

## 3.7 節 「言語・知識」 詳細資料

(2024 年 8 月 30 日版)

近年では、企業や個人のコンピュータにテキストデータが大量に蓄積されるようになった。インターネット上に公開されている web ページにはテキストデータが大量に含まれている。また、アンケート、レビューコメントや日報などもテキストデータである。この節では、テキストデータに含まれる**自然言語** (natural language) を主に扱う。自然言語とは、前述の例のように、日常で私たちが話したり書いたりしている言語のことである。他方、自然言語でないものの例としては、プログラミング言語、人工言語、形式言語などがあげられる。自然言語と異なり、意味の解釈に曖昧さがないので、機械的な処理に向いている。

### A.3.7.1 テキストマイニング

**テキストマイニング**とは、アンケートやレビューコメントなど特定の目的のために収集されたテキストデータを統計的に分析することにより、新たな知見を獲得する技術である。

#### (1) テキストマイニングの必要性

なぜテキストマイニングという技術が必要であるか、自由記述コメントのアンケートを例にとり考えてみよう。もし回答が百件程度の分量であれば、担当者が全部読んで、その内容を分析することも可能であろう。しかし、これが1万件となった場合には、全部読むということは困難であろう。また、たとえ読めたとしても、その内容を分析した結論は、担当者が何に大きく共感したのかという主観が入り込みがちである。すなわち、それは単なる担当者の感想であって、分析結果として信頼できるか疑わしいということになる。そこで、テキストを定量化し、客観的な分析をすることが求められる。これがテキストマイニング

なのである。統計的に示すことによって、説得力をもって分析結果を主張することができ、また、気が付いていなかった新たな知見が見つかることもある。

## (2) テキストデータの前処理

テキスト分析をおこなう際には、分析の前処理として、収集したデータを整形する作業が必要である。このことをクレンジングまたはクリーニングとよぶ。たとえば、「コーヒー」という単語は、カタカナで書いたり、漢字で「珈琲」と書いたりできるが、統計をとるときには、同じ単語としてカウントしたほうが有利なことが多い。その場合には、どちらかに統一しておく必要がある。また、日本語の文字コードには全角文字と半角文字があり、半角の「Windows」が全角の「Windows」と表記されることがある。これらはカタカナ表記の「ウィンドウズ」や「ウインドーズ」も含め、**表記の統一**が必要になる。

テキストの中には、絵文字や矢印など、テキスト分析の対象とならない文字列が含まれていることがある。これらについては、あらかじめ削除しておくことが一般的である。

さらに、テキストデータから、個人の名前・電話番号・住所といった個人情報を削除する必要がある。この処理は**匿名化**とよばれる。単純に該当部分の文字だけを削除すると文としての構造が崩れるので、該当部分の文字を、何か特殊な文字(伏せ字)にする、あるいは、該当部分を含む文をまるごと削除する、といったことがおこなわれる。

ここまで前処理の項目をいくつか紹介してきたが、使用するテキストマイニングのツールによっては、それらを自動でおこなってくれる場合もある。

## (3) 自然言語処理のための単語分割

テキストマイニングにおいては、「単語の出現回数を数える」という処理が基本となる。しかし、図 A.3.3 の例にあるように、日本語の文章は、単語単位に分ち書きされていないのが普通である。これが英語などの欧米系の言語と大きく異なる点で、日本語特有の難しさとなっている。そこで、分析に先立ち、**MeCab** や **ChaSen** などのツールを用いて、単語ごとに分ち書きをしておく。これらのツールは、**形態素解析** (morphological analysis) ツールとよばれ

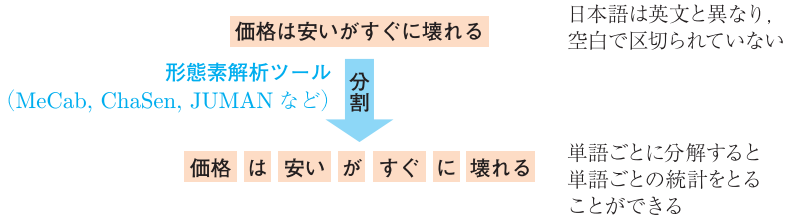


図 A.3.3 自然言語処理のための単語分割

ていて、文を単語の単位に分割し、さらに、それぞれの品詞や読みなどの情報を付与してくれるものである。

#### (4) テキストマイニングの処理の流れ

本節では、次に示すステップをテキストマイニングの進め方のモデルとして紹介している。

- ステップ 1. 単語頻度の分析
- ステップ 2. 単語共起の分析
- ステップ 3. コーディングルールの作成
- ステップ 4. クロス分析

さらに進んだ分析として、次を紹介する。

- 時系列分析
- 好不評分析

以下の小節で、それぞれの処理のステップを詳しくみていこう。

#### ステップ 1. 単語頻度の分析

テキストマイニングにおいて最初におこなうべきステップは、**単語頻度の分析**である<sup>3)</sup>。図 A.3.4 に電気ポットについて書かれた 4 つの文書がある。単語ごとの出現回数を数えてみよう。文書はすでに単語単位に分割されており、かつ、元の文書から助詞や助動詞を除外し、動詞など活用形があるものは原形に

<sup>3)</sup> 使用するテキストマイニングツールや形態素解析ツールによっては**複合語**の登録ができる。たとえば「電気ポット」が 2 つの単語「電気」「ポット」に分かれなくようにすることができる。単語頻度の分析の前に実施しておこう。

助詞や助動詞を無視、  
活用形は原形（終止形）に揃える

文書 1 価格 安い すぐ 壊れる ← 価格は安いが、すぐに壊れた。

文書 2 持つ ところ 取れる やすい

文書 3 十分 満足 低 価格

文書 4 ハンドル 壊れる やすい ← ハンドルが壊れやすい。

↓ 単語の出現回数を数える

単語	回数
価格	2
壊れる	2
やすい	2
安い	1
⋮	⋮

→ 「価格」や「壊れる」ことについての  
関心が高いのだろう。

分析

大局的な傾向を知る

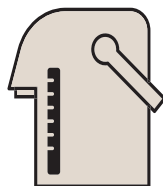


図 A.3.4 単語頻度の分析

揃えてあることに注意されたい<sup>4)</sup>。たとえば文書 1 で、元の文書は「価格は安い  
がすぐに壊れた」であった。「価格は」の「は」や、「安いが」の「が」は助詞な  
ので、削除してある。また「壊れ」の動詞は原形の「壊れる」に変換してある。

