

# aa-12. 自然言語処理

金子邦彦



# 「人工知能」第12回の内容



自然言語処理（NLP）＝人間の言葉をコンピュータに処理させる技術

言葉には  
難しさがある

曖昧性  
(同じ言葉が  
複数の意味)

表現の多様性  
(同じ意味を  
別の言い方)

常識・文化的  
背景の理解

応用技術

分析応用系

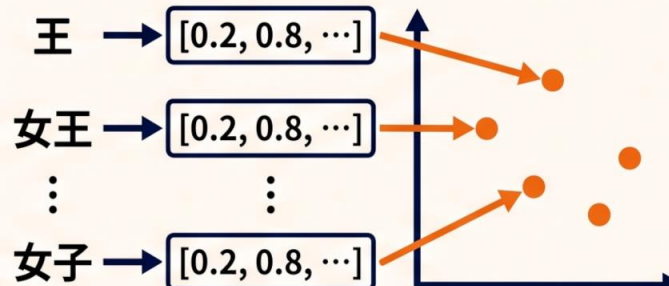
固有表現抽出／  
類似度分析／  
意味的検索／  
トピックモデリング  
感情分析

構造変換系

構文解析／  
意味解析／翻訳

ベクトル化

言葉を数値の並び（ベクトル）に変換



意味の近さ＝距離

類似語の取得  
意味的検索  
文の類似度判定

実際に使う

KNP:  
構文解析

DeepL:  
翻訳

Transformer:  
感情分析

Web上の  
単語頻度分析



## 12-1. 自然言語処理

人間の言葉をコンピュータに理解・処理させる技術＝自然言語処理



**検索エンジン**

質問に答えを返す

**音声アシスタント**

話しかけて操作する

**迷惑メール判定**

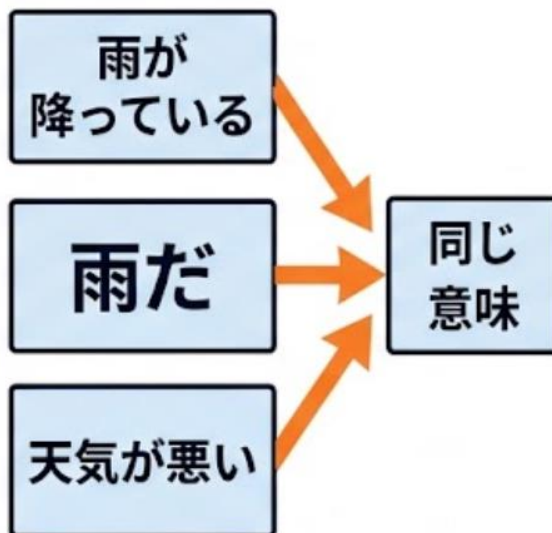
不要なメールを自動で分類

## コンピュータにとって言葉は難しい — 自然言語処理 3つの課題

### ① 言葉の曖昧性



### ② 表現の多様性



### ③ 常識・文化的背景

彼は青い顔を  
をしている

常識・文化が必要

= 顔色が悪い (体調不良)

