ベイズ推論の例: 簡単なテキスト解析「森鴎外か夏目漱石か」

Author: 藤原 義久 yoshi.fujiwara@gmail.com (mailto:yoshi.fujiwara@gmail.com)
Data: ディレクトリ"data"以下

- 青空文庫 (https://www.aozora.gr.jp/)にある森鴎外と夏目漱石の作品抜粋

```
data/docs/01.txt | 森鴎外『雁』
data/docs/02.txt | 森鴎外『かのように』
data/docs/03.txt | 森鴎外『鶏』
data/docs/04.txt | 森鴎外『ヰタ・セクスアリス』
data/docs/05.txt | 夏目漱石『永日小品』
data/docs/06.txt | 夏目漱石『硝子戸の中』
data/docs/07.txt | 夏目漱石『思い出す事など』
data/docs/08.txt | 夏目漱石『夢十夜』
```

形態素解析のツールmecab

- mecab 本家: https://taku910.github.io/mecab/ (https://taku910.github.io/mecab/)
- mecab 自体のインストール
  - Windows: https://github.com/ikegami-yukino/mecab/releases/tag/v0.996 (https://github.com/ikegami-yukino/mecab/releases/tag/v0.996)
    「MeCab 0.996 64bit version (旧)」にあるインストーラ"mecab-0.996-64.exe"を使う
    注：環境変数 PATH にmecab をインストールしたパスを追加(例: C:\w10\mecab\bin)
    注：環境変数 MECABRC を新たに追加して設定(例: C:\w10\mecab\etc\mecabrc)
  - Linux: "mecab linux"でググるとUbuntu, CentOS などでのインストール方法が分かる
  - Mac: "mecab mac"でググる(未確認)
- python からmecab を使うパッケージのインストール
  jupyter notebook を新規に開いて以下を実行する(先頭の"!"に注意)
  - Windows:

```
!pip install mecab-python-windows
```

  - Linux:

```
!pip install mecab-python3
```

  - Mac: Linuxと同じ(?)

パッケージの読み込み

In [1]:
```
import MeCab
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
import glob
```

準備：形態素解析

In [2]:
```
# tagger = MeCab.Tagger()  # Windows の場合
tagger = MeCab.Tagger("-r /etc/mecabrc")  # Linux の場合
```

準備：名詞、動詞、形容詞、助詞だけを選んで文書の「ダイジェスト」を作る

In [3]:
```
# 名詞、動詞、形容詞、助詞だけを選んで文書の「ダイジェスト」を作る関数を定義

def digest_doc(filename):
    with open(filename, encoding="utf-8") as fin:
        s = fin.read()
        node = tagger.parseToNode(s)

    l = []
    while node:
        x = node.feature.split(',')[0]
        if x == "名詞" or x == "動詞" or x == "形容詞" or x == "助詞":
            l.append(node.feature.split(',')[6])  # 原形を用いる
        node = node.next

    return " ".join(l)
```

すべての文書について処理

In [4]:
```
docs = []
for fn in sorted(glob.glob("data/docs/*")):
    print(fn)
    d = digest_doc(fn)
    docs.append(d)
```

```
data/docs/01.txt
data/docs/02.txt
data/docs/03.txt
data/docs/04.txt
data/docs/05.txt
data/docs/06.txt
data/docs/07.txt
data/docs/08.txt
```

すべての文書について語とその頻度の表を作る

メモ：テキスト解析で、語とその頻度の表はterm-frequency matrix と呼ばれている
以下では機械学習の学習用パッケージ scikit-learn からテキスト解析のツールを用いる

In [5]:
```
count_vec = CountVectorizer()

x = count_vec.fit_transform(np.array(docs))
# 疎な行列として扱われている
# print(type(X))

td = x.toarray()  # term-document matrix

# 出現したすべての語のリスト
terms = count_vec.get_feature_names()
print(terms)

# term-frequency matrix の次元 = 文書数 * 全語数
print(td.shape)

# term-frequency の中身
print(td)
```

