Machine Learning Techniques for Text

Module 10: Clustering Speech-to-Text Transcriptions

Dr. Nikos Tsourakis

Course outline

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Module 0: Python Crash Course
- Module 1: Intro to Machine Learning
- Module 2: Detecting Spam Emails
- Module 3: Classifying Topics of Newsgroup Posts
- Module 4: Extracting Sentiments from Product Reviews
- Module 5: Recommending Music Titles
- Module 6: Teaching Machines to Translate
- Module 7: Summarizing Wikipedia Articles
- Module 8: Detecting Hateful and Offensive Language
- Module 9: Generating Text in Chatbots
- **Module 10: Clustering Speech-to-Text Transcriptions**

Overview

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- When dealing with real-world datasets, the most common situation is that they come unlabeled—manually labeling each sample is often unrealistic
- Unsupervised learning algorithms are applicable in this case and, in this module, we deal with a particular kind for grouping similar data under the same category
- We incorporate clustering methods that allow to identify the general theme in each cluster
  - While the previous modules focused mainly on supervised learning techniques, we dedicate the current one solely to unsupervised methods
  - Another differentiation is the creation of the text corpus using speech-to-text technology
  - Next, we present hard and soft clustering techniques, providing insight into their mechanics
  - Finally, we discuss how to evaluate the clustering results

Module objectives

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

After completing this module, you should be able to:

- Understanding the different techniques for text clustering
- Implementing and configuring the methods for text clustering
- Assessing the performance of the implemented systems
- Applying and evaluating speech-to-text for creating data

Machine Learning Techniques for Text

Section 1: Understanding text clustering

Introduction

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Various stakeholders benefit from discovering insights in the chaos of unstructured data and seizing potential opportunities
- Algorithms that learn the structure of this data without any assistance (no labels or classes given) are part of ***unsupervised learning*** intending to cluster text data into different categories automatically
- ***Text clustering*** is the process of dividing a population of samples into various groups such that the data points in the same category are more similar than those in other ones
- The aim is to locate functional patterns within each group and decipher why this happens

Introduction

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- ***Hard clustering*** is about grouping each data observation into a different cluster
  - For example, in a marketing survey, each customer is assigned to just one of the market segments
- ***Soft clustering*** is about grouping each observation in more than one category, providing a probability or likelihood for each cluster
  - For example, a recommender system based on customer reviews can associate a new user with more than one cluster of products

Machine Learning Techniques for Text

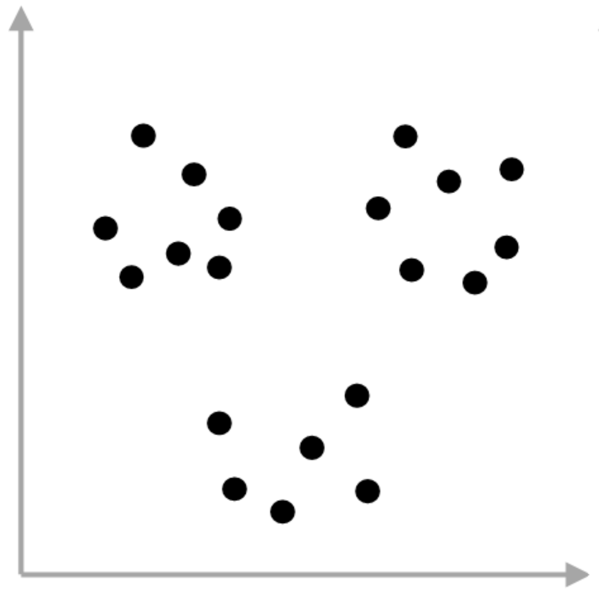
Section 2: Introducing hard clustering algorithms

K-means algorithm

packt
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

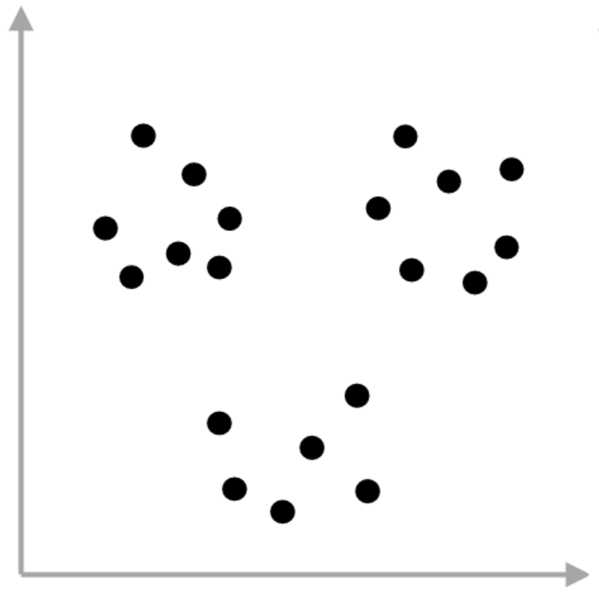
- The ***K-means*** algorithm is a predominant unsupervised learning algorithm for clustering data due to its simplicity and efficiency
- It aims to group similar items in the form of ***K*** clusters
- After selecting ***K*** random centroids, it repeatedly moves them around to group the most similar samples to the center of each cluster
- As a similarity measure, we can use metrics such as the ***Euclidean distance***, ***cosine similarity***, ***Pearson correlation coefficients***, and so forth
- Let's see an example …

K-means algorithm



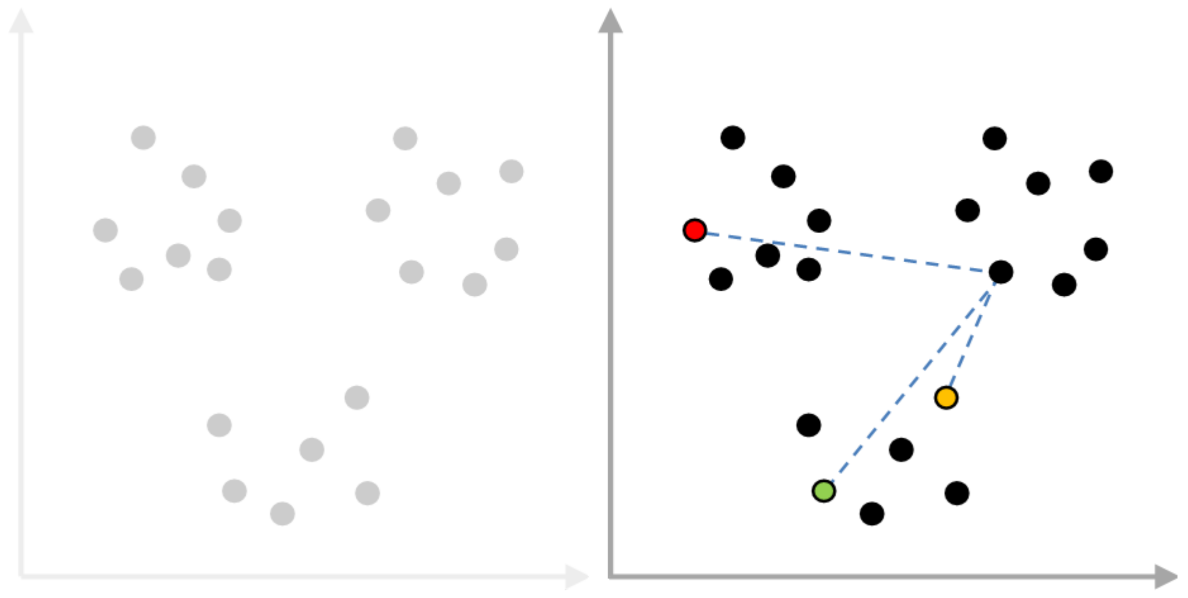
- It’s straightforward to identify that the data points can be grouped into three clusters
- Unfortunately, ***K-means*** does not possess any visual capacity to spot the clusters easily, and it needs to follow a series of steps to reach the same assumption

