aa-12. 自然言語処理

金子邦彦

「人工知能」第１２回の内容

Database Lab.

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

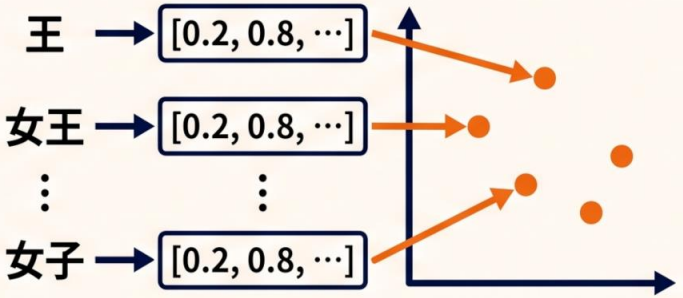
固有表現抽出／
類似度分析／
意味的検索／
トピックモデリング
感情分析

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

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

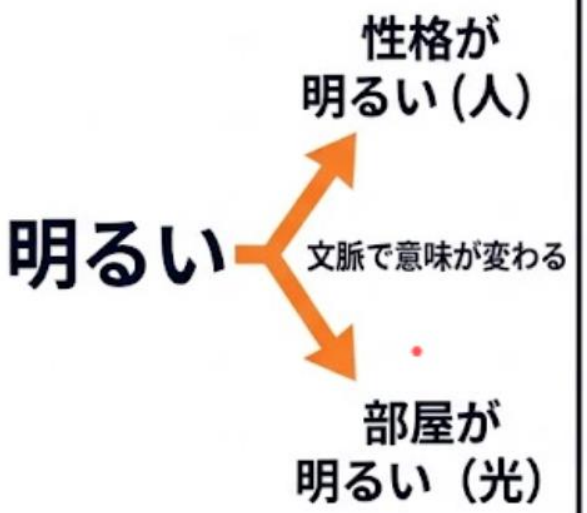
音声アシスタント
話しかけて操作する

迷惑メール判定
不要なメールを自動で分類

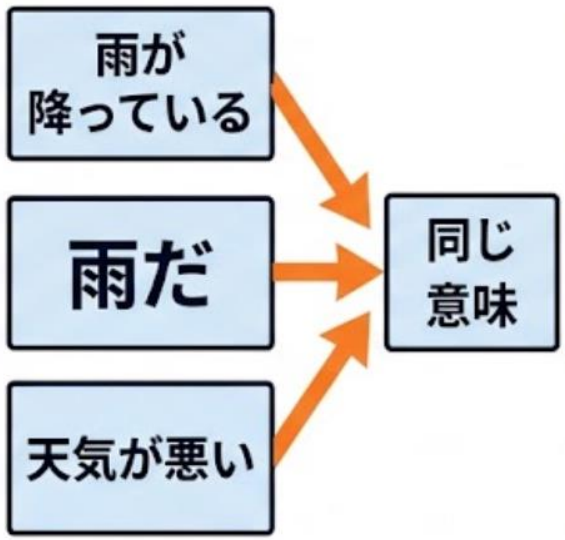
自然言語処理の難しさ

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）

Database Lab.