['あい', 'あう', 'あか', 'あかるい', 'あがる', 'あきらめる', 'あきれる', 'あく', 'あける', 'あざやか', 'あすこ', 'あそこ', 'あたり', 'あちこち', 'あっけない', 'あっち', 'あて', 'あてがう', 'あてる', 'あと', 'あながち', 'あなた', 'あびせる', 'あやす', 'あやぶむ', 'あらあらしい', 'あらわれる', 'あり', 'ありがたい', 'ある', 'あるく', 'あれ', 'あんた', 'いい', 'いう', 'いえる', 'いかん', 'いく', 'いくつ', 'いくら', 'いける', 'いさむ', 'いじくる', 'いじる', 'いずれ', 'いたす', 'いただく', 'いただける', 'いたわる', 'いた事', 'いち', 'いっさい', 'いっしょ', 'いっす', 'いつ', 'いつか', 'いぬ', 'いも', 'いや', 'いやいや', 'いやがる', 'いやしい', 'いらいら', 'いらう', 'いらっしゃい', 'いらっしゃる', 'いる', 'いるか', 'いろいろ', 'うかがう', 'うさ', 'うさんい', 'うすい', 'うずくまる', 'うずめる', 'うそ', 'うた', 'うたう', 'うち', 'うっとうしい', 'うねる', 'うの', 'うぶ', 'うまい', 'うむ', 'うる', 'うるさい', 'うろ覚え', 'ええ', 'えな', 'えらい', 'えり', 'える', 'おいで', 'おう', 'おおい', 'おかしい', 'おく', 'おくれる', 'おこす', 'おこる', 'おさまる', 'おしまい', 'おじ', 'おっくう', 'おつもり', 'おと', 'おとなしい', 'おなり', 'おば', 'おばさん', 'おまわりさん', 'おもちゃ', 'おりる', 'おる', 'おれ', 'おろか', 'おろす', 'お上', 'お世話', 'お作り', 'お前', 'お嬢さん', 'お客', 'お方', 'お次', 'お正月', 'お歯黒', 'お母さん', 'お母様', 'お玉', 'お目見え', 'お祭', 'お笑い', 'お蔭', 'お辞儀', 'かい', 'かう', 'かお', 'かかる', 'かかわる', 'かく', 'かける', 'かしく', 'かしぐ', 'かしら', 'かじかむ', 'かじる', 'かす', 'かた', 'かたまる', 'かつ', 'かな', 'かね', 'かねる', 'かのう', 'かぶせる', 'かぼちゃ', 'かも', 'から', 'からから', 'かる', 'かるい', 'かわす', 'かん', 'がかる', 'がた', 'がらがら', 'がり', 'がる', 'きが', 'きたない', 'きの', 'きまり', 'きまる', 'きめる', 'きゃしゃ', 'きょう', 'きり', 'きりなし', 'くい', 'くし', 'くせ', 'くっつく', 'くに', 'くべる', 'くも', 'くらい', 'くり', 'くりくり', 'くる', 'くるむ', 'くるめる', 'くれる', 'ぐらい', 'ぐらつく', 'ぐる', 'ぐるり', 'けしからん', 'けた', 'けたたましい', 'けち', 'けり', 'ける', 'けれども', 'けんつく', 'げんげ', 'こいつ', 'こう', 'こける', 'ここ', 'こしらえる', 'こじつける', 'こそ', 'こそばゆい', 'こちら', 'こっち', 'こと', 'ことわり', 'ことわる', 'こねる', 'この間', 'こびりつく', 'こびり付く', 'こむ', 'こめかみ', 'こりる', 'こる', 'これ', 'これら', 'ころ', 'ころり', 'こわい', 'こわれる', 'ござる', 'ごと', 'ごまかす', 'ごろ', 'さえ', 'さえる', 'さし', 'さす', 'さする', 'させる', 'さっき', 'さておく', 'さばける', 'さびしい', 'さま', 'さらう', 'さる', 'さん', 'さんざん', 'ざす', 'ざる', 'ざわつく', 'しいる', 'しか', 'しかける', 'しかる', 'しきる', 'しく', 'したたか', 'しだい', 'してやる', 'しまう', 'しまる', 'しめる', 'しもた屋', 'しゃがむ', 'しゃくる', 'しゃくれる', 'しゃべる', 'しょ', 'しよう', 'しようが', 'しん', 'しんみり', 'じい', 'じいさん', 'じき', 'じゃ', 'じゃあ', 'じゅう', 'じれったい', 'じん', 'すう', 'すか', 