人間は自然に理解できるが、コンピュータには難しい。AIの進化でこれらを解決できるようになってきた。

# ① 言語の曖昧性（あいまい性）



同じ言葉でも、文脈によって意味が変わる — これがコンピュータには難しい



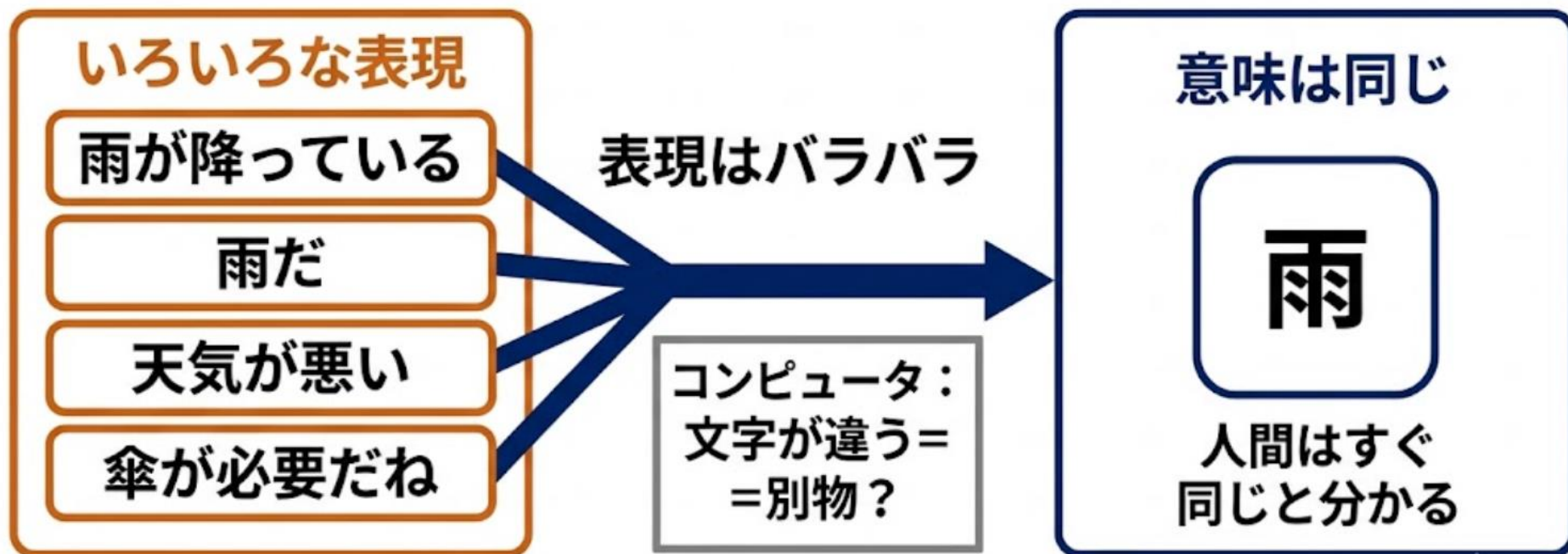
言葉は同じ → 意味は別

別の例:「はし」→「橋」「箸」「端」(音は同じでも意味が違う)

