Document Clustering & Topic Models

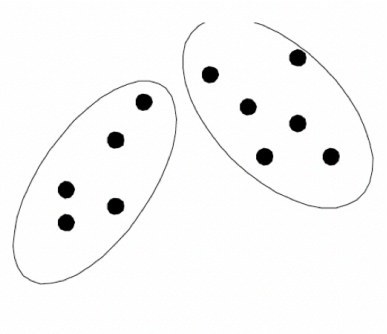
Ch.4 Text Analytics with Python. Dipanjan Sankar. 2019. Apress

Clustering

an unsupervised learning to group data points (documents) into groups or clusters

Partitional

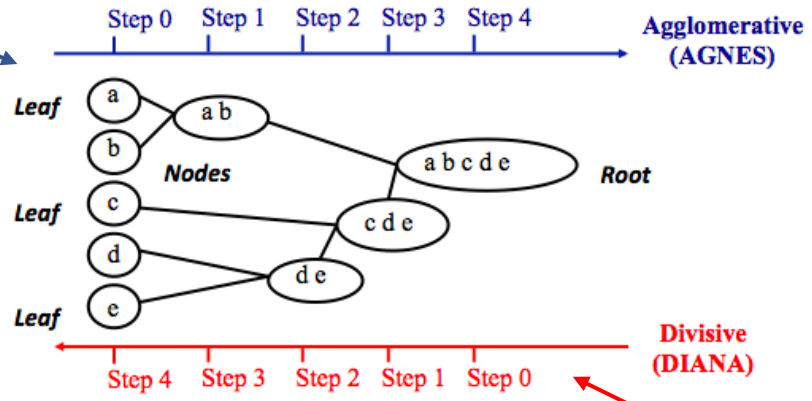
A division in non-overlapping groups



Hierarchical

Subsets are nested

bottom-up



top-down

UC Berkeley. 2017. Tutorial on Clustering

Agglomerative Algorithm

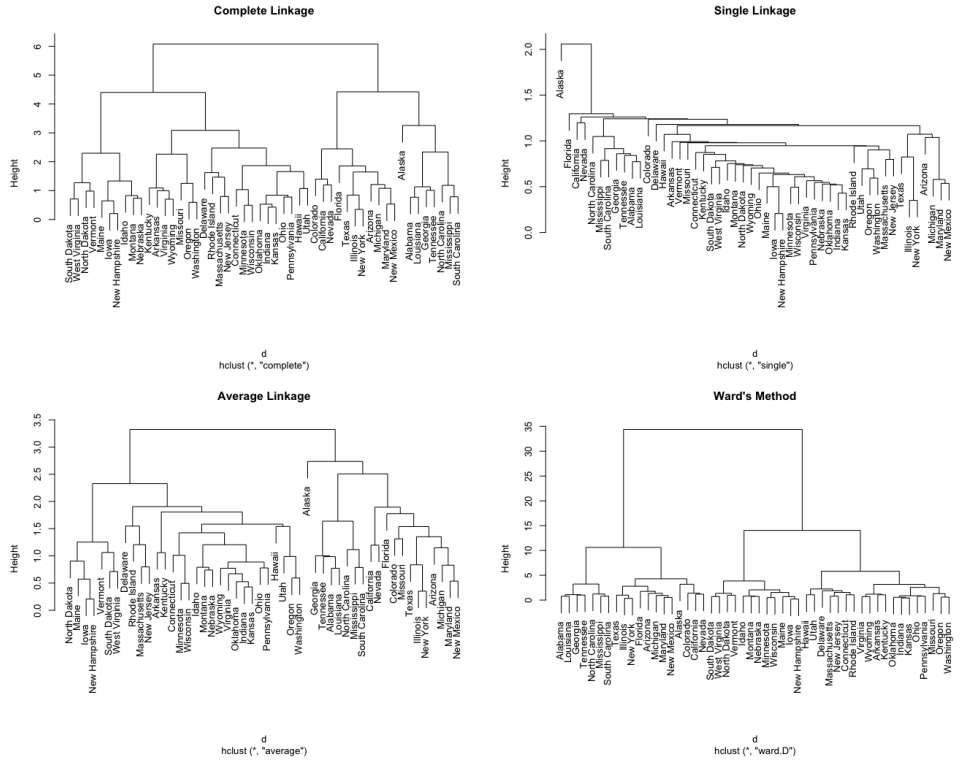
How to measure dissimilarity?

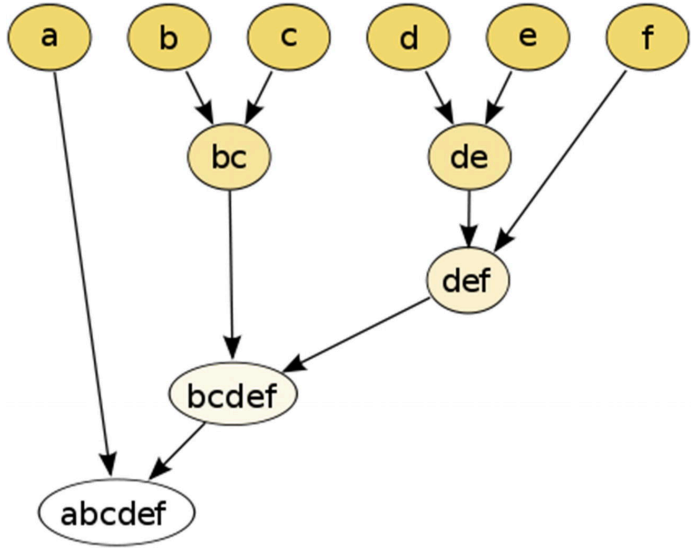
Maximum or complete linkage

computes all pairwise dissimilarities and considers the largest value (i.e., maximum value)

Mean or average linkage

computes all pairwise dissimilarities and considers the average value between two clusters





Minimum or single linkage

computes all pairwise dissimilarities and considers the smallest value (i.e., minimum value)

Ward linkage

minimized the total within-cluster variance.

https://uc-r.github.io/hc_clustering

Dipanjan. Sarkar. 2019. Chapter 4.

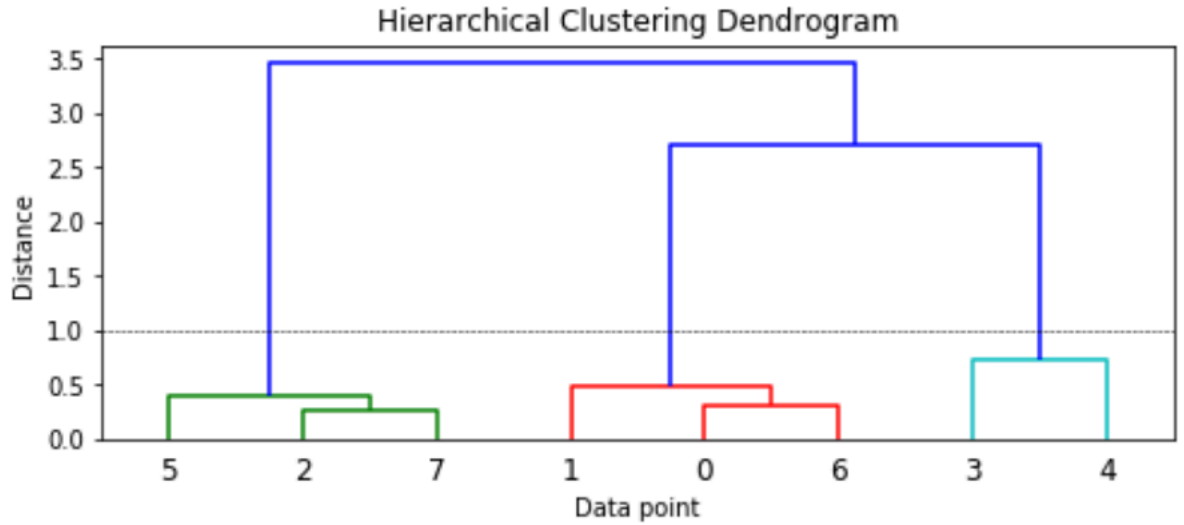
Pairwise Document Similarity

	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

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_matrix = cosine_similarity(tv_matrix)
```

tf-idf scores from the normalized corpus (Ch.4. Sankar)

