Import necessary dependencies and settings

In [1]:

```
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt

pd.options.display.max_colwidth = 200
%matplotlib inline
```

Sample corpus of text documents

In [2]:

```
corpus = ['The sky is blue and beautiful.',
          'Love this blue and beautiful sky!',
          'The quick brown fox jumps over the lazy dog.',
          "A king's breakfast has sausages, ham, bacon, eggs, toast and beans
          'I love green eggs, ham, sausages and bacon!',
          'The brown fox is quick and the blue dog is lazy!',
          'The sky is very blue and the sky is very beautiful today',
          'The dog is lazy but the brown fox is quick!'
]
labels = ['weather', 'weather', 'animals', 'food', 'food', 'animals', 'weathe

corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus,
                          'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
```

Out[2]:

	Document	Category
0	The sky is blue and beautiful.	weather
1	Love this blue and beautiful sky!	weather
2	The quick brown fox jumps over the lazy dog.	animals
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food
4	I love green eggs, ham, sausages and bacon!	food
5	The brown fox is quick and the blue dog is lazy!	animals
6	The sky is very blue and the sky is very beautiful today	weather
7	The dog is lazy but the brown fox is quick!	animals

Simple text pre-processing

In [3]:

```
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)
```

In [4]:

```
norm_corpus = normalize_corpus(corpus)
norm_corpus
```

Out[4]:

```
array(['sky blue beautiful', 'love blue beautiful sky',
       'quick brown fox jumps lazy dog',
       'kings breakfast sausages ham bacon eggs toast beans',
       'love green eggs ham sausages bacon',
       'brown fox quick blue dog lazy', 'sky blue sky beautiful today',
       'dog lazy brown fox quick'], dtype='<U51')
```

Bag of Words Model

In [5]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix = cv_matrix.toarray()
cv_matrix
```

Out[5]:

```
array([[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0],
       [1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1],
       [0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]],
      dtype=int64)
```

In [6]:

```
# get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)
```

Out[6]:

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	king
0	0	0	1	1	0	0	0	0	0	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	1	1	0	1	0	0	1	
3	1	1	0	0	1	0	0	1	0	0	1	0	
4	1	0	0	0	0	0	0	1	0	1	1	0	
5	0	0	0	1	0	1	1	0	1	0	0	0	
6	0	0	1	1	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	1	1	0	1	0	0	0	

Bag of N-Grams Model

In [7]:

```
# you can set the n-gram range to 1,2 to get unigrams as well as bigrams
bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = bv_matrix.toarray()
vocab = bv.get_feature_names()
pd.DataFrame(bv_matrix, columns=vocab)
```

Out[7]:

	bacon eggs	beautiful sky	beautiful today	blue beautiful	blue dog	blue sky	breakfast sausages	brown fox	dog lazy	eggs ham	...	lazy dog	lc b
0	0	0	0	1	0	0	0	0	0	0	...	0	
1	0	1	0	1	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	1	0	0	...	1	
3	1	0	0	0	0	0	1	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	1	...	0	
5	0	0	0	0	1	0	0	1	1	0	...	0	
6	0	0	1	0	0	1	0	0	0	0	...	0	
7	0	0	0	0	0	0	0	1	1	0	...	0	

8 rows × 29 columns

TF-IDF Model

In [8]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()

vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

Out[8]:

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	kir
0	0.00	0.00	0.60	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
1	0.00	0.00	0.49	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
2	0.00	0.00	0.00	0.00	0.00	0.38	0.38	0.00	0.38	0.00	0.00	0.53	0
3	0.32	0.38	0.00	0.00	0.38	0.00	0.00	0.32	0.00	0.00	0.32	0.00	0
4	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.00	0.47	0.39	0.00	0
5	0.00	0.00	0.00	0.37	0.00	0.42	0.42	0.00	0.42	0.00	0.00	0.00	0
6	0.00	0.00	0.36	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0.00	0.00	0.45	0.45	0.00	0.45	0.00	0.00	0.00	0

Document Similarity

In [9]:

```python
from sklearn.metrics.pairwise import cosine_similarity

similarity_matrix = cosine_similarity(tv_matrix)
similarity_df = pd.DataFrame(similarity_matrix)
similarity_df
```

Out[9]:

	0	1	2	3	4	5	6	7
0	1.000000	0.820599	0.000000	0.000000	0.000000	0.192353	0.817246	0.000000
1	0.820599	1.000000	0.000000	0.000000	0.225489	0.157845	0.670631	0.000000
2	0.000000	0.000000	1.000000	0.000000	0.000000	0.791821	0.000000	0.850516
3	0.000000	0.000000	0.000000	1.000000	0.506866	0.000000	0.000000	0.000000
4	0.000000	0.225489	0.000000	0.506866	1.000000	0.000000	0.000000	0.000000
5	0.192353	0.157845	0.791821	0.000000	0.000000	1.000000	0.115488	0.930989
6	0.817246	0.670631	0.000000	0.000000	0.000000	0.115488	1.000000	0.000000
7	0.000000	0.000000	0.850516	0.000000	0.000000	0.930989	0.000000	1.000000

Clustering documents using similarity features

In [10]:

```
from scipy.cluster.hierarchy import dendrogram, linkage

Z = linkage(similarity_matrix, 'ward')
pd.DataFrame(Z, columns=['Document\Cluster 1', 'Document\Cluster 2',
                         'Distance', 'Cluster Size'], dtype='object')
```

Out[10]:

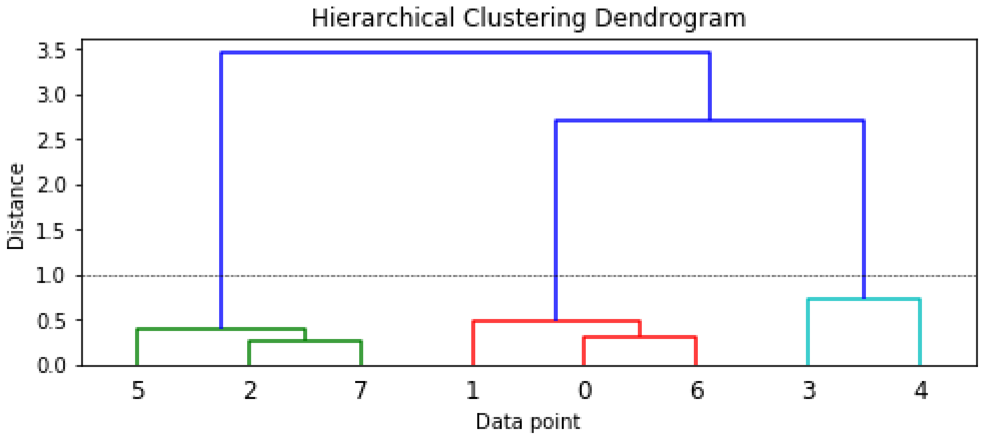
	Document\Cluster 1	Document\Cluster 2	Distance	Cluster Size
0	2	7	0.253098	2
1	0	6	0.308539	2
2	5	8	0.386952	3
3	1	9	0.489845	3
4	3	4	0.732945	2
5	11	12	2.69565	5
6	10	13	3.45108	8

In [11]:

```
plt.figure(figsize=(8, 3))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data point')
plt.ylabel('Distance')
dendrogram(Z)
plt.axhline(y=1.0, c='k', ls='--', lw=0.5)
```

Out[11]:

```
<matplotlib.lines.Line2D at 0x2168ecea688>
```



In [12]:

```
from scipy.cluster.hierarchy import fcluster
max_dist = 1.0

cluster_labels = fcluster(Z, max_dist, criterion='distance')
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
pd.concat([corpus_df, cluster_labels], axis=1)
```

Out[12]:

	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	2
1	Love this blue and beautiful sky!	weather	2
2	The quick brown fox jumps over the lazy dog.	animals	1
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	3
4	I love green eggs, ham, sausages and bacon!	food	3
5	The brown fox is quick and the blue dog is lazy!	animals	1
6	The sky is very blue and the sky is very beautiful today	weather	2
7	The dog is lazy but the brown fox is quick!	animals	1

Topic Models

In [14]:

```
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(3, max_iter=10000, random_state=0)
dt_matrix = lda.fit_transform(cv_matrix)
features = pd.DataFrame(dt_matrix, columns=['T1', 'T2', 'T3'])
features
```

Out[14]:

	T1	T2	T3
0	0.832191	0.083480	0.084329
1	0.863554	0.069100	0.067346
2	0.047794	0.047776	0.904430
3	0.037243	0.925559	0.037198
4	0.049121	0.903076	0.047802
5	0.054902	0.047778	0.897321
6	0.888287	0.055697	0.056016
7	0.055704	0.055689	0.888607

Show topics and their weights

In [15]:

```
tt_matrix = lda.components_
for topic_weights in tt_matrix:
    topic = [(token, weight) for token, weight in zip(vocab, topic_weights)]
    topic = sorted(topic, key=lambda x: -x[1])
    topic = [item for item in topic if item[1] > 0.6]
    print(topic)
    print()
```

```
[('sky', 4.3324394424701325), ('blue', 3.373774254787669), ('beautiful', 3.
3323650509884386), ('today', 1.3325579855138987), ('love', 1.33041581821754
8)]

[('bacon', 2.33269586574902), ('eggs', 2.33269586574902), ('ham', 2.3326958
6574902), ('sausages', 2.33269586574902), ('love', 1.3354610533796556), ('b
eans', 1.3327735190105536), ('breakfast', 1.3327735190105536), ('kings', 1.
3327735190105536), ('toast', 1.3327735190105536), ('green', 1.3325431515674
175)]

[('brown', 3.3323473548404405), ('dog', 3.3323473548404405), ('fox', 3.3323
473548404405), ('lazy', 3.3323473548404405), ('quick', 3.3323473548404405),
('jumps', 1.3324193772908193), ('blue', 1.2919423137963386)]
```

Clustering documents using topic model features

In [16]:

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=3, random_state=0)
km.fit_transform(features)
cluster_labels = km.labels_
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
pd.concat([corpus_df, cluster_labels], axis=1)
```

Out[16]:

	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	1
1	Love this blue and beautiful sky!	weather	1
2	The quick brown fox jumps over the lazy dog.	animals	2
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	0
4	I love green eggs, ham, sausages and bacon!	food	0
5	The brown fox is quick and the blue dog is lazy!	animals	2
6	The sky is very blue and the sky is very beautiful today	weather	1
7	The dog is lazy but the brown fox is quick!	animals	2