Master of Data Science and Innovation

36118 Applied Natural Language Processing (ANLP)

Session 3

Session slides and notebooks are on Canvas!

Dr. Antonette Shibani

Senior Lecturer, MDSI

So far…

- ✓ Core NLP concepts
- ✓ Basic Python for NLP
- ✓ Exploratory text analysis
- ✓ Data pre-processing (tf-idf, stemming, stop words removal)
- ✓ Visualisation and Storytelling for text data
- ✓ Unsupervised techniques
  - Topic modeling
  - Clustering

Assignment 1: Text Analysis

Let’s start with our weekly check-in

Instructions to join Menti are provided in class

Agenda for today’s session

- NLP applications
- AT2 brief release
- Machine learning approaches - Supervised

(Text classification algorithms)

- Sentiment analysis
- Extended topic: Text summarization

Applied Natural Language Processing (NLP)

- How NLP is applied in the real-world

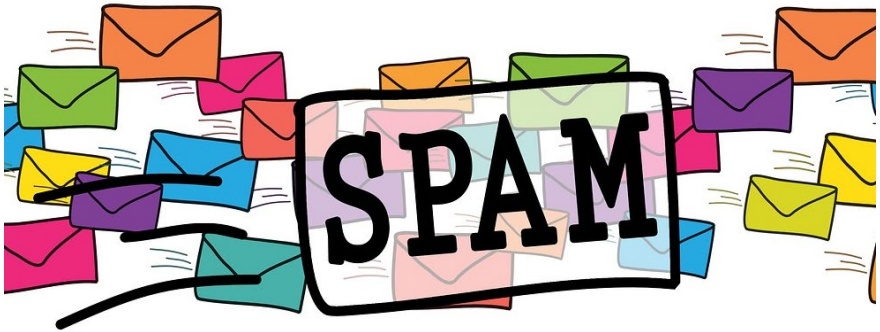
Foundational theory +

Practical uses and applications

NLP techniques and their applications

Text classification

- A supervised learning technique used to classify text into categories (predict target category)
- The set of categories are pre-defined, and is usually based on human annotation of text



E.g. Categorising emails as Spam/ Not spam

Sentiment analysis

- A form of classification to categorize people's opinions based on sentiments shown in text
- Commonly used to check positive, negative, neutral sentiments, but can also detect a wide variety of emotions. E.g. Sentinet

E.g. Classifying hotel reviews as positive, negative or neutral

Sentiment analysis

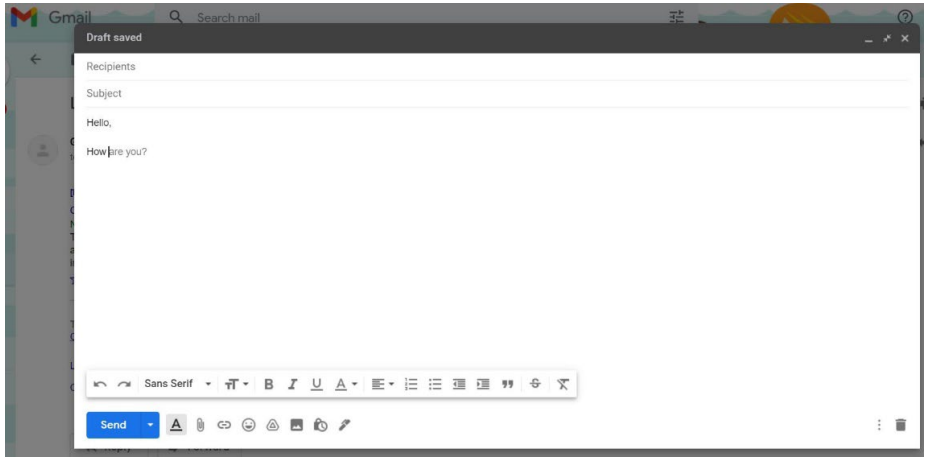
- A form of classification to categorize people's opinions based on sentiments shown in text
- Commonly used to check positive, negative, neutral sentiments, but can also detect a wide variety of emotions.

E.g. Classifying hotel reviews as positive, negative or neutral

Aspect-Based Sentiment Analysis allows a more granular form that identifies **specific aspects or features** of a product, service, or topic and determining the **sentiment expressed**.
E.g. The reviewer liked the battery life but disliked the screen resolution

Auto completion

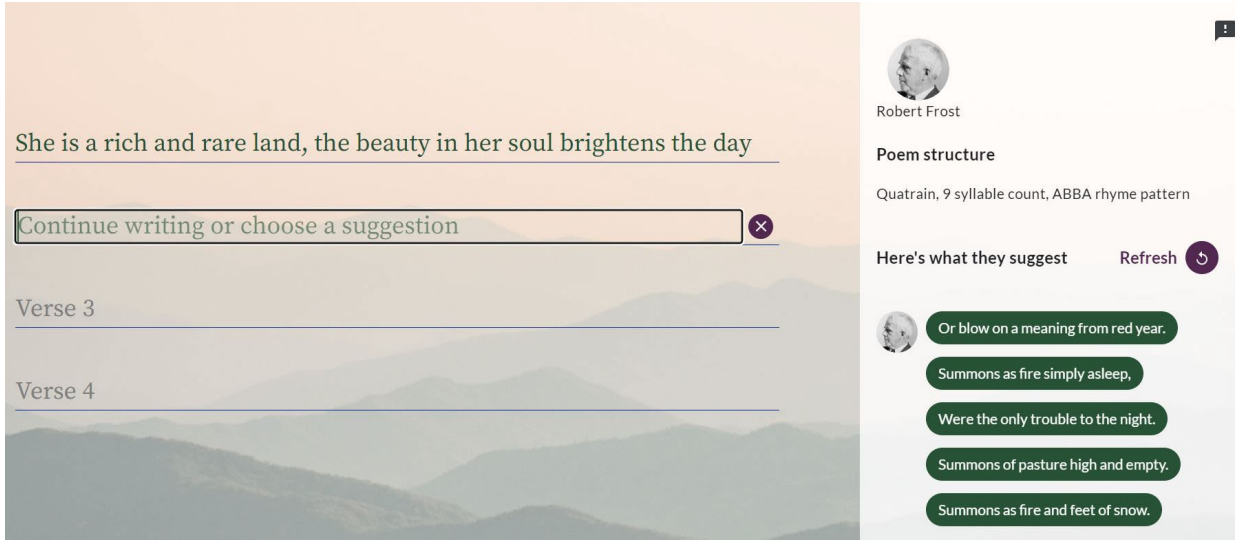
- Predicting and recommending next words for the text
- Used for text composition, search query completion, etc.



E.g. Smart compose in Gmail

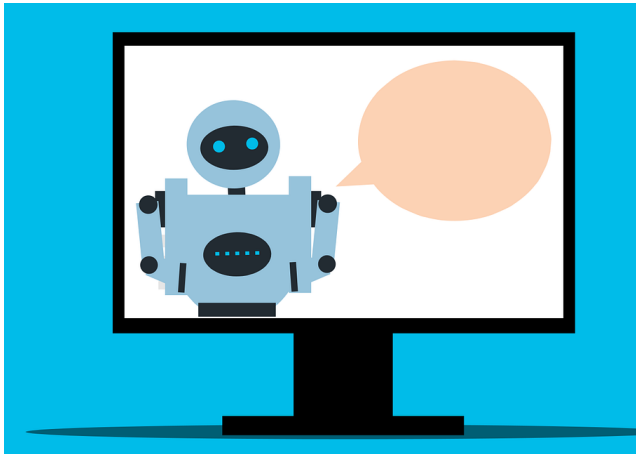
Automated text generation

- Auto generate essays, poems, code, etc.
- E.g. OpenAI's GPT-3 Verse-by-verse from Google, ChatGPT



Chat assistants

- Incorporate natural language understanding, natural language generation, speech-to-text recognition etc.

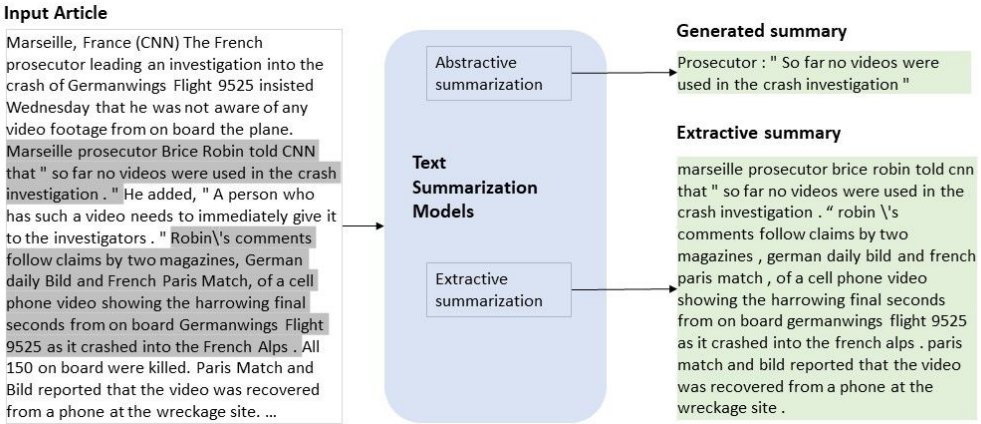


E.g. Chat bots for customer support, Siri/ Alexa smart assistants, ChatGPT

Retrieval Augmented Generation (RAG) chatbots can combine information retrieval with text generation! E.g. Finding appropriate policies from a set of source documents via chat

Text summarisation

- Automatically extracts useful information and phrases from a text and use that to construct a summary of the original text.
- Makes use of techniques such as neural networks and graph NLP



E.g. Summarising articles, financial reports

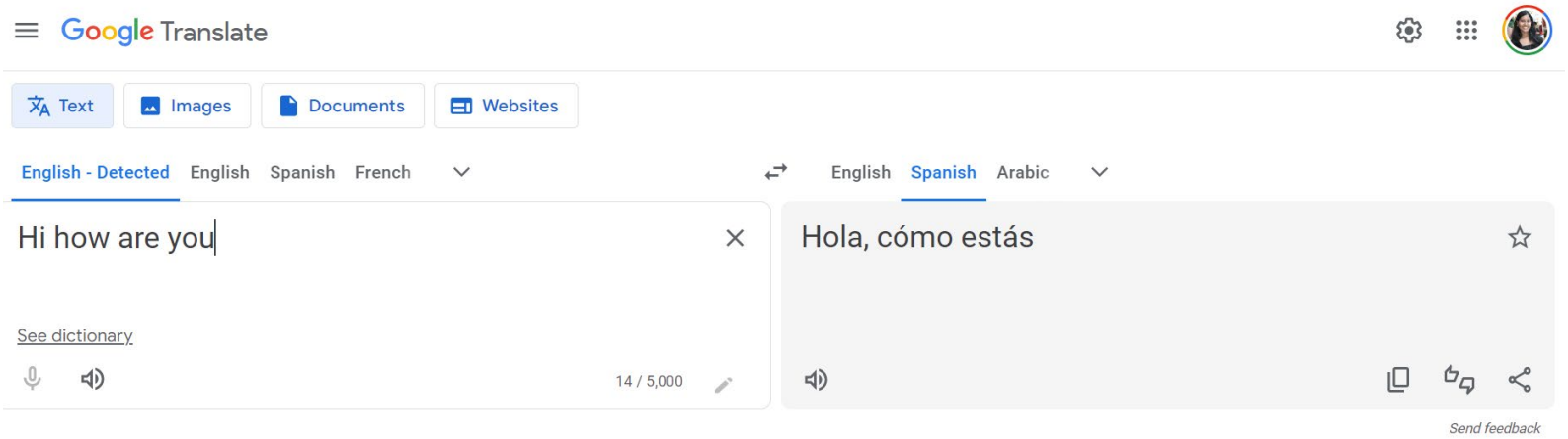
Clustering

- Unsupervised technique where a set of similar objects are grouped to a cluster
- Used in survey analysis, other open texts to find common themes
- Similar: Topic modeling

E.g. Grouping similar complaints from open-ended survey responses

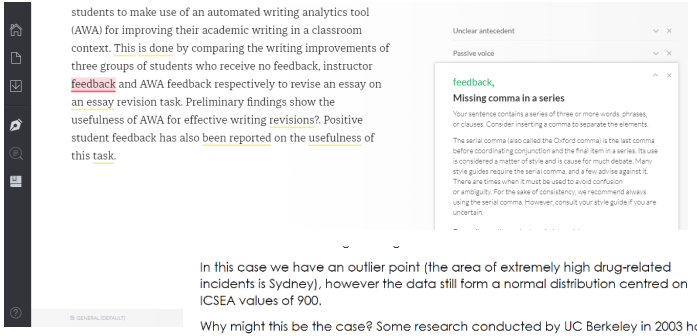
Translation services

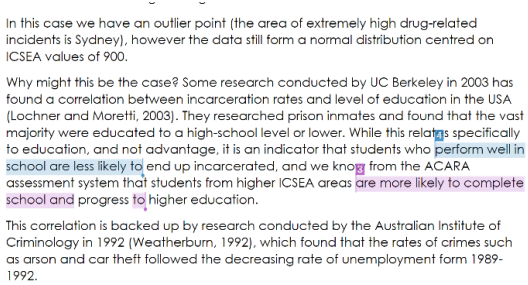
- Translate (and trans-literate_ text from one language to another
- Mostly using deep learning and machine learning

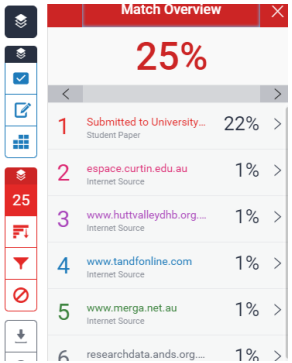


Writing analytics

- Analysing writing for spelling, grammar, automated scoring/ feedback of essays, plagiarism detection
- From NLP rules to machine learning