人は文脈で瞬時に判断できるが、  
コンピュータには文脈の理解が必要

## ② 表現の多様性

同じ意味でも言い方はいろいろ  
— だからコンピューターは"同じ"と気づきにくい



課題：表層（文字列）が違ってても意味は同じ場合がある  
コンピューターには"意味が同じ"の判断が難しい（表現の多様性）

### ③ 常識・文化的背景



同じ言葉でも、常識や文化がないと本当の意味は分からない

入力される文

2通りの解釈

なぜ人間は分かるのか

彼は青い顔  
をしている

× 字義どおりの解釈

顔が青色に塗られている  
→ コンピュータが陥りやすい

○ 本当に伝えたい意味

体調が悪い・具合が悪そう  
→ 人間はすぐ分かる

常識・文化的背景の  
知識があるから

言葉に書かれていない  
知識を補って読む

他の例

猫の手も借りたい  
→ 字義: 動物の手を借りる  
／ 実際: とても忙しい

油を売る  
→ 字義: 油を販売する  
／ 実際: 仕事中に怠ける

Time flies. (英語)  
字義: 時間のハエ  
実際: 時間が経つのは速い



## 12-2. 自然言語処理の応用技術

# 自然言語処理の意義



膨大なテキストは人手では処理しきれない。自動処理がそれを可能にする

毎日生まれる  
膨大なテキスト

SNS投稿

ニュース

レビュー

論文

人手だけでは  
処理しきれない

自動処理

情報検索の高度化

必要な情報を素早く探せる

多言語コミュニケーション支援

言葉の壁を越える

業務効率化

作業時間を大きく削減

自然言語処理（NLP）の技術は、文章を『分析・応用する』ものと、文章の『構造をとらえ変換する』ものに大きく分けられる

## 分析・応用系

### 固有表現抽出

文中の人名・地名・組織名などを見つけ出す

### 類似度分析

2つの文がどれくらい似ているかを測る

### 意味的検索

意味の近い文書を探し出す

### トピックモデリング

大量の文書を話題ごとに分類する

### 感情分析

文章がポジティブかネガティブかを判定する

## 構造・変換系

### 構文解析

文の文法的な組み立て（主語・述語など）を調べる

### 意味解析

単語や文が表す意味を読み取る

### 翻訳

ある言語を別の言語に置き換える



# 固有表現抽出



文章の中から人名・地名・組織名などの固有の名称を自動で見つけ出す技術



人名  
PERSON

地名  
LOCATION

組織名  
ORGANIZATION

日付  
DATE

金額  
MONEY

# 固有表現抽出と AI



AIが大量の例文からパターンを学習し、固有表現を5つのステップで抽出する

## 1. 文章 入力

文章をそのまま  
入力

## 2. 単語に 分割

文を単語ごとに  
区切る

## 3. 品詞を 判定

各単語が名詞・  
動詞などかを判定

## パターン を認識

固有表現らしい  
並びを見つける

## 5. 種類を 判定して 抽出

人名・地名などに  
分類して取り出す

大量の例文

AIがパターンを学習

新しい文章でも  
同じパターンを発見

王

ベクトルに変換

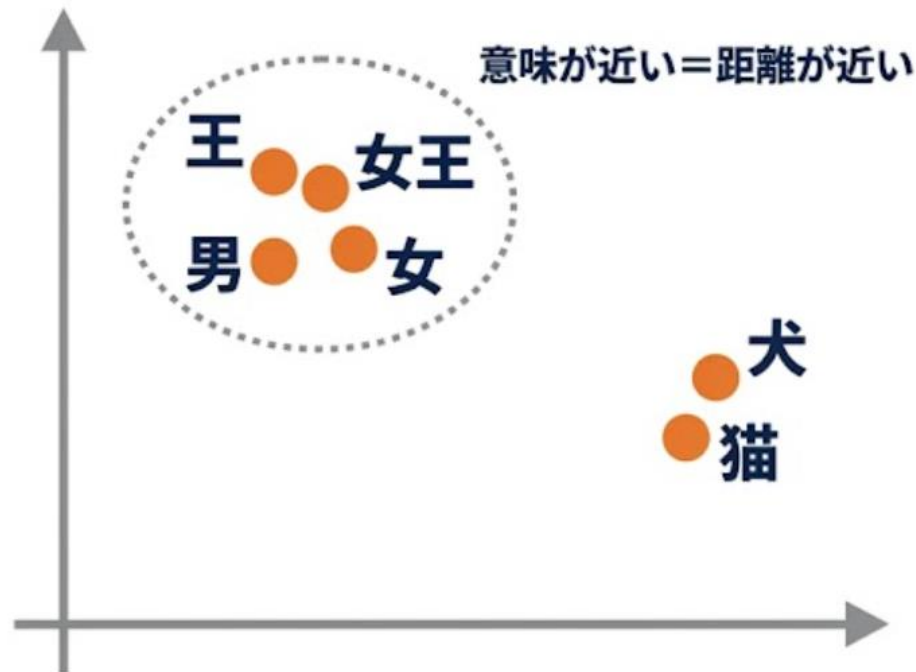

$$\begin{bmatrix} 0.8 \\ 0.2 \\ 0.7 \\ \dots \end{bmatrix}$$

単語のベクトル化（数値の並びに変換）

単語のベクトル化により距離計算が可能になる



## 意味が近い単語を近くに集める



# 似た単語の取得 = 距離が近い単語の取得

「王」に似た単語と類似度

意味が近い = 距離が近い



犬  
猫



|    |      |
|----|------|
| 女王 | 0.85 |
| 男  | 0.72 |
| 王子 | 0.68 |

類似度が高い順に並ぶ

入力した単語に意味が近い単語を、似ている順に取得できる

文も数値の列に変換すれば、意味の近さを数値や検索に使える

## 文の類似度分析

A: 今日は天気が良い → [ ... ]

