

# Text Classification with Sequential Models

Zeham Management Technologies BootCamp by SDAIA

September 5<sup>th</sup>, 2024



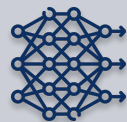
**SDAIA**

الهيئة السعودية للبيانات  
والذكاء الاصطناعي  
Saudi Data & AI Authority

# Agenda



**Introduction to text classification**



**Text classification with sequential model**



**Evaluation of text classification models**

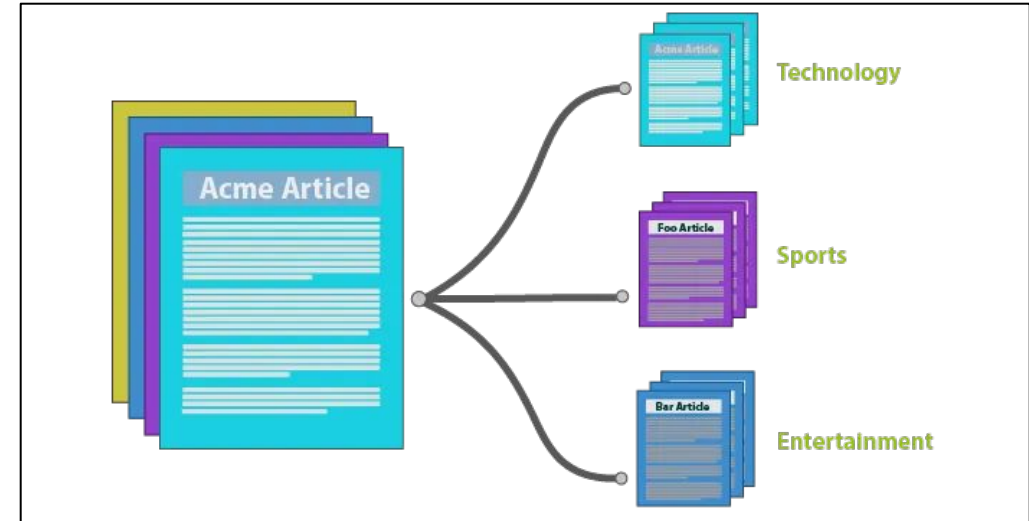


# Introduction to text classification



# What is Text Classification?

Text Classification is the process of sorting and categorizing text into predefined classes. It helps organize, structure, and filter text data according to various parameters.



Source





# Why Text Classification Matters

- Automates tasks that would be too time-consuming for humans, such as sorting large volumes of text data.
- Enhances data analysis by allowing organizations to derive meaningful insights from unstructured text.
- Improves decision-making by providing quick, data-driven categorizations.

