1번

In [1]:
```
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
import numpy as np
dataset = fetch_20newsgroups(shuffle=True, random_state=1, remove=('headers', 'footers', 'quotes
documents = dataset.data

print('샘플의 수 :',len(documents))
```

```
샘플의 수 : 11314
```

In [2]:
```
documents[1]
```

Out[2]:
```
"WnWnWnWnWnWnWnYeah, do you expect people to read the FAQ, etc. and actually accept hardWnatheis
m?  No, you need a little leap of faith, Jimmy.  Your logic runs outWnof steam!WnWnWnWnWnWnWnWnJi
m,WnWnSorry I can't pity you, Jim.  And I'm sorry that you have these feelings ofWndenial about t
he faith you need to get by.  Oh well, just pretend that it willWnall end happily ever after anyw
ay.  Maybe if you start a new newsgroup,Wnalt.atheist.hard, you won't be bummin' so much?WnWnWnWn
WnWnWnBye-Bye, Big Jim.  Don't forget your Flintstone's Chewables!  :) Wn--WnBake Timmons, III"
```

In [3]:
```
print(dataset.target_names)
```

```
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sy
s.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.bas
eball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'soc.religio
n.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politics.misc', 'talk.religio
n.misc']
```

In [4]:
```
news_df = pd.DataFrame({'document':documents})
# 특수 문자 제거
news_df['clean_doc'] = news_df['document'].str.replace("[^a-zA-Z]", " ")
# 길이가 3이하인 단어는 제거 (길이가 짧은 단어 제거)
news_df['clean_doc'] = news_df['clean_doc'].apply(lambda x: ' '.join([w for w in x.split() if
# 전체 단어에 대한 소문자 변환
news_df['clean_doc'] = news_df['clean_doc'].apply(lambda x: x.lower())
```

```
C:WUsersW82108WAppDataWLocalWTemp/ipykernel_8724/1974753330.py:3: FutureWarning: The default valu
e of regex will change from True to False in a future version.
  news_df['clean_doc'] = news_df['document'].str.replace("[^a-zA-Z]", " ")
```

In [5]:
```
news_df['clean_doc'][1]
```

Out[5]:
```
'yeah expect people read actually accept hard atheism need little leap faith jimmy your logic run
s steam sorry pity sorry that have these feelings denial about faith need well just pretend that
will happily ever after anyway maybe start newsgroup atheist hard bummin much forget your flintst
one chewables bake timmons'
```

In [6]:
```
# NLTK로부터 불용어를 받아온다.
stop_words = stopwords.words('english')
tokenized_doc = news_df['clean_doc'].apply(lambda x: x.split()) # 토큰화
tokenized_doc = tokenized_doc.apply(lambda x: [item for item in x if item not in stop_words])
# 불용어를 제거합니다.
```

In [7]:
```
print(tokenized_doc[1])
```

```
['yeah', 'expect', 'people', 'read', 'actually', 'accept', 'hard', 'atheism', 'need', 'little',
'leap', 'faith', 'jimmy', 'logic', 'runs', 'steam', 'sorry', 'pity', 'sorry', 'feelings', 'denia
l', 'faith', 'need', 'well', 'pretend', 'happily', 'ever', 'anyway', 'maybe', 'start', 'newsgrou
p', 'atheist', 'hard', 'bummin', 'much', 'forget', 'flintstone', 'chewables', 'bake', 'timmons']
```

In [8]:
```
# 역토큰화 (토큰화 작업을 역으로 되돌림)
detokenized_doc = []
for i in range(len(news_df)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)

news_df['clean_doc'] = detokenized_doc
```

In [9]:
```
news_df['clean_doc'][1]
```

Out[9]:
```
'yeah expect people read actually accept hard atheism need little leap faith jimmy logic runs ste
am sorry pity sorry feelings denial faith need well pretend happily ever anyway maybe start newsg
roup atheist hard bummin much forget flintstone chewables bake timmons'
```

