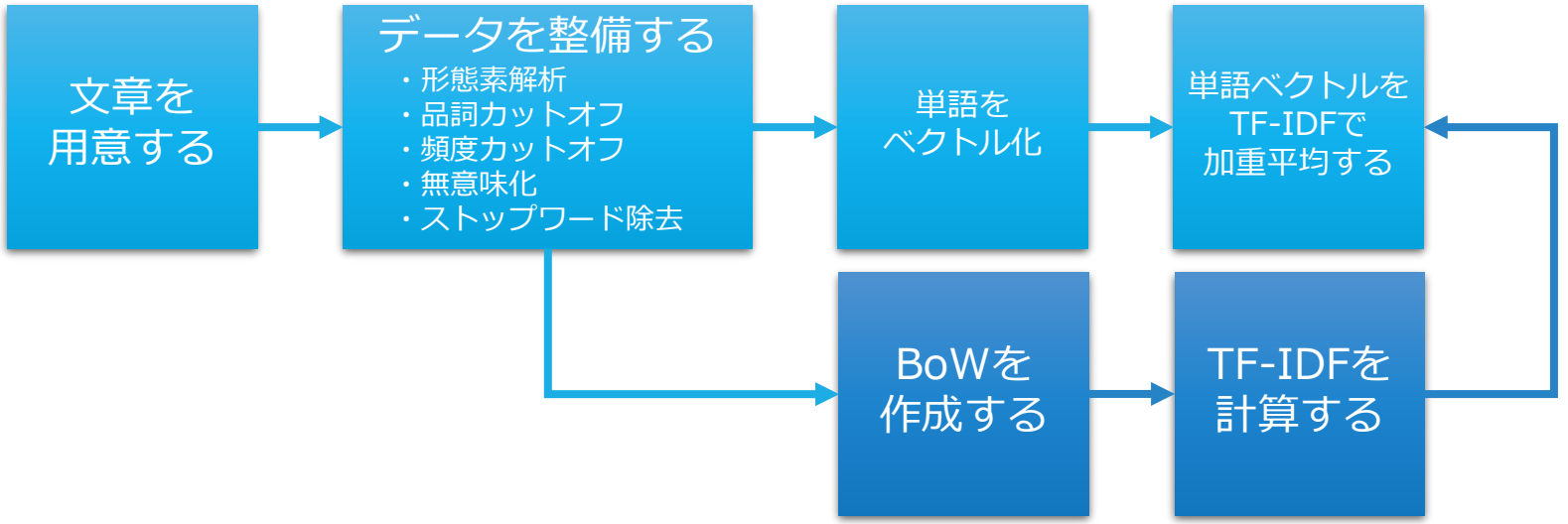
Word2Vec で学習した単語ベクトルを公開してくれている方がいます。
有名なものの１つに東北大の鈴木先生が公開されている「日本語 Wikipedia エンティティベクトル」があります。（200次元のベクトルになっています。）
http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/

この学習済みの単語ベクトルを使って、文書をベクトル化する事ができます。

単純には、文書中に出てくる単語ベクトルの総和＝文書のベクトルですが、その方法だと単語の重要性に全く配慮せずにベクトル和をする事になります。
この問題を回避するために、TF-IDFで加重平均してベクトル和をするという方法があります。さらに利用時はL2ノルムで正規化して利用したりします。
（L2ノルム正規化＝文書ベクトルの内積がコサイン類似度と同じ値になります。）

