University of Northampton

Week 4

Classical ML for Text Classification

CSY3055 – Natural Language Processing

Dr Oluseyi Oyedeji

Learning Goals for Week 4

By the end of week two, you should be able to:

- Explain the difference between **generative and discriminative models**.
- Train and compare **Naïve Bayes**, **Logistic Regression** for text data.
- Use **TF-IDF** features and **evaluate results** with **precision, recall, and F1**.
- Reflect on **bias and fairness** in text classification.

Relevance

- **Classical ML** still powers *real-world NLP* applications
- Many spam filters and **sentiment tools** uses the algorithm
- It builds the **foundation for understanding** modern deep learning.

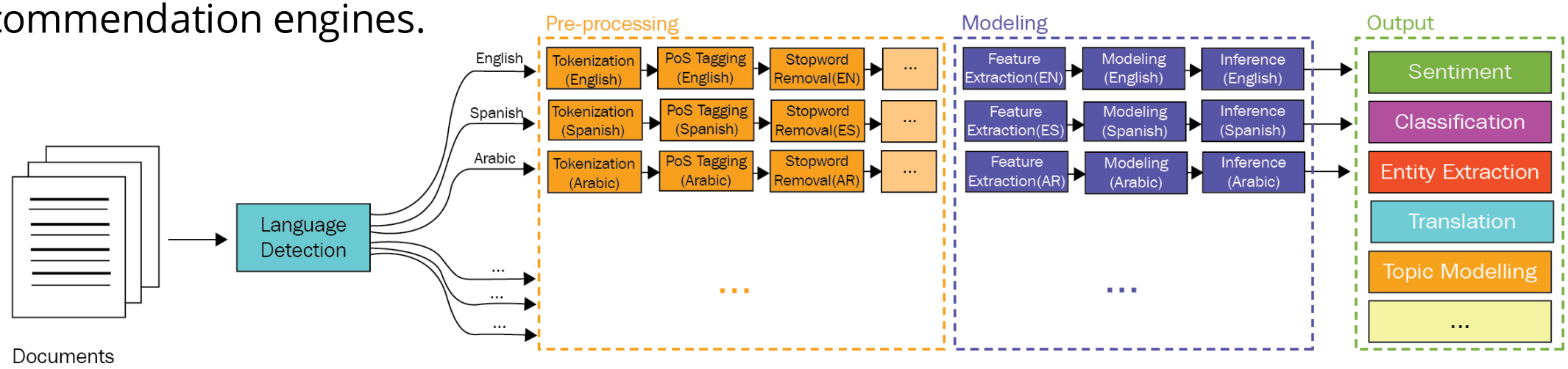
Today's Session

- Generative vs Discriminative models
- Naïve Bayes
- Logistic Regression
- Evaluation metrics
- Practical: IMDB sentiment classification
- Ethical reflection on bias

Introduction

Why Classical ML Still Matters

- Before deep learning became dominant, classical ML algorithms **powered** nearly all **text classification systems**
  - Spam filters
  - Sentiment analysers
  - Topic classifiers, and
  - Recommendation engines.

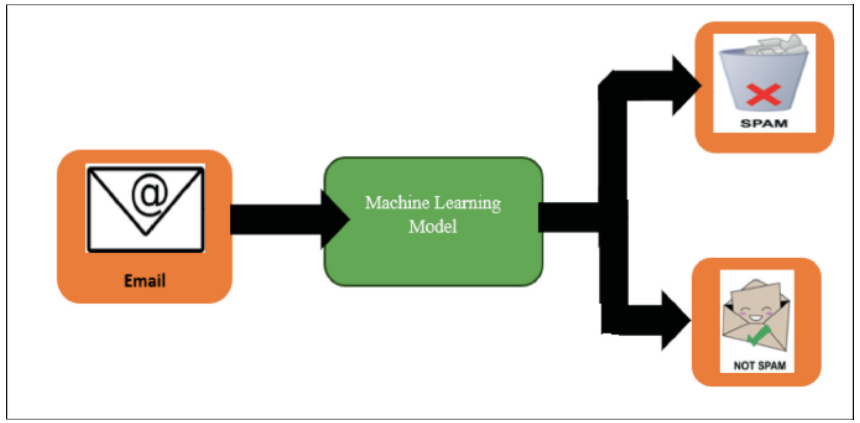


Why Classical ML Still Matters contd.