In [10]:
```
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000, # 상위 1,000개의 단어를 보
max_df = 0.5, smooth_idf=True)

X = vectorizer.fit_transform(news_df['clean_doc'])

# TF-IDF 행렬의 크기 확인
print('TF-IDF 행렬의 크기 :',X.shape)
```

```
TF-IDF 행렬의 크기 : (11314, 1000)
```

In [11]:
```
svd_model = TruncatedSVD(n_components=20, algorithm='randomized', n_iter=100, random_state=122)
svd_model.fit(X)
len(svd_model.components_)
```

Out[11]:
```
20
```

In [12]:
```
np.shape(svd_model.components_)
```

Out[12]:
```
(20, 1000)
```

In [13]:
```
terms = vectorizer.get_feature_names() # 단어 집합. 1,000개의 단어가 저장됨.

def get_topics(components, feature_names, n=5):
    for idx, topic in enumerate(components):
        print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(5)) for i in topic.argso
get_topics(svd_model.components_,terms)
```

```
Topic 1: [('like', 0.21386), ('know', 0.20046), ('people', 0.19293), ('think', 0.17805), ('good',
0.15128)]
Topic 2: [('thanks', 0.32888), ('windows', 0.29088), ('card', 0.18069), ('drive', 0.17455), ('mai
l', 0.15111)]
Topic 3: [('game', 0.37064), ('team', 0.32443), ('year', 0.28154), ('games', 0.2537), ('season',
0.18419)]
Topic 4: [('drive', 0.53324), ('scsi', 0.20165), ('hard', 0.15628), ('disk', 0.15578), ('card',
0.13994)]
Topic 5: [('windows', 0.40399), ('file', 0.25436), ('window', 0.18044), ('files', 0.16078), ('pro
gram', 0.13894)]
Topic 6: [('chip', 0.16114), ('government', 0.16009), ('mail', 0.15625), ('space', 0.1507), ('inf
ormation', 0.13562)]
Topic 7: [('like', 0.67086), ('bike', 0.14236), ('chip', 0.11169), ('know', 0.11139), ('sounds',
0.10371)]
Topic 8: [('card', 0.46633), ('video', 0.22137), ('sale', 0.21266), ('monitor', 0.15463), ('offe
r', 0.14643)]
```

```
Topic 9: [('know', 0.46047), ('card', 0.33605), ('chip', 0.17558), ('government', 0.1522), ('vide
o', 0.14356)]
Topic 10: [('good', 0.42756), ('know', 0.23039), ('time', 0.1882), ('bike', 0.11406), ('jesus',
0.09027)]
Topic 11: [('think', 0.78469), ('chip', 0.10899), ('good', 0.10635), ('thanks', 0.09123), ('clipp
er', 0.07946)]
Topic 12: [('thanks', 0.36824), ('good', 0.22729), ('right', 0.21559), ('bike', 0.21037), ('probl
em', 0.20894)]
Topic 13: [('good', 0.36212), ('people', 0.33985), ('windows', 0.28385), ('know', 0.26232), ('fil
e', 0.18422)]
Topic 14: [('space', 0.39946), ('think', 0.23258), ('know', 0.18074), ('nasa', 0.15174), ('proble
m', 0.12957)]
Topic 15: [('space', 0.31613), ('good', 0.3094), ('card', 0.22603), ('people', 0.17476), ('time',
0.14496)]
Topic 16: [('people', 0.48156), ('problem', 0.19961), ('window', 0.15281), ('time', 0.14664), ('g
ame', 0.12871)]
Topic 17: [('time', 0.34465), ('bike', 0.27303), ('right', 0.25557), ('windows', 0.1997), ('fil
e', 0.19118)]
Topic 18: [('time', 0.5973), ('problem', 0.15504), ('file', 0.14956), ('think', 0.12847), ('israe
l', 0.10903)]
Topic 19: [('file', 0.44163), ('need', 0.26633), ('card', 0.18388), ('files', 0.17453), ('right',
0.15448)]
Topic 20: [('problem', 0.33006), ('file', 0.27651), ('thanks', 0.23578), ('used', 0.19206), ('spa
ce', 0.13185)]
```

