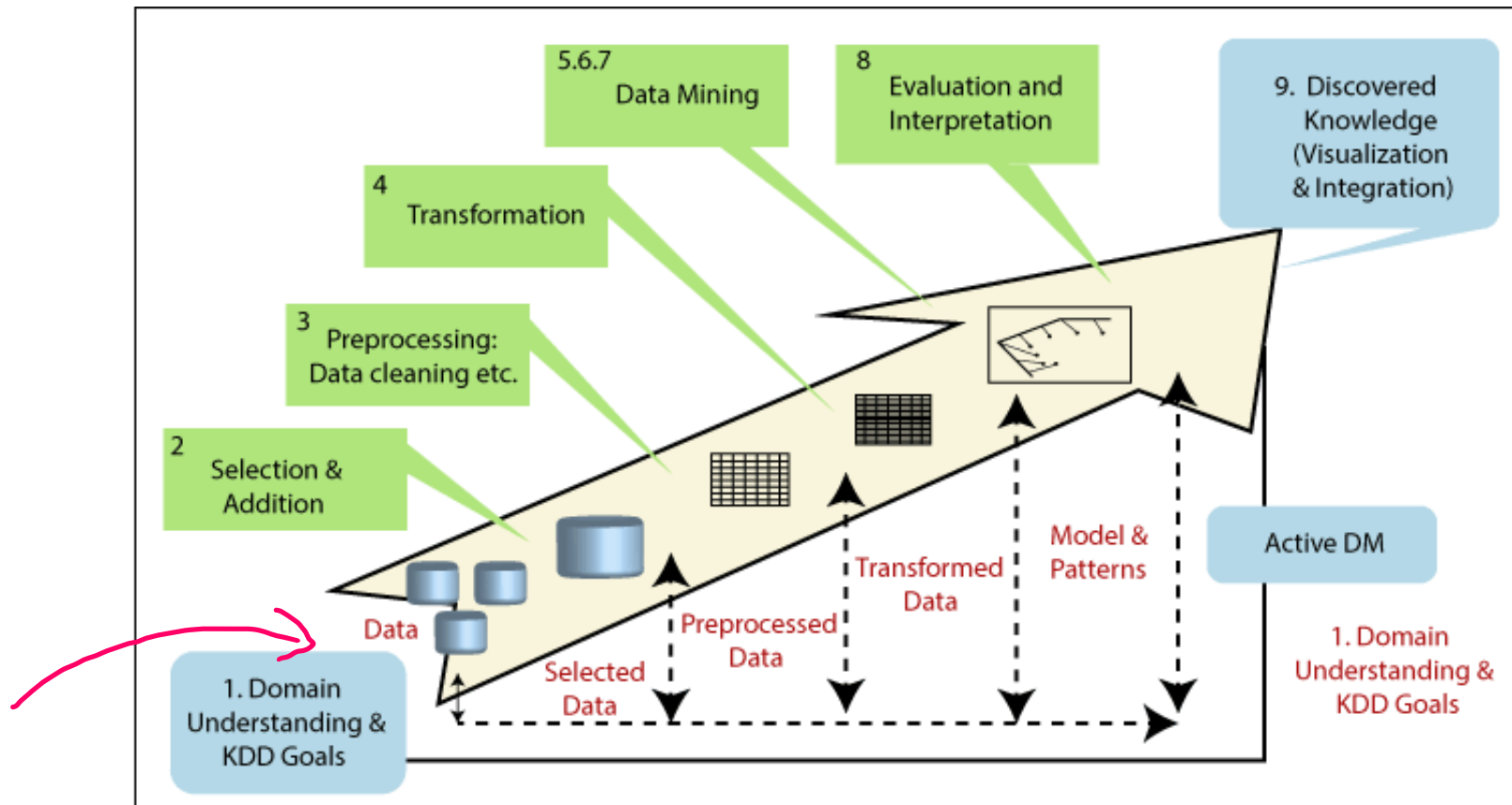


SE 953482 Natural Language
Processing for SE
66/2
Text Extractions

Where we are now

1.	NLP Overview	3
2.	Data science methodology	3
3.	Word Tokenization, Text preprocessing	3
3.	Text extraction methods	6
4.	Machine-learning models in NLP	6
5.	Deep-learning models in NLP	6
6.	Transformers	3
7.	Evaluation metrics and explain-ability	3
8.	NLP-based Systems	3
9.	Case studies and Project	9

Knowledge Discovery in Databases Process



Definition

- Open question : Where are the data come from?

Let's discuss about the previous workshop...

How would you obtain the data, entities, and attributes

Primary data vs Secondary data

- **Primary data** – firsthand data or raw data
 - Expensive
 - Various methods (e.g., surveys, interview, focus groups, case studies, etc.).
- **Secondary data** – been collected by someone else (published data)
 - Easily available (Kaggle, githib, reddit, UCI repo, data.go.th
 - Irrelevance, redundant, and less accuracy
 - Books, reports, censuses, government publications, etc.

Example

- John's experiment used data from a book
- Marry conducted her experiments through questionnaire by surveying his organization

Where the data come from?

- Massive of digital information
- Credit cards
- Social networking
- Capturing traffic flow
- Measuring pollution
- Interviews, Surveys, Questionnaires
- Etc.

What's Driving Data Deluge?



Mobile
Sensors



Social
Media



Video
Surveillance



Video
Rendering



Smart
Grids



Geophysical
Exploration



Medical
Imaging



Gene
Sequencing

Comparison

Metrics	Primary	Secondary
Accuracy	High	Low
Control	High	Low
Relevancy	High	Low
Ownership	?	?
Accessibility	?	? high
Bias	?	?
Up-to-dated	?	?