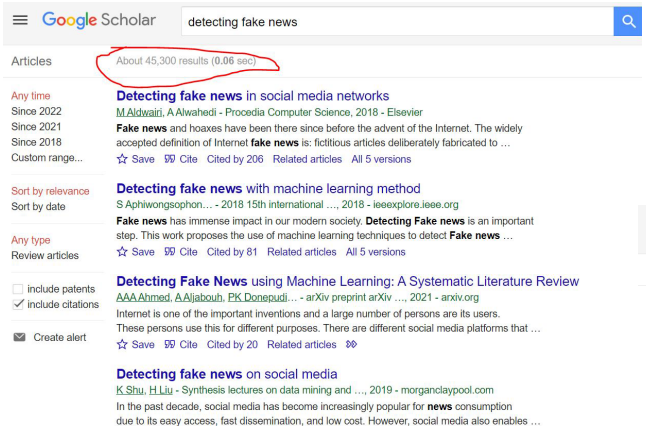
Custom applications

Specific applications that solve a particular problem in a niche

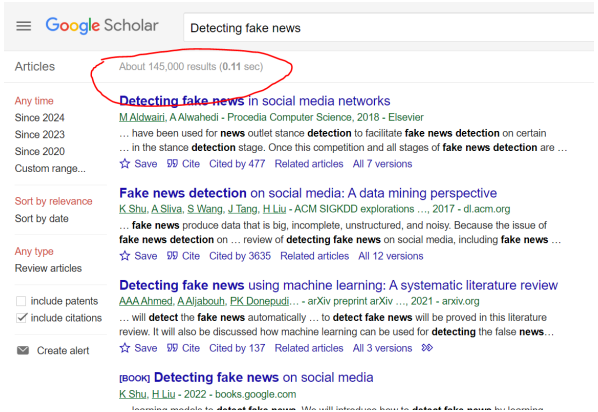
E.g. Academic discipline mapping (Citation graphs: See http://paperscape.org/)

Custom applications to solve problems/ research topics

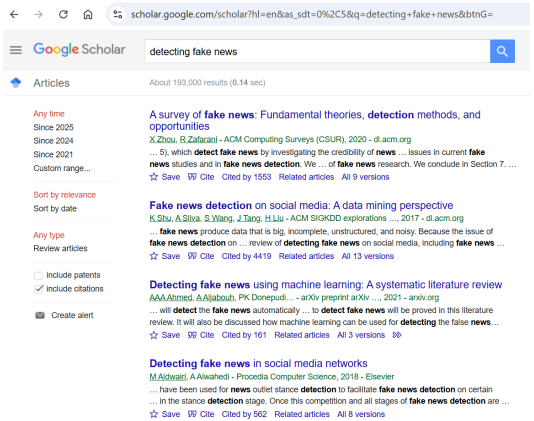
E.g. Detecting fake speech, hate speech and cyberbullying in social media



In 2023…

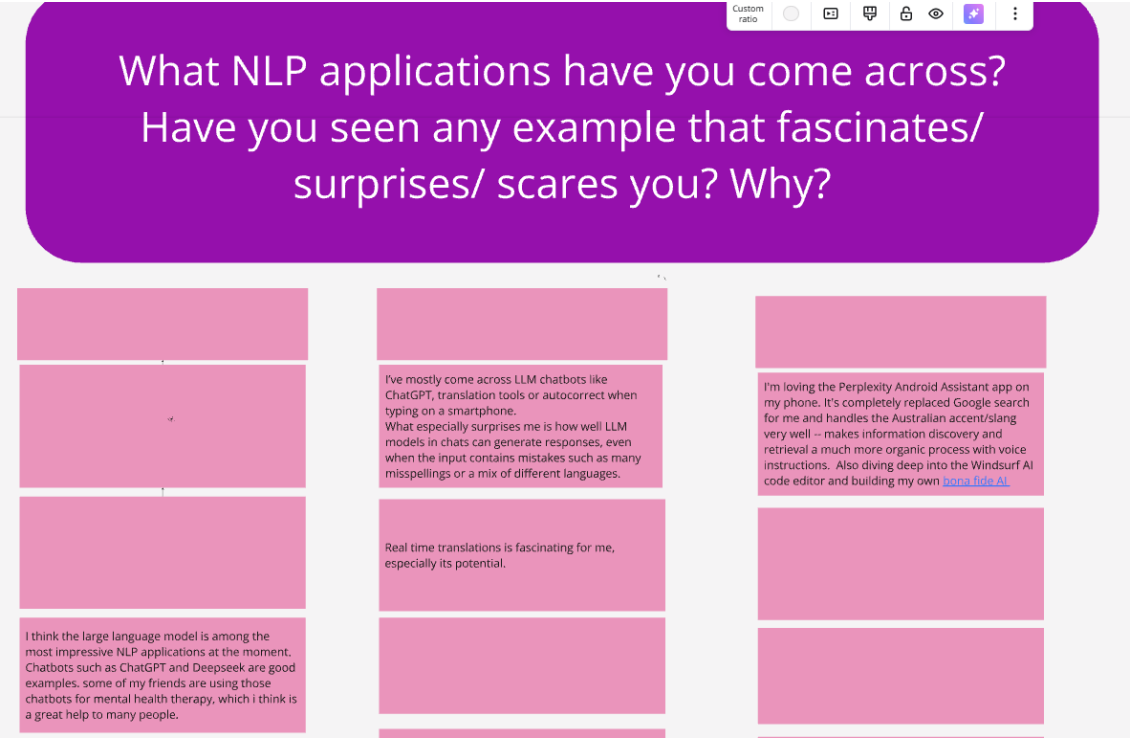


145,000 results as of 12 Aug 2024



193,000 results as of 17 Aug 2025

Applications you've come across….



Examples from sectors

Knowledge Management

Searchable databases, Q&A systems

- Healthcare
- Legal
- Finance
- Education
- Manufacturing
- Government
- Telcom
- Retail
- ….

Sentiment analysis

Analysing product reviews, trends, social media

- Tourism
- Hospitality
- Entertainment
- Sports
- Manufacturing
- Airlines
- Beauty
- Disaster mgmt
- Government
- ….

Chat bot

Customer support, Personal assistant

- Banking
- E-commerce
- Insurance
- Real estate
- Education
- ….

You know the drill….

Research

….

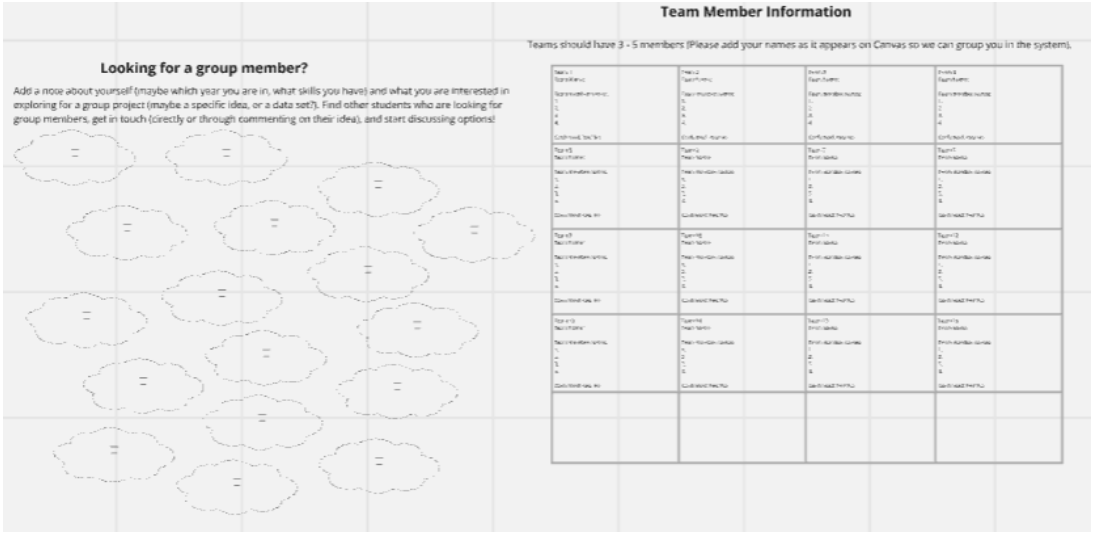
AT2 Release

Assignment 2

Group project (4 - 5 members)

Step 1 - Build your team

(And let us know so we can add you on Canvas)



Step 2 – Complete your project

(Don’t forget the peer review component)

This is an important contribution to your own DS portfolio, **spend a lot of time on it**! (It has more weeks allocated for a reason)

AT2 Assignment brief – Detailed instructions are on Canvas

Group projects in Assignment 2 are about taking your NLP projects to the real-world. The focus should be on **building end-user applications** that benefit user groups for specific purposes. To this end, you are encouraged to extend prior work on models that are already available open-source (if available) to build on the work of the community. You are **free to choose the technical approaches** you would like to implement in the back end of your application - for example, exploratory data analysis, machine learning, or large language models, as long as your choice is justified in your report. Comparison between traditional and newer approaches while not mandatory to discuss will be highly favoured.

Assignment 2

- Open-ended project – Build what interests you (and your team)
- Ideally has a full NLP pipeline (Data acquisition, pre-processing, exploratory analysis, model building, evaluation, reporting), but you may also focus on the main application by using an existing model/ API.
- Review the Detailed Assessment Brief for requirements
  - AT2 Part A Poster – Submission due 24 Sep
  - AT2 Part B – Submission due 13 Oct (Presentation 20 Oct at the final showcase)

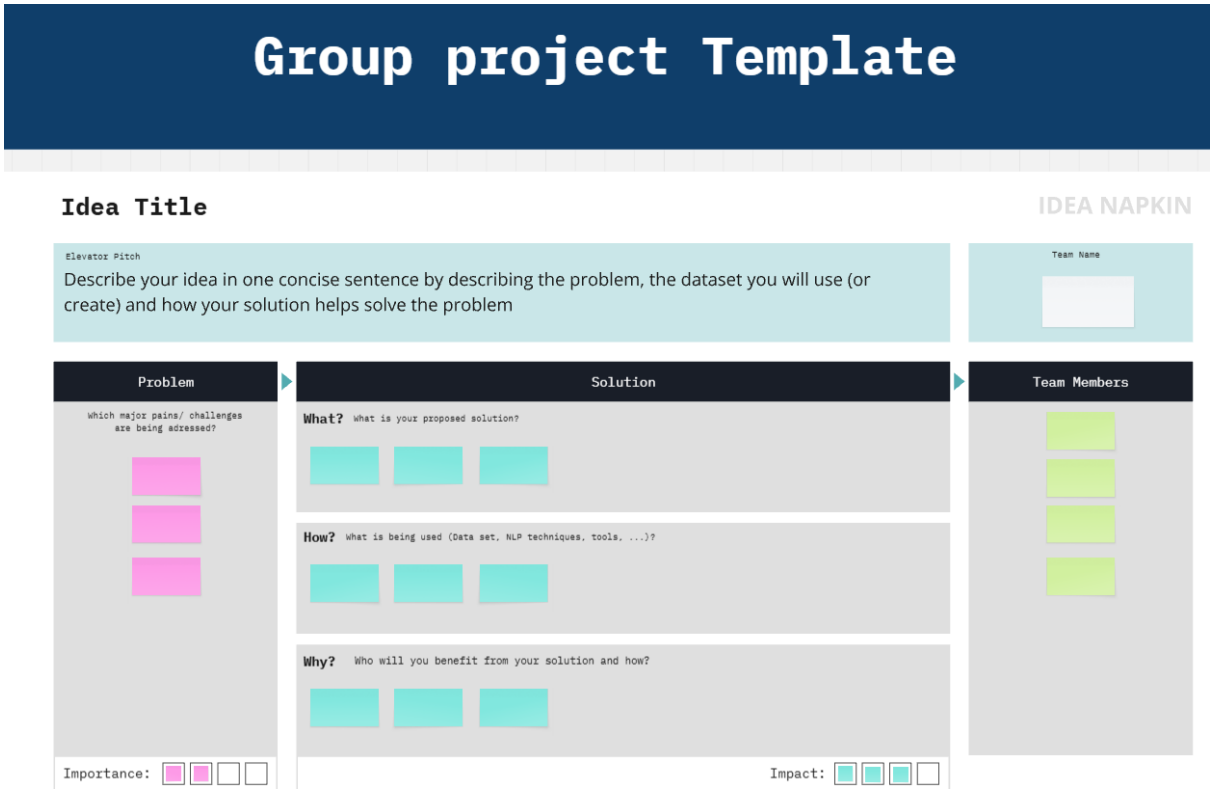
Reminder: Your assignment should not have been submitted elsewhere!

AT2 steps

- Start by forming your team of 4-5 members
- Think about a topic/ application that interests your team (is there a problem you'd like to contribute to?)
- The main requirement is that there is a user-facing application which involves an NLP model or task. You are also free to use APIs for accessing LLMs.

- The teaching team will provide certain problem statements in the next session if you don't have a project by then!