このようにしてから、単語ごとに出現回数を数えよう。図 A.3.4 の下表のよ  
うに、「壊れる」は 2 回出現していることがわかる。この例では説明のために文  
書の数を 4 つだけにしたが、文書の数が多ければ単語ごとに出現回数を集計す  
ることによって、何についての関心が高いのかが見えてくる。すなわち、テキス  
トデータを全部読むことなしに、全体の傾向をおおづかみに知ることができる。

## ステップ 2. 単語共起の分析

次のステップは、**単語共起**の分析である。これは 2 つの単語が 1 つの文、ある  
いは文書に同時に出現する頻度を集計する。さきほどの単語頻度の分析で、「価  
格」や「壊れる」といった単語がよく出現するということがわかった。そこで、  
それらの単語と同時に現れやすい単語を探してみよう。たとえば、図 A.3.5 の文  
書群では、「価格」と「安い」という単語が同時に含まれている文書が 1 個見ら

<sup>4)</sup> 品詞や原形の情報も、形態素解析のツールで取得することができる。

単語の組み合わせに注目して、意味を把握する

- 文書 1 価格 安い すぐ 壊れる
- 文書 2 持つ ところ 取れる やすい
- 文書 3 十分 満足 低 価格
- 文書 4 ハンドル 壊れる やすい

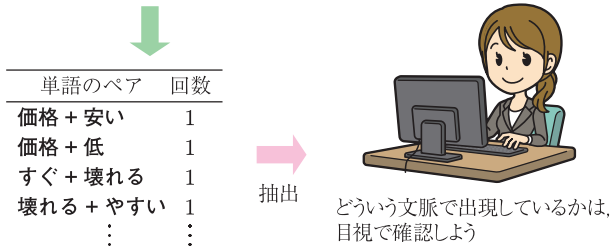


図 A.3.5 単語共起の分析

れる。一方で、「価格」または「安い」のどちらか1個でも含まれている文書は2個ある。ここで、単語と単語の結びつきの強さの指標として、次の **Jaccard 係数**を導入しよう。単語  $A$  と単語  $B$  に着目するとき、

$$\text{Jaccard 係数} = \frac{\text{単語 } A \text{ と単語 } B \text{ の両方が含まれる文書の数}}{\text{単語 } A \text{ または単語 } B \text{ が含まれる文書の数}} \quad (\text{A.3.6})$$

となる。この係数は1に近づくほど、それらの単語が出現するときには2つ同時に出現しやすいことを示す。「価格」と「安い」の Jaccard 係数は0.5となる。文書の数が少ないため、明確には言えないが、「価格」という単語と「安い」という単語の結びつきの強さは大きいと推察できる。

このようにして、同時に現れやすい単語ペアがわかれば、今度は分析の担当者がそれらのペアが含まれる文だけを抽出して読んでみるができる。元のテキストデータが膨大であったとしても、このように絞り込めば効率よくチェックすることができ、人の目でチェックすれば、実際にどういう文脈で単語ペアが使われたのかを感覚的に理解することができる。

### ステップ 3. コーディングルールの作成

人の目で元の文のチェックをおこなうと、いくつか気がつくことが出てくる。たとえば、「価格」と「安い」の単語ペアも、「価格」と「低」の単語ペアも、同じようなコンセプト、ここでは「低価格」を、別の表現で言い換えているだけである。つまり、本当に集計したい対象は、個別の単語や単語ペアではなくて、図 A.3.6 の下表の左側の列に示すようなコンセプトだということがわかる。

したがって、分析の担当者は、こういったコンセプトを表現するパターンを網羅的に用意する必要がある。また、対象とするコンセプトも決めなければならない。このあたり、現在の多くのテキストマイニングツールでは、自動でおこなってはくれないので、分析の担当者が元のテキストを丹念に読んで、発見的に定めているのが実状である。逆に言えば、ここが分析の担当者の腕の見せ所となっていて、同じデータとテキストマイニングのツールを使ったのに、担当者によって知見を得られたり、得られなかったりする、ということは珍しくない。

パターンの記述の方法は、テキストマイニングのツールごとにさまざまである。ここで、示した例のパターンでは、単語の共起だけを定義しており、and の記号は 2 つの単語が共起する条件を表している。or の記号は、前後の条件の 1 つが成立すれば、そのパターンが成立することを表している。このようなパターンを記述した文法や式を、**コーディングルール**とよぶ。

単語共起を調べ、実際の文書のテキストを眺め、勘どころを得られたら、コンセプトを抽出するためのルールを人手で作成する

文書 1 価格 安い すぐ 壊れる

文書 2 持つ ところ 取れる やすい

文書 3 十分 満足 低 価格

文書 4 ハンドル 壊れる やすい

コンセプト	パターン
低価格	( 価格 and 安い ) or ( 価格 and 低 ) or ( お手頃 )
取っ手	( 持つ and ところ ) or ( ハンドル ) or ( 掴む and 部分 )
壊れやすい	( すぐ and 壊れる ) or ( 壊れる and やすい ) or ( 取れる and やすい )

図 A.3.6 コーディングルール

コーディングルールの記述について、高度なテキストマイニングのツールでは、単語の共起だけでなく、単語と単語の**係り受け**（単語や文節同士の関係）まで記述できるものがある。たとえば、図 A.3.7 の2つの文を見てみよう。どちらも、「価格」と「安い」という2つの単語が同時に現れているが、文2のほうでは「価格」と「安い」には係り受けの関係がない<sup>5)</sup>。別の文脈なので、これはコーディングルールの対象にしたいわけではないわけである。したがって、係り受けを指定した単語の共起をコーディングルールとして記述することもおこなわれる。

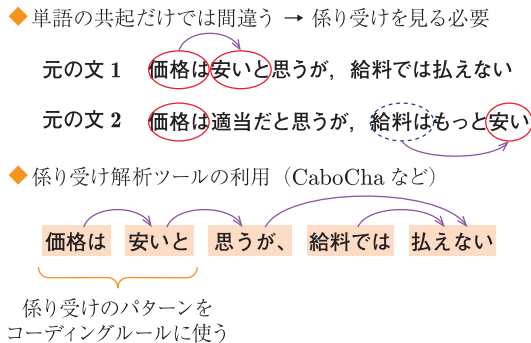


図 A.3.7 係り受け解析

#### ステップ 4. クロス分析

次のステップは、クロス集計による分析である。テキストデータには、それに紐づけられた外部の変数があることが多い。たとえば、アンケートであればそれを書いた人の性別や年齢層、あるいは所属といった属性が**外部変数**となる。これらは自由記述の意見を記入する際に、選択肢として入力されたりする。**クロス分析**は外部変数を用いて層別に分析することである。

図 A.3.8 では、さきほどのコーディングルールを用いて集計したコンセプトの出現頻度を性別ごとに分析した。ここでは、女性層は男性層よりも、壊れた部分により注目している、という、新たな知見を得ることができた。ただし、ここでは説明のために文書の数进行絞ったが、実際の分析ではもっと多くの文書が必要である。

<sup>5)</sup> この例では、係り受けの解析結果を文節単位で示している。

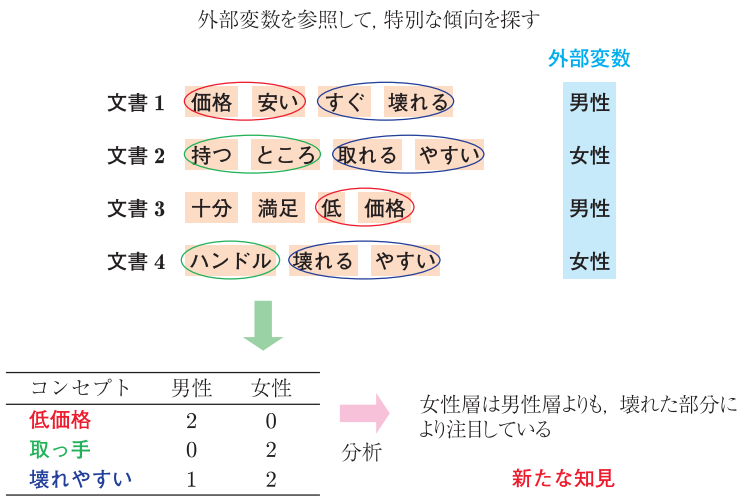


図 A.3.8 クロス分析

(5) 時系列分析

月や週などの時系列の項目を外部変数として、前述のクロス集計をおこなうと、新たな知見が見つかることもある。これを**時系列分析** (time-series analysis) とよぶ。たとえば、お客様相談室での相談内容のメモや返送されるアンケートは毎月集計することができる。図 A.3.9 の表では、5 月から 7 月にかけて件数が急増している項目がある<sup>6)</sup>。この時期に特有の問題がある可能性が高い。新製品の発売が関係していないか、あるいは特定の季節要因がなかったかなどをさらに調べてみる必要がある。

(6) 好不評価分析

次に紹介するのは、**好不評価分析**である。アンケートや商品レビューでは、たとえば「箱根の温泉宿」とか「カメラ」といった対象となる分野が想定されていて、その中にホテルや機種といった具体的な対象があることが多い。すると、対象が他と比べて、評判が良いのか悪いのか、あるいは過去と比べてどうなのか、ということを知りたくなってくる。この分析が**好不評価分析**である。

好不評価分析は、分野について特有の肯定的な表現や否定的な表現をコーディ

<sup>6)</sup> ここでは件数で集計したが、月ごとの合計の件数で除して割合として集計することもある。



時間方向に増加や減少がないか分析する

コンセプト & 性別	1月	2月	3月	4月	5月	6月	7月	8月	9月
低価格 & 男性	15	24	21	18	13	22	23	17	29
取っ手 & 男性	9	4	7	5	18	16	23	12	5
壊れやすい & 男性	13	12	8	16	21	18	20	17	7
低価格 & 女性	6	6	2	3	6	8	8	9	2
取っ手 & 女性	14	23	17	28	49	39	57	18	21
壊れやすい & 女性	19	26	22	25	32	44	39	21	27

➡ 5月発売の商品の取っ手に何かあった？

図 A.3.9 時系列分析

シングルルールを用いて抽出することから始める。この頻度を対象ごとあるいは時系列にクロス分析する。

ここで注意が必要なのは、図 A.3.10 の例のように、同じ「取れやすい」という表現でも、対象物によっては、良い意味で使われたり、悪い意味で使われたりすることがある、ということである。

◆ コメントが肯定的か否定的かを分類する

文書 1	価格	安い	すぐ	壊れる	好評	不評
文書 2	持つ	ところ	取れる	やすい	不評	
文書 3	十分	満足	低	価格	好評	
文書 4	ハンドル	壊れる	やすい		不評	

◆ 表現が肯定的か否定的かは対象に依存する

電気ポッド	持つ	ところ	取れる	やすい	不評	
毛玉取り器	毛玉	取れる	やすい		好評	

図 A.3.10 好不評分析

### A.3.7.2 テキストデータの分類問題

同じ意味を表す文は、完全には一致していなくても、単語の並びはある程度似ているだろうと予想できる。ここで、文のグループが複数あると想定しよう。グループごとに特定の文意があって、そのグループにはその文意を表現する複数の文が入っている。今、新たな文が1つ与えられたとする。似たものの探しをおこなえば、この文をどれかのグループに分類することができるだろう。

社会で稼働しているシステムには、このようなテキストデータの分類問題として実装されているものが多い。

#### (1) テキストデータの分類問題の事例

社会で稼働しているシステムには、このようなテキストデータの分類問題として実装されているものが多い。一例として、図 A.3.11 に示すコールセンターのオペレータを支援する仕組みがある。オペレータの音声とカスタマーの音声は音声認識の仕組みでテキストデータに変換される。変換されたテキストデータは、いわゆる FAQ 文書、すなわち、これまで多くあった問い合わせ内容