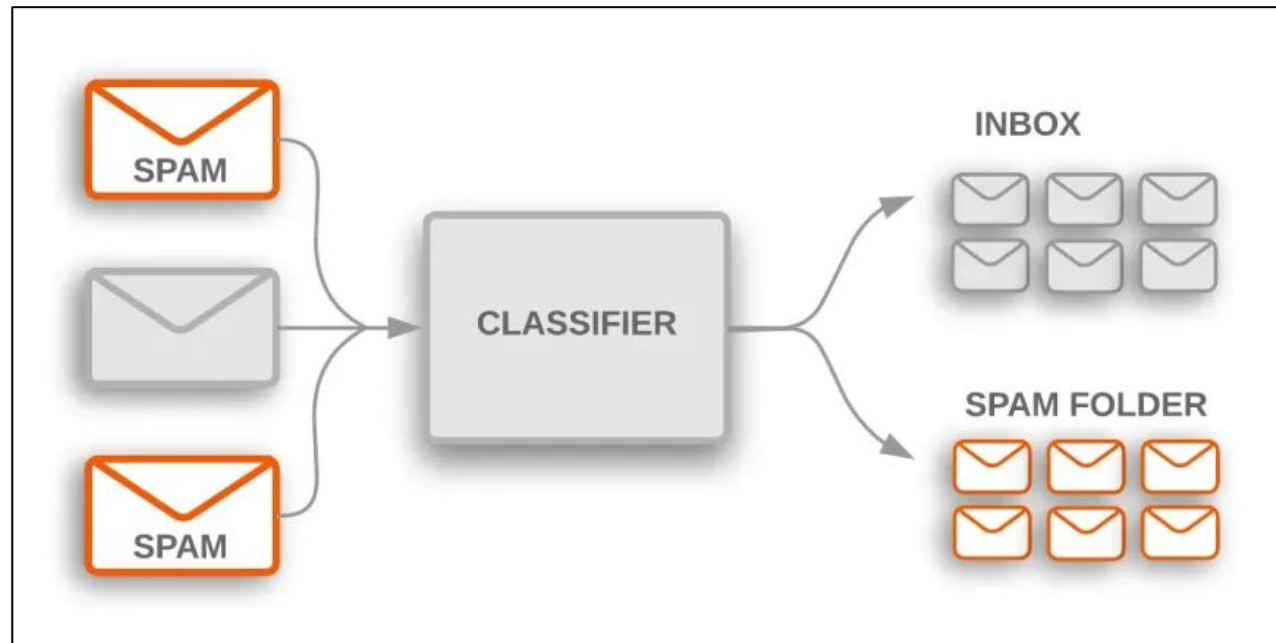




# Applications of Text Classification:

## Email Filtering:

Automatically sort emails into categories such as spam, promotional, or important.



Source

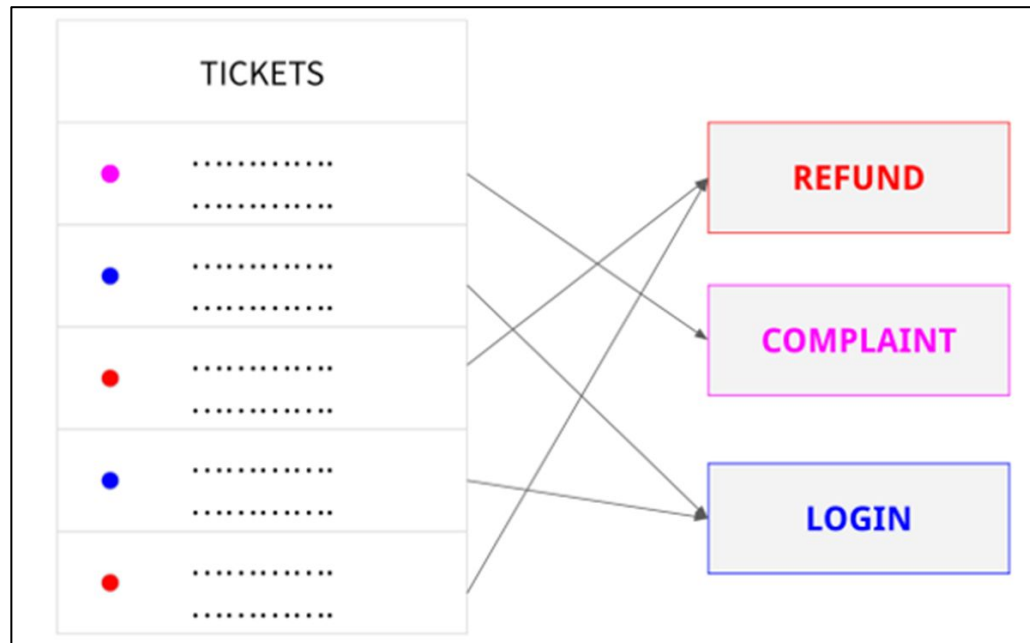




# Applications of Text Classification:

## Customer Service:

Automate the routing of customer queries to the appropriate department or agent.