K-means algorithm



1. Select the number of clusters, ***K***, that we want to identify. Suppose that in this example, ***K=3***

K-means algorithm

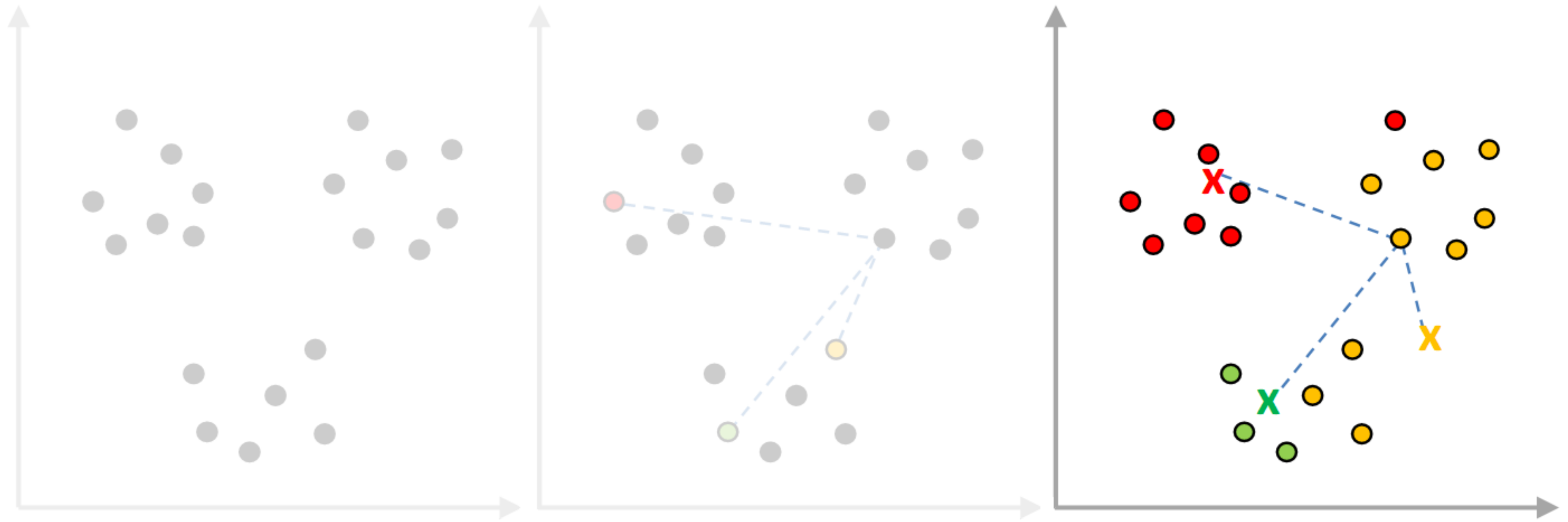


1. Select the number of clusters, K, that we want to identify. Suppose that in this example, ***K=3***

2. Randomly select three distinct data points as the cluster centroids and measure the distance from all points to the centroids

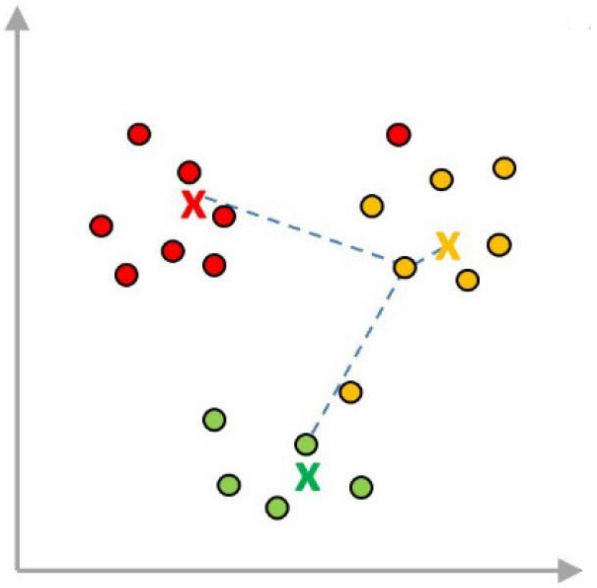
K-means algorithm

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



2. Randomly select three distinct data points as the cluster centroids and measure the distance from all points to the centroids
3. Assign each point to the closest cluster centroid and calculate the mean of the newly created cluster (depicted with the **X** symbol)

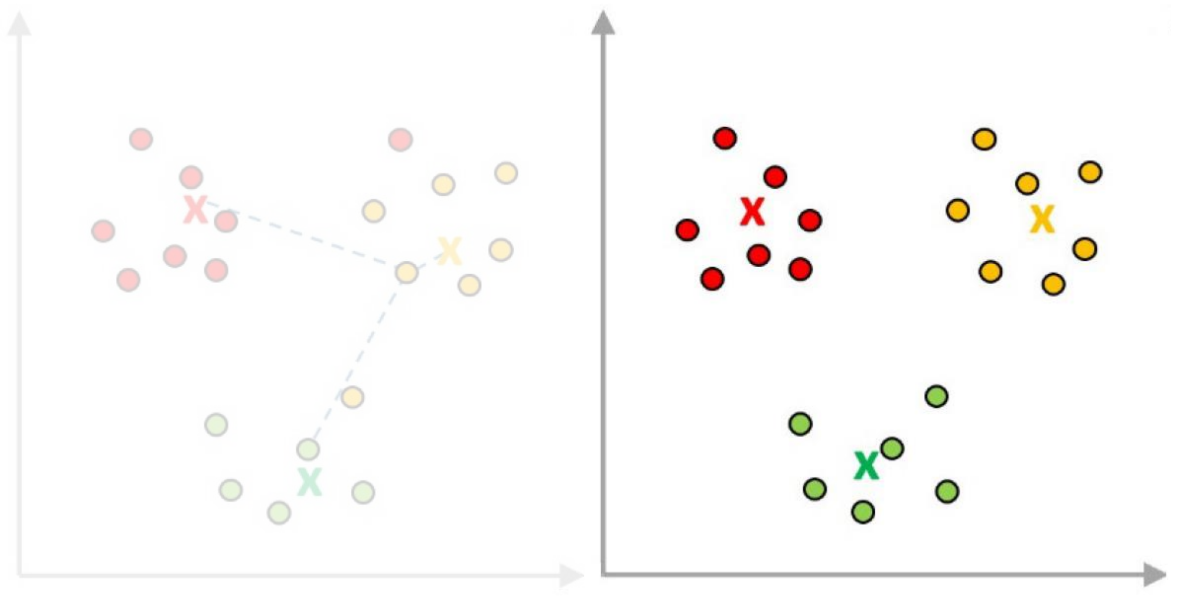
K-means algorithm



3. Assign each point to the closest cluster centroid and calculate the mean of the newly created cluster (depicted with the X symbol)
4. Repeat steps 2 and 3 using the mean values as centroids

K-means algorithm

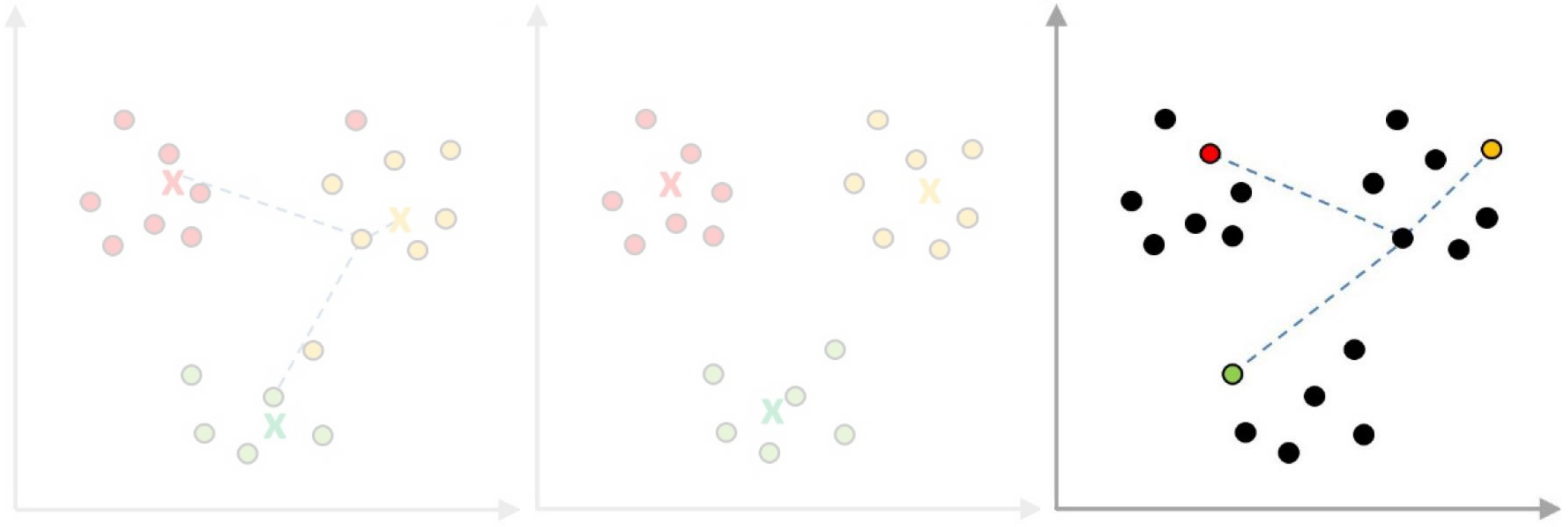
‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



4. Repeat steps 2 and 3 using the mean values as centroids
5. Stop the iterations when the clusters no longer change or the maximum number of iterations is reached

K-means algorithm

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



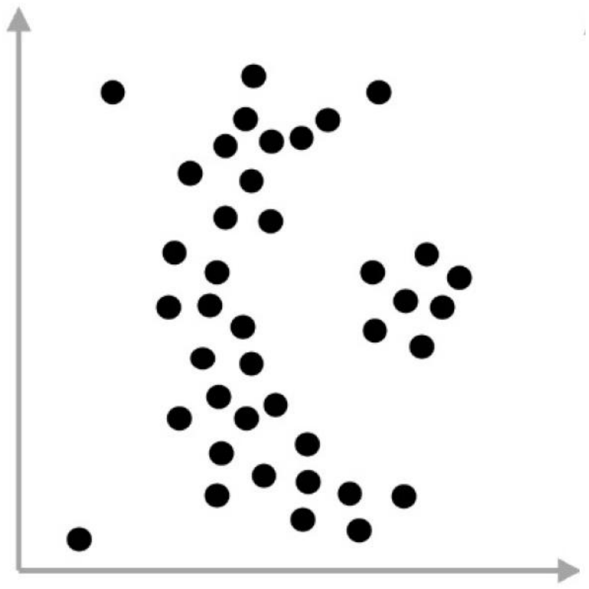
4. Repeat steps 2 and 3 using the mean values as centroids
5. Stop the iterations when the clusters no longer change or the maximum number of iterations is reached
6. Repeat from step 2 using a new set of random points

DBSCAN algorithm

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

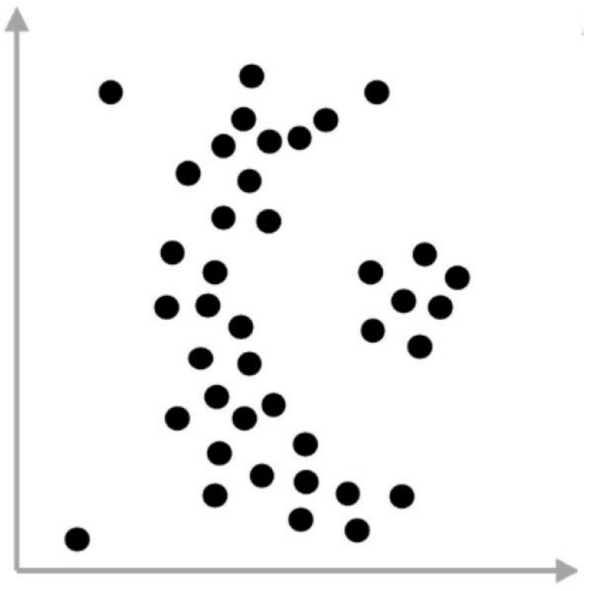
- The ***density-based spatial clustering of applications with noise*** (DBSCAN) algorithm clusters regions of high point density, separated from other clusters by low point density regions
- The algorithm takes each point in the dataset to identify the high-density regions and checks whether its neighborhood contains a minimum number of points
- Unlike K-means, DBSCAN does not require manually specifying the number of clusters; it is more immune to outliers and more appropriate when the clusters have complex shapes

DBSCAN algorithm



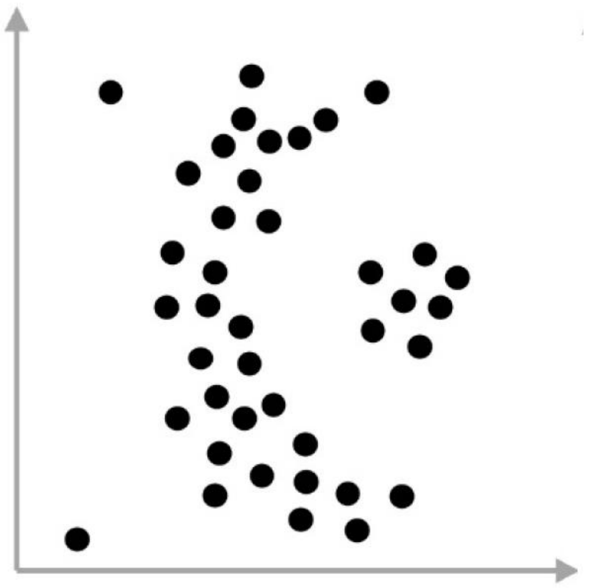
- Looking again at the specific plot, how many clusters can you identify?

DBSCAN algorithm



- Looking again at the specific plot, how many clusters can you identify?
  - Most probably two, one big and one smaller nested one

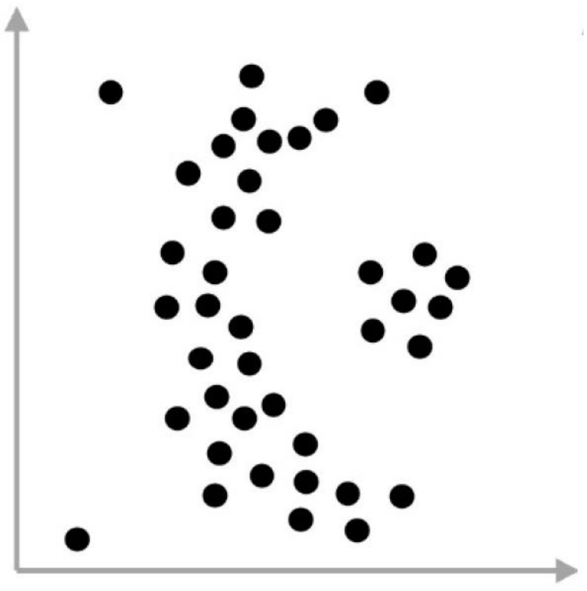
DBSCAN algorithm



- To employ the algorithm, we need to set two hyperparameters:
  - ***epsilon*** is the radius of the circle to be created around each point to check the region's density
  - ***minPts*** determines the minimum number of data points within the circle to label its center as a core point