```
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")
```




```
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)
plt.show()
```

Document Labels

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

	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, e...	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 beaut...	weather	2
7	The dog is lazy but the brown fox is quick!	animals	1

```
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)
```

Topic Models

The process of extracting key themes or concepts from a corpus of documents

Document-term Matrix

document-topic matrix (feature matrix)

topic-term matrix (potential topics in the corpus)

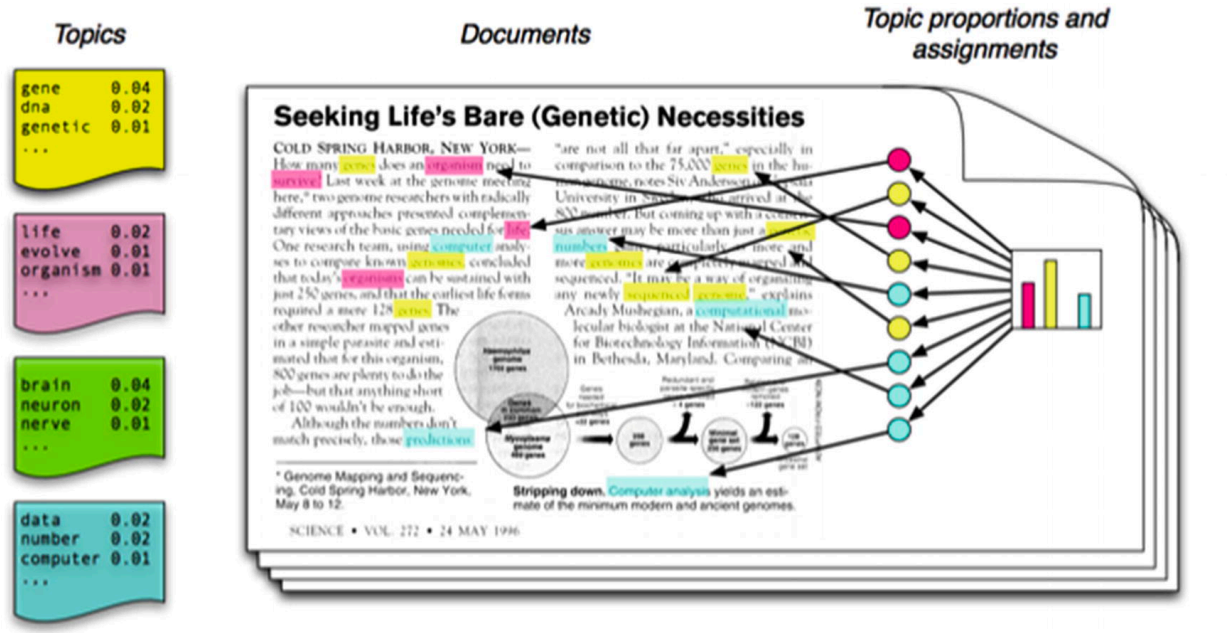


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

Topic Models

	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

n_components=
number of topics

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

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

```
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), ('lov
e', 1.330415818217548)]

[('bacon', 2.33269586574902), ('eggs', 2.33269586574902), ('ham', 2.33269586574902), ('sausages', 2.33269586574902), ('love', 1.33
54610533796558), ('beans', 1.3327735190105536), ('breakfast', 1.3327735190105536), ('kings', 1.3327735190105536), ('toast', 1.3327
735190105536), ('green', 1.3325431515674175)]

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