Text Mining - Clustering

MK Data Mining 2

Dr. Maryamah, S.Kom.

Program Studi S1 Teknologi Sains Data

Fakultas Teknologi Maju dan Multidisiplin

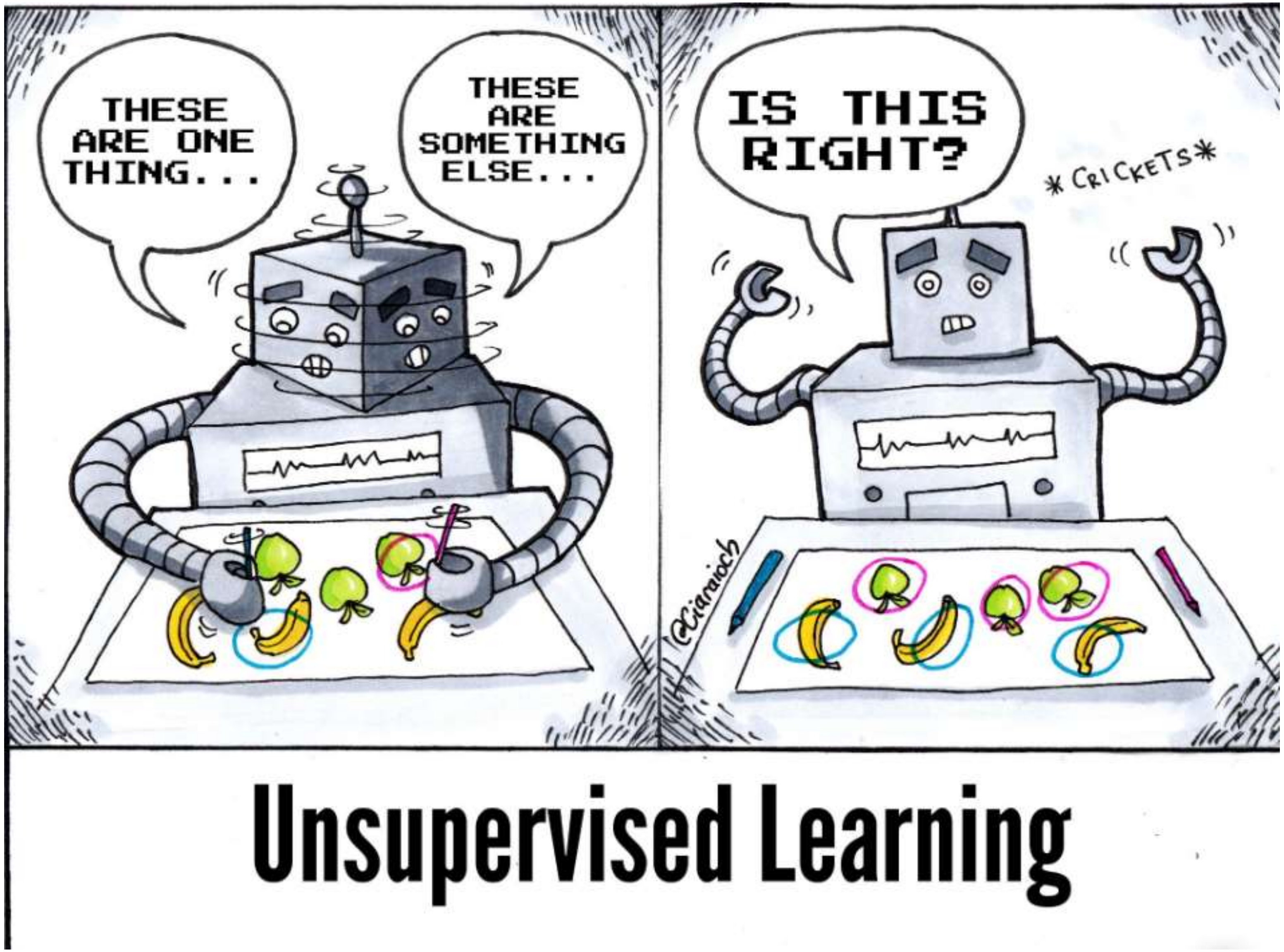
Universitas Airlangga Indonesia

Outline

- Review Unsupervised Learning & its method
- Text as a data
- Text Representation / Feature Extraction
- Descriptive Text Mining
- Term Frequency
- Text Clustering

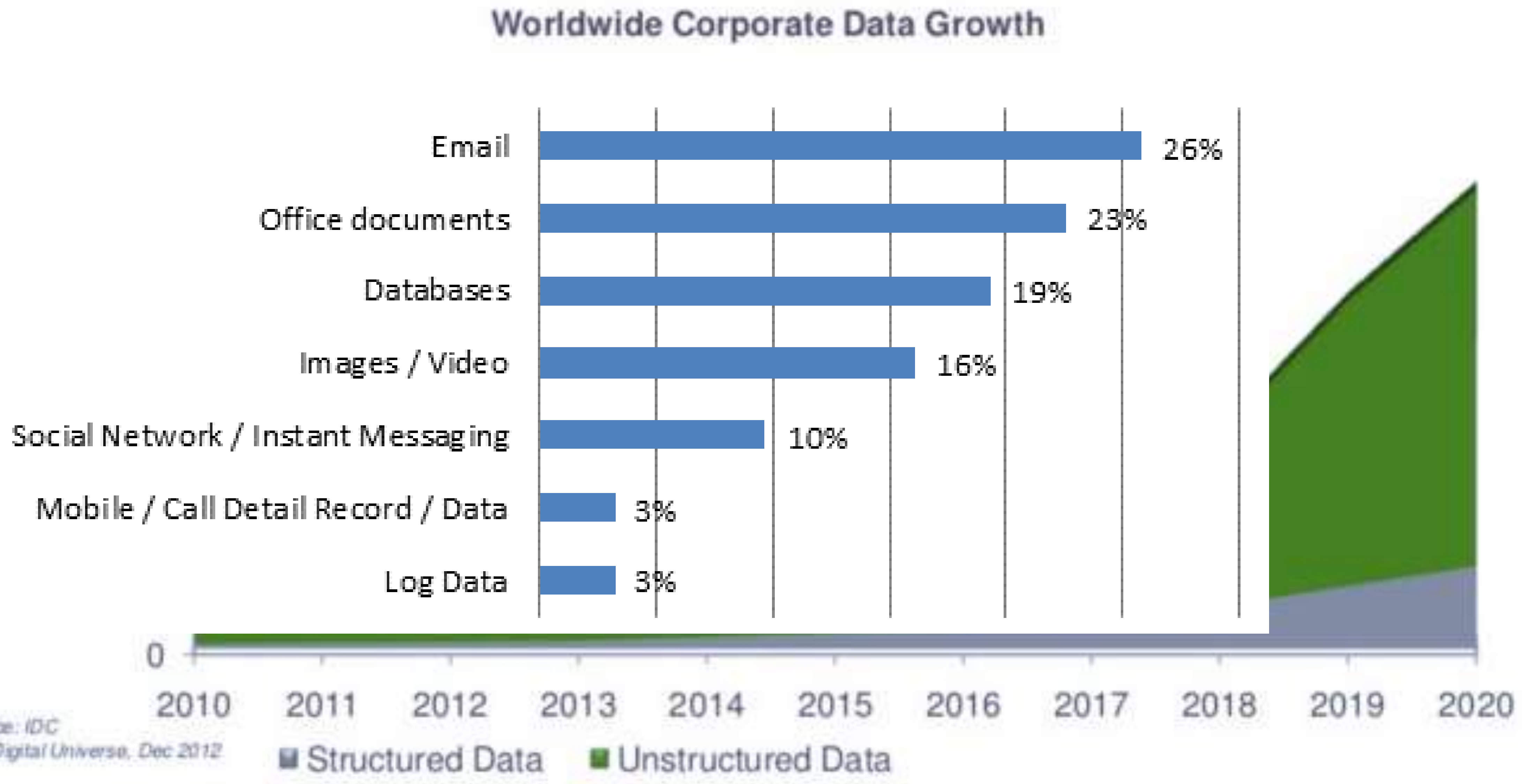
What is Unsupervised Learning?