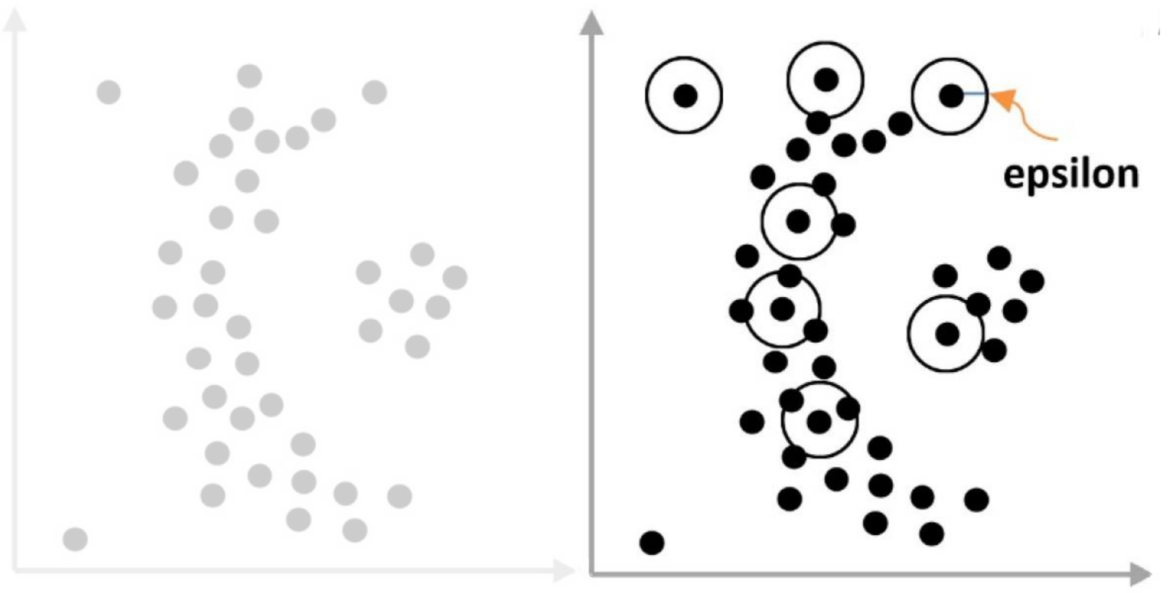
DBSCAN algorithm

Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



1. Select a value for epsilon and minPts. Suppose that in this example, ***epsilon=1*** and ***minPts=3***

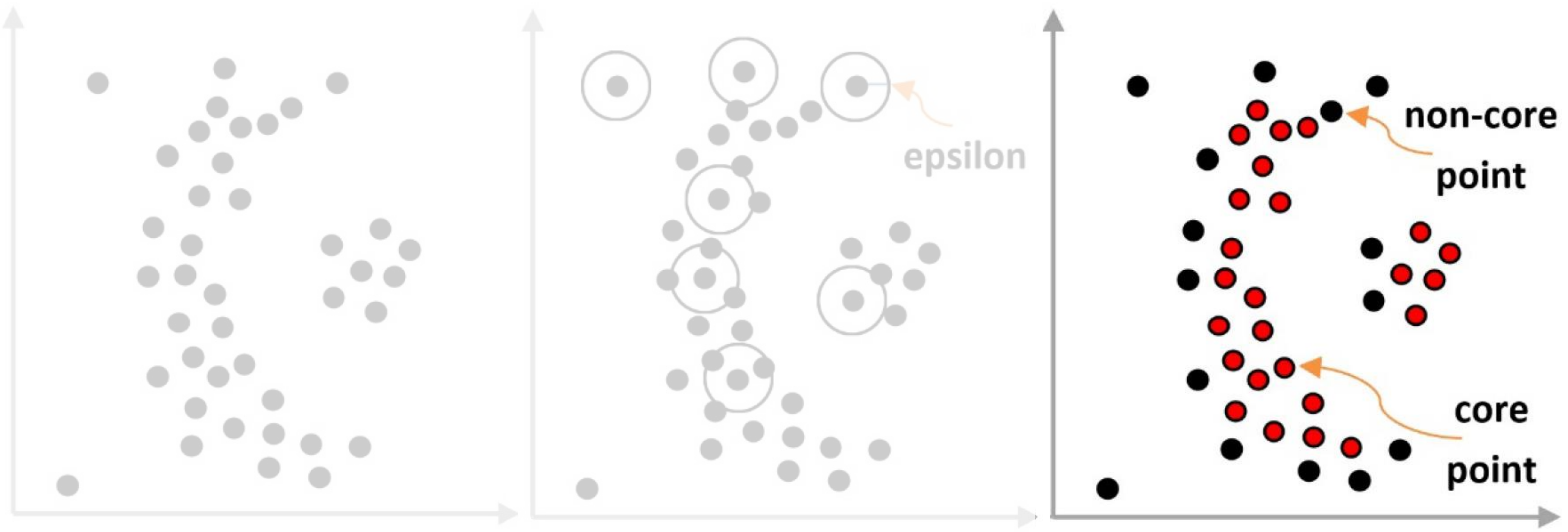
DBSCAN algorithm



1. Select a value for epsilon and minPts. Suppose that in this example, ***epsilon=1*** and ***minPts=3***
2. Choose a random point and check whether the minimum points criterion applies within the ***epsilon*** radius

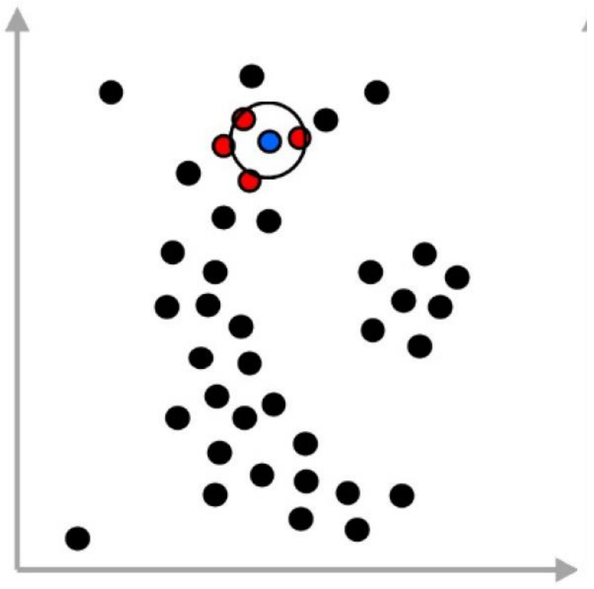
DBSCAN algorithm

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



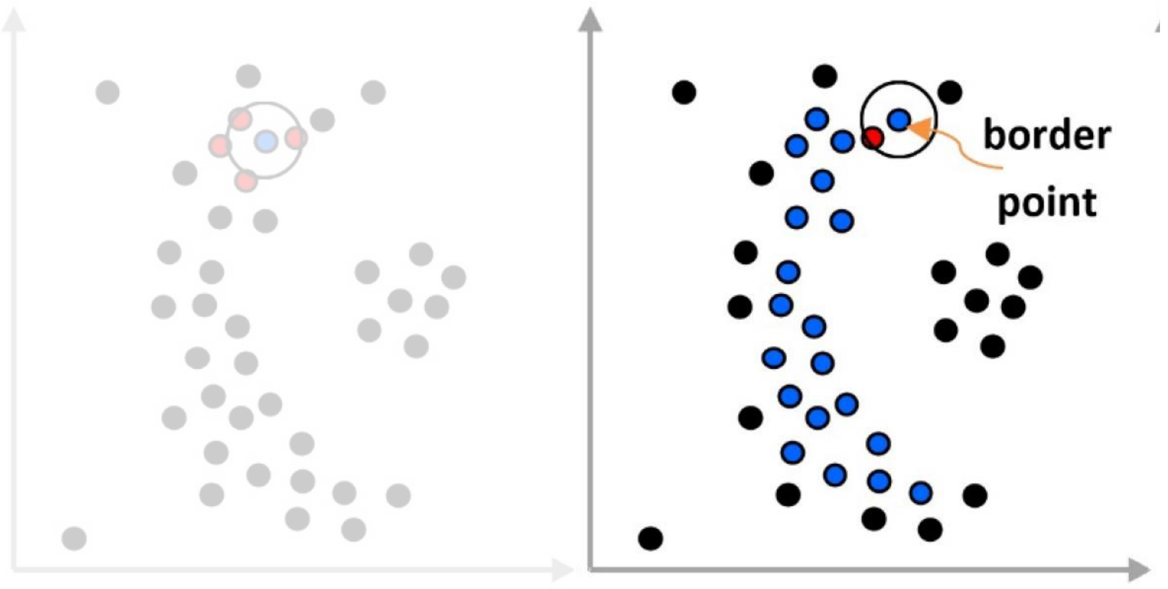
2. Choose a random point and check whether the minimum points criterion applies within the *epsilon* radius
3. If the answer to step 2 is positive, label the point as a ***core point***. Otherwise, it is a ***non-core*** one

DBSCAN algorithm



3. If the answer to step 2 is positive, label the point as a ***core point***. Otherwise, it is a ***non-core*** one
4. Choose a random core point and cluster together all core points inside the radius. Then, move to a core point close to the expanding cluster and repeat