Source

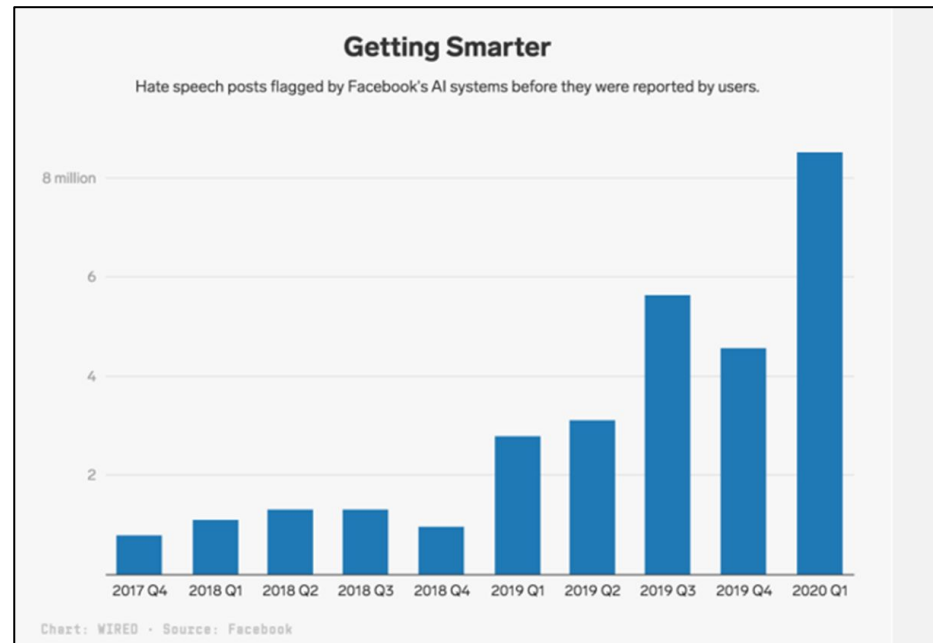




# Applications of Text Classification:

## Social Media Monitoring:

Detect and manage hate speech and harmful content on social media platforms.



Source







# Types of Text Classification





# Types of Text Classification

1. Binary Classification
2. Multi-Class Classification
3. Multi-Label Classification





# Binary Classification

Binary classification is the simplest form of classification, where data is divided into two distinct classes. This approach is used when the task requires deciding between two options, such as yes or no, true or false, or positive and negative categories.

Pick one

Label 1	✓
Label 2	

Binary

Source





# Binary Classification

## Use Cases:

**Spam Detection:** Classifies emails as either spam or not spam (ham), helping filter unwanted messages from inboxes.

**Sentiment Analysis:** Determines whether a text expresses positive or negative sentiment, useful in gauging public opinion.






# Multi-class Classification

Multiclass classification, also known as single-label classification, involves categorizing text into one of several distinct classes. Each text is assigned to one and only one category, making this approach suitable for scenarios where text must belong exclusively to a single group.

Pick one

Label 1
Label 2
Label 3
Label 4 
...
...
Label L

**Multi-class**

Source





# Multi-class Classification

## Use Cases:

**News Article Classification:** Categorizes articles into distinct topics like sports, politics, entertainment, or technology.

**Customer Feedback Analysis:** Classifies feedback into categories such as product quality, delivery issues, or customer service.





# Multi-label Classification

Multi-label classification involves assigning multiple labels to a single text instance, making it ideal for situations where text can simultaneously belong to several categories. This approach is used when multiple relevant topics or attributes are present in the same text.

Pick all applicable

Label 1	
Label 2	✓
Label 3	
Label 4	✓
...	
...	
Label L	✓

**Multi-label**

Source





# Multi-label Classification

## Use Cases:

**Scene Classification:** Tags an image with multiple relevant labels, such as beach, sunset, and people.

**Text Categorization:** Classifies articles with multiple topics, reflecting the variety of subjects covered in the text.





# Differences between Binary, Multi-class and Multi-label Classification



Source





# Approaches to Text Classification





# Approaches to Text Classification

Text classification can be accomplished using various methods, ranging from traditional algorithms to advanced deep learning models. The choice of method often depends on the size of the dataset, the complexity of the task, and the desired level of accuracy.

Traditional Methods	Deep Learning Approaches
Naïve Bayes	Sequential Models (e.g., RNN, LSTM)
Support Vector Machines (SVM)	Convolutional Neural Networks (CNN) for Text
Decision Trees	Transformer Models (e.g., BERT, GPT)





# Key Considerations

**Dataset Size:** Traditional methods may work well for smaller datasets, while deep learning models often require larger datasets for effective training.

**Task Complexity:** For simple binary tasks, traditional models may suffice. For complex tasks with nuanced language understanding, deep learning models offer more power and flexibility.