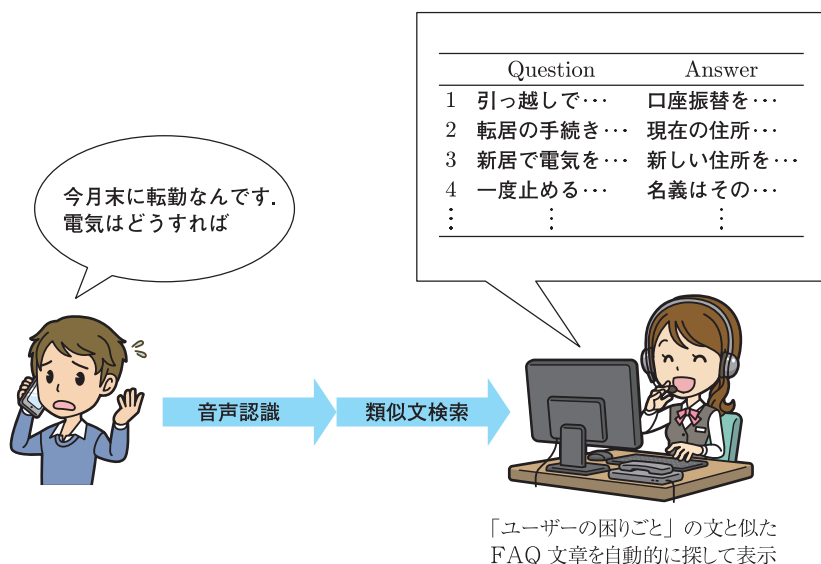
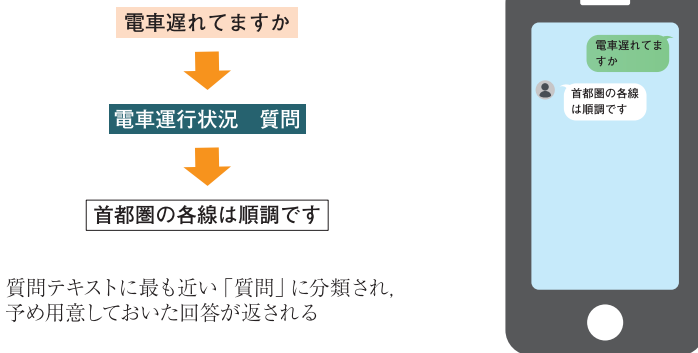


図 A.3.11 コールセンターのオペレータ支援

とその回答が蓄積されたデータ、最も類似度が高いものからオペレータの画面に表示される。ここではテキストデータの似たもの探しがおこなわれていることに注意しよう。

図 A.3.12 に示す例も、カスタマーからの問い合わせに回答するものである。ただし、人間のオペレータは存在せず、AI が自動で回答している。問い合わせ側はスマートフォンやパソコンにテキストで質問を入力し、AI 側は質問を AI が知っている質問タイプに分類し、対応した回答を返す、いわゆるチャットボットである。ここでは、入力したテキストが質問タイプに「分類される」という点に留意しよう。

テキストで入力した簡単な質問に回答してくれる



質問テキストに最も近い「質問」に分類され、  
予め用意しておいた回答が返される

図 A.3.12 質問応答システム

図 A.3.13 は、スマートスピーカの例である。スマートスピーカとは、音楽の再生や家電の操作などを音声だけで指示できるもので、Amazon Echo, Google Home, Apple Siri といった商品名のものが知られている。ユーザーの発声は音声認識によりテキストに変換され、そのテキストがスマートスピーカーの機能に分類され、対応した機能が実行されるという仕組みである。

## (2) 文のクラスタ

さきほどの例では、入力したテキストが該当する意味に分類されるということがおこなわれた。「文を理解する」というと大げさであるが、「コンピュータ

やって欲しいことを声でお願いすると、  
テキストに変換され、最も近い「機能」に分類される

「ミュージック聞きたい」  
「音楽を再生して」



音楽再生機能

音楽を聴きたいときのテキストは  
どれも似たような表現

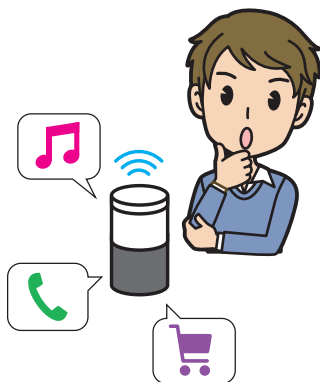


図 A.3.13 スマートスピーカ

がまるで文の意味を理解したように対応する動作をおこなう」ことは、「知っているクラスに文を分類する」という処理にはかならないのである。

たとえば、「ニュース読み上げ」「音楽再生」「電灯 ON」の3つの機能をもつスマートスピーカを作ることを考えよう。最初におこなうことは学習データの収集で、機能ごとにどのような発話があるかを考えて、そのテキストデータを集めることである。機能ごとに多様な言い方があるので、多くのバリエーションを集める必要がある。ここで、集めたテキストデータの文をそれぞれベクトルとして表すことにする。このようなベクトルの作り方については次に説明する。

図 A.3.14 は、それらのベクトルを2次元に可視化したイメージである。意味が似た文のベクトルは似た位置に配置されている。機能ごとに集めたテキストデータは、互いに似ていて、クラスターを構成していることがわかる。

ここで、新たな入力「音楽開始」が与えられたとしよう。図 A.3.14 では、一番近いものは音楽再生機能のグループであるので、分類すべき先は音楽再生機能となる。

### (3) 基本的な文ベクトル

さて、ここまで似たものの探しをベースに分類をおこなう方法を考えてきたが、文の意味を表すベクトルはどのように作成すればよいだろうか。この小節では、

「この機能をよぶには、こんなふうに話すはず」という例文を、事前に大量に用意しておく → 入力に一番近いグループが分類先

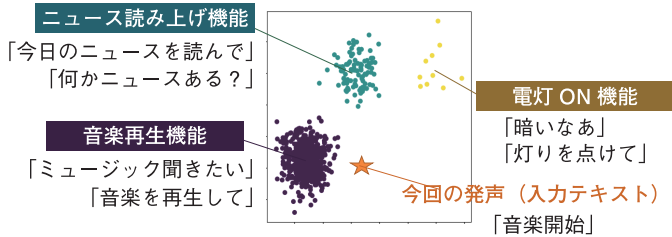


図 A.3.14 テキストデータの分類

bag-of-words 法の基本的な形式を説明する。図 A.3.15 は、機能ごとに集めた文を単語ごとにバラバラにして、機能ごとの袋に入れたイメージである。単語の順番を考えずに袋に入れてしまうので、係り受けは見えていない。

新たな入力も、やはり単語ごとにバラバラにして袋に入れる。それを、袋の間で似たものの探しをおこなって、分類先を決める。これが bag-of-words 法のイメージである。

