A photograph of a modern building's exterior featuring a complex, angular facade made of light-colored panels and glass windows. The building is set against a bright blue sky with wispy white clouds.

MIS710 Machine Learning in Business

Topic 10: Sentiment Analysis using Machine Learning

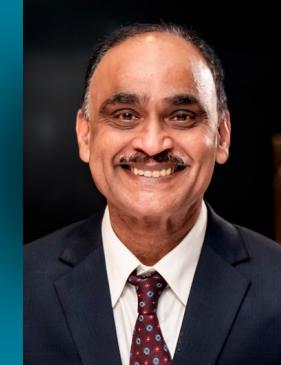
Associate Professor Lemai Nguyen





Unlocking the Power of AI: Strategy, Platforms, and Practice

Tuesday 23rd Sep 2024, 12.00pm AEST, 7.30am IST



Prof Rajeev Sharma
Head of
Department,
Information
Systems and
Business Analytics

Alan Longmuir
Delivery Manager,
Digital Solutions
Digital Services

Lizy Lal
Senior Manager,
Emergent
Technologies
Digital Services

Dr Rituraj Kunwar
AI Specialist,
Emergent
Technologies
Digital Services

Dr Mohsen Laali
AI Specialist,
Emergent
Technologies
Digital Services

Prof Raju Varanasi
Professor of Practice
Deakin Business
School

**A/Prof Lemai
Nguyen**
Information Systems
and Business
Analytics



Mastering Digital Business Analysis: Digital Sustainable Futures and Career Insights

Tuesday, 24th Sep 2024, 3.00m AEST



**Assoc./Prof. Lemai
Nguyen**
Information Systems
and Business
Analytics



Sureikha Ratnatunga,
Senior Business Analyst
Digital Learning, Deakin
Learning Futures



Emre Denis
CEO at GYLD Agency,
New Zealand

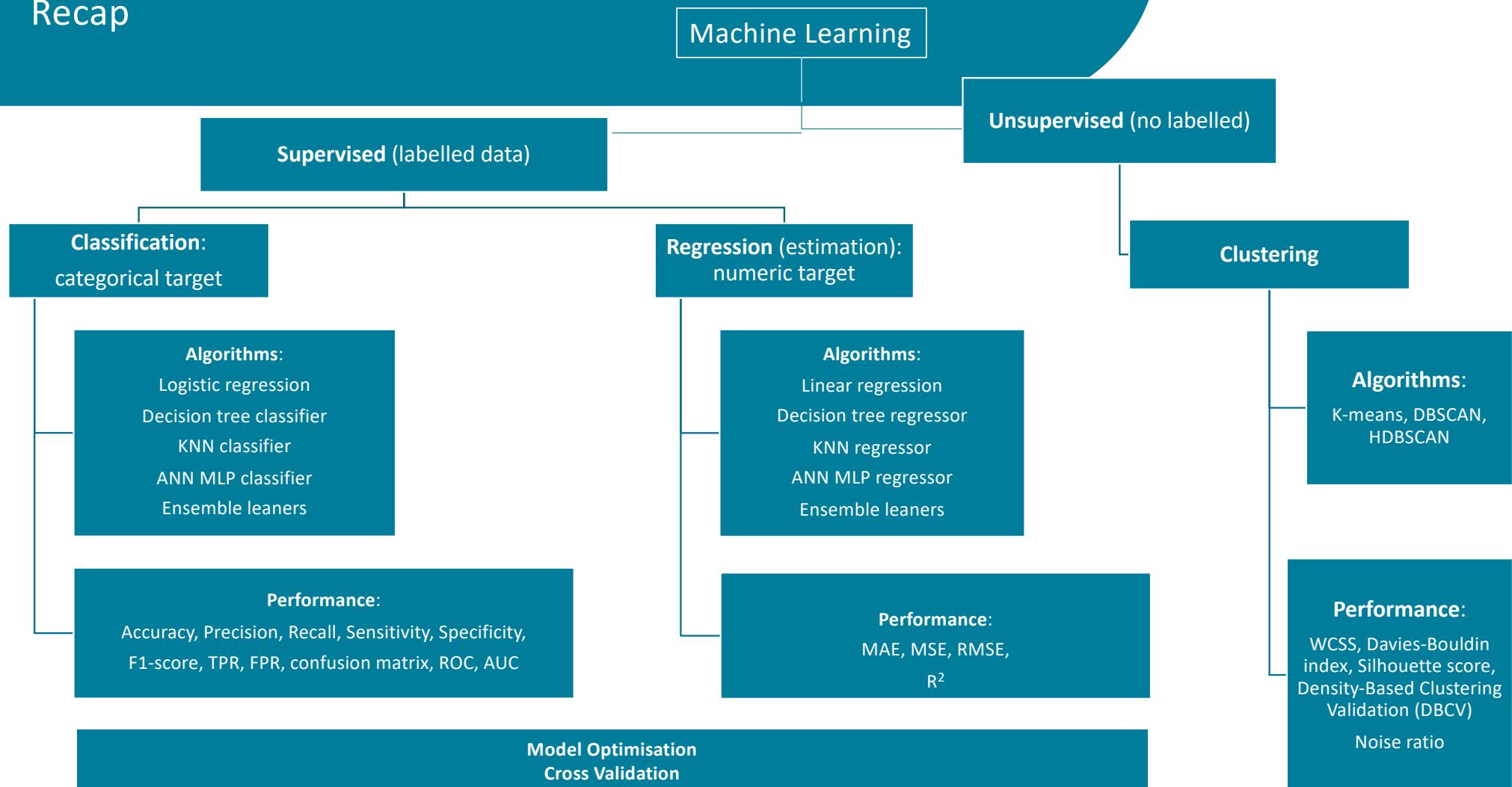


Denley C. Rebello
IT Team Leader – IT
Infrastructure &
Services
Neami National

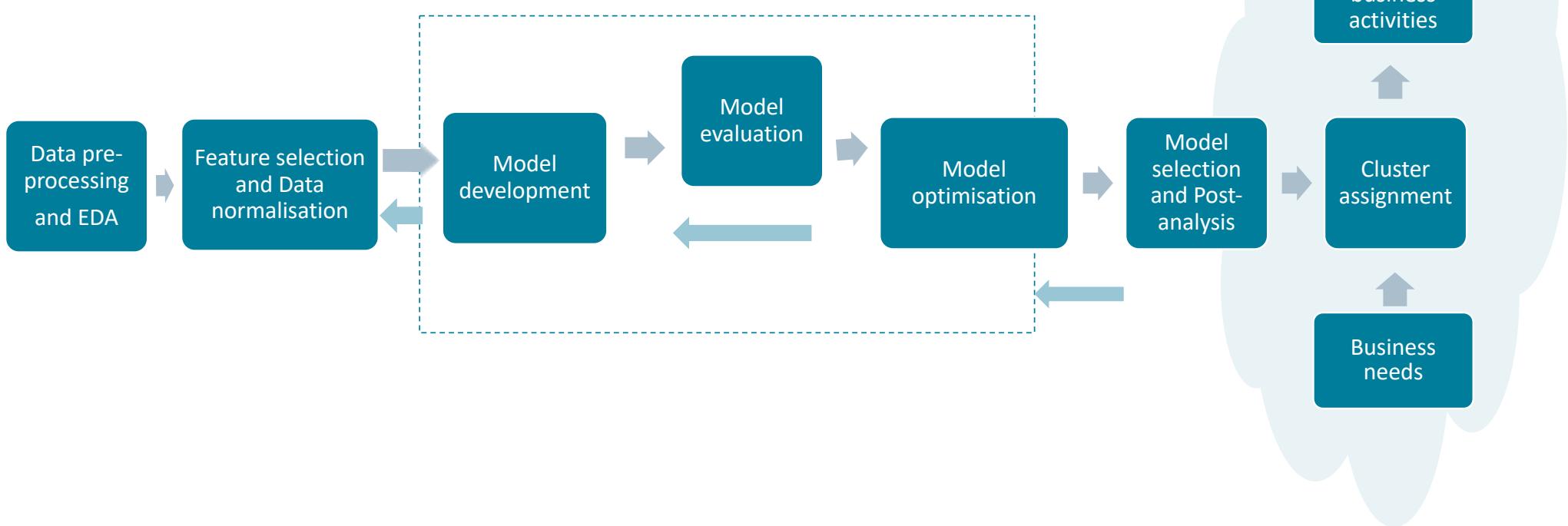


Fuzail Ahmed
Business Analyst &
Project Manager
Light and Wonder

Recap



Recap: the clustering modelling process



A2 consultation sessions



Learning
Analytics for
Primary Schools

A2
Case Study
A2 Tasks
A2 Deliveries

- **Late Penalty:** 3 marks per day.
- Submit the **correct files**.
- Uphold **academic Integrity**

Week 10:

- Friday 20th Sep 3.00-4.00pm AEST i.e., 10.30-11.30am IST: Lemai - How to write a business report? Zoom

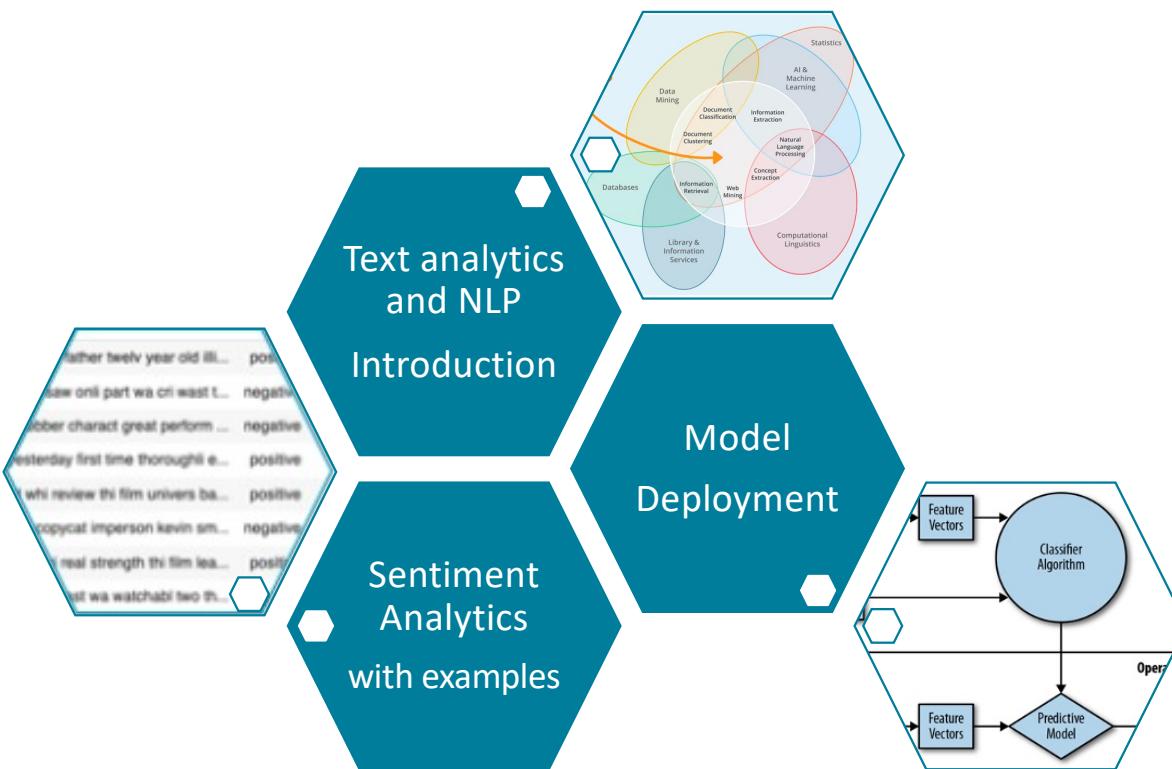
Week 11:

- Tuesday 24th Sep 3.00-4.00pm AEST, Burwood B3.17, Dat Le
- Wednesday 25th Sep 7:30-8:30pm, 3.00-4.00pm IST, Zoom Emran
- Thursday 26th Sep 3.00-4.00pm AEST, Burwood B3.17, Thuc

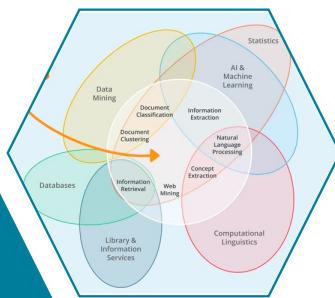
Week 12 - study week – Zoom only

- Monday 30th Sep 3.00-4.00pm AEST i.e., 10.30-11.30am IST, Zoom, Dat Le
- Tuesday 1st Oct 6.00-7.00pm i.e., 1.30-2.30pm IST, zoom, Durgesh
- Wednesday 2nd Oct 6.00-7.00pm i.e., 1.30-2.30pm IST, zoom, Emran

Participate in the labs and A2 forum



Text analytics and NLP Introduction



- Understanding Text Data
- Understanding the Text analytics space
- NLP principles and overall workflow

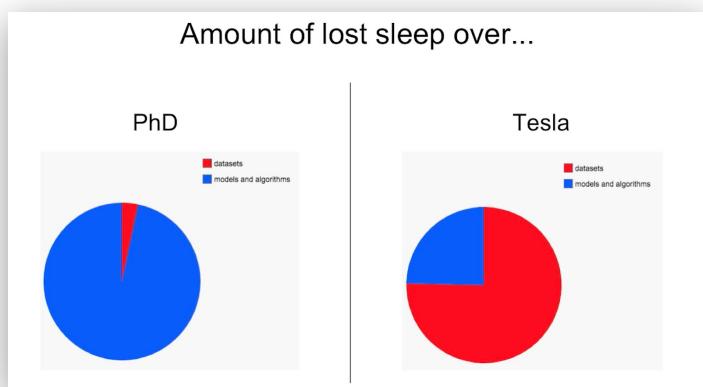
Data

Data in Labs

- More or less clean
- Often contained
- Historical
- Structured/formatted

Data in Practice

- Massive amount of data, unbounded
- Structured and Unstructured
- Data quality issues: messy, noisy, missing, real-time and constantly shifting, imbalance, outdated, inaccurate, no/expensive labels
- Data acquisition and negotiation
- Privacy and data regulations
- Inclusion, equity, fairness and AI ethics
- **Most companies focus on Data**