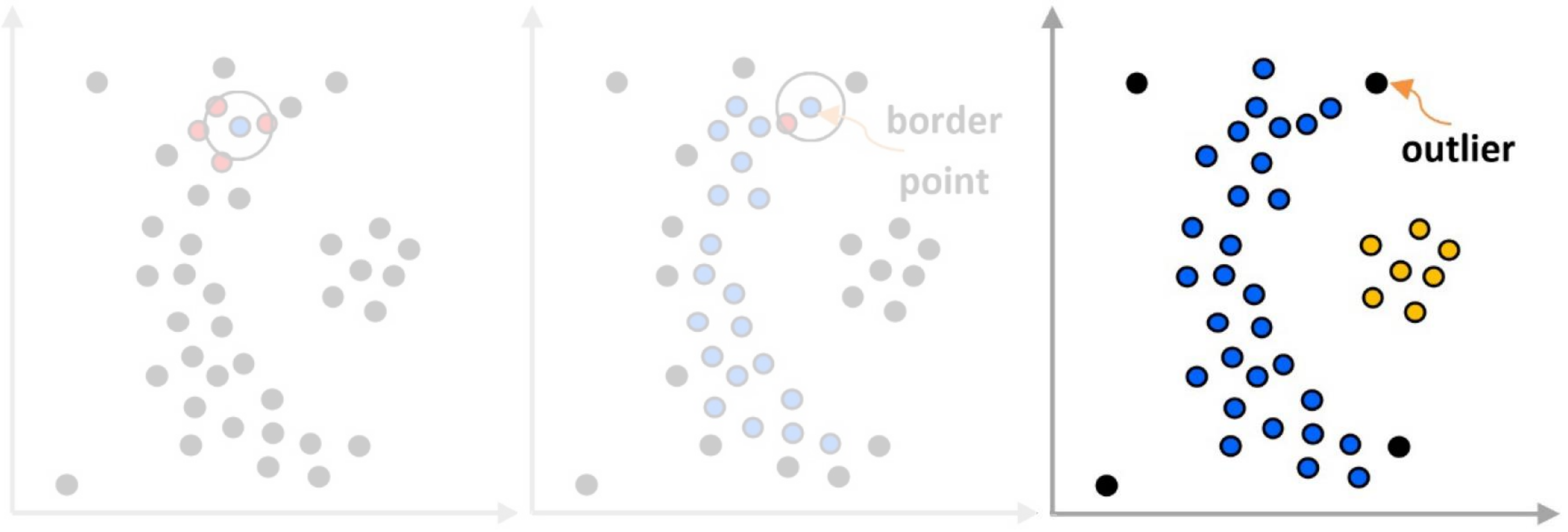
DBSCAN algorithm



4. Choose a random core point and cluster together all core points inside the radius. Then, move to a core point close to the expanding cluster and repeat

5. A non-core point is part of the same cluster only if it contains at least one core point. In this case, it is called a ***border point***

DBSCAN algorithm



5. A non-core point is part of the same cluster only if it contains at least one core point. In this case, it is called a ***border point***

6. Repeat the process for the next cluster starting from step 2. Points not part of any group are considered ***outliers*** (noise)

Hierarchical clustering algorithm

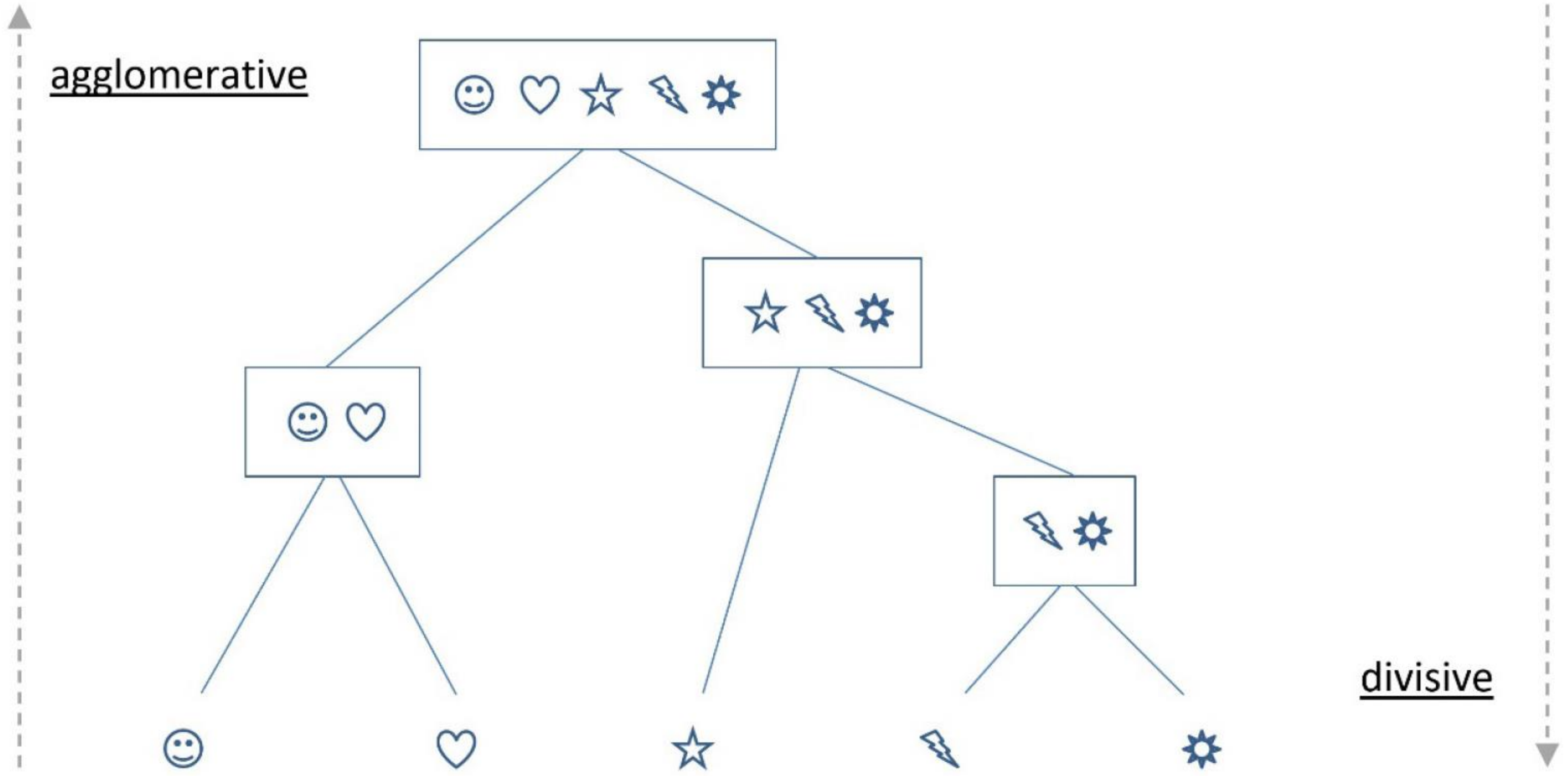
‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

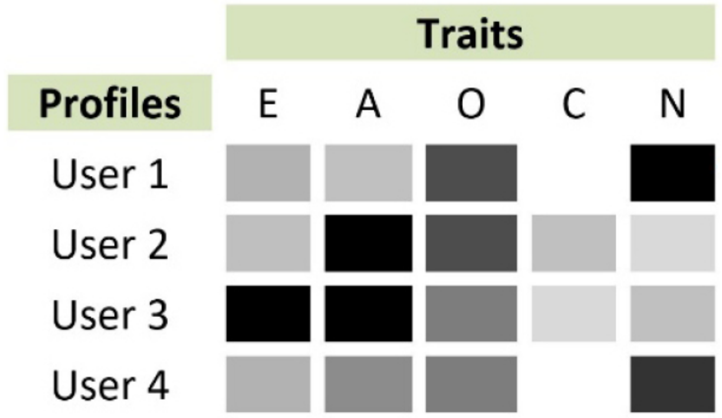
- ***Hierarchical clustering*** is another unsupervised machine learning algorithm that seeks to build a hierarchy of clusters
- It constructs a tree-like structure called a ***dendrogram*** that shows the hierarchical relationship between objects in a dataset
- Typically, there are two ways to construct the dendrogram: the ***agglomerative clustering*** approach or the ***divisive clustering*** one
- The first option is more common and follows a bottom-up approach by sequentially merging similar clusters
- In divisive clustering, we put all observations in one big cluster and then successively split the clusters