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

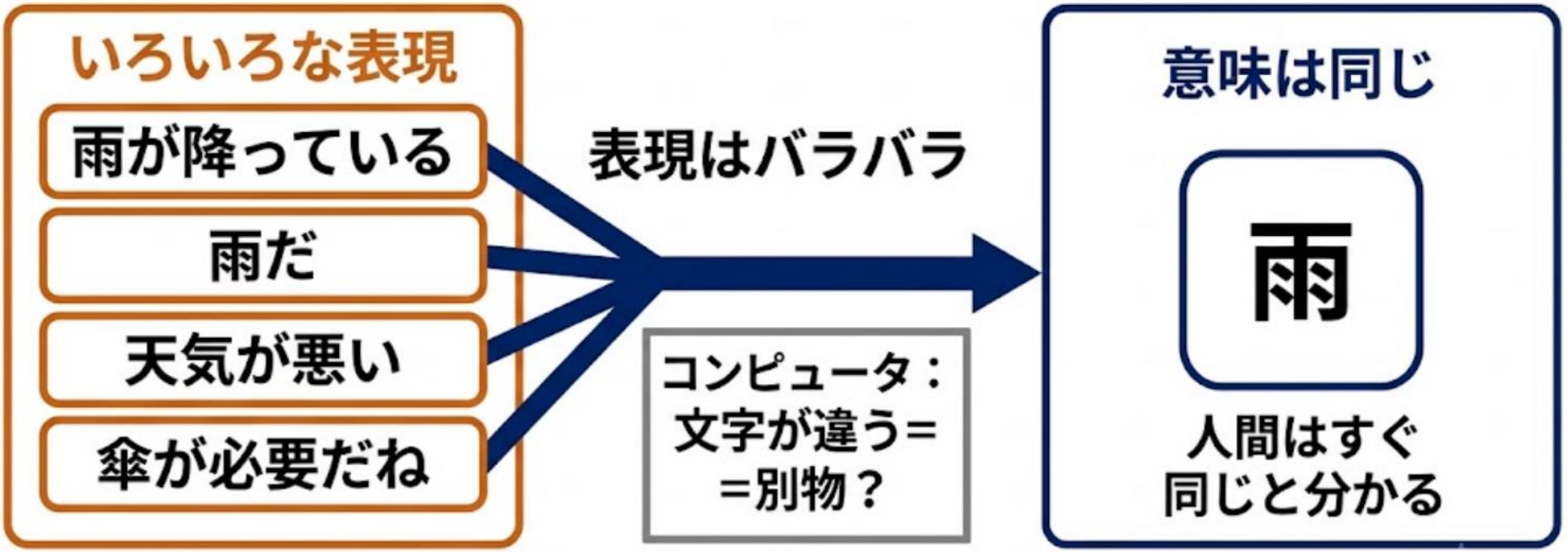
言葉は同じ → 意味は別

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

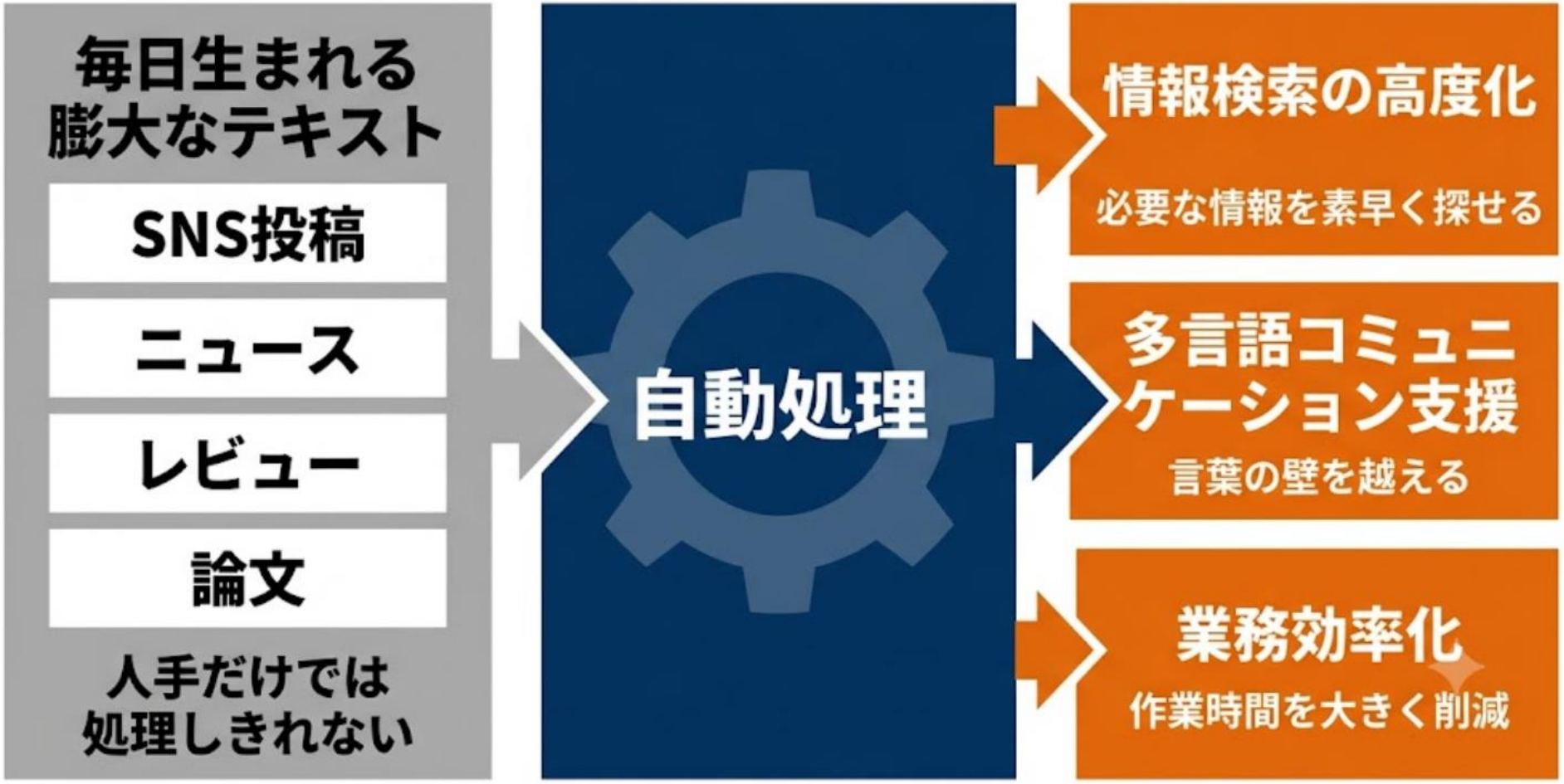
他の例

猫の手も借りたい	油を売る	Time flies.（英語）
→ 字義:動物の手を借りる ／実際:とても忙しい	→ 字義:油を販売する ／実際:仕事中に怠ける	字義:時間のハエ 実際:時が経つのは速い

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



自然言語処理の応用技術

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