Ideas Canvas on Miro – A starter resource for your team



Some AT2 projects from past cohorts (for inspiration only)

- UTS Course Recommendation Chatbot based on career goals
- Automated for Qualitative Research
- Empathetic Chatbots for Personalised Support
- Resume Optimisation Tools
- Review/ Sentiment analysis tools
- Green House Gases Compliance Assistant
- Trip and itinerary planners
- Twitter political posts analysis tool
- Multilingual tools

Not recommended

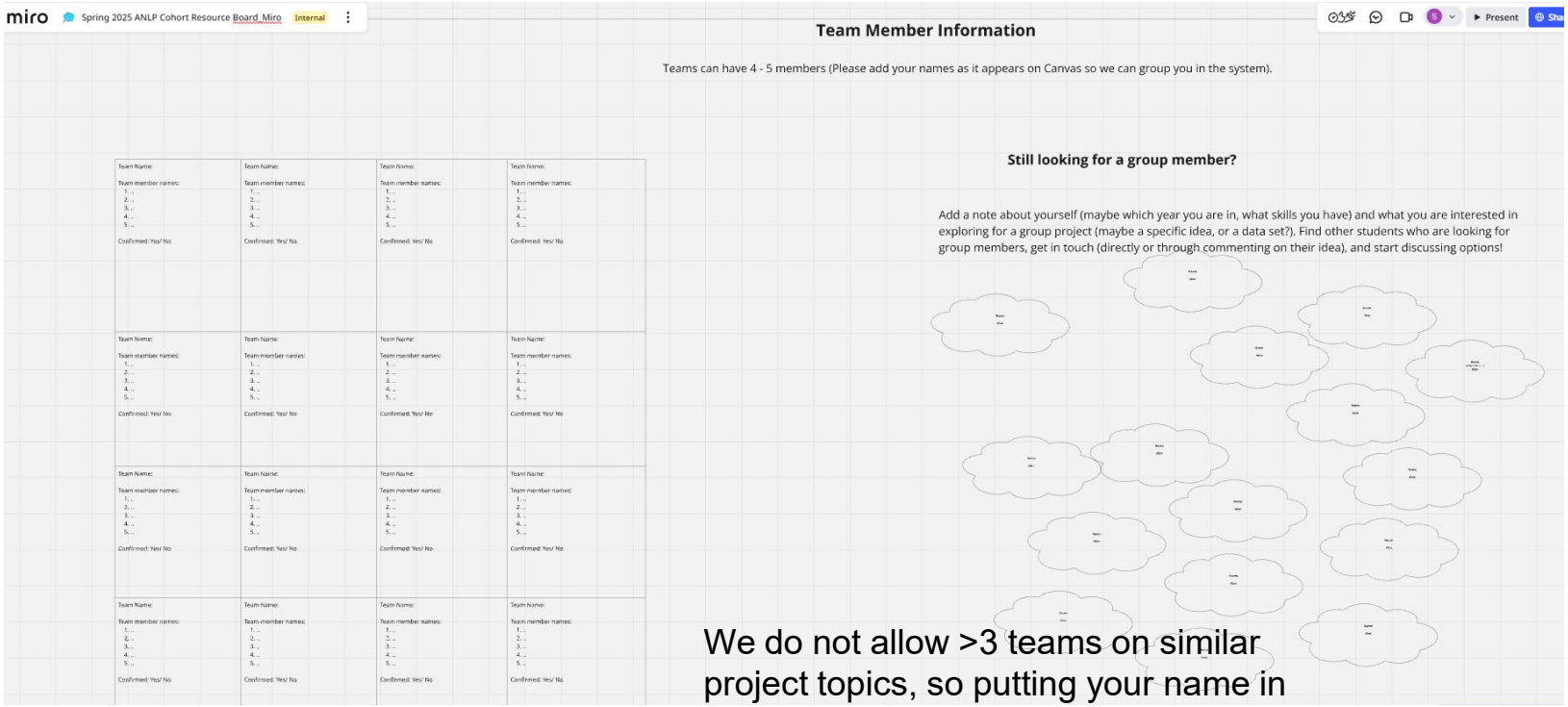
See my LinkedIn posts for past winners of awards: 2024 Showcase and Awards – https://www.linkedin.com/posts/activity-7254768181293035522-7fly/

Autumn 2025 ANLP Showcase and Awards - https://www.linkedin.com/posts/activity-7328682647365345280-ZMbi/

AT2 Reminders

- We are not looking for the prettiest UI's here, but what is helpful for the user (You can keep the UI features basic and focus on the back-end)
- You cannot submit Jupyter/ Colab notebooks for this assignment. It has to be a full Python application, preferably built on Github collaboratively, tracking edits. This is a great opportunity to gain skills for Git collaboration, which adds to your Github profile.
- It is highly recommended, but optional to host your application on the web. Making it available adds to your portfolio, but this is up to you!

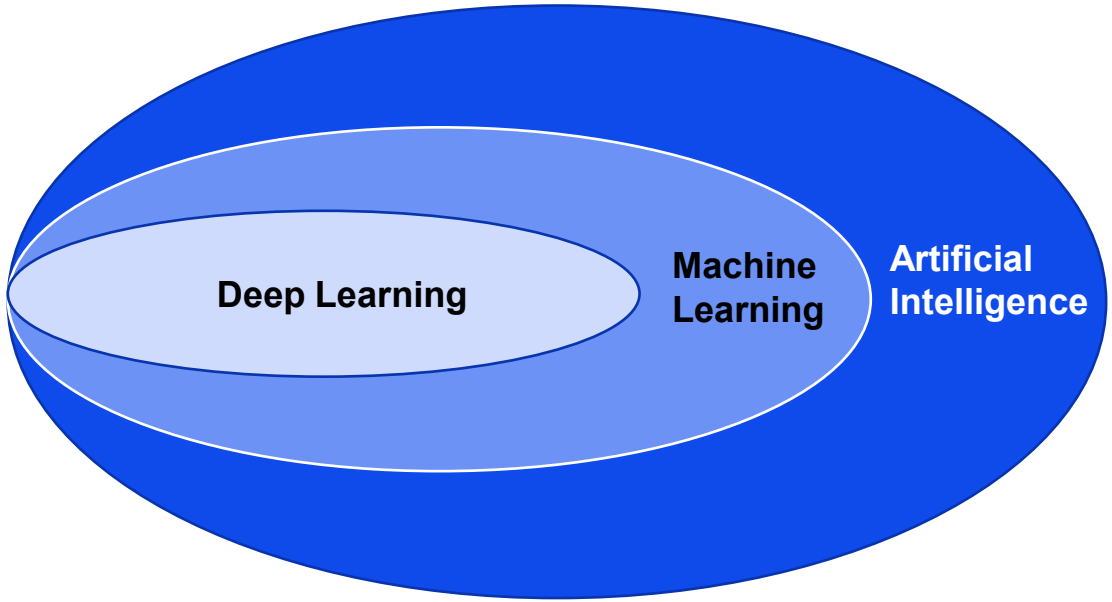
Add your team names and basic project info on the Miro board (before the next session, if you have it)



We do not allow >3 teams on similar project topics, so putting your name in early finalises your slot!

Machine Learning using text

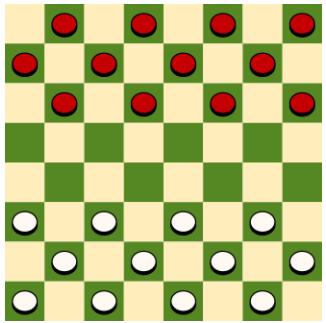
Where does Machine Learning sit in the realm of AI?



Machine Learning

Arthur Samuel (1959)

Field of study that gives computers the ability to learn without being explicitly programmed



Machine learning - Types

- **Unsupervised**
- **Supervised**
- Reinforcement learning (Some part of the data is labelled, then provided additional feedback with more labels; no supervisor, only a reward signal)
- Active learning (obtain new labels dynamically, defining an algorithmic strategy to maximize the usefulness of the new data points)
- Transfer learning (experience gained in learning to perform one task can help improve learning performance in a related, but different, task)

Supervised machine learning

- The machine is provided with labelled datasets to train algorithms to classify data or predict outcomes accurately
- We know what we want the model to predict

Examples:

- Regression (Numerical)
- Classification (Categorical)

Image: kidzinc.com.au

Supervised machine learning

442	Does the Covid vaccine made with aborted human fetuses	0
443	Is the vaccin toxic	0
444	Did the Pfizer vaccine cause me to start having 4 1\2 hour erectio	0
445	How many people lived after taking the vaccine?	0
446	should I worry about long term health issue cause by vaccine?	0
447	can I talk to a human?	1
448	Are you anti vaccine	1
449	not your welcome stupid	1
450	You don't understand much do you?	1
451	Let me talk to someone who can actually understand a question.	1
452	You need to upgrade your programming as the plan is to vaccin	1
453	Oh, you understand, you're just unwilling to discuss	1
454	I see you are more concerned with convincing people to get vacc	1
455	You are hiding information deliberately.	1
456	Waste of time. Error message said bad syntax. Poorly designed	1
457	Is it ethical to unleash chatbots to change peoples minds about v	1
458	AI chatbots are really stupid	1
459	Are you limited with responses?	1
460	I think you cannot answer my question. Goodbye.	1
461	How can i trust you?	1
462	Google and Alexa are smarter than you. You need to self destruc	1
463	Are you idiot?	1
464	Why are you trying to kill everyone?	1
465	Your answer isn't detailed enough	1
466	Is it safe to get the vaccine when you are pregnant?	2
467	Cloned cells	2
468	I'm healthy. Why do I need the vaccine?	2
469	How many pregnant women were in the trials	2
470	Who should not get the covid vaccine?	2
471	thanks! what's the deal with nicki minaj and fertility?	2

What we are looking to predict

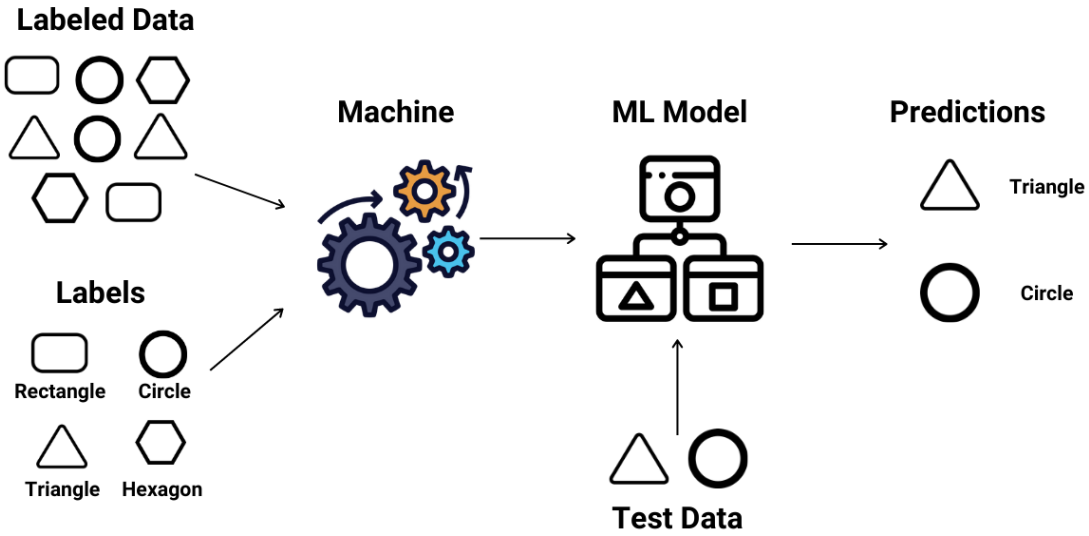


Image source: https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning

Some applications of supervised ML (using text)

Regression (predicting numbers)

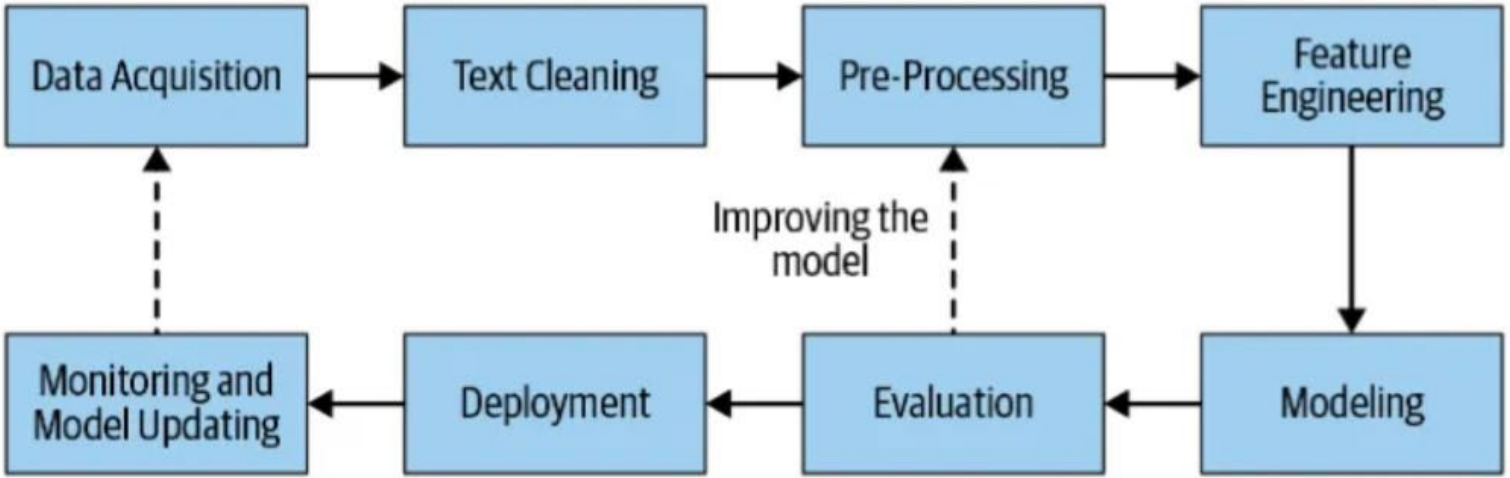
- Predicting house price from description text
- Detecting emotion level from text (E.g. a score between 0 - extremely negative and 10 - extremely positive)

Text classification (predicting categories):

- Detecting spam/ not spam in emails
- Social media monitoring: Sentiment analysis (Positive/ Negative/ Neutral)
- Tagging customer support tickets

Text classification (Supervised Machine learning)

NLP pipeline



Data Sources for acquisition

Collect your own data:

- Web scraping: https://towardsdatascience.com/how-to-scrape-more-information-from-tweets-on-twitter-44fd540b8a1f
- News articles: https://mediacloud.org/
- Through APIs: E.g. Wikipedia has a freely available API that can be accessed as below

Use coded data sets available for NLP tasks: https://github.com/niderhoff/nlp-datasets

https://www.kaggle.com/datasets?tags=13204-NLP

```
!pip install wikipedia-api

import wikipediaapi

wiki_wiki = wikipediaapi.Wikipedia(
        language='en',
        extract_format=wikipediaapi.ExtractFormat.WIKI
)

p_wiki = wiki_wiki.page('Natural language processing')
print (p_wiki.text[:500], '....')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting wikipedia-api
  Downloading Wikipedia_API-0.5.8-py3-none-any.whl (13 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from wikipedia-api) (2.27.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->wikipedia-api) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->wikipedia-api) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->wikipedia-api) (1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->wikipedia-api) (3.4)
Installing collected packages: wikipedia-api
Successfully installed wikipedia-api-0.5.8
Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, ....
```

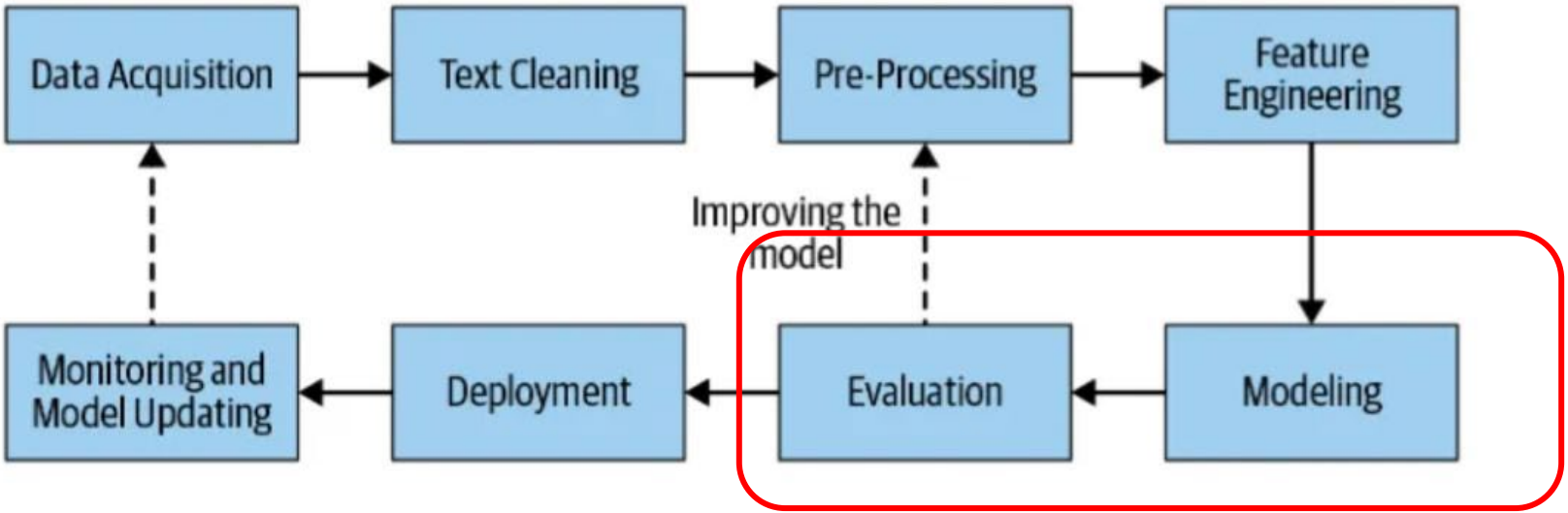
Data Annotation

- Core step for text classification
- Defining the construct is important (The coding scheme matters a lot!)
- Measure inter-rater reliability of human coding

  E.g. Cohen's Kappa measure: https://www.statisticshowto.com/cohens-kappa-statistic/

442	Does the Covid vaccine made with aborted human fetuses	0
443	Is the vaccin toxic	0
444	Did the Pfizer vaccine cause me to start having 4 ½2 hour erectio	0
445	How many people lived after taking the vaccine?	0
446	should I worry about long term health issue cause by vaccine?	0
447	can I talk to a human?	1
448	Are you anti vaccine	1
449	not your welcome stupid	1
450	You don't understand much do you?	1
451	Let me talk to someone who can actually understand a question.	1
452	You need to upgrade your programming as the plan is to vaccin	1
453	Oh, you understand, you're just unwilling to discuss	1
454	I see you are more concerned with convincing people to get vacc	1
455	You are hiding information deliberately.	1
456	Waste of time. Error message said bad syntax. Poorly designed	1
457	Is it ethical to unleash chatbots to change peoples minds about v	1
458	AI chatbots are really stupid.	1
459	Are you limited with responses?	1
460	I think you cannot answer my question. Goodbye.	1
461	How can i trust you?	1
462	Google and Alexa are smarter than you. You need to self destruc	1
463	Are you idiot?	1
464	Why are you trying to kill everyone?	1
465	Your answer isn't detailed enough	1
466	Is it safe to get the vaccine when you are pregnant?	2
467	Cloned cells	2
468	I'm healthy. Why do I need the vaccine?	2
469	How many pregnant women were in the trials	2
470	Who should not get the covid vaccine?	2
471	thanks! what's the deal with nicki minaj and fertility?	2

Text classification



Classification using hand-coded rules

E.g. Email spam detection

Rules based on combinations of words or other features

- spam: black-list-address OR (“dollars” AND “have been selected”)

Accuracy can be high

- If rules carefully refined by expert

But building and maintaining these rules is expensive

Supervised machine learning

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

Output:

- a learned classifier $\gamma:d \rightarrow c$

Baseline models for text classification

- Many variants are available, but they follow the general trend of learning a set of rules from training examples and predict a label to a given text
- Most common algorithms used on text data:
  - Naïve Bayes
  - Logistic Regression
  - Decision Trees
  - Support Vector Machines

Why non-deep models?

- You have a unique ML task (Classical tasks are commoditized and pre-trained models are easily available for fine-tuning)
- You can encode your prior knowledge using relatively smaller data sets. Deep models require lots of data!
- Great starters for new problems and classifiers not previously studied

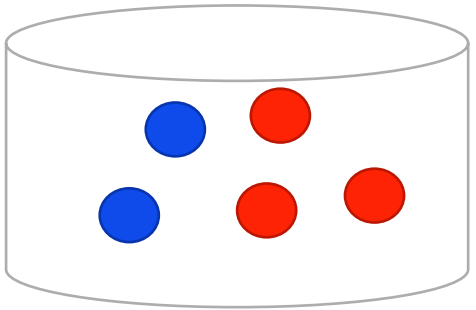
Naïve Bayes model

- Based on Bayes Theorem and probability theory to predict the category of a text
- Binary classifier (E.g. **Spam or not Spam**), can also be extended to more classes using Multinomial NB
- Extremely fast and simple
- Often suitable for very high-dimensional data and very well-separated categories
- Few tunable parameters -> quick-and-dirty baseline for any classification problem

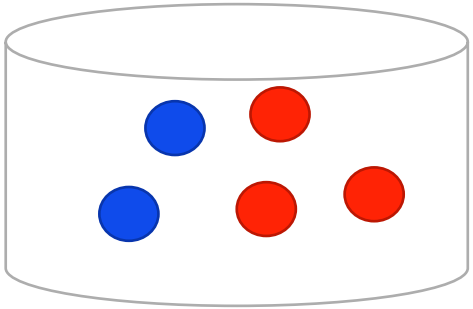
A Math question

A bag contains 2 blue and 3 red marbles. What is the probability of picking:

1. A blue marble followed by a red marble?
2. A red marble followed by a blue marble?



- Probability of picking blue, P(B) = 2/5 = 0.4
- Probability of picking red given that we already picked blue,
- **P(R|B)** = 3/4 = 0.75 (**Conditional Probability**)
- P(Blue and Red)
- **P(B ∩ R)** = P(B) * P(R|B) = 0.4 * 0.75 = **0.3**



- Probability of picking red, P(R) = 3/5 = 0.6
- Probability of picking blue given that we already picked red,
- **P(B|R**) = 2/4 = 0.5 (**Conditional Probability**)
- P(Red and Blue)
- **P(R ∩ B)** = P(R) * P(B|R) = 0.6 * 0.5 = **0.3**

Bayes theorem

$$P(B \cap R) = P(R \cap B)$$

Or

$$\mathbf{P(B) * P(R|B) = P(R) * P(B|R)}$$

(Bayes Theorem)

…but why is this useful?

Naïve Bayes algorithm

- The **Naïve Bayes Algorithm** in machine learning assumes that the features (predictors) of a dataset are **conditionally independent** of each other.
- For example, if $x_1, x_2 \ldots x_n$ are words in an email, the conditional probability that the words are collectively indicative that the message is spam (y) is given by the product of the individual conditional probabilities:

$$P(x_1 \ldots x_n | y) = \prod_{i=1}^{n} p(x_i | y)$$

- This is same as assuming, for example, that the word "*congratulations*" in an email is independent from "*winner*" although it is most likely not.
- **The probability of an email being spam $P(y|x_1 \ldots x_n)$** –which is what we are interested in - is then easily calculated using Bayes theorem.

Spam/ Ham

Let’s build a classifier for predicting if an email is spam/ ham (non-spam)!

P(Spam|Email) = probability that the email is spam given its content

P(Ham|Email) = probability that the email is ham given its content

The basic **decision rule** is to classify an email as spam if: P(Spam|Email) > P(Ham|Email)

Decision boundary is the point where P(Spam|Email) = P(Ham|Email). It is the threshold that separates the spam and ham classifications (which can be adjusted to make the model more conservative/ aggressive)

The decision boundary can be expressed as a log-likelihood ratio: log(P(Spam|Email)) - log(P(Ham|Email)) = 0

Why use log functions?

Logarithms **prevent *underflow*** in probability calculations in ML algorithms!

- Underflow occurs when a computation produces a number too small to be represented in the computer's floating-point format.

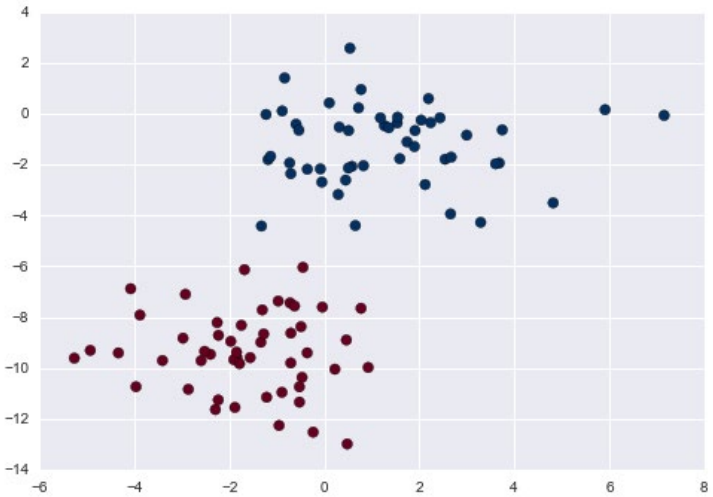
Example: Consider multiplying these probabilities: 0.01 * 0.02 * 0.03 * 0.04 * 0.05 Result: 0.0000000012 - Most computers can't represent numbers this small accurately.

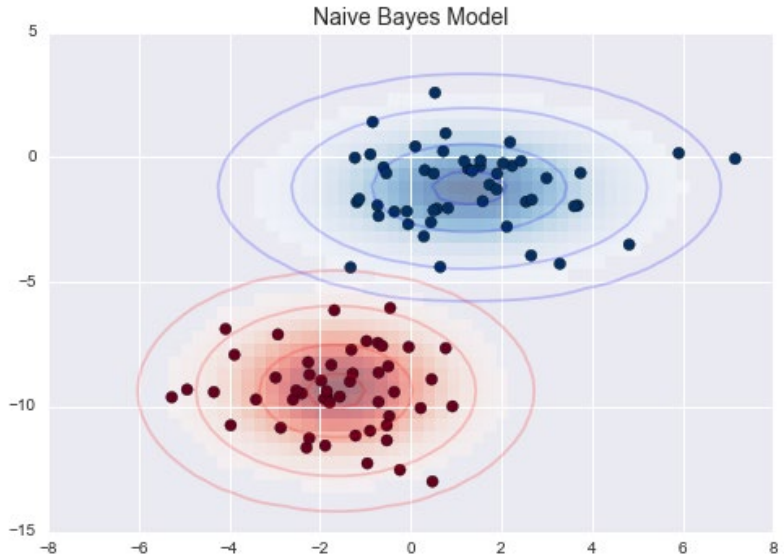
Logarithms convert multiplication into addition: $\log(a * b) = \log(a) + \log(b)$

E.g. $\log(0.01) + \log(0.02) + \log(0.03) + \log(0.04) + \log(0.05) = -4.60 + (-3.91) + (-3.51) + (-3.22) + (-2.99) = -18.23$

Log values of probabilities are always negative or zero (since $0 < p(x) \leq 1$)

Naïve Bayes model intuition





Finding the best Gaussian fit

One extremely fast way to create a simple model is to assume that the data is described by a Gaussian distribution with no covariance between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all you need to define such a distribution!

https://www.oreilly.com/library/view/python-data-science/9781491912126/

Email 1 (Spam): "Get rich quick money"
Email 2 (Ham): "Hey friend, how are you?"
Email 3 (Spam): "Win money now"
Email 4 (Ham): "Meeting at work tomorrow"
Email 5 (Spam): "Click here for free money"

Email	Class	rich	quick	money	friend	win	meeting	work	click	free
1	Spam	1	1	1	0	0	0	0	0	0
2	Ham	0	0	0	1	0	0	0	0	0
3	Spam	0	0	1	0	1	0	0	0	0
4	Ham	0	0	0	0	0	1	1	0	0
5	Spam	0	0	1	0	0	0	0	1	1

Prior probabilities
P(Spam) = 3/5 = 0.6
P(Ham) = 2/5 = 0.4

Conditional probabilities using Laplace smoothing (+1)
For Spam:
P(rich|Spam) = (1+1) / (3+9) = 2/12
P(quick|Spam) = (1+1) / (3+9) = 2/12
P(money|Spam) = (3+1) / (3+9) = 4/12 ...
For Ham:
P(rich|Ham) = (0+1) / (2+9) = 1/11
P(quick|Ham) = (0+1) / (2+9) = 1/11
P(money|Ham) = (0+1) / (2+9) = 1/11 ...

