실습 1 - 간단한 토픽모델링 구현 LSA

```
[ ]  1 docs = ['바나나 사과 포도 포도',
     2          '사과 포도',
     3          '포도 바나나',
     4          '짜장면 짬뽕 탕수육',
     5          '볶음밥 탕수육',
     6          '짜장면 짬뽕',
     7          '된장찌개 김치찌개 김치 비빔밥',
     8          '김치 된장 비빔밥',
     9          '비빔밥 김치',
    10          '사과 볶음밥 김치 된장']
```

```
[ ]  1 from sklearn.feature_extraction.text import TfidfVectorizer
     2 from sklearn.decomposition import TruncatedSVD
     3
     4 n_topic= 3
     5 #선언
     6 tfidf_vect = TfidfVectorizer()
     7 tfidf = tfidf_vect.fit_transform(docs)
     8 svd = TruncatedSVD(n_components=n_topic)
     9 u_sigma = svd.fit_transform(tfidf)
    10 svd.components_
    11
```

```
array([[ 0.39094311,  0.08663945,  0.2634605 ,  0.08663945,  0.31533599,
         0.1677167 ,  0.33012626,  0.42118555,  0.01907605,  0.01907605,
         0.05422822,  0.58747296],
       [ 0.49683512,  0.13720069,  0.26962551,  0.13720069, -0.29774469,
         0.10773075,  0.49275818, -0.18567019,  0.02131272,  0.02131272,
         0.04450862, -0.51222803],
       [-0.06285924, -0.02901254, -0.00762375, -0.02901254, -0.01957621,
         0.19210545, -0.09004258,  0.00851823,  0.60658392,  0.60658392,
         0.46061907, -0.02794911]])
```

```
[ ]  1 vocab = tfidf_vect.get_feature_names_out()
     2 n = 3
     3 for idx, topic in enumerate(svd.components_):
     4     print("Topic %d:" % (idx), [(vocab[i], topic[i].round(5)) for i in topic.argsort()[:-n - 1:-1]])
     5
```

```
Topic 0: [('포도', 0.58747), ('사과', 0.42119), ('김치', 0.39094)]
Topic 1: [('김치', 0.49684), ('비빔밥', 0.49276), ('된장', 0.26963)]
Topic 2: [('짬뽕', 0.60658), ('짜장면', 0.60658), ('탕수육', 0.46062)]
```

단어벡터간의 상관관계

```
[ ]  1 #단어벡터
     2 for i in range(len(vocab)) :
     3     print("{} : {}".format(vocab[i], svd.components_.T[i]))
```

```
김치 : [ 0.39094311  0.49683512 -0.06285924]
김치찌개 : [ 0.08663945  0.13720069 -0.02901254]
된장 : [ 0.2634605   0.26962551 -0.00762375]
된장찌개 : [ 0.08663945  0.13720069 -0.02901254]
바나나 : [ 0.31533599 -0.29774469 -0.01957621]
볶음밥 : [0.1677167  0.10773075 0.19210545]
비빔밥 : [ 0.33012626  0.49275818 -0.09004258]
사과 : [ 0.42118555 -0.18567019  0.00851823]
짜장면 : [0.01907605 0.02131272 0.60658392]
짬뽕 : [0.01907605 0.02131272 0.60658392]
탕수육 : [0.05422822 0.04450862 0.46061907]
포도 : [ 0.58747296 -0.51222803 -0.02794911]
```

```
[ ]  1 import numpy as np
     2 from numpy import dot
     3 from numpy.linalg import norm
     4 # 코사인유사도
     5 def cosine_similarity(a, b) :
     6     return dot(a, b)/(norm(a)*norm(b))
     7 # 코사인유사도를 사용해서 행렬의 유사도 구하기.
     8 def calc_similarity_matrix(vectors) :
     9     n_word = len(vectors)
    10     similarity_matrix = np.zeros((n_word, n_word))
    11
    12     for i in range(n_word) :
    13         # 위에서 정의한 코사인 유사도 사용
    14         for j in range(i, n_word) :
    15             similarity_matrix[j, i] = cosine_similarity(vectors[i], vectors[j]).round(4)
    16
    17     return similarity_matrix
    18
```