Basic methods to access the data

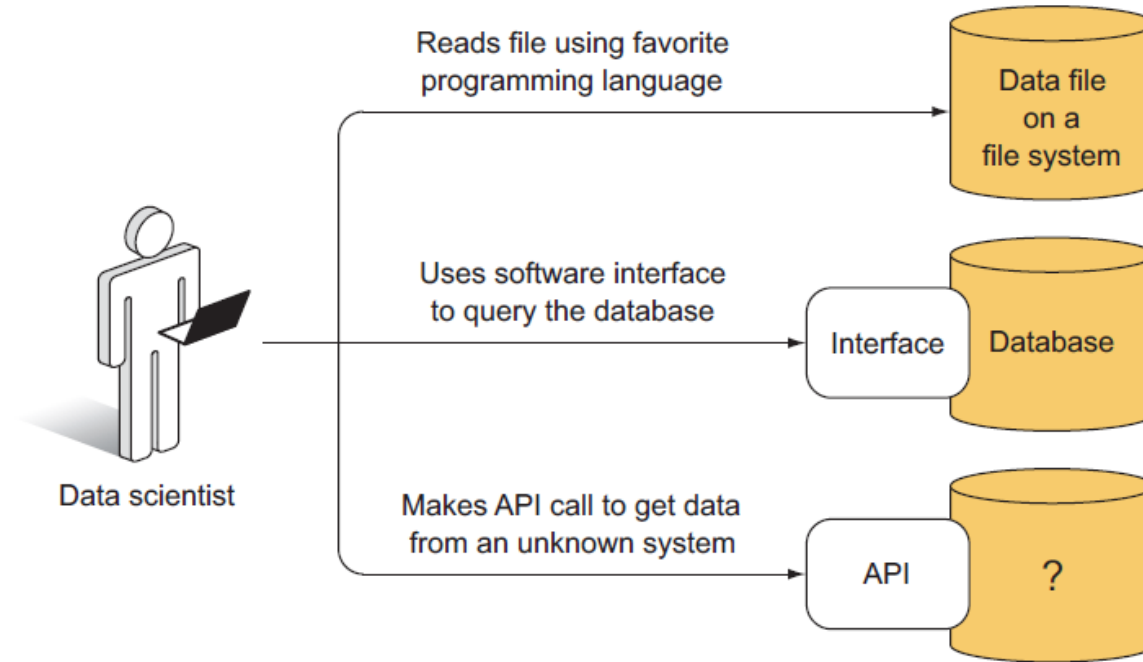
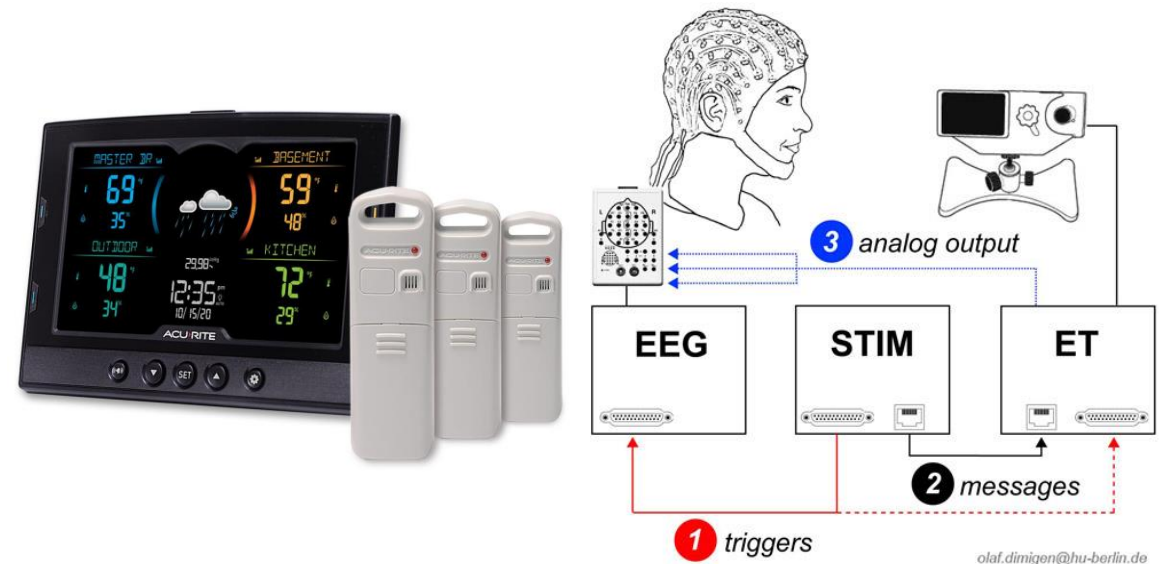


Figure 3.3 Three ways a data scientist might access data: from a file system, database, or API

Data acquisition (Time-series)

- Hardware, Software, questionnaire, interview
 - To allow us to measure something in the real world.
 - Weather station (Temp. and humidity)
 - EYE-EEG (operate between 500-1000mhz)



Open question: There should be way to collected data, could you explore more method?

Surveys/ Questionnaires (Multivariate)

- Usually deployed by utility companies, building management companies, energy analysis companies or government.
- **Field Interviewer** from the Energy Information Administration (EIA)
 - Use questionnaire to collect data from selected housing uniting.
 - Data includes
 - Building characteristic
 - Energy consumption and expense
 - Household demographics
- Data from interview + data from energy suppliers
 - Estimate energy costs
 - Usage for heating and cooling

Home Energy Check

1. What sort of home do you live in?
☐ Detached ☐ Top Flat
☐ Semi-detached ☐ Mid Flat
☐ End Terrace ☐ Bottom Flat
☐ Mid Terrace ☐ Maisonette

