Assignment 5: 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
- When generating tfidf weights, set the min_df to 5.
- Use **KMeans** to cluster documents in $train\_file$ into 3 clusters by **cosine similarity** and **Euclidean distance** separately. Use sufficient iterations with different initial centroids to make sure clustering converge
- 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. (hint: if you use pandas, look for "idxmax" function)
  - Calculate **precision/recall/f-score** for each label, compare the results from the two clustering models, and write your analysis in a pdf file
- This function has no return. Print out confusion matrix, precision/recall/f-score.

Q2: LDA Clustering

Q2.1. Define a function **cluster_lda()** as follows:

1. 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
2. Use **LDA** to train a topic model with documents in $train\_file$ and the number of topics $K = 3$. Keep min_df to 5 when generating tfidf weights, as in Q1.
3. Predict the topic distribution of each document in $test\_file$ and select the topic with highest probability. Similar to Q1, apply **majority vote rule** to map the topics to the labels and show the classification report.
4. Return the array of topic proportion array

Q2.2. Find similar documents

- Define a function **find_similar_doc(doc_id, topic_mix)** to find **top 3 documents** that are the most similar to a selected one with index **doc_id** using the topic proportion array **topic_mix**.
- You can calculate the cosine or Euclidean distance between two documents using the topic proportion array
- Return the IDs of these similar documents.

Q2.3. Provide a pdf document which contains:

- performance comparison between Q1 and Q2.1
- describe how you tune the model parameters, e.g. alpha, max_iter etc. in Q2.1.
- discuss how effective the method in Q2.2 is to find similar documents, compared with the tfidf weight cosine similarity we used before.

Q3 (Bonus): Biterm Topic Model (BTM)

- There are many variants of LDA model. BTM is one designed for short text, while lDA in general expects documents with rich content.
- Read this paper carefully http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4032&rep=rep1&type=pdf (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4032&rep=rep1&type=pdf) and try to understand the design
- Try the following experiments:
  - Script a few thousand tweets by different hastags
  - Run LDA and BTM respectively to discover topics among the collected tweets. BTM package can be found at https://pypi.org/project/biterm/ (https://pypi.org/project/biterm/)
  - Compare the performance of each model. If one model works better, explain why it works better,
- Summarize your experiment in a pdf document.
- Note there is no absolute right or wrong answer in this experiment. All you need is to give a try and understand how BTM works and differences between BTM and LDA

Note: Due to randomness involved in these alogorithms, you may get the same result as what I showed below. However, your result should be close after you tune parameters carefully.

In [5]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# addd your import
```

In [6]:
```
# Q1
def cluster_kmean(train_file, test_file):

    # add your code here

    return None
```

In [11]:
```
# Q2
def cluster_lda(train_file, test_file):

    topic_assign = None

    # add your code here

    return topic_assign

def find_similar(doc_id, topic_assign):

    docs = None

    # add your code here

    return docs
```

In [12]:
```
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
    print("Q1")
    cluster_kmean('../../dataset/train_text.json', \
                  '../../dataset/test_text.json')

    # Q2
    print("\nQ2")
    topic_assign =cluster_lda('../../dataset/train_text.json', \
        '../../dataset/test_text.json')
    doc_ids = find_similar(10, topic_assign)
    print ("docs similar to {0}: {1}".format(10, doc_ids))
```

```
Q1
cosine
actual_class  Disaster and Accident  News and Economy  Travel & Transportation
cluster
0                                61                 2                      152
1                               109                 7                       25
2                                40               197                        7
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       0.77      0.52      0.62       210
       News and Economy       0.81      0.96      0.88       206
Travel & Transportation       0.71      0.83      0.76       184

              micro avg       0.76      0.76      0.76       600
              macro avg       0.76      0.77      0.75       600
           weighted avg       0.76      0.76      0.75       600

L2
actual_class  Disaster and Accident  News and Economy  Travel & Transportation
cluster
0                               174                34                      174
1                                31               166                       10
2                                 5                 6                        0
Cluster 0: Topic Disaster and Accident
Cluster 1: Topic News and Economy
Cluster 2: Topic News and Economy
                         precision    recall  f1-score   support

  Disaster and Accident       0.46      0.83      0.59       210
       News and Economy       0.79      0.83      0.81       206
Travel & Transportation       0.00      0.00      0.00       184

              micro avg       0.58      0.58      0.58       600
              macro avg       0.41      0.55      0.47       600
           weighted avg       0.43      0.58      0.48       600


Q2

/Users/rliu/anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/class
ification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
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, perplexity: 3494.8408
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, perplexity: 3416.5917
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, perplexity: 3382.7160
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, perplexity: 3377.7126
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, perplexity: 3375.9923
actual_class  Disaster and Accident  News and Economy  Travel & Transportation
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

cluster
0    Travel & Transportation
1           News and Economy
2      Disaster and Accident
dtype: object
docs similar to 10: [337  38 222]
```

In [ ]: