データマイニング　レポート3

195755E 宮良一生

Level 1: 文書ファイル毎に、Bag-of-Wordsで特徴ベクトルを生成せよ。

まず nltk のインストールを行った.

```
$ pip install nltk
$ python
> import nltk
> nltk.download(['wordnet', 'stopwords', 'punkt'])
```

その後は講義資料を参考に codebook を作成し，コサイン類似度を計算した.

> (a) 13x13のマトリックス表記で文書間類似度（コサイン類似度）を記せ。表示有効桁は3桁とせよ。

以下がコサイン類似度である.

ch00	ch01	ch02	ch03	ch04	ch05	ch06	ch07	ch08	ch09	ch10	ch11	ch12
1.0	0.826	0.825	0.801	0.802	0.832	0.87	0.83	0.864	0.814	0.808	0.849	0.886
0.826	1.0	0.991	0.996	0.994	0.992	0.973	0.981	0.984	0.971	0.985	0.96	0.726
0.825	0.991	1.0	0.991	0.988	0.994	0.968	0.978	0.971	0.958	0.966	0.957	0.715
0.801	0.996	0.991	1.0	0.995	0.991	0.963	0.978	0.974	0.966	0.98	0.957	0.703
0.802	0.994	0.988	0.995	1.0	0.99	0.966	0.984	0.974	0.962	0.976	0.959	0.706
0.832	0.992	0.994	0.991	0.99	1.0	0.975	0.983	0.978	0.961	0.972	0.961	0.722
0.87	0.973	0.968	0.963	0.966	0.975	1.0	0.97	0.978	0.961	0.963	0.959	0.78
0.83	0.981	0.978	0.978	0.984	0.983	0.97	1.0	0.974	0.955	0.961	0.969	0.757
0.864	0.984	0.971	0.974	0.974	0.978	0.978	0.974	1.0	0.979	0.983	0.953	0.768
0.814	0.971	0.958	0.966	0.962	0.961	0.961	0.955	0.979	1.0	0.981	0.932	0.747
0.808	0.985	0.966	0.98	0.976	0.972	0.963	0.961	0.983	0.981	1.0	0.939	0.718
0.849	0.96	0.957	0.957	0.959	0.961	0.959	0.969	0.953	0.932	0.939	1.0	0.796
0.886	0.726	0.715	0.703	0.706	0.722	0.78	0.757	0.768	0.747	0.718	0.796	1.0

> (b) コード上工夫した箇所があるなら該当箇所だけを示し、解説せよ。ほぼコード例通りなら省略OK。

コード例通りなので省略する．この課題で使用したコードはGitLabにアップした.

> (c) 最も類似している2文書について、それらが類似していると判定された理由を検討し、述べよ。

コサイン類似度の表から，ch01 と ch03 の 0.996 が最も類似度が高い(同一HTMLファイルを除く)．しかしながら，これらのHTMLファイルは英文だけでなく，HTML や CSS, JavaScript も書かれており，また ch01 については冒頭から345行目までは全て英文ではない．また，345行目以降も HTML等は書かれている．つまり，ch00 と ch02 の類似度が 0.996 と最も高いがそれらは英文を比較して類似していると判定されたわけではなく，その他の文(HTML等) によって判定された可能性がある.

Level2

主に sklean を用いて tf-idfを計算した.

> (a) 13x13のマトリックス表記で文書間類似度（コサイン類似度）を記せ。表示有効桁は3桁とせよ。

以下がコサイン類似度である.

ch00	ch01	ch02	ch03	ch04	ch05	ch06	ch07	ch08	ch09	ch10	ch11	ch12
1.0	0.731	0.72	0.707	0.685	0.714	0.736	0.691	0.757	0.758	0.741	0.692	0.869
0.731	1.0	0.988	0.992	0.988	0.982	0.941	0.965	0.963	0.949	0.976	0.954	0.622
0.72	0.988	1.0	0.986	0.984	0.983	0.944	0.966	0.957	0.945	0.965	0.955	0.617
0.707	0.992	0.986	1.0	0.991	0.981	0.934	0.967	0.958	0.943	0.972	0.955	0.597
0.685	0.988	0.984	0.991	1.0	0.979	0.932	0.968	0.951	0.932	0.963	0.954	0.578
0.714	0.982	0.983	0.981	0.979	1.0	0.946	0.971	0.955	0.941	0.961	0.951	0.616
0.736	0.941	0.944	0.934	0.932	0.946	1.0	0.931	0.926	0.917	0.924	0.919	0.659
0.691	0.965	0.966	0.967	0.968	0.971	0.931	1.0	0.948	0.925	0.942	0.94	0.602
0.757	0.963	0.957	0.958	0.951	0.955	0.926	0.948	1.0	0.958	0.963	0.922	0.663
0.758	0.949	0.945	0.943	0.932	0.941	0.917	0.925	0.958	1.0	0.958	0.907	0.666
0.741	0.976	0.965	0.972	0.963	0.961	0.924	0.942	0.963	0.958	1.0	0.931	0.636
0.692	0.954	0.955	0.955	0.954	0.951	0.919	0.94	0.922	0.907	0.931	1.0	0.608
0.869	0.622	0.617	0.597	0.578	0.616	0.659	0.602	0.663	0.666	0.636	0.608	1.0

> (b) コード上工夫した箇所があるなら該当箇所だけを示し、解説せよ。

ほとんどコード例通りなので省略する.