Hierarchical clustering algorithm



Hierarchical clustering algorithm

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



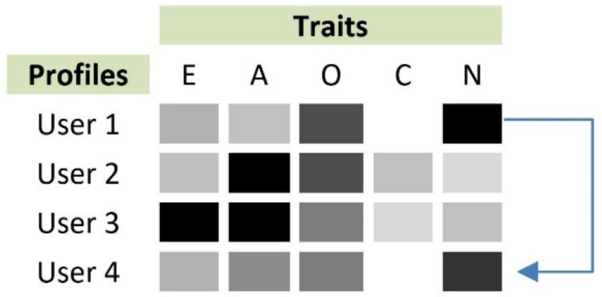
- We reuse the example where we visualize the personality traits of users with a personalized grayscale vector consisting of five elements (each for each trait)
- This time we aim to cluster four user profiles

Hierarchical clustering algorithm

‹packt›

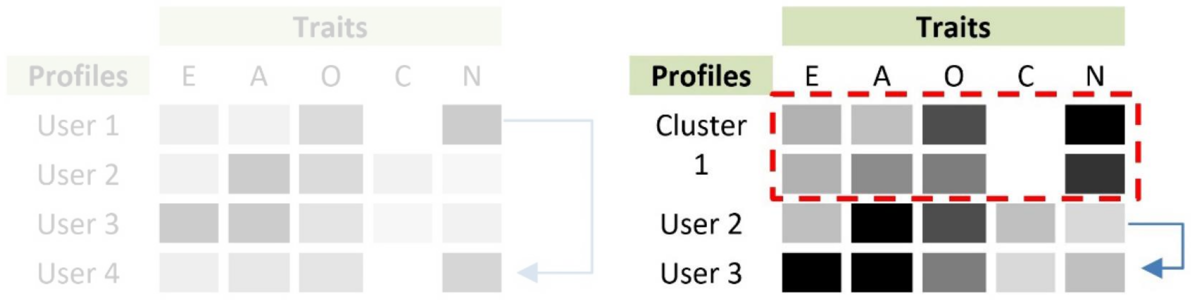
Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



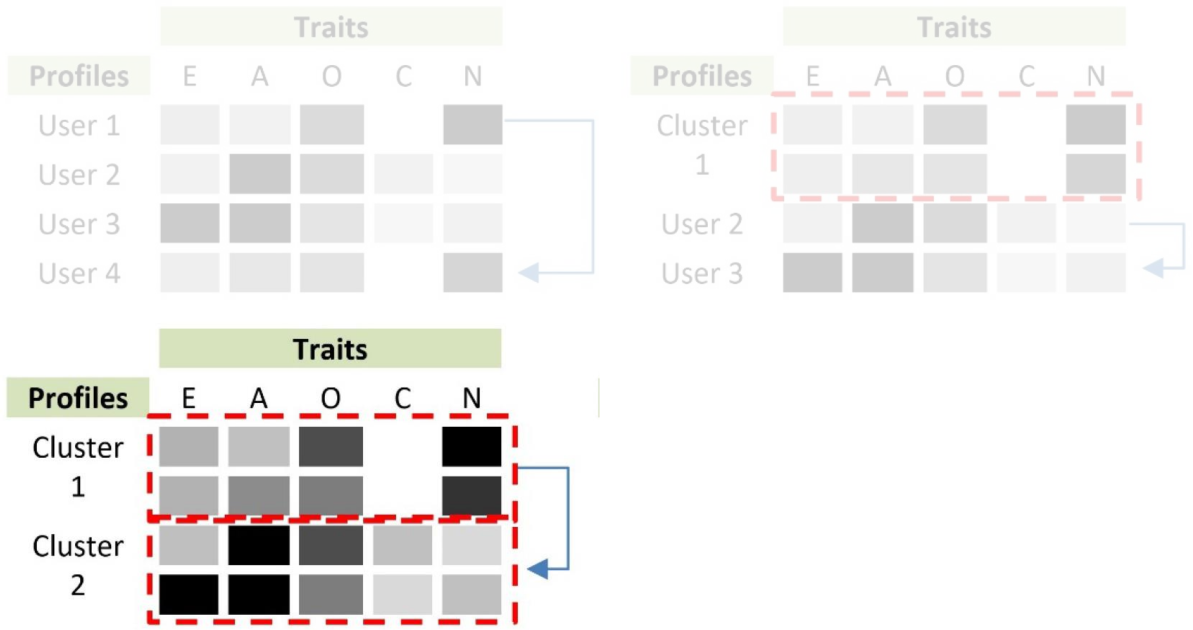
1. Compare each user vector with the others and find the most similar pair. In our example, ***User 1*** and ***User 4*** are more similar than any other combination (you can visually compare the grayscale values in their vectors)

Hierarchical clustering algorithm



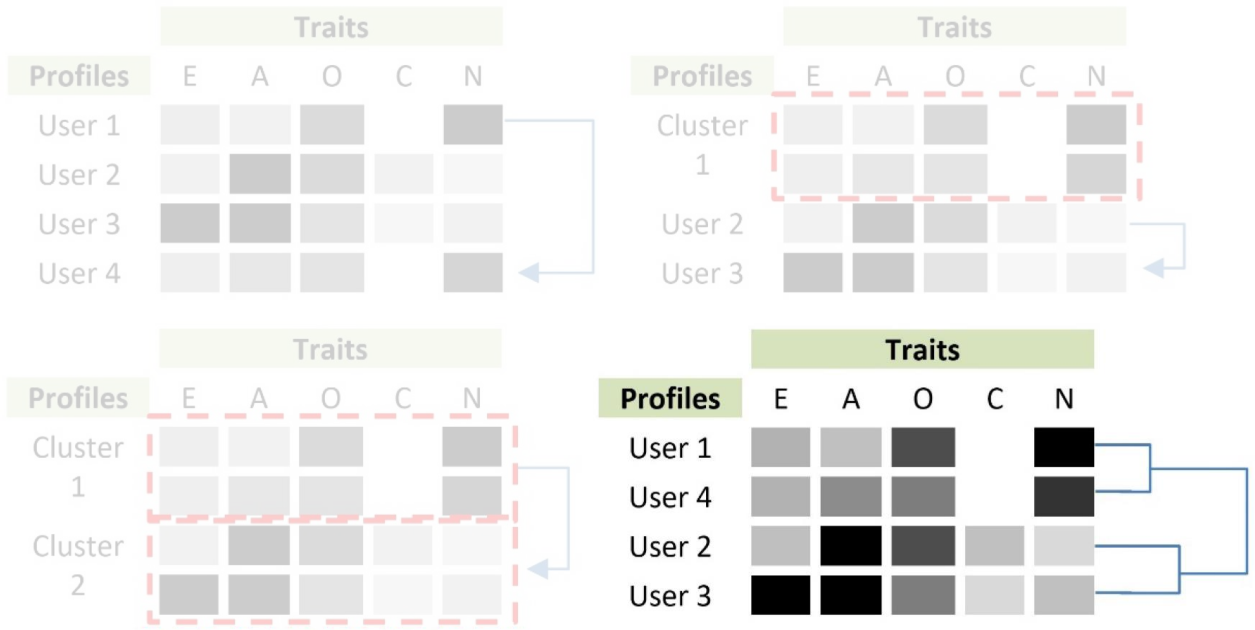
2. Merge the two profiles under the same cluster (***Cluster 1***). Repeat step 1 and use the profile vectors and the merged cluster for the comparisons. The pair of ***User 3*** and ***User 4*** is now the most similar combination

Hierarchical clustering algorithm



3. Merge the two profiles under the same cluster (***Cluster 2***). As there are only two remaining clusters, we stop the iterations

Hierarchical clustering algorithm



4. Merge ***Cluster 1*** and ***Cluster 2*** to build the dendrogram. Notice the height of the branches, which signifies the order that the clusters were formed. The cluster of ***User 1*** and ***User 2*** is created earlier than the one for ***User 3*** and ***User 4***. The smaller the height of a branch, the more similar the clusters underneath

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- Exploratory data analysis
- Hard clustering