Andrej Karpathy, head of AI at Tesla, in Chip Huyen's ML in production, 2023



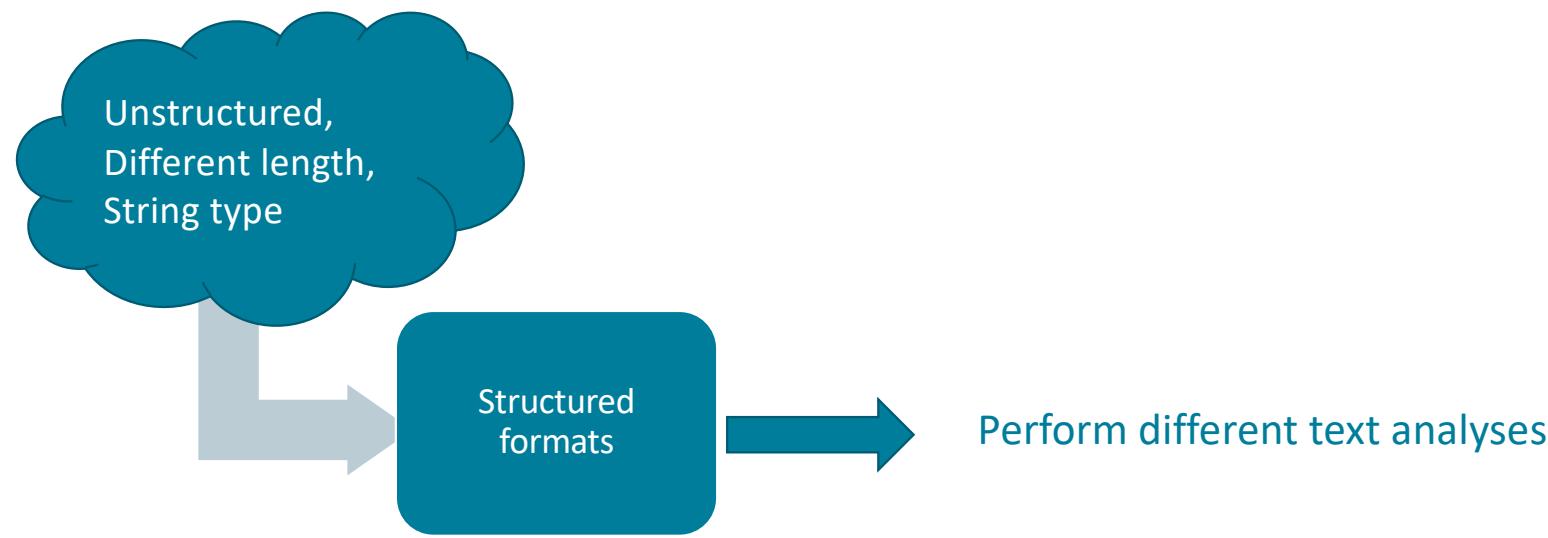
“a majority of data (80% to 90%, according to multiple analyst estimates) is unstructured information like text, video, audio, web server logs, social media, and more.”

Tam, Harbert, 2021 <https://mitsloan.mit.edu/ideas-made-to-matter/tapping-power-unstructured-data>

- More challenging to manage and analyse
- Opportunities to gain insights

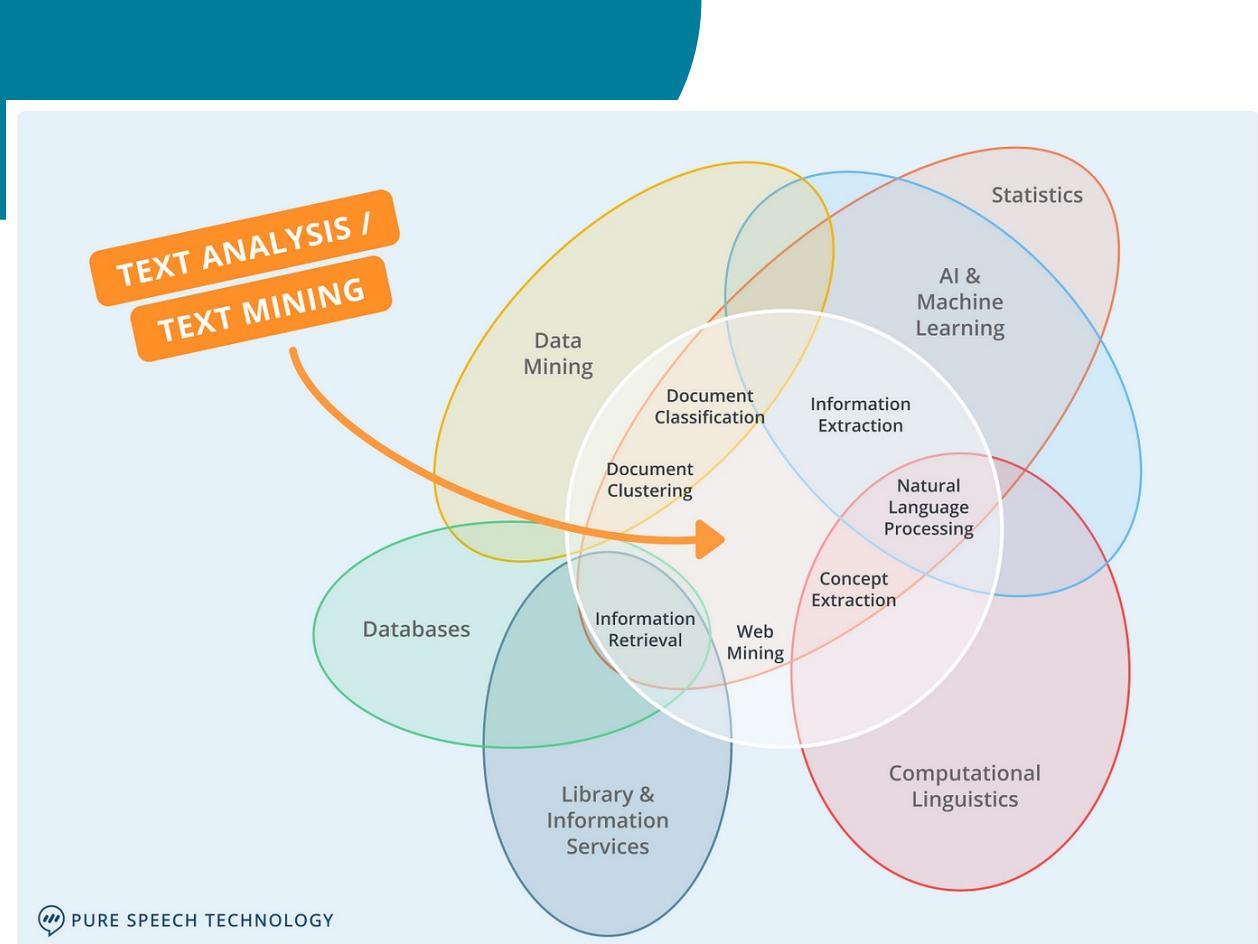
Text can be

- Categorical data: e.g., “doctor”, “patient”, “nurse” etc
- Structured string data: first name surname: e.g., Alan Turing
- Free strings that can be semantically mapped to categories: e.g., “my GP printed out a prescription and explained it to me”
- Any free text data: tweets, reviews, Facebook posts, books, news, any documents



Text analysis /NLP

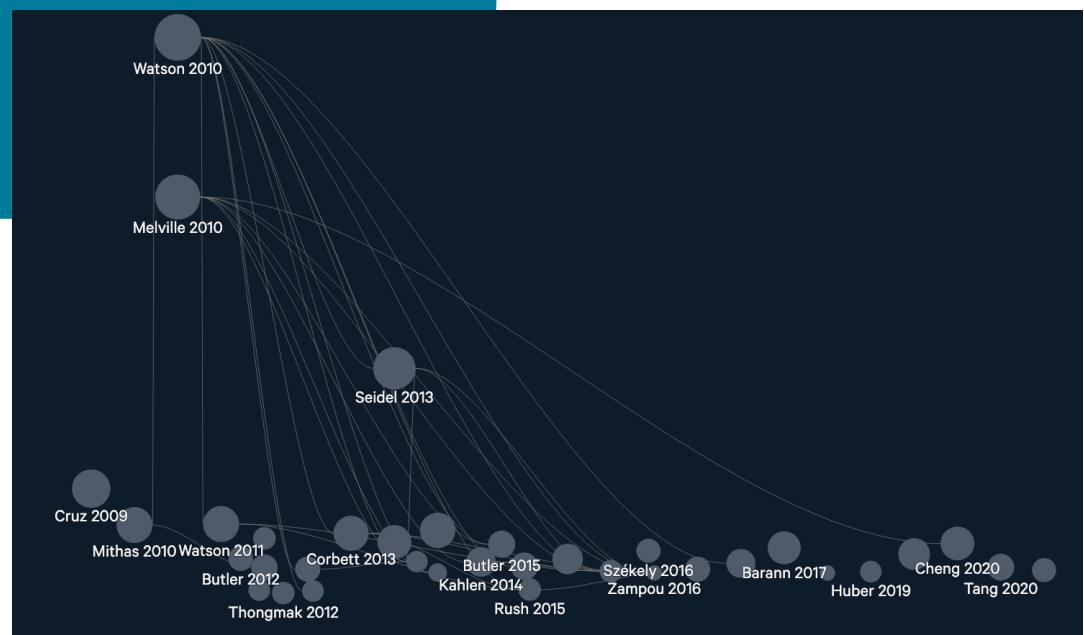
- **Multiple fields**
 - Text analysis
 - NLP
 - Others



<https://towardsdatascience.com/a-guide-to-text-analysis-text-analytics-text-mining-f62df7b78747>

Text analysis /NLP

- Multiple fields
 - **Text analysis**
 - NLP
 - Others



Text Analysis: analysing large collections of unstructured and semi-structured text data for valuable insights, trends and patterns.

- frequency of words, length of sentence, and presence or absence of words, information retrieval (keyword extraction), visualisation, stats analyses.
- semantics may not be considered.

Qualitative data analysis: data coding, Nvivo

Nguyen et al, SLR SET 2024

<https://app.litmaps.com/shared/b54468f1-1d93-4782-bd56-e5c6f48704a1>

Text analysis /NLP

- Multiple fields
 - Text analysis
 - **NLP**
 - Others

Natural language processing is a subset of text analysis

- in the interaction of text analysis, computational linguistics, AI and machine learning.
- aims to understand, and replicate human speech
- grammatical structures and semantics are analysed
- LLM and ML/DL are approaches (among others)

1	What is known about the topic ?	NEUTRAL
2	Studies reporting nurses ' and midwives ' experiences of using EMR are scarce and mostly based in countries where wi	NEGATIVE
3	What does this paper add ?	NEUTRAL
4	Nurses and midwives perceive benefits of using an EMR relatively soon after implementation in terms of their workflow	POSITIVE
5	However , in the first year after EMR implementation , nurses and midwives experience some negative effects on work	NEGATIVE
6	The effects on clinical workflow are further compounded by EMR downtime (scheduled and unscheduled) and hybrid s	NEUTRAL
7	What are the implications for practitioners ?	NEUTRAL
8	In countries like Australia , whole - of - service , simultaneous implementation of EMR systems using best - available s	NEUTRAL
9	In these circumstances , nurses and midwives may initially experience increased workload and frustration .	NEGATIVE
10	Ongoing training and technical support should be provided to nurses and midwives for several months following impl	NEUTRAL
11	Including nurses and midwives in the design of theEMRwill result in better alignment with their specific workflow , thus	POSITIVE

<https://corenlp.run/>

<https://towardsdatascience.com/natural-language-processing-using-stanfords-corenlp-d9e64c1e1024>

NLP common tasks

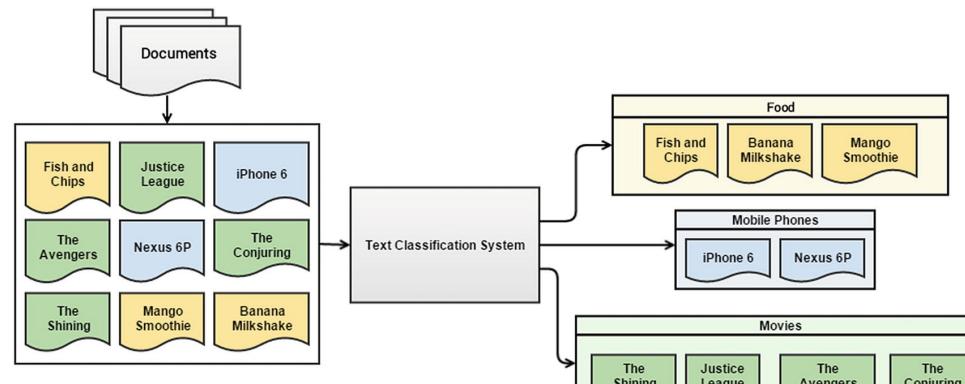
- Sentiment analysis
- Machine translation
- Named entity recognition (NER)
- Question answering
- Text summarisation
- Text classification
- Text clustering

For example:

https://demo.johnsnowlabs.com/healthcare/MEDICAL_LLM/

Named Entity Recognition:					
	PERSON	ORGANIZATION	COUNTRY		
1	THEME Co-creating Sustainable Digital Futures Track Chairs Valerie Graf - Drasch , University of Hohenheim , Germany .				
		EMAIL	PERSON	ORGANIZATION	COUNTRY
2	E-mail : valerie.graf-drasch@uni-hohenheim.de Gilbert Fridgen , University of Luxembourg , Luxembourg .				
		EMAIL	PERSON	ORGANIZATION	COUNTRY
3	E-mail : gilbert.fridgen@uni.lu Richard T. Watson , University of Georgia , US .				
		EMAIL			
4	E-mail : rwatson@terry.uga.edu				

<https://corenlp.run>



Sarkar 2019

Large Language Modelling

Type of artificial intelligence (AI) that is trained to predict the probability of the next word in a sequence of words or sentences based on the preceding context.

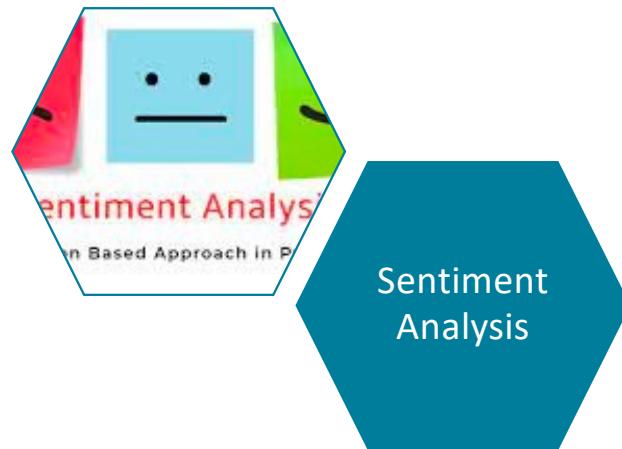
- Machine translation, text generation, and question-answering systems.
- Based on deep learning architectures, such as transformers.
- Trained on very large corpora

- GPT-3's dataset (OpenAI): 0.57 trillion tokens.
- Gopher's dataset (DeepMind): 1 trillion tokens
- RedPajama (Together): 1.2 trillion tokens
- LLaMa's dataset (Meta): 1.4 trillion tokens

Updated from <https://huyenchip.com/2023/05/02/rlhf.html>

GPT-3 with around 175 billion parameters





Sentiment analysis

- Sentiment analysis is a Natural Language Processing (NLP) task.
- Also called opinion mining
- Aims to determine the sentiment or emotional tone behind a piece of text.
- Classifying text into categories such as positive, negative, or neutral, based on the emotions, opinions, or attitudes expressed.

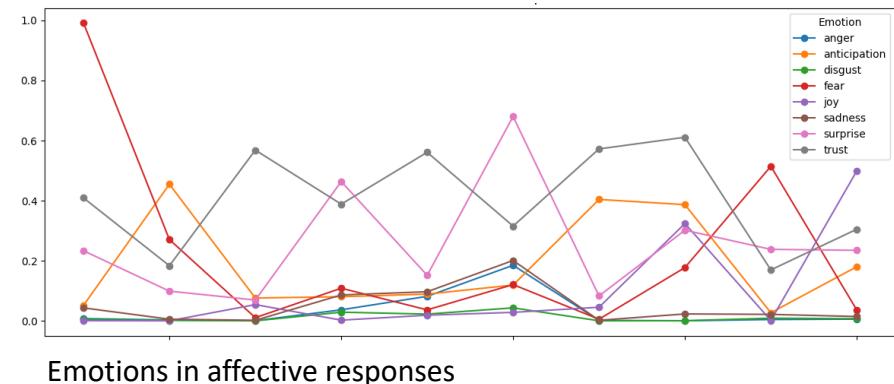
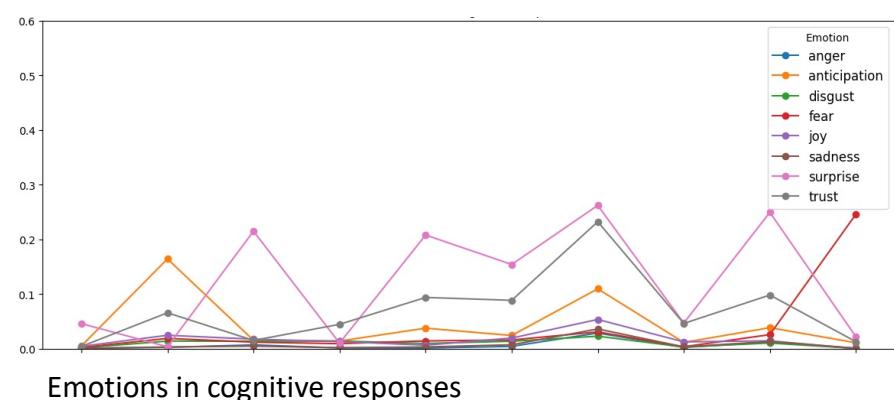
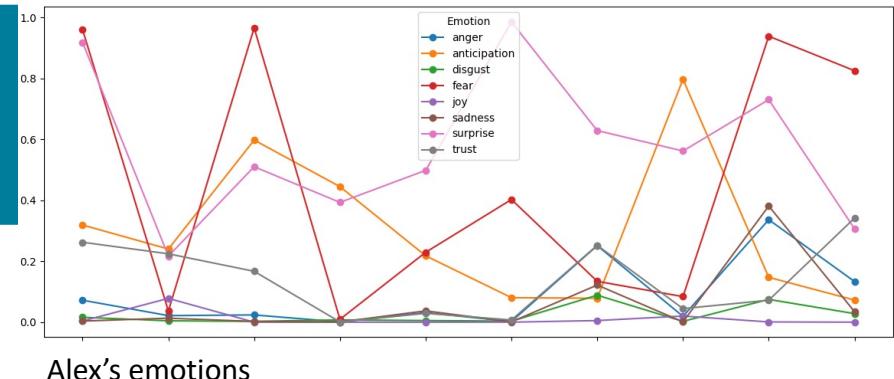


<https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/>

Empathetic generative AI

Nguyen et al 2024, work-in-progress report

- Dynamically recognise emotions expressed in user's inputs
 - Respond to the recognised emotions dynamically during the conversation
 - Provide relevant and useful information
 - Evaluate effects on user's satisfaction, trust and acceptance.



Common use cases

- **Social media monitoring:**
 - gauge public sentiment towards their brand, products, or services.
 - identify trends, potential issues, or areas of improvement.
- **Customer feedback analysis:**
 - analyse customer reviews, survey responses, or support tickets
 - identify common themes, monitor customer satisfaction, and improve their offerings
- **Content and ad optimization:**
 - analyse users' reactions to content or advertisements
 - optimise messaging, targeting, and creative strategies to maximize engagement and conversions.



<https://www.mentionlytics.com/>

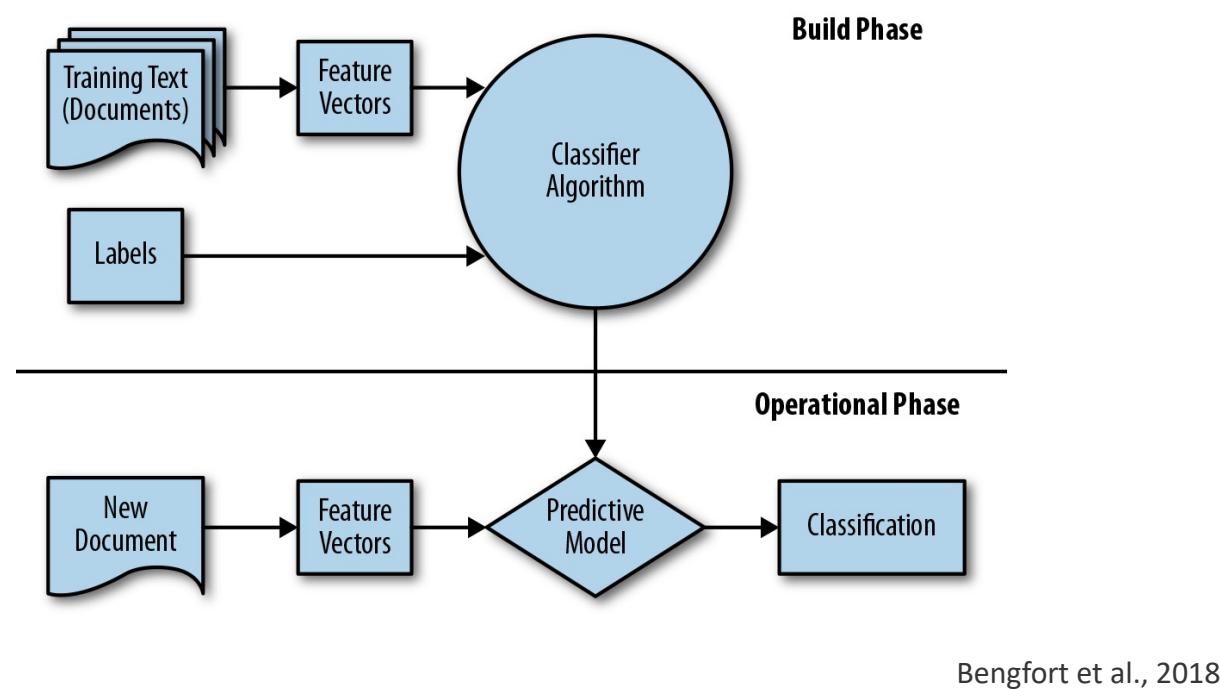
Common use cases

- **Public relations and reputation management:**
 - monitor news articles, press releases, and public statements
 - track their public image, identify potential PR crises, and devise appropriate communication strategies.
- **Financial markets and trading:**
 - analyse news articles, social media, and other sources
 - predict market trends, identify investment opportunities and risk.
- **Politics and public opinion:**
 - gauge public opinion on political candidates, policies, or events
 - provide insights for campaign strategy, policy development, or election forecasting.



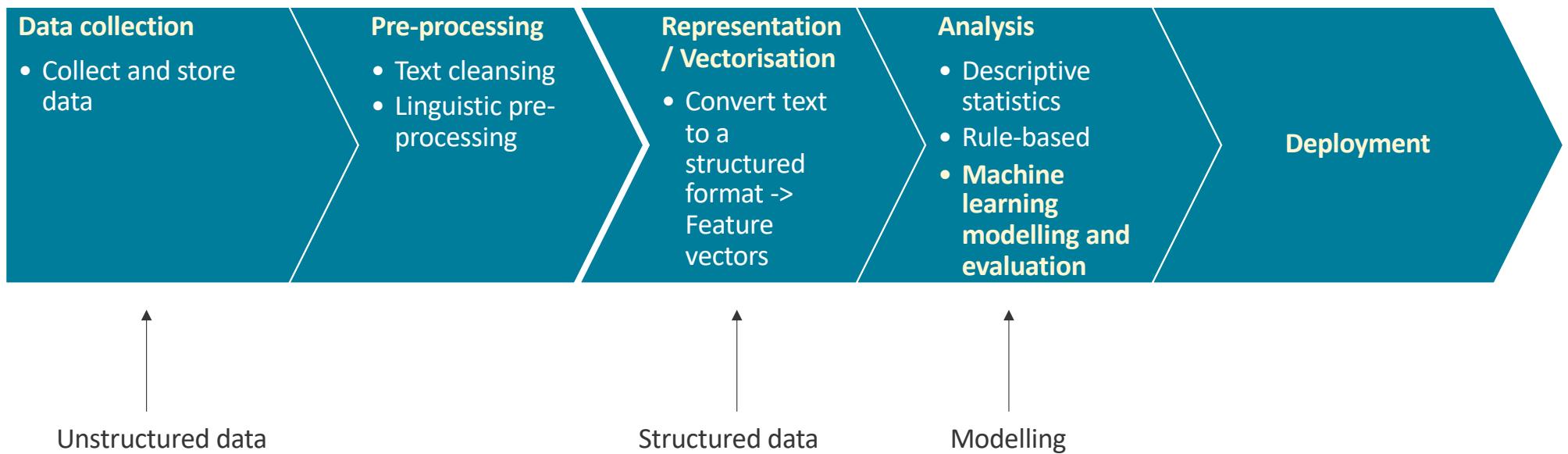
<https://ieee-dataport.org/open-access/usa-nov2020-election-20-mil-tweets-sentiment-and-party-name-labels-dataset>

Workflow overview:



Bengfort et al., 2018

Workflow with more details



Data collection

- Data sources: digital or physical documents, books, newspapers, websites, social media platforms, databases, APIs.
- Data sources can provide raw, unprocessed data that might require further cleaning, pre-processing, or transformation before being used for analysis.

Corpora

In natural language processing, a corpus is a structured and organised collection of text data

- A **corpus** is a structured collection of text data for linguistic research, NLP tasks, or machine learning.
- Typically **pre-processed, cleaned, and sometimes annotated** (e.g., with labels like sentiment or entities).
- Can be **domain-specific or general**.
- The purpose is to provide a **representative sample** of language for analysis and to develop language technologies.