**Computational Resources:** Deep learning models require more computational power and resources, whereas traditional models can be run on standard hardware.



**Text classification with sequential model**



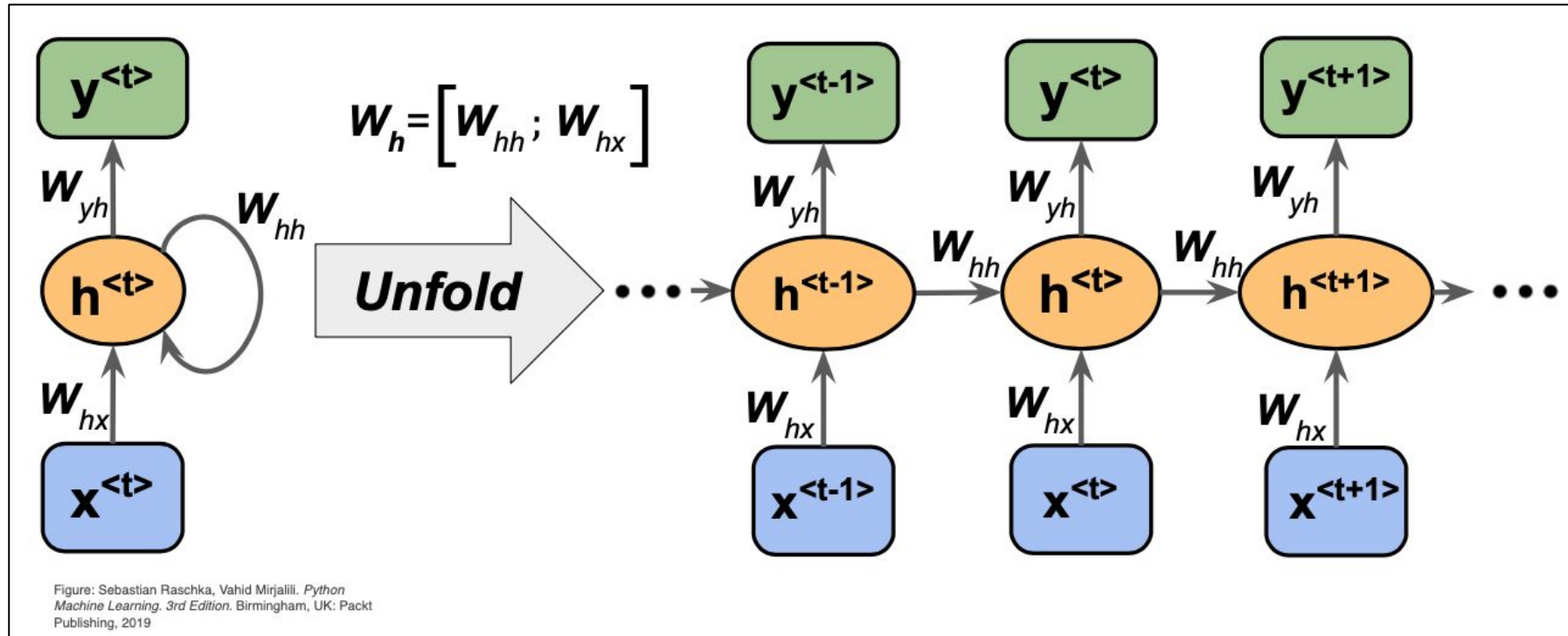
## Recap: What are Sequential Models?

Sequential models, such as RNNs, LSTMs, and GRUs, process data in sequence, making them ideal for tasks where order and context are important, like text classification.