'すかす', 'すくう', 'すすめる', 'すべて', 'すます', 'すむ', 'すら', 'する', 'ずく', 'ずくめ', 'ずつ', 'ずる', 'せい', 'せがむ', 'せき', 'せしめる', 'せっつく', 'せっぱつまる', 'せまい', 'せる', 'せん', 'せんだって', 'そう', 'そこ', 'そちら', 'そっけ', 'そっち', 'そのもの', 'その他', 'その後', 'その間', 'その頃', 'そり', 'そりゃあ', 'それ', 'それら', 'そん', 'たかだか', 'たかる', 'たぎる', 'たくさん', 'たくる', 'たけ', 'たしか', 'たしかめる', 'ただ', 'たち', 'たって', 'たつ', 'たでる', 'たなびく', 'たび', 'たま', 'たまらない', 'たまる', 'ため', 'ためる', 'たやすい', 'たより', 'たり', 'たる', 'たんび', 'だい', 'だけ', 'だって', 'だの', 'だまくらかす', 'だまし', 'だまる', 'だめ', 'だり', 'だれ', 'ちい', 'ちぎる', 'ちぢめる', 'ちゃ', 'ちゃあ', 'ちゃきちゃき', 'ちゃぼ', 'ちゃり', 'ちゃん', 'ちらつく', 'ちる', 'っけ', 'って', 'っぽい', 'ついで', 'つかる', 'つき', 'つぎ', 'つく', 'つくせる', 'つけ', 'つける', 'つっけんどん', 'つつ', 'つつく', 'つつましい', 'つづく', 'つづける', 'つぶやく', 'つまらない', 'つまる', 'つめる', 'つもり', 'つれる', 'づく', 'づらい', 'てい', 'てっぺん', 'てる', 'できる', 'でも', 'でる', 'とい', 'という', 'といった', 'とか', 'とかいう', 'とき', 'とぎれとぎれ', 'とぎれる', 'とく', 'ところ', 'として', 'とっさ', 'ととのえる', 'とともに', 'とまる', 'とも', 'とら', 'とり', 'とる', 'とんび', 'と共に', 'どく', 'どこ', 'どす黒い', 'どちら', 'どっち', 'どてら', 'どの人', 'ども', 'どる', 'どれ', 'なあ', 'ない', 'なか', 'ながら', 'なくす', 'なくなる', 'なさる', 'なす', 'なする', 'なぞ', 'なだれ', 'など', 'なに', 'なり', 'なる', 'なれる', 'なん', 'なんか', 'なんぞ', 'なんの', 'にあたる', 'において', 'における', 'にくい', 'にこにこ', 'について', 'につれて', 'にとって', 'にゃ', 'によって', 'に対して', 'に対する', 'に従って', 'に関する', 'ぬき', 'ぬく', 'ねえ', 'ねだる', 'ねむたい', 'ねる', 'のう', 'のす', 'のっける', 'ので', 'のに', 'のぶ', 'のみ', 'のめる', 'のる', 'はいる', 'はおる', 'はか', 'はかない', 'はかばかしい', 'はぐ', 'はじける', 'はじめ', 'はじめる', 'はず', 'はずれ', 'はずれる', 'はせる', 'はだし', 'はなし', 'はにかむ', 'はね返る', 'ばかし', 'ばかり', 'ばっかり', 'ばった', 'ひく', 'ひす', 'ひそめる', 'ひとり', 'ひどい', 'ひびく', 'ひろげる', 'びっくり', 'びり', 'ふう', 'ふく', 'ふくらます', 'ふざける', 'ふち', 'ふっくり', 'ふむ', 'ふり', 'ふりしきる', 'ふる', 'ふるう', 'ぶつ', 'ぶつける', 'ぶらつく', 'ぶら下がる', 'ぶら下げる', 'ぶり', 'ぶり返す', 'ぶる', 'へる', 'へん', 'べた', 'ほう', 'ほか', 'ほど', 'ほどよい', 'ほる', 'ほん', 'ぼける', 'ぼる', 'ぼんやり', 'ぽい', 'まいる', 'まう', 'まえる', 'まぎらわしい', 'まさる', 'まし', 'まじる', 'ます', 'まずい', 'まつ', 'まで', 'まとも', 'まま', 'まめ', 'まり', 'まるい', 'まれ', 'まわり', 'まわる', 'まんま', 'みごと', 'みる', 'みんな', 'むき', 'むく', 'むす', 'むずかしい', 'むせる', 'むつかしい', 'むやみ', 'めく', 'めす', 'めちゃめちゃ', 'めった', 'めでたい', 'めでる', 'もうこ', 'もった