Unsupervised Methods?



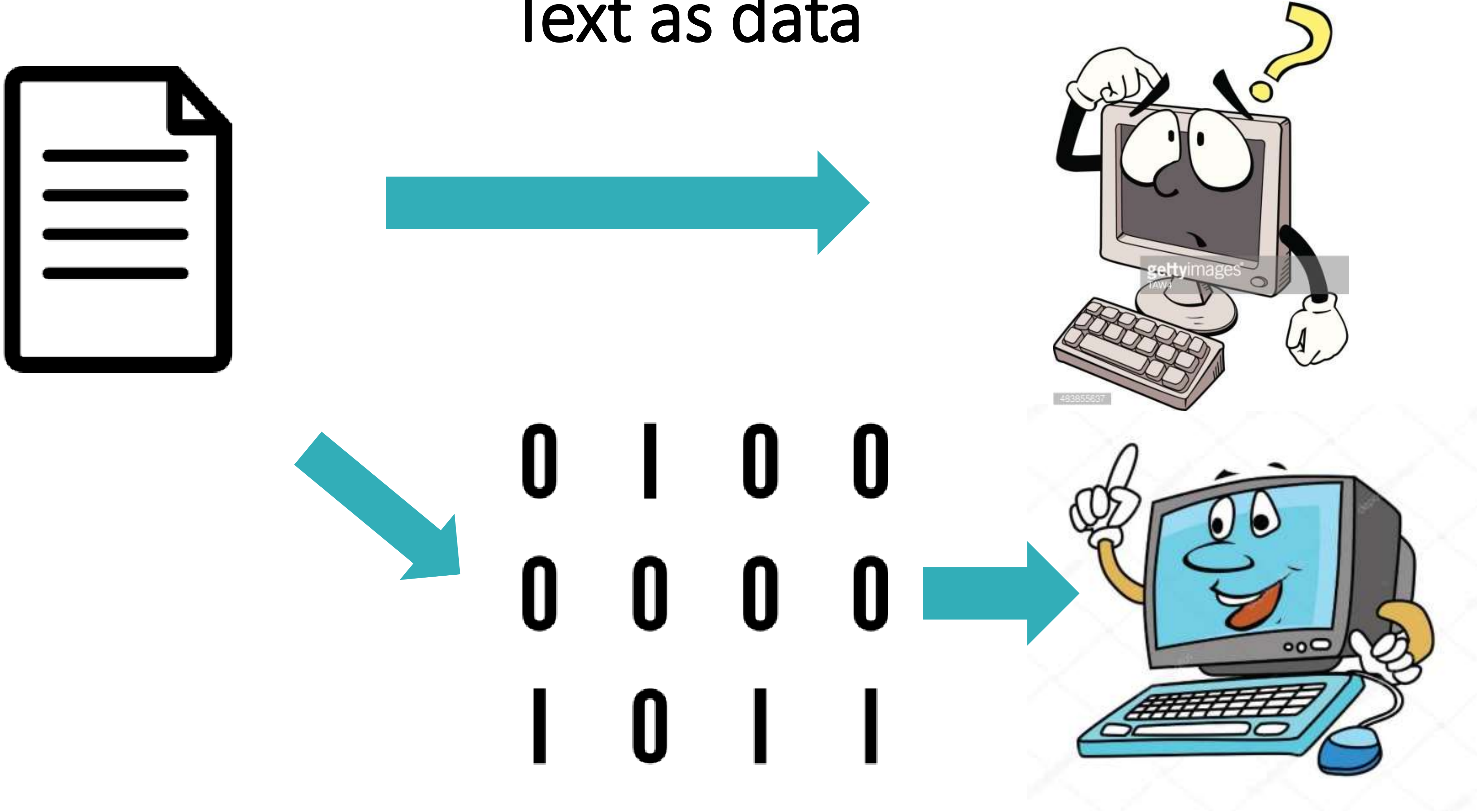
MASSIVE GROWTH IN UNSTRUCTURED CONTENT

RECOMMIND



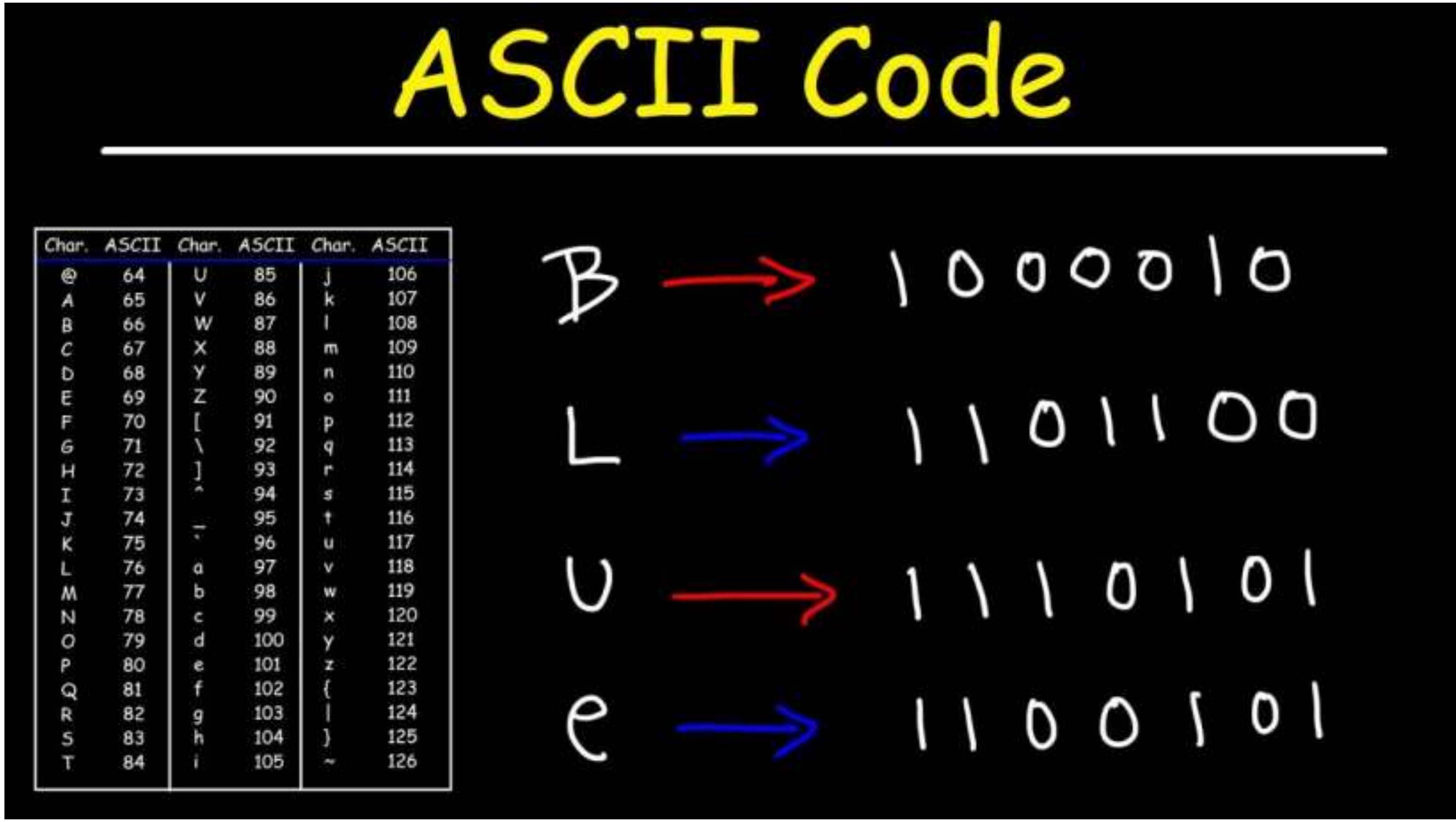
Source: IDC
The Digital Universe, Dec 2012