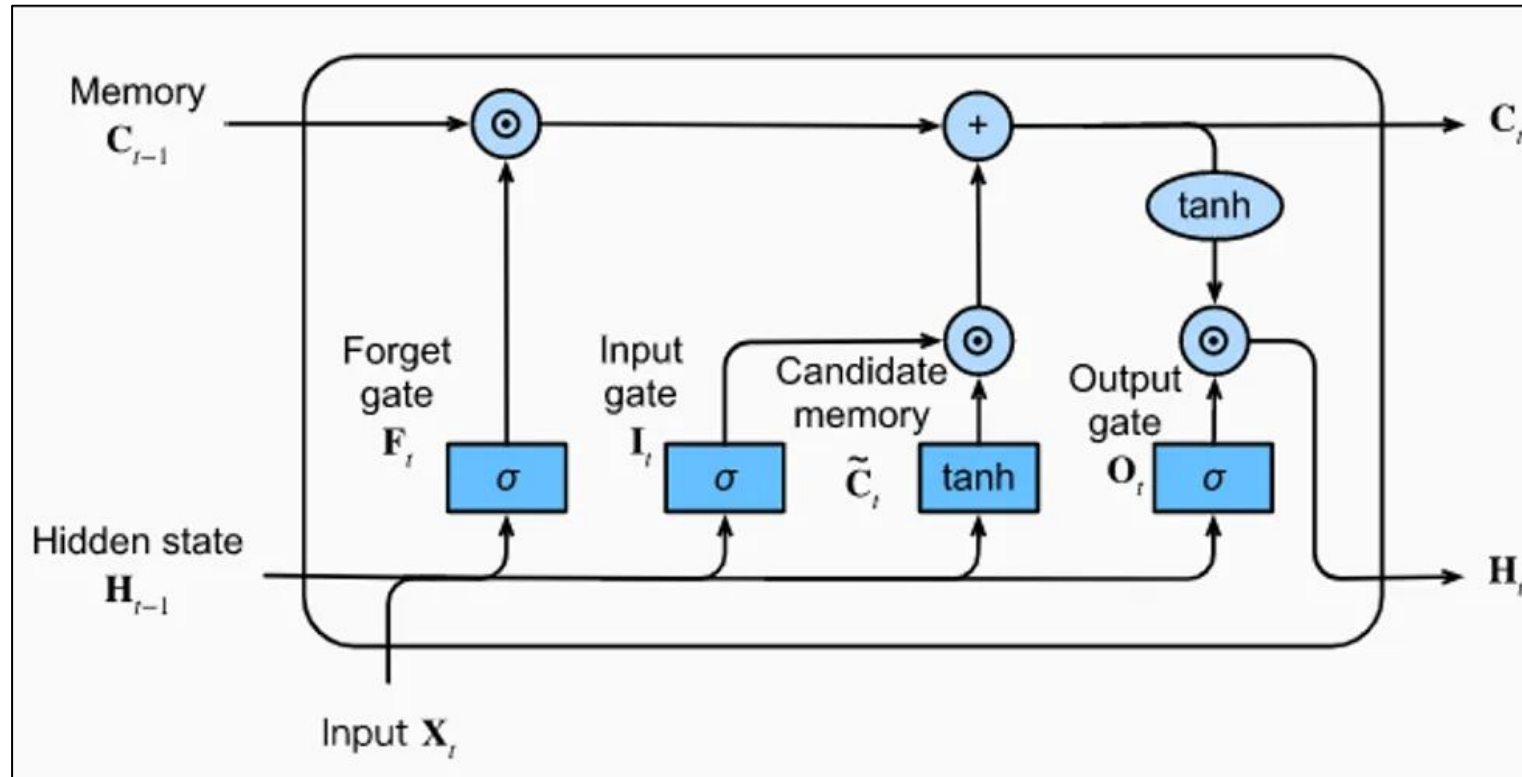


# Recap: Architecture of RNN





# Recap: Architecture of LSTM



Source







# Why Use Sequential Models in Text Classification?

## Capturing Context and Dependencies:

Sequential models, like RNNs and LSTMs, can capture the context and relationships between words, which is crucial for understanding the meaning of sentences.

**Example:** In the sentence "I didn't like the movie," the word "didn't" changes the meaning of "like." Sequential models capture these nuances by processing text in order.





# Why Use Sequential Models in Text Classification?

## Handling Variable-Length Inputs:

Text data can vary significantly in length (from a few words to entire documents). Sequential models process input data one step at a time, making them well-suited to handle variable-length inputs without needing to truncate or pad excessively.

**Example:** A model can process both short product reviews and long blog posts effectively, retaining the essential context and meaning.