2번

In [14]:
```
svd_model = TruncatedSVD(n_components=10, algorithm='randomized', n_iter=100, random_state=122)
svd_model.fit(X)
len(svd_model.components_)
```

Out[14]:
```
10
```

In [15]:
```
np.shape(svd_model.components_)
```

Out[15]:
```
(10, 1000)
```

In [16]:
```
terms = vectorizer.get_feature_names() # 단어 집합. 1,000개의 단어가 저장됨.

def get_topics(components, feature_names, n=5):
    for idx, topic in enumerate(components):
        print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(5)) for i in topic.args
get_topics(svd_model.components_,terms)
```

```
Topic 1: [('like', 0.21386), ('know', 0.20046), ('people', 0.19293), ('think', 0.17805), ('good',
0.15128)]
Topic 2: [('thanks', 0.32888), ('windows', 0.29088), ('card', 0.18069), ('drive', 0.17455), ('mai
l', 0.15111)]
Topic 3: [('game', 0.37064), ('team', 0.32443), ('year', 0.28154), ('games', 0.2537), ('season',
0.18419)]
Topic 4: [('drive', 0.53324), ('scsi', 0.20165), ('hard', 0.15628), ('disk', 0.15578), ('card',
0.13994)]
Topic 5: [('windows', 0.40399), ('file', 0.25436), ('window', 0.18044), ('files', 0.16078), ('pro
gram', 0.13894)]
Topic 6: [('chip', 0.16114), ('government', 0.16009), ('mail', 0.15625), ('space', 0.1507), ('inf
ormation', 0.13562)]
Topic 7: [('like', 0.67086), ('bike', 0.14236), ('chip', 0.11169), ('know', 0.11139), ('sounds',
0.10371)]
Topic 8: [('card', 0.46633), ('video', 0.22137), ('sale', 0.21266), ('monitor', 0.15463), ('offe
r', 0.14643)]
Topic 9: [('know', 0.46047), ('card', 0.33605), ('chip', 0.17558), ('government', 0.1522), ('vide
o', 0.14356)]
```

```
Topic 10: [('good', 0.42756), ('know', 0.23039), ('time', 0.1882), ('bike', 0.11406), ('jesus', 0.09027)]
```

- topic이 20개였을 때 상위 10개만 추려서 동일하게 나온 것을 2번에서 확인할 수 있다.

3번

In [17]:
```
tokenized_doc[:5]
```

Out[17]:
```
0    [well, sure, story, seem, biased, disagree, st...
1    [yeah, expect, people, read, actually, accept,...
2    [although, realize, principle, strongest, poin...
3    [notwithstanding, legitimate, fuss, proposal, ...
4    [well, change, scoring, playoff, pool, unfortu...
Name: clean_doc, dtype: object
```

In [18]:
```
from gensim import corpora
dictionary = corpora.Dictionary(tokenized_doc)
corpus = [dictionary.doc2bow(text) for text in tokenized_doc]
print(corpus[1]) # 수행된 결과에서 두번째 뉴스 출력. 첫번째 문서의 인덱스는 0
```

```
[(52, 1), (55, 1), (56, 1), (57, 1), (58, 1), (59, 1), (60, 1), (61, 1), (62, 1), (63, 1), (64, 1), (65, 1), (66, 2), (67, 1), (68, 1), (69, 1), (70, 1), (71, 2), (72, 1), (73, 1), (74, 1), (75, 1), (76, 1), (77, 1), (78, 2), (79, 1), (80, 1), (81, 1), (82, 1), (83, 1), (84, 1), (85, 2), (86, 1), (87, 1), (88, 1), (89, 1)]
```

In [19]:
```
print(dictionary[66])
```

```
faith
```

In [20]:
```
len(dictionary)
```

Out[20]:
```
64281
```

In [21]:
```
import gensim
NUM_TOPICS = 20 # 20개의 토픽, k=20
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dictionary,
topics = ldamodel.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