B: 本日は晴れています → [ ... ]

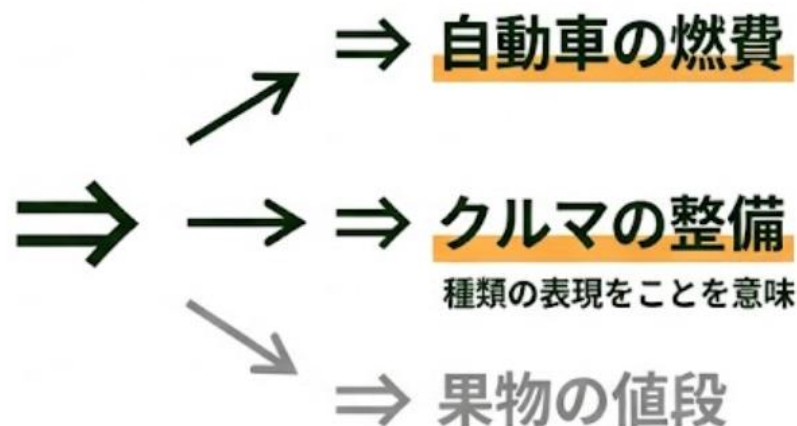
距離を測る

**類似度 = 0.92**



1.0に近いほど似ている  
表層の一致でなく意味で判定

## 意味的検索



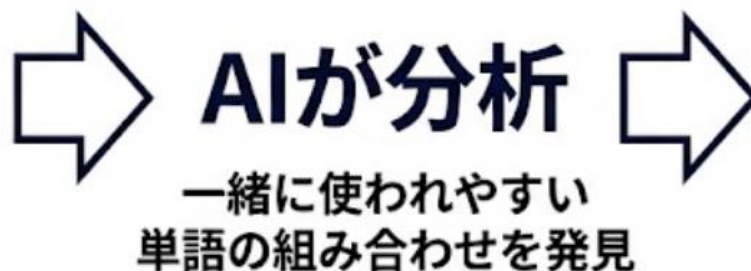
キーワードの完全一致ではなく、  
意味の近さで探す

# トピックモデリング



大量の文書から、隠れたテーマ（トピック）を自動で見つけ出す技術

たくさんの文書



テーマを自動抽出

[トピックA：スポーツ]  
選手／試合／得点

[トピックB：経済]  
株価／市場／投資

[トピックC：政治]  
政策／選挙／政府

各トピックは複数の  
関連単語で表される

文章に込められた感情を、AIが自動で判定する技術

## 文章を入力

この製品は最高でした



AIが判定

### 二値分類

2つに分ける

[ポジティブ]

[ネガティブ]

