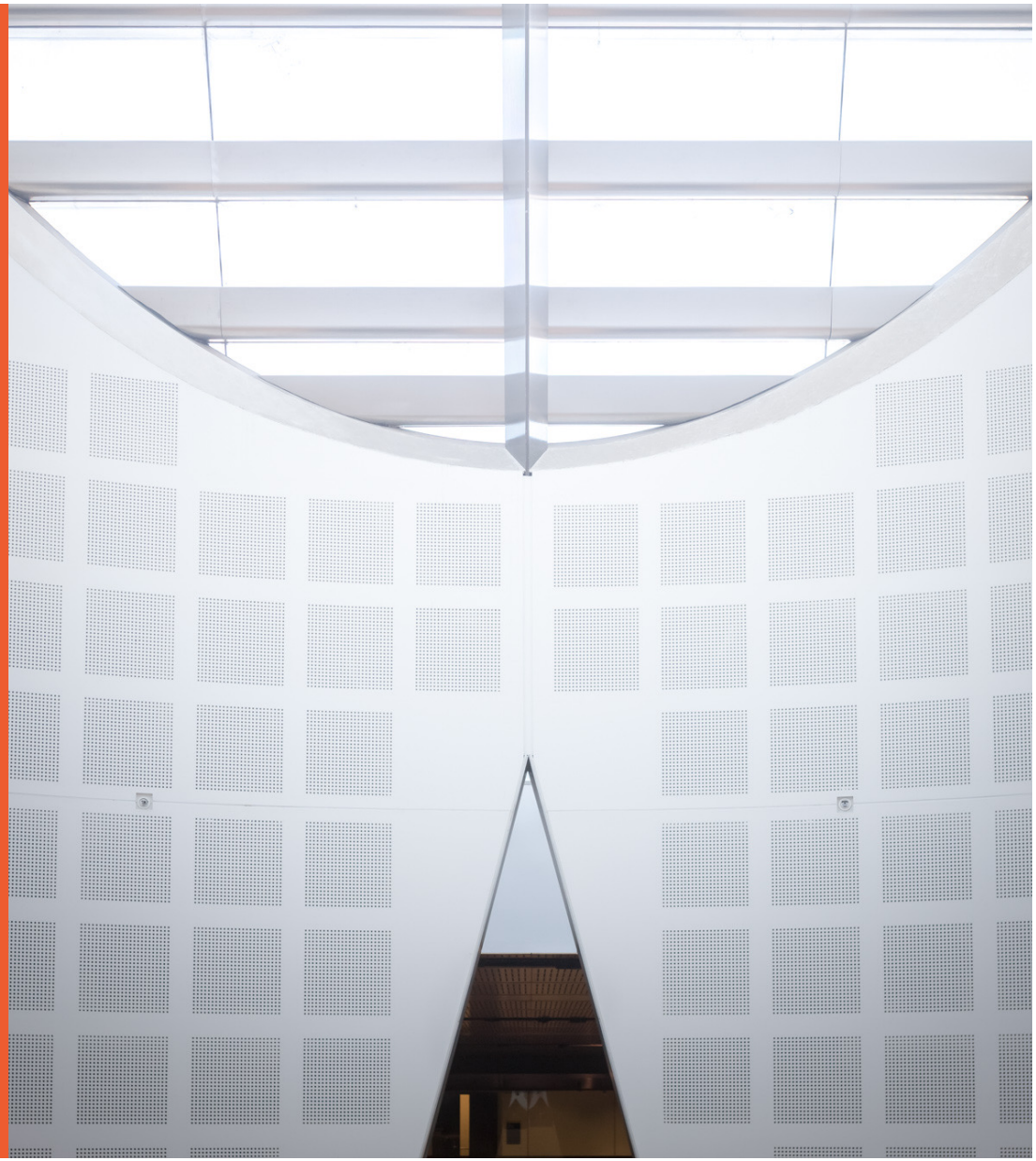
COMP5339: Data Engineering

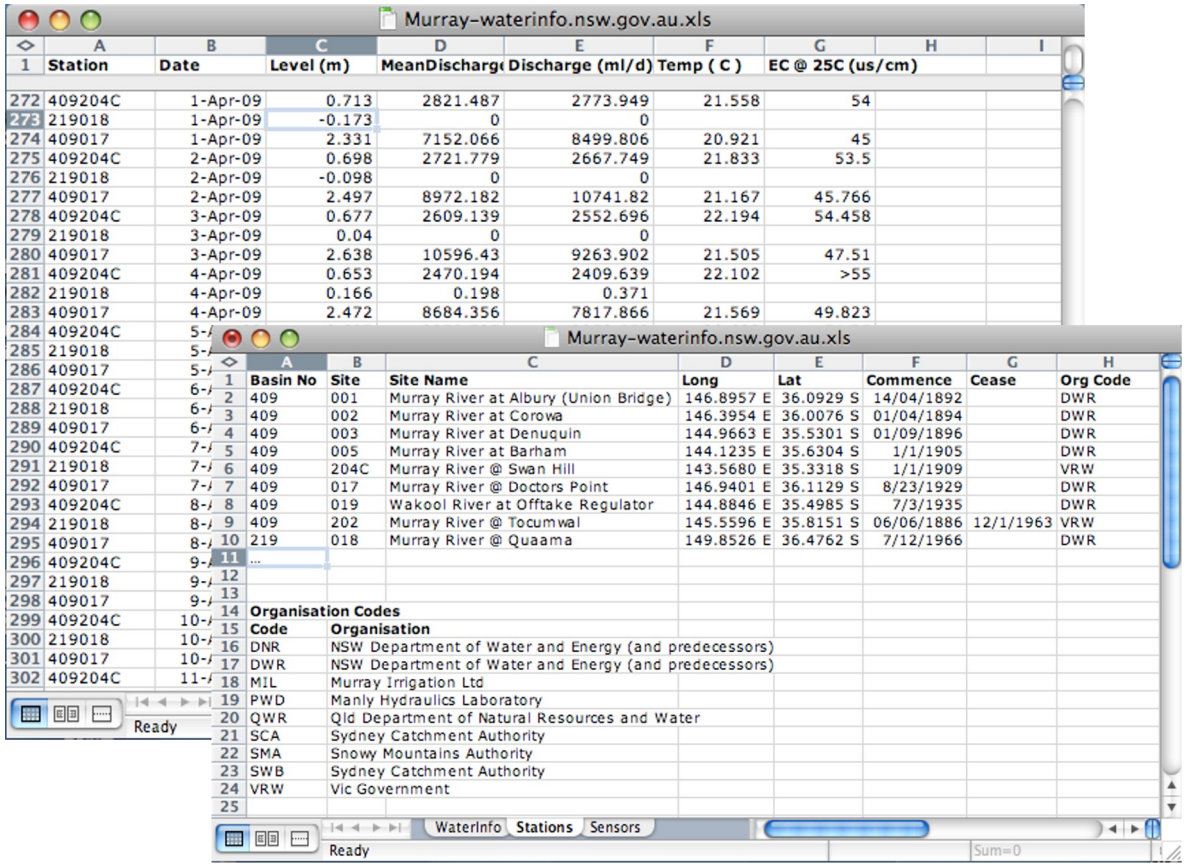
Week 8: Unstructured Data

Presented by
Uwe Roehm
School of Computer Science

THE UNIVERSITY OF SYDNEY



So far: (Semi-) Structured Data



- Data in fields
- Easily stored in databases
- E.g.:
  - Sensor data
  - Financial data
  - Click streams
  - Measurements

How to Work with Unstructured Data?