Text as data



Text as a data

- Books and writings has been around for a thousand (or more) years
- Every knowledge known to human is usually recorded in writings
- These texts become data (because it is from observations)
- In computers, text is encoded using numbers (ASCII, Unicode, etc.)



Char.	ASCII	Char.	ASCII	Char.	ASCII
@	64	U	85	j	106
A	65	V	86	k	107
B	66	W	87	l	108
C	67	X	88	m	109
D	68	Y	89	n	110
E	69	Z	90	o	111
F	70	[	91	p	112
G	71	\	92	q	113
H	72	]	93	r	114
I	73	^	94	s	115
J	74	_	95	t	116
K	75	`	96	u	117
L	76	a	97	v	118
M	77	b	98	w	119
N	78	c	99	x	120
O	79	d	100	y	121
P	80	e	101	z	122
Q	81	f	102	{	123
R	82	g	103	|	124
S	83	h	104	}	125
T	84	i	105	~	126

Text Preprocessing

- Cleaning
- Tokenization
- Filtering
- Stemming / Lemmatizing

Text Cleaning

- “Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @AmazonHelp”

- Computers need to clean the text, and separate it into meaningful feature
  - Remove symbols (irrelevant to meaning)
  - Remove links
  - Remove mentions
  - Normalize text (ASAP)

Text cleaning

- Computers encode “Hey” and “hey” differently. Because?
- Computers will also process - : , “ + etc. It is not usable for analysis
- How to make all letters in lower case?
- How to remove symbols / special characters / numbers?

Text Cleaning

- To clean text from symbol, Regular Expression can be used
- Regular Expression search for patterns, and can be used to remove unwanted patterns (and symbols)

```
text = re.sub(r"[-()\"#/@;:<>{}-=~|.?,]","",text)
```

- In Python, string method lower() can be used to decapitalized all the letters

Text Tokenization

- Simply means splitting
- Paragraphs become list of sentences, Sentences become list of words
- Words can be processed further individually
- Sometimes named Bag of Words
- the bag of words approach to text mining is the most common method for performing computations on string data
- the data are broken down into tokens.
- A token can be thought of as a unit of a certain or uncertain length, depending on the context in which string data is tokenized.
- Tokens can be as small as a single character or as large as an entire text document.

Tokenization with NLTK

1. Sentence tokenization : split a paragraph into **list of sentences** using **sent_tokenize()** method
2. Word tokenization : split a sentence into **list of words** using **word_tokenize()** method

- from nltk.tokenize import sent_tokenize, word_tokenize

```
#STEP 1 :TOKENIZATION : Breaking complex data into simple units
#Sentence Tokenizer
sentences=sent_tokenize(data)
print(sentences)
#Word Tokenizer
words=word_tokenize(data)
print(words)
```

Some important thing in tokenization

- Sometimes single word cannot represent meanings
- “Kemarin aku beli sepeda pinarello baru”
- N-gram
  - Unigram = [“kemarin”,”aku”,”beli”,”sepeda”,”pinarello”, ”baru”]
  - Bigram = [“kemarin aku”,”aku beli”,”beli sepeda”,”sepeda pinarello”,”pinarello baru”]
  - Trigram = [“kemarin aku beli”,”beli sepeda pinarello”,”sepeda pinarello baru”]

```
from nltk.util import ngrams

n = 3
sentence = 'Whoever is happy will make others happy too'
unigrams = ngrams(sentence.split(), n)

for item in unigrams:
    print(item)
```

```
from nltk.util import ngrams

n = 2
sentence = 'The purpose of our life is to happy'
unigrams = ngrams(sentence.split(), n)

for item in unigrams:
    print(item)
```

Filtering

- Sometimes, we don’t want specific word to appear in our data / analysis
- Stop words are basically a set of commonly used words in any language, and usually useless in analysis
- E.g. the, and, for, a, an, another, in, under, ada, adanya, adalah, di, ke, dari,

```
from nltk.corpus import stopwords

# tokenize text
freq_tokens

# get Indonesian stopword
list_stopwords = set(stopwords.words('indonesian'))

#remove stopword pada list token
tokens_without_stopword = [word for word in freq_tokens if not word in list_stopwords]


print(tokens_without_stopword)
```

Stemming / Lemmatizing

- Returning words to its root form
- E.g. Cleaning -> clean, swollen -> swell, menangis -> tangis
- Can be very hard, since words can transform

In [138]:

```python
# import Sastrawi package
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# token without stopword
list_tokens = tokens_without_stopword

# stem
output   = [(token + " : " + stemmer.stem(token)) for token in list_tokens]

output
```

Out[138]:

```
['positif : positif',
 'virus : virus',
 'corona : corona',
 'april : april',
 'orang : orang',
 'pasien : pasien',
 'sembuh : sembuh',
 'ri : ri',
 'meninggal : tinggal']
```

```python
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

plurals = ['caresses', 'flies', 'dies', 'mules', 'denied',
           'died', 'agreed', 'owned', 'humbled', 'sized',
           'meeting', 'stating', 'siezing', 'itemization',
           'sensational', 'traditional', 'reference', 'colonizer',
           'plotted']

singles = [(plural + " : " + stemmer.stem(plural)) for plural in plurals]
```

Discussion

Apakah Perlu ?

Stemming

Pos Tagging

POS tagging merupakan proses pemberian tanda berupa kelas kata pada setiap kata yang terdapat di dalam corpus.

