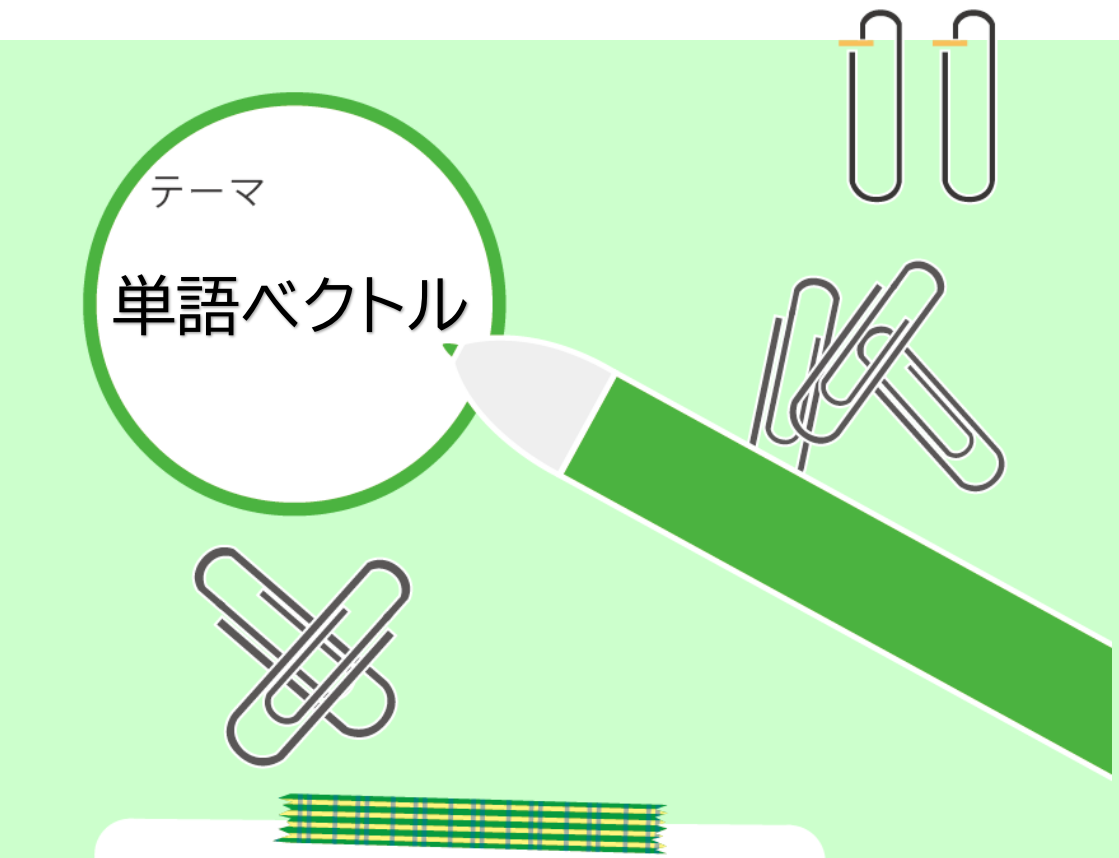
さくっと

テーマ

単語ベクトル



「さくっと」とは？
興味のある分野について、さくさくと勉強が進むように作成された調べ方ガイド（パスファインダー）です。
みなさんの学習支援を行う図書館学生サポーターが作成しました。
ぜひ学習の際に参考にしてください。

図書館学生サポーター　上野

1-1） 単語ベクトルとは？

単語ベクトルとはその名の通り，単語の意味をコンピュータが理解しやすいように数値（ベクトル）で表現したものです．大まかなイメージとしては，下図のような感じです．（値は適当です）

	リンゴ	犬	プログラム
知性	0.00	0.78	0.99
癒し	0.13	0.96	0.08
赤色	1.00	0.36	0.00
…	…	…	…

単語ベクトルに関連するキーワード

自然言語処理（NLP），生成AI, Semantic Search，機械翻訳，チャットボット，コサイン類似度，Word2Vec, Transformer

リンゴ，犬，プログラムはベクトル化したい単語です．知性，癒し，赤色 は，意味的な軸（特徴次元）と呼ばれるものになります．実際には，数百～数千もの高次元ベクトルになります．