```
(0, '0.023*"food" + 0.013*"cancer" + 0.009*"diet" + 0.008*"mary"')
(1, '0.013*"government" + 0.013*"encryption" + 0.012*"chip" + 0.011*"security"')
(2, '0.011*"caps" + 0.011*"baltimore" + 0.010*"lost" + 0.009*"john"')
(3, '0.017*"space" + 0.009*"information" + 0.008*"available" + 0.007*"data"')
(4, '0.071*"drive" + 0.031*"disk" + 0.027*"hard" + 0.022*"drives"')
(5, '0.027*"water" + 0.021*"tobacco" + 0.014*"dept" + 0.013*"smokeless"')
(6, '0.020*"medical" + 0.016*"disease" + 0.015*"pain" + 0.015*"patients"')
(7, '0.010*"people" + 0.007*"would" + 0.006*"president" + 0.005*"states"')
(8, '0.060*"scsi" + 0.023*"simms" + 0.012*"slots" + 0.010*"terminals"')
(9, '0.025*"wire" + 0.023*"ground" + 0.015*"neutral" + 0.015*"gordon"')
(10, '0.012*"would" + 0.011*"like" + 0.009*"think" + 0.009*"time"')
(11, '0.017*"bike" + 0.013*"engine" + 0.009*"cars" + 0.009*"miles"')
(12, '0.012*"windows" + 0.008*"software" + 0.008*"using" + 0.007*"also"')
(13, '0.010*"hash" + 0.009*"lyme" + 0.008*"qemm" + 0.007*"offenses"')
(14, '0.044*"file" + 0.027*"output" + 0.025*"entry" + 0.020*"program"')
(15, '0.021*"armenian" + 0.018*"armenians" + 0.015*"turkish" + 0.012*"people"')
(16, '0.011*"people" + 0.010*"would" + 0.009*"jesus" + 0.007*"think"')
(17, '0.025*"game" + 0.024*"team" + 0.020*"games" + 0.019*"play"')
(18, '0.010*"would" + 0.007*"israel" + 0.007*"people" + 0.005*"jews"')
(19, '0.018*"would" + 0.014*"please" + 0.013*"know" + 0.012*"thanks"')
```

In [22]:
```
print(ldamodel.print_topics())
```