"In the history of cinematic mustaches, few have been as disgusting as that of Rye Gerhardt (Kieran Culkin), the youngest scion of North Dakota's reigning crime family and the stray spark that sets off the powder-keg second season of Fargo."

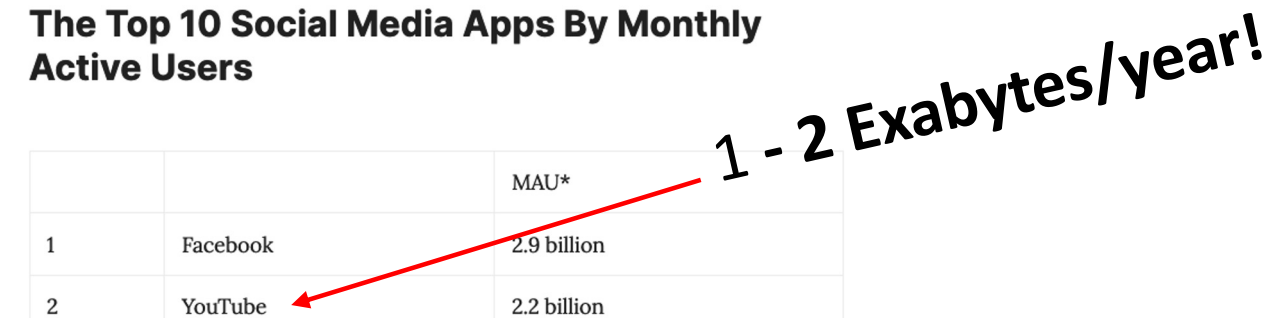
(From a Slate review of Fargo Season 2)

- 80-90% of all potentially usable business information is unstructured
- E.g.:
  - Images
  - Video
  - Email
  - Social media

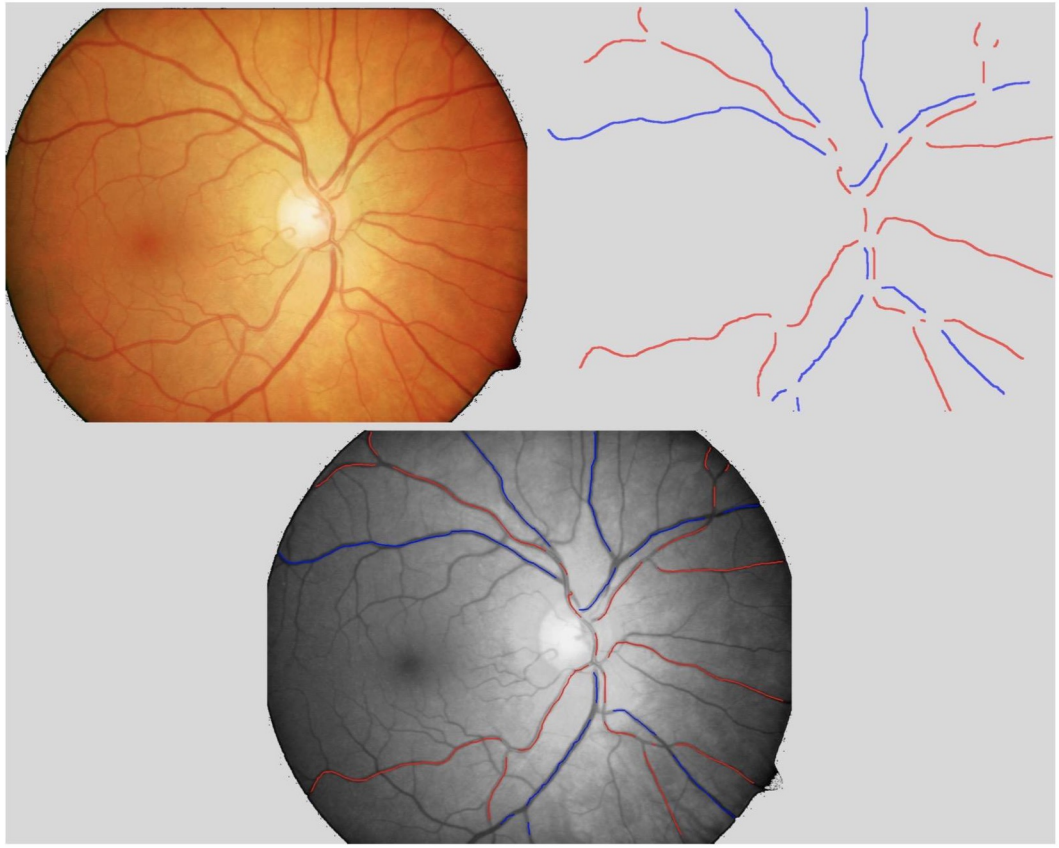
Example: Social Media

The Top 10 Social Media Apps By Monthly Active Users

		MAU*
1	Facebook	2.9 billion
2	YouTube	2.2 billion
3	WhatsApp	2 billion
4	Instagram	2 billion
5	TikTok	1 billion
6	Snapchat	538 million
7	Pinterest	444 million
8	Reddit	430 million
9	LinkedIn	250 million
10	Twitter	217 million



Example: Blood Vessel Classification in Medical Images



[Source: Kondermann and Yen: “Blood Vessel Classification into Arteries and Veins in Retinal Images“, Proceedings of SPIE - The International Society for Optical Engineering 6512:6512 · March 2007]

Classification result:
- upper left: original image
- upper right: classified vessels (red: correctly classified arteries, blue: correctly classified veins, yellow: vein classified as artery)
- bottom: superposition of classification result and original green channel

Text and Image Data

THE UNIVERSITY OF SYDNEY

Text Data

Text data usually does not have a pre-defined data model and is unstructured, but may contain dates, numbers and entities as well.

This results in ambiguities that make it more difficult to understand than data in structured databases.

NLP and Feature Extraction

- **Natural language processing (NLP):** focuses traditionally on semantic analysis or determining the intended meaning of text
  - By finding relationships between the constituent parts of language
    - for example, the letters, words, and sentences found in a text dataset.

- Important for, e.g., sentiment analysis, topic modeling, machine translation, named entity recognition, etc.

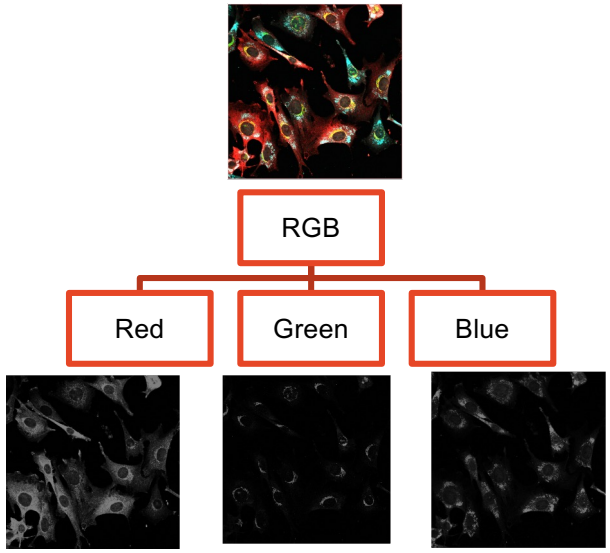
- Most conventional machine-learning techniques work on features – generally numbers that describe a document in relation to the corpus that contains it
  - Bag-of-Words, TF-IDF, word embeddings, or generic features

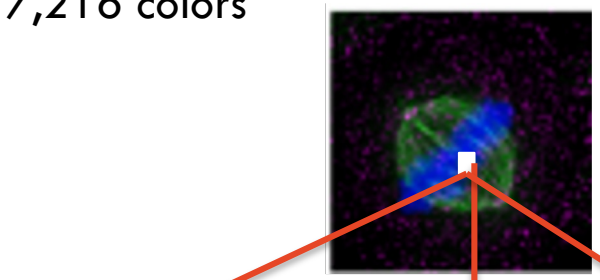
Image Data

- Images can be described as vector graphics or **raster data**
- Raster images
  - Matrix with fixed number of rows and columns
  - Digital images consist of fixed number of picture elements, called ***pixels***
  - Each pixel represents brightness of a given color
    - Color depth => different number of ***channels***
- Raster images can be created in multiple ways
  - Digital photography / video
  - Scanners
  - Image sensors in (scientific) instruments (e.g. satellite images, DNA sequencers, …)
  - Medical instruments (e.g. Xray, CET, MRT)
  - …

Types of Images

- **TrueColor or RGB Image:** Each pixel has a particular color; that color is described by the value of pixels in RGB channel (red, green and blue)
  - Number of bits per pixel (bpp) defines how many colors can be represented
    - e.g. 8 bpp = 2^8 = 256 colors or 16 bpp = 2^{16} = 65356 colors
    - Truecolor typically 3x8 bpp = 2^{24} = 16,777,216 colors





Red

111	206	250	185	64
170	253	154	67	6
61	20	185	5	91
73	203	176	254	209
58	32	190	109	19
6	134	93	18	222
178	168	145	99	191
138	175	52	231	88
163	69	250	236	24

Green

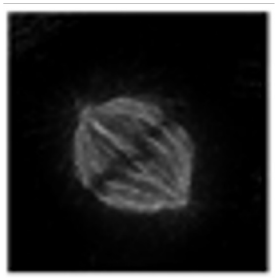
149	76	202	206	140
213	215	11	239	196
145	17	16	193	37
47	196	200	180	199
109	163	147	60	198
107	133	245	226	1
81	228	110	95	24
101	119	72	253	231
136	76	19	244	108

Blue

34	186	146	5	183
111	59	218	21	254
71	9	13	52	229
130	116	89	191	19
83	142	254	140	69
144	77	247	112	60
27	5	116	65	228
107	141	174	51	194
59	254	38	241	218

Types of Images (con't)

- Gray-scale image
  - Single channel
  - Each pixel is a shade of gray.
  - Number of bits per pixel (bpp) defines how many shades of gray can be represented
    - e.g. 8 bpp = 2^8 = 256 grays
    - or 16 bpp = 2^{16} = 65356 grays



Types of Images (cont'd)

- Binary image
  - Each pixel is just black (0) or white (1)
  - Single channel; can be 1 bpp
    (but also works with higher bpp, but just 0 and 1 used)
  - Referred as 'mask' in the image processing domain

Image Analysis

- Old way: analyse image for "features"
  - areas that match some criteria

- New way: use a Large Language Model approach
  - tools like ChatGPT can now analyse images and produce text descriptions
  - or even program code!

Summary

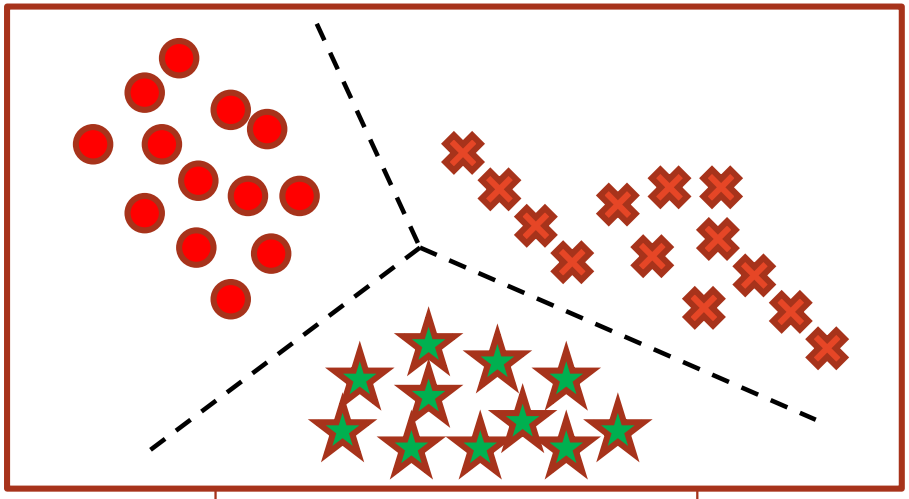
- Unstructured Data
  - Text Data
  - Images
- Before being used for data analysis, need preprocessing and feature extraction

Feature Extraction from Unstructured Data for ML

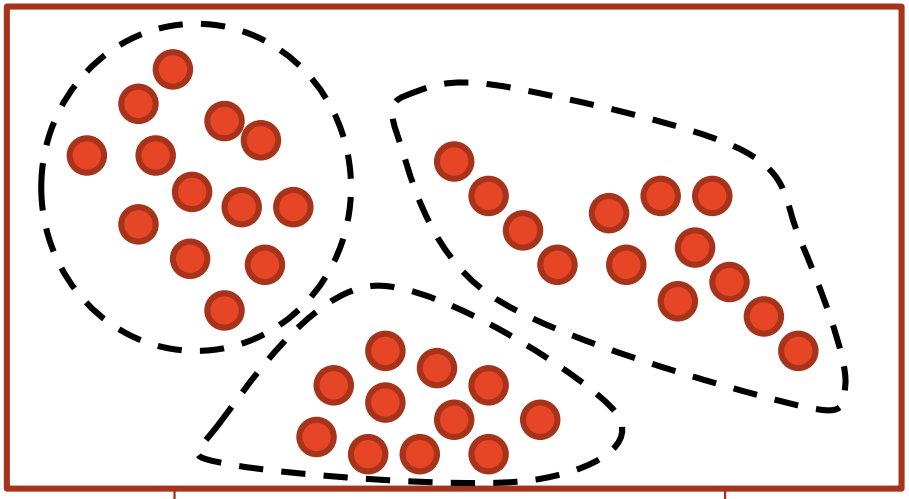
Machine Learning tasks

- Supervised learning – predict a value where ground truth is available in the training data
  - Prediction
  - Classification (categorical - discrete labels), Regression (quantitative - numeric values)
- Unsupervised learning – find patterns without ground truth in training data
  - Clustering
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction

Supervised vs Unsupervised Learning

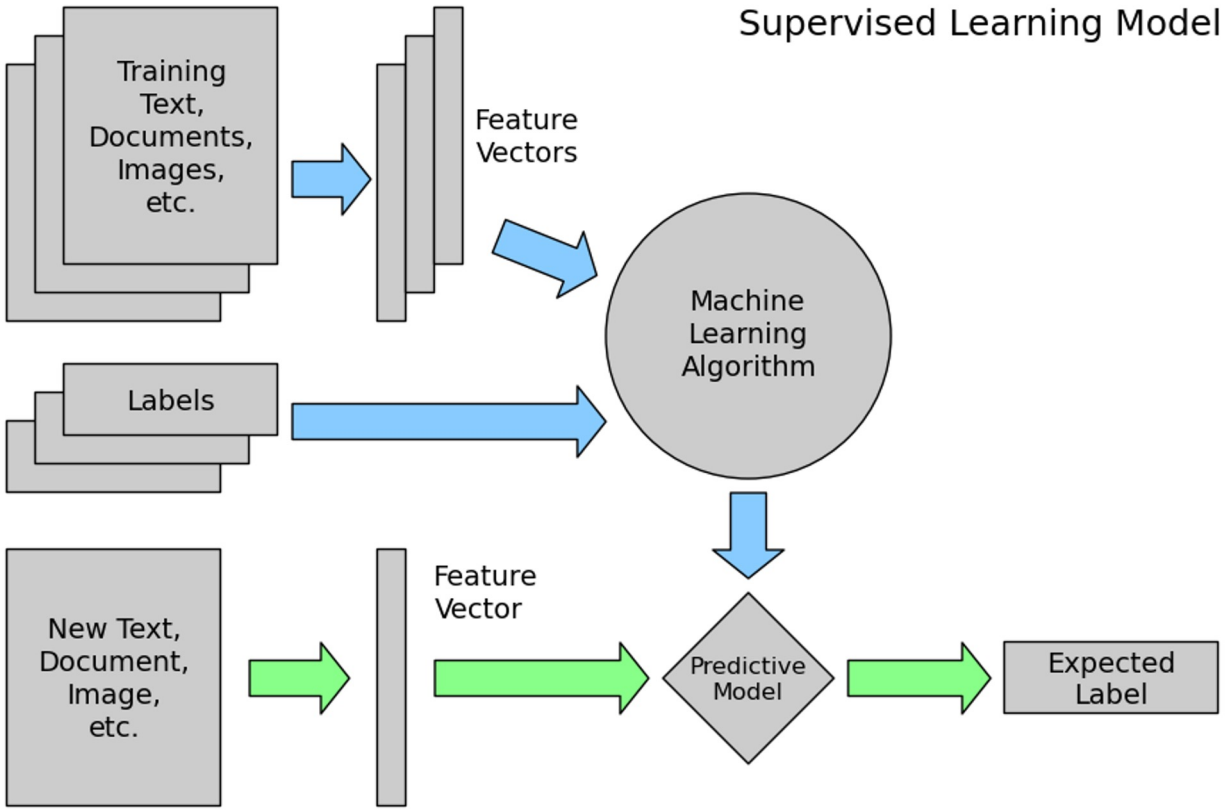


Supervised learning



Unsupervised learning

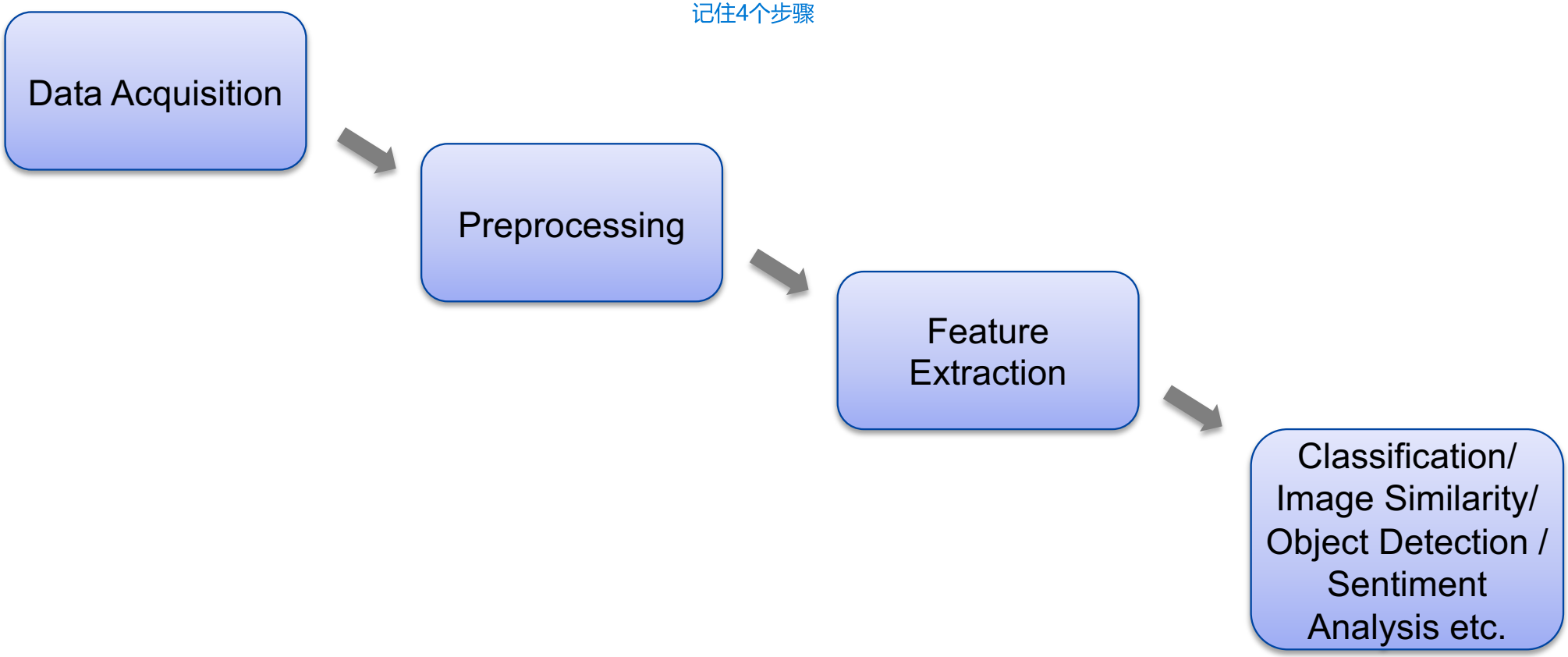
Unstructured Data for Supervised Classification



http://www.astroml.org/sklearn_tutorial/general_concepts.html#supervised-learning-model-fit-x-y

Unstructured Data Analysis Process

记住4个步骤



Data Pipeline for ML

- Importing Data
  - Optional: data transformation and cleaning
- Feature Extraction
  - Unstructured data to vectors
- Machine Learning:
  - Split data into test and training sets
  - Choose and train model(s)
  - Evaluate

Summary

- Machine learning algorithms require numerical features as input
- To apply ML to unstructured data, Feature Extraction required
- Preprocessing steps typically required too to prepare for feature extraction

Feature Extraction from Text

Motivation: Legitimate eMail – or SPAM?

Administrative Support, workshop

mail@ncoa.com.au
06/04/2019 at 08:59:15
To: Uwe Roehm <uwe.roehm@sydney.edu.au> Details

2 Attachment(s) Total 768.8 KB View

To: Human Resources

Please find attached our workshop brochure, Administrative Support.

Kindest regards,

Kathryn

Customer Service Representative

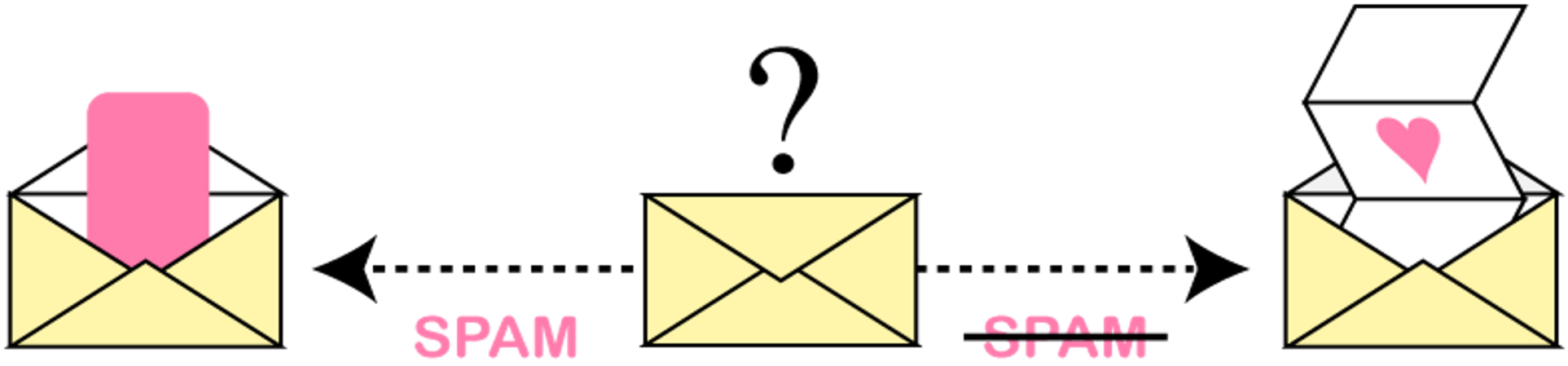
National College of Administration

Telephone: 1300 796 551

Email: [redacted]

www. collegeofadministration.com.au

Motivation Task: Spam/Not-spam Detection



http://blog.dato.com/how-to-evaluate-machine-learning-models-part-2a-classification-metrics

Modelling Spam Detection as Classification

- Input:
  - Email
  - SMS messages
  - social media pages

    …
- Predict:
  - 1 (spam)
  - 0 (not-spam)

Core Idea: Text to Feature Vectors

- **Word Embeddings:** Using models like **Word2Vec, GloVe,** or **BERT** to represent text data in a dense vector format.
- **TF-IDF:** Term Frequency-Inverse Document Frequency representations of text.

- Example TF-IDF:
  - Represent document as a multiset or bag of words
    - Tokenisation
  - Keep frequency information
  - Disregard grammar and word order
  - **Feature Vector:**
    Which words occurs how often in a given text.

http://www.python-course.eu/text_classification_python.php

Example of Tokenisation

Convert the following document into a bag of terms:

```
<blockquote
cite="http://www.brainyquote.com/quotes/quotes/r/richardpf104988.html">
For a <b>successful</b> technology, reality must take precedence over
public relations, for <u>Nature</u> cannot be fooled.
</blockquote>
```

For a **successful** technology, reality must take precedence over public relations, for Nature cannot be fooled.

rendered

a be cannot fooled for for must nature over precedence public reality relations successful take technology

raw text (lower cased)

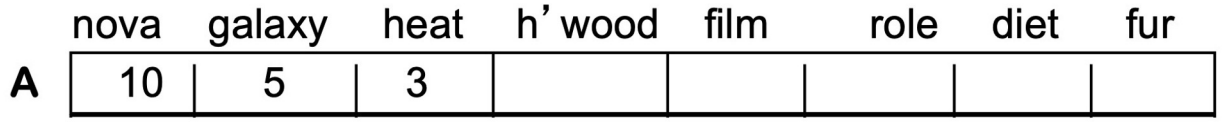
cannot fool must nature over precede public real relation success take technology

Normalisation:

- Stemming
  reduce words to their ba
- Stop-word removal

Vector Space Model

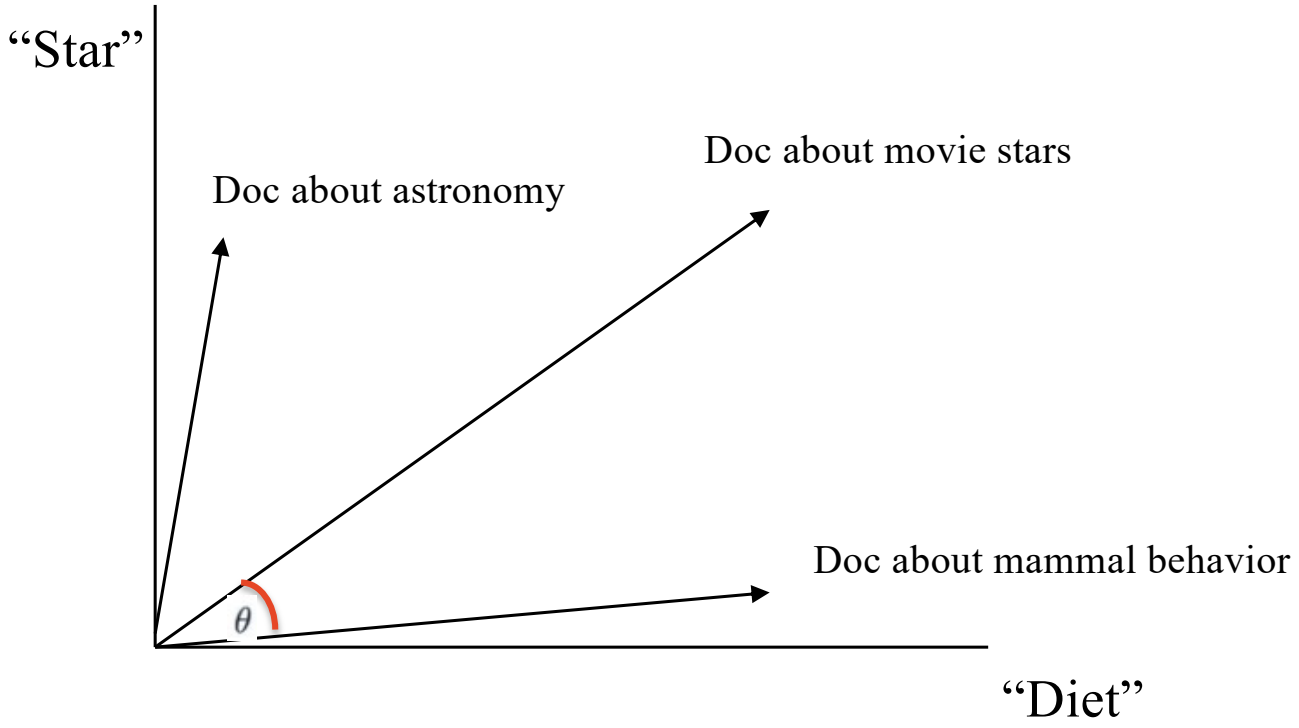
- Documents are represented as *vectors* in token space
  - Tokens are usually stems



    - "nova" occurs 10 times in text document A
      "galaxy" occurs 5 times in document A
      "heat" occurs 3 time in document A
      (blank means 0 occurrences)

  - Document vector values can be weighted by, e.g., frequency or IDF
  - **IDF: Inverse-Document-Frequency** gives less weight to terms that are common across documents *(|total docs|/|docs containing term|)*
    - ***TFIDF = TF * IDF***

Document Similarity => Vector Distance



Assumption: Documents that are close in direction and length are similar to one another.

For example: **Cosine-Similarity** $\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$

TF-Feature Extraction in Python

- Scikit-learn library provides corresponding functionality via its **CountVectorizer**
- Example:

```
from sklearn.feature_extraction.text import CountVectorizer
from pprint import pprint
corpus = [ 'This is the first document.',
           'This is the second second document.',
           'And the third one.',
           'Is this the first document?', ... ]
vectorizer = CountVectorizer()
matrix     = vectorizer.fit_transform(corpus)
pprint(matrix)
```

- https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
- See also: https://adataanalyst.com/scikit-learn/countvectorizer-sklearn-example/

Feature Extraction in Python (cont'd)

- **CountVectorizer** can be configured in quite some detail
  - By default, CounterVectorizer does tokenization for single words of minimum length 2
    - Change to also consider bigrams (terms consisting of 2 words):

```
vectorizer = CountVectorizer(ngram_range=(1,2))
```

  - Convert input text to lower case; also ignore certain accents in text:

```
vectorizer = CountVectorizer(lowercase=True, strip_accents='ascii')
```

  - Use indicator features (0 or 1) rather than term frequencies

```
vectorizer = CountVectorizer(binary=True)
```

  - Specify a list of *stop words* that get ignored

```
vectorizer = CountVectorizer(stop_words=['the','a'])
```

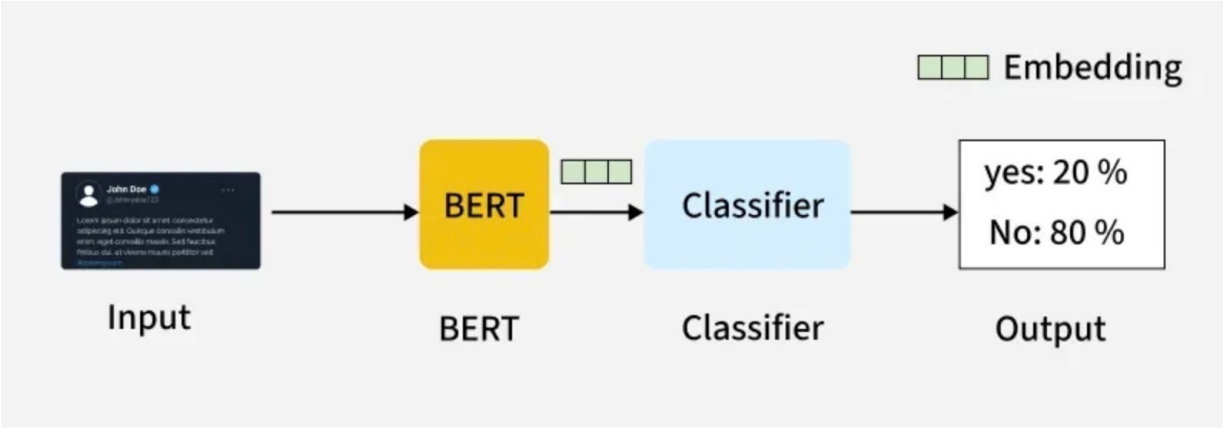
  - Only keep features within a certain document frequency range

```
vectorizer = CountVectorizer(min_df=0.1, max_df=0.5)
```

- Example: `CountVectorizer(lowercase=True,strip_accents='ascii',binary=True)`

BERT – Bidirectional Encoder Representations from Transformers

- BERT: an open-source machine learning framework for NLP
  - pre-trained transformer-based neural network (encoder-only architecture)
  - can be fine-tuned on labeled data for specific NLP tasks
  - uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially
  - returns contextual word embeddings (vectors)



Source: https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/

Example: BERT Tokenizer in Python

Text still needs to be pre-processed and tokenized when using BERT:

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

```
# Encode a sample input sentence
sample_sentence = 'I liked this movie'
token_ids = tokenizer.encode(sample_sentence, return_tensors='np')
print(f'Token IDs: {token_ids}')

# Convert the token IDs back to tokens to reveal the special token
tokens = tokenizer.convert_ids_to_tokens(token_ids)
print(f'Tokens   : {tokens}')
```

```
Token IDs: [ 101 1045 4669 2023 3185  102]
Tokens   : ['[CLS]', 'i', 'liked', 'this', 'movie', '[SEP]']
```


Source: https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11/

BERT vs GPT

	BERT	GPT
Architecture	Bidirectional; predicts masked words based on left, right context.	Unidirectional; predicts next word given preceding context.
Pre-training Objectives	BERT is pre-trained using a masked language model objective and next sentence prediction.	GPT is pre-trained using Next word prediction only.
Context Understanding	Strong at understanding and analyzing text.	Strong in generating coherent and contextually relevant text.
Tasks and Use Cases	Commonly used in tasks like text classification, NER, sentiment analysis, and QA	Applied to tasks like text generation, chat, summarization, etc.
Fine-tuning vs Few-Shot Learning	Fine-tuning with labeled data to adapt its pre-trained representations to the task at hand.	GPT is designed to perform few-shot or zero-shot learning, where it can generalize with minimal task-specific data.

Source: https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/

Other Feature Extraction Tools

BERT:

https://huggingface.co/docs/transformers/model_doc/bert

Word2Vec: Open Source Tool by Google

https://code.google.com/archive/p/word2vec/

spaCy:

https://spacy.io/usage/spacy-101

Summary

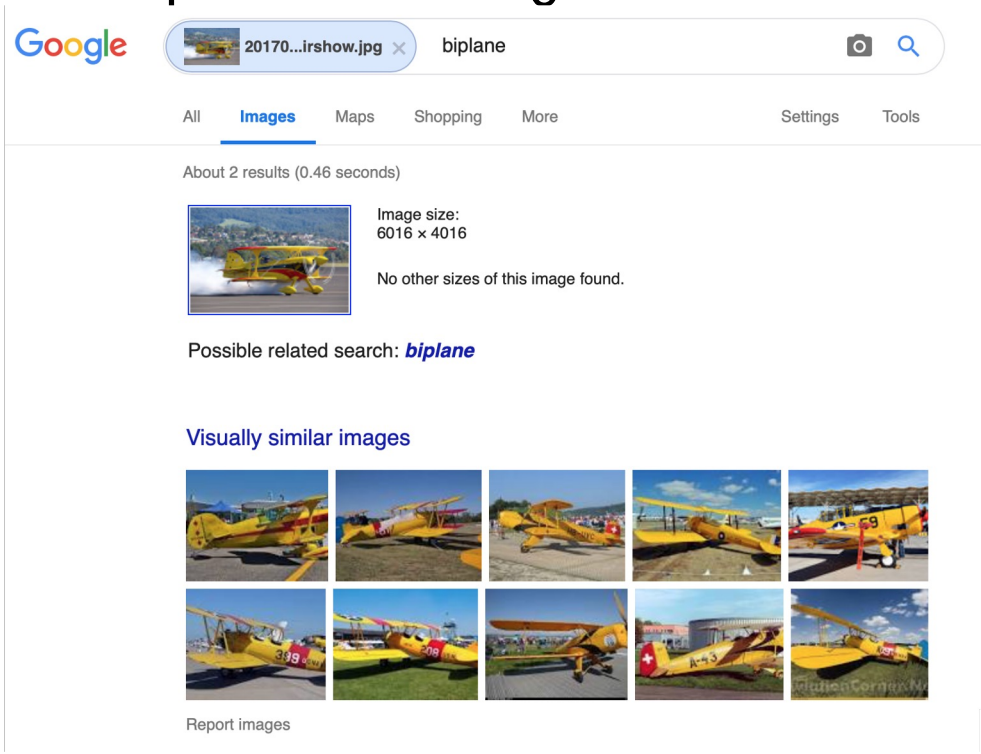
- Feature Extraction from Text Data
  - Tokenization, Normalization
  - **Word Embedding** into a vector space
    - either 'manually' using TF-IDF for bag-of-words model
    - or via neural-network-based encoders such as Word2Vec (word embeddings) or BERT (contextual word embeddings)
- Potential Applications:
  - Text Classification
  - Named Entity Recognition (NER)
  - Sentiment Analysis
  - …

Feature Extraction from Images

THE UNIVERSITY OF SYDNEY

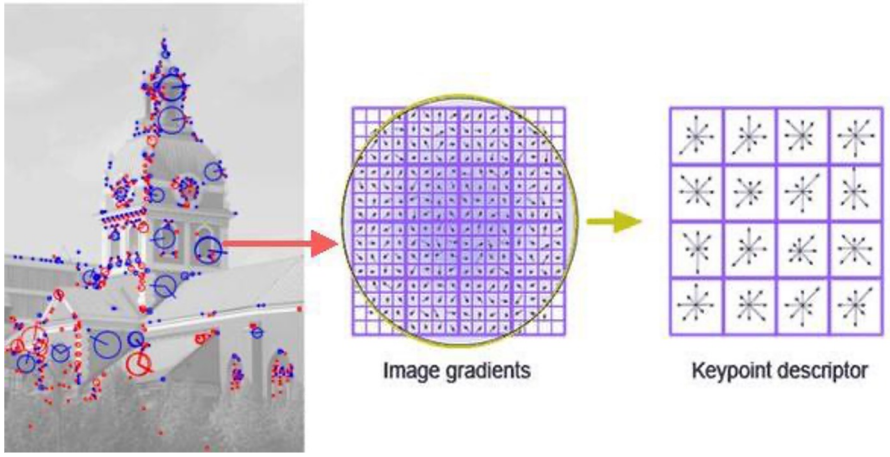
Image Search Example: Google

- Example: Image search with uploaded photo or with textual description
  - Search engine is capable of finding "visual similar images" – how?



Feature Extraction

- Image **feature:**
  - Simple pattern within image
  - Color information
  - Metadata

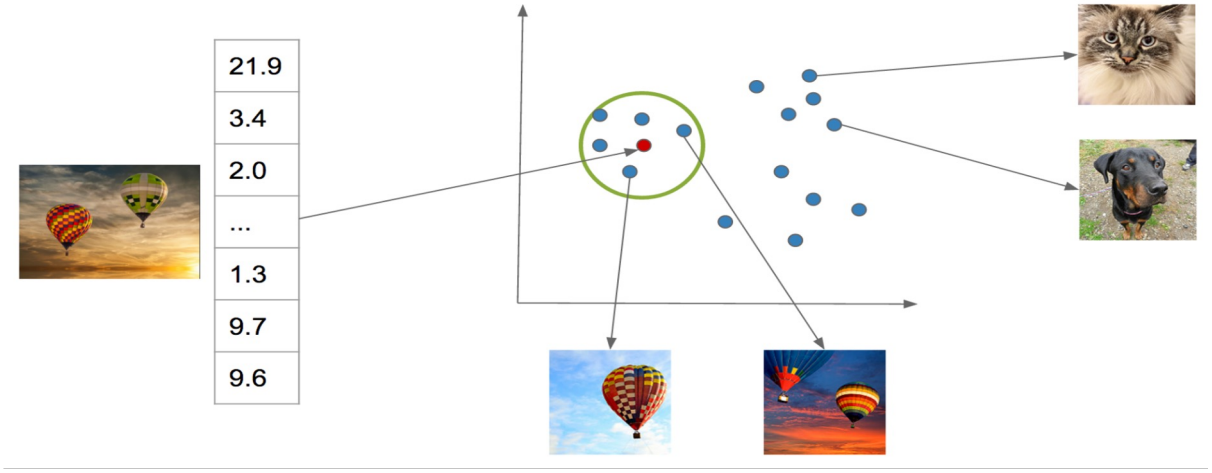


- Core idea of using *features*: transform visual information into the vector space
  - gives possibility to perform mathematical operations on them, for example finding similar vectors

[Image Source: https://medium.com/machine-learning-world/tagged/computer-vision]

Image Similarity Search

- Core idea: Extract **feature vector** from images and build **index** of them
- Search:
  - Convert query image into feature vector by the same method
  - Compare feature vectors: close vectors => visually similar images



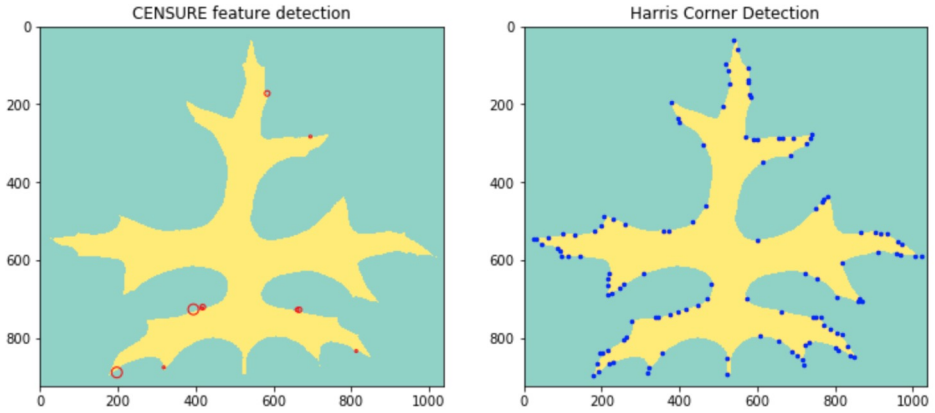
[Image Source: http://code.flickr.net/2017/03/07/introducing-similarity-search-at-flickr/]

How to extract features from images?