2. What type of outside walls do you have?
☐ Solid ☐ Cavity
☐ Timber Frame ☐ Don't Know

3. Are these walls insulated?
☐ Yes ☐ No ☐ Don't know

4. Does your home have loft insulation?
☐ 4 inches ☐ 6 inches ☐ 8 inches
☐ More ☐ Don't know ☐ None

5. What kind of windows do you have?
☐ Single glazing
☐ Double glazing
☐ Secondary glazing

6. Are external doors and windows draught-proofed?
☐ Yes ☐ No

7. How old is your heating system or boiler?
Approx. years

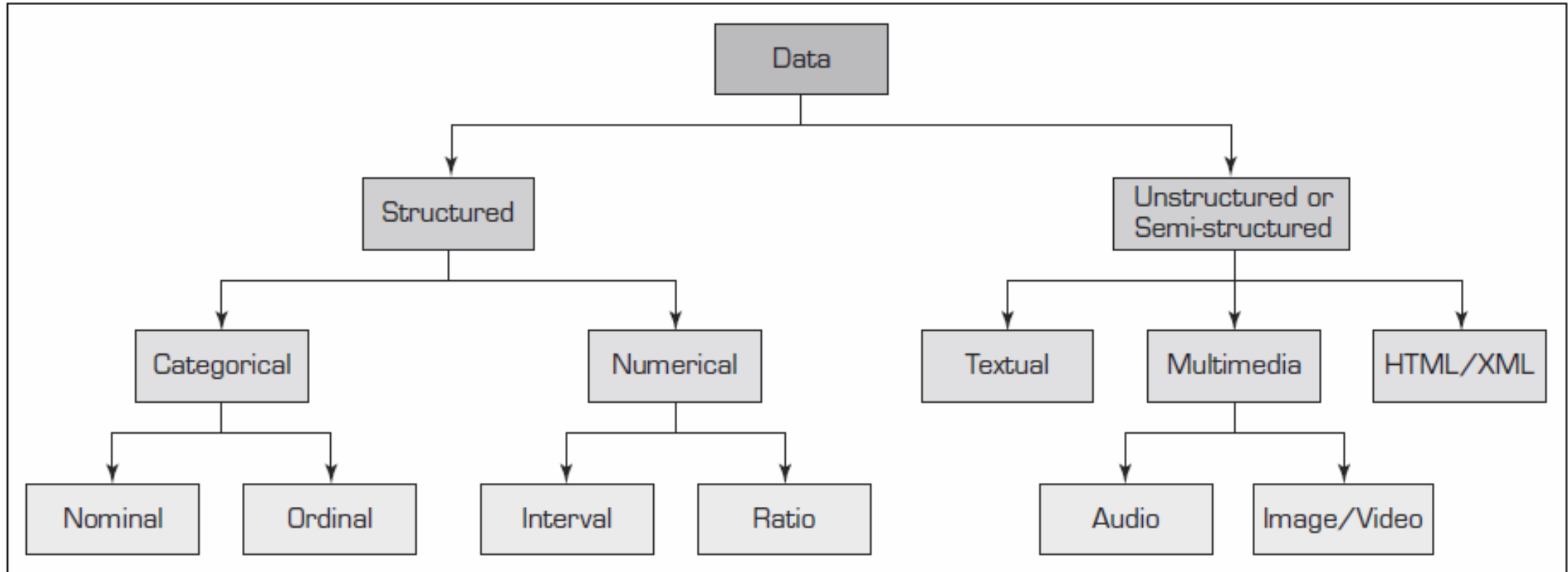
8. What is the main fuel used for room heating?
☐ Mains gas ☐ Stored LPG
☐ Bottled gas ☐ Coal
☐ Oil ☐ Wood
☐ Electricity (on peak) ☐ Electricity (off peak)

9. What heating controls do you have?
☐ Room thermostat
☐ Thermostatic radiator valves
☐ Automatic charge control for storage heaters
☐ Timer / programmer

Social media (Multivariate + Textual)

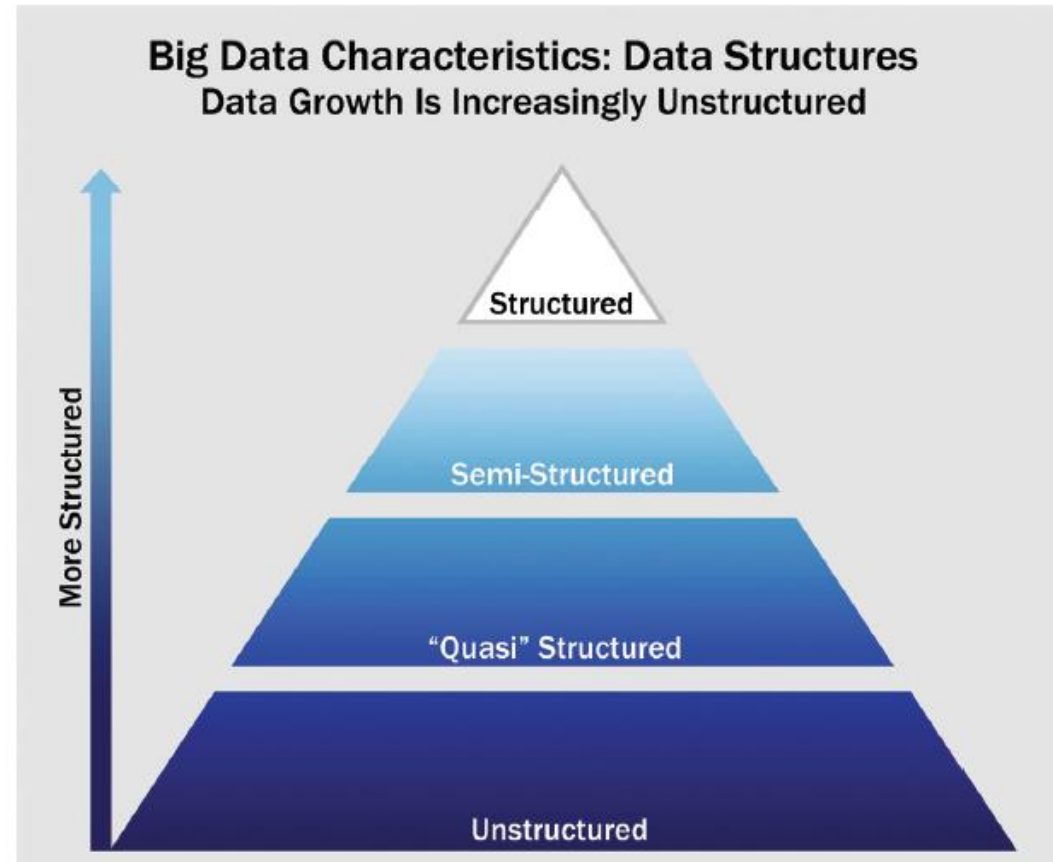
- Twitter
- Facebook
- Pantip
- Ecommerce website like JD.com, Wongnai, etc
- WebCrawler /Scraper

Data Mining – Taxonomy of Data in DM



(Turban, Sharda, & Delen, 2014)

Big data growth increase in unstructure



Structure data

- Data containing a defined data type, format, and structure
- E.g. Transaction data, online analytical processing [OLAP] data cubes, traditional RDBMS, CSV files, and even simple spreadsheets).