# Why Use Sequential Models in Text Classification?

## Learning Long-Term Dependencies:

Sequential models, particularly LSTMs and GRUs, are designed to remember important information over longer sequences, unlike traditional models that may struggle with distant dependencies.

**Example:** In a paragraph, understanding a pronoun ("he") depends on knowing who it refers to earlier in the text. Sequential models can maintain this information over long spans.

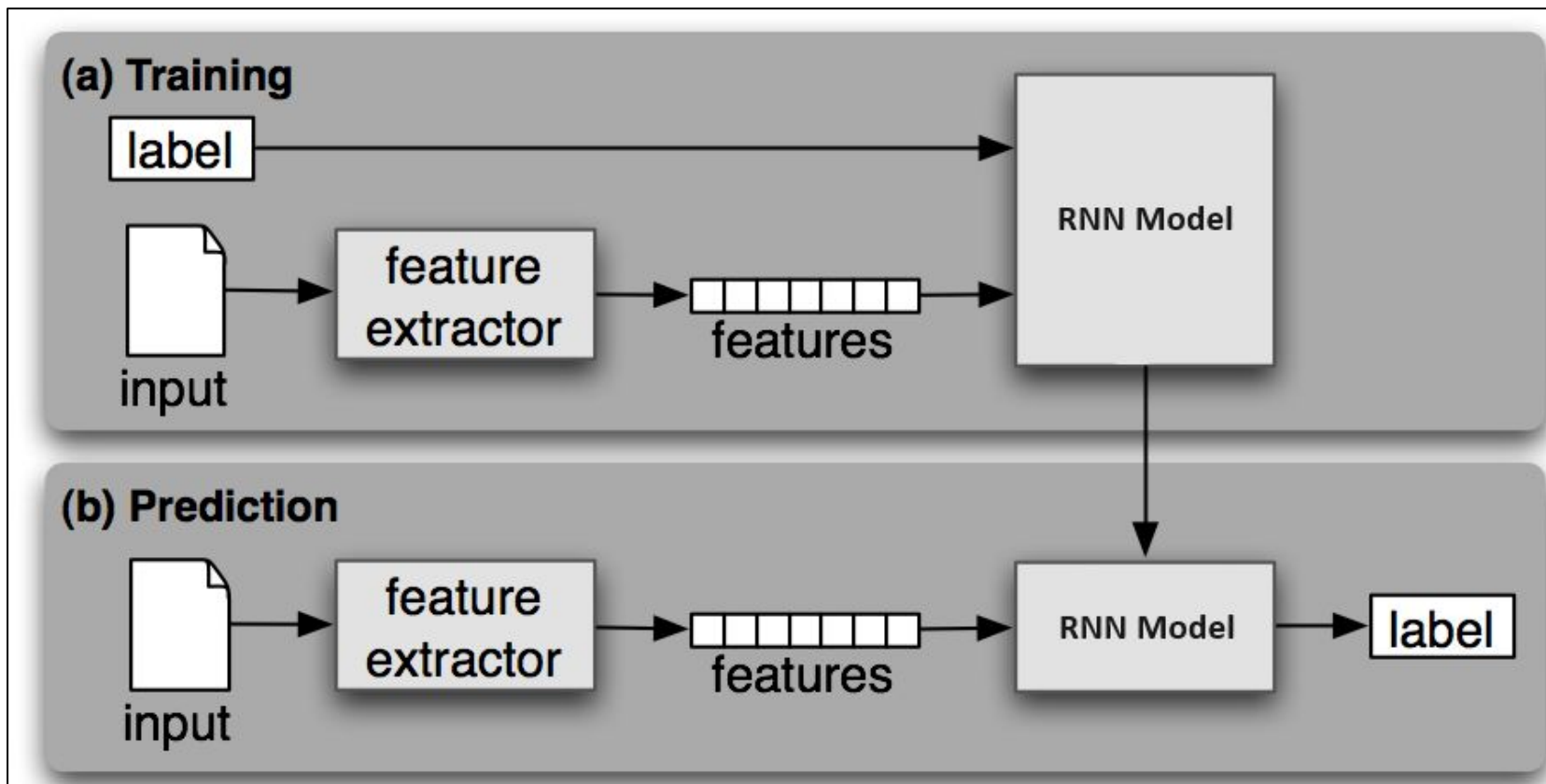




# Text Classification with RNN



# Overview



Source





# Step 1 – Text preprocessing

## Key Steps:

**Lowercasing:** Convert all text to lowercase for consistency.

**Removing Punctuation and Special Characters:** Eliminate non-alphanumeric symbols that do not contribute to meaning.

**Tokenization:** Split text into individual words (tokens) or subwords.

**Stopword Removal:** Remove common words (e.g., “is,” “the,” “and”) that may not carry significant meaning in the context.