Popular Corpora in NLP

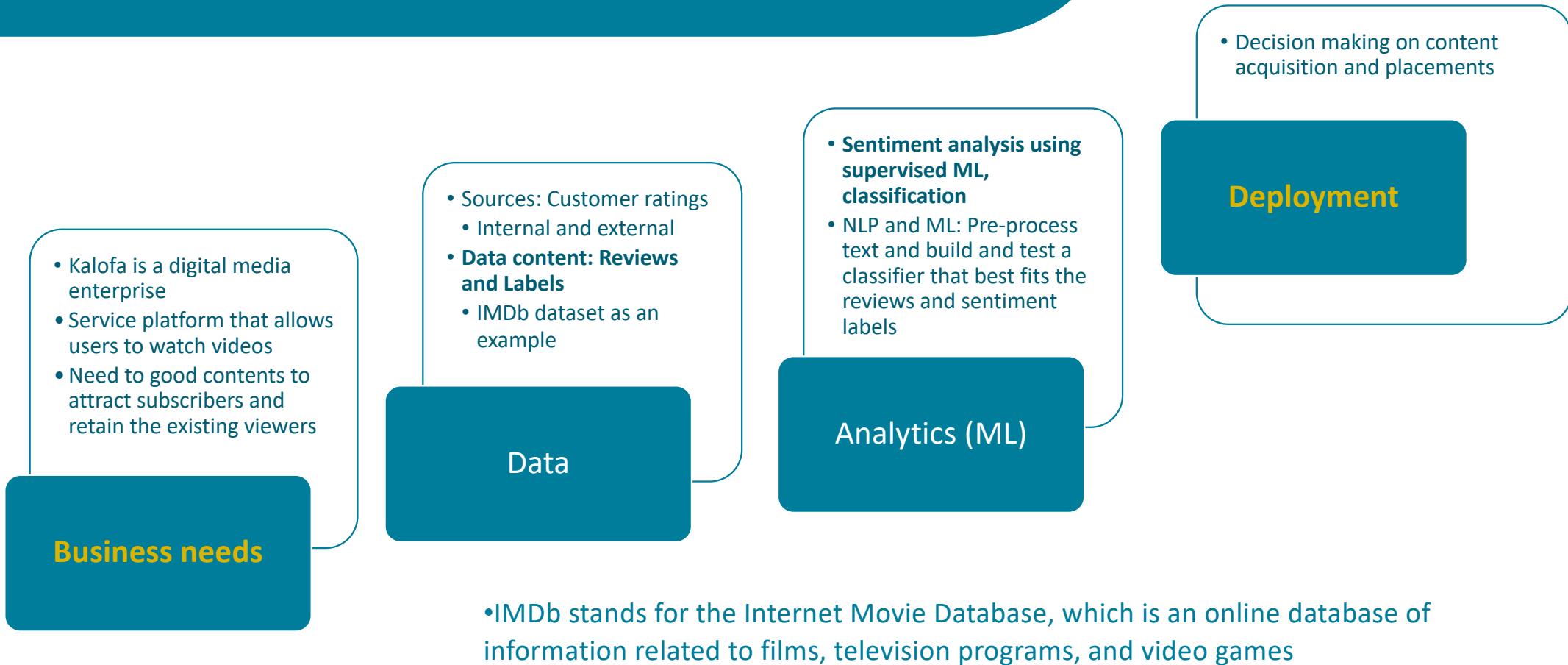
- **Wikipedia:** Large, general corpus for pretraining language models like BERT and GPT.
- **PubMed Central:** A corpus of biomedical literature used in medical text mining and clinical NLP applications.
- **IMDB Movie Reviews:** A sentiment-labelled dataset.
- **CoNLL-2003:** A corpus annotated for named entity recognition (NER), covering entities like people, locations, and organisations.

see also <https://www.nltk.org/book/ch02.html>

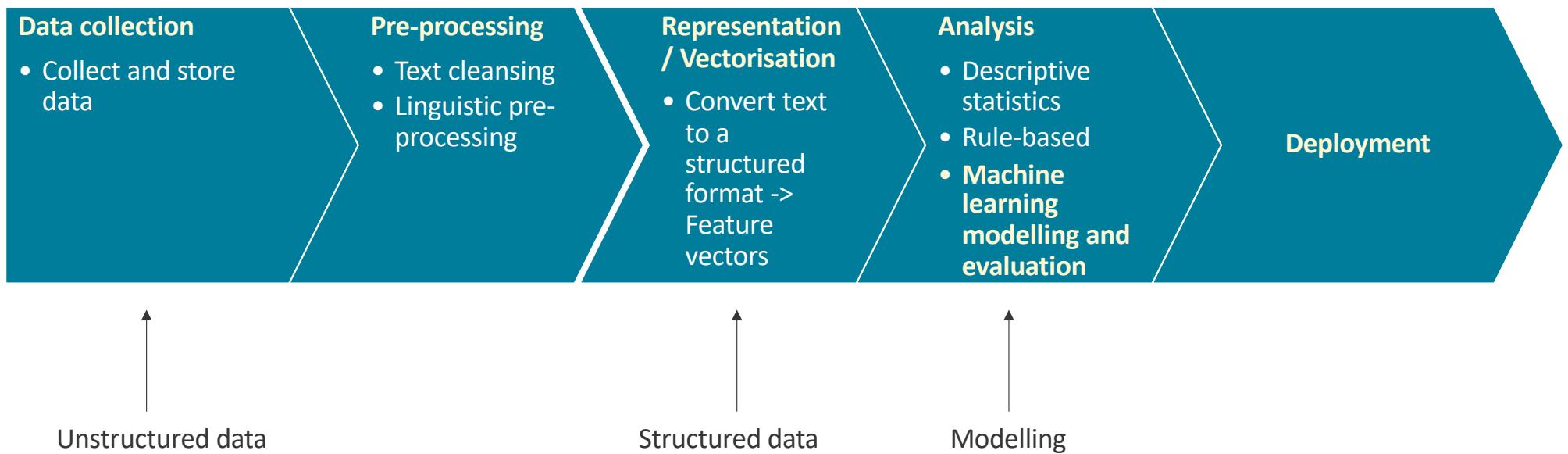
Example1: IMDB

rather twelv year old illi...	pos
saw onli part wa cri wast t...	negativ
ubber charact great perform ...	negative
yesterday first time thoroughli e...	positive
whi review thi film univers ba...	positive
copycat imperson kevin sm...	negative
i real strength thi film lea...	positi
ast wa watchabl two th...	neg

ML in Business Framing

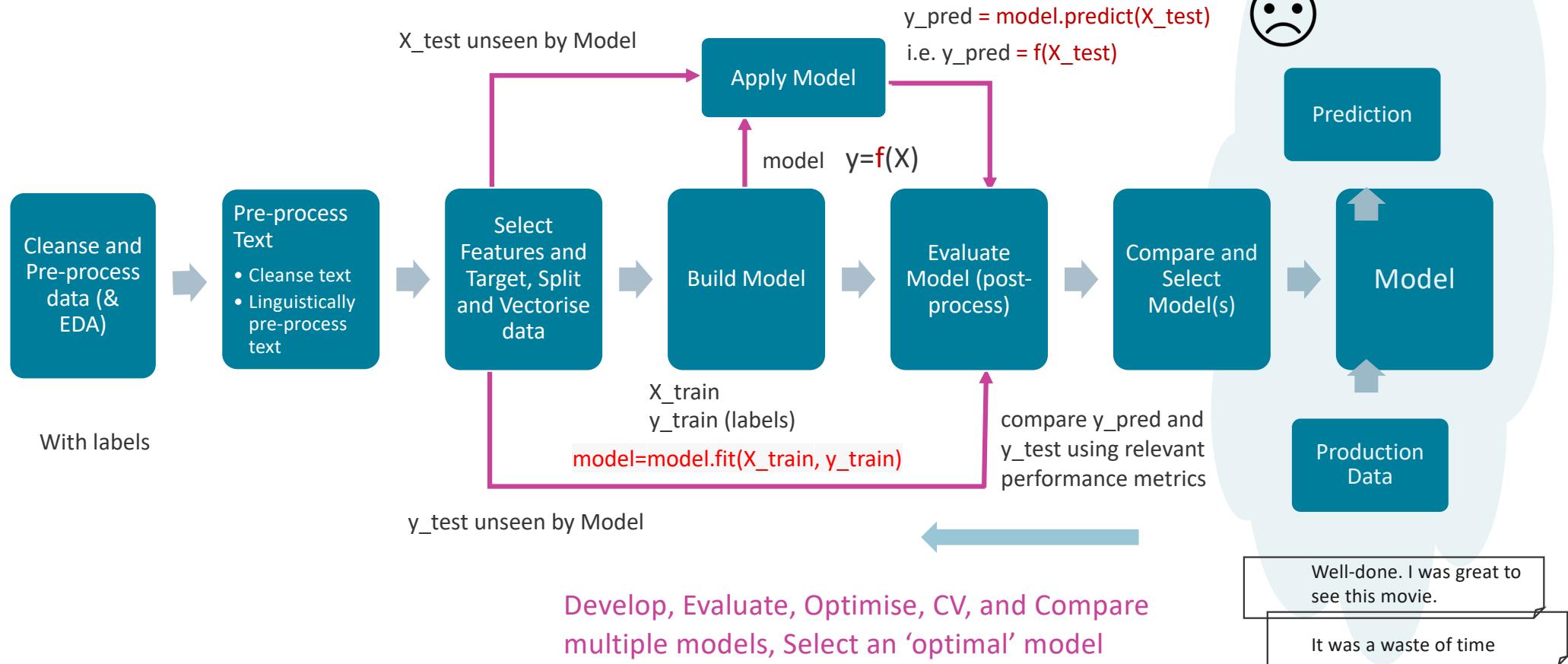


Workflow with more details



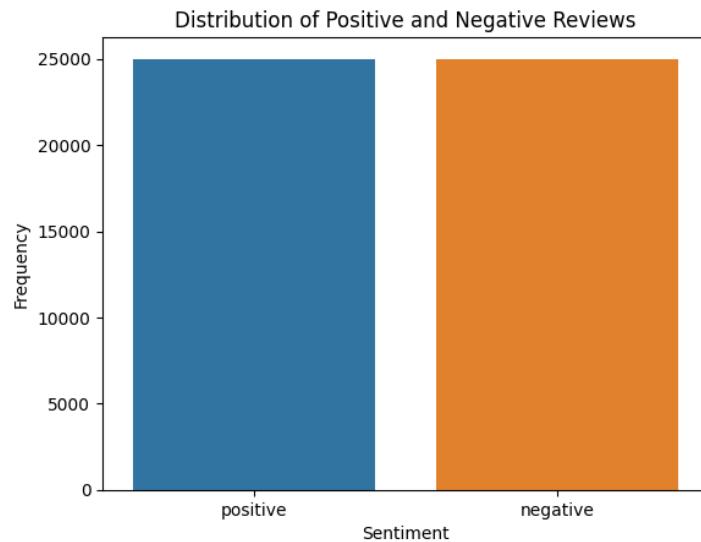
Supervised machine learning: label is sentiment

Problem: binary classification (positive / negative)



The dataset

```
records.sample(10)
```



- Remove irrelevant data
- Handling missing data
- EDA

	review	sentiment
14632	Wow...where to begin...picked this up at Big L...	negative
27790	The title pretty much lets you know what you'r...	negative
26547	I haven't written a review on IMDb for the lon...	positive
5924	Okay, make no mistake - this is a pretty awful...	negative
12421	I still count "Police Squad!" as the absolute ...	positive
41370	One of the funniest, most romantic, and most m...	positive
5285	I saw this movie once on late night t.v. and k...	positive
19641	Bizarre Tobe Hooper exercise regarding an unfo...	negative
19854	How can stuff like this still be made? Didn't ...	negative
33156	Just as in Rififi, the most compelling scene i...	positive

Key steps in Sentiment Analysis

Text Pre-processing

Text cleansing

- Remove irrelevant data
- Handle missing data
- Text cleaning (remove html tags, special characters...)
- Lowercasing

Linguistic preprocessing

- Tokenization
- Stopword removal
- Text normalization: Stemming or Lemmatization
- Part of speech
- Name-entity recognition

Text Representation

- Vectorisation, e.g. Bag-of-words, **TF-IDF**, Word2Vec, BERT, GPT

Model training, evaluation and selection

- Machine learning workflow

Text pre-processing: cleansing

Text cleansing:

- Remove any HTML tags, punctuation, and special characters from the reviews.

- **Regular expressions** are patterns used to match sequences of alphanumeric and special characters in text.
- The **re module** in Python provides functions for working with regular expressions, allowing for powerful text searching, matching, and manipulation.

A wonderful Australian production.
 The Australian filming technique is very unassuming, reminiscent very old-time-BBC.
'



A wonderful Australian product The Australian film production technique is very unassuming reminiscent very oldtimeBBC

Text pre-processing: cleansing

`re.sub()` to replace occurrences of a pattern with a replacement.

```
#import re to work with regular expressions
import re

# Define function to clean text
def clean_text(text):
    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # Remove punctuation and special characters
    text = re.sub(r'[^w\s]', '', text)
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()
return text

#let's get reviews to run clean_text
records['review']= records['review'].apply(clean_text)
```

- The . (dot) metacharacter matches any single character except for a newline character.
- The * (asterisk) metacharacter matches zero or more occurrences of the preceding character or group.
- ? preceding token can appear zero or one time
- \w: Matches any word character, which includes letters, digits, and underscores ([a-zA-Z0-9_]).
- \s matches any whitespace character, which includes spaces, tabs, newlines, and other types of whitespace.
- ^: Inside square brackets, the ^ negates the pattern, [^w\s] matches any character that is **not** a word character and **not** a whitespace.
- \s+ matches one or more consecutive whitespace characters. The + (plus) is a metacharacter that means "one or more occurrences of the preceding pattern".