- Even today, models like Logistic Regression, Naïve Bayes, and SVM are still considered:
  - **Lightweight**: suitable for fast and real-time inference.
  - **Interpretable**: we can inspect learned weights and understand why a decision was made.
  - **Strong baselines**: good starting point before moving to neural architectures.

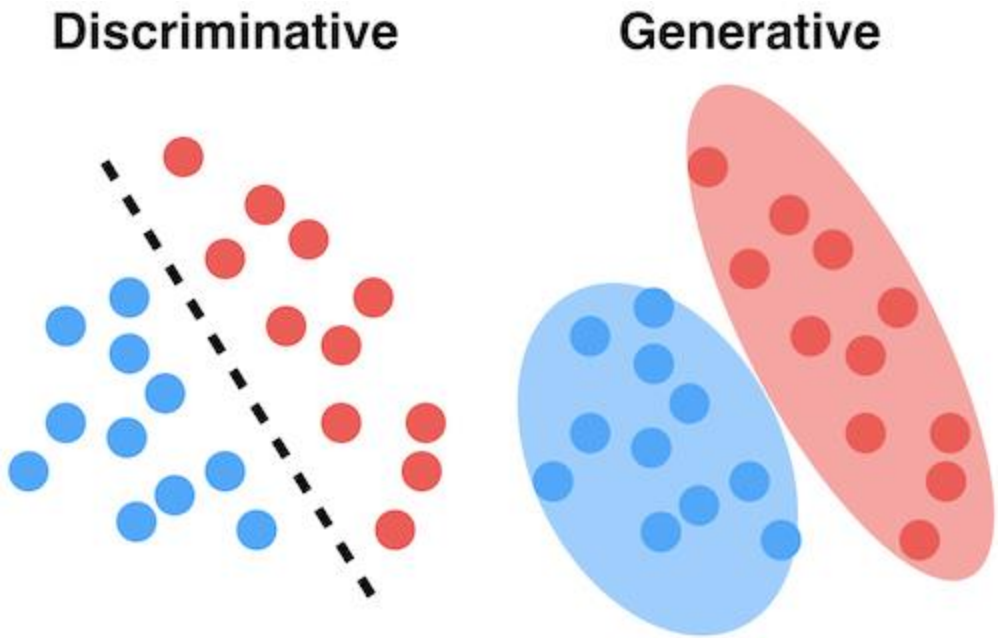
Example applications

- Email spam detection
- Sentiment classification on reviews
- Detecting toxic or fake comments



Classical ML Paradigms

- Classical machine learning models can be broadly divided into **generative** and **discriminative** approaches based on what they learn from data.



Generative Models

- Learns the **joint probability** $P(X, Y) = P(X|Y)P(Y)$ i.e. how the data X is generated for each class Y.
- It can *generate new samples* and classifies using Bayes' rule:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- **Example:** Naive Bayes, Gaussian Mixture Models.

Discriminative Models

- Learns the **conditional probability** `P(Y|X)` or a direct decision boundary between classes.
- It focuses on *distinguishing* classes rather than modelling how data is produced.

  Example: Logistic Regression, Support Vector Machines.

Generative vs Discriminative Models

- A generative model = understand what a "**positive**" review typically looks like and what a "**negative**" one looks like.
- A discriminative model just looks for the *boundary between positive and negative reviews*.

Naïve Bayes

Naïve Bayes Classifier (Generative Model)

- Naive Bayes is a **probabilistic classifier** based on Bayes' Theorem assuming feature independence.
- Widely applied in NLP tasks
  - text classification, spam detection, and sentiment analysis.
- Efficiently manages **large vectors of word frequencies** or binary indicators in documents.
- Provides scalable, interpretable solutions with clear probabilistic predictions.

Naïve Bayes Classifier contd.

- It uses **Bayes' Theorem** to compute the probability of a class given words in a document

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

  - **P(y) :** prior probability of class (e.g., 0.5 for positive, 0.5 for negative)
  - **P(x|y):** likelihood of the features given class
  - **"Naïve":** assumption: words are independent given class

Variants of Naïve Bayes

Type	Use case
Multinomial NB	Word counts or term frequencies
Bernoulli NB	Binary word presence/absence

Example

- If “**fantastic**” and “**excellent**” appear often in **positive reviews**:

$$P(\text{positive} \mid \text{‘fantastic excellent’}) \propto P(\text{‘fantastic’} \mid \text{positive}) \times P(\text{‘excellent’} \mid \text{positive})$$

- The class with the **higher posterior** wins.

The Core Idea Behind Naïve Bayes

- Bayes' Theorem

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

- For Prediction

$$\hat{y} = \arg\max_y P(y) \prod_i P(x_i|y)$$

- "Naïve" assumption: *features are independent given the class*.
- E.g. Predict if an email is spam using word probabilities.

The Core Idea Behind Naïve Bayes contd.

```
# Simplified example
P_y = {'Spam': 0.4, 'Ham': 0.6}
P_x_given_y = {'Spam': [0.8, 0.7], 'Ham': [0.2, 0.3]}

for y in P_y:
    score = P_y[y]
    for p in P_x_given_y[y]:
        score *= p
    print(y, score)
```

$$P(y) \times \prod_i P(x_i|y)$$

The Core Idea Behind Naïve Bayes contd.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

X = CountVectorizer().fit_transform(docs)
model = MultinomialNB().fit(X, labels)
model.predict(X)
```

- The library computes (**P(y)**) and (**P(x_i|y)**) internally.
- *Focus on data preparation, not the probability math*.

The Core Idea Behind Naïve Bayes contd.

Feature Representation	Theoretical Fit	Common Use	Notes
CountVectorizer	Exact for Multinomial NB	Text classification	Models actual word counts
TF-IDF Vectorizer	Approximation	Works empirically	Uses weighted frequencies, not true probabilities

"Naïve Bayes was built for counts. TF-IDF still works, but it bends the probabilistic assumption a bit."

Example

- If “**fantastic**” and “**excellent**” appear often in **positive reviews**:

$$P(\text{positive} \mid \text{‘fantastic excellent’}) \propto P(\text{‘fantastic’} \mid \text{positive}) \times P(\text{‘excellent’} \mid \text{positive})$$

- The class with the **higher posterior** wins.

Weaknesses of Naïve Bayes

- **Independence assumption** unrealistic
- Sensitive to **rare words**
- **Can underperform** on complex boundaries

Logistic Regression

Logistic Regression (Discriminative Model)

- Logistic regression is widely used in NLP to classify data into categories like spam detection or sentiment analysis.
  - The sigmoid function transforms linear combinations of features into probabilities between 0 and 1 for classification decisions.
  - Multinomial Logistic Regression uses softmax function for multiple classes
  - Simple and interpretable, making it ideal for practical NLP applications.
- Key aspects include the **cost function** and **optimization techniques** essential for training logistic regression models.

Logistic Regression contd.

- **Logistic Regression** learns *weights* for each feature (word) that best predict the target class:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

(x): vectorised features (e.g., TF-IDF values)

(w): learned weights

(b): bias term

Logistic Regression Core Formula

- The **sigmoid function** turns any number into a value between 0 and 1.
- (P>0.5) → class 1 (P≤0.5) → class 0

$$z = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}}$$

How Logistic Regression learns

- **Logistics regression** simply learns by:
  - Adjusting weights (w_i) to make predicted probabilities match true labels.
  - Using cross-entropy (log-loss) to measure error.
  - Learning which features push a text toward or away from a class.

Word	Weight	Meaning
“free”	+2.1	more spammy
“meeting”	−1.3	more normal

Multiclass Extension

- Binary uses sigmoid → one probability (0–1).
- Multi-class uses softmax → many probabilities that add to 1.

$$P(y = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Logistic Regression in an NLP Pipeline