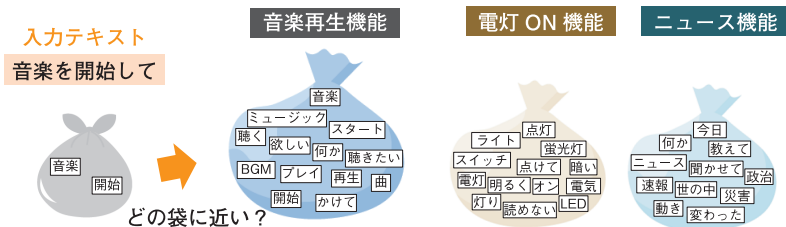
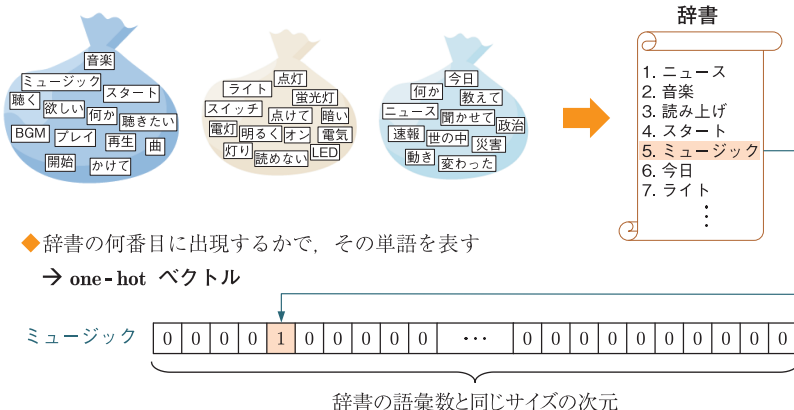


図 A.3.15 似たものの探しのための最も簡単な方法

bag-of-words 法における文ベクトルの作り方をみていこう。図 A.3.16 のように、まず**単語辞書**を作る。単語辞書は、すべての袋の中の単語を取り出して、存在する単語を一覧にして並べたものである。先頭からインデックス番号を付けていくと、単語をインデックス番号で参照することができる。次に、単語を表現するベクトルを考える。図 A.3.16 の下部に示すように、辞書にある単語の

◆集めたテキストに出現する単語の一覧をつくる → 辞書



◆辞書の何番目に出現するかで、その単語を表す  
→ one-hot ベクトル

図 A.3.16 似たもの探しのための辞書

総数と同じ次元のベクトルである<sup>7)</sup>。ここで「ミュージック」という単語は、辞書の5番目に登録されているので、ベクトルの5番目の要素を1として、それ以外の要素は0とする。このような形式で単語を表現したものを **one-hot ベクトル**とよぶ。

次に、one-hot 表現の**単語ベクトル**を元にして**文ベクトル**を作成しよう。図 A.3.17のように、機能ごとの袋に複数の単語が入っているので、その one-hot 単語ベクトルを寄せ集めて、文のベクトルを作る。これが bag-of-words 法の文ベクトルの最も基本的な構成であって、ベクトルの要素が1になっていれば、それに対応する単語がもとの文(ここでは袋)に含まれているということである。

#### (4) コサイン類似度による分類

ここまでで、機能の袋ごとの文ベクトルと新たな入力文ベクトルが用意できた。これらを用いて似たもの探しをおこなおう。図 A.3.18において、入力文に対応する文ベクトルが一番上であり、機能ごとの文ベクトルが、下3つであるとする。さて、入力文ベクトルはどの機能のベクトルと一番似ているだろうか？

<sup>7)</sup> たとえば、辞書の単語数が5000であれば、5000次元のベクトルとなる。通常では見られないような長いベクトルであることに留意されたい。

- ◆ 単語の順番を考えない(簡便な表現)
- ◆ 単語の one-hot 表現を集めて、文のベクトルを作る
- ◆ 「は」「に」「を」などの助詞や「。」「、」は除外してよい

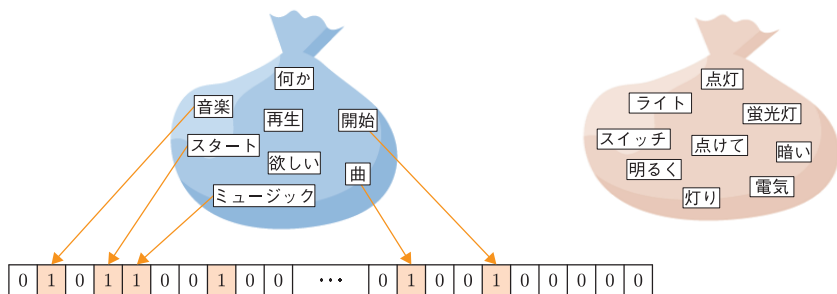


図 A.3.17 文のベクトル表現：bag-of-words 法

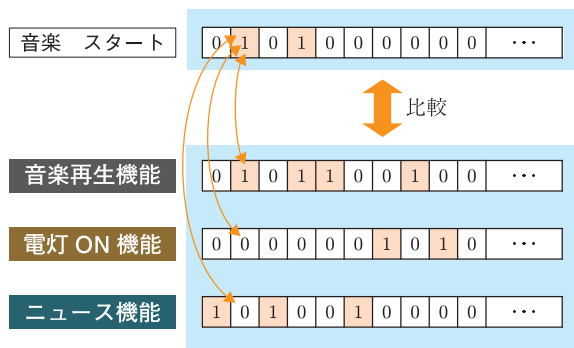


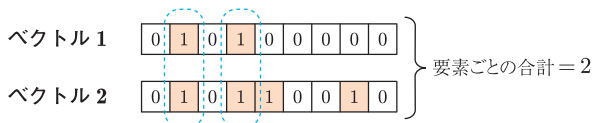
図 A.3.18 文ベクトルを使った似たものの探し

似ている、似ていないをチェックするときの観点を考えよう。入力文ベクトルで1の値となっている要素は、対応する単語があるということである。比較対象の文ベクトルの、対応する要素も1であれば、似ているということの根拠になる。一方で、入力文ベクトルで0の値となっている要素は、単にその単語がないということであって、特に強い意味は持っていない。ここで掛け算の関係  $1 \times 1 = 1$ ,  $1 \times 0 = 0$ ,  $0 \times 1 = 0$ ,  $0 \times 0 = 0$  を思い出そう。すなわち、要素ごとの掛け算をおこなえば、似ているかどうかの判断の根拠を得ることができる。