Text pre-processing: cleansing

```
#import re to work with regular expressions
import re

# Define function to clean text
def clean_text(text):
    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # Remove punctuation and special characters
    text = re.sub(r'[^w\s]', '', text)
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()
return text

#let's get reviews to run clean_text
records['review']= records['review'].apply(clean_text)
```

A wonderful Australian production.
 The Australian filming technique is very unassuming, reminiscent very old-time BBC.
'

A wonderful Australian product The Australian film production technique is very unassuming reminiscent very oldtimeBBC

Text pre-processing: lowercasing

Lowercasing:

- Convert all text to lowercase to ensure that words with the same spelling but different capitalization are treated as the same word.
- The `lower()` method comes from Python's built-in `str (string)` class.

```
def lowercasing(text):  
    # Convert to lowercase  
    text = text.lower()  
    return text  
  
records['review']=records['review'].apply(lowercasing)
```

A wonderful little production The filming technique is very unassuming very oldtimeBBC



a wonderful australian product the australian film production technique is very unassuming reminiscent very oldtimebbc

Text pre-processing: Linguistic pre-processing

Tokenization:

- Split the text into individual words or tokens.

Stopword removal:

- Remove common words such as "the", "and", "a", etc. that do not add much meaning to the text.

Lemmatization:

- Reduces words to their base form to normalize the text and preserves the original meaning of words better than stemming.
- E.g., studying, studies, studied -> study; flew -> fly; better -> good

Stemming

- Reduces words to their base or root form by removing suffixes and prefixes
- E.g., studying, studies, studied -> studi; flew -> fle; better -> bett

NLTK (Natural Language Toolkit) is a Python library for natural language processing (NLP) e.g., tokenization, stemming, lemmatization, part-of-speech tagging, and more.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,
WordNetLemmatizer
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

stopwords is part of the **NLTK** library, a list of commonly used stopwords in different languages, including English.

```
# define stopwords without negation words
stop_words = set(stopwords.words('english'))
negation_words = {'no', 'not', 'nor', 'neither', 'none', 'never'}
filtered_words = [word for word in stop_words if word not in negation_words]

#define a function to perform tokenization and lemmatization
def tokenize_lemmatize(text):
    #tokenization
    tokens = nltk.word_tokenize(text)

    #initialize stemmer OR lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Filtering out the stopwords and lemmatizing the tokens
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens
                         if token not in filtered_words]
    return ' '.join(lemmatized_tokens)
```

Use PorterStemmer for stemmatisation



```
a wonderful australian  
product the australian film  
production technique is very  
unassuming reminiscent very  
oldtimebbc
```

```
# Tokenize and lemmatize the review text  
processed_text = records['review'].apply(tokenize_lemmatize)
```

Result of stemming would be:

wonder australian product australian film
product techniqu unassum reminisc oldtimebbc



```
wonderful australian product  
australian film production  
technique unassuming  
reminiscent oldtimebbc
```



Text representation: vectorization

- **Vectorisation** is a process of converting textual data into numerical vectors that can be used as input for machine learning algorithms.
- **Bag of Words, TF-IDF, Word2Vec, BERT, GPT**
- **CountVectorizer** is an implementation of **Bag of Words (BoW)**

```
from sklearn.feature_extraction.text import CountVectorizer  
# Create the vectorizer object and fit on the corpus  
vectorizer = CountVectorizer()  
vectorizer = vectorizer.fit(processed_text)
```

wonderful australian product
australian film production
technique unassuming
reminiscent oldtimebbc



australian
film
oldtimebbc
product
production
reminiscent
technique
unassuming
wonderful

Text representation: vectorization

wonderful australian product
australian film production
technique unassuming
reminiscent oldtimebbc

ngram: related to both tokenisation and vectorisation

- In NLP, an n-gram is a sequence of words or characters in a sentence.
- Used in language modelling, text classification, and information retrieval to capture the local context and dependencies between words or characters in a sentence or text.
- n-grams (e.g., bigrams or trigrams) are more informative and capture the contextual meaning of words better than individual tokens.

```
from sklearn.feature_extraction.text import CountVectorizer  
# Create the vectorizer object and fit on the corpus  
vectorizer = CountVectorizer(ngram_range=(1, 2))  
vectorizer = vectorizer.fit(processed_text)
```

australian
australian film
australian product
film
film production
oldtimebbc
product
product australian
production
production technique
reminiscent
reminiscent oldtimebbc
technique
technique unassuming
unassuming
unassuming reminiscent
wonderful
wonderful australian

Text representation: vectorization

ngram: related to both tokenisation and vectorisation

- In NLP, an n-gram is a sequence of words or characters in a sentence.
- Used in language modelling, text classification, and information retrieval to capture the local context and dependencies between words or characters in a sentence or text.
- n-grams (e.g., bigrams or trigrams) are more informative and capture the contextual meaning of words better than individual tokens.

```
from sklearn.feature_extraction.text import CountVectorizer  
# Create the vectorizer object and fit on the corpus  
vectorizer = CountVectorizer(ngram_range=(2, 2))  
vectorizer = vectorizer.fit(processed_text)
```

wonderful australian product
australian film production
technique unassuming
reminiscent oldtimebbc



australian film
australian product
film production
product australian
production technique
reminiscent oldtimebbc
technique unassuming
unassuming reminiscent
wonderful Australian

Text representation: vectorization

my_text

This Australian film is a wonderful film.

- **unigram n=1**

australian
film
oldtimebbc
product
production
reminiscent
technique
unassuming
wonderful

australian: 1
film: 0
wonderful: 1

```
my_vector= vectorizer.transform(my_text)
```

A vector in this 9-dimentional vocabulary space

australian	film	oldtimebbc	product	production	reminiscent	technique	unassuming	wonderful
1	2	0	0	0	0	0	0	1



Text representation: vectorization

- **Bigram n=2 (ngram_range=(1, 2))**

There are other tokens not found in the vocabulary due to the small size of this example

```
australian
australian film
australian product
film
film production
oldtimebbc
product
product australian
production
production technique
reminiscent
reminiscent oldtimebbc
technique
technique unassuming
unassuming
unassuming reminiscent
wonderful
wonderful australian
```

This Australian film is a wonderful film.

australian film wonderful
film

australian: 1
australian film: 1
film: 2
wonderful: 1

A vector in this 18-dimentional vocabulary space



Text representation

TF-IDF the frequency of each word weighted by its frequency across all documents.

$$TF(t,d) = (\text{freq. of word } t \text{ in a document}) / (\#\text{words in a document } d)$$

A higher TF for a word indicates that the word appears more frequently in the document.

$$IDF(t,D) = \log(\#\text{documents } D) / (\#\text{documents containing the word } t)$$

The higher the IDF of a word, the rarer it is in the corpus.

TF-IDF (t, d, D) (term frequency * inverse document frequency)

a higher TF-IDF generally indicates that a word is more important in the given document, as well as in the overall corpus

the quick brown fox jumps
over the lazy dog

the dog chased the fox

the fox is quick and agile

'the': 2/9, 'quick': 1/9,
'brown': 1/9, 'fox': 1/9,
'jumps': 1/9, 'over': 1/9,
'lazy': 1/9, 'dog': 1/9

'the': $\log(3/3)=0$, 'quick': $\log(3/2)=0.405$, 'brown': $\log(3/1)=1.099$,
fox: $\log(3/2)=0.176$...

agile	and	brown	chased	dog	fox	is	jumps	lazy	over	quick	the
0	0	0.077	0	0.032	0	0	0.077	0.077	0.077	0.032	0

Text representation: vectorization

- Bigram n=2 (`ngram_range=(1, 2)`)

actress
hilarious
look
entertaining
annoying
unnecessary
scene
love
drama
much

actress hilarious
look entertaining
annoying unnecessary
unnecessary scene
love drama
drama much

This actress is hilarious
Looks very entertaining
annoying unnecessary scene
I love this drama so much

actress hilarious
look entertaining
annoying unnecessary scene
love drama much

	actress	actress	hilarious	annoying	annoying	unnecessary	drama	drama	much	entertaining	hilarious	look	look	entertaining	love	love	drama	much	scene	unnecessary	unnecessary	scene
0	0.57735	0.57735	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.57735	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
1	0.00000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.57735	0.000000	0.57735	0.57735	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2	0.00000	0.00000	0.447214	0.447214	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.447214	0.447214	0.447214	0.447214	0.447214	0.447214	0.447214
3	0.00000	0.00000	0.000000	0.000000	0.447214	0.447214	0.447214	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.447214	0.447214	0.447214	0.447214	0.000000	0.000000	0.000000	0.000000	

Text representation: some vectorization methods

Method	Key Feature	Captures Word Context?	Size of Vectors
Bag of Words (BoW)	Frequency-based	No	High (sparse matrix)
TF-IDF	Weights frequent words by rarity	No	High (sparse matrix)
Word Embeddings (Word2Vec, GloVe)	Dense vectors capturing semantic similarity	Somewhat	Low (dense vectors)
Pre-trained Embeddings (BERT, GPT)	Context-aware vectors	Yes	Medium (contextual)
Doc2Vec	Dense vectors for entire documents	Yes	Low (dense vectors)

Split data

- If we vectorize the whole dataset before splitting into training and test sets, there is a risk of data leakage.
- Fit the initialised vectorizer only on the training data. Use this fitted vectorizer to transform both the training and test data.

```
X=processed_text
y=records['sentiment']

# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_text, records['sentiment'],
test_size=0.2, random_state=2023)

# Create the vectorizer object
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.7, ngram_range=(1, 2))

X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Build and apply a model

```
# Train logistic regression model
lr = LogisticRegression(penalty='l2', C=2, solver='liblinear', max_iter=1000, random_state=2023)
lr.fit(X_train_vec, y_train)

# Evaluate model
y_pred = lr.predict(X_test_vec)

# get predicted probabilities for the main class
y_pred_probs = lr.predict_proba(X_test_vec)
y_pred_probs = y_pred_probs[:, 1]
```

	review	Actual	Predicted
27114	one onli imagin film mr well might finish with...	positive	positive
30284	thi could breakout role valeria golino film in...	negative	positive
27763	check spoiler alert caseperhap one horrend mov...	negative	negative
7243	thi per se abov averag film whi name bog wa ma...	negative	negative
10676	imagin watch slide show projector linger everi...	negative	positive
48216	trooper side road make sure every1 obey speed ...	negative	negative
10378	opt see film recent dubai film festiv becaus s...	negative	positive
33926	thi movi wa lame lame lame build let form subs...	negative	negative

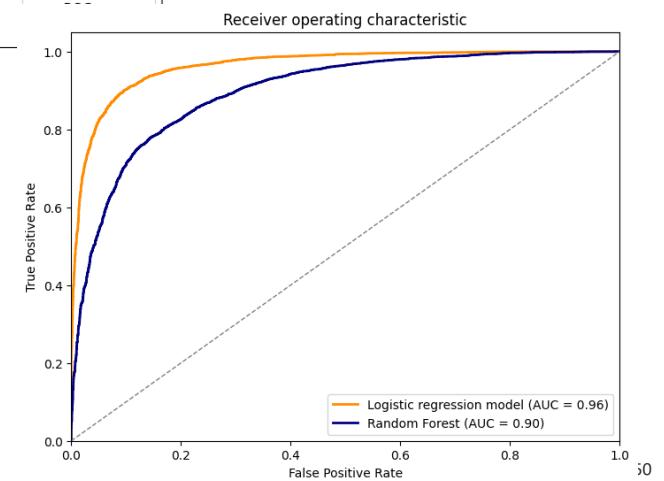
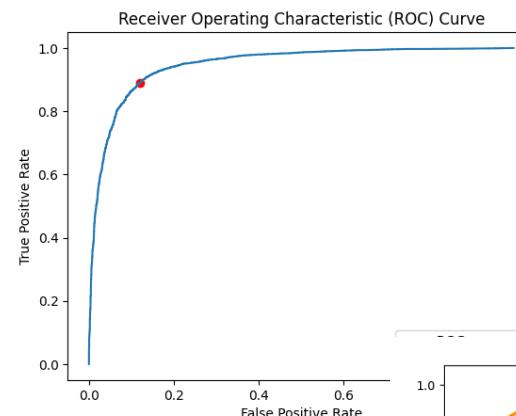
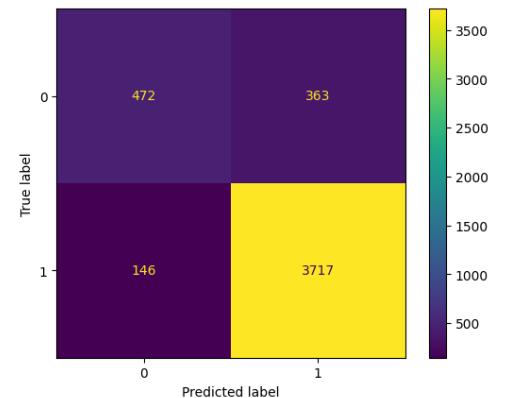
Evaluate the model

```
# Computer TN FP FN TP
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()