```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

model = make_pipeline(TfidfVectorizer(), LogisticRegression())
model.fit(train_texts, train_labels)
```

- Vectorizer → numbers → Logistic Regression → class label

Comparison with Naïve Bayes

Feature	Naïve Bayes	Logistic Regression
Type	Generative	Discriminative
Assumptions	Features independent	None
Speed	Very fast	Slower but accurate
Interpretability	High	High
Use case	Spam detection, baselines	Sentiment analysis, real-time predictions

Model Evaluation

Confusion Matrix – Binary Classification

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Confusion Matrix – Multi-class Classification

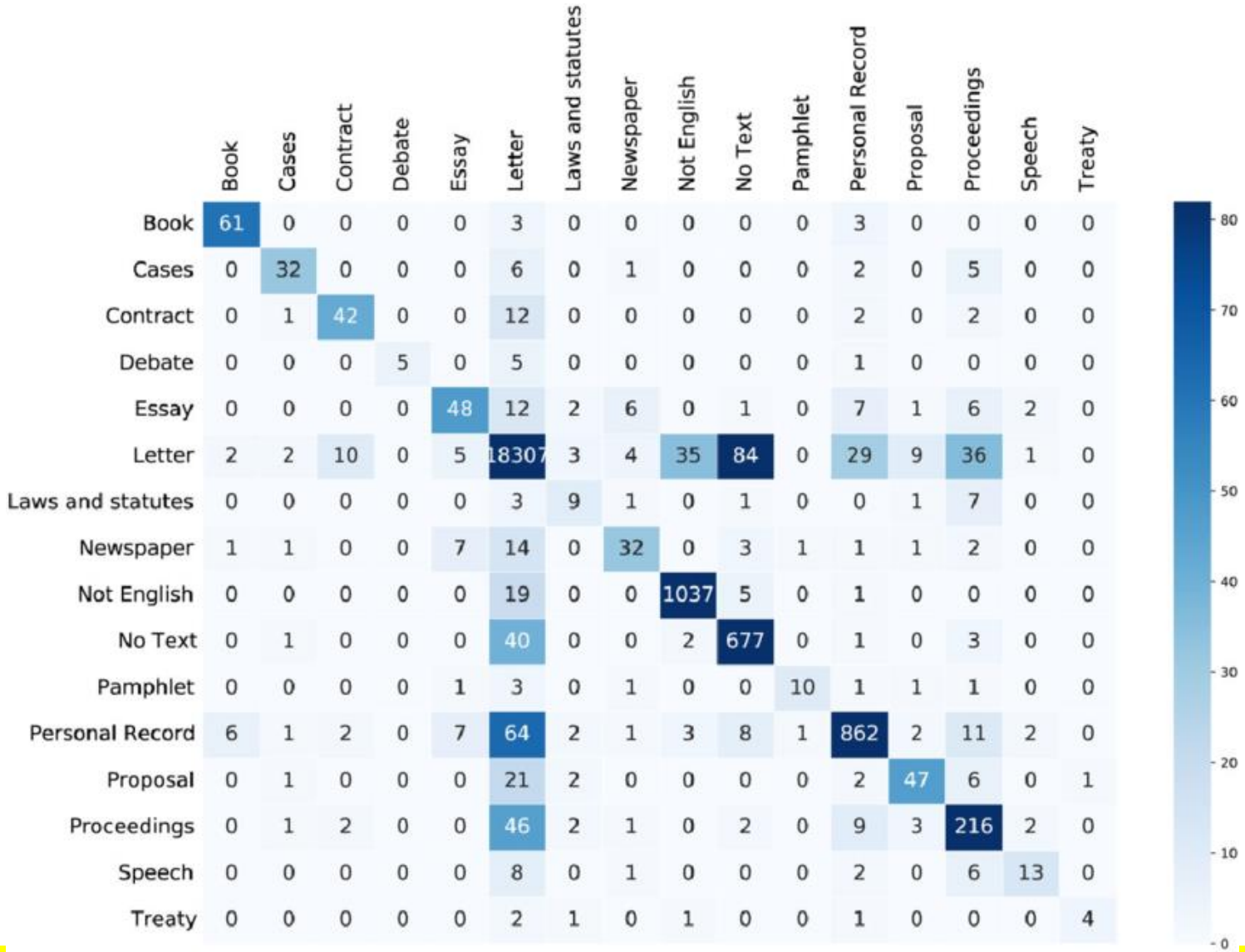


Figure: Multiclass Confusion Matrix Showing Model Performance Across Document Categories

Metrics

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Interpretation:
  - Precision → Of the predicted positives, how many were correct?
  - Recall → Of all true positives, how many were detected?
  - F1-score → Balance between the two.

Ethical Dimension: Bias in Text Classification

- Spam filters and sentiment models can **unfairly** treat minority language patterns:
- Non-standard English or dialect messages misclassified as spam/toxic.
- Bias introduced through under-representation in training data.
    - How could you detect this bias?
    - How can data collection improve fairness?

Practical & Wrap-Up

Thank you