Information Processing and Retrieval

Instituto Superior Técnico 2020

Lab 5: Organizing document collections

Today we will use the *20 Newsgroup* dataset.
You can find more information at http://qwone.com/~jason/20Newsgroups/
The scikit-learn library already provides access to the *20 Newsgroups* dataset.

```
from sklearn.datasets import fetch_20newsgroups
collection = fetch_20newsgroups()
```

The actual data is in text format. You need to transform it into numeric weight vectors (using for instance the term frequency or TF-IDF vector space model). As we saw in earlier labs, the scikit-learn library provides methods for this:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer( use_idf=False )
vectorspace = vectorizer.fit_transform(collection.data)
```

1 Clustering the 20 NewsGroup collection

Clustering of documents can be also achieved using scikit-learn library.

1.1

Extract the vector space of the 20 Newsgroup (note that the 'collection.data' instruction removed the class variable), and cluster the collection using agglomerative clustering available from scikit-learn.

```
from sklearn.cluster import AgglomerativeClustering
clustering = AgglomerativeClustering().fit(vectorspace)
print(clustering.labels_)
```

Parameterize the clustering search to use cosine as the distance function.

```
model = AgglomerativeClustering(n_clusters=n_clusters,
                                linkage="average",
                                affinity="cosine")
```

1.2

Plot the learned dendogram.
Compare the clustering solutions produced under single and complete linkage criteria.

```
from scipy.cluster.hierarchy import dendrogram

def plot_dendrogram(model, **kwargs):
    # create linkage matrix and then plot the dendrogram
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)
    dendrogram(linkage_matrix, **kwargs)

model = model.fit(X)
plot_dendrogram(model, truncate_mode='level', p=3) # plot top 3 levels
plt.show()
```

1.3

Evaluate the clustering solution by computing an internal measure (e.g. *silhouette*) and an external measure (e.g. *adjusted rand index*) for the produced clustering solution.

```
silhouette_score(vectorspace, cluster_labels, metric='cosine')
adjusted_rand_score(cluster_labels, true_labels)
```

1.4 (*homework*)

Principal component analysis (PCA) offers a way of projecting our high-dimensional vector space into a space with lower dimensionality. Map the original vector space into a two-dimensional space.

```
newspace = PCA(n_components=2).fit(vectorspace)
```

Challenge: Plot the documents as points in this new space. You can also color documents according to their cluster to assess how well-separated are the clusters in this space.

2 Pen-and-paper exercises

2.1 Performing clustering

Consider the following collection of 4 text documents.

ID	text document
1	shipment of gold damaged in fire
2	delivery of silver arrived in silver truck
3	shipment of silver arrived in truck
4	truck damaged in fire

Consider documents to be represented as a Boolean model, and their similarity assessed under the Manhattan distance, $\|\mathbf{d}_1 - \mathbf{d}_2\|_1 = \sum_{i=1}^{n} |d_1^i - d_2^i|$.

Compute the pairwise distance matrix and the dendogram obtained with the complete (maximum) link criterion.

2.2 Evaluating clustering

Considering the clustering solution produced for the collection of documents in previous exercise, assume the presence of the following ground truth:

$$\mathcal{C}(\{d_1, d_2, d_3, d_4\}) =< A, B, B, B >$$

2.2.1 Compute the following external scores:

(a) *purity*

(b) *rand index*

2.2.2 Compute the silhouette of each cluster and of the overall clustering solution.