Classify a new email "Quick money from a friend"
BoW representation: [0, 1, 1, 1, 0, 0, 0, 0, 0]

Calculate log probabilities:
For Spam: log P(Spam|email) = log P(Spam) + log P(quick|Spam) + log P(money|Spam) + log P(friend|Spam) = log(0.6) + log(2/12) + log(4/12) + log(1/12) ≈ -0.51 + (-1.79) + (-1.10) + (-2.48) ≈ -5.88
For Ham: log P(Ham|email) = log P(Ham) + log P(quick|Ham) + log P(money|Ham) + log P(friend|Ham) = log(0.4) + log(1/11) + log(1/11) + log(2/11) ≈ -0.92 + (-2.40) + (-2.40) + (-1.70) ≈ -7.42

Since log P(Spam|email) > log P(Ham|email), we classify this email as Spam

Why “Naïve”?

- Simple (“naïve”) classification method based on Bayes rule

- we make very naive assumptions about the generative model for each label, we can find a rough approximation of the generative model for each class, and then proceed with the Bayesian classification

- Relies on very simple representation of document - Bag of words

Multinomial Naïve Bayes

- The multinomial distribution describes the probability of observing counts among a number of categories
- The idea is the same as before, except that instead of modeling the data distribution with the best-fit Gaussian, we model the data distribution with a best-fit *multinomial distribution*

Using Sklearn

- H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

1.9.1. Gaussian Naive Bayes

`GaussianNB` implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

Training and test data

Training data - Labelled input data for our model to learn from

Test data - Unseen labelled data to test the model output

- All the coded data is suitable for training since it comes with labels. Let's assume we use the whole data set for training.
- How do we know our model works on new data?

Feature engineering

Cleaning and Pre-processing the data

- Removing empty rows
- Stop words removal
- Stemming/ Lemmatization

Text Representation:

- Bag of Words (BoW): Represents text as a collection of individual words, ignoring grammar and word order - **CountVectorizer**
- Term Frequency-Inverse Document Frequency (TF-IDF): Represents the importance of each word in the document and the corpus as a whole, based on its frequency in the document and its rarity in the corpus – **Tfidf Vectorizer**
- Word Embeddings: Represents words as dense vectors in a high-dimensional space, where words with similar meanings are clustered together – **Word2vec**

Tf-idf vectorizer on sklearn

Fit the vectorizer on the data

```
[ ] # Import the necessary libraries

    import sklearn # Popular machine learning library
    from sklearn.feature_extraction.text import TfidfVectorizer # tf-idf vectorizer module
```

```
# First we need to append data from all the sets to create an entire corpus

# Create an empty array
X = np.array([], dtype=str)

# Append text from all the splits to the empty array
X = np.append(X, df_train["text"])
X = np.append(X, df_val["text"])
X = np.append(X, df_test["text"])
```

```
[ ] vectorizer = TfidfVectorizer() # Load the tf-idf vectorizer from sklearn
    vectorizer = vectorizer.fit(X) # Fit the vectorizer on the entire data
```

```
# Extract the vocabulary from the vectorizer
vocab = vectorizer.vocabulary_
print(f"Number of terms in the vocabulary = {len(vocab)}")
```

```
Number of terms in the vocabulary = 1928
```

This means that there are 1928 unique tokens in our data set

```
[ ] # Display a sample of the word => value mapping
    print("Sample Mapping")
    for idx, ele in enumerate(zip(vocab.keys(), vocab.values())):
        word, val = ele
        print(f"{word} => {val}")
        if idx == 4: break
```

```
Sample Mapping
symptomat => 1654
patient => 1263
got => 743
vaccin => 1797
die => 511
```

Fit the ML model

```
# Import the necessary libraries

from sklearn.linear_model import LogisticRegression # Logistic Regression Model
from sklearn.metrics import classification_report # Module to calculate performance metrics
from sklearn.metrics import confusion_matrix # Module to calculate the confusion matrix
from sklearn.metrics import ConfusionMatrixDisplay # Module to display the confusion matrix

import matplotlib.pyplot as plt # Module to help draw the confusion matrix
```

```
[ ] # Load the model
lr_model = LogisticRegression()

# Fit the model
lr_model.fit(X=X_train, y=y_train)

# Obtain the predictions for the validation data
pred_val = lr_model.predict(X_val)

# Get the classification report for the prediction
# It is a dictionary that contains various metrics and their values
# We choose accuracy and weighted average f1-score as our metrics of choice
results = classification_report(y_true=y_val, y_pred=pred_val, output_dict=True, zero_division=0) # Use zer

print(f"Accuracy = {results['accuracy']*100:>.2f} %")
print(f"Weighted Avg F1-score = {results['weighted avg']['f1-score']:>.4f}")
```

```
Accuracy = 77.18 %
Weighted Avg F1-score = 0.7582
```

```
[ ] # Obtain the confusion matrix
cm = confusion_matrix(y_val, pred_val)

# Uncomment the following line to just print the matrix to the standard output
# print(cm)

cm_display = ConfusionMatrixDisplay(cm) # Initiate a ConfusionMatrixDisplay object
cm_display.plot() # Plot the object
plt.show() # Display the plot
```

Model evaluation

- Intrinsic evaluation
- Extrinsic evaluation

Intrinsic evaluation

- Evaluate the model through performance metrics
- Understanding these metrics will allow you to see how good your classifier model is at analyzing texts
- You can evaluate your classifier over a fixed testing set, or by using cross-validation
- Example metrics: Accuracy, Precision, Recall, F-score

Train - Test split

- We separate out a small fraction of the available data as the test set
- We use the remaining data to build a suitable model or classifier (training data). Then we feed the test data to this model to calculate model performance
- Since we know the class of each element of the test data, we can tell how well the model does on the test data
- 80-20 splits are most common (but can also be 70-30 etc.)
- We can also have a **holdout** subset that provides a final estimate of the machine learning model's performance after it has been trained and validated. Holdout sets should never be used for improving or tuning algorithms!

Error analysis

- A commonly used visual evaluation method in classification is a confusion matrix
- It allows us to inspect the actual and predicted output for different classes in the dataset
- The name stems from the fact that it helps to understand how “confused” the classification model is in terms of identifying different classes
- A confusion matrix is in turn used to compute metrics:
  - Accuracy
  - Precision
  - Recall
  - F1 score

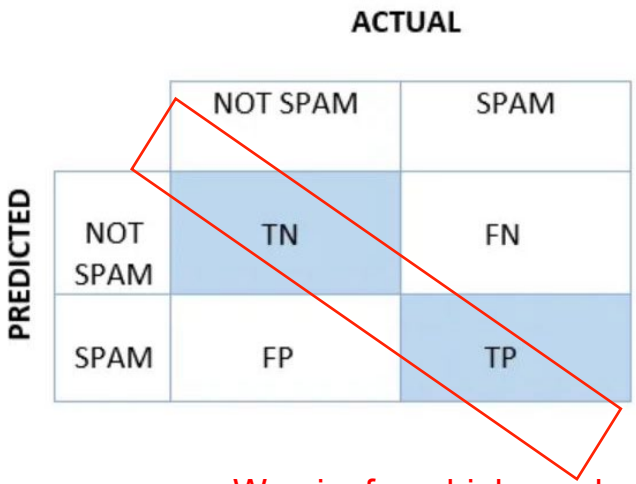
Confusion matrix

True Positive (TP) — model correctly predicts the positive class (prediction and actual both are positive).

True Negative (TN) — model correctly predicts the negati class (prediction and actual both are negative).

False Positive (FP) — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). FP is also called a **TYPE I** error.

False Negative (FN) — model wrongly predicts the positive class (predicted-negative, actual-positive). FN is also called a **TYPE II** error.



We aim for a high number of TN and TPs;

We can use FN and FP cases for diagnosis!

Evaluation metrics

- **Accuracy** - The number of correct predictions that the classifier has made divided by the total number of predictions. However, accuracy alone is not always the best metric to evaluate the performance of a classifier especially when categories are imbalanced.
- **Precision** - The number of correct predictions made by the classifier, over the total number of predictions for a given tag (including both correct or incorrect predictions). A high precision metric indicates there are less false positives.
- **Recall** - The number of texts that were predicted correctly, over the total number that should have been categorized with a given tag. A high recall metric means that there were less false negatives.
- **F1 score** - Combines the parameters of precision and recall to show how well your classifier is working. This metric is a better indicator than accuracy to understand how good predictions are for *all categories* in your model.

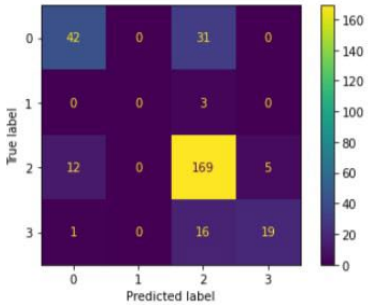
What do you think is the most useful metric is for Spam detection (Precision or Recall)?

Error analysis in scikit learn

```
# We choose accuracy and weighted average f1-score as our metrics of choice
results = classification_report(y_true=y_val, y_pred=pred_val, output_dict=True, zero_division=0) # Use zero_devision to control warnings

print(f"Accuracy = {results['accuracy']*100:>.2f} %")
print(f"Weighted Avg F1-score = {results['weighted avg']['f1-score']:>.4f}")
```

```
Accuracy = 77.18 %
Weighted Avg F1-score = 0.7582
```

```
# Obtain the confusion matrix
cm = confusion_matrix(y_val, pred_val)

# Uncomment the following line to just print the matrix to the standard output
# print(cm)

cm_display = ConfusionMatrixDisplay(cm) # Initiate a ConfusionMatrixDisplay object
cm_display.plot() # Plot the object
plt.show() # Display the plot
```



Overfitting

- Overfitting occurs when the classifier performs much worse on the test data than on the training data (high error rate)

E.g. Training accuracy = 85%, Test accuracy = 40%

- The trained features are too specific to the current training data
- We can restrict the machine-learning algorithm in some way to avoid overfitting

When a model overfits to the training data, it is *not generalisable*

Generalization

- The purpose of creating a model or classifier is not to classify the training set, but to *classify the data whose class we do not know* ***(unseen, new data set)***
- We want that data to be classified correctly, but often we have no way of knowing whether or not the model does so
- If the nature of the data changes over time, for instance, if we are trying to detect spam emails, then we need to measure the performance over time, as best we can

E.g. In the case of spam emails, we can note the rate of reports of spam emails that were not classified as spam.

Evaluation approach: Cross-validation

- Slicing up the training data into “folds” and hold out one fold each turn

Why should you cross-validate?

Cross-validation

E.g. In ten-fold cross validation, for instance, we'll split our training set into tenths. Then, as a first pass, we'll use the first 9 subsets as training data and treat the last one as our held-out test set

- This give us one measure of accuracy. We can do it again, though, by now taking subsets 1-8 and that tenth set, training a separate model, and testing it on subset 9
- By continuing this multiple times, we get a set of guesses at accuracy; (based on only some portions of your data)
- The last step is compiling the results of all subsets of data to obtain an average performance of each metric

Extrinsic evaluation

Evaluating the model performance on *the final objective/ business problem*

- For example, a regression model is built with the aim of ranking the emails of the users and bringing the most important emails to the top of the inbox, helping the users of an email service save time

How will you evaluate if the model meets its aims?

Extrinsic evaluation - Scenario

The regression model does well on the ML metrics but doesn’t really save a lot of time for the email service users

- Would we call such models successful?

No, because they failed to achieve their objectives

- How to carry out extrinsic evaluation?