SUMMER FOOD SERVICE PROGRAM 1]				
(Data as of August 01, 2011)				
Fiscal Year	Number of Sites	Peak (July) Participation	Meals Served	Total Federal Expenditures 2]
	-----Thousands-----		--Mil.--	---Million \$---
1969	1.2	99	2.2	0.3
1970	1.9	227	8.2	1.8
1971	3.2	569	29.0	8.2
1972	6.5	1,080	73.5	21.9
1973	11.2	1,437	65.4	26.6
1974	10.6	1,403	63.6	33.6
1975	12.0	1,785	84.3	50.3
1976	16.0	2,453	104.8	73.4
TQ 3]	22.4	3,455	198.0	88.9
1977	23.7	2,791	170.4	114.4
1978	22.4	2,333	120.3	100.3
1979	23.0	2,126	121.8	108.6
1980	21.6	1,922	108.2	110.1
1981	20.6	1,726	90.3	105.9
1982	14.4	1,397	68.2	87.1
1983	14.9	1,401	71.3	93.4
1984	15.1	1,422	73.8	96.2
1985	16.0	1,462	77.2	111.5
1986	16.1	1,509	77.1	114.7
1987	16.9	1,560	79.9	129.3
1988	17.2	1,577	80.3	133.3
1989	18.5	1,652	86.0	143.8
1990	19.2	1,692	91.2	163.3

Data Type - Qualitative

- Extract from field notes, interview transcripts
- Data can be expressed in discrete (i.e. categorical, enumerated) as follows:
 - **Nominal**- variable with no inherent order or ranking sequence (e.g. gender, nationality)
 - **Ordinal** (socio-economic status)

Open question Qualitative vs Quantitative?

Qualitative vs Quantitative data

Qualitative	Quantitative
Origin = SC	Origin = NS
Sample size = Small	Sample size = Large
Cost = Low-High	Cost = Low-High
Style = personal voice, literary	Style = formal, scientific
Type = Description	Type = numerical
Source = Interviews	Source = Instruments
2+3 more..	

Data type - Nominal data

Also known as categorical
Finite set, qualitative
No order

What is your gender?

- ☒ M – Male
- ☐ F – Female

What is your hair color?

- ☒ 1 – Brown
- ☐ 2 – Black
- ☐ 3 – Blonde
- ☐ 4 – Gray
- ☐ 5 – Other

Where do you live?

- ☒ A – North of the equator
- ☐ B – South of the equator
- ☐ C – Neither: In the international space station

Data type - Ordinal data

Similar with nominal but with rank order

How do you feel today?


- ☒ 1 – Very Unhappy
- ☐ 2 – Unhappy
- ☐ 3 – OK
- ☐ 4 – Happy
- ☐ 5 – Very Happy

How satisfied are you with our service?

- ☒ 1 – Very Unsatisfied
- ☐ 2 – Somewhat Unsatisfied
- ☐ 3 – Neutral
- ☐ 4 – Somewhat Satisfied
- ☐ 5 – Very Satisfied

Dataset with attribute and class

DATASET WITH ATTRIBUTE AND CLASS



The diagram illustrates the structure of the dataset. An arrow labeled 'Attribute' points to the first four columns: Sepal Length (cm), Sepal Width (cm), Petal Length (cm), and Petal Width (cm). Another arrow labeled 'Class/Label' points to the fifth column: Type.

	Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Type
1	5.1	3.5	1.4	0.2	<i>Iris setosa</i>
2	4.9	3.0	1.4	0.2	<i>Iris setosa</i>
3	4.7	3.2	1.3	0.2	<i>Iris setosa</i>
4	4.6	3.1	1.5	0.2	<i>Iris setosa</i>
5	5.0	3.6	1.4	0.2	<i>Iris setosa</i>
...					
51	7.0	3.2	4.7	1.4	<i>Iris versicolor</i>
52	6.4	3.2	4.5	1.5	<i>Iris versicolor</i>
53	6.9	3.1	4.9	1.5	<i>Iris versicolor</i>
54	5.5	2.3	4.0	1.3	<i>Iris versicolor</i>
55	6.5	2.8	4.6	1.5	<i>Iris versicolor</i>
...					
101	6.3	3.3	6.0	2.5	<i>Iris virginica</i>
102	5.8	2.7	5.1	1.9	<i>Iris virginica</i>
103	7.1	3.0	5.9	2.1	<i>Iris virginica</i>
104	6.3	2.9	5.6	1.8	<i>Iris virginica</i>
105	6.5	3.0	5.8	2.2	<i>Iris virginica</i>
...					

The data type of an attribute (numeric, ordinal, nominal) affects the methods we can use to analyze and understand the data.

Scaling for numerical input

- To range (-1, +1)
- To speed up the convergence

[0] 30 male 38000.00 urban democrat

[1] 36 female 42000.00 suburban republican

[2] 52 male 40000.00 rural independent

[3] 42 female 44000.00 suburban other

Scaling for numerical input (cont.)