In [6]:
```
# pandas のデータフレームに変換する

df_td = pd.DataFrame(data=td, columns=terms)
df_td
```

Out[6]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	…	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
0	0	0	0	0	1	1	1	0	1	0	…	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	…	0	0	1	2	0	0	1	0	0	0
2	0	0	0	0	0	1	0	0	0	0	…	0	0	0	7	0	0	0	0	0	0
3	1	1	0	0	0	0	0	1	0	0	…	0	0	0	0	0	1	0	0	0	0
4	2	0	0	1	1	0	0	0	0	1	…	0	0	0	1	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	0	0	…	0	0	0	6	1	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	…	1	0	0	1	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	…	0	1	0	6	0	0	0	0	1	0

8 rows × 5436 columns

In [7]:
```
# 1番目の文書(森鴎外)について、出現頻度によって語をソート
i = 0
df_td[i:i+1].sort_values(by=i, axis=1, ascending=False)
```

Out[7]:

	する	いる	云う	なる	から	ある	よう	見る	こと	それ	…	家中	宵の口	実生活	害する	宮本	室町	室内	宣告	実行	鼻糞
0	217	190	117	92	89	78	61	52	52	44	…	0	0	0	0	0	0	0	0	0	0

1 rows × 5436 columns

In [8]:
```
# 5番目の文書(夏目漱石)について、出現頻度によって語をソート
i = 4
df_td[i:i+1].sort_values(by=i, axis=1, ascending=False)
```

Out[8]:

	いる	する	から	自分	云う	よう	ある	来る	なる	出る	…	一筋	愛す	愛らしい	愛宕	愛情	愛想	愛敬	愛読	一瞥	鼻糞
4	201	163	104	91	82	53	52	48	47	37	…	0	0	0	0	0	0	0	0	0	0

1 rows × 5436 columns

In [9]:
```
# 各文書について、頻度の合計を計算
df_td.sum(axis=1)
```

Out[9]:
```
0    3784
1    3923
2    3813
3    3744
4    3717
5    3670
6    3807
7    3429
dtype: int64
```

「学習用」データを作る

In [10]:
```
df_td_train = df_td.iloc[[0,1,4,5]]  # 0,1(森鴎外); 4,5(夏目漱石)
df_td_train
```

Out[10]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	…	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
0	0	0	0	0	1	1	1	0	1	0	…	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	…	0	0	1	2	0	0	1	0	0	0
4	2	0	0	1	1	0	0	0	0	1	…	0	0	0	1	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	0	0	…	0	0	0	6	1	0	0	0	1	0

4 rows × 5436 columns

各作家について，語ごとの出現頻度を合計する

In [11]:
```
# 森鴎外
x = df_td_train[0:2].sum()
freq_ogai = pd.DataFrame(x,columns=["ogai"]).transpose()
freq_ogai
```

Out[11]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	…	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
ogai	0	0	0	0	1	1	1	0	1	0	…	0	0	1	2	0	0	1	0	1	0

1 rows × 5436 columns

In [12]:
```
# 夏目漱石
x = df_td_train[2:4].sum()
freq_soseki = pd.DataFrame(x,columns=["soseki"]).transpose()
freq_soseki
```