1-2） 単語をベクトル化するメリット

一般的にコンピュータは，文字を認識しても単なる字面の解釈に過ぎず，意味や文脈（コンテキスト）まで理解することはできません．

そこで，単語をベクトル化することで，似た意味の単語が近い位置関係で表されるようになり，コンピュータが「意味の近さ」や「文の構造」を数学的に扱えるようになります．

これにより，翻訳や要約，質問応答などの自然言語処理タスクで，より柔軟かつ人間らしい理解や出力が可能になります．

従来の検索クエリ

- 「柴犬」と「秋田犬」のような部分一致検索は可能

```
SELECT * FROM animals WHERE name LIKE '%犬';
```

- 「犬」と「ワンちゃん」は，別物とみなされる．

単語ベクトル

- 意味に着目するため，「犬」と「ワンちゃん」を同義とみなせる

1-3） コサイン類似度　～意味が似ている度合いの定量化～

英単語のDog(犬)とPuppy(子犬)の意味が似ているのは自明ですが,「どのくらい似ているのか？」と問われると, 答えに困るかと思います. そこで登場するのが単語ベクトルです.
単語ベクトルなら, “似ている度合い”を定量化できます！
実際に計算してみると…

```
'dog' と 'puppy' のコサイン類似度: 0.8072
```

コサイン類似度は,

$$-1 \leq \cos(x, y) = \frac{\langle x, y \rangle}{||x||||y||} \leq 1$$

という式で計算され, −1が真逆の意味, 1が同一の意味となりますので, DogとPuppyはかなり高い度合いで意味が似ていることが定量的に分かります.

1-4） 言葉の足し算・引き算

単語ベクトルもベクトルの一種なので, 高校数学で習ったように加減算ができます.
例えば,「パリ」−「フランス」+「日本」=「東京」 といった具合です.
この演算は, パリ - フランスで『フランスに対する首都』という意味成分を抽出し, それを日本に加えることで『日本の首都』にあたる単語(＝東京)を推論するイメージです.
実際に演算してみると…

```
paris - france + japan =
  tokyo: 0.6345
```

「東京」が出てきました！

2. 学習のために

一般向けに書かれた本

・**Pythonで動かして学ぶ自然言語処理入門. 柳井孝介, 庄司美沙.**
飯塚分館, 閲覧3階　548.96 Y-100，006080151

理解を深めるための本

・**ゼロから作るDeep Learning ❷―自然言語処理編. 斎藤　康毅.**
飯塚分館, 閲覧3階 548.91 Sa252，006078615

3. 付録　実行したソースコード（Google Colabで実行できます）

```
!pip install gensim numpy scipy

# Gensimのモデルローダーを使う
from gensim.models import KeyedVectors

# 事前学習済みモデルをダウンロード
import gensim.downloader as api
print("Downloading model (これには数分かかります)...")
model = api.load('fasttext-wiki-news-subwords-300')
print("Download complete.")

# 単語ベクトルの演算を実行
result = model.most_similar(positive=["paris", "japan"], negative=["france"], topn=2)

# 結果の表示
print("paris - france + japan =")
for word, score in result:
    print(f"  {word}: {score:.4f}")

def calculate_cosine_similarity(model, word1, word2):

  try:
    similarity = model.similarity(word1, word2)
    return similarity
  except KeyError:
    print(f"'{word1}' または '{word2}' はモデルに存在しません。")
    return None

# DogとPuppyのコサイン類似度を計算
word_a = "dog"
word_b = "puppy"

similarity_ab = calculate_cosine_similarity(model, word_a, word_b)

if similarity_ab is not None:
  print(f"'{word_a}' と '{word_b}' のコサイン類似度: {similarity_ab:.4f}")
```

4. 参考資料

[1] 株式会社Spot. DeepAge. “Word2Vec：発明した本人も驚く単語ベクトルの驚異的な力”. 2016-09-2. https://deepage.net/bigdata/machine learning/2016/09/02/word2vec power of word vector.html, （参照 2025-07-03）

コードの作成には, 生成AI（ChatGPT）を使用しています.