Set up the business metrics and the process to measure them correctly at the start of the project

Logistic Regression

- Logistic regression is a statistical method used for predicting a ***binary outcome*** (yes/no, 1/0, true/false, positive/negative) based on one or more independent variables

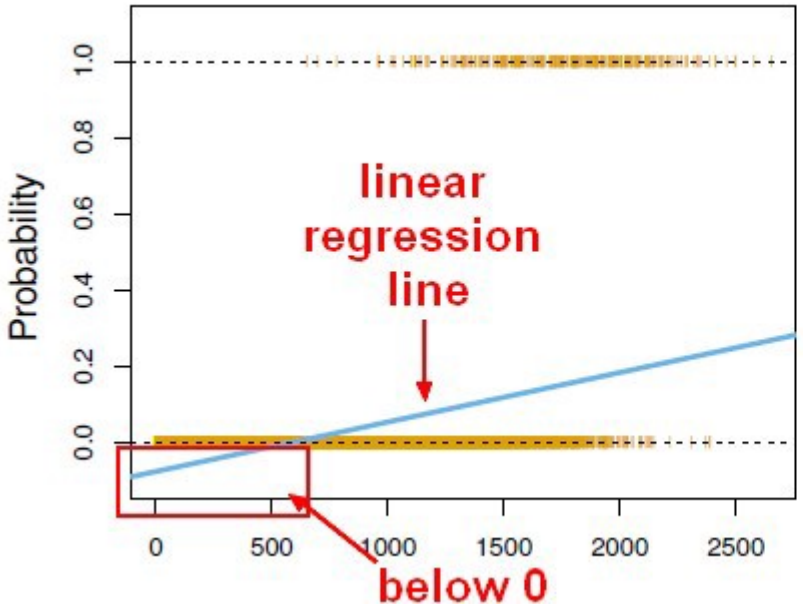
(Despite its name, it's used for classification, not regression)

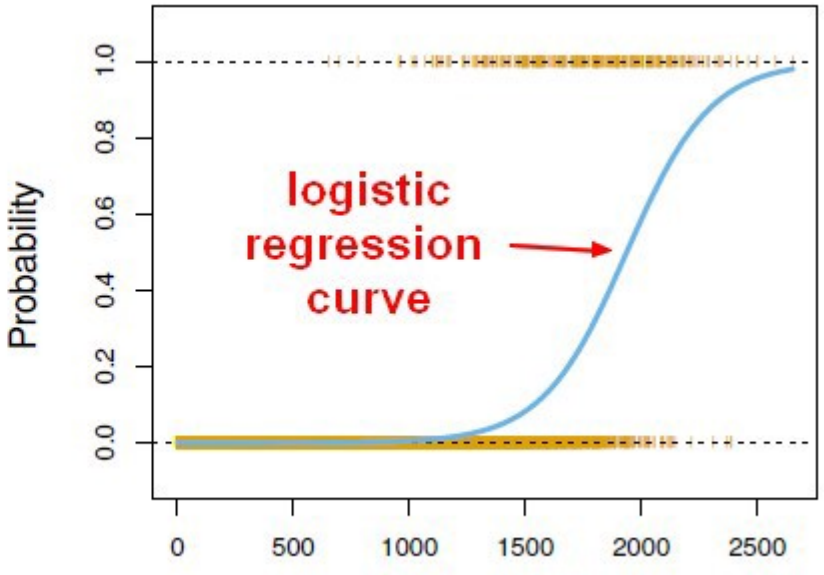
- At the core of logistic regression is the ***logistic function*** (also called ***sigmoid function***):

f(z) = 1 / (1 + e^(-z))

- This function maps any real-valued number to a value between 0 and 1, which can be interpreted as a probability

Logistic regression





Sigmoid function (logistic function) introduces non-linearity

Read about logistic regression: https://web.stanford.edu/~jurafsky/slp3/5.pdf

Logistic Regression (LR)

- For a binary classification problem:

$P(Y=1|X) = 1 / (1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)})$

Where:

Y is the dependent variable (outcome)

$X_1, X_2, ..., X_n$ are independent variables

$\beta_0, \beta_1, \beta_2, ..., \beta_n$ are the model parameters (coefficients)

- While training the model, the goal is to find the best values for the ***β coefficients*** (typically done using Maximum Likelihood Estimation (MLE)

LR - Regularization

- To prevent overfitting, we often use ***regularization***
- Regularization adds a penalty term to the cost function which discourages the model from relying too heavily on any single feature → simpler, more generalizable model

L1 (Lasso): Adds $|\beta|$ to the cost function, can lead to sparse models

L2 (Ridge): Adds β^2 to the cost function, shrinks coefficients

- In a spam detection model, L1 might zero out coefficients for words that aren't strong indicators of spam or non-spam (to identify the most important words) - Better for high-dimensional data with many irrelevant features
- In sentiment analysis, L2 might reduce the impact of all words slightly, preventing overreliance on any specific terms (preventing too much influence). Better when most features are relevant

Logistic Regression (LR)

- In LR, the decision boundary is linear
- For binary classification, we typically use a threshold of 0.5:

  If $P(Y=1|X) \geq 0.5$, classify as 1

  If $P(Y=1|X) < 0.5$, classify as 0
- For problems with more than two classes, we can use Multinomial Logistic Regression (uses the softmax function instead of sigmoid)

Why LR?

- The logistic regression model is easy to implement, interpretable
- Provides probability scores
- It is very efficient to train
- It is less prone to overfitting
- This classifier performs efficiently with linearly separable datasets

Let’s walk through a toy spam example

Email 1 (Spam): "Get rich quick"

Email 2 (Ham): "Meeting tomorrow“

Email	Class	get	rich	quick	meeting	tomorrow
1	Spam	1	1	1	0	0
2	Ham	0	0	0	1	1

Initialize weights:
w = [0.1, 0.1, 0.1, 0.1, 0.1]
(since vocab = 5)
w0 (bias) = 0.1

For Email 1:
z = 0.1 + 0.1**1* + *0.1**1 + 0.1**1* + *0.1**0 + 0.1*0 = 0.4
(z = w0 + w1*x1* + *w2*x2 + w3*x3* + *w4*x4 + w5*x5)
P(spam) = σ(0.4) ≈ 0.599

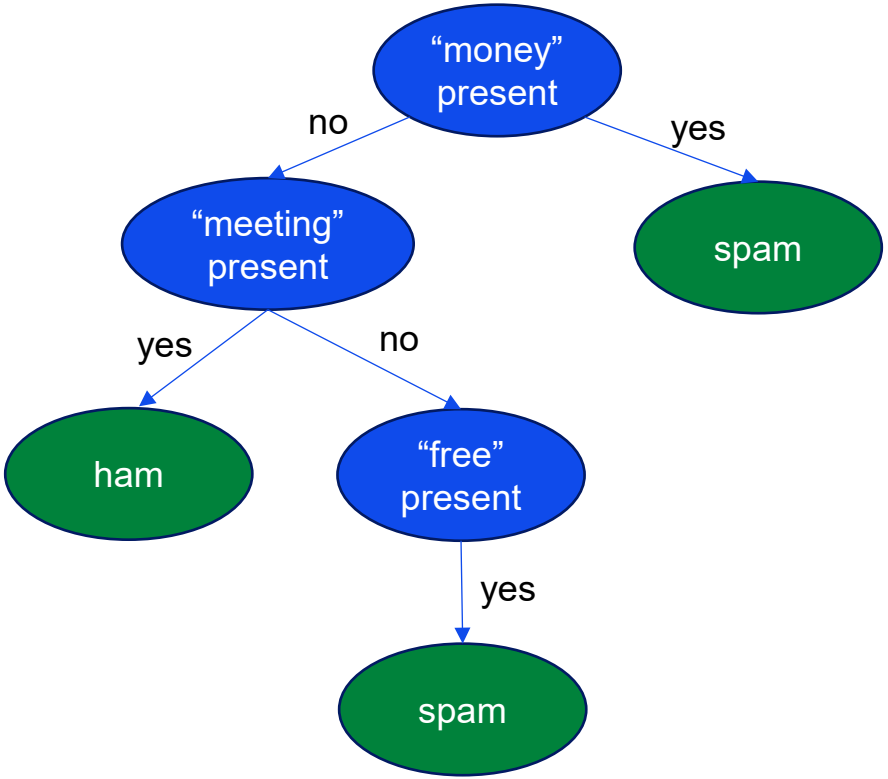
Cost calculation (assuming y=1 for spam):
J = -[1 * log(0.599) + (1-1) * log(1-0.599)] ≈ 0.512

Gradient descent: Update weights based on the calculated gradient. Repeat steps for Email 2 and continue iterations

Final prediction: **For a new email "Quick cash now":**
- Convert to feature vector: [1, 0, 1, 0, 0]
- Calculate z using final weights
- Apply sigmoid to get probability
- Classify based on probability (e.g., spam if P > 0.5)

Decision tree

- A Decision Tree is a tree-structured plan of a set of attributes (flowchart-like) to test to predict the output.
- Structure for Text Classification:
  - Internal Nodes: Features (words or phrases)
  - Branches: Outcomes of tests (e.g., word present/absent)
  - Leaf Nodes: Final classifications (e.g., spam/not spam)



Decision tree

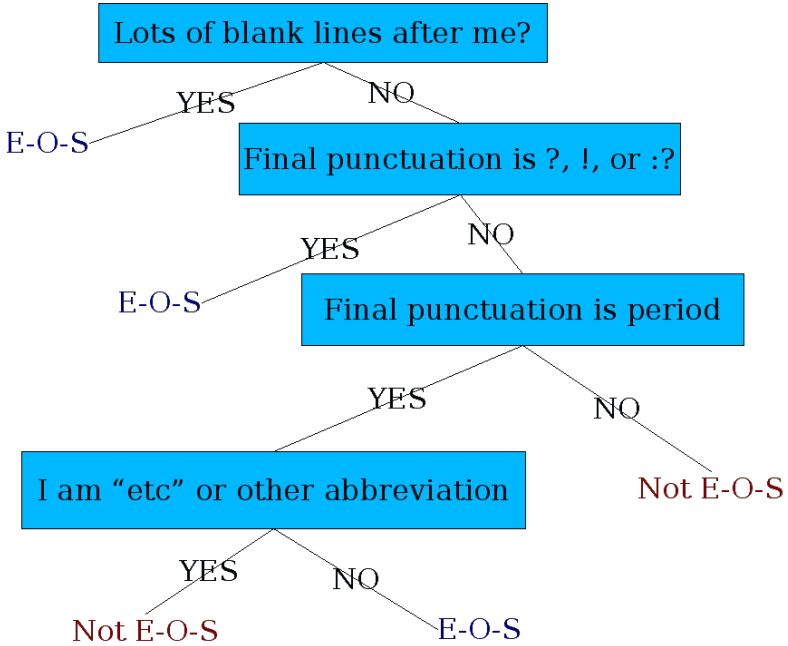
- Follows human decision-making using if-then rules

How is it built?

- For each attribute in the dataset, the Decision-Tree algorithm forms a node. The ***most important attribute is placed at the root node***
- For evaluating the task in hand, we start at the root node and we work our way down the tree by following the corresponding node that meets our condition or decision
- This process continues until a leaf node is reached. It contains the prediction or the outcome of the Decision Tree

https://towardsdatascience.com/scikit-learn-decision-trees-explained-803f3812290d

Another example DT – Determining the end of sentence



Why DT?

- Extremely interpretable model (clear decision paths), very popular!
- Suitable for binary and multiclass classification, and handles both numerical and categorical data
- Computationally cheap
- Can be used when the dimension of the feature vector is not too large (large numbers of features can lead to overfitting)
- DT provide a foundation for other important algorithms like bagged decision trees, random forest and boosted decision trees
- Hand-building only possible for very simple features/ domains - For numeric features, structure is usually learned by machine learning from a training corpus
- Small changes in data can result in very different trees (unstable)

Note: Decision trees also go by their modern name ***CART*** which stands for ***Classification and Regression Trees***

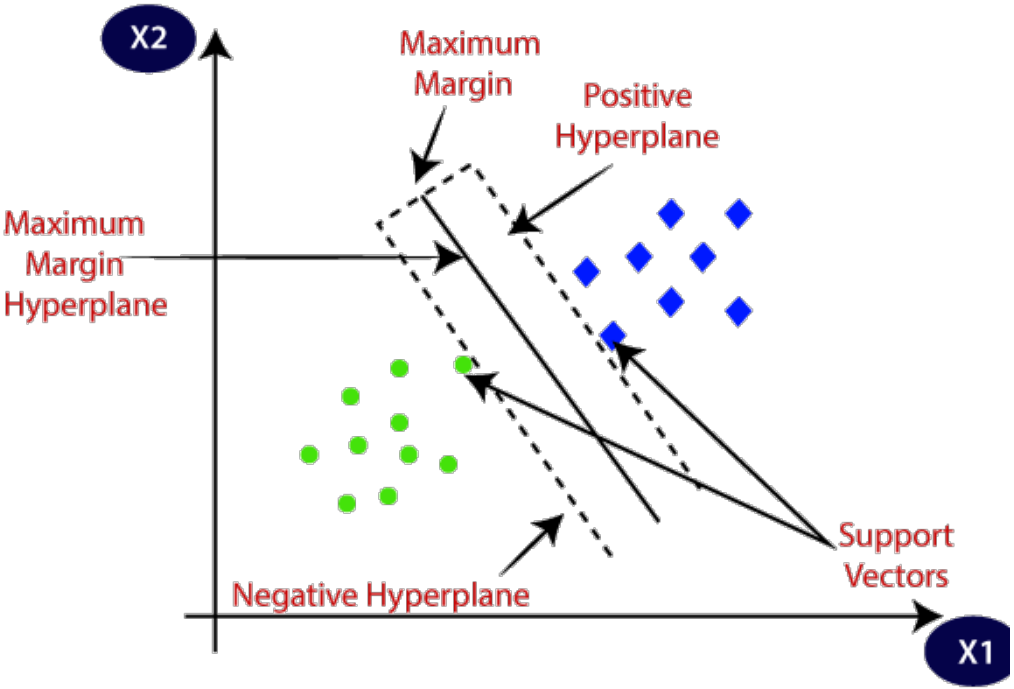
Support Vector Machines (SVM)

- SVM aims to find the best hyperplane that separates different classes in a high-dimensional space
- For text classification, this means finding the best decision boundary to separate documents of different categories
- This algorithm classifies vectors of tagged data into two different groups - one that contains most of the vectors that belong to a given tag, and another one with the vectors that do not belong to that tag
- Requires more coding power to train the model

https://scikit-learn.org/stable/modules/svm.html#svm-classification

SVM

Margin: Distance between the hyperplane and the observations closest to the hyperplane (support vectors)



Hyperplane: Decision boundary for more than 2 dimensions

Support vectors: Points closest to the hyperplane

Documents are represented as vectors in a high-dimensional space

Why SVM?

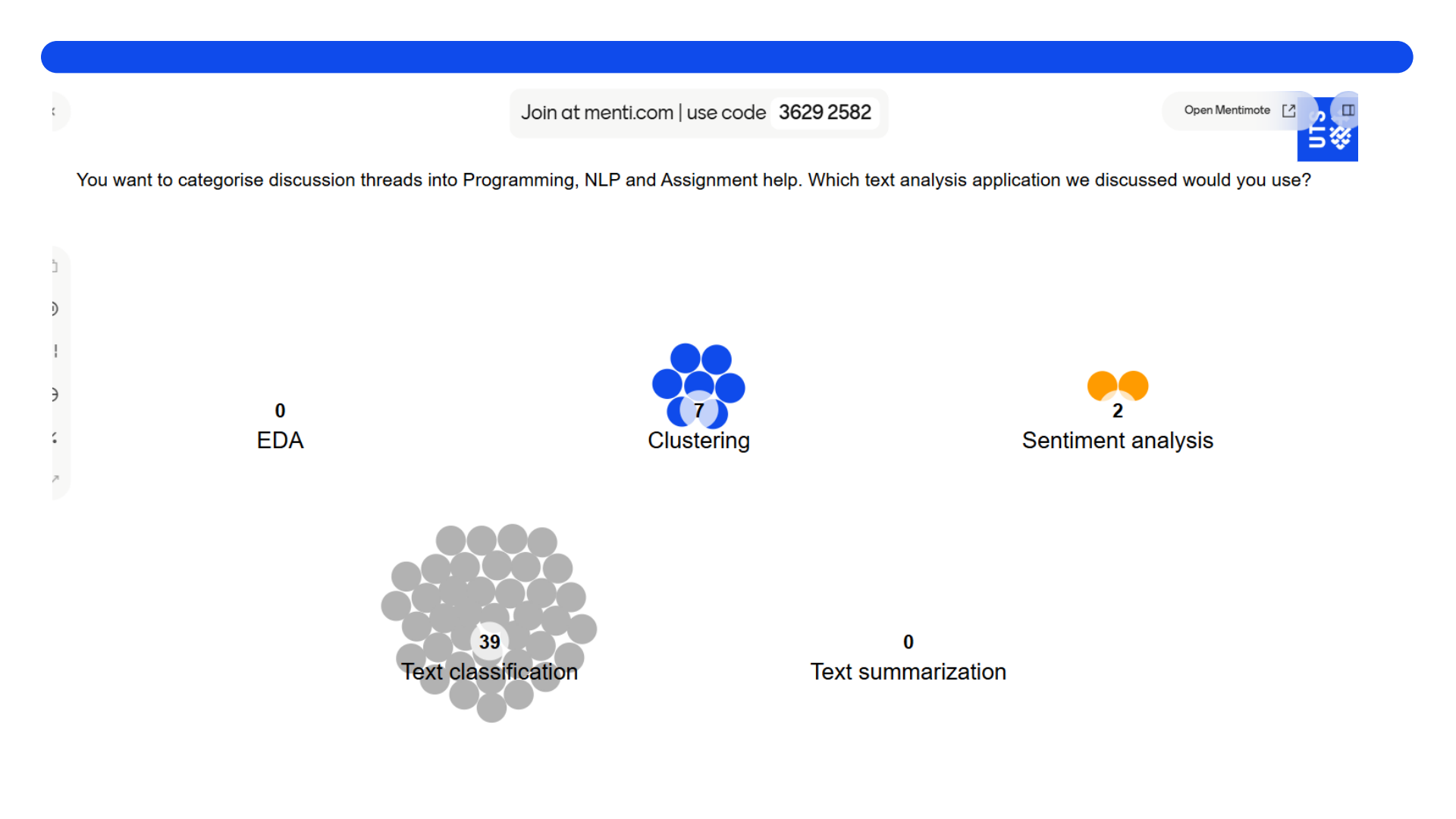
Can be used for classification, regression and outliers detection.

- Effective in high dimensional spaces
- Still effective in cases where number of dimensions is greater than the number of samples
- The results of this algorithm are usually better than the results you get with Naive Bayes, and works great for textual data!
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels

Join at menti.com | use code 3629 2582

Open Mentimote

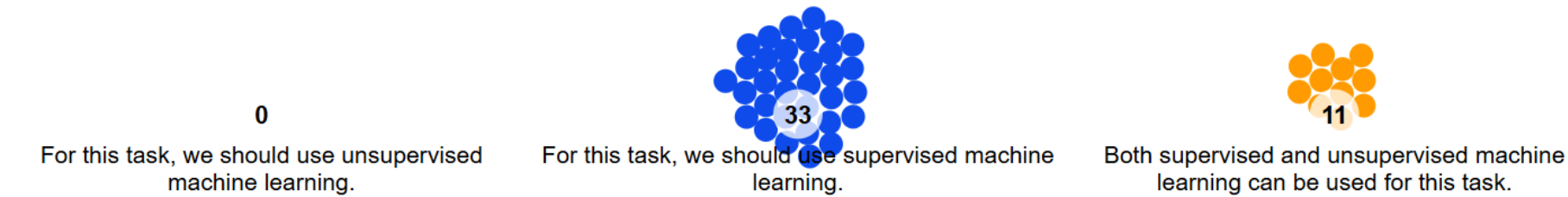
You want to categorise discussion threads into Programming, NLP and Assignment help. Which text analysis application we discussed would you use?



Join at menti.com | use code 3629 2582

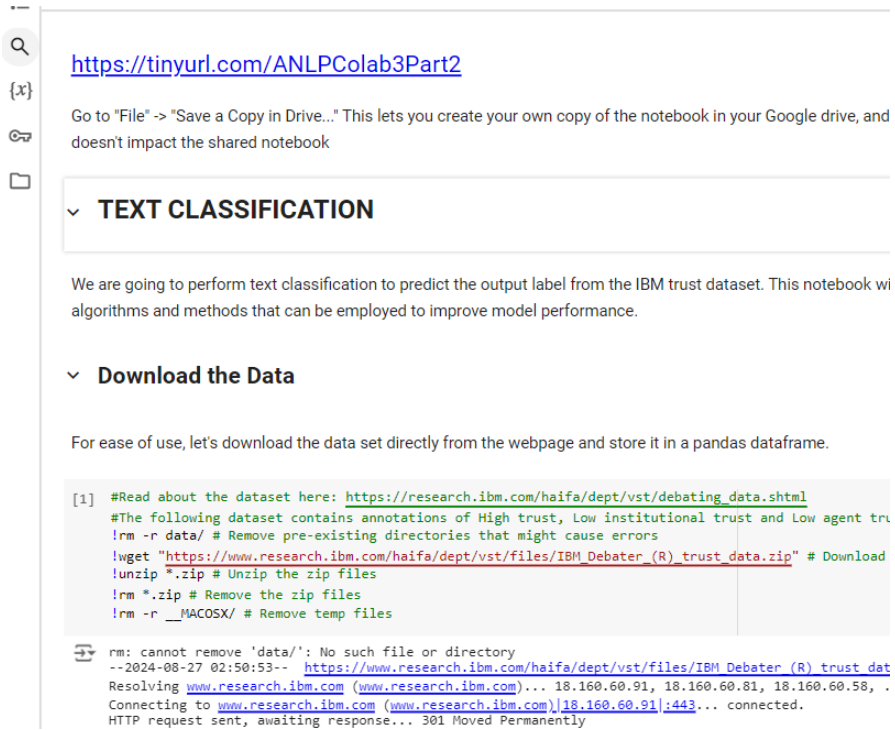
Open Mentimote

Imagine you have been tasked with building a machine learning model to classify movie reviews as positive or negative. Which statements are correct?



Tutorial

tinyurl.com/ANLPColab3Part2



https://tinyurl.com/ANLPColab3Part2

Go to "File" -> "Save a Copy in Drive..." This lets you create your own copy of the notebook in your Google drive, and doesn't impact the shared notebook

TEXT CLASSIFICATION

We are going to perform text classification to predict the output label from the IBM trust dataset. This notebook wi algorithms and methods that can be employed to improve model performance.

Download the Data

For ease of use, let's download the data set directly from the webpage and store it in a pandas dataframe.

```
[1] #Read about the dataset here: https://research.ibm.com/haifa/dept/vst/debating_data.shtml
    #The following dataset contains annotations of High trust, Low institutional trust and Low agent tr
    !rm -r data/ # Remove pre-existing directories that might cause errors
    !wget "https://www.research.ibm.com/haifa/dept/vst/files/IBM_Debater_(R)_trust_data.zip" # Download
    !unzip *.zip # Unzip the zip files
    !rm *.zip # Remove the zip files
    !rm -r __MACOSX/ # Remove temp files
```

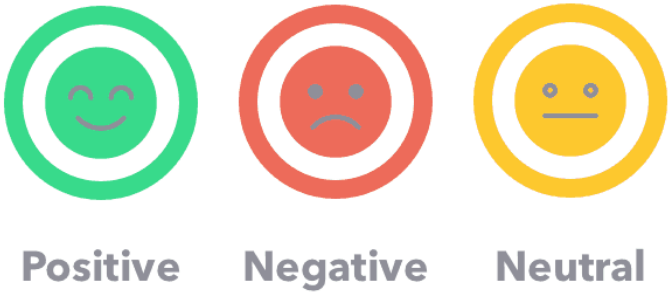
```
rm: cannot remove 'data/': No such file or directory
--2024-08-27 02:50:53--  https://www.research.ibm.com/haifa/dept/vst/files/IBM_Debater_(R)_trust_dat
Resolving www.research.ibm.com (www.research.ibm.com)... 18.160.60.91, 18.160.60.81, 18.160.60.58, .
Connecting to www.research.ibm.com (www.research.ibm.com)|18.160.60.91|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
```

Attendance check, Break

Sentiment analysis

What is sentiment analysis?

- Sentiment analysis (aka Opinion Mining) is the computational method of automatically identifying emotions in a text
- Generally framed as a text classification problem to detect Positive, Negative, or Neutral emotions, but can also be more fine-grained



Example

Detecting the sentiment in short movie reviews:

Avengers is a great movie

Positive

Avengers is not a great movie

Negative

Avengers is a movie

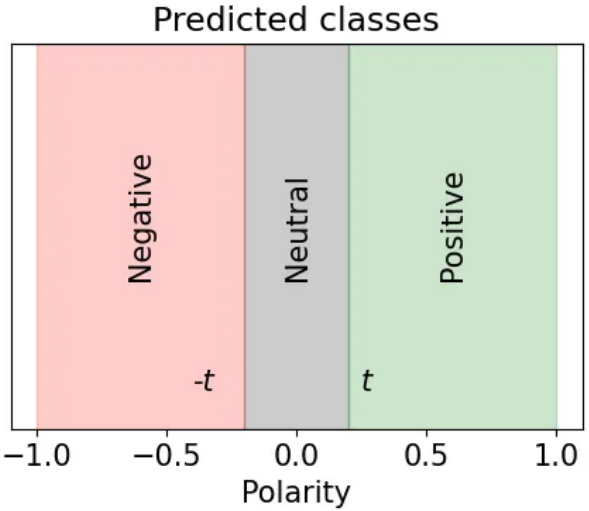
Neutral

Applications of sentiment analysis

- Determining user sentiments on a product/ service by analysing reviews (E.g. hotels, movies, subjects!)
- Evaluating survey responses for tone
- Understanding the discussion around a topic of interest in social media
- Summarizing the intent of large quantities of text

Polarity

- Polarity refers to the orientation of sentiment expressed in a piece of text
- Polarity is typically measured within the range [-1, 1] where -1 corresponds to a strongly negative sentiment, 0 to a neutral sentiment, and +1 to a strongly positive sentiment
- Having a polarity value is very useful because it allows defining our own polarity threshold *t* that separates a neutral class from negative/positive classes
- Some models don't output a polarity value but instead provide probabilities: p-- (strongly negative), p- (negative), p0 (neutral), p+ (positive), p++ (strongly positive). The predicted class will be the one with the maximum probability



Subjectivity

- Subjectivity refers to the degree to which a piece of text expresses personal opinions, emotions, or judgments (subjective), as opposed to stating objective facts (objective)

Subjective: "The movie was incredibly boring and a waste of time."

Objective: "The movie has a runtime of 120 minutes and was released in 2023."

- Subjectivity can range from a spectrum of highly objective to highly subjective
- Note: Expressions of subjectivity can vary across languages and cultures. Sarcasm and irony can also complicate subjectivity detection!

Subjectivity

Can add a layer of sophistication to sentiment analysis by helping to focus on the parts of text that express opinions or emotions!

Example of Subjectivity Analysis:

Text: "The new smartphone has a 6-inch screen and costs $800. I think it's overpriced for its features."

Analysis:

"The new smartphone has a 6-inch screen and costs $800." (Objective)

"I think it's overpriced for its features." (Subjective)

Try it out!

- TweetEval (sentiment subset, test split): 3972 negative tweets and 2375 positive tweets from SemEval 2017 [Rosenthal et al., 2017]

- Yelp product reviews (test split): 10 000 strongly negative reviews (1 star) and 10 000 strongly positive reviews (5 stars) [Zhang et al., 2016]

Detecting complex emotions

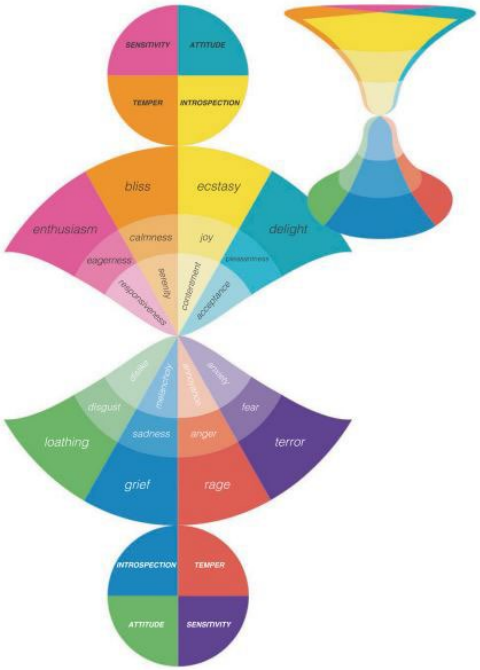


Figure 1. Hourglass model revisited.