日本語 Wikipedia エンティティベクトルを利用した文書のベクトル化

極性辞書の内容を確認し、簡単に利用してみましょう。

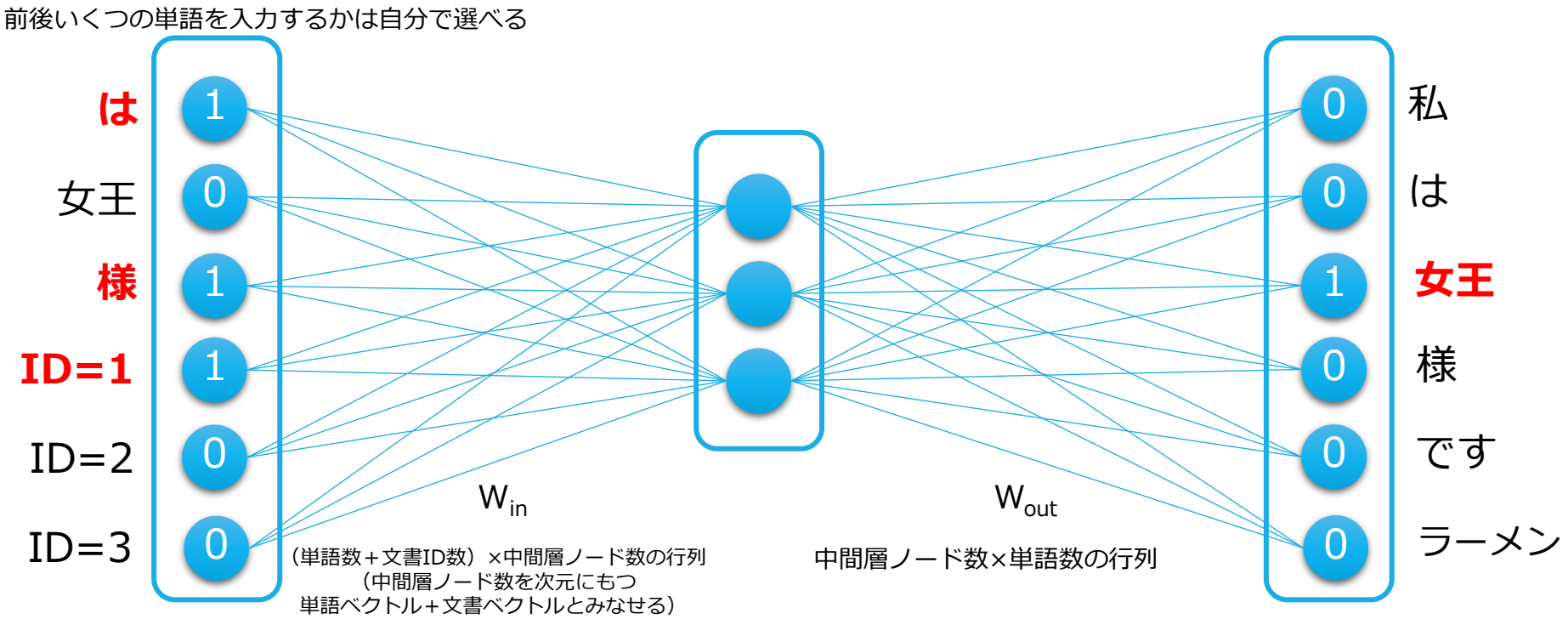


NLP_11_word2vec_wikipedia.ipynb を用意しましたので
ご確認ください。

文書のベクトル化以外に、単語ベクトルの演算なども確認しましょう。

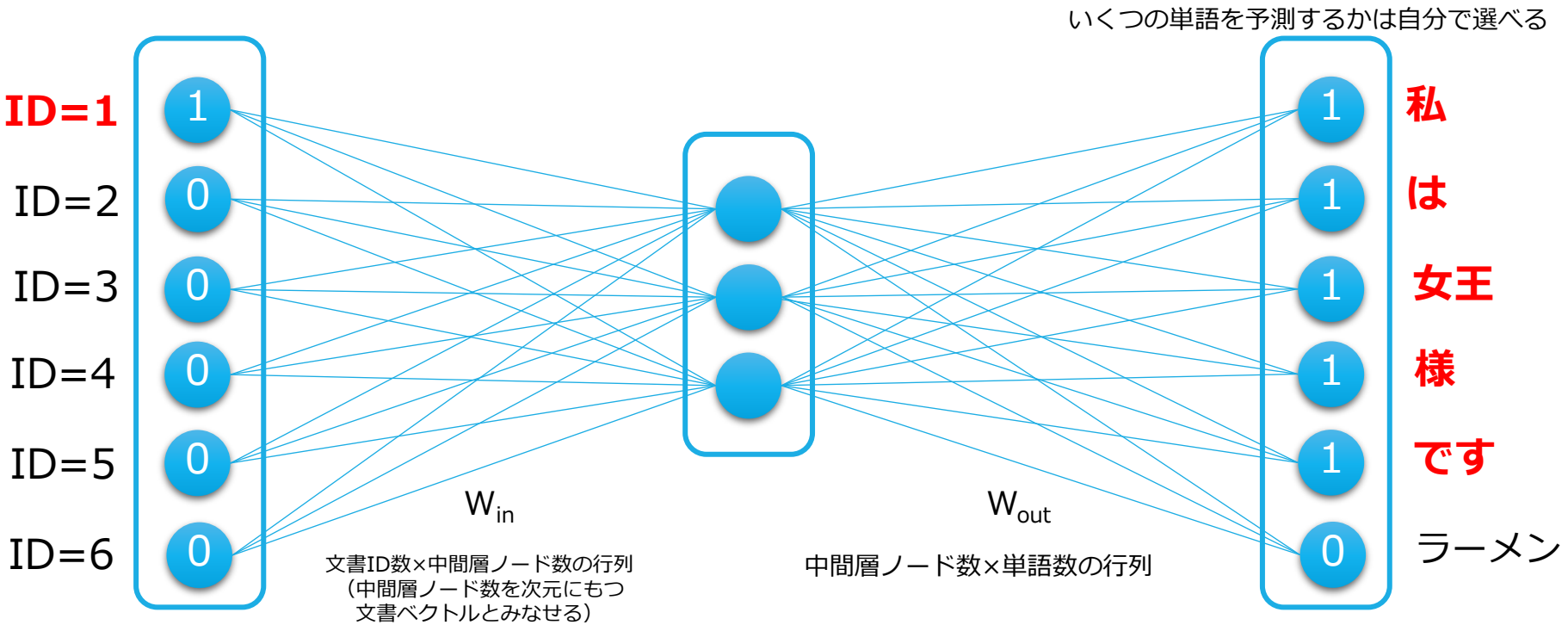
Doc2Vec：DMPV（Distributed Memory Model of Paragraph Vectors）

Word2Vecを少し改良すると、文書を直接ベクトル化することも可能です。
その１つが DMPV です。DMPV もニューラルネットワークのモデルで、
文章以外に文書IDも入力します。そうすることで、文書ID＝文章自体を