# Compute FPT and TPR for this k
tpr = tp / (tp + fn)
fpr = fp / (fp + tn)

print('True Positive Rate: ', '%.3f' % tpr)
print('False Positive Rate: ', '%.3f' % fpr)
```

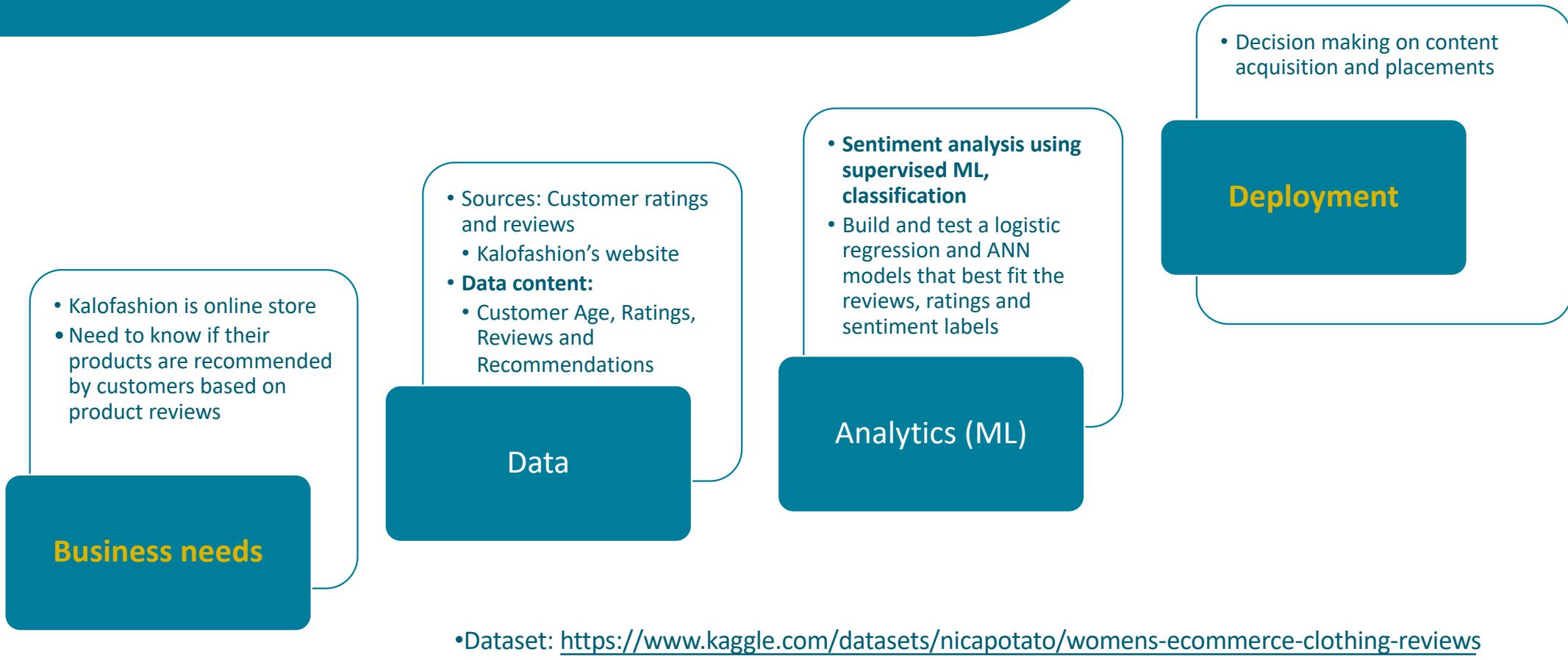
True Positive Rate: 0.890
False Positive Rate: 0.120



Example 2: Kalofashion

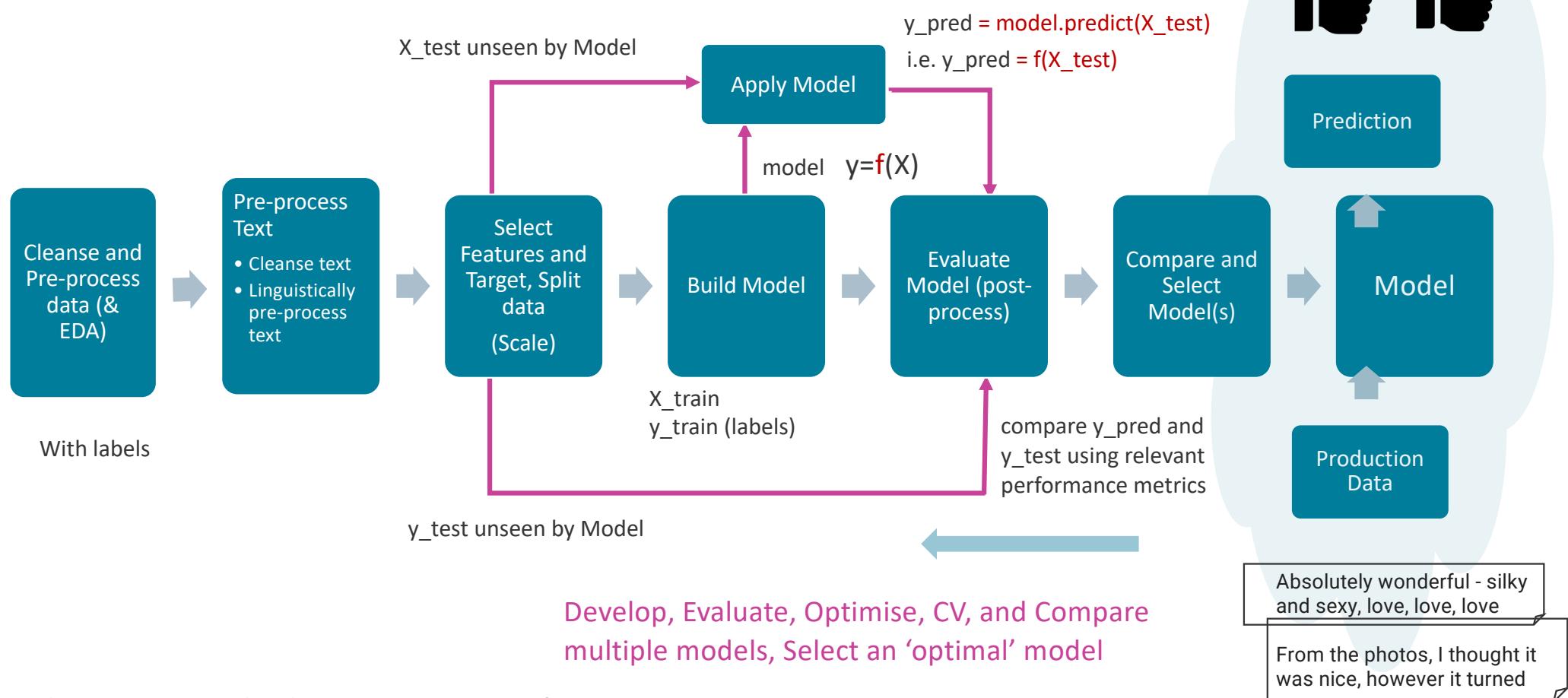
rather twelv year old illi...	pos
saw onli part wa cri wast t...	negativ
ubber charact great perform ...	negative
yesterday first time thoroughli e...	positive
whi review thi film univers ba...	positive
copycat imperson kevin sm...	negative
i real strength thi film lea...	positi
ast wa watchabl two th...	neg

ML in Business Framing



Supervised machine learning: label is Recommended

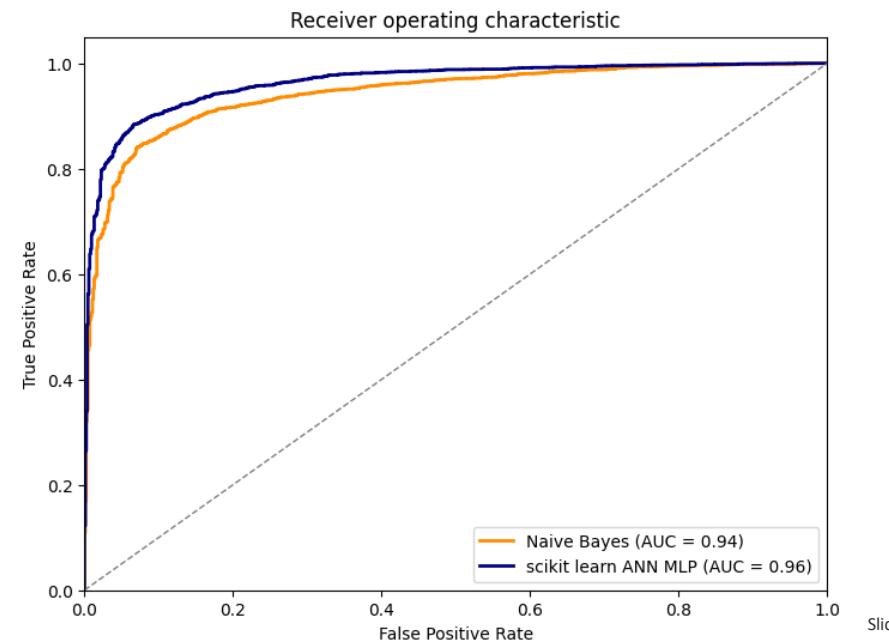
Problem: binary classification (1 / 0)

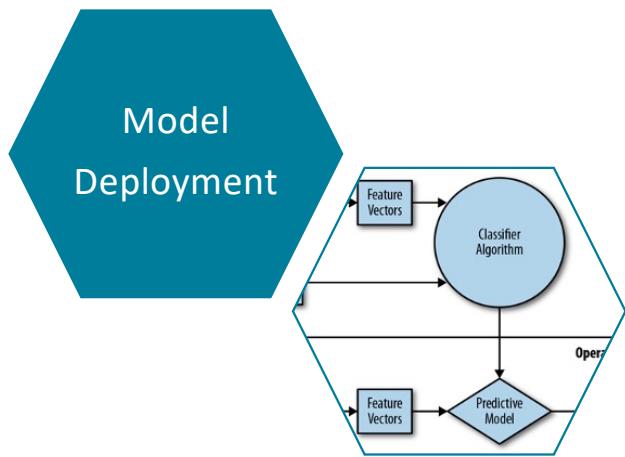


Mixed text and other variables

- Same process with some modifications
- Merge Title and Reviews to handle missing data
- Pre-process the text (Merge Title and Reviews)
- X consists of Processed_Text, Age and Rating
- Vectorize Processed_Text in X_train and X_test, keep Age and Rating
- Build and Test a Multinomial Naïve Bayes model and Optimise threshold
- Build and Test an ANN MLP model and Optimise threshold
- Conduct model comparison

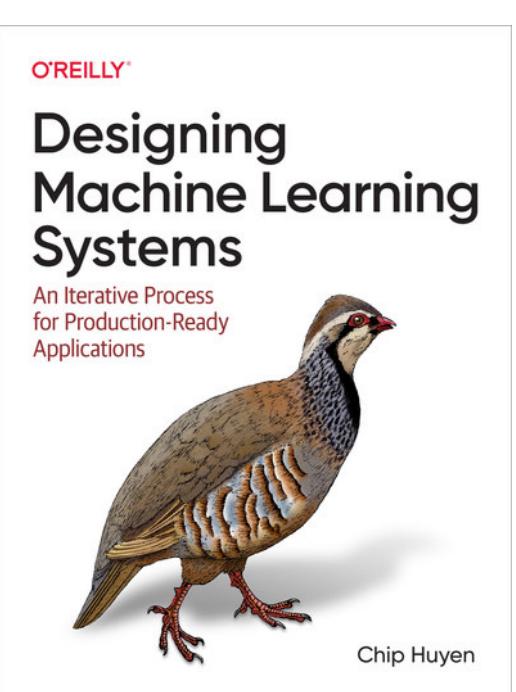
Processed_Text	Age	Rating	Actual	Predicted
great springsumm cardiganwa look someth lighte...	38	5	1	1
cute design imposs fiti tri thi regular size w...	46	2	0	0
thi blous veri beauti flatter littl bit see we...	21	5	1	1
realli like thi top color great summer fall be...	52	4	1	1
uniqu topi love thi top return becaus stiff ap...	59	4	1	1
weird fitbought thi base review thi wa miss co...	46	2	0	0





- **Data Engineering** focuses on the infrastructure for data handling.
- **ML Engineering** involves building and optimizing ML models.
- **ML Ops** is a specialized form of DevOps for ML models, focusing on deployment and lifecycle management.
- **AI Engineering** covers a broader scope of AI, including ML but also more advanced AI techniques.

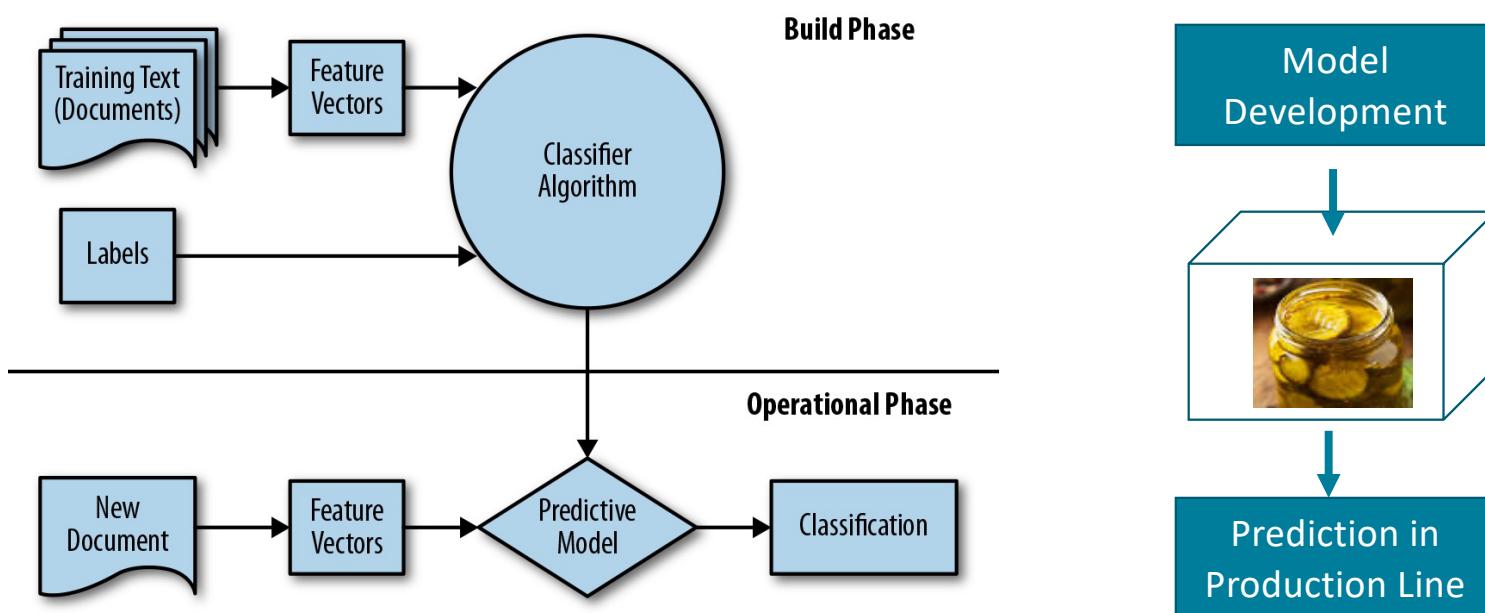
MLOps



*Ops in MLOps comes from DevOps, short for Developments and Operations. To operationalize something means to bring it into production, which includes **deploying, monitoring, and maintaining** it. MLOps is a set of tools and best practices for bringing ML into production.*

Chip Huyen

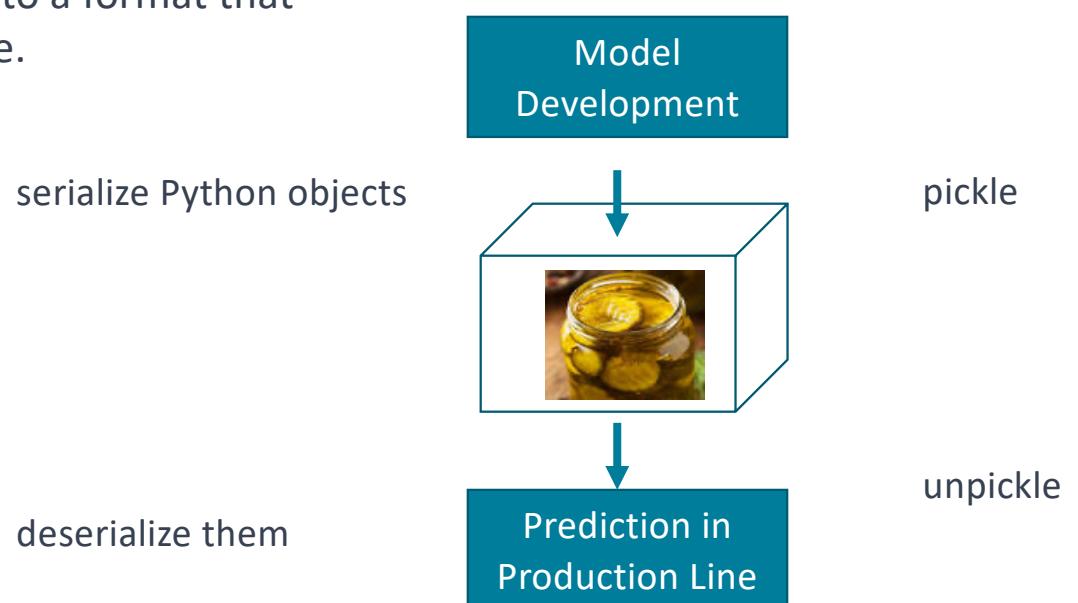
Pickle



Bengfort et al., 2018

Pickle objects

Pickle converts complex data structures into a format that could be stored, shared and loaded for use.



Save your model

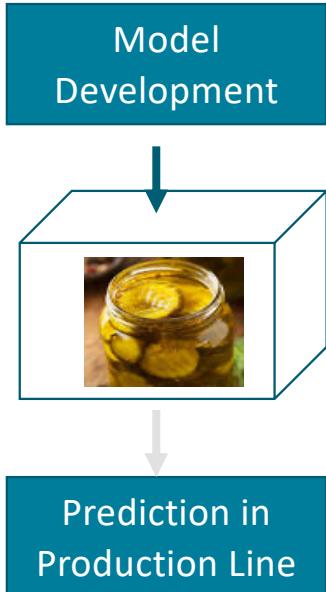
```
import pickle