Out[12]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	…	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
soseki	2	0	1	1	1	0	0	0	0	1	…	0	0	0	7	1	0	0	1	1	0

1 rows × 5436 columns

In [13]:
```
# データフレームを連結する
freqs_df = pd.concat([freq_ogai, freq_soseki])
freqs_df
```

Out[13]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	...	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
ogai	0	0	0	0	1	1	1	0	1	0	...	0	0	1	2	0	0	1	0	1	0
soseki	2	0	1	1	1	0	0	0	0	1	...	0	0	0	7	1	0	0	1	1	0

2 rows × 5436 columns

In [14]:
```
# 各文書について、語ごとの出現確率を計算
# (1) 頻度が0の場合は確率も0とするナイーブな方法

freqs = np.array(freqs_df, np.float)
freq_sums = np.array(freqs_df.sum(axis=1), np.float).reshape(2,1)  # For num
py's broadcast

probs = freqs / freq_sums

for i in range(probs.shape[1]):
    print("%s\t%f\t%f" % (terms[i],probs[0,i],probs[1,i]))
```

あい	0.000000	0.000271
あう	0.000000	0.000000
あか	0.000000	0.000135
あかるい	0.000000	0.000135
あがる	0.000130	0.000135
あきらめる	0.000130	0.000000
あきれる	0.000130	0.000000
あく	0.000000	0.000000
あける	0.000130	0.000000
あざやか	0.000000	0.000135
あすこ	0.000000	0.000000
あそこ	0.000000	0.000000
あたり	0.000779	0.000135
あちこち	0.000000	0.000000
あっけない	0.000000	0.000000
あっち	0.000260	0.000000
あて	0.000130	0.000000
あてがう	0.000000	0.000135
あてる	0.000000	0.000000
あと	0.000000	0.001083
あながち	0.000000	0.000000
あなた	0.000000	0.002978
あびせる	0.000130	0.000000
あやす	0.000000	0.000000
あやぶむ	0.000130	0.000000
あらあらしい	0.000130	0.000000
あらわれる	0.000000	0.000135
あり	0.000130	0.000000
ありがたい	0.000000	0.000271
ある	0.018036	0.012996
あるく	0.000000	0.000000
あれ	0.000519	0.000135
あんた	0.000000	0.000000
いい	0.000000	0.000677
いう	0.000000	0.002031
いえる	0.000000	0.000135
いかん	0.000000	0.000000
いく	0.000260	0.000000
いくつ	0.000000	0.000135
いくら	0.000000	0.000000
いける	0.000260	0.001083
いさむ	0.002076	0.000000
いじくる	0.000000	0.000135
いじる	0.000130	0.000000
いずれ	0.000000	0.000135
いたす	0.000130	0.000000
いただく	0.000000	0.000812
いただける	0.000000	0.000135
いたわる	0.000130	0.000000
いた事	0.000260	0.000000
いち	0.000000	0.000135
いっさい	0.000000	0.000271
いっしょ	0.000000	0.000541
いっす	0.000000	0.000000
いつ	0.000649	0.002031
いつか	0.000130	0.000271
いぬ	0.000000	0.000135
いも	0.000000	0.000135
いや	0.000260	0.000135
いやいや	0.000260	0.000000
いやがる	0.000130	0.000000
いやしい	0.000000	0.000135
いらいら	0.000000	0.000000
いらう	0.000000	0.000000
いらっしゃい	0.000000	0.000000
いらっしゃる	0.000389	0.000271
いる	0.053458	0.049817
いるか	0.000000	0.000000

In [15]:
```
# 各文書について、語ごとの出現確率を計算
# (2) 頻度が0の場合は1として扱って確率は0にならないようにする方法

freqs = np.array(freqs_df, np.float)
freq_sums = np.array(freqs_df.sum(axis=1), np.float).reshape(2,1)  # For num
py's broadcast

probs = (freqs + 1.0) / (freq_sums + len(terms))

for i in range(probs.shape[1]):
    print("%s\t%f\t%f" % (terms[i],probs[0,i],probs[1,i]))
```

