解釈不能なAIとの対話と創作

Songs dedicated to Rock in Japan Fes. 2021

テキストマイニングの基礎編

2021/7/25

Masaki Open Lab

Outline

- テキストマイニングの基本的な考え方

- 実習：テキストの“ベクトル化”手法
  - BoW
  - TF-IDF
  - Doc2Vec

基本的な考え方①：プロセス、マテリアルを例として

品質A	反応器圧力	反応器温度	…
0.9	1.2	37.4	…
0.75	3.1	35.8	…
…	…	…	…

触媒性能B	極性	電気陰性度	…
70	0.2	0.4	…
25	0.7	3	…
…	…	…	…

・AIの動作原理：
　目的変数を説明変数（ベクトル）の関数に直す（例：品質、触媒性能予測）
　説明変数の類似度で分類する（例：これまで開発してきた触媒のマッピング）
・着目する物事をいかにうまくベクトル化するかが予測モデル、マッピングの出来を左右
　（分子：RDKit, CDK　無機物：XenonPy, etc…）

基本的な考え方②：テキストの場合

作品名	ベクトル$_1$	ベクトル$_2$	…
坊ちゃん	0.4	4.4	…
斜陽	0.05	15.9	…
…	…	…	…

・テキストの場合も基本は同じ→文章をどのようにして「ベクトル化するか」がカギ
・ベクトル化さえできてしまえば、色んな事ができる
　（例：小論文の自動採点、類似度の計算）
・テキストマイニング技術の進化は、ベクトル化技術の進化とも言える

実習：テキストのベクトル化手法

BoW (Bag of Words)

文章	私	は	犬	が	好き	だ	彼	猫
私は犬が好きだ	1	1	1	1	1	1	0	0
彼は猫が好きだ	0	1	0	1	1	1	1	1
…	…	…						

・全ての文章を単語に分割し、その出現頻度でベクトル化
　〇単純で簡便、その割に文章の特徴をとらえやすい
　×助詞や助動詞など、出現頻度の高い成分に引っ張られる
　　語順の違いをベクトルに反映することは出来ない

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Pythonによる実装例（scikit-learnからコピペ）

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

https://ja.wikipedia.org/wiki/Tf-idf

TF-IDF

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \cdot \mathrm{idf_i}$$

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d : d \ni t_i\}|}$$

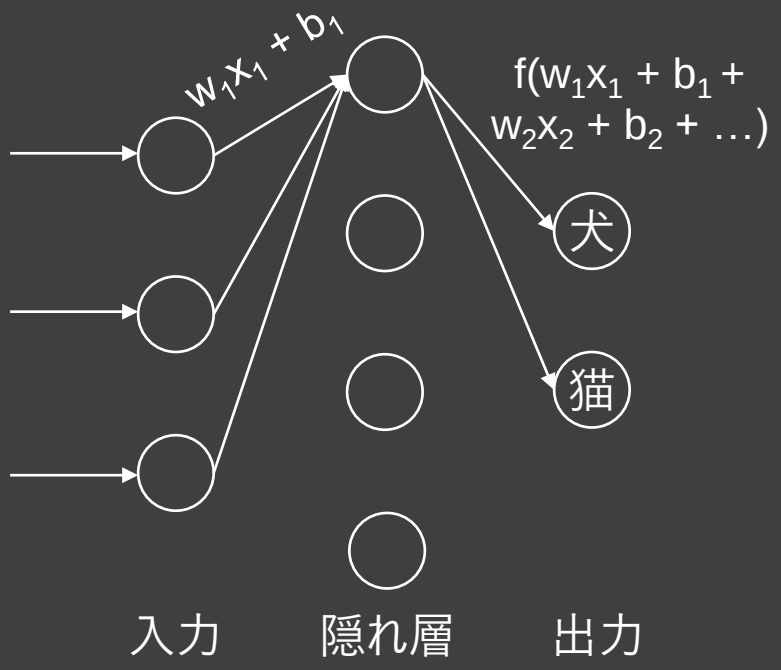
・tf：文章における着目単語の出現頻度
・idf：log（着目単語を含む文の数 / 文章を構成する文の数）$^{-1}$
・tf-idf：その単語がどれだけ「重要」かつ「特別」であるかを示す
　〇比較的単純かつ簡便、助詞や助動詞に引っ張られづらい
　×：語順の違いをベクトルに反映することはできない