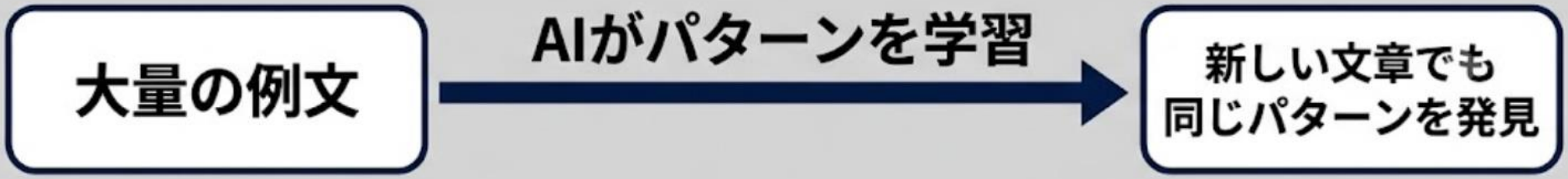
入力文 → 構文解析（構造をとらえる） → 意味解析（意味をとらえる） → 翻訳など（別の形へ変換）

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

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

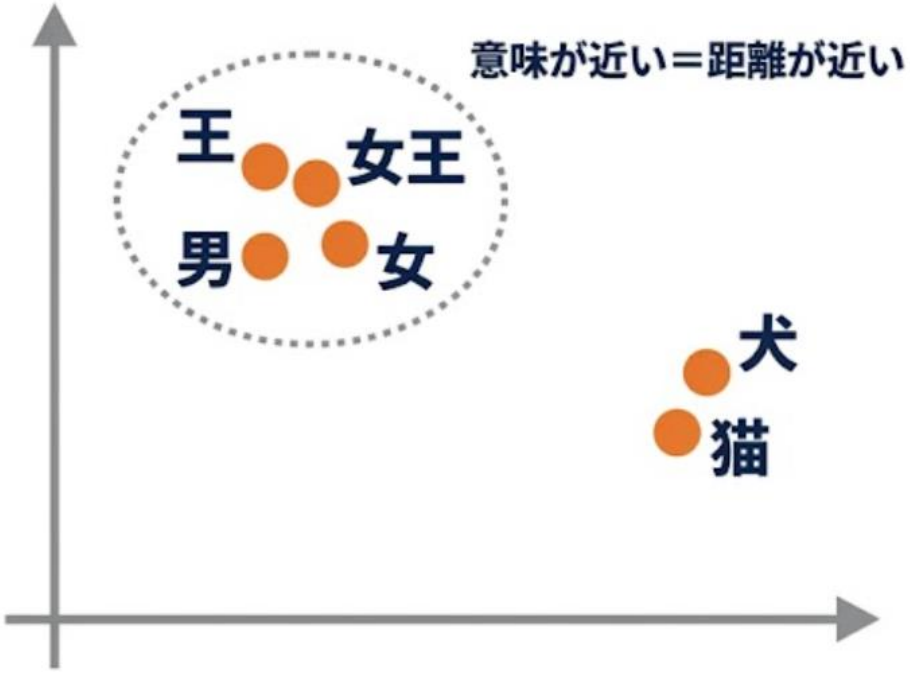


単語のベクトル化（数値の並びに変換）

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

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

文の類似度分籍と意味的検索

Database Lab.