ベクトル化する事が可能になります。



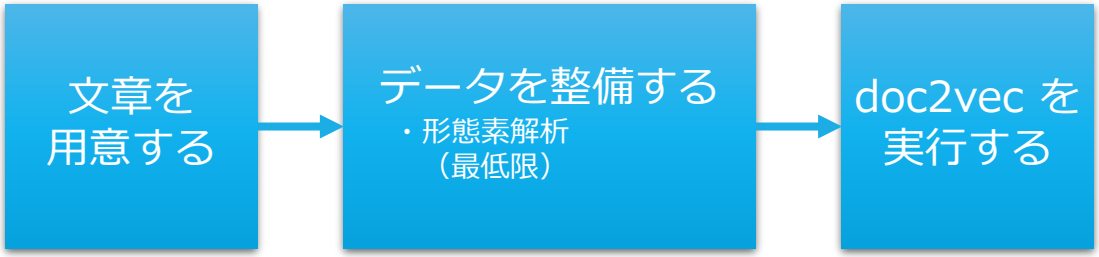
Doc2Vec : DBoW (Distributed Bag of Words)

DBoW は文書IDを入力して、単語「私は女王様です」を予測するタスクを設定して学習を行います。この方法でも、文章をベクトル化する事が可能です。 DMPV と比べてシンプルなモデルになり高速に計算できますが、計算の精度は DMPV の方が優れているといわれています。



doc2vec を利用した文書のベクトル化

doc2vec の実行方法を確認しましょう。



NLP_12_doc2vec.ipynb を用意しましたのでご確認ください。

演習問題④：分析の実施

この章で覚えた分析を駆使し、演習③までで集めたデータをつかって、
何か示唆を出してください。