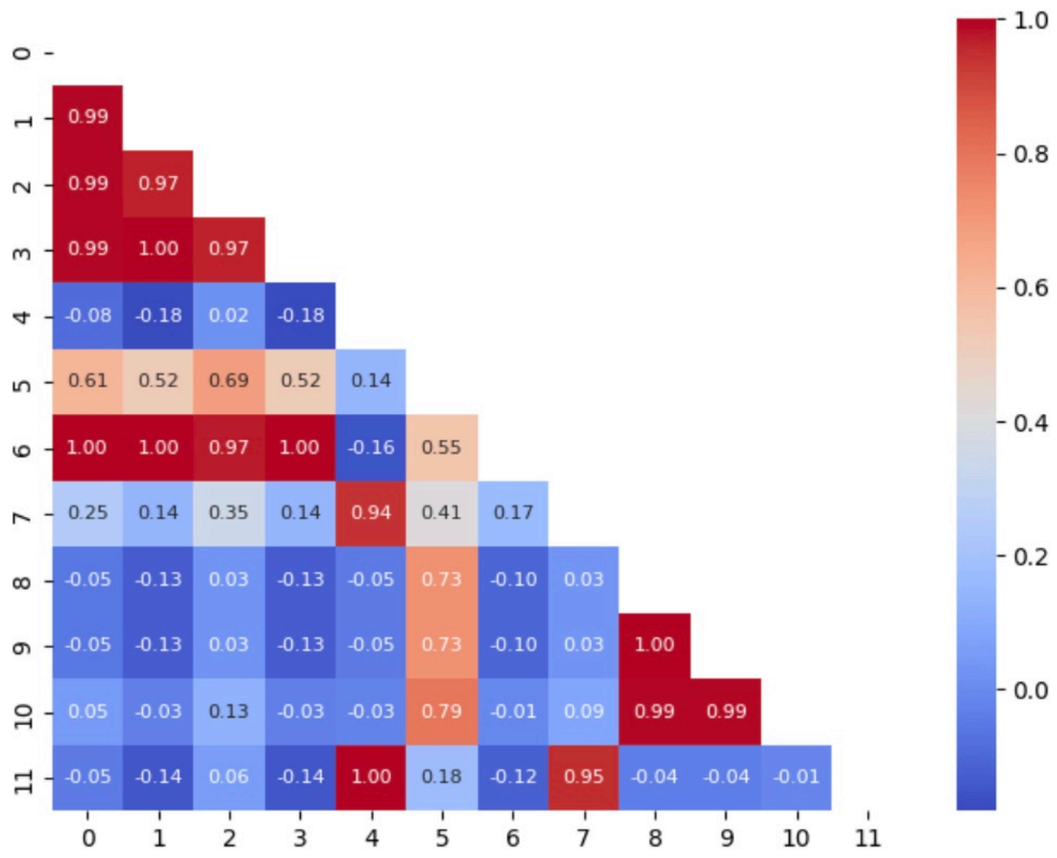
```
     1 import matplotlib.pyplot as plt
     2 import seaborn as sns
     3
     4 def visualize_similarity(similarity_matrix) :
     5     uniform_data = similarity_matrix
     6     mask = np.triu(np.ones_like(similarity_matrix, dtype=np.bool))
     7     plt.rcParams['figure.figsize'] = [8, 6]
     8     ax = sns.heatmap(uniform_data, mask=mask,
     9                      annot=True, fmt=".2f",annot_kws={'size':8},
    10                      cmap='coolwarm')
    11
```

```
[ ]  1 print(vocab)
```

```
['김치' '김치찌개' '된장' '된장찌개' '바나나' '볶음밥' '비빔밥' '사과' '짜장면' '짬뽕' '탕수육' '포도']
```

```
[ ]  1 word_vectors = svd.components_.T
     2 word_similarity_matrix = calc_similarity_matrix(word_vectors)
     3 visualize_similarity(word_similarity_matrix)
```

```
<ipython-input-8-93e11ef974bf>:6: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing th
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.triu(np.ones_like(similarity_matrix, dtype=np.bool))
```



단어벡터 시각화

```
[ ]  1 # 한글 폰트 설정
     2 # 설치하고 한글 적용이 안된다면 , 런타임 > 런타임 다시시작 ; 하고 설치코드 제외하고 나머지 코드 실행
     3 !sudo apt-get install -y fonts-nanum
     4 !sudo fc-cache -fv
     5 !rm ~/.cache/matplotlib -rf
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
```

```
[ ]  1 %matplotlib inline
     2 import matplotlib.font_manager as fm
     3 import matplotlib
     4 font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
     5 fontprop = fm.FontProperties(fname=font_path, size=12)
```

```
     1 from sklearn.manifold import TSNE
     2 import numpy as np
     3 vectors = word_vectors
     4 labels = tfidf_vect.get_feature_names_out()
     5
     6 def visualize_vectors(vectors, labels):
     7     tsne = TSNE(n_components=2, random_state=0, n_iter=10000, \
     8                 perplexity=2)
     9     np.set_printoptions(suppress=True)
    10     T = tsne.fit_transform(vectors)
    11
    12     plt.figure(figsize=(10, 6))
    13     plt.scatter(T[:, 0], T[:, 1], c='orange', edgecolors='r')
    14     for label, x, y in zip(labels, T[:, 0], T[:, 1]):
    15         plt.annotate(label, xy=(x+1, y+1), xytext=(0, 0), \
    16                      textcoords='offset points',\
    17                      fontproperties=fontprop)
    18
    19 visualize_vectors(vectors, labels)
```