# Define the path
path_vectorizer = '/content/drive/MyDrive/Colab Notebooks/tfidf_vectorizer.pkl'
path_model = '/content/drive/MyDrive/Colab Notebooks/IMDB_lr_clf.pkl'

# Save to file
with open(path_vectorizer, 'wb') as f:
    pickle.dump(tfidf_vectorizer, f)

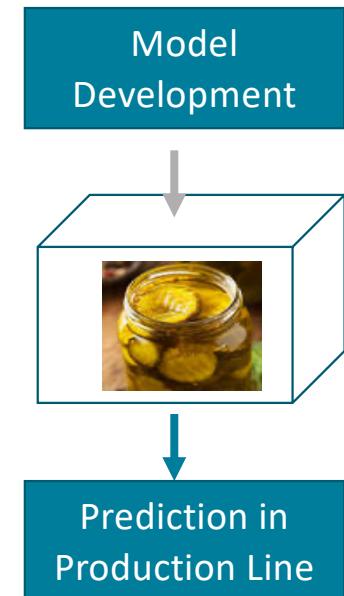
with open(path_model, 'wb') as f:
    pickle.dump(lr, f)

wb: write binary
```



Load and use your pre-trained model

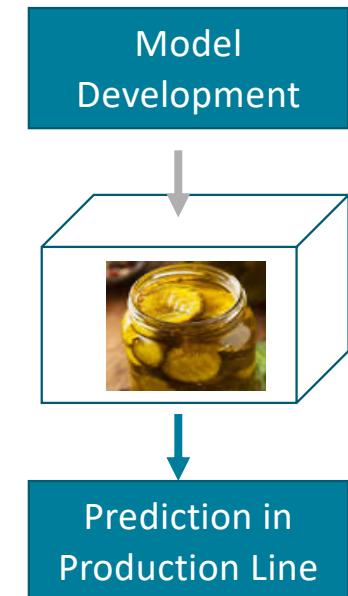
```
with open(path_vectorizer, 'rb') as f:  
    loaded_vectorizer = pickle.load(f)  
  
with open(path_model, 'rb') as f:  
    loaded_lr = pickle.load(f)  
  
X_new_tfidf =  
    loaded_vectorizer.transform(new_processed_text)  
  
rb: read binary
```



Load and use your pre-trained model

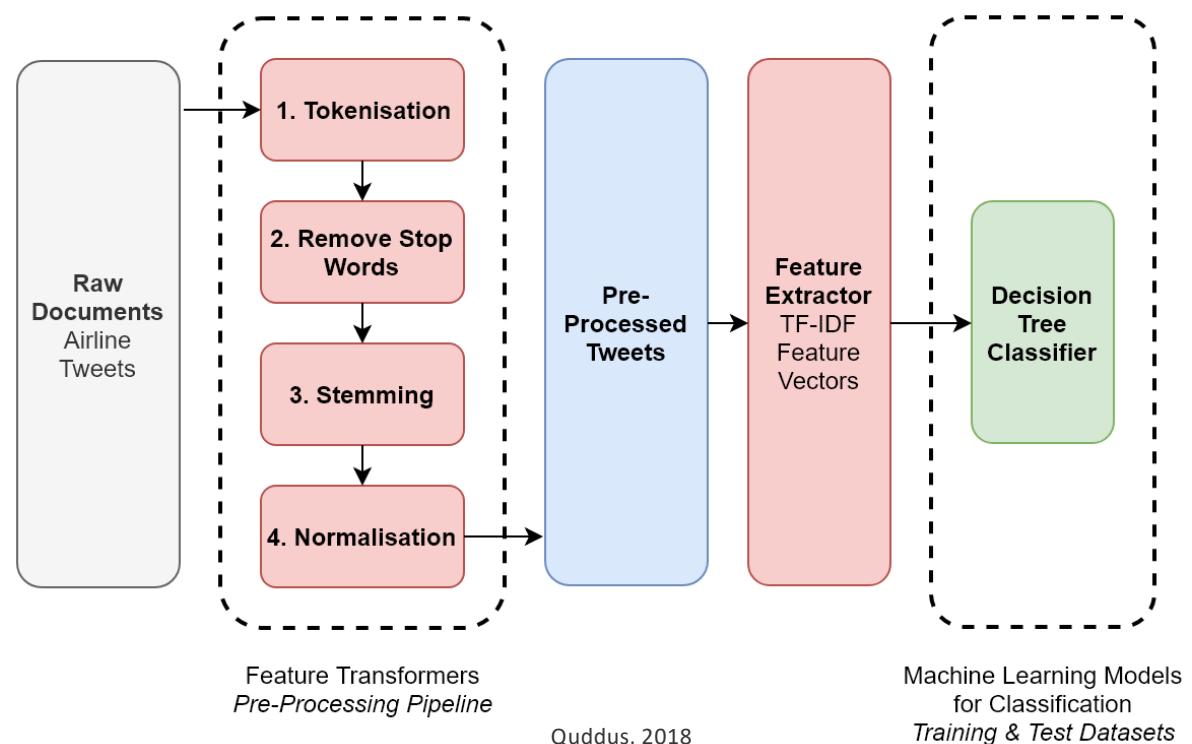
```
#Use loaded vectorizer to vectorize the new processed text  
X_new_tfidf = loaded_vectorizer.transform(new_processed_text)
```