```
[(0, '0.023*"food" + 0.013*"cancer" + 0.009*"diet" + 0.008*"mary" + 0.008*"sexual" + 0.008*"homos
exuality" + 0.008*"homosexual" + 0.008*"symbol" + 0.008*"homosexuals" + 0.007*"eating"'), (1, '0.
013*"government" + 0.013*"encryption" + 0.012*"chip" + 0.011*"security" + 0.010*"public" + 0.010
*"system" + 0.010*"keys" + 0.010*"clipper" + 0.008*"privacy" + 0.007*"technology"'), (2, '0.011
*"caps" + 0.011*"baltimore" + 0.010*"lost" + 0.009*"john" + 0.009*"maine" + 0.006*"astros" + 0.00
6*"standings" + 0.006*"oilers" + 0.006*"braves" + 0.006*"rochester"'), (3, '0.017*"space" + 0.009
*"information" + 0.008*"available" + 0.007*"data" + 0.006*"mail" + 0.006*"nasa" + 0.006*"send" +
0.006*"list" + 0.006*"university" + 0.005*"research"'), (4, '0.071*"drive" + 0.031*"disk" + 0.027
*"hard" + 0.022*"drives" + 0.020*"controller" + 0.015*"floppy" + 0.015*"scsi" + 0.013*"tape" + 0.
009*"bios" + 0.009*"motherboard"'), (5, '0.027*"water" + 0.021*"tobacco" + 0.014*"dept" + 0.013
*"smokeless" + 0.009*"doug" + 0.007*"tube" + 0.007*"part" + 0.006*"label" + 0.006*"plants" + 0.00
6*"cooling"'), (6, '0.020*"medical" + 0.016*"disease" + 0.015*"pain" + 0.015*"patients" + 0.014
*"april" + 0.013*"doctor" + 0.012*"montreal" + 0.009*"treatment" + 0.008*"medicine" + 0.008*"queb
ec"'), (7, '0.010*"people" + 0.007*"would" + 0.006*"president" + 0.005*"states" + 0.005*"governme
nt" + 0.005*"state" + 0.004*"health" + 0.004*"said" + 0.004*"well" + 0.004*"control"'), (8, '0.06
0*"scsi" + 0.023*"simms" + 0.012*"slots" + 0.010*"terminals" + 0.010*"chip" + 0.009*"simm" + 0.00
8*"obfuscate" + 0.008*"conductor" + 0.008*"vram" + 0.008*"pointers"'), (9, '0.025*"wire" + 0.023
*"ground" + 0.015*"neutral" + 0.015*"gordon" + 0.014*"pitt" + 0.013*"wiring" + 0.013*"banks" + 0.
012*"radar" + 0.011*"soon" + 0.011*"surrender"'), (10, '0.012*"would" + 0.011*"like" + 0.009*"thi
nk" + 0.009*"time" + 0.008*"good" + 0.007*"know" + 0.007*"well" + 0.006*"back" + 0.006*"year" +
0.006*"much"'), (11, '0.017*"bike" + 0.013*"engine" + 0.009*"cars" + 0.009*"miles" + 0.008*"cove
r" + 0.007*"condition" + 0.006*"excellent" + 0.006*"model" + 0.006*"motorcycle" + 0.006*"spee
d"'), (12, '0.012*"windows" + 0.008*"software" + 0.008*"using" + 0.007*"also" + 0.007*"window" +
0.007*"version" + 0.007*"system" + 0.006*"available" + 0.006*"files" + 0.006*"problem"'), (13,
'0.010*"hash" + 0.009*"lyme" + 0.008*"qemm" + 0.007*"offenses" + 0.006*"inkjet" + 0.006*"acids" +
0.005*"chase" + 0.004*"trek" + 0.004*"circumcision" + 0.004*"genetics"'), (14, '0.044*"file" + 0.
027*"output" + 0.025*"entry" + 0.020*"program" + 0.012*"line" + 0.011*"section" + 0.011*"widget"
+ 0.011*"build" + 0.009*"info" + 0.009*"name"'), (15, '0.021*"armenian" + 0.018*"armenians" + 0.0
15*"turkish" + 0.012*"people" + 0.012*"said" + 0.009*"turkey" + 0.008*"greek" + 0.008*"killed" +
0.007*"armenia" + 0.007*"turks"'), (16, '0.011*"people" + 0.010*"would" + 0.009*"jesus" + 0.007
*"think" + 0.007*"believe" + 0.005*"christian" + 0.005*"know" + 0.005*"many" + 0.005*"bible" + 0.
005*"even"'), (17, '0.025*"game" + 0.024*"team" + 0.020*"games" + 0.019*"play" + 0.015*"season" +
0.015*"hockey" + 0.014*"period" + 0.012*"league" + 0.009*"power" + 0.008*"pittsburgh"'), (18, '0.
010*"would" + 0.007*"israel" + 0.007*"people" + 0.005*"jews" + 0.005*"even" + 0.005*"israeli" +
0.004*"many" + 0.004*"human" + 0.004*"jewish" + 0.003*"fact"'), (19, '0.018*"would" + 0.014*"plea
se" + 0.013*"know" + 0.012*"thanks" + 0.012*"anyone" + 0.011*"like" + 0.008*"mail" + 0.007*"used"
+ 0.007*"need" + 0.007*"good"')]
```

In [23]:
```
for i, topic_list in enumerate(ldamodel[corpus]):
    if i==5:
        break
    print(i,'번째 문서의 topic 비율은',topic_list)
```