要素ごとに掛け算をおこない、その結果を合計する操作は、1.2 節でみた通り

数学的には内積とよばれる操作である。たとえば図 A.3.19 では、ベクトル 1 とベクトル 2 の内積は 2 と計算できる。この内積の値が、似たもの探しの指標となり得る。ただし、このままだと不都合なことがある。たとえばほとんどの要素が 1 であるベクトルは他のどのようなベクトルと内積をとったとしても大きい値になりやすいのでそれをもって似たもの度合いが大きいとはいえない。そこで図 A.3.19 のように、内積をベクトル 1 とベクトル 2 のそれぞれの大きさとで割り算をおこない、比較を公平にする。この指標を**コサイン類似度**とよぶ。

◆ さきほどの計算 → 2 つのベクトルの内積



◆ コサイン類似度は、内積をそれぞれのベクトルの大きさとで割ったもの

$$\frac{(\text{ベクトル 1 とベクトル 2 の内積})}{(\text{ベクトル 1 の大きさ}) \times (\text{ベクトル 2 の大きさ})} = \frac{2}{\sqrt{2} \times 2} = 0.70$$

➡ 1に近いほど、2つのベクトルは同じ方向を向いている（最大値は1）

図 A.3.19 コサイン類似度による似たもの度合いの計測

図 A.3.18 における比較をコサイン類似度を用いておこなうと、一番上の入力について、音楽再生機能は他の機能よりもコサイン類似度<sup>8)</sup>が大きくなる。したがって、入力は音楽再生機能に分類される。

ここまでの説明で、2つの文が似ている度合いは2つの文ベクトルのコサイン類似度で計算できるということがわかった。図 A.3.20 に示すように、文が似ているときは2つのベクトルが同じ方向を向いているといえる。2つのベクトルがなす角を  $\theta$  とすると、コサイン類似度はこの  $\cos \theta$  である。コサイン類似度の最大値は1であり、これは角度でいうと0度に相当し、2つのベクトルはまったく同じ方向を向いているということになる。図 A.3.19 の例では、コサイン類似度として0.7という値が得られた。角度でいうと45度くらいとなり、やや同じ方向を向いているベクトルと観察される。

<sup>8)</sup> コサイン類似度は、多次元の空間の中で2つのベクトルが同じような向きであるかを示す尺度であって、2つのベクトルがなす角度が  $\theta$  であるときに  $\cos \theta$  として求められる。



2つのベクトルが同じ方向を向く → 2つの文が似ていると解釈

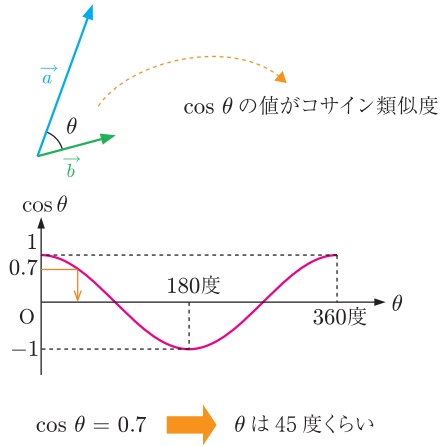


図 A.3.20 コサイン類似度のイメージ

#### (5) TF-IDF を用いた文ベクトル

さて、ここまで学んだ bag-of-words 法の文ベクトルについて、改善ができないか考えてみよう。ここまでは文ベクトルとして 0 と 1 の要素からなるベクトルを使用し、どの単語も平等な扱いであった。

しかし、図 A.3.21 に示すように、それぞれの機能の袋には、機能を区別するために重要な単語とどの機能にも共通の単語が混在していることがわかる。そこで、分類に役に立つ重要な単語には大きな重みを付けたほうがよいという考え方があろう。また、袋に単語を詰めるときに多数回出現した単語は機能を表現するためには重要な単語だろうから大きな重みを付けたほうがよいとも考えられる。この重み付けの係数を **TF-IDF** とよぶ。

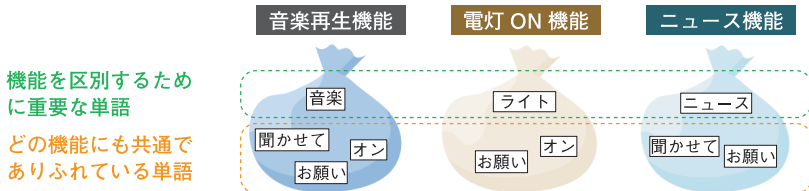


図 A.3.21 機能を区別するために重要な単語

TF-IDF は、Term Frequency (TF) と inversed Document Frequency (iDF) の積として求められる。ここで、さきほどの「機能の袋」を文書とよぶことにすると、TF は、着目する単語が文書ごとに何回現れるかを割合で表したものである。

$$TF = \frac{\text{対象単語が対象文書中に現れる回数}}{\text{対象文書中の全ての単語数}^{9)} \quad (\text{A.3.7})$$

iDF は、着目する単語がどの文書に含まれているかを見て、含まれている文書の数を割合で表したものである。ただし、その逆数をとって対数をとっておく。

$$iDF = \log \frac{\text{文書総数}}{\text{その単語を含む文書の数}} \quad (\text{A.3.8})$$

TF と iDF の積が TF-IDF であり、文ベクトルの要素にこれを用いる。

◆ Term Frequency (TF)

$$TF = \frac{\text{対象単語が対象文書中に現れる回数}}{\text{対象文書中の全ての単語数}}$$

その単語が文書に何度も出てきていれば、重要な単語だろうという重み

◆ inversed Document Frequency (iDF)

$$iDF = \log \frac{\text{文書総数}}{\text{その単語を含む文書の数}}$$

その単語が他の文書にもありふれて出現していれば、重要でないだろうという重み

◆ 上記 (TF 値) × (iDF 値) を 0,1 の代わりに用いて、文ベクトルを作成する



図 A.3.22 TF-IDF による重みづけ

### A.3.7.3 機械学習によるテキストデータの分類

前項では、bag-of-words 法による文ベクトルをコサイン類似度で比較することにより、分類する方法を学んだ。この節では、テキストデータの分類問題を

<sup>9)</sup> 分母には対象文書に含まれる単語数を用いる (この場合、同じ単語が複数出現しても区別して数える)。

- ◆ 入力テキスト(特徴量)に対応する正解の分類(クラス)をモデルに学習させる
- ◆ 学習後は、モデルにテキストデータの特徴量を入力すると、それに対応する分類クラスの予測が出力される

