Assignment 6: Clustering and Topic Modeling

In this assignment, you'll need to use the following dataset:

- text_train.json: This file contains a list of documents. It's used for training models
- text_test.json: This file contains a list of documents and their ground-truth labels. It's used for testing performance. This file is in the format shown below. Note, each document has a list of labels. You can load these files using json.load()

Text	Labels
paraglider collides with hot air balloon ...	['Disaster and Accident', 'Travel & Transportation']
faa issues fire warning for lithium ...	['Travel & Transportation']
....	...

Q1: K-Mean Clustering

Define a function **cluster_kmean()** as follows:

- Take two file name strings as inputs: $train\_file$ is the file path of text_train.json, and $test\_file$ is the file path of text_test.json
- Use **KMeans** to cluster documents in $train\_file$ into 3 clusters by **cosine similarity**
- Test the clustering model performance using $test\_file$:
  - Predict the cluster ID for each document in $test\_file$.
  - Let's only use the **first label** in the ground-truth label list of each test document, e.g. for the first document in the table above, you set the ground_truth label to "Disaster and Accident" only.
  - Apply **majority vote** rule to dynamically map the predicted cluster IDs to the ground-truth labels in $test\_file$. **Be sure not to hardcode the mapping** (e.g. write code like {0: "Disaster and Accident"}), because a cluster may corrspond to a different topic in each run.
  - Calculate **precision/recall/f-score** for each label
- This function has no return. Print out confusion matrix, precision/recall/f-score.

Q2: LDA Clustering

Define a function **cluster_lda()** as follows:

- Take two file name strings as inputs: $train\_file$ is the file path of text_train.json, and $test\_file$ is the file path of text_test.json
- Use **LDA** to train a topic model with documents in $train\_file$ and the number of topics $K = 3$
- Predict the topic distribution of each document in $test\_file$, and select **only the topic with highest probability** as the predicted topic
- Evaluates the topic model performance as follows:
  - Similar to Q1, let's use the **first label** in the label list of $test\_file$ as the ground_truth label.
  - Apply **majority vote rule** to map the topics to the labels.
  - Calculate **precision/recall/f-score** for each label and print out precision/recall/f-score.
- Return topic distribution and the original ground-truth labels of each document in $test\_file$
- Also, provide a document which contains:
  - performance comparison between Q1 and Q2
  - describe how you tune the model parameters, e.g. min_df, alpha, max_iter etc.

Q3 (Bonus): Overlapping Clustering

In Q2, you predict one label for each document in $test\_file$. In this question, try to discover multiple labels if appropriate. Define a function **overlapping_cluster** as follows:

- Take the outputs of Q2 (i.e. topic distribution and the labels of each document in $test\_file$) as inputs
- Set a threshold for each topic (i.e. $TH = [th_0, th_1, th_2]$). A document is predicted to belong to a topic i only if the topic probability > th_i for $i \in [0, 1, 2]$.
- The threshold is determined as follows:
  - Vary the threshold for each topic from 0.05 to 0.95 with an increase of 0.05 in each round to evalute the topic model performance:
    - Apply **majority vote rule** to map the predicted topics to the ground-truth labels in $test\_file$
    - Calculate **f1-score** for each label
  - For each label, pick the threshold value which maximizes the f1-score
- Return the threshold and f1-score of each label

In [145]:

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.cluster import KMeansClusterer, cosine_distance
from sklearn.decomposition import LatentDirichletAllocation

# add more
```

In [146]:

```python
def cluster_kmean(train_file, test_file):

    # add your code
```

In [148]:

```python
def cluster_lda(train_file, test_file):
    topic_assig = None
    labels = None

    # add your code here

    return topic_assign, labels
```

In [ ]:

```python
def overlapping_cluster(topic_assign, labels):
    final_thresh, f1 = None, None

    # add your code here

    return final_thresh, f1
```

In [150]:

```python
if __name__ == "__main__":

    # Due to randomness, you won't get the exact result
    # as shown here, but your result should be close
    # if you tune the parameters carefully

    # Q1
    cluster_kmean('../../dataset/train_text.json', \
                  '../../dataset/test_text.json')

    # Q2
    topic_assign, labels =cluster_lda('../../dataset/train_text.json', \
                          '../../dataset/test_text.json')

    # Q2
    threshold, f1 = overlapping_cluster(topic_assign, labels)
    print(threshold)
    print(f1)
```

```
actual_class  Disaster and Accident  News and Economy  Travel & Tran
sportation
cluster
0                               70                 0
135
1                              130                 7
```

```
8
2                                 10               199
41
Cluster 0: Topic Travel & Transportation
Cluster 1: Topic Disaster and Accident
Cluster 2: Topic News and Economy
                         precision    recall  f1-score   support

  Disaster and Accident       0.90      0.62      0.73       210
       News and Economy       0.80      0.97      0.87       206
Travel & Transportation       0.66      0.73      0.69       184

              micro avg       0.77      0.77      0.77       600
              macro avg       0.78      0.77      0.77       600
           weighted avg       0.79      0.77      0.77       600

iteration: 1 of max_iter: 25
iteration: 2 of max_iter: 25
iteration: 3 of max_iter: 25
iteration: 4 of max_iter: 25
iteration: 5 of max_iter: 25
iteration: 6 of max_iter: 25
iteration: 7 of max_iter: 25
iteration: 8 of max_iter: 25
iteration: 9 of max_iter: 25
iteration: 10 of max_iter: 25
iteration: 11 of max_iter: 25
iteration: 12 of max_iter: 25
iteration: 13 of max_iter: 25
iteration: 14 of max_iter: 25
iteration: 15 of max_iter: 25
iteration: 16 of max_iter: 25
iteration: 17 of max_iter: 25
iteration: 18 of max_iter: 25
iteration: 19 of max_iter: 25
iteration: 20 of max_iter: 25
iteration: 21 of max_iter: 25
iteration: 22 of max_iter: 25
iteration: 23 of max_iter: 25
iteration: 24 of max_iter: 25
iteration: 25 of max_iter: 25
actual_class  Disaster and Accident  News and Economy  Travel & Tran
sportation
cluster
0                                 30                18
138
1                                 12               182
8
2                                168                 6
38
Cluster 0: Topic Travel & Transportation
Cluster 1: Topic News and Economy
Cluster 2: Topic Disaster and Accident
```

```
                         precision    recall  f1-score   support

  Disaster and Accident       0.79      0.80      0.80       210
       News and Economy       0.90      0.88      0.89       206
Travel & Transportation       0.74      0.75      0.75       184

              micro avg       0.81      0.81      0.81       600
              macro avg       0.81      0.81      0.81       600
           weighted avg       0.81      0.81      0.81       600

Disaster and Accident      0.45
News and Economy           0.55
Travel & Transportation    0.30
dtype: float64
Disaster and Accident      0.798122
News and Economy           0.888889
Travel & Transportation    0.773218
dtype: float64
```

In [ ]: