In [207]:
```
import numpy as np
import pandas as pd
import stop_words
import os

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.utils import shuffle
```

In [5]:
```
# Load whole dataset

root_folder = './newsdataset/20news-bydate-train'

docs = []

for subdir in os.listdir(root_folder):
    for doc in os.listdir(root_folder + '/' + subdir):
        with open(root_folder + '/' + subdir + '/' + doc, 'r') as f:
            docs.append(f.read())
```

In [209]:
```
# Load some selected topics

docs = []

for doc in os.listdir(root_folder + '/comp.windows.x'):
    with open(root_folder + '/comp.windows.x/' + doc, 'r') as f:
        docs.append(f.read())

for doc in os.listdir(root_folder + '/soc.religion.christian'):
    with open(root_folder + '/soc.religion.christian/' + doc, 'r') as f:
        docs.append(f.read())
```

In [213]:
```
print('{} docs of CompSci + Theology has been loaded'.format(len(docs)))
```

```
1192 docs of CompSci + Theology has been loaded
```

In [214]:
```
class PLSA(object):
    def __init__(self, num_topics, max_tokens, num_docs):
        self.num_topics = num_topics
        self.max_tokens = max_tokens
        self.num_docs = num_docs

        self.p_w_z = None
        self.p_z_d = None
        self.p_z_w_d = None

        self.p_d = None
        self.vectorizer = None

    def expectation(self):
        for token in range(self.max_tokens):
            for document in range(self.num_docs):
                denominator = np.sum(self.p_w_z[token, :] * self.p_z_d[:, document])
                self.p_z_w_d[:, token, document] = self.p_w_z[token, :] * self.p_z_d[:, document] /
denominator


    def maximization_w_z(self, X):
        # recalculate p_w_z
        for token in range(self.max_tokens):
            for topic in range(self.num_topics):
                numerator = np.sum(X[:, token] * self.p_z_w_d[topic, token, :].reshape(-1))
                denominator = np.sum(np.sum(X[:, :] * self.p_z_w_d[topic, :, :].T))

                self.p_w_z[token, topic] = numerator / denominator

    def maximization_z_d(self, X):
        # recalculate p_z_d
        for topic in range(self.num_topics):
            for document in range(self.num_docs):
                numerator = np.sum(X[document, :] * self.p_z_w_d[topic, :, document])
                self.p_z_d[topic, document] = numerator / np.sum(X[document:])


    def fit(self, X, y=None):
        self.vectorizer = CountVectorizer(ngram_range=(1, 1), max_features=self.max_tokens, stop_wor
ds=stop_words.get_stop_words('en'))
        X = np.asarray(self.vectorizer.fit_transform(X).todense())

        # initialization of parameters to be estimated
        self.p_w_z = np.random.uniform(0,1, size=(self.max_tokens, self.num_topics))
        self.p_z_d = np.random.uniform(0,1, size=(self.num_topics, self.num_docs))
        self.p_z_w_d = np.zeros((self.num_topics, self.max_tokens, self.num_docs))
        #
        for i in range(20):
            print('iteration {}'.format(i+1))
            print('E-step')
            self.expectation()
            print('M-step')
            self.maximization_w_z(X)
            self.maximization_z_d(X)
        return self.p_z_w_d
```

In [215]:
```
docs = shuffle(docs)
```

In [217]:
```
plsa = PLSA(2, 1000, len(docs))
X = plsa.fit(docs)
```

```
iteration 0
E-step
M-step
iteration 1
E-step
M-step
iteration 2
E-step
M-step
iteration 3
E-step
M-step
iteration 4
E-step
M-step
iteration 5
E-step
M-step
iteration 6
E-step
M-step
iteration 7
E-step
M-step
iteration 8
E-step
M-step
iteration 9
E-step
M-step
iteration 10
E-step
M-step
iteration 11
E-step
M-step
iteration 12
E-step
M-step
iteration 13
E-step
M-step
iteration 14
E-step
M-step
iteration 15
E-step
M-step
iteration 16
E-step
M-step
iteration 17
E-step
M-step
iteration 18
E-step
M-step
iteration 19
E-step
M-step
```

In [218]:
```
df = pd.DataFrame(plsa.p_w_z, index = plsa.vectorizer.get_feature_names())
```

In [223]:
```
keywords = []
for i in range(df.columns.size):
    keywords.append(df[i].sort_values(ascending=False).iloc[:10].index.tolist())
```

In [224]:
```
pd.DataFrame(keywords, index = list(map(lambda x: 'topic_%d' % x, range(2))))
```

Out[224]:

	0	1	2	3	4	5	6	7	8	9
topic_0	god	edu	can	one	will	re	people	subject	lines	jesus
topic_1	com	edu	window	can	subject	lines	file	organization	use	mit

In [ ]:
```

```