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-10/text-clustering.ipynb

Machine Learning Techniques for Text

Section 3: Introducing topic modeling

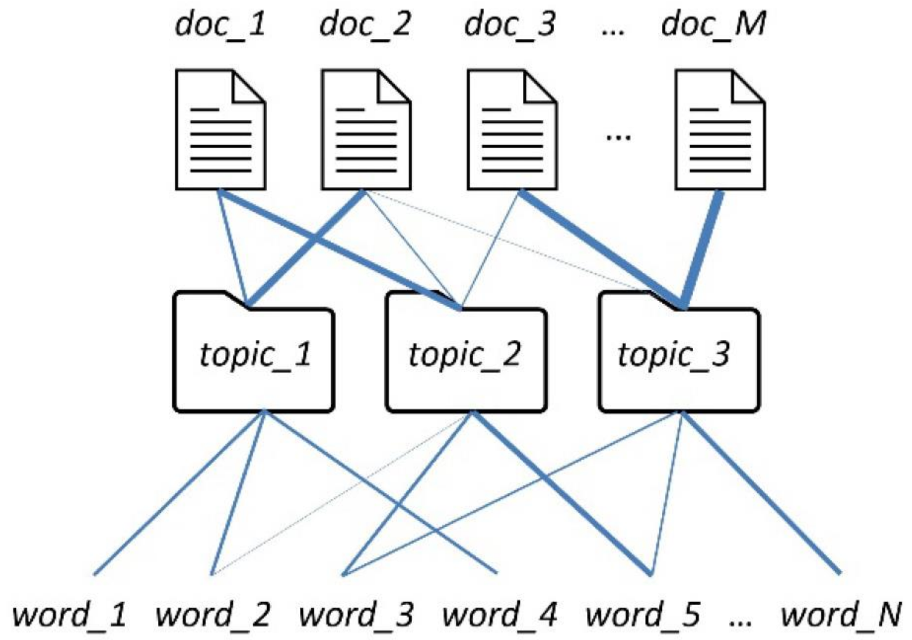
Topic modeling

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- ***Topic modeling*** refers to the task of identifying groups of items, in our case words, that best describes a collection of documents or sentences
- A popular topic modeling technique to extract the hidden topics from a given corpus is the ***latent dirichlet allocation*** (LDA)
- The topics emerge during the topic modeling; hence they are called latent
- Strictly speaking, LDA is not a clustering algorithm because it produces a distribution of groupings over the sentences being processed
- However, as a document can be a part of multiple topics, LDA resembles a soft clustering algorithm in which each data point belongs to more than one cluster

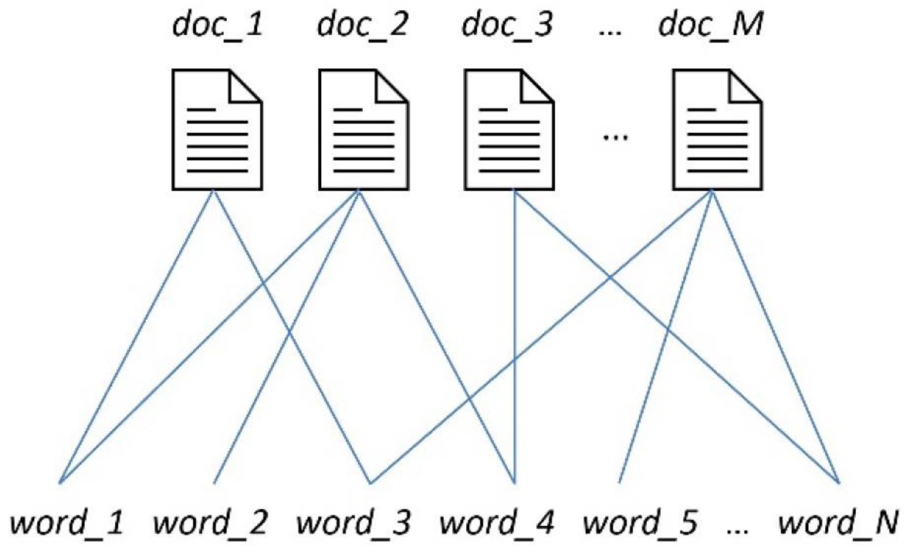
LDA algorithm

- The main idea behind the algorithm is that each document can be described as a ***distribution of topics*** and each topic as a ***distribution of words***
- LDA aims to find the topics of a document based on the words in it
- As in the case of hard clustering, we need the expert opinion of humans to evaluate the outcome of LDA



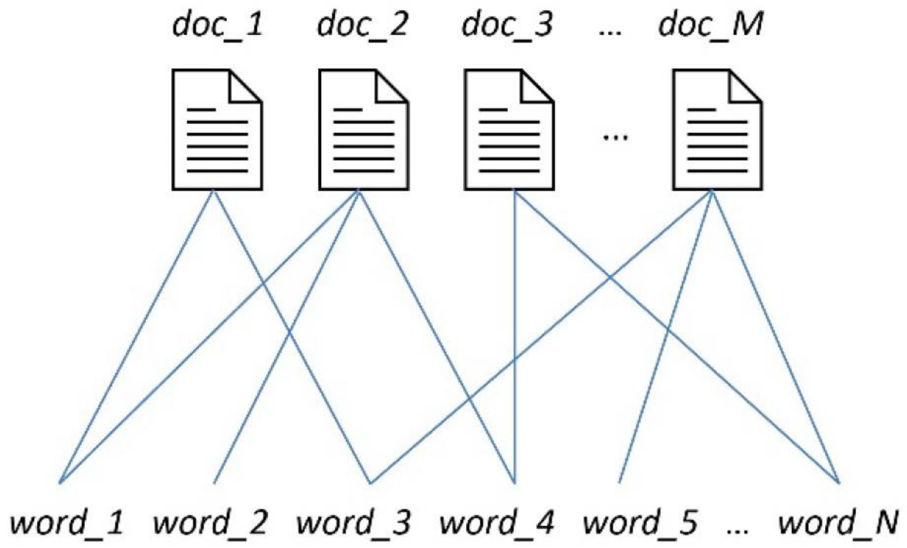
LDA algorithm

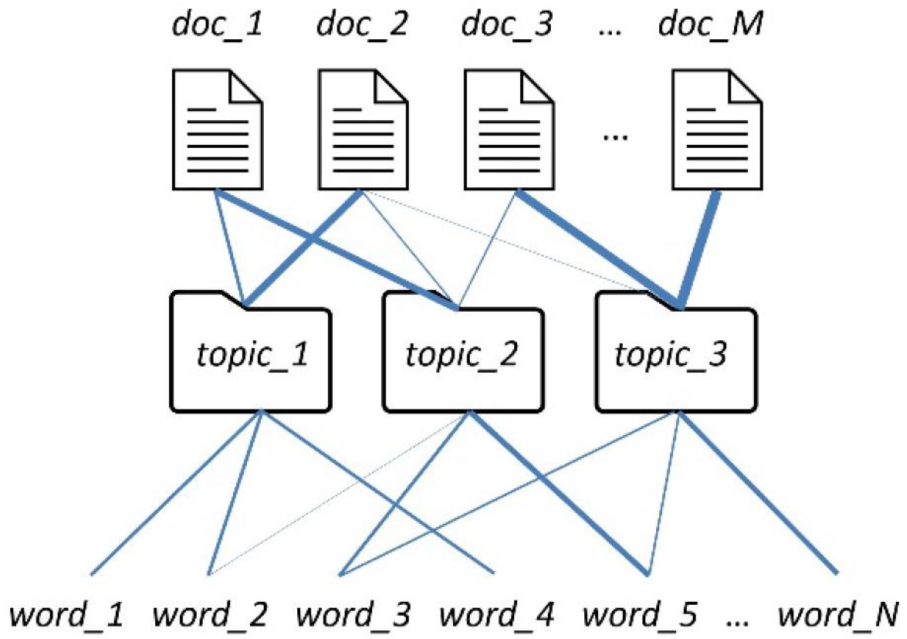
- Starting with ***M*** documents and a set of ***N*** words included in the documents, we can create the plot
- The plot shows the connections of each document to the words it contains
- Identifying the topics requires checking all the possible connections for all documents which is not practical



LDA algorithm

- We introduce a latent layer with three topics
- The number of connections is reduced as the documents connect only to topics and the latter to words
- LDA must find the weight of the connections, which are depicted schematically with the different thickness levels of each connection line