```
# Make predictions  
y_pred_new = loaded_lr.predict(X_new_tfidf)
```



Machine Learning Pipeline for NLP

- A machine learning pipeline for NLP automates the process of analysing and understanding human language using machine learning techniques.
- It encompasses a series of steps to process text data. E.g., transform features, train models, and make predictions.
- Benefits:
 - Efficiency and Consistency
 - Modularity and Reusability
 - Experimentation and Iteration
 - Scalability and Deployment
 - Interpretability and Explainability

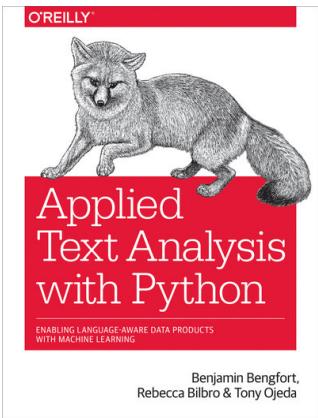


Text analysis - Recap



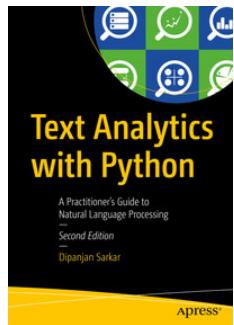
- Text analysis, NLP, LLM
- Sentiment analysis: overview, use cases and workflow
 - Examples: Two datasets
- MLOps
 - Save your model
 - Load the model and use it on new data
 - ML pipeline

Useful readings

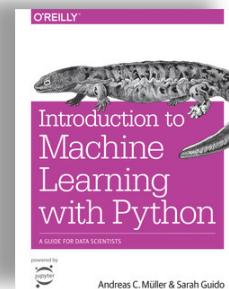


[https://learning.oreilly.com/library/
view/applied-text-
analysis/9781491963036/ch05.html](https://learning.oreilly.com/library/view/applied-text-analysis/9781491963036/ch05.html)

Chapter 5

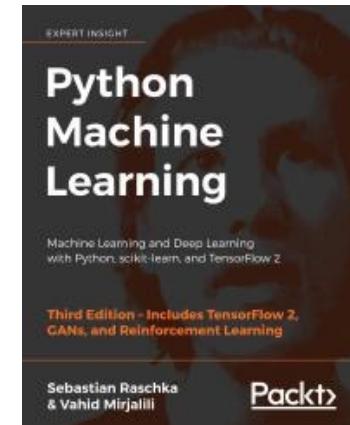


[https://learning.oreilly.com/library/
view/text-
analytics-with/9781484243541/](https://learning.oreilly.com/library/view/text-analytics-with/9781484243541/)



[https://learning.oreilly.com/library/
view/introduction-to-
machine-learning/9781449369880/?sso_link=
yes&sso_link_from=Deakin](https://learning.oreilly.com/library/view/introduction-to-machine-learning/9781449369880/?sso_link=yes&sso_link_from=Deakin)

Chapter 7



[https://ebookcentral-proquest-com.ezproxy-
b.deakin.edu.au/lib/deakin/reader.action?docID=6005547&pr](https://ebookcentral-proquest-com.ezproxy-b.deakin.edu.au/lib/deakin/reader.action?docID=6005547&pr)

Raschka and Mirjalili, 2019

Chapter 8

https://huggingface.co/models?pipeline_tag=text-classification&sort=trending

Self-study Intro to LLM

rather twelv year old illi...	pos
saw onli part wa cri wast t...	negativ
ubber charact great perform ...	negative
yesterday first time thoroughli e...	positive
whi review thi film univers ba...	positive
copycat imperson kevin sm...	negative
i real strength thi film lea...	positi
est wa watchabl two th...	neg

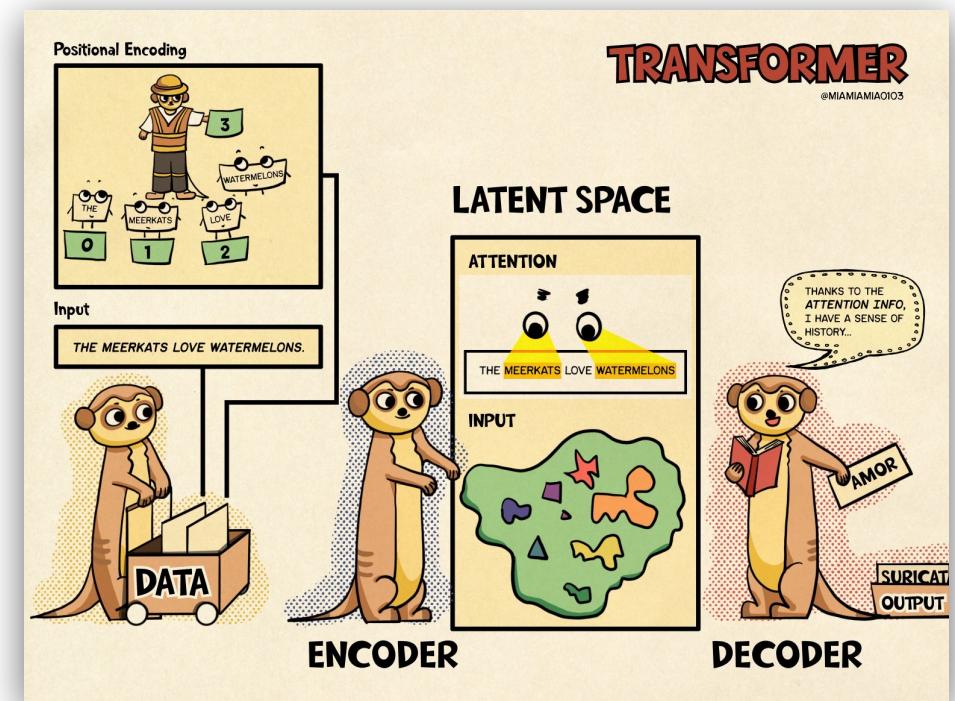
Transformer architecture

- ChatGPT – transformer-based language model

Attention is all you need

Vaswani, et al., 2017

- The encoder takes an input sequence and encodes it into context vectors.
- The decoder uses the context vectors to generate an output sequence.
- Self-attention focuses on different words of the input sequence when processing each word.

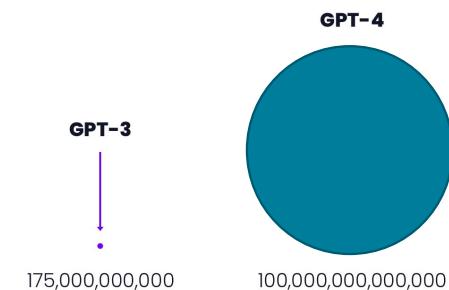


<https://twitter.com/Miamiamia0103/status/1550863605519847425/photo/1>

LLM training phases

- Phase 1: Pre-trained on large corpora of text data.
 - Unsupervised: self-supervised learning.
 - The training process involves learning the patterns and structures within the text by optimising the model parameters, which enable the model to generate coherent and contextually relevant responses when given a prompt.
 - Once trained, it can generate texts (sequences of tokens) based on the probability of preceding words in the training corpora.

- Input: "I" Target: "will"
- Input: "I will" Target: "always"
- Input: "I will always" Target: "love"
- Input: "I will always love" Target: "you"



Redrawn based on acquisition.com

Pre-training phase: Mathematical formulation

- LLM_ϕ : the language model being trained, parameterized by ϕ . The goal is to find ϕ for which the cross entropy loss is minimized.
- $[T_1, T_2, \dots, T_V]$: vocabulary – the set of all unique tokens in the training data.
- V : the vocabulary size.
- $f(x)$: function mapping a token to its position in the vocab. If x is T_k in the vocab, $f(x) = k$.
- Given the sequence (x_1, x_2, \dots, x_n) , we'll have n training samples:
 - Input: $x = (x_1, x_2, \dots, x_{i-1})$
 - Ground truth: x_i
- For each training sample (x, x_i) :
 - Let $k = f(x_i)$
 - Model's output: $LLM(x) = [\bar{y}_1, \bar{y}_2, \dots, \bar{y}_V]$. Note: $\sum_j \bar{y}_j = 1$
 - The loss value: $CE(x, x_i; \phi) = -\log \bar{y}_k$
- Goal: find ϕ to minimize the expected loss on all training samples. $CE(\phi) = -E_x \log \bar{y}_k$

GPT-3 was trained on 570GB of data

GPT-3 models (between 125M and 175B parameters)

<https://openai.com/research/improving-language-model-behavior>

Chip Huyen
<https://huyenchip.com/2023/05/02/rlhf.html>

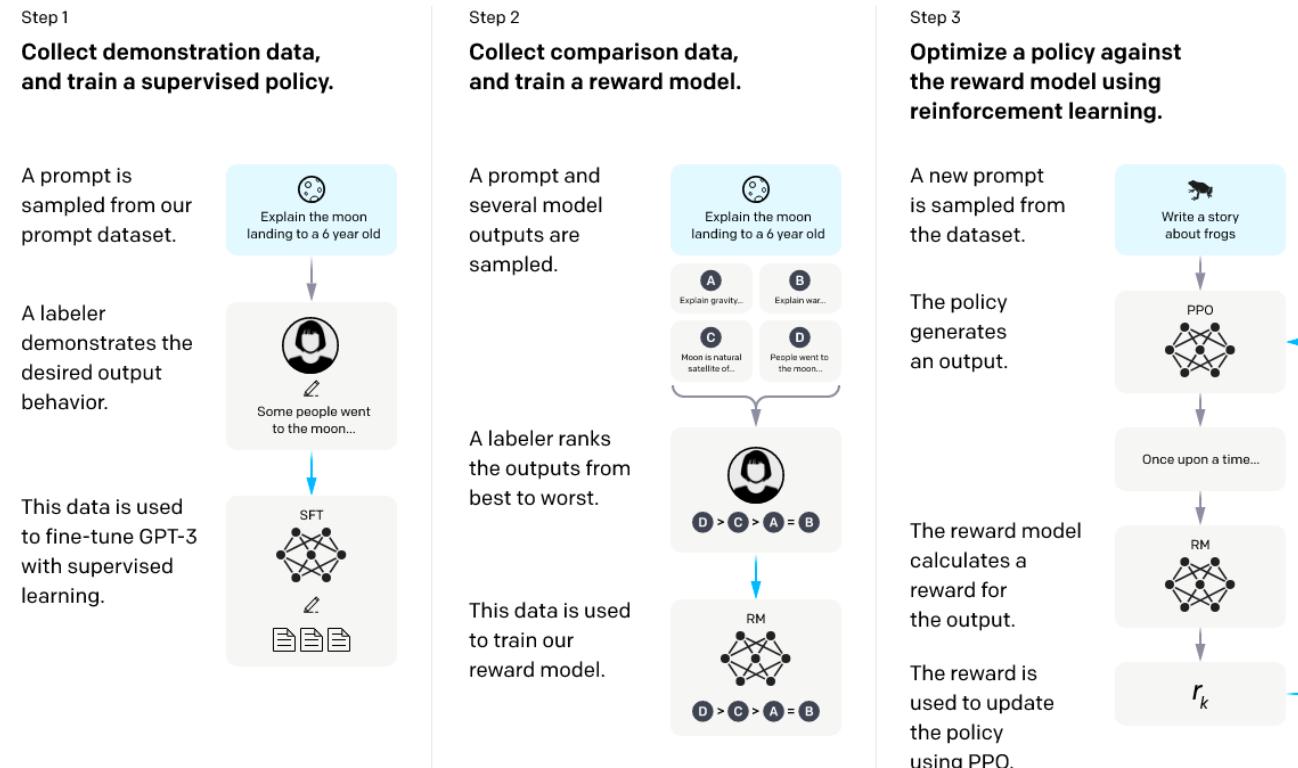
Supervised Finetuning (SFT) phase: Mathematical formulation

Phase 2:

- ML task: language modelling
- Training data: high-quality data in the format of (prompt, response)
- Data scale: 10,000 - 100,000 (prompt, response) pairs
 - InstructGPT: ~14,500 pairs (13,000 from labellers + 1,500 from customers)
 - Alpaca: 52K ChatGPT instructions
 - Databricks' Dolly-15k: ~15k pairs, created by Databricks employees
 - OpenAssistant: 161,000 messages in 10,000 conversations -> approximately 88,000 pairs
 - Dialogue-finetuned Gopher: ~5 billion tokens, which I estimate to be in the order of 10M messages.
 - NOTE: these are filtered out using heuristics from the Internet, so not of the highest quality.
- Model input and output
 - Input: prompt
 - Output: response for this prompt
- Loss function to minimize during the training process: cross entropy, but only the tokens in the response are counted towards the loss.

Chip Huyen
<https://huyenchip.com/2023/05/02/rlhf.html>

Phase 3: Reinforcement Learning from Human Feedback



<https://openai.com/research/instruction-following>

ChatGPT limitations



Summarise your limitations



Here's a summary of the limitations of ChatGPT:

1. Lack of true understanding: Generates responses based on learned patterns, without human-like understanding.
2. Sensitivity to input phrasing: Responses can vary with slight changes in phrasing or wording of queries.
3. Verbosity: Tends to be overly verbose or repetitive, occasionally overusing certain phrases.
4. Ambiguity handling: May guess user intentions instead of seeking clarifications for ambiguous questions.
5. Inaccurate or nonsensical answers: Can produce plausible-sounding but nonsensical responses.
6. Safety concerns: Despite efforts, may occasionally generate inappropriate or offensive content.
7. Limited knowledge: Knowledge is capped at the time of the last update, missing recent information.

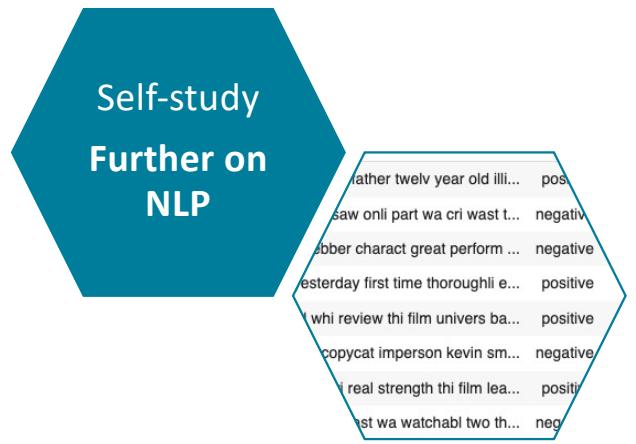


Summarise your answer: Is the text content produced by ChatGPT plagiarism?



Text content produced by ChatGPT is not plagiarism in the traditional sense, as it generates original responses based on learned patterns rather than directly copying existing text. However, users should be aware of potential similarities to existing content and provide proper attribution when using AI-generated text in their work.





Part of Speech Tagging

- **Part of Speech Tagging (POS) :**
 - the process of labeling each word in a text with its corresponding part of speech based on its definition and context.
 - POS of a word refers to its grammatical function within a sentence, such as noun, verb, adjective, adverb, preposition, pronoun, conjunction, and interjection.
 - POS tagging is important for text classification, information retrieval, and machine translation.
 - POS tagset: a collection of all POS tags used in a corpus

We learn Python

POS

- We: PRP (pronoun, personal)
- learn: VBP (verb, present tense, non-3rd person singular)
- Python: NN (noun, singular or mass)

Named Entity Recognition

Named Entity Recognition (NER): a subtask of information extraction

- aims to identify and classify named entities in text into predefined categories such as persons, organizations, locations, and other types.
- important in question-answering systems, text summarization, and information retrieval.

OpenAI and Google DeepMind lead cutting-edge research and development in the field of generative AI.

NER

- OpenAI ORG
- Google DeepMind ORG
- AI PRODUCT

Text representation

Bag of words (BoW): a simple and popular representation of text data that treats each document as a collection or "bag" of individual words, without considering their order or context.

Document 1 vector:

[2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

the quick brown fox jumps
over the lazy dog

the dog chased the fox

the fox is quick and agile

Vocabulary: ["the", "quick",
"brown", "fox", "jumps",
"over", "lazy", "dog",
"chased", "is", "and",
"agile"]

Text representation

Vectorisation: is a process of converting textual data into numerical vectors that can be used as input for machine learning algorithms.

Term occurrence

the quick brown fox jumps
over the lazy dog

the dog chased the fox

the fox is quick and agile

Vocabulary: ["the", "quick",
"brown", "fox", "jumps",
"over", "lazy", "dog",
"chased", "is", "and",
"agile"]

'the': 2, 'quick': 1, 'brown': 1, 'fox': 1,
'jumps': 1, 'over': 1, 'lazy': 1, 'dog': 1

Text representation

Vectorisation: is a process of converting textual data into numerical vectors that can be used as input for machine learning algorithms.

- Using a BoW model, a vectoriser can count the frequency of each word in a document and represents the document as a vector of word frequencies.

Binary term occurrence

the quick brown fox jumps
over the lazy dog

the dog chased the fox

the fox is quick and agile

Vocabulary: ["the", "quick",
"brown", "fox", "jumps",
"over", "lazy", "dog",
"chased", "is", "and",
"agile"]

'brown': 1, 'lazy': 1, 'fox': 1, 'jumps': 1,
'is': 0, 'dog': 1, 'chased': 0, 'agile': 0,
'quick': 1, 'and': 0, 'the': 1, 'over': 1

Text representation

TF-IDF the frequency of each word weighted by its frequency across all documents.

TF = (freq. of word in a document) / (#words in a document)

A higher TF for a word indicates that the word appears more frequently in the document.

IDF = $\log(\frac{\# \text{documents}}{\#\text{documents containing the word}})$

The higher the IDF of a word, the rarer it is in the corpus.

TF-IDF (term frequency * inverse document frequency)

a higher TF-IDF generally indicates that a word is more important in the given document, as well as in the overall corpus

the quick brown fox jumps
over the lazy dog

the dog chased the fox

the fox is quick and agile

'the': 2/9, 'quick': 1/9,
'brown': 1/9, 'fox': 1/9,
'jumps': 1/9, 'over': 1/9,
'lazy': 1/9, 'dog': 1/9

'the': $\log(\frac{3}{3})=0$, 'quick': $\log(\frac{3}{2})=0.405$, 'brown': $\log(\frac{3}{1})=1.099$,
fox: $\log(\frac{3}{2})=0.176$

'the': $2/9 * 0$, 'quick': $(1/9) * 0.405 = 0.045$, 'brown': $(1/9) * 1.099 = 0.122$, 'fox': 0.020 ...