```
0 번째 문서의 topic 비율은 [(5, 0.11463869), (15, 0.2969293), (16, 0.13634701), (18, 0.4391697)]
1 번째 문서의 topic 비율은 [(5, 0.027602544), (8, 0.02751689), (10, 0.2914214), (16, 0.63238925)]
2 번째 문서의 topic 비율은 [(10, 0.26908287), (18, 0.537357), (19, 0.17958945)]
3 번째 문서의 topic 비율은 [(1, 0.3945108), (4, 0.017411016), (5, 0.015174772), (7, 0.23985073),
(10, 0.28099492), (16, 0.041083086)]
4 번째 문서의 topic 비율은 [(6, 0.039634727), (10, 0.3716009), (14, 0.091631874), (17, 0.467493
7)]
```

In [24]:
```
def make_topictable_per_doc(ldamodel, corpus):
    topic_table = pd.DataFrame()

    # 몇 번째 문서인지를 의미하는 문서 번호와 해당 문서의 토픽 비중을 한 줄씩 꺼내온다.
    for i, topic_list in enumerate(ldamodel[corpus]):
        doc = topic_list[0] if ldamodel.per_word_topics else topic_list
        doc = sorted(doc, key=lambda x: (x[1]), reverse=True)
        # 각 문서에 대해서 비중이 높은 토픽순으로 토픽을 정렬한다.
        # EX) 정렬 전 0번 문서 : (2번 토픽, 48.5%), (8번 토픽, 25%), (10번 토픽, 5%), (12번 토픽,
        # Ex) 정렬 후 0번 문서 : (2번 토픽, 48.5%), (8번 토픽, 25%), (12번 토픽, 21.5%), (10번 토
        # 48 > 25 > 21 > 5 순으로 정렬이 된 것.

        # 모든 문서에 대해서 각각 아래를 수행
```

```
        for j, (topic_num, prop_topic) in enumerate(doc): #  몇 번 토픽인지와 비중을 나눠서 저장
            if j == 0:  # 정렬을 한 상태이므로 가장 앞에 있는 것이 가장 비중이 높은 토픽
                topic_table = topic_table.append(pd.Series([int(topic_num), round(prop_topic,4),
                # 가장 비중이 높은 토픽과, 가장 비중이 높은 토픽의 비중과, 전체 토픽의 비중을 저
            else:
                break
    return(topic_table)
```

In [25]:

```
topictable = make_topictable_per_doc(ldamodel, corpus)
topictable = topictable.reset_index() # 문서 번호을 의미하는 열(column)로 사용하기 위해서 인덱스
topictable.columns = ['문서 번호', '가장 비중이 높은 토픽', '가장 높은 토픽의 비중', '각 토픽의
topictable[:10]
```

Out[25]:

	문서 번호	가장 비중이 높은 토픽	가장 높은 토픽의 비중	각 토픽의 비중
0	0	18.0	0.4392	[(5, 0.11463941), (15, 0.29690138), (16, 0.136...
1	1	16.0	0.6324	[(5, 0.027602546), (8, 0.027516892), (10, 0.29...
2	2	18.0	0.5374	[(10, 0.2690452), (18, 0.5373771), (19, 0.1796...
3	3	1.0	0.3945	[(1, 0.3945128), (4, 0.017411036), (5, 0.01517...
4	4	17.0	0.4675	[(6, 0.039634775), (10, 0.3716026), (14, 0.091...
5	5	16.0	0.4731	[(5, 0.22837992), (10, 0.18837404), (16, 0.473...
6	6	12.0	0.3863	[(3, 0.052971452), (4, 0.018874437), (10, 0.12...
7	7	10.0	0.4645	[(4, 0.018652836), (9, 0.054341014), (10, 0.46...
8	8	9.0	0.3818	[(0, 0.025999222), (7, 0.10962836), (9, 0.3818...
9	9	19.0	0.3911	[(1, 0.06340133), (4, 0.04152465), (10, 0.3171...

4번

In [26]:

```
import gensim
NUM_TOPICS = 10 # 10개의 토픽, k=10
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dictionary,
topics = ldamodel.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

```
(0, '0.007*"bike" + 0.005*"ground" + 0.005*"power" + 0.005*"engine"')
(1, '0.011*"windows" + 0.011*"file" + 0.010*"window" + 0.008*"files"')
(2, '0.018*"health" + 0.014*"medical" + 0.012*"disease" + 0.010*"patients"')
(3, '0.016*"game" + 0.015*"team" + 0.012*"year" + 0.011*"games"')
(4, '0.008*"people" + 0.006*"said" + 0.006*"government" + 0.005*"armenian"')
(5, '0.015*"encryption" + 0.012*"chip" + 0.011*"clipper" + 0.011*"government"')
(6, '0.012*"drive" + 0.009*"card" + 0.009*"thanks" + 0.008*"would"')
(7, '0.010*"space" + 0.009*"information" + 0.007*"mail" + 0.006*"available"')
(8, '0.014*"would" + 0.008*"people" + 0.008*"think" + 0.008*"like"')
(9, '0.027*"file" + 0.025*"entry" + 0.025*"output" + 0.014*"program"')
```

In [27]:

```
print(ldamodel.print_topics())
```

```
[(0, '0.007*"bike" + 0.005*"ground" + 0.005*"power" + 0.005*"engine" + 0.004*"high" + 0.004*"wate
r" + 0.004*"used" + 0.004*"wire" + 0.004*"miles" + 0.003*"launch"'), (1, '0.011*"windows" + 0.011
*"file" + 0.010*"window" + 0.008*"files" + 0.007*"version" + 0.007*"program" + 0.007*"display" +
0.007*"image" + 0.007*"using" + 0.007*"server"'), (2, '0.018*"health" + 0.014*"medical" + 0.012
*"disease" + 0.010*"patients" + 0.009*"pain" + 0.008*"doctor" + 0.007*"gordon" + 0.007*"cancer" +
0.007*"pitt" + 0.006*"diseases"'), (3, '0.016*"game" + 0.015*"team" + 0.012*"year" + 0.011*"game
s" + 0.010*"play" + 0.009*"season" + 0.008*"players" + 0.008*"league" + 0.007*"hockey" + 0.007*"p
eriod"'), (4, '0.008*"people" + 0.006*"said" + 0.006*"government" + 0.005*"armenian" + 0.004*"isr
ael" + 0.004*"armenians" + 0.004*"state" + 0.004*"jews" + 0.004*"president" + 0.003*"states"'),
```

```
(5, '0.015*"encryption" + 0.012*"chip" + 0.011*"clipper" + 0.011*"government" + 0.011*"keys" + 0.009*"security" + 0.008*"privacy" + 0.006*"algorithm" + 0.006*"secure" + 0.005*"escrow"'), (6, '0.012*"drive" + 0.009*"card" + 0.009*"thanks" + 0.008*"would" + 0.008*"know" + 0.008*"like" + 0.007*"disk" + 0.007*"system" + 0.007*"scsi" + 0.007*"anyone"'), (7, '0.010*"space" + 0.009*"information" + 0.007*"mail" + 0.006*"available" + 0.006*"also" + 0.006*"data" + 0.005*"send" + 0.005*"program" + 0.005*"list" + 0.005*"system"'), (8, '0.014*"would" + 0.008*"people" + 0.008*"think" + 0.008*"like" + 0.007*"know" + 0.006*"time" + 0.006*"even" + 0.005*"well" + 0.005*"could" + 0.005*"good"'), (9, '0.027*"file" + 0.025*"entry" + 0.025*"output" + 0.014*"program" + 0.009*"build" + 0.009*"entries" + 0.009*"printf" + 0.009*"char" + 0.009*"rules" + 0.008*"oname"')]
```