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

0.0 ——— 1.0

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

⇒ → ⇒ 自動車の燃費

⇒ → ⇒ クルマの整備

種類の表現をことを意味

⇒ → ⇒ 果物の値段

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

AIが分析

一緒に使われやすい
単語の組み合わせを発見

テーマを自動抽出

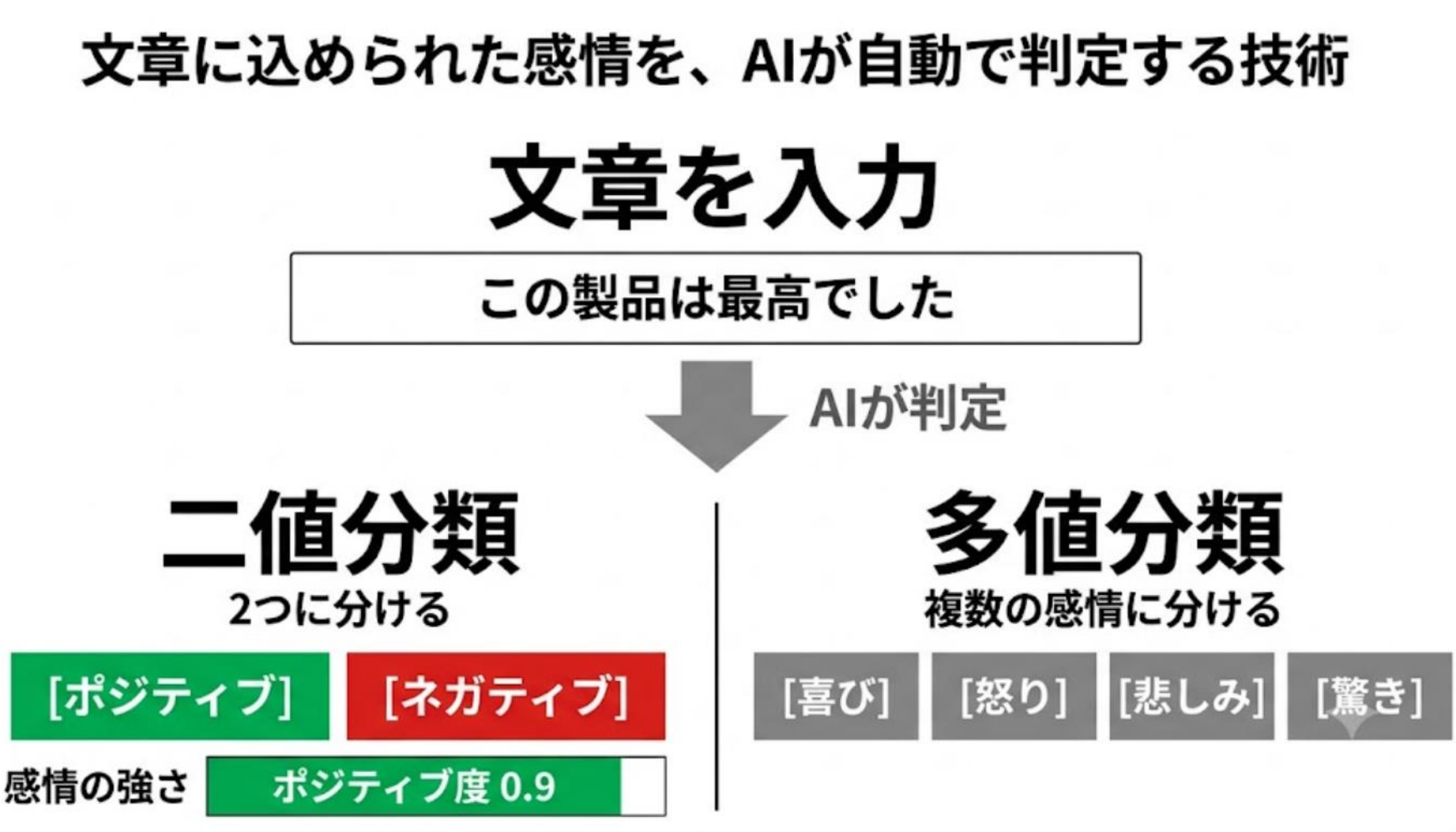
[トピックA：スポーツ]
選手／試合／得点

[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

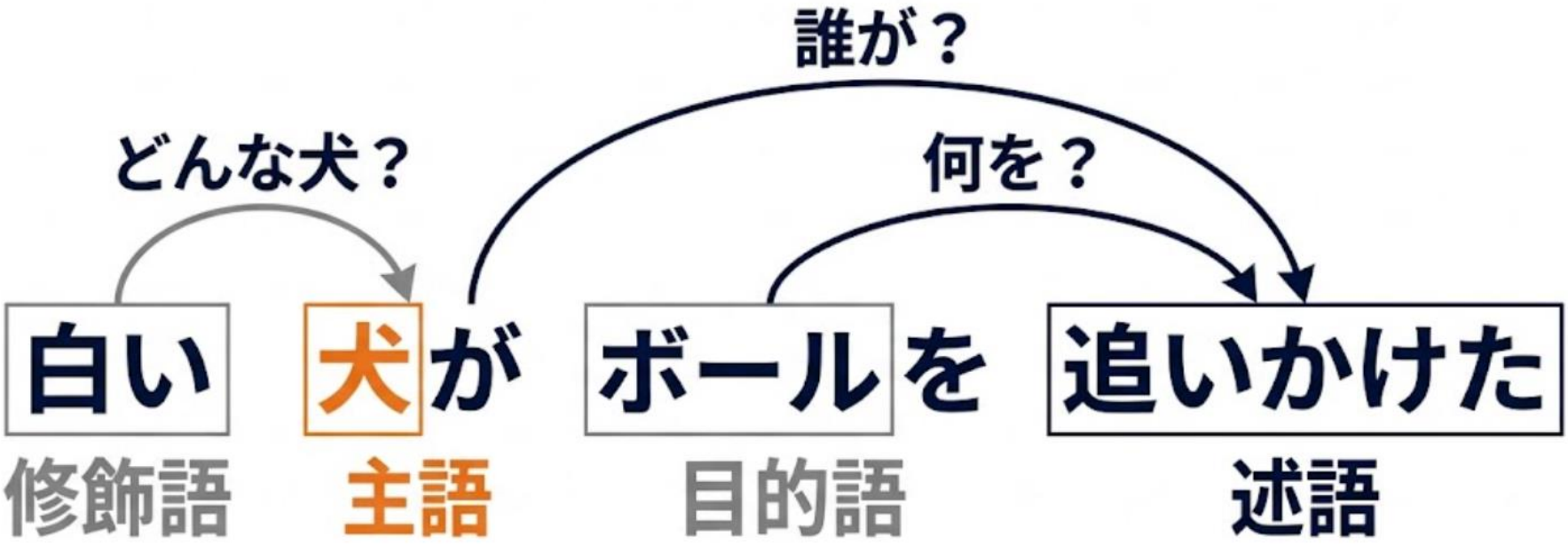
各トピックは複数の
関連単語で表される

感情分析


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]