あい	0.000076	0.000234
あう	0.000076	0.000078
あか	0.000076	0.000156
あかるい	0.000076	0.000156
あがる	0.000152	0.000156
あきらめる	0.000152	0.000078
あきれる	0.000152	0.000078
あく	0.000076	0.000078
あける	0.000152	0.000078
あざやか	0.000076	0.000156
あすこ	0.000076	0.000078
あそこ	0.000076	0.000078
あたり	0.000533	0.000156
あちこち	0.000076	0.000078
あっけない	0.000076	0.000078
あっち	0.000228	0.000078
あて	0.000152	0.000078
あてがう	0.000076	0.000156
あてる	0.000076	0.000078
あと	0.000076	0.000702
あながち	0.000076	0.000078
あなた	0.000076	0.001794
あびせる	0.000152	0.000078
あやす	0.000076	0.000078
あやぶむ	0.000152	0.000078
あらあらしい	0.000152	0.000078
あらわれる	0.000076	0.000156
あり	0.000152	0.000078
ありがたい	0.000076	0.000234
ある	0.010652	0.007565
あるく	0.000076	0.000078
あれ	0.000380	0.000156
あんた	0.000076	0.000078
いい	0.000076	0.000468
いう	0.000076	0.001248
いえる	0.000076	0.000156
いかん	0.000076	0.000078
いく	0.000228	0.000078
いくつ	0.000076	0.000156
いくら	0.000076	0.000078
いける	0.000228	0.000702
いさむ	0.001293	0.000078
いじくる	0.000076	0.000156
いじる	0.000152	0.000078
いずれ	0.000076	0.000156
いたす	0.000152	0.000078
いただく	0.000076	0.000546
いただける	0.000076	0.000156
いたわる	0.000152	0.000078
いた事	0.000228	0.000078
いち	0.000076	0.000156
いっさい	0.000076	0.000234
いっしょ	0.000076	0.000390
いっす	0.000076	0.000078
いつ	0.000457	0.001248
いつか	0.000152	0.000234
いぬ	0.000076	0.000156
いも	0.000076	0.000156
いや	0.000228	0.000156
いやいや	0.000228	0.000078
いやがる	0.000152	0.000078
いやしい	0.000076	0.000156
いらいら	0.000076	0.000078
いらう	0.000076	0.000078
いらっしゃい	0.000076	0.000078
いらっしゃる	0.000304	0.000234
いる	0.031424	0.028776
いるか	0.000076	0.000078

In [16]:
```
# 各文書について，出現確率の合計は1になるはず
probs.sum(axis=1)
```

Out[16]:
```
array([1., 1.])
```

「テスト用」データを作る

In [17]:
```
df_td_test = df_td.iloc[[2,3,6,7]]
df_td_test
```

Out[17]:

	あい	あう	あか	あかるい	あがる	あきらめる	あきれる	あく	あける	あざやか	...	黒船	黒雲	黙り込む	黙る	黙然	黙許	鼈甲	鼓膜	鼠色	鼻糞
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	7	0	0	0	0	0	0
3	1	1	0	0	0	0	0	1	0	0	...	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	...	1	0	0	1	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	...	0	1	0	6	0	0	0	0	1	0

4 rows × 5436 columns

In [18]:
```
freqs = np.array(df_td_test, np.float)
freqs.shape
```

Out[18]:
```
(4, 5436)
```

尤度**(likelihood)**の対数をそれぞれのモデル**(**作家**)**の場合に計算する

In [19]:
```
ll = np.dot(freqs, np.log(probs.T))
ll
```

Out[19]:
```
array([[-27768.81926869, -28175.17554912],
       [-26742.9561593 , -27142.98133145],
       [-29156.93288585, -28371.13203413],
       [-25254.46151005, -24544.39640224]])
```

テスト用の各文書について，どちらのモデル**(**作家**)**があてはまるか

In [20]:
```
writers = ["森鴎外","夏目漱石"]
for k in np.argmax(ll,axis=1):
    print(writers[k])
```

```
森鴎外
森鴎外
夏目漱石
夏目漱石
```