> (c) 最も類似している2文書について、それらが類似していると判定された理由を検討し、述べよ。

コサイン類似度の表から，ch01 と ch03 の 0.992 が最も類似度が高い．今回はsklearn の　TfidfVectorizer.get*feature*names というメソッドを使用しワードリストを取得し，ワード毎の出現回数を2文書間で比較してみた．まず，ch01 について出現回数の大きい順に10個のワードを取得した.

```
'span': 2756.0, 'class': 2360.0, 'pysrc': 888.0, 'td': 699.0, 'tt': 664.0, 'gt': 594.0, 'doctest': 584.
0, 'pre': 459.0, 'tr': 392.0, 'table': 348.0,
```

次に ch03 について出現回数の大きい順に10個のワードを取得した.

```
'span': 4430.0, 'class': 3762.0, 'pysrc': 1554.0, 'tt': 1096.0, 'doctest': 996.0, 'gt': 990.0, 'td': 93
3.0, 'pre': 785.0, 'tr': 592.0, 'string': 566.0,
```

まず，ch01 について考察する．span や class などは英文にも度々出てきているがほとんどは HTMLのタグである．他にも pysrc は span タグの class につけられた名前であったりなど，ほとんどが英文に関係ないワードであった．また，ch03 についても同様のことがいえる.

以上より，2文書が類似していると判定された理由はHTML等の構文によるものだと推測する.

> (d) Level 1, 2の(a)を比較し、順位が入れ替わっている箇所があるか探せ。もしあれば、それらの文書群について順序が入れ替わった理由を検討せよ。

ch00 と ch08 , ch00 と ch09 を比較した時，Level1 のコサイン類似度は ch00 & ch08 > ch00 & ch09 であるが，Level2 のコサイン類似度は ch00 & ch09 > ch00 & ch08 となっている.

まず，Level1 の ch00, ch08, ch09 でどの単語がよく出現したか調べた.

```
ch00
'>': 521, '-': 342, '<': 339, '="': 306, '</': 276, ';': 249, 'class': 234, 'td': 227, ':': 208, 'span'
: 159,

ch08
'="': 2395, 'span': 1746, 'class': 1537, '>': 1527, '<': 1497, '-': 1103, '</': 1042, '">': 922, '"': 7
84, 'td': 631,

ch09
'="': 3342, 'span': 2176, 'class': 1908, '<': 1658, '>': 1584, '-': 1216, '"': 1164, 'td': 1031, '">':
940, '</': 936
```

次に Level2 の ch00, ch08, ch09 でどの単語がよく出現したか調べた.

```
ch00
'class': 229.0, 'td': 227.0, 'span': 159.0, 'tr': 124.0, 'tt': 82.0, 'field': 80.18833419727278, 'borde
r': 67.0, 'table': 63.0, 'doctest': 59.0, 'python': 59.0

ch08
'span': 1745.0, 'class': 1531.0, 'td': 631.0, 'pysrc': 448.0, 'tt': 448.0, 'doctest': 314.0, 'tr': 298.
0, 'pre': 288.0, 'table': 278.0, 'gt': 259.0,

ch09
'span': 2176.0, 'class': 1903.0, 'td': 1031.0, 'pysrc': 595.0, 'tt': 570.0, 'table': 426.0, 'tr': 418.0
, 'gt': 412.0, 'doctest': 390.0, 'pre': 385.0
```

Level1, Level2 の ch00 を比較すると，Level2 の 頻出単語から '>' や '=' などの英文に関係のないノイズが無くなっている．そのため Level 1 ではノイズも含めてコサイン類似度を算出しており，Level2 ではノイズを除去しコサイン類似度を算出しているため順位の違いが出たと考える.

> (e) (d)で入れ替わっている場合、この変化が好ましいか否か検討し、その理由とともに述べよ。

ノイズを取り除いた方がより正確な英文比較を行うことができるため，この変化は望ましい.

Level 3: 単語の共起行列から特徴ベクトルを生成せよ。

> (a) “natural”, “language”, “text”, “count”に加え、もう1個以上の単語を自由に選べ。合計5個以上の単語についてマトリックス表記で単語間類似度（コサイン類似度）を記せ。表示有効桁は3桁とせよ

今回は python という単語を選択した． そして作成したコサイン類似度は下記である.

natural	language	text	count	python
1.0	0.365	0.321	0.175	0.279
0.365	1.0	0.688	0.456	0.585
0.321	0.688	1.0	0.617	0.775
0.175	0.456	0.617	1.0	0.608
0.279	0.585	0.775	0.608	1.0

> (b) コード上工夫した箇所があるなら該当箇所だけを示し、解説せよ。ほぼコード例通りなら省略OK。

ほとんど例題のままのため省略する.

> (c) 最も類似している2単語について、それらが類似していると判定された理由を検討し、述べよ。（単にコサイン類似度が大きいからではなく、何故大きいのかを検討してみよう）

python と text という 2つの単語が最も類似していると判定された． その理由として，今回使用した文書では text を python を使用して処理しているため，そのような結果が得られたと考える.

参考文献

今回使用したソース: https://github.com/Issei0804-ie/datamining-report3

sklearnのリファレンス: https://scikit-learn.org/stable/modules/generated/sklearn.feature*extraction.text.TfidfVectorizer.html#sklearn.feature*extraction.text.TfidfVectorizer.get*feature*names

コサイン類似度について : https://tex2e.github.io/blog/python/docs-similarity

python の print文 で改行しない方法 : https://www.headboost.jp/python-print-without-adding-new-line/