- [0] -1.23 -1.0 -1.34 (0.0 1.0) (0.0 0.0 0.0 1.0)
- [1] -0.49 1.0 0.45 (1.0 0.0) (0.0 0.0 1.0 0.0)
- [2] 1.48 -1.0 -0.45 (-1.0 -1.0) (0.0 1.0 0.0 0.0)
- [3] 0.25 1.0 1.34 (1.0 0.0) (1.0 0.0 0.0 0.0)

Data hidden in text

The image is a screenshot of a web browser displaying the Twitter profile of Donald J. Trump (@realDonaldTrump). The browser's address bar shows the URL <https://twitter.com/realDonaldTrump?lang=en>. The page features a header with navigation links for Home, Notifications (with a 4 badge), and Messages. A search bar and a 'Tweet' button are also present. The profile banner image shows a crowd at a rally with 'BUILD THE WALL' signs. The profile picture is a circular portrait of Donald Trump. Below the banner, statistics are listed: 41.3K Tweets, 45 Following, 59.7M Followers, 8 Likes, and 6 Moments. A 'Follow' button is on the right. The left sidebar contains the name 'Donald J. Trump' with a verified badge, his handle '@realDonaldTrump', his title '45th President of the United States of America', his location 'Washington, DC', a link to his Instagram profile, and his join date 'March 2009'. Below this is a 'Tweet to Donald J. Trump' button and a section for '5 Followers you know' with small profile pictures. The main content area shows three tweets from Donald J. Trump, each with a dropdown arrow. The first tweet, posted 2 hours ago, says 'Congress should come back to D.C. now and FIX THE IMMIGRATION LAWS!' and has 5.7K replies, 8.3K retweets, and 35K likes. The second tweet, also 2 hours ago, discusses Mueller's findings and accuses Hillary Clinton, the DNC, and 'Dirty Cops' of crimes, with a call to 'INVESTIGATE THE INVESTIGATORS!'. It has 7.0K replies, 9.0K retweets, and 32K likes. The third tweet, posted 3 hours ago, is a longer statement about branding and the Boeing 737 MAX. The right sidebar shows a 'Who to follow' section with suggestions for 'President Trump', 'Hillary Clinton', and 'CNN', each with a 'Follow' button. At the bottom of this section is a 'Find people you know' link to import contacts from Gmail.

Donald J. Trump (@realDonaldTrump)

41.3K Tweets 45 Following 59.7M Followers 8 Likes 6 Moments

Donald J. Trump @realDonaldTrump

45th President of the United States of America

Washington, DC

[Instagram.com/realDonaldTrump](https://www.instagram.com/realDonaldTrump)

Joined March 2009

[Tweet to Donald J. Trump](#)

5 Followers you know

Tweets Tweets & replies Media

Donald J. Trump @realDonaldTrump · 2h
Congress should come back to D.C. now and FIX THE IMMIGRATION LAWS!
5.7K 8.3K 35K

Donald J. Trump @realDonaldTrump · 2h
Mueller, and the A.G. based on Mueller findings (and great intelligence), have already ruled No Collusion, No Obstruction. These were crimes committed by Crooked Hillary, the DNC, Dirty Cops and others! INVESTIGATE THE INVESTIGATORS!
7.0K 9.0K 32K

Donald J. Trump @realDonaldTrump · 3h
What do I know about branding, maybe nothing (but I did become President!), but if I were Boeing, I would FIX the Boeing 737 MAX, add some additional great features, & REBRAND the plane with a new name.
No product has suffered like this one. But again, what the hell do I know?

Who to follow · Refresh · View all

- President Trump** @POT... [Follow](#)
- Hillary Clinton** @Hillary... [Follow](#)
- CNN** @CNN [Follow](#)

[Find people you know](#)
Import your contacts from Gmail

[Connect other address books](#)

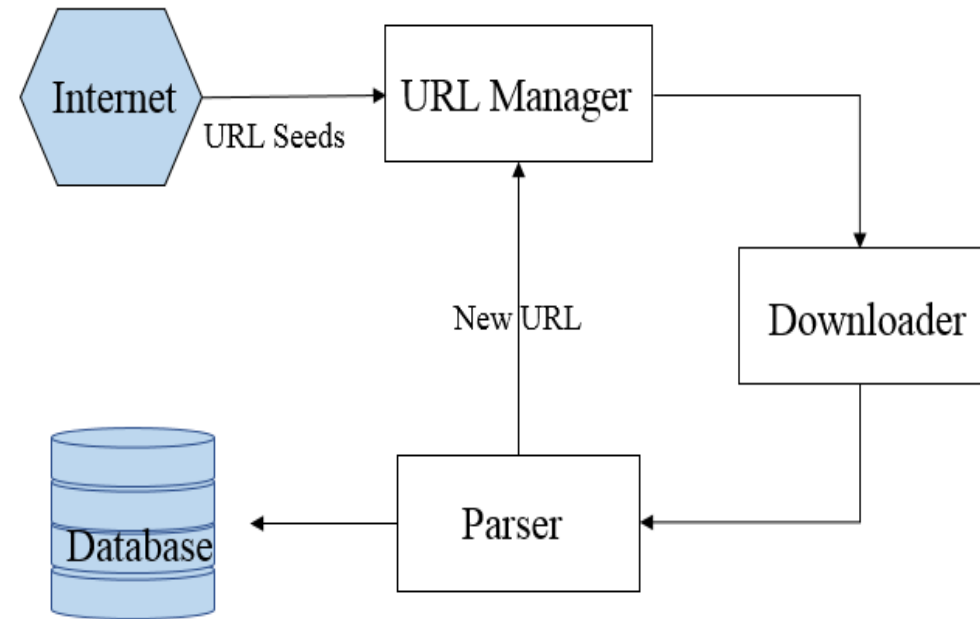
WebCrawler / scraper

- <https://www.twitter.com>
- Beautiful soup
- Scrapy
- Selenium

Architecture of Web Crawler

The web crawlers can continuously download content of web pages and search for new URLs.

A crawler framework mainly includes the URL managers, downloaders, and parsers



Architecture of Web crawler

Handling Text in Python

```
>>> txt1 = 'The Radical Left Democrats will never be satisfied with anything we give them. They will always Resist and Obstruct!'
```

```
>>> len(txt1)
```

```
116
```

```
>>> txt2 = txt1.split(' ');
```

```
>>> len(txt2);
```

```
19
```

```
>>> txt2
```

```
['The', 'Radical', 'Left', 'Democrats', 'will', 'never', 'be', 'satisfied', 'with', 'anything', 'we', 'give', 'them.', 'They', 'will', 'always', 'Resist', 'and', 'Obstruct!']
```

```
>>>
```

Handling Text in Python (cont.)