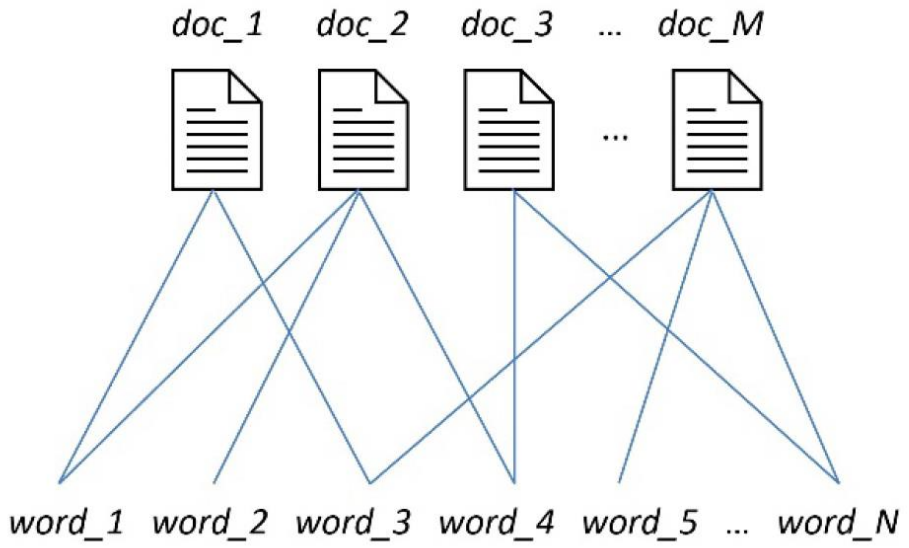


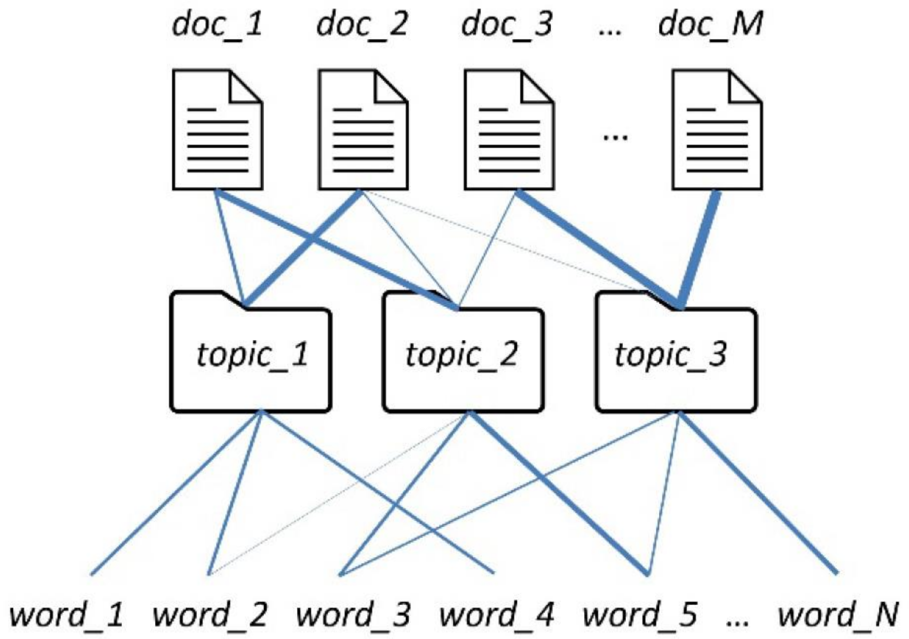


LDA algorithm

packt
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

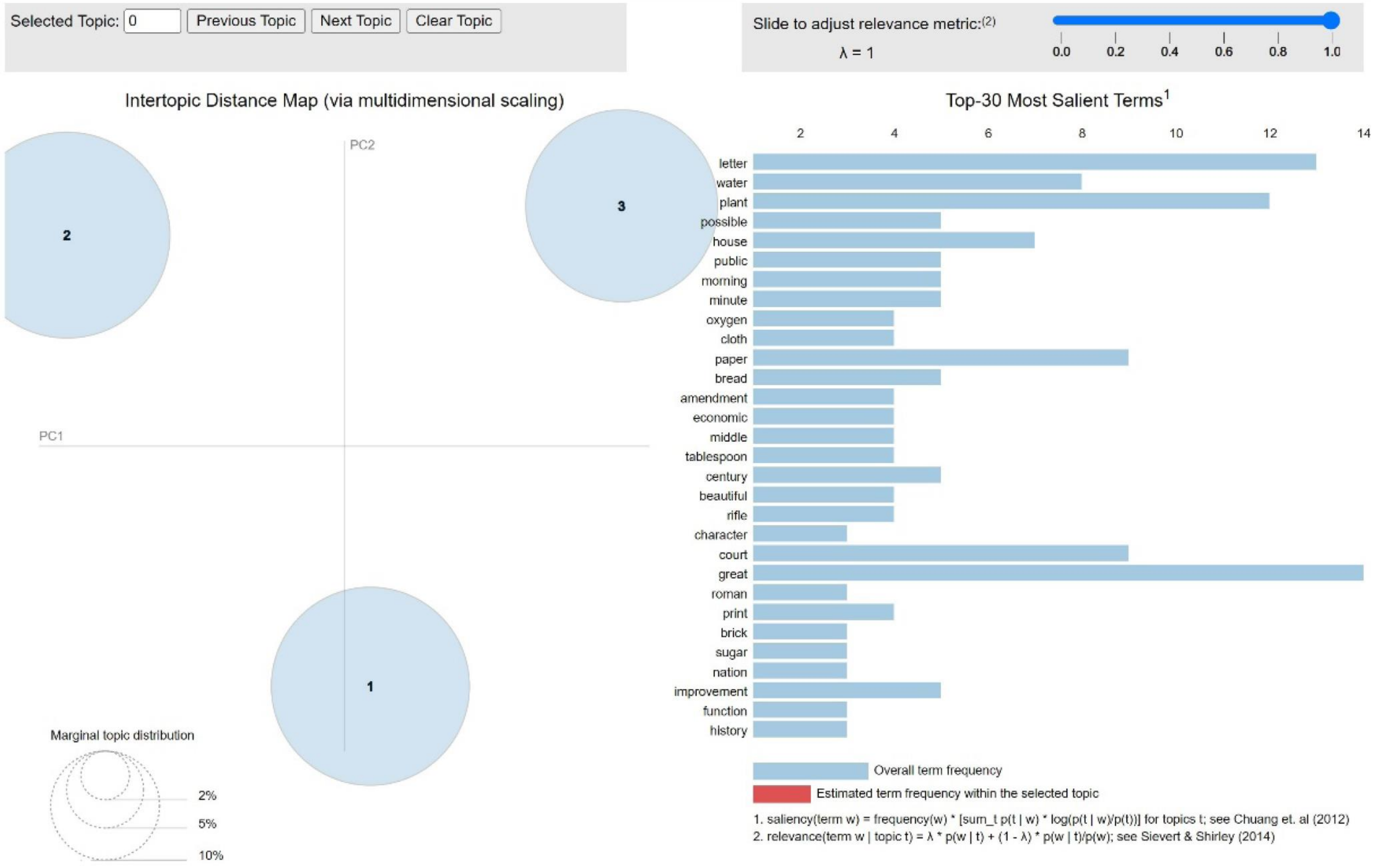
- For example, ***doc_2*** may consist of the following mix: 42% ***topic_1***, 36% ***topic_2***, and 22% ***topic_3***
- The most important hyperparameter for the algorithm is the number of clusters to aim for





LDA visualization

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



🟦	Overall term frequency
🟥	Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- Exploratory data analysis
- Hard clustering
- Topic modeling

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-10/topic-modeling.ipynb

Key takeaways

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Visualizations

- pyLDAvis plots

ML concepts

- Clustering

ML algorithms & models

- k-means
- DBSCAN
- Hierarchical clustering
- Latent Dirichlet Allocation

Performance metrics

- Word Error Rate
- Silhouette coefficient

Tools

- Google Speech API

Machine Learning Techniques for Text

Questions?