- Two approaches
  - Image descriptors (white box algorithms)
  - Neural networks (black box algorithms)
- Many (white box) algorithms for feature extraction are typically based on **image gradient:** change of intensity or color of image
  - But other features are possible too
- Support in Python
  - E.g. Scikit Learn's **skimage** library
  - Algorithms are beyond the scope of this lecture

skimage Example

```
# check a few scikitlearn image feature extractions, if they can help us

from skimage.feature import corner_harris, corner_subpix, corner_peaks, CENSURE

detector = CENSURE()
detector.detect(img)

coords = corner_peaks(corner_harris(img), min_distance=5)
coords_subpix = corner_subpix(img, coords, window_size=13)

plt.subplot(121)
plt.title('CENSURE feature detection')
plt.imshow(img, cmap='Set3')
plt.scatter(detector.keypoints[:, 1], detector.keypoints[:, 0],
              2 ** detector.scales, facecolors='none', edgecolors='r')

plt.subplot(122)
plt.title('Harris Corner Detection')
plt.imshow(img, cmap='Set3')  # show me the leaf
plt.plot(coords[:, 1], coords[:, 0], '.b', markersize=5)
plt.show()
```



[Source: https://www.kaggle.com/lorinc/feature-extraction-from-images]

Image Data Analysis Outcomes

- After we have extracted ROIs and/or Features from images, what can we do with those features?
  - Classification
  - Image Similarity Search
  - Object Detection
  - Intensity-based Analysis
  - Combining results from multiple images for conversion to raw data
  - etc.

Example:
DNA Sequencing

Primary Data Analysis

Summary

- Feature Extraction from Image Data
  - Preprocessing such as noise reduction, morphological operations, thresholding
  - White box (e.g. based on image gradients) vs black box (NN) algorithms
  - High-Dimensional Feature Space
- Language support, e.g. in Python
- Tool support: LLMs

Meta Data Analysis

THE UNIVERSITY OF SYDNEY

Meta Data Analysis

- Unstructured data, especially Digital images (photos) and videos, but also texts (e.g. PDFs) have metadata associated
  - Especially digital photography
  - standard by iptc.org
  - Several standard format, such as EXIF or XMP

- Typically includes
  - when captured, how, which camera/instrument, which settings
  - GPS location
  - author, copyright
  - ….

- Tools access metadata
  - GUI tools such as Photoshop, Lightroom, Preview, …
  - Web services such as https://www.metadata2go.com
  - Command line: exiftool

- exiftool (https://exiftool.org/)
  - supports a wide variety of image and video file formats (and more) (https://exiftool.org/#supported)
  - read/write access to meta-data of most formats
  - can be scripted

Image Metadata Analysis

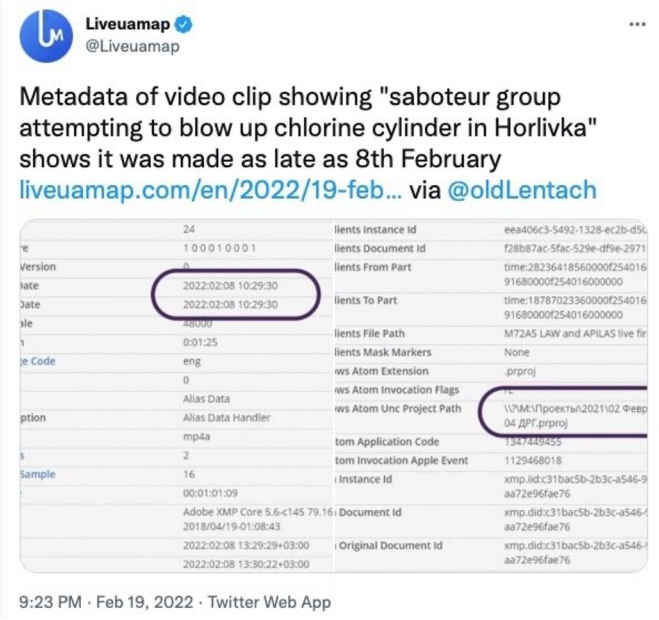
- Remember: The meta-data of one user is data for another…
- Image Metadata in the form of key-value pairs
  - Geo-location analysis of images
  - Extraction of meta-data such as author and description
  - Possibly also keywords
- Without Metadata: Analysing the actual image content is required, which is more complex and resource-intensive.

Metadata Doesn't Lie (mostly)

- Analyzing the metadata in videos and images – including timestamps, geo-location, etc. – has become an important tool in gathering intelligence and fighting disinformation
  - https://pureinsights.com/blog/2022/metadata-and-the-information-war-in-ukraine/

Duration	0:02:17
Preferred Rate	1
Preferred Volume	100.00%
Preview Time	0 s
Preview Duration	0 s
Poster Time	0 s
Selection Time	0 s
Selection Duration	0 s
Current Time	0 s
Next Track Id	3
Track Header Version	0
Track Create Date	2022:02:16 17:07:24
Track Modify Date	2022:02:16 17:07:24
Track Id	1
Track Duration	0:02:17
Track Layer	0
Track Volume	0.00%
Image Width	1920
Image Height	1080

[https://www.axios.com/2022/02/18/telegram-ukraine-russia-separatists-evacuation]



[https://twitter.com/Liveuamap/status/1494980844024385540]

exiftool Examples

- Read all meta-data of a file 了解每个是干嘛的

  exiftool *filename*
- Read specific meta-data such as CreateDate from an image

  exiftool –CreateDate *filename*
- Read all GPS related data from a file

  exiftool "-GPS*" *filename*
- Extract common EXIF metadata from multiple images into a CSV file

  exiftool –**common** –**csv**=metadata.csv *.jpg
  - or for JSON output: **exiftool** –**common** –**json**=metadata.json *.jpg
- Also supports other document file formats such as Word or PDF:

  exiftool –Author example.pdf

- Can also modify or rename, e.g.: rename JPG images based on when taken – and adjust file creation time accordingly – recursively for all files in current directory

  exiftool -d %Y%m%d "-filename<datetimeoriginal" "-filemodifydate<datetimeoriginal#" **-ext** jpg **-r** .

Summary

- Many digital assets, in particular photos and videos, contain a large variety of meta-data
- Both temporal and spatial data
- Many tools available to inspect / access meta-data
- Exiftool to easily and automatically extract meta-data using scripts