- **Finding long words: e.g. words that has more than 3 letters.**

```
>>> [w for w in txt2 if len(w) > 3]
```

```
['Radical', 'Left', 'Democrats', 'will', 'never', 'satisfied', 'with', 'anything',  
'give', 'them.', 'They', 'will', 'always', 'Resist', 'Obstruct!']
```

- **Finding capitalized words:**

```
>>> [w for w in txt2 if w.istitle()]
```

```
['The', 'Radical', 'Left', 'Democrats', 'They', 'Resist', 'Obstruct!']
```

Handling Text in Python (cont.)

Find words which ends with 's':

```
>>> [w for w in txt2 if w.endswith('s')]
['Democrats', 'always']
```

Find unique words: using set():

```
>>> txt3 = 'To be or not to be'
```

```
>>> txt4 = txt3.split(' ')
```

```
>>> len(txt4)
```

```
6
```

```
>>> len(set(txt4))
```

```
5
```

```
>>> set(txt4)
```

```
{'be', 'not', 'to', 'To', 'or'}
```

Handling Text in Python (cont.)

```
>>> set([w.lower() for w in txt4])  
{'be', 'not', 'or', 'to'}  
>>>
```

Handling Text in Python (cont.)

- **Word comparison in Python**

- `s.startswith(t)`
- `s.endswith(t)`
- `t in s`
- `s.istitle()`
- `s.islower()`
- `s.isupper()`
- `s.isdigit()`, `s.isalnum()`, `s.isalpha()`

String operations

- `s.lower(); s.upper(); s.titlecase()`
- `s.split()`
- `s.splitlines()`
- `s.join(t) //`
- `s.strip() //` take out all the white space
- `s.find(t)//` find the specific substring
- `s.replace(u, v)//` u will be replace by v

Words to characters

```
>>> txt5 = 'ouagadougou'
```

```
>>> txt6 = txt5.split('ou')
```

```
>>> txt6
```

```
['', 'agad', 'g', '']
```

```
>>> 'ou'.join(txt6)
```

```
'ouagadougou'
```

Words to characters (cont.)

```
>>> txt5.split("")
```

Traceback (most recent call last):

File "<pyshell#7>", line 1, in <module>
txt5.split("")

ValueError: empty separator

```
>>> list(txt5)
```

```
>> [c for c in txt5]
```

```
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

Cleaning text

```
>>> txt8 = ' A quick brown fox jumped over the lazy dog. '
```

```
>>> txt8.split(' ')
```

```
['', ' ', '\t', 'A', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog.', '']
```

```
>>> txt8.strip()
```

```
'A quick brown fox jumped over the lazy dog.'
```

Changing text

```
>>> txt9 = txt8.strip()
```

```
>>> txt9
```

```
'A quick brown fox jumped over the lazy dog.'
```

```
>>> txt9.find('o')
```

```
10
```

```
>>> txt9.rfind('o')
```

```
40
```

```
>>> txt9.replace('o', 'O')
```

```
'A quick brOwn fOx jumped Over the lazy dOg.'
```

Handling larger texts

- **Reading files line by line**

```
>>> f = open('/Users/Pree/Desktop/notre dame.txt')
```

```
>>> f.readline()
```

```
'Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval Catholic cathedral on the Île de la Cité in the 4th arrondissement of Paris, France.\n'
```

- **Reading the full file**

```
>>> f.seek(0)
```

```
0
```

```
>>> txt12 = f.read()
```

```
>>> len(txt12)
```

Handling larger texts (cont.)

```
>>> txt13 = txt12.splitlines()
```

```
>>> len(txt13)
```

```
>>> txt13
```

```
['Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval  
Catholic cathedral on the Île de la Cité in the 4th arrondissement of Paris,  
France.', 'The cathedral is consecrated to the Virgin Mary and considered to be  
one of the finest examples of French Gothic architecture.']
```

```
>>> txt13[0]
```

```
'Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval  
Catholic cathedral on the Île de la Cité in the 4th arrondissement of Paris,  
France.'
```

File operations

- `f = open(filename, mode)`
- `f.readline(); f.read(); f.read(n)`
- `for line in f: doSomething(line)`
- `f.seek(n)`
- `f.write(message)`
- `f.close()` // close the file handler
- `f.closed` // check if the file close

Processing free-text

```
>>> txt10 = '#DrainTheSwamp - @GreggJarrett: No one takes anything  
Schiff says seriously because he lost all credibility. For 2 years he  
claimed there was a mound of criminal evidence. Where is it? Show  
us... because it doesn't exist. #MAGA #AmericaFirst #Dobbs'
```

How to find out the callouts and hashtags?

Finding specific words

Hashtags

```
>>> [w for w in txt10.split() if w.startswith('#')]
['#DrainTheSwamp', '#MAGA', '#AmericaFirst', '#Dobbs']
```

Callouts

```
>>> [w for w in txt10.split() if w.startswith('@')]
['@GreggJarrett:', '@']
```

Regular expression

- Known as "regex" or "regexp", are a powerful tool used in computing for pattern matching within strings.
- *Searching and Extracting* – emails, URLs, etc.
- *Data Validation* – checking input format
- *String Manipulation*- split string

Finding patterns with regular expressions

- Callouts are more than just tokens starting with '@'
- @username @UK_Spokesperon
- Match something after '@'
 - Alphabets
 - Numbers
 - Special symbols such as '_'

Finding patterns with regular expressions

Callouts

```
>>> w for w in txt10.split() if w.startswith('@')]  
['@GreggJarrett:', '@']
```

Import regular expressions first

```
import re  
[w for w in txt10.split(' ') if re.search('@[A-Za-z0-9_]+', w)]  
['@GreggJarrett:']
```

Parsing the callout regular expression

- `@[A-Za-z0-9_]+`
- Starts with `@`
- Followed by any alphabet (upper or lower case), digit, underscore
- That repeats at least once, but any number of times

Metacharacters

Metacharacters are characters with a special meaning:

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence	"\d"
.	Any character(except newline character)	"he..o"
^	Starts with	"^hello"
\$	Ends with	"world\$"
* (repetitions)	Zero or more occurrences	"aix*"
+ (repetition) ? (repetition)	One or more occurrences Zero or one occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
	Either or	"falls stays"

Metacharacters

- `\d` – any digit
- `\D` – any non-digit
- `\s` – any white space char, `[\t\n\r\f\v]`
- `\S` – opposite the above
- `\w` – Alphanumeric character , `[a-zA-Z0-9_]`
- `\W` – `[^ a-zA-Z0-9_]`

Sets

A set is a set of characters inside a pair of square brackets

Set	Description
[arn]	Returns a match where one of the specified characters (a , r , or n) are present
[a-n]	Returns a match for any lower-case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a , r , and n
[0123]	Returns a match where any of specified digits(0 , 1 , 2 , or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit number from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z , lower case OR upper case

Example 1 – search() function

The search() function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

```
import re
```

```
#Check if the string starts with "The" and ends with " ChiangMai":
```

```
txt = "The rain in ChaingMai"
```

```
x = re.search("^The.*ChiangMai$", txt)
```

```
if (x):
```

```
    print("YES! We have a match!")
```

```
else:
```

```
    print("No match")
```

Example 2 – findall() function

- The findall() function returns a list containing all matches.
- import re

#Return a list containing every occurrence of "ai":

```
str = "The rain in Chaing Mai"  
x = re.findall("ai", str)  
print(x)  
['ai', 'ai']
```

Example 2.1 findall() function

- Finding specific characters

```
>>> txt12 = 'ouagadougou'
```

```
>>> re.findall(r'[aeiou]', txt12)
```

```
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']
```

```
>>> re.findall(r'^[aeiou]', txt1)
```

```
['g', 'd', 'g']
```

Example 3 – sub() function

- The sub() function replaces the matches with the text of your choice:

```
import re
```

```
str = "The rain in Chaing Mai"
```

```
x = re.sub("\s", "9", str)
```

```
print(x)
```

```
The9rain9in9Chiang Mai
```

Discussion and class activity

Step 1 KDD (Or most of the DM/ML Process)

- **Initial Selection (data understanding)**
- **Select one of the dataset week3.1 or week3.2 that suit your background.**
 - Read the dataset (Python or R)
 - Use your domain knowledge
 - What is your DS task?
 - Identify type of data (nominal, ordi...)
 - How many samples and feature?
 - Remove feature(s) that is not relevant for your model
 - **Extra credit** (is there any instance/row) that is useless. How many customer in the dataset should you get rid off?

What we have learned so far

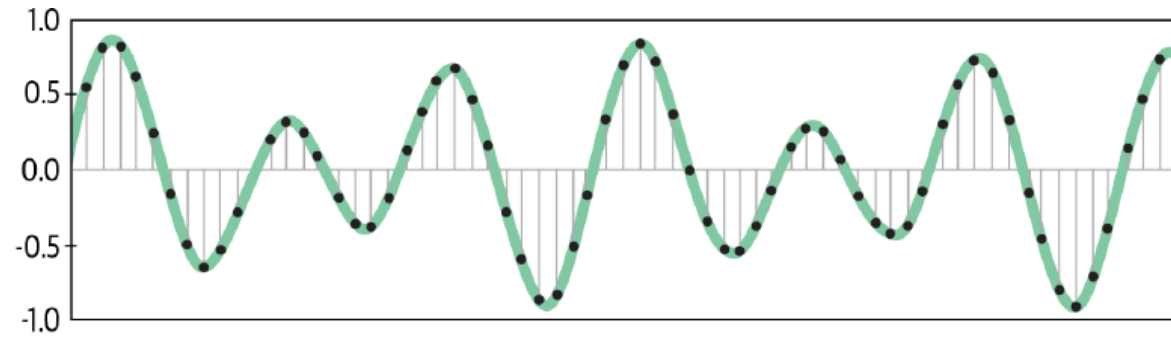
- First stage of KDD
- Data pre-processing
 - Handling multi-variance attrs
 - Handling text sentences
 - Splitting sentences into words
 - Splitting words into characters
 - Finding unique words
 - Intro to Regular Expression and operations
 - Words to vectors
 - Word tokenization
 - Feature extraction
- BOW Modelling

Human vs Computer (Image)



08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 57 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 24 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 34 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

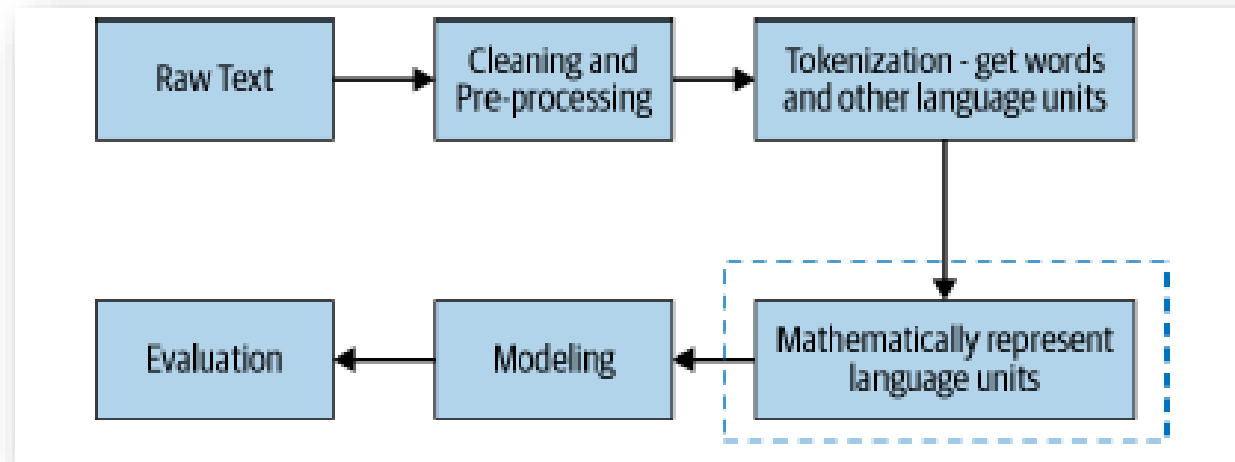
Human vs Computer (Time serie)



```
[-1274, -1252, -1160, -986, -792, -692, -614, -429, -286, -134, -57, -41,  
-169, -456, -450, -541, -761, -1067, -1231, -1047, -952, -645, -489, -448,  
-397, -212, 193, 114, -17, -110, 128, 261, 198, 390, 461, 772, 948, 1451,  
1974, 2624, 3793, 4968, 5939, 6057, 6581, 7302, 7640, 7223, 6119, 5461,  
4820, 4353, 3611, 2740, 2004, 1349, 1178, 1085, 901, 301, -262, -499,  
-488, -707, -1406, -1997, -2377, -2494, -2605, -2675, -2627, -2500, -2148,  
-1648, -970, -364, 13, 260, 494, 788, 1011, 938, 717, 507, 323, 324, 325,  
350, 103, -113, 64, 176, 93, -249, -461, -606, -909, -1159, -1307, -1544]
```

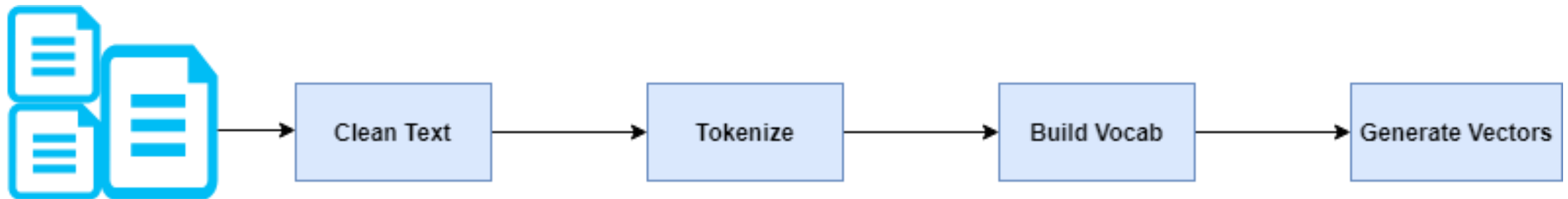

Text representations

- One of the most important step in ML problem
- Poor feature -> poor results



Text pre-processing pipeline with BOW

- Clean the text (white space, upper case, etc.)
- Tokenizing
- Build dictionary
- Filter is also useful (sorting histogram and take top 50)
- The vectors will be use in ML algorithms for document classification or clustering.



Bag of Words (BOW)

- Machine learning can't work with raw text
- Text must be convert to numbers, vectors of numbers
- Usually this is called “feature extraction”
- Bag of words is the most simple approach.
- Represent of string based on frequency of entities



Bag of Words Example

	Document 1	Document 2
The quick brown fox jumped over the lazy dog's back.		
Now is the time for all good men to come to the aid of their party.		

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List
for
is
of
the
to

BOW (cont.)

- Sentence 1: "I love apples."
- Sentence 2: "I do not love oranges."
- Vocab = ["I", "love", "apples", "do", "not", "oranges"]
- Vector for Sentence 1: [1, 1, 1, 0, 0, 0]
- Vector for Sentence 2: [1, 1, 0, 1, 1, 1]

BOW in action

```
1 sentences = ['sky is nice', 'clouds are nice', 'Sky is nice and Clouds are nice']
2
3 cleaned_sentence = []
4
5 for sentence in sentences:
6     word = sentence.lower()
7     ##lowering all the letters becaz we dont want it to treat uppercase and lower case words differently
8
9     word = word.split()    ##splitting our sentence into words
10
11     ##removing stop words
12     word = [i for i in word if i not in set(stopwords.words('english'))]
13     word = " ".join(word)    ##joining our words back to sentences
14     cleaned_sentence.append(word)    ##appending our preprocessed sentence into a new list
15
16
17 ## printing our new list
18 print(cleaned_sentence)
```

BOW in action

```
1 from sklearn.feature_extraction.text import CountVectorizer
2
3 cv = CountVectorizer(max_features = 10)
4 Bagofwords = cv.fit_transform(cleaned_sentence).toarray()
5
6 print(Bagofwords)
```

BOW modeling in action

```
wordfreq = {} # create dict and iterate through each sentence
for sentence in corpus:
    tokens = nltk.word_tokenize(sentence)
    for token in tokens:
        if token not in wordfreq.keys():
            wordfreq[token] = 1
        else:
            wordfreq[token] += 1
|
# filter dimension to 300
import heapq
most_freq = heapq.nlargest(300, wordfreq, key=wordfreq.get)
```

```
sentence_vectors = [] # create sentence list and iterate through each sentence
for sentence in corpus:
    sentence_tokens = nltk.word_tokenize(sentence)
    sent_vec = []
    for token in most_freq:
        if token in sentence_tokens:
            sent_vec.append(1)
        else:
            sent_vec.append(0)
```

Col 1

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 matrix = CountVectorizer(max_features=300)
3 X = matrix.fit_transform(data).toarray()
```

Limitation of BOW

- **High Dimensionality:** With a large vocabulary, the feature space becomes very large, leading to issues like the curse of dimensionality.
- **Context Ignorance:** doesn't look at the order or context of words, so lines with different meanings but the same words would be represented the same way.
- **Sparsity:** The vectors tend to be sparse (mostly zeros), which can be inefficient for computation, especially with large vocabularies.

Bag of Ngrams (BoN)

- N-grams - contiguous sequences of 'n' items from a given sample of text or speech
- Unigram – “The quick brown fox”
- Bi-gram - "The quick", "quick brown", "brown fox"
- Tri-gram - "The quick brown", "quick brown fox"

BoN in action

```
1 def generate_ngrams(text, n):
2     # Split the text into words
3     words = text.split()
4     # Create n-grams
5     ngrams = zip(*[words[i:] for i in range(n)])
6     return [" ".join(ngram) for ngram in ngrams]
7
8 # Example usage
9 text = "The quick brown fox jumps over the lazy dog"
10 bigrams = generate_ngrams(text, 2)
11 trigrams = generate_ngrams(text, 3)
12
13 print("Bigrams:", bigrams)
14 print("Trigrams:", trigrams)
```

Next classes

- Other extraction methods
- Cleaning