感情の強さ

ポジティブ度 0.9

### 多値分類

複数の感情に分ける

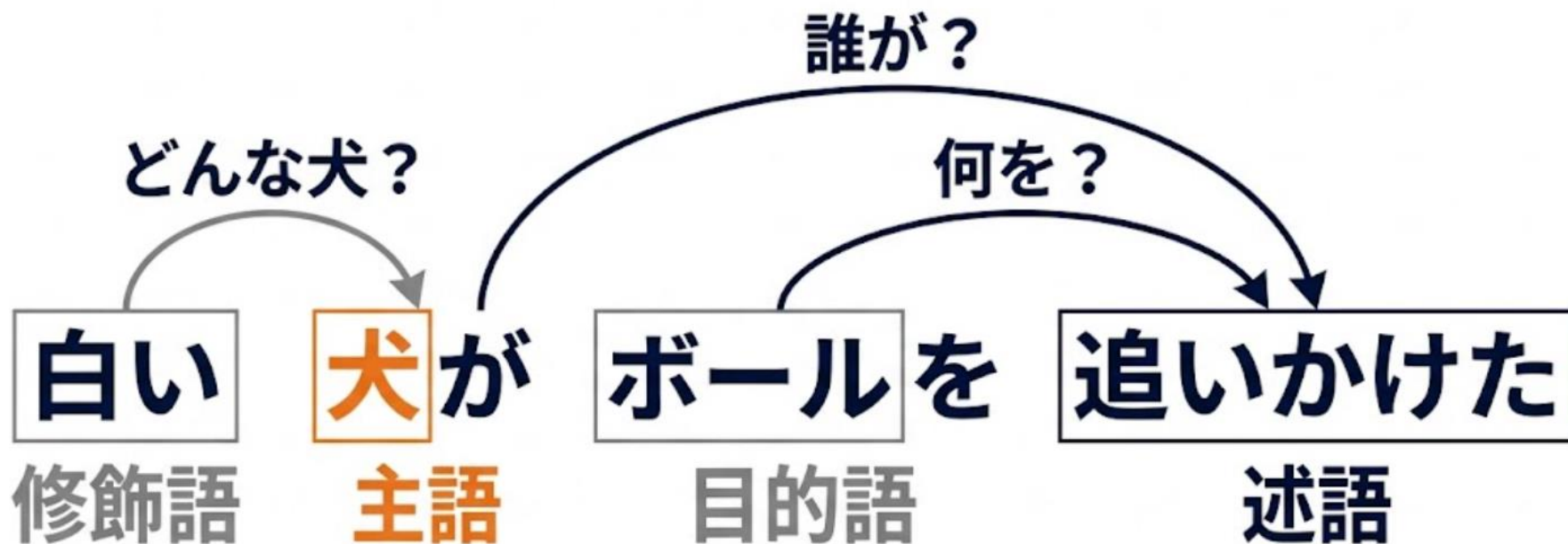
[喜び]

[怒り]

[悲しみ]

[驚き]

構文解析：文の骨組みを見つけ、どの言葉がどの言葉にかかるかを解き明かす



1. 文を部品に分ける：主語・述語・修飾語などを特定
2. 部品どうしのつながり（係り受け）を調べる

## 意味解析 — コンピュータは同じ言葉でも文脈で意味を判断する

油を売る

文脈A：料理の場面

例文：市場で 油を 売る 商人  
→ 本<sup>ほん</sup>当<sup>とう</sup>に "油" を販<sup>はん</sup>売<sup>ばい</sup>している

文脈B：仕事の場面

例文：仕事<sup>しごと</sup>中に 油<sup>あぶら</sup>を 売<sup>う</sup>る 部<sup>ぶ</sup>下<sup>か</sup>  
→ = さ<sup>さ</sup>ぼ<sup>ぼ</sup>っ<sup>て</sup>い<sup>い</sup>る (慣<sup>かん</sup>用<sup>よう</sup>句<sup>く</sup>)

① 文を読み取る → ② 前後の言葉（文脈）を確認 → ③ 正しい意味を選ぶ

同じ単語でも、まわりの言葉によって意味が変わる。この "文脈から意味を決める" 処理が意味解析。

# 演習

# 演習 1. KNP による構文解析



文を入力すると、KNPが単語のかたまり（文節）どうしの"つながり"を自動で見つけてくれる

## 1. 文を入力

クロールで  
泳いでいる  
人を見た

Webのデモサイトに文を  
打ち込む

## 2. KNPが解析



KNPの「ツリー表示」を再現。  
各文節が右端の語（「見た」）、  
に向かって係る様子。

## 3. つながりを読み取る

どの語がどの語を修飾  
するかが一目で分かる

例：『クロールで』は  
『泳いでいる』に係る  
(見た人ではなく泳ぎ方)

