HW2 Text Clustering

2-1: SBERT Tutorial

What is the size of the representation vector?

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')

#Our sentences we like to encode
sentences = ['This framework generates embeddings for each input sentence',
        'Sentences are passed as a list of string.',
        'The quick brown fox jumps over the lazy dog.']

#Sentences are encoded by calling model.encode()
embeddings = model.encode(sentences)
```

2-2: SBERT Clustering

Limiting data to 5000, since agglomerative costs O(n^2) memory and O(n^3) time.

```
import os
import csv
from sentence_transformers import SentenceTransformer, util
# We donwload the Quora Duplicate Questions Dataset (https://www.quora.com/q/quoradata/Firs
# and find similar question in it
url = "http://qim.fs.quoracdn.net/quora_duplicate_questions.tsv"
dataset_path = "quora_duplicate_questions.tsv"
max_corpus_size = 5000  # We limit our corpus to only the first 5k questions


# Check if the dataset exists. If not, download and extract
# Download dataset if needed
if not os.path.exists(dataset_path):
        print("Download dataset")
        util.http_get(url, dataset_path)

# Get all unique sentences from the file
corpus_sentences = set()
with open(dataset_path, encoding='utf8') as fIn:
        reader = csv.DictReader(fIn, delimiter='\t', quoting=csv.QUOTE_MINIMAL)
        for row in reader:
                corpus_sentences.add(row['question1'])
                corpus_sentences.add(row['question2'])
```

```
            if len(corpus_sentences) >= max_corpus_size:
                break

corpus_sentences = list(corpus_sentences)
print("Encode the corpus. This might take a while")

embedder = SentenceTransformer('all-MiniLM-L6-v2')
corpus_embeddings = embedder.encode(corpus_sentences)
```

KMeans

What is the running time cost of this algorithm?

```
%%time
from sklearn.cluster import KMeans
# Perform kmean clustering
num_clusters = 5
clustering_model = KMeans(n_clusters=num_clusters)
clustering_model.fit(corpus_embeddings)
cluster_assignment = clustering_model.labels_
```

```
CPU times: user 3.16 s, sys: 1.43 s, total: 4.58 s
Wall time: 3.12 s
```

How many clusters are created for this algorithm?

```
len(set(cluster_assignment))
```

```
5
```

Agglomerative

What is the running time cost of this algorithm?

```
%%time
from sklearn.cluster import AgglomerativeClustering
clustering_model = AgglomerativeClustering(n_clusters=None, affinity='cosine', linkage='average
clustering_model.fit(corpus_embeddings)
cluster_assignment = clustering_model.labels_
```

```
CPU times: user 5.23 s, sys: 19.5 ms, total: 5.25 s
Wall time: 5.32 s
```

How many clusters are created for this algorithm?

```
len(set(cluster_assignment))
```

```
1
```

2-3: PCA

Size of reduced embeddings:

```
reduced_emb.shape[1]
```

```
128
```

KMeans

What is the running time cost of this algorithm?

```
%%time
from sklearn.cluster import KMeans
# Perform kmean clustering
num_clusters = 5
clustering_model = KMeans(n_clusters=num_clusters)
clustering_model.fit(reduced_emb)
cluster_assignment = clustering_model.labels_
```

```
CPU times: user 1.54 s, sys: 1.07 s, total: 2.61 s
Wall time: 1.49 s
```

How many clusters are created for this algorithm?

```
len(set(cluster_assignment))
```

```
5
```

Agglomerative

What is the running time cost of this algorithm?

```
%%time
from sklearn.cluster import AgglomerativeClustering
clustering_model = AgglomerativeClustering(n_clusters=None, affinity='cosine', linkage='average
clustering_model.fit(reduced_emb)
cluster_assignment = clustering_model.labels_
```

```
CPU times: user 2.09 s, sys: 15.8 ms, total: 2.1 s
Wall time: 2.09 s
```

How many clusters are created for this algorithm?

```
len(set(cluster_assignment))
```

```
5
```

2-4: Real-world Application

KMeans

```
%%time
from sklearn.cluster import KMeans
# Perform kmean clustering
num_clusters = len(set(category))
clustering_model = KMeans(n_clusters=num_clusters)
clustering_model.fit(reduced_emb)
km_result = clustering_model.labels_
```

```
CPU times: user 6.02 s, sys: 3.69 s, total: 9.71 s
Wall time: 5.05 s
```

```
adjusted_rand_score(km_result,category)
```

```
0.496962390634S7176
```

```
normalized_mutual_info_score(km_result,category)
```

```
0.7882765311198084
```

Agglomerative

```
%%time
from sklearn.cluster import AgglomerativeClustering
num_clusters = len(set(category))
clustering_model = AgglomerativeClustering(n_clusters=num_clusters, affinity='cosine', linkage=
clustering_model.fit(reduced_emb)
ag_result = clustering_model.labels_
```

```
CPU times: user 985 ms, sys: 6.57 ms, total: 991 ms
Wall time: 1.22 s
```

```
adjusted_rand_score(ag_result,category)
```

```
0.3405396121349602
```

```
normalized_mutual_info_score(ag_result,category)

    0.763132026149143
```