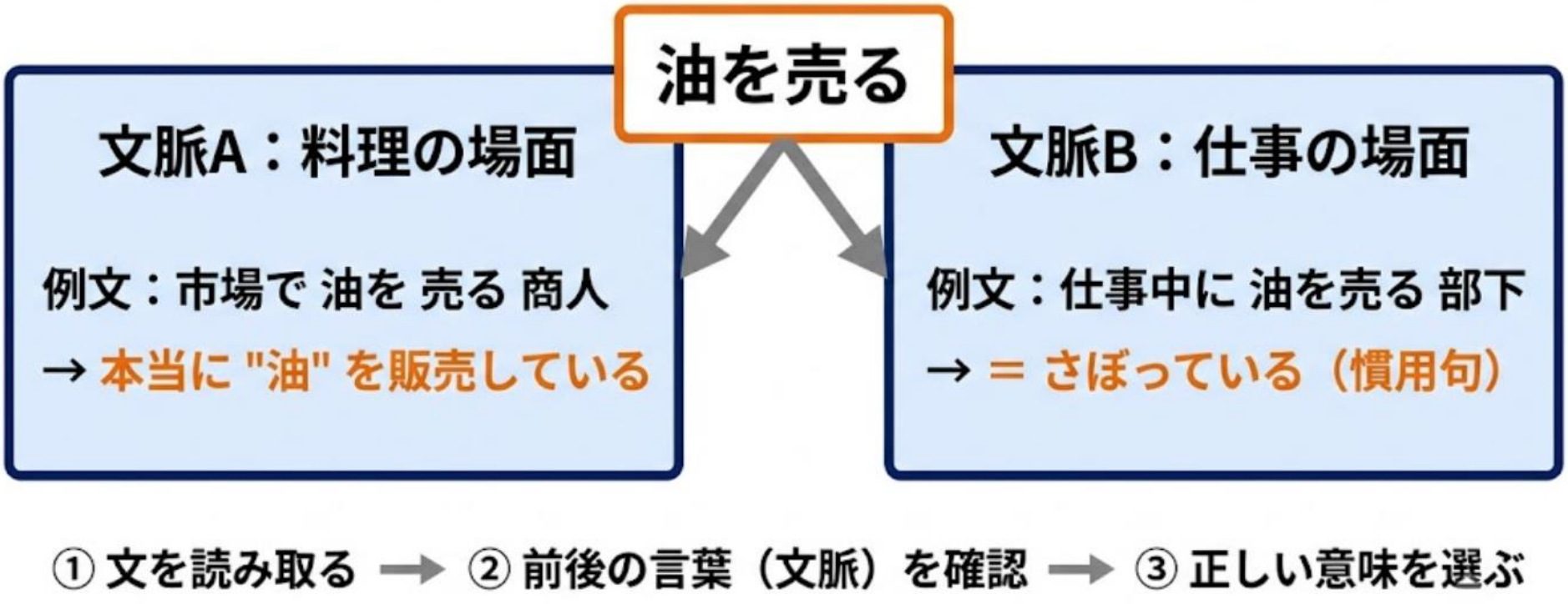
構文解析

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


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
→ 本当に "油" を販売している
文脈B：仕事の場面
例文：仕事中に 油を売る 部下
→ ＝ さぼっている（慣用句）
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

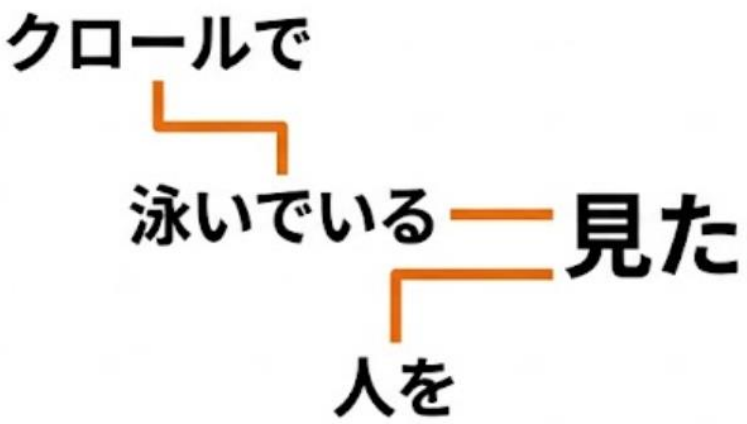
文を入力すると、KNPが単語のかたまり（文節）どうしの"つながり"を自動で見つけてくれる

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

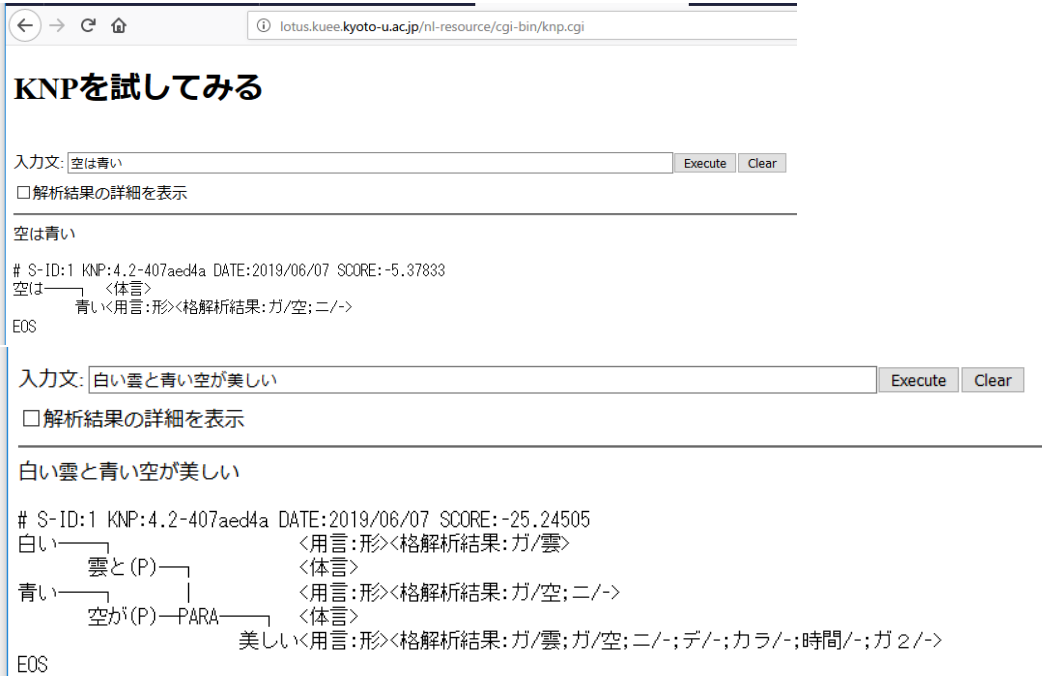
構文解析は，自然言語処理の一部で，文章の構造を分析し，文章内の単語の**係り受け**の解析を行う