同じ文でも係り方が変われば意味が  
変わる。KNPは大規模データから  
確率的に最も自然なつながりを選ぶ

学べること：日本語の文は"単語の並び"ではなく"文節どうしの係り受け構造"でできている。KNPを使えば、その隠れた文の骨組みを可視化でき、意味の曖昧さ（例：『望遠鏡で泳ぐ人を見た』は誰が望遠鏡を使う？）を構造として捉えられる

# 演習 1 . KNP による構文解析

**構文解析**は、自然言語処理の一部で、文章の構造を分析し、文章内の単語の**係り受け**の解析を行う

手順

1. **構文解析** KNP のデモサイトにアクセス

<http://lotus.kuee.kyoto-u.ac.jp/nl-resource/cgi-bin/knp.cgi>

2. 自分でいくつかの文章をいれ結果を確認



KNPを試してみる

入力文: 空は青い

☐ 解析結果の詳細を表示

空は青い

# S-ID:1 KNP:4.2-407aed4a DATE:2019/06/07 SCORE:-5.37833

空は— <体言>  
青い<用言>形<格解析結果:ガ/空;ニ/>  
EOS

入力文: 白い雲と青い空が美しい

☐ 解析結果の詳細を表示

白い雲と青い空が美しい

# S-ID:1 KNP:4.2-407aed4a DATE:2019/06/07 SCORE:-25.24505

白い— <用言>形<格解析結果:ガ/雲>  
雲と (P) — <体言>  
青い— <用言>形<格解析結果:ガ/空;ニ/>  
空が (P) — PARA — <体言>  
美しい<用言>形<格解析結果:ガ/雲;ガ/空;ニ/-;デ/-;カラ/-;時間/-;ガ 2/->  
EOS

## DeepL翻訳の使い方

DeepLは文章もファイルもそのまま高精度に翻訳できる無料の翻訳ツール

### テキスト翻訳

原文を入力  
**This is a pen.**



訳文が即表示  
**これはペンです。**

言語は自動で判別・手動でも選べる  
無料版はテキスト最大5000文字まで

### ファイル翻訳

ファイルを  
ドラッグ&  
ドロップ

PDF・Word・  
PowerPoint



訳文の  
言語を  
選ぶ



レイアウトを  
保ったまま  
翻訳ファイル  
を取得

見た目や書式はそのまま維持される  
無料版はファイル翻訳が月3件まで

考察：レポートや論文の下訳に便利。ただし固有名詞や専門用語は誤訳しやすいため、必ず自分で確認・修正することが大切

# 演習 2 . DeepL 翻訳

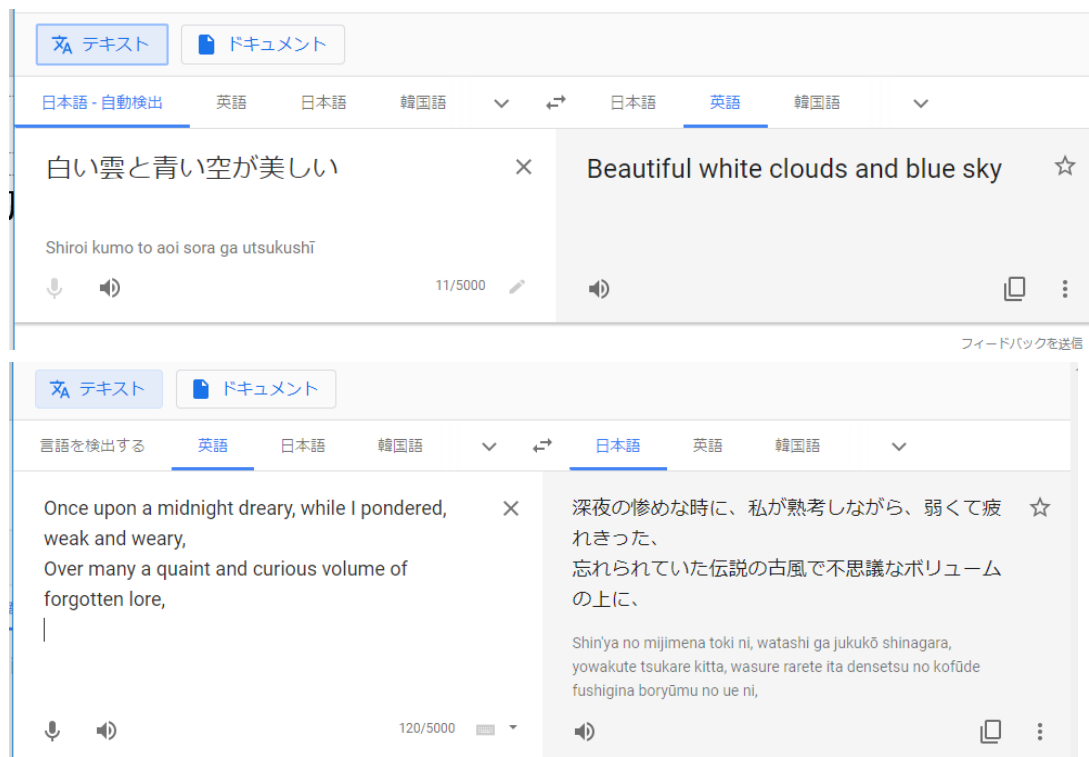


## 手順

### 1. DeepLのサイトにアクセス

<https://www.deepl.com//translator>

### 2. DeepL による**翻訳**を試してみる



# 演習3. 「AIは言葉の『感情』を理解できるか」 Transformer と Python による感情分析



文章を入力するだけで、AIがポジティブかネガティブかを自動で判定する



```
from transformers import pipeline
判定 = pipeline("sentiment-analysis")
判定("この映画は最高だった!")
```

たった3行で感情分析ができる

「感情」を数値化することで、  
コンピュータが計算できるようになる

確信度（スコア）が付くので、  
判定の自信の強さも分かる

商品レビューやSNS投稿の分析など、  
身近な場面で活用できる

# 演習 3. 「AIは言葉の『感情』を理解できるか」 Transformer と Python による感情分析

文章が「ポジティブ (positive)」か「ネガティブ (negative)」かを判定。Python プログラムのソースコードと実行結果

手順

## 1. Colab のページ

<https://colab.research.google.com/drive/1-HgN6cDnIROObU-xlVpXPiOqrVbxNkOZ?usp=sharing>

2. Google アカウントを持っている場合は、コードを変更して、別の文章を試すことができる

```
# 日本語BERTのトークナイザに必要なライブラリをインストールする
!pip install fugashi unidic-lite

# --- Transformersによる感情分析 (日本語版・否定表現対応モデル) ---
from transformers import pipeline

# 1. 感情分析パイプラインを作成 (初回実行時に日本語モデルをダウンロードする)
# 既設コーパスで学習された否定表現を含む文に対応したモデルを用いる
classifier = pipeline("sentiment-analysis", model="lm-book/bert-base-japanese-v3-wrtne-sentiment")

# 2. テキストを入力して判定
texts = [
    "プログラミングが大好きです！", # 肯定的な表現
    "今日はとても悲しいです。", # 否定的な表現
    "あまり嬉しくありません。", # 否定表現の検証用
]

results = classifier(texts)

# 3. 結果を表示
for text, result in zip(texts, results):
    print(f"文章: {text}")
    print(f"判定: {result['label']}, スコア: {result['score']:.4f}")
```