Part of Speech	Definition	Some Examples	
Nouns	people, places, things (and animals)	dog, cat, garden, work, music, town, Manila, teacher, Bob	The sun shines. Anna goes to school.
Pronouns	replace nouns	he, I, its, me, my, she, that, this, those, us, who, whom, you,	John is hungry. He wants to eat.
Verbs	show action or being	run, go, have, invite, laughed, listen, playing, singing, walk	The dog and cat are running.
Adjectives	describe nouns	angry, brave, healthy, little, old, red, smart, two, some, good, big, interesting	Brown dog, Fat cat, Big garden
Adverbs	describe verbs, adjectives or other adverbs	badly, fully, hardly, nearly, never, quickly, silently, well, very, really, almost	Runs quickly, Eats very slowly
Articles	signal that a noun is going to follow	the, a, an	The dog, The cat
Prepositions	show relationship between words in a sentence	above, before, except, from, in, near, of, since, between, upon, with, to, at, after, on	I am going to my garden (Prep) (Object of the P)
Conjunctions	connect words, phrases, clauses or sentences	and, or, but, so, after, before, unless, either, neither, because, since,	I was tired so I went to sleep.
Interjections	exclamations that express strong feelings	aha!, gosh!, great!, hey!, hi!, hooray!, oh!, oops!, phew!, oh!, ouch!, hi!, well	Oops! I spilled the milk.

Descriptive Text Analysis

- After preprocessing the text data, you have a corpus of words (in whatever document you have)
- You can start counting the frequency of words (most common words), counting the basic dominant topics (in sentiment analysis), etc
- You can also visualize the wordcloud or top 10 used words
- Word Cloud can be used in the analysis of words present in the corpus. Suppose you have a 2000–3000 words and we want to analyse which is the most common words or repeated words in the document.