In [28]:

```
for i, topic_list in enumerate(ldamodel[corpus]):
    if i==5:
        break
    print(i,'번째 문서의 topic 비율은',topic_list)
```

```
0 번째 문서의 topic 비율은 [(4, 0.40621686), (8, 0.5808447)]
1 번째 문서의 topic 비율은 [(0, 0.06662666), (8, 0.9122829)]
2 번째 문서의 topic 비율은 [(1, 0.035949886), (4, 0.33516723), (8, 0.6174815)]
3 번째 문서의 topic 비율은 [(1, 0.05237408), (5, 0.262706), (6, 0.049558748), (7, 0.06902635), (8, 0.55849475)]
4 번째 문서의 topic 비율은 [(3, 0.39486828), (6, 0.21888791), (8, 0.29118967), (9, 0.07281698)]
```

In [29]:

```
def make_topictable_per_doc(ldamodel, corpus):
    topic_table = pd.DataFrame()

    # 몇 번째 문서인지를 의미하는 문서 번호와 해당 문서의 토픽 비중을 한 줄씩 꺼내온다.
    for i, topic_list in enumerate(ldamodel[corpus]):
        doc = topic_list[0] if ldamodel.per_word_topics else topic_list
        doc = sorted(doc, key=lambda x: (x[1]), reverse=True)
        # 각 문서에 대해서 비중이 높은 토픽순으로 토픽을 정렬한다.
        # EX) 정렬 전 0번 문서 : (2번 토픽, 48.5%), (8번 토픽, 25%), (10번 토픽, 5%), (12번 토픽,
        # Ex) 정렬 후 0번 문서 : (2번 토픽, 48.5%), (8번 토픽, 25%), (12번 토픽, 21.5%), (10번 토
        # 48 > 25 > 21 > 5 순으로 정렬이 된 것.

        # 모든 문서에 대해서 각각 아래를 수행
        for j, (topic_num, prop_topic) in enumerate(doc): #  몇 번 토픽인지와 비중을 나눠서 저장
            if j == 0:  # 정렬을 한 상태이므로 가장 앞에 있는 것이 가장 비중이 높은 토픽
                topic_table = topic_table.append(pd.Series([int(topic_num), round(prop_topic,4),
                # 가장 비중이 높은 토픽과, 가장 비중이 높은 토픽의 비중과, 전체 토픽의 비중을 저
            else:
                break
    return(topic_table)
```

In [30]:

```
topictable = make_topictable_per_doc(ldamodel, corpus)
topictable = topictable.reset_index() # 문서 번호을 의미하는 열(column)로 사용하기 위해서 인덱스
topictable.columns = ['문서 번호', '가장 비중이 높은 토픽', '가장 높은 토픽의 비중', '각 토픽의
topictable[:10]
```

Out[30]:

	문서 번호	가장 비중이 높은 토픽	가장 높은 토픽의 비중	각 토픽의 비중
0	0	8.0	0.5808	[(4, 0.40622187), (8, 0.5808397)]
1	1	8.0	0.9123	[(0, 0.066626824), (8, 0.9122827)]
2	2	8.0	0.6175	[(1, 0.035957728), (4, 0.33517027), (8, 0.6174...
3	3	8.0	0.5585	[(1, 0.052375074), (5, 0.26270616), (6, 0.0495...
4	4	3.0	0.3949	[(3, 0.39486882), (6, 0.21888953), (8, 0.29118...
5	5	8.0	0.8906	[(4, 0.07298025), (8, 0.8905782)]
6	6	6.0	0.5037	[(0, 0.094115466), (1, 0.060698856), (6, 0.503...
7	7	8.0	0.6578	[(4, 0.20946598), (6, 0.029368341), (7, 0.0934...
8	8	8.0	0.5608	[(4, 0.10120084), (7, 0.31721175), (8, 0.56084...

	문서 번호	가장 비중이 높은 토픽	가장 높은 토픽의 비중	각 토픽의 비중
9	9	8.0	0.6826	[(0, 0.16989952), (6, 0.13774808), (8, 0.68260...

- 토픽이 20개였던 3번과는 다른 결과인 토픽이 10개인 4번의 결과였다. 동일한 단어는 존재하지만 이 또한 비중이 다르며 각 토픽의 비중 또한 다른 것도 확인할 수 있다.