文章: プログラミングが大好きです！

判定: positive, スコア: 0.9994

文章: 今日はとても悲しいです。

判定: negative, スコア: 0.9993

文章: あまり嬉しくありません。

判定: positive, スコア: 0.6754

限界: 使用技術は「ありません」のような否定語が苦手

# 演習 4 . Webページの中の主要単語



## Webテキスト処理：ネット上の文章を集めて分析する4つの手順

こんなことに役立つ

市場動向の把握

競合製品の分析

SNSの感情分析

stopwords (ありふれた単語) を取り除く

1

ダウンロード

`urllib.request`

Webページを丸ごと  
手元に取り込む

2

タグ除去

`BeautifulSoup`

HTMLの飾りを外し、  
文章だけ残す

3

単語分割

`split()`

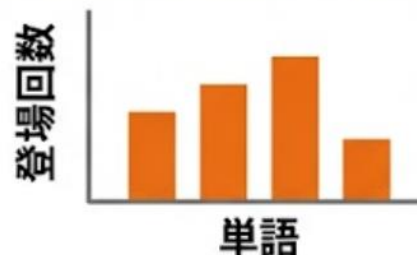
文章をハサミで  
単語に切り分ける

4

頻度分析

`nltk / FreqDist`

よく出る単語を  
数えてグラフ化



よく登場する単語が  
ひと目で分かる

# 演習4. Webページの中の主要単語



## Wikipedia の富士山の記事（自動ダウンロードされる）



ウィキペディア  
フリー百科事典





### 富士山

129 個の言語版

ページ [ノート](#)

閲覧 [ソースを編集](#) [履歴を表示](#) [☆ウォッチリストに追加](#)

出典: フリー百科事典『ウィキペディア (Wikipedia)』

座標:  [北緯35度21分38秒 東経138度43分39秒](#)<sup>[[注釈 1](#)]</sup>

この項目では、日本の山梨県・静岡県にある富士山について説明しています。

- 世界遺産としての富士山については「[富士山-信仰の対象と芸術の源泉](#)」をご覧ください。
- その他の富士山については「[富士山 \(曖昧さ回避\)](#)」をご覧ください。

「**日本最高峰**」はこの項目へ[転送](#)されています。日本が台湾を領有していた時期において日本最高峰だった台湾の新高山については「[玉山 \(台湾\)](#)」をご覧ください。



この記事には**複数の問題があります**。[改善](#)や[ノートページ](#)での議論にご協力ください。

- 出典**がまったく示されていないか不十分です。内容に関する**文献や情報源**が必要です。(2020年3月)
- 古い情報を**更新**する必要があります。(2021年3月)

出典検索?: "[富士山](#)" - [ニュース](#)・[書籍](#)・[スカラー](#)・[CiNii](#)・[J-STAGE](#)・[NDL](#)・[dlib.jp](#)・[ジャパンサーチ](#)・[TWL](#)

**富士山**（ふじさん）は、日本の**活火山**である<sup>[[注釈 2](#)]</sup>。**標高**は3775.56 m<sup>[[注釈 3](#)]</sup>、山体の最高地点は3776.12mで<sup>[[注釈 4](#)]</sup>、**日本最高峰**（**剣ヶ峰**）<sup>[[注釈 5](#)]</sup>の**独立峰**で、その優美な風貌は日本国外でも**日本の象徴**として広く知られている。**山梨県**（**富士吉田市**、**南都留郡鳴沢村**）と**静岡県**（**富士宮市**、**富士市**、**裾野市**、**御殿場市**、**駿東郡小山町**）に跨る。

**富士山**



# 演習 4 . Webページの中の主要単語



手順 :

## 1 . Colab のページ

[https://colab.research.google.com/drive/1sTPY7DBmlcFVf5UF1v81JBIO-HWujeEu?usp=drive\\_link](https://colab.research.google.com/drive/1sTPY7DBmlcFVf5UF1v81JBIO-HWujeEu?usp=drive_link)

## 2 . 結果を確認

富士山:423

. :383

年:375

[ :293

] :293

富士:252

れる:250

月:207

- :185

日:179

2:170

1:154

( :142

日本:125

):113

3:112

4:110

山:105

標高:103

m:100

<Axes: xlabel='