Example: sentiment analysis of citayam

Positive Tweet Documents

Negative Tweet Documents

Wordcloud

```python
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

comment_words = ''

for val in twitter['id_cleaned']:  #for each tweet in corpus
    val = str(val) #typecast to str

    comment_words += " ".join(val)+" " #join as one

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Term Frequency

Term Frequency

- Proses penghitungan bobot tiap term yang dicari pada setiap dokumen sehingga dapat diketahui ketersediaan dan kemiripan suatu term di dalam dokumen
- Term Frequency:
  - Bag of Words
  - Term Weighting

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Introduction

Term weighting (pembobotan kata) merupakan proses **penghitungan bobot** tiap **term** pada setiap **dokumen**.

Dokumen?

- Email (1 halaman)
- Buku (1 halaman)
- Text pidato (1 paragraf)

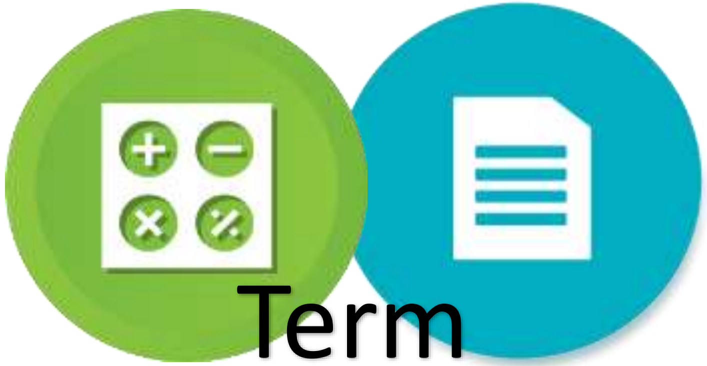
d1	d2
Ukuran nasi sangatlah kecil, namun saya selalu makan nasi	Nasi berasal dari beras yang ditanam di sawah. Sawah berukuran kecil hanya bisa ditanami sedikit beras

Introduction

- Setelah melalui Preprocessing (lower case, stopword, stemming):

d1	d2
ukur, nasi, makan	nasi, beras, tanam, sawah

- Tujuan term weighting: mengetahui ketersediaan dan kemiripan term di dalam dokumen.



$\rightarrow$ $w_{t,d}$

Metode Term Weighting

- Binary term weighting
- Term Frequency (TF)
- Inverse Document Frequency (IDF)
- TF-IDF

Binary Term Weighting

- Bobot term adalah **1** (jika term **muncul** pada dokumen) atau **0** (jika term **tidak muncul** di dokumen)

$$w_{t,d} = \begin{cases} 1, jika\ d\ mengandung\ t \\ 0, jika\ d\ tidak\ mengandung\ t \end{cases}$$

Binary Term Weighting

Contoh

d1 : Kucing makan ikan

d2 : Manusia makan nasi dan makan ikan

Kata	d1	d2
Kucing	1	0
Makan	1	1
Ikan	1	1
Manusia	0	1
Nasi	0	1

Kelebihan :

- Mudah diimplementasikan

Kekurangan :

- Tidak dapat membedakan term yang sering muncul ataupun term yang hanya sekali m uncul

Term Frequency (TF)

Term frequency menghitung bobot term berdasarkan jumlah kemunculan term tersebut pada dokumen

$$tf_{ij} = n_{ij}$$

$$tf_{ij} = \frac{n_{ij}}{\sum_{k=1}^{T} n_{kj}}$$

$n_{i,j}$ = jumlah kemunculan (frekuensi) term *i* pada dokumen *j*

k = total terms yang ada pada dokumen *j*

Term Frequency (TF)

Variants of TF weight

weighting scheme	TF weight
binary	$0, 1$
raw frequency	$f_{t,d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Maximum Frequency of any term in the document

Variants of IDF weight

weighting scheme	IDF weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t}$
inverse document frequency smooth	$\log\left(\frac{N}{1 + n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

Inverse Document Frequency (IDF)

- Menghitung **bobot DF** terlebih dahulu berdasarkan **jumlah dokumen** yang **mengandung kata** tersebut**.**