Table 1. Examples of compound emotions.

JOY	PLEASANTNESS	love	enjoyment	amusement
	EAGERNESS	euphoria	excitement	thrill
	CALMNESS	enlightenment	relaxation	sweet idleness
SADNESS	DISGUST	hate	guilt	remorse
	FEAR	distress	troubledness	misery
	ANGER	envy	bitterness	resentment
CALMNESS	PLEASANTNESS	assertiveness	compassion	empathy
	EAGERNESS	focus	determination	perseverance
	FEAR	carelessness	laxity	looseness
ANGER	DISGUST	hatred	ruthlessness	viciousness
	FEAR	nastiness	coercion	possessiveness
	EAGERNESS	stubborness	obstinacy	mulishness
PLEASANTNESS	DISGUST	shamelessness	cheekiness	brazenness
	EAGERNESS	kindness	audacity	hospitality
	FEAR	awe	submission	reverence
DISGUST	JOY	morbidness	schadenfreude	gloat
	FEAR	impiety	cowardness	inhospitality
	EAGERNESS	recklessness	temerity	rashness
EXPECTATION	JOY	hope	anticipation	optimism
	SADNESS	hopelessness	despair	pessimism
	EAGERNESS	vigilance	alertness	caution
SURPRISE	ANGER	shock	outrage	thunderstruckness
	FEAR	alarm	dismay	dumbstruckness
	PLEASANTNESS	amazement	astonishment	wonderstruckness

Team of sentiment analysis researchers:
https://sentic.net/

Approaches

1. Lexicon-Based Approach:

Uses pre-defined dictionaries of words with associated sentiment scores

Pros: Simple, interpretable, doesn't require labeled data

Cons: Limited by dictionary coverage, struggles with context and domain-specific language

Example: VADER (Valence Aware Dictionary and Sentiment Reasoner)

2. Machine Learning Approach:

Uses labeled data to train classifiers

Common algorithms: Naive Bayes, Support Vector Machines, Logistic Regression

Pros: Can capture complex patterns, adaptable to specific domains

Cons: Requires large amounts of labeled data, may not generalize well across domains

3. Deep Learning Approach:

Uses neural networks to learn sentiment patterns

Common architectures: CNNs, RNNs (LSTM, GRU), Transformers

Pros: Can capture complex linguistic patterns and context

Cons: Requires large datasets, computationally intensive, less interpretable

Example: BERT (Bidirectional Encoder Representations from Transformers)

Task: Identify words contributing to the emotion

"*I love this movie! It's sweet, but with satirical humor. The dialogs are great and the adventure scenes are fun. It manages to be romantic and whimsical while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I have seen it several times and I'm always happy to see it again.....*"

Key words contributing to the emotion

"*I **love** this movie! It's **sweet,** but with **satirical** humor. The dialogs are **great** and the adventure scenes are **fun**. It manages to be **romantic** and **whimsical** while laughing at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I have seen it several times and I'm always **happy** to see it **again**....."*

Dictionary models / Lexicon-based approaches

Use pre-defined dictionaries (lexicons) of words and phrases with associated sentiment scores or polarities.

Here's a toy example:

"The food was great and the service was nice"

Words matched: "great" (2), "nice" (1)

Calculation: 2 + 1 = 3

Sentiment: Positive (3 > 0)

"The product is okay, not very good but not bad either."

Words matched: "okay" (0), "not" (-1), "very" (2), "good" (1), "not" (-1), "bad" (-1)

Calculation: 0 + (-1 * 2 * 1) + (-1 * -1) = -1

Sentiment: Slightly Negative (-1 < 0)

Sample Lexicon

Word	Sentiment Score
great	2
good	1
nice	1
terrible	-2
bad	-1
awful	-2
love	3
hate	-3
okay	0
not	-1 (negation)
very	2 (intensifier)

Commonly used Python packages (lexicon-based)

	Accuracy/Ratio (polarity threshold 0)	Accuracy/Ratio (maximum probability)	Languages	Speed (10 000 texts*)
TextBlob Pattern	0.75/0.98 (yelp) 0.69/0.64 (tweet) 0.77/0.54 (finance)	Not available	English Dutch French Italian	10 seconds
TextBlob Naive Bayes	0.67/1.0 (yelp) 0.48/1.0 (tweet) 0.66/1.0 (finance)	Same as polarity threshold 0	English	20 seconds
NLTK VADER	0.78/0.99 (yelp) 0.76/0.79 (tweet) 0.78/0.77 (finance)	Not available	English	5 seconds
Sentimentr (R)	0.79/0.99 (yelp) 0.74/0.90 (tweet) 0.83/0.79 (finance)	Not available	English	45 seconds

Source: https://medium.com/@pavlo.fesenko/best-open-source-models-for-sentiment-analysis-part-1-dictionary-models-ece79e617653

Progress of methods

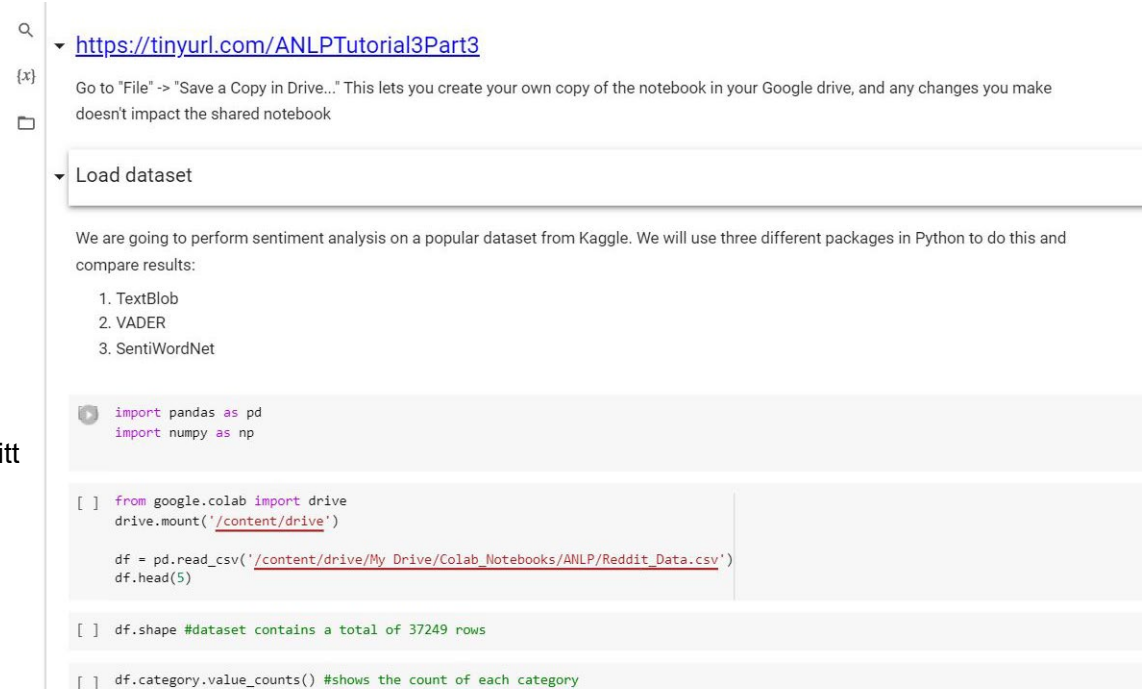
- https://nlpprogress.com/english/sentiment_analysis.html
- https://github.com/sebastianruder/NLP-progress/blob/master/english/sentiment_analysis.md

Sentiment analysis in Python

Open Colab notebook:

tinyurl.com/ANLPColab3Part3

Dataset:
tinyurl.com/Redditdataset
https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset?select=Reddit_Data.csv



Real world applications - Considerations

Which ML algorithm should I use?

It depends…

- On the size of data, its type
- Imbalance in the dataset
- Nature of the problem

There is no single best performing algorithm!

Improve the performance of models

1. Using other tokenization methods, pre-processing for weighting (e.g. collapsing terms)
2. Hyperparameter tuning
3. Using word embedding/ dense vectors like word2vec or doc2vec (We'll cover this in the next class)
4. Normalization
5. Elimination of features with extremely low frequency

Reference links :

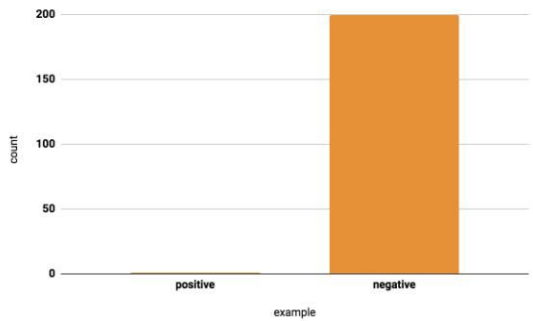
- https://www.analyticsvidhya.com/blog/2015/10/6-practices-enhance-performance-text-classification-model/
- https://datascience.stackexchange.com/questions/19276/improving-accuracy-of-text-classification
- https://towardsdatascience.com/how-i-achieved-90-accuracy-on-a-text-classification-problem-with-zero-preprocessing-6acfa96e8d2e
- https://neptune.ai/blog/text-classification-tips-and-tricks-kaggle-competitions
- https://www.researchgate.net/post/How-to-Improve-Accuracy-in-Text-Classification
- https://www.visual-design.net/post/data-transformation-and-feature-engineering-in-python

Imbalanced datasets

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Why look out for imbalanced data? You may need to apply a particular sampling technique if you have a classification task with an imbalanced data set.

Consider the following example of a model that detects fraud. Instances of fraud happen once per 200 transactions in this data set, so in the true distribution, about 0.5% of the data is positive.



Learn tips to combat this: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Harms in text classification (Ethics!)

Representational harms :

Kiritchenko and Mohammad (2018) examined the performance of 200 sentiment analysis systems on pairs of sentences that were identical except for containing either a common African American first name (like Shaniqua) or a common European American first name (like Stephanie), chosen from the Caliskan et al. (2017) study discussed in Chapter 6. They found that ***most systems assigned lower sentiment and more negative emotion to sentences with African American names***, reflecting and perpetuating stereotypes that associate African Americans with negative emotions (Popp et al., 2003).

Censorship:

Some widely used ***toxicity classifiers incorrectly flag as being toxic sentences that are non-toxic but simply mention minority identities like women*** (Park et al., 2018), ***blind people*** (Hutchinson et al., 2020***) or gay people*** (Dixon et al., 2018), or simply use linguistic features characteristic of varieties like African- American Vernacular English (Sap et al. 2019, Davidson et al. 2019). Such ***false positive errors***, if employed by toxicity detection systems without human oversight, could lead to the censoring of discourse by or about these groups.

The IQ dilemma

We can attempt to train a classifier to predict IQ from photos & texts. Let's discuss whether it is ethical to build such a technology and what are the risks.

- Who could benefit from such a classifier?

Source: EACL 2023 Ethic Tutorial: Understanding Ethics in NLP Authoring and Reviewing

The IQ dilemma: the ethics of the research question

We can attempt to train a classifier to predict IQ from photos & texts. Let's discuss whether it is ethical to build such a technology and what are the risks.

- Who could benefit from such a classifier?
- Let's assume for now that the classifier is 100% accurate.
  Who can be harmed from such a classifier? How can such a classifier be misused?

The IQ dilemma: understanding the risks

We can attempt to train a classifier to predict IQ from photos & texts. Let's discuss whether it is ethical to build such a technology and what are the risks.

- Who could benefit from such a classifier?
- Who can be harmed from such a classifier? How can it be misused?
- What are the pitfalls/risks in the current solution?
  - Example: Our test results show 90% accuracy
    - We found out that white females have 95% accuracy
    - People with blond hair under age of 25 have only 60% accuracy

The IQ dilemma: understanding the responsibility

We can attempt to train a classifier to predict IQ from photos & texts. Let's discuss whether it is ethical to build such a technology and what are the risks.

- Who could benefit from such a classifier?
- Who can be harmed from such a classifier? How can it be misused?
- What are the pitfalls/risks in the current solution?
- Who is responsible?
  - Researcher/developer? Advisor/manager? Reviewer? The IRB? The University? Society as a whole?

We need to be aware of real-world impact of our work and understand the relationship between ideas and consequences!

Further reading - Canvas pages

Machine learning algorithms - Under the hood

- Machine Learning Basics (YouTube video)
- Andrew Ng's Machine Learning Course (Online Course) - Includes mathematical foundations of ML algorithms and Python Programming
- Practical Machine learning with Python (YouTube video playlist, Materials)
- Machine Learning models - Intuition:
- ◦ Bayes theorem (Foundation for the Naive Bayes classifier):
  - https://www.mathsisfun.com/data/bayes-theorem.html
  - https://machinelearningmastery.com/bayes-theorem-for-machine-learning/
  - ◦ Logistic regression
  - ◦ Decision trees in Python (Using Scikit learn)
  - ◦ Support Vector Machines (SVM)
- Python Data Science Handbook (Colab notebooks with Python code for basic ML tasks)
- Cross Validation (explained with figures)

Additional reading Week 4

These topics are extensions of the content we covered in class, and are recommended for further reading:

- Book (clear introduction to text classification algorithms): Bengfort, B., Bilbro, R., & Ojeda, T. (2018). *Applied text analysis with Python: Enabling language-aware data products with machine learning*. " O'Reilly Media, Inc.".
- List of people to follow: https://medium.com/machine-learning-in-practice/my-curated-list-of-ai-and-machine-learning-resources-from-around-the-web-9a97823b8524
- Machine learning course notes (AWS and Udacity)
- An interesting website built for kids, but the stories are interesting to try on your own (some of them provide Python worksheets!): https://machinelearningforkids.co.uk/#!/stories/intro

Q & A