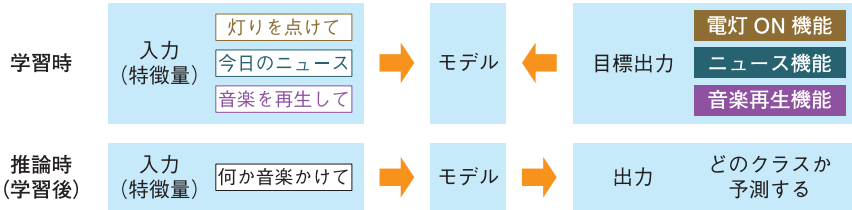


図 A.3.23 テキストデータの機械学習 (分類問題)

より高度な機械学習によって実現する。

テキストデータの分類問題は、図 A.3.23 に示すように、機械学習の枠組みで実装することができる。まず、想定する機能ごとにテキストデータを大量に集めておく。これが学習データとなる。テキストデータの特徴量に変換してモデルに入力し、対応する機能の分類を正解のクラスとして学習をおこなう。学習が済んだモデルは、入力となるテキストデータを与えると、その分類結果を予測してくれる。

テキストデータの分類問題を機械学習で実装する際には、いろいろな機械学習の手法を適用することができる。以前はサポートベクターマシンを用いることが多かったが、近年はニューラルネットワークを使うケースが多くなっている。

### (1) 文ベクトルを用いた機械学習

前項で学んだ文ベクトルをそのまま特徴量として用いても、分類問題のための機械学習モデルを構成することができる。図 A.3.24 にニューラルネットワークを用いた例を示す。出力層のユニットは一つ一つが分類クラスに対応しており、一番大きい値を出力したユニットが、推定された分類クラスとなる。

### (2) 単語ベクトルを用いた機械学習

ここまで説明した方式では、単語の順番を考慮しない文ベクトルを入力して用いてきたが、それで十分だろうか？

図 A.3.25 の文例について考えてみよう。もし、係り受けを考慮しないと、「暗

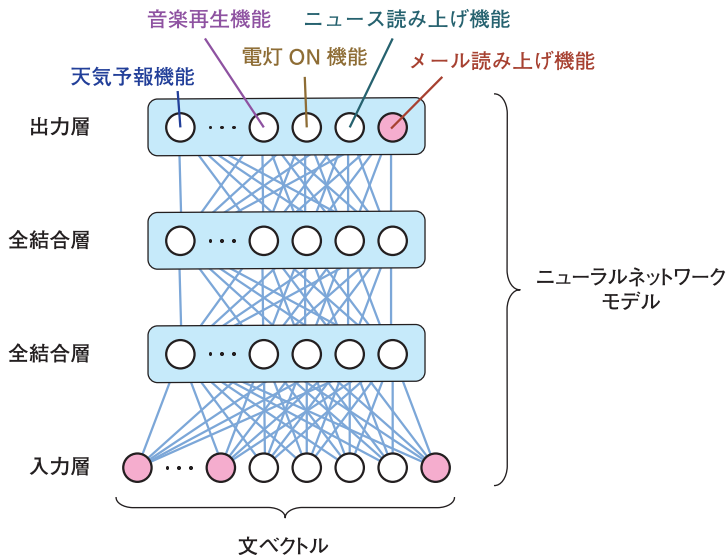


図 A.3.24 文ベクトルを用いた機械学習モデル

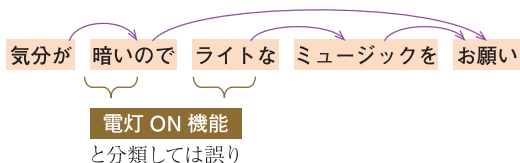


図 A.3.25 単語順を考慮しない文ベクトルでは不十分

い」と「ライト」という単語があるので、この文は「電灯 ON 機能」と誤って解釈されてしまう可能性が高い。

そこで、単語を単語ベクトルに変換し、それを順番に入力するというモデルが期待されるのである。その場合、係り受けの情報は明示的にはモデルに入力されないが、単語を順番にモデルに入力することにより、係り受けと同様の情報がモデルに学習されると期待される。

単語ベクトルを順番に入力できるタイプのニューラルネットワークとして、再帰型ニューラルネットワーク (RNN) がある。RNN では、その時点の層の出力を入力側に戻し、次の時点での入力と並列で層に入力するというものである。

図 A.3.26 を見てみよう。ここで「暗い」という単語を入力したときには、そ

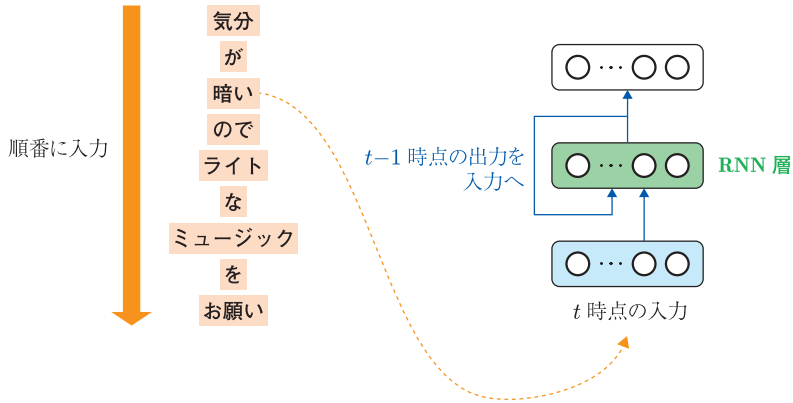


図 A.3.26 単語ベクトルを順に入力する RNN

れまでの「気分」と「が」という単語の入力は既に済んでいて、それらの情報は、層の出力から戻された信号に埋め込まれていると考えることができる。また、RNN の一種である LSTM や GRU (3.5 節参照) を使えば、ユニットの中に長期の記憶を貯めておくことができるので、より正確に文脈を捉えることができる。

RNN は、図 A.3.27 のように時間方向 (単語列方向) に展開した形で表示できる。時間方向に展開することで、時系列の入力をイメージしやすくなる。また、このように展開した形式で学習をおこなうことで、学習の処理が易しくなる。

ここでは、RNN 層の上に全結合層を重ねて、分類モデルの全体像として示されている。出力層には分類クラスに対応したユニットが並ぶ。文の末尾の単語が入力された時点の出力層をチェックし、一番大きい値を出しているユニットが対応している分類クラスが、予測結果となる。

図 A.3.27 では、順に入力する単語のそれぞれの上位に中間層があるので混乱しやすいが、これらの中間層は実態としては 1 つなので注意しよう。すなわち複数の層が並列に並んでいるように見えても、パラメータは共有されていることに留意されたい。