- **Bobot IDF** semakin **besar** jika kata lebih **sedikit muncul** pada dokumen

$$idf_t = \log_{10}(\frac{N}{df_t})$$

N = jumlah dokumen

$df_t =$ jumlah dokumen yang mengandung term t

TF-IDF

- Bobot term dihitung berdasarkan banyaknya term muncul dari semua dokumen.
- Term yang **sering muncul** pada seluruh dokumen akan mendapatkan **nilai tinggi.**
- Bobot **term tertinggi** menunjukkan term yang **penting** dalam dokumen berdasarkan keseluruhan dokumen

$$w_{t,d} = tf_{t,d} \times (\log_{10}(\frac{N}{df_t}) + 1)$$

TF-IDF

Contoh

d1 : Kucing makan ikan

d2 : Manusia makan nasi dan makan ikan

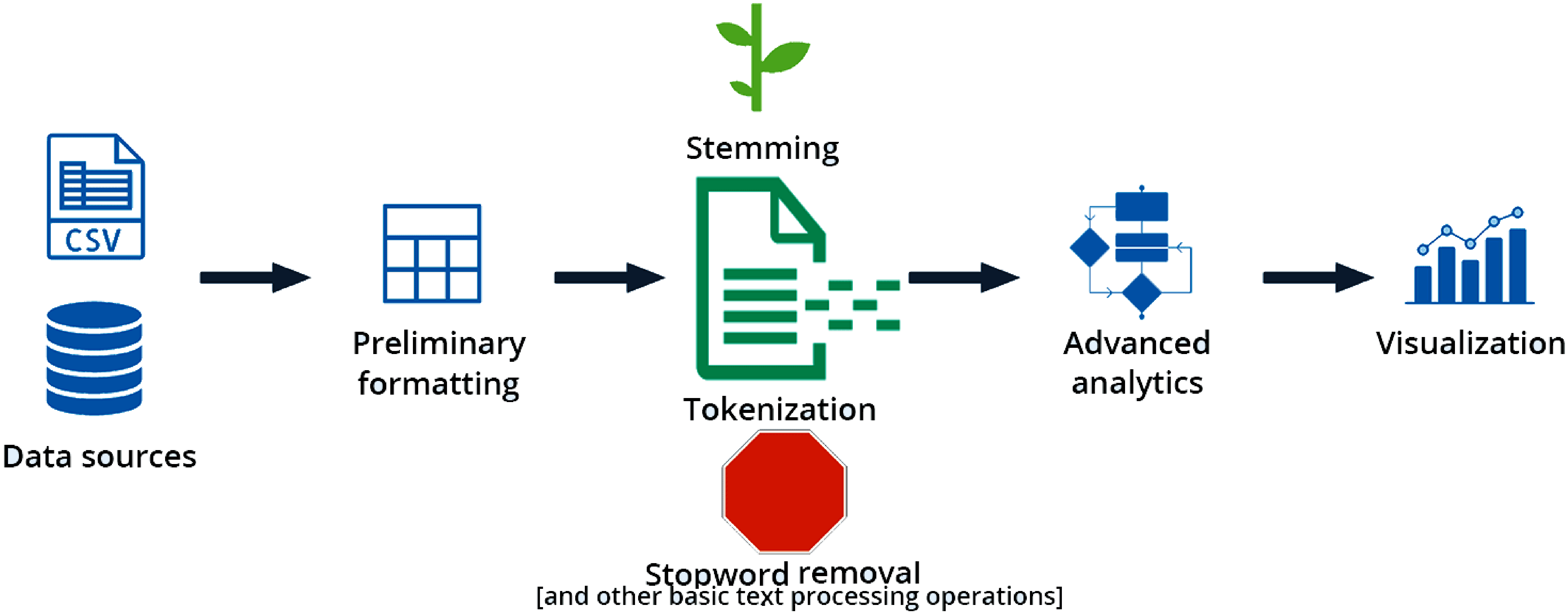
Kata	TF		DF	IDF	TF-IDF	
	d1	d2			d1	d2
Kucing	1	0	1	0,3	**1,3**	0
Makan	1	2	2	0	1,0	**2,0**
Ikan	1	1	2	0	1,0	1,0
Manusia	0	1	1	0,3	0	1,3
Nasi	0	1	1	0,3	0	1,3

TF-IDF

- Kelebihan :
  - Mempertimbangkan frekuensi kata yang dibandingkan dengan seluruh kata pada seluruh dokumen
- Kekurangan :
  - Tidak memperhatikan **makna** kata.

Word embedding

Text Clustering



CSV
Data sources
Preliminary formatting
Stemming
Tokenization
Stopword removal
[and other basic text processing operations]
Advanced analytics
Visualization

Clustering

Hard clustering

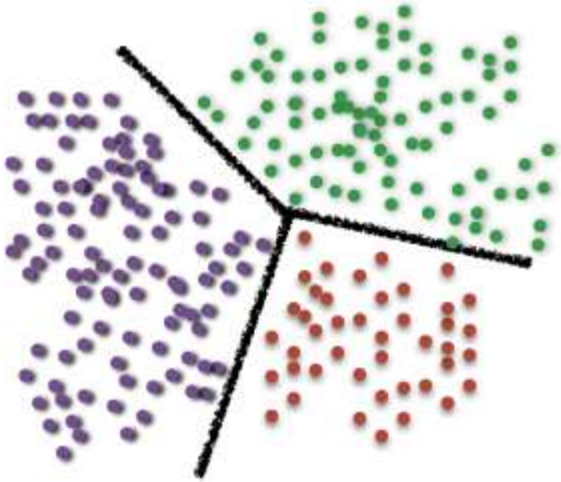
each instance belongs to a cluster or it doesn't

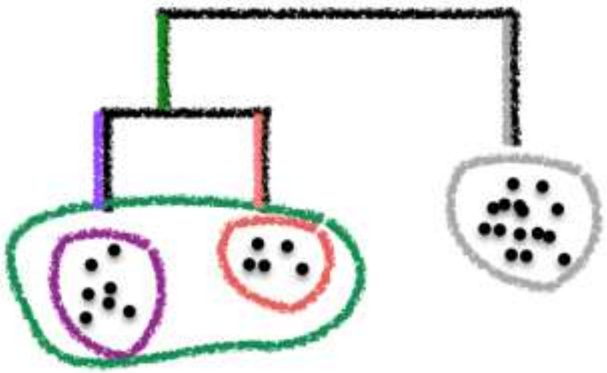
Hierarchical clustering

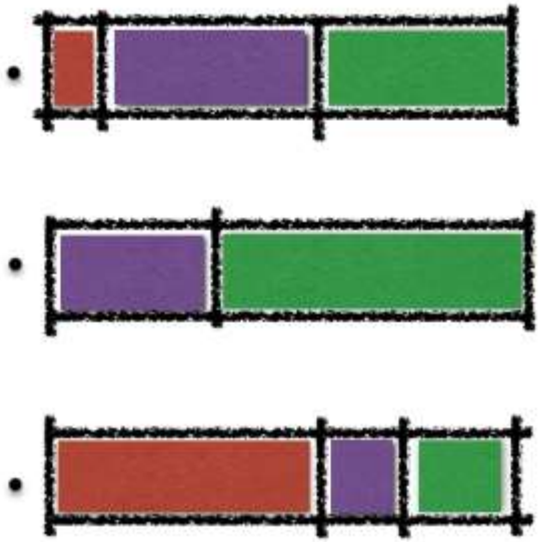
instances that belong to a child cluster also belong to the parent cluster

Soft/fuzzy clustering

each instance can belong to each cluster to a certain degree







Kmeans Clustering

- With vectors of text, any string can be represented with numbers
- If plotted into imaginary space, similar vectors will lay closely
- We can utilize Kmeans to cluster it based on this!
- How to do Kmeans? Anyone?

Kmeans

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

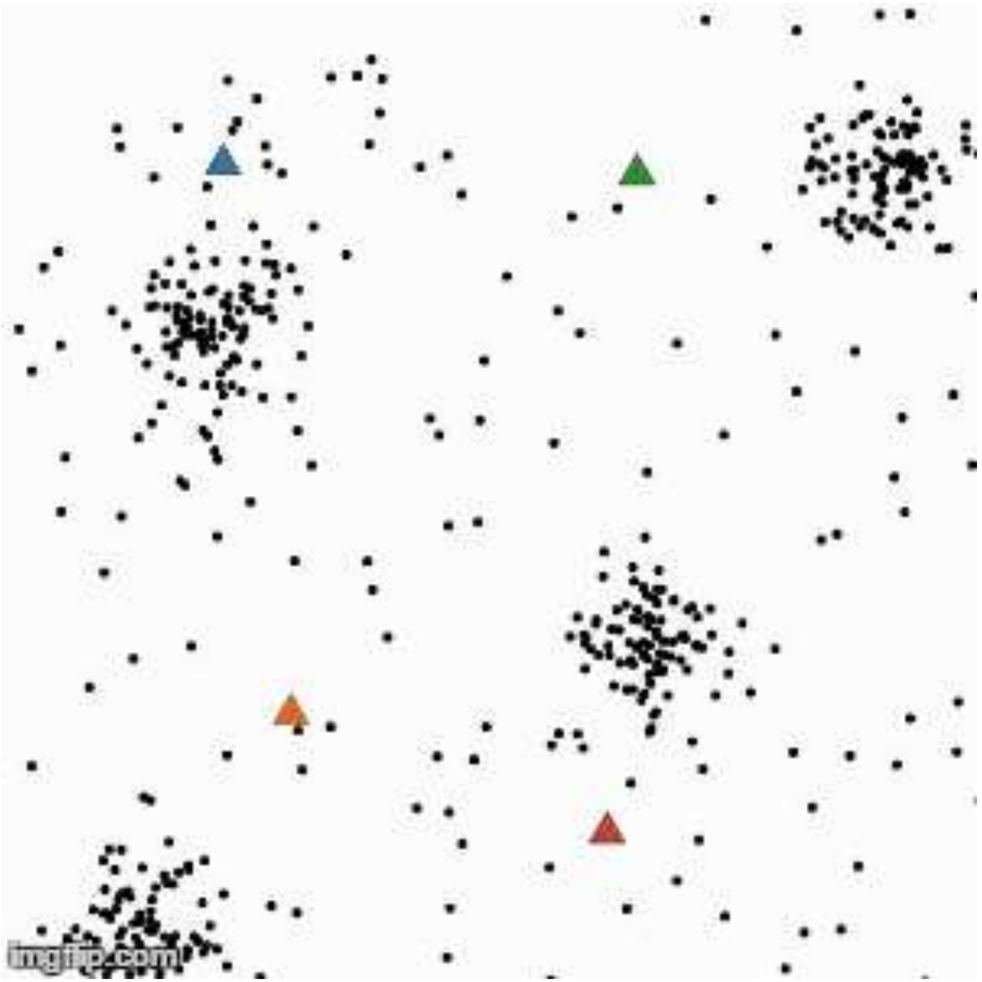
    For every i, set

    $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

    For each j, set

    $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

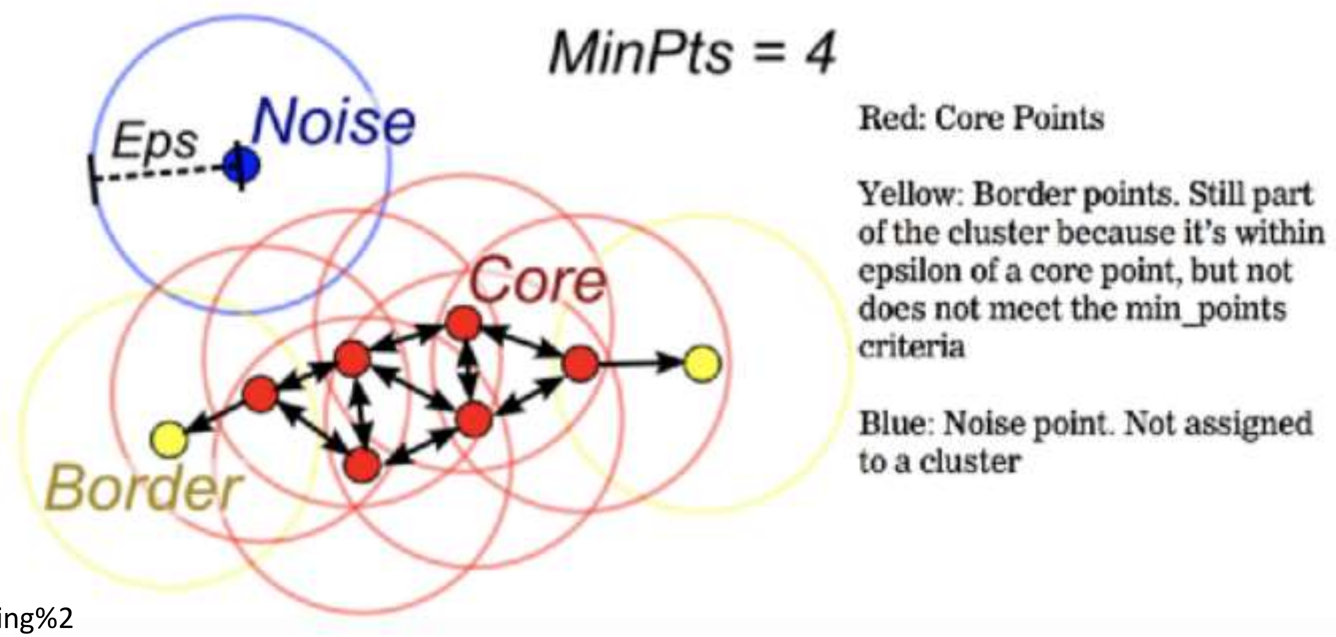
}



DBScan (Density-Based Spatial Clustering of Applications with Noise)

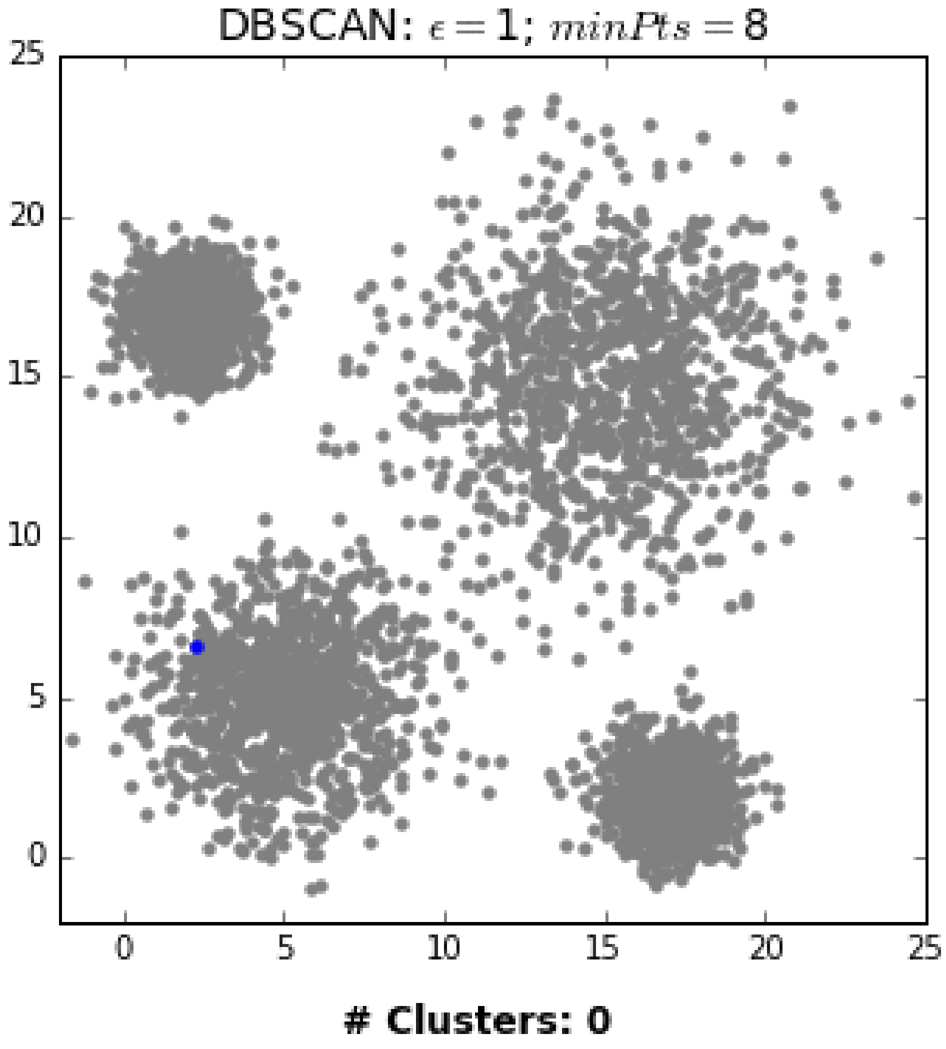
Algoritma DBSCAN:

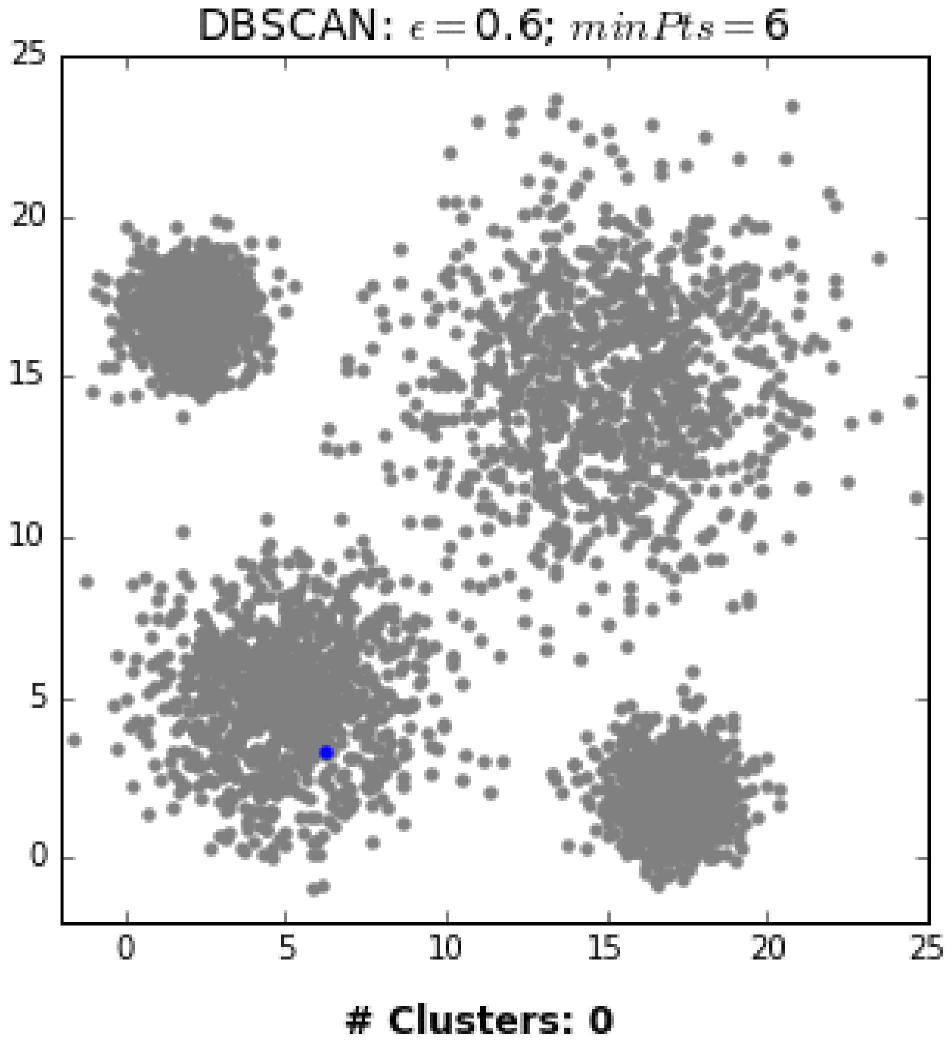
- Pilih poin p sebagai inisialisasi poin
- Pilih neighbour point berdasarkan Eps dan MinPts
- Jika p adalah core point maka terbentuklan cluster 1
- Proses diulangi untuk point yang tidak termasuk ke dalam cluster



https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/#:~:text=DBSCAN%20(Density%2DBased%20Spatial%20Clustering,and%20classifying%20outliers%20as%20noise.

DBScan


DBSCAN: $\epsilon = 1$; $minPts = 8$
25
20
15
10
5
0
0
5
10
15
20
25
Clusters: 0


DBSCAN: $\epsilon = 0.6$; $minPts = 6$
25
20
15
10
5
0
0
5
10
15
20
25
Clusters: 0

Using sample dataset