手順

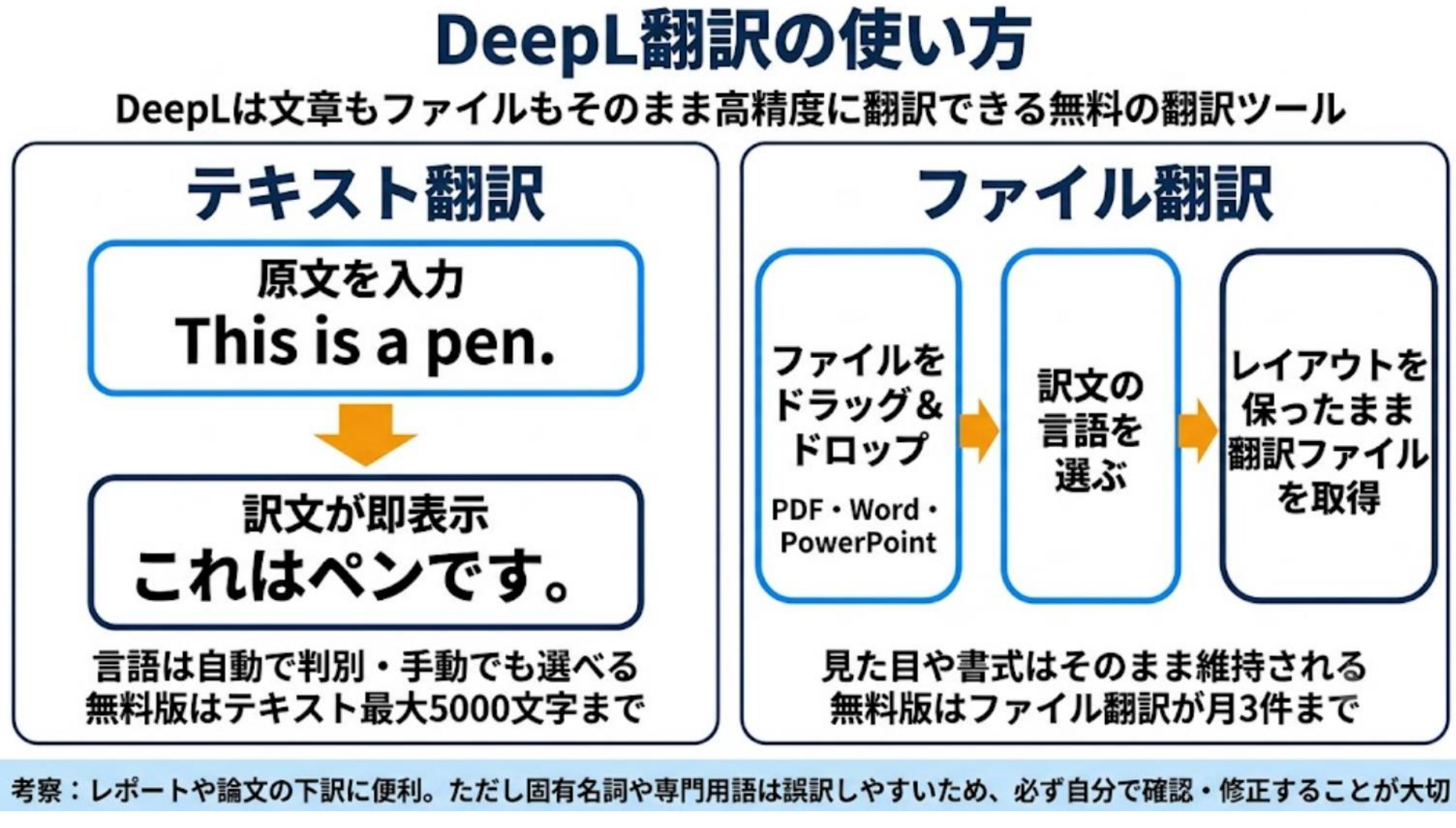
１．**構文解析** KNP のデモサイトにアクセス

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



演習２．DeepL 翻訳



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

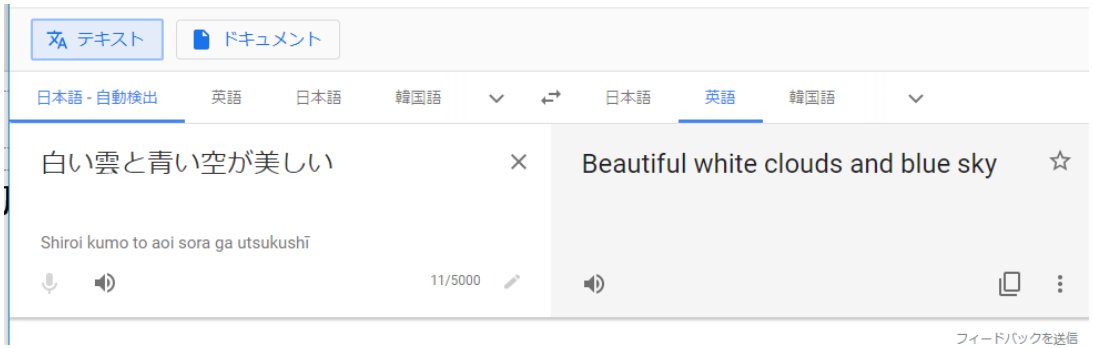
演習２．DeepL 翻訳

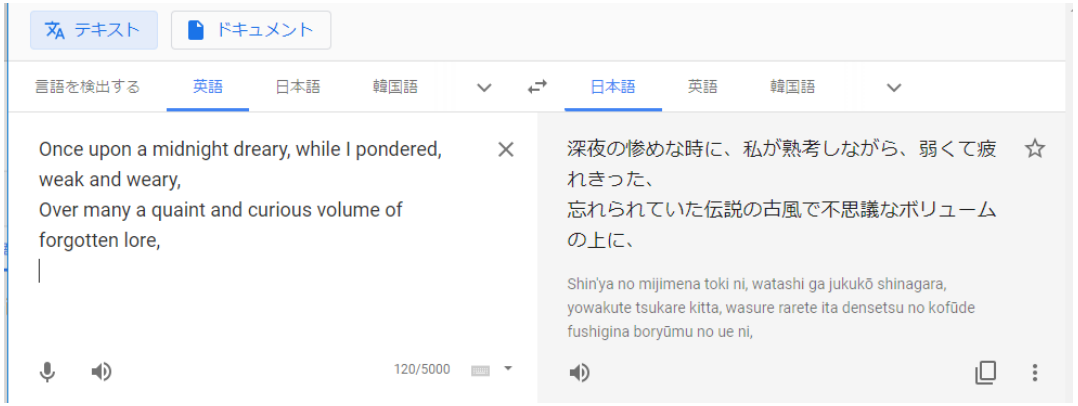
手順

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

「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-HgN6cDnIRO0bU-xIVpXPiOqrVbxNkOZ?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）
#    WRIME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="llm-book/bert-base-japanese-v3-wrime-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

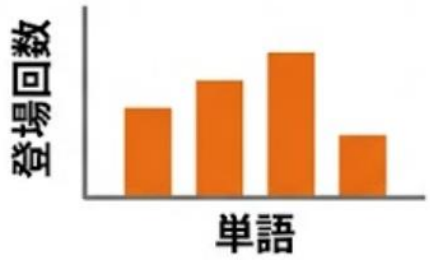
Webテキスト処理：ネット上の文章を集めて分析する4つの手順

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）

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/1sTPY7DBmlcFVf5UF1v81JBlO-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='
```