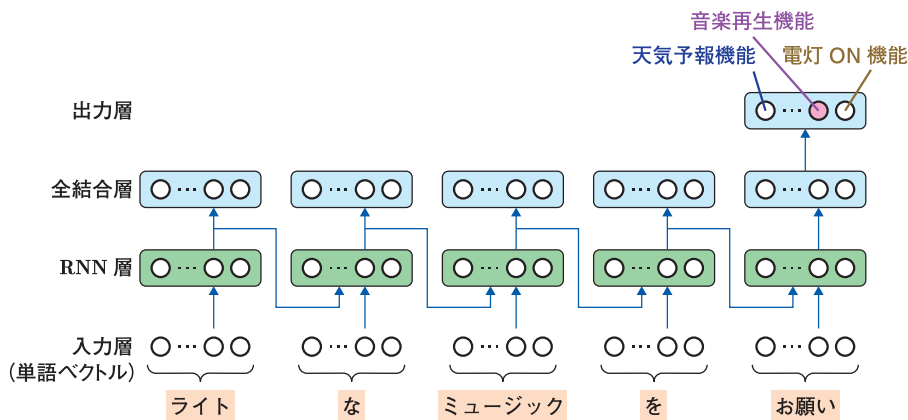


図 A.3.27 単語ベクトルを順に入力する RNN の時間展開

### (3) 単語ベクトル

RNN により単語をモデルへ順番に入力することができるようになったが、単語を表現する単語ベクトルはどのようなものを使用すればよいだろうか。

1 つの案は、既に説明した単語の one-hot ベクトル表現である。この方式はわかりやすいが、デメリットもある。似た意味の単語が複数あるときに、それらがまったく別のベクトルとして表現されてしまうのである。たとえば、「コーヒー」と「珈琲」はほぼ同じ意味の単語であるが、まったく異なるベクトルになってしまう。

一方、近年よく用いられている方式として、**分散表現**のベクトルがある。この方式では、数百次元という比較的サイズの小さいベクトルに、実数値を詰め込んだ形式となっている。代表的な方式として **Word2Vec** があり、似たような意味の単語は似たベクトルとすることができる。この変換は、単語の**埋め込み** (embedding) とよばれる。分散表現のベクトルは埋め込みベクトルとよばれる。

埋め込みベクトルは単語ごとに用意されるが、これはニューラルネットワークを用いた機械学習により獲得される。図 A.3.28 に、Word2Vec の continuous bag-of-words (**CBOW**) モデルによる学習の仕組みを示す。

まず学習用のテキストデータを大量に用意する。CBOW モデルでは、テキストの場所を移動しながら、連続する数単語を取り出し、中心の単語を前後の単語

## 単語列 私は牛乳が嫌い

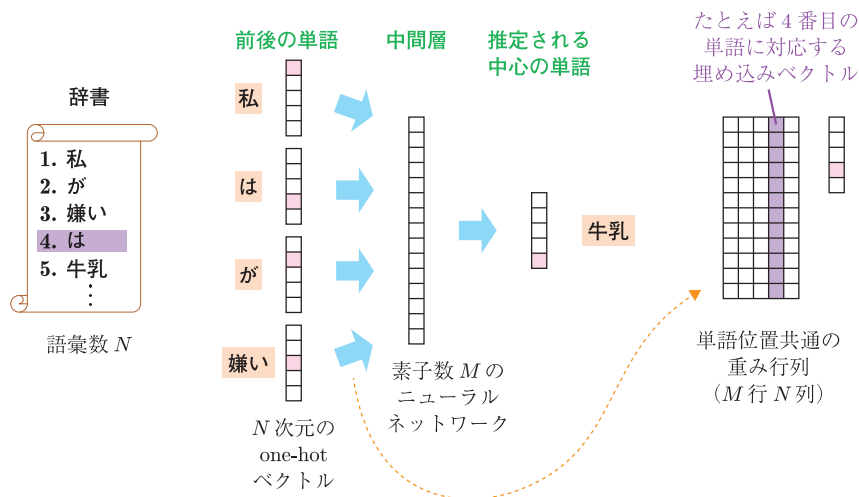


図 A.3.28 Word2Vec の学習 (CBOW モデル)

から推定するニューラルネットワークモデルを学習する。各単語は one-hot ベクトルとして表現され、単語位置共有の重みパラメータで、中間層と結ばれる。学習が完了すると、この中間層への重みパラメータが、所望の埋め込みベクトルとなる。

図 A.3.29 に、Word2Vec の単語ベクトルの例を示す。ここでは「コーヒー」と「珈琲」の単語が 200 次元のベクトルとして表現されている。表記は異なるがほぼ同じ意味の単語なので、似たベクトルとなっていることがわかる。

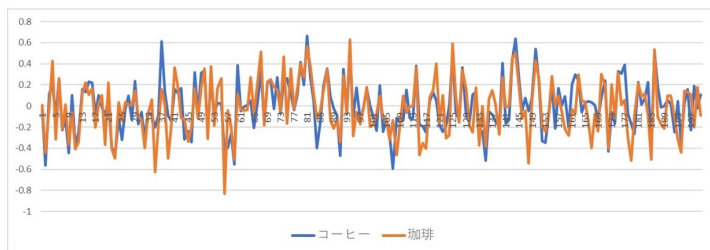


図 A.3.29 Word2Vec の単語ベクトルの例

## 課題学習

- A.3.7-1** テキストマイニングの説明の際には、「Windows」と「ウィンドウズ」のような、単語表記の揺れを統一する前処理の必要性を述べた。この前処理は、bag-of-words 法によるテキストの分類問題の際にも必要だろうか？ その理由とともに回答せよ。また、その前処理は、分散表現による単語ベクトルを使用したテキストの分類モデルにも必要だろうか？ その理由とともに回答せよ。
- A.3.7-2** テキストマイニングツール KH Coder を使用し、分析をおこなってみよ。データ素材としては、「沖縄観光客満足度調査」の『観光客の声』の公表（平成 18 年度観光統計実態調査・別冊）ページ <https://www.pref.okinawa.jp/shigoto/kankotokusan/1011671/1011816/1011825/1020290.html> にある「沖縄観光についての自由意見」の Excel ファイルを使うとよい。ツールのインストール方法や分析方法の説明は、動画学習サービス gacco で公開されている動画教材「大学生のためのデータサイエンス (Ⅲ) 問題解決編」の第 4 週や、そのオフィシャルスタディノートなどを参照するとよい。
-