```python
# import the dataset from sklearn
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# import other required libs
import pandas as pd
import numpy as np

# string manipulation libs
import re
import string
import nltk
from nltk.corpus import stopwords

# viz libs
import matplotlib.pyplot as plt
import seaborn as sns


categories = [
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'alt.atheism',
 'soc.religion.christian',
]
dataset = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, remove=('headers',
'footers', 'quotes'))
```

	corpus
0	\nThey tried their best not to show it, believ...
1	\nStankiewicz? I doubt it.\n\nKoufax was one ...
2	\n[deletia- and so on]\n\nI seem to have been ...
3	Excuse the sheer newbieness of this post, but ...
4	===...
...	...
3446	\n Or, with no dictionary available, they cou...
3447	\n\nSorry to disappoint you but the Red Wings ...
3448	\n: Can anyone tell me where to find a MPEG vi...
3449	\n
3450	\nHey Valentine, I don't see Boston with any w...

3451 rows × 1 columns

```
from sklearn.feature_extraction.text import TfidfVectorizer
# initialize the vectorizer
vectorizer = TfidfVectorizer(sublinear_tf=True, min_df=5, max_df=0.95)
# fit_transform applies TF-IDF to clean texts - we save the array of vectors in X
X = vectorizer.fit_transform(df['cleaned'])
```

```python
from sklearn.cluster import KMeans

# initialize kmeans with 3 centroids
kmeans = KMeans(n_clusters=3, random_state=42)
# fit the model
kmeans.fit(X)
# store cluster labels in a variable
clusters = kmeans.labels_
```

Visualizing Cluster

- If we look at the dimensionality of X with *X.shape* we see that it is (3451, 7390).
- There are 3451 vectors (one for each text), each with 7390 dimensions. It is definitely impossible to visualize them.
- But, we can use PCA (Principal Component Analysis) to reduce (or flatten) the dimension into 2

```python
from sklearn.decomposition import PCA

# initialize PCA with 2 components
pca = PCA(n_components=2, random_state=42)
# pass our X to the pca and store the reduced vectors into pca_vecs
pca_vecs = pca.fit_transform(X.toarray())
# save our two dimensions into x0 and x1
x0 = pca_vecs[:, 0]
x1 = pca_vecs[:, 1]
```