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer

Pythonによる実装例（scikit-learnからコピペ）

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.shape)
(4, 9)
```

Doc2Vec導入：ニューラルネットワークの話

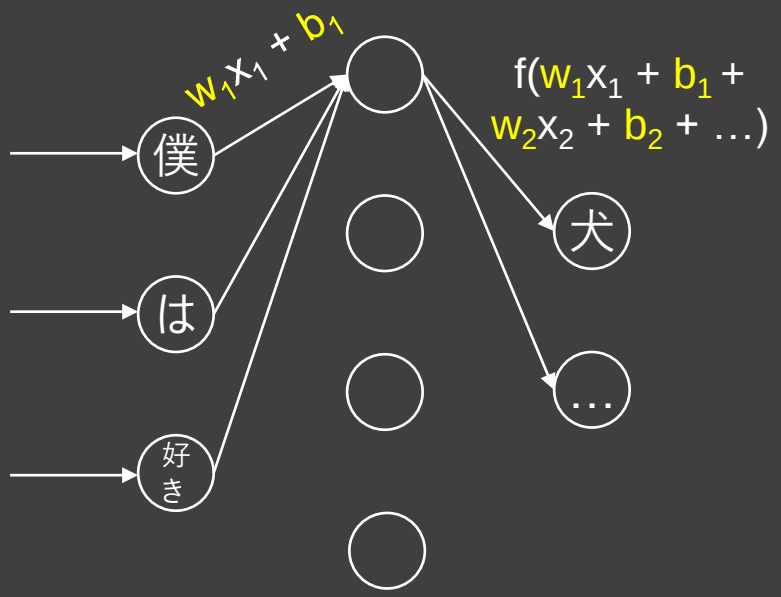


ラベル	出力
犬度	0.97
猫度	0.03

・神経ニューロンを模擬した予測モデル、非線形な判別、回帰が可能
・入力ベクトルに重みwを掛け、バイアスbを和したものを隠れ層に代入
・隠れ層ではf（シグモイド関数）による変換が行われ、出力層へ

https://arxiv.org/pdf/1405.4053.pdf

Doc2Vec（2013）



文章	w_1	w_2	…	b_1	b_2	…
私は犬が好きだ	0.2	0.7	…	0.1	0.4	…
彼は猫が好きだ	0.5	0.6	…	0.8	0.7	…
…	…	…	…	…	…	…

・以下2つの予測モデルを、ニューラルネットワークにて構築する

・文中の単語を一つ抜いたもの＋文の位置を入力とし、抜けた単語を予測

・単語と文の位置を入力とし、その前後に来やすい単語群を予測

★ニューラルネットワークのパラメータ（w, b）をその文章の構成ベクトルとする

→文脈や単語の順序をベクトルに反映することができる

https://github.com/M-asaki-K/text_mining_sample/blob/main/20210725_%E6%AD%8C%E3%82%BD%E3%83%A0%E3%83%AA%E3%82%A8doc2vec_github.py

Pythonによる実装例（歌ソムリエより引用）

```
"""
Doc2Vecを適用するため、各文章を単語ごとにリスト化します。
"""
sentences = []
for text in train_data_df_wakati[0]:
    text_list = text.split(' ')
    sentences.append(text_list)

"""
Doc2Vecモデルにより、各歌詞に対するニューラルネットワークを学習し、ハイパーパラメータにてベクトル化を行います。
よく見ると色々 調整すべきパラメータがあるので、まだ詰め代はあるかもしれません
"""
documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(sentences)]
model = Doc2Vec(documents, vector_size=2, window=5, min_count=1, workers=4)

"""
0行目に格納されたテキストに対し、最も似ているものの順に歌詞をランキングします。
"""
res = model.docvecs.most_similar(0)
```

https://www.cse.kyoto-su.ac.jp/~g0846020/keywords/cosinSimilarity.html

補足：類似度

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\cdot\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

・ベクトル同士の類似度はcos類似度を用いることが多い
・cos類似度：2ベクトルの内積 / ノルム積、ベクトル同士の角度（cos）に相当
・1に近いほど類似度が大きい