```
1 x0
```

```
array([-0.00152024, -0.03644996, -0.06548033, ...,  0.18883194,
        0.03557314, -0.05786917])
```

```
1 x1
```

```
array([-0.00430002, -0.03914495,  0.08785332, ...,  0.05718469,
       -0.03828756, -0.10444227])
```

```python
# assign clusters and pca vectors to our dataframe
df['cluster'] = clusters
df['x0'] = x0
df['x1'] = x1
```

	corpus	cleaned	cluster	x0	x1
0	\nThey tried their best not to show it, believ...	tried best show believe surprised find sprint ...	0	-0.001520	-0.004300
1	\nStankiewicz? I doubt it.\n\nKoufax was one ...	stankiewicz doubt koufax one two jewish hofs h...	0	-0.036450	-0.039145
2	\n[deletia- and so on]\n\nI seem to have been ...	deletia seem rather unclear asking please show...	2	-0.065480	0.087853
3	Excuse the sheer newbieness of this post, but ...	excuse sheer newbieness post looking decent pa...	1	0.164120	0.062958
4	==...		0	0.035573	-0.038288
...	...	...	...	...	...
3446	\n Or, with no dictionary available, they cou...	dictionary available could gain first hand kno...	0	-0.010061	-0.010073
3447	\n\nSorry to disappoint you but the Red Wings ...	sorry disappoint red wings earned victory easi...	0	-0.029921	-0.158443
3448	\n: Can anyone tell me where to find a MPEG vi...	anyone tell find mpeg viewer either dos window...	1	0.188832	0.057185
3449	\n		0	0.035573	-0.038288
3450	\nHey Valentine, I don't see Boston with any w...	hey valentine see boston world series rings fi...	0	-0.057869	-0.104442

3451 rows × 5 columns

```python
ef get_top_keywords(n_terms):
    """This function returns the keywords for each centroid of the KMeans"""
    df = pd.DataFrame(X.todense()).groupby(clusters).mean() # groups the TF-IDF vector by cluster

    terms = vectorizer.get_feature_names_out() # access tf-idf terms
    for i,r in df.iterrows():
        print('\nCluster {}'.format(i))
        print(','.join([terms[t] for t in np.argsort(r)[-n_terms:]])) # for each row of the
dataframe, find the n terms that have the highest tf idf score

get_top_keywords(10)
```

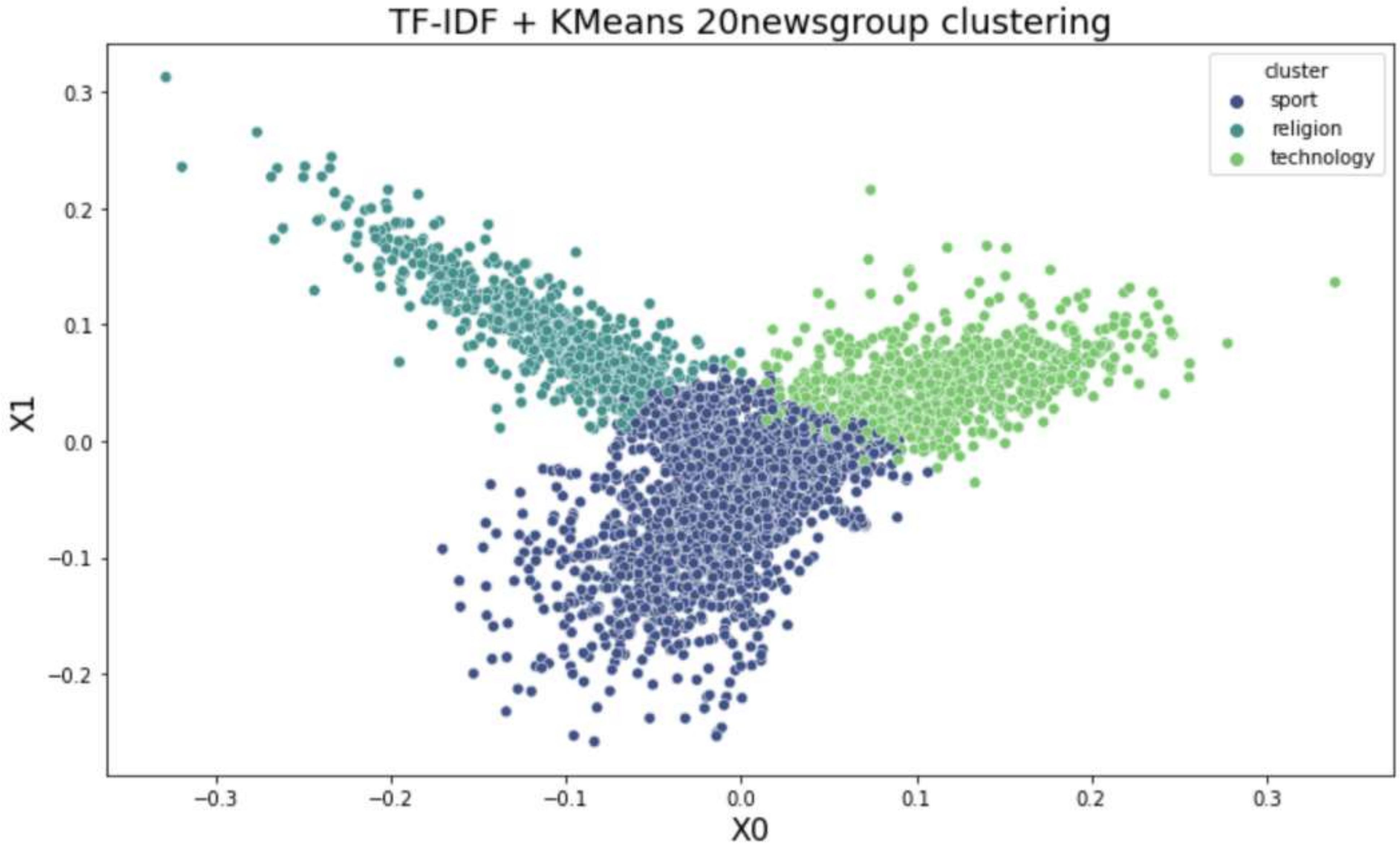
```
Cluster 0
good,last,games,like,would,year,think,one,team,game

Cluster 1
please,dos,use,know,program,anyone,files,file,thanks,windows

Cluster 2
christians,say,think,bible,believe,jesus,one,would,people,god
```

```python
# map clusters to appropriate labels
cluster_map = {0: "sport", 1: "tech", 2: "religion"}
# apply mapping
df['cluster'] = df['cluster'].map(cluster_map)
```

```python
# set image size
plt.figure(figsize=(12, 7))
# set a title
plt.title("TF-IDF + KMeans 20newsgroup clustering", fontdict={"fontsize": 18})
# set axes names
plt.xlabel("X0", fontdict={"fontsize": 16})
plt.ylabel("X1", fontdict={"fontsize": 16})
# create scatter plot with seaborn, where hue is the class used to group the data
sns.scatterplot(data=df, x='x0', y='x1', hue='cluster', palette="viridis")
plt.show()
```


TF-IDF + KMeans 20newsgroup clustering
cluster
sport
religion
technology
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
X1
X0

Any Questions?