# Step 1 – Text preprocessing (cont.)

## Key Steps:

### Stemming and Lemmatization:

- **Stemming:** Reduce words to their root form (e.g., “running” to “run”).
- **Lemmatization:** Convert words to their base or dictionary form, preserving meaning (e.g., “better” to “good”).

**Output:** Cleaned, tokenized text ready for numerical conversion.





## Step 2 – Feature Extraction

Convert text data into numerical representations that models can process.

### Key Techniques:

**Bag of Words (BoW):** Represents text as word frequency counts in a document, ignoring grammar and word order.

**TF-IDF (Term Frequency-Inverse Document Frequency):** Weights terms based on their frequency within a document and across documents, highlighting important words.







## Step 2 – Feature Extraction (cont.)

### Key Techniques:

**Word Embeddings:** Use Word2Vec or GloVe to capture semantic meaning by representing words in a continuous vector space, preserving their contextual relationships.

**Output:** Numerical vectors or matrices representing the text data.





## Step 3 - Model Building Using RNN

Train an RNN model to learn patterns in text data and classify it into the desired categories. **Steps to build the model:**

### 1- Define the RNN Architecture:

- Use an embedding layer to convert input integers into dense vectors.
- Add one or more Recurrent Neural Network (RNN) layers to process sequences of text, capturing dependencies between words.





## Step 3 - Model Building Using RNN (cont.)

Train an RNN model to learn patterns in text data and classify it into the desired categories. **Steps to build the model:**

### 2- Compile the Model:

- Select a loss function appropriate for the classification task (e.g., categorical cross-entropy for multi-class classification).
- Choose an optimizer (e.g., Adam) to adjust the model weights and minimize the loss function.





## Step 3 - Model Building Using RNN (cont.)

Train an RNN model to learn patterns in text data and classify it into the desired categories. **Steps to build the model:**

### 3- Train the Model:

- Feed the model with training data, adjusting weights through backpropagation.
- Use a validation dataset to monitor performance and adjust hyperparameters.





## Step 3 - Model Building Using RNN (cont.)

Train an RNN model to learn patterns in text data and classify it into the desired categories. **Steps to build the model:**

### 4- Evaluate the Model:

Test on unseen data to measure performance using metrics like accuracy, precision, recall, and F1-score. **We will delve deeper into evaluation techniques in the next section.**





# Summary

## **Summary of the Text Classification Process:**

**1- Text Preprocessing:** Clean and prepare text data by tokenization, removing stopwords, and lemmatization.

**2- Feature Extraction:** Convert text into numerical representations using embeddings and padding.

**3- Model Building with RNN:** Design, compile, train, and evaluate an RNN model to classify text data.

**Outcome:** A robust RNN-based text classification pipeline capable of processing raw text input and generating accurate predictions.



# Evaluation of Text Classification Models



# Overview

## Introduction to Evaluation Metrics:

- When evaluating text classification models, it's crucial to go beyond simple accuracy.
- Different metrics provide different insights into the model's performance, especially when dealing with imbalanced data.

## Example Scenario: Email Classification

We will explore these evaluation metrics using an example of a classifier trained to distinguish between **ham (non-spam)** and **spam** emails.







# Accuracy

Let's say You have 100 emails, and a classifier is trained to identify spam.

The classifier **correctly** identifies:

- 85 ham (non-spam) emails.
- 5 spam emails.

$$Accuracy = \frac{Correct\ Prediction}{Total\ Prediction} = \frac{85 + 5}{100} = 0.90 \text{ (90\%)}$$

Now let's discover the limitations of accuracy.





## Limitations of Accuracy

**If the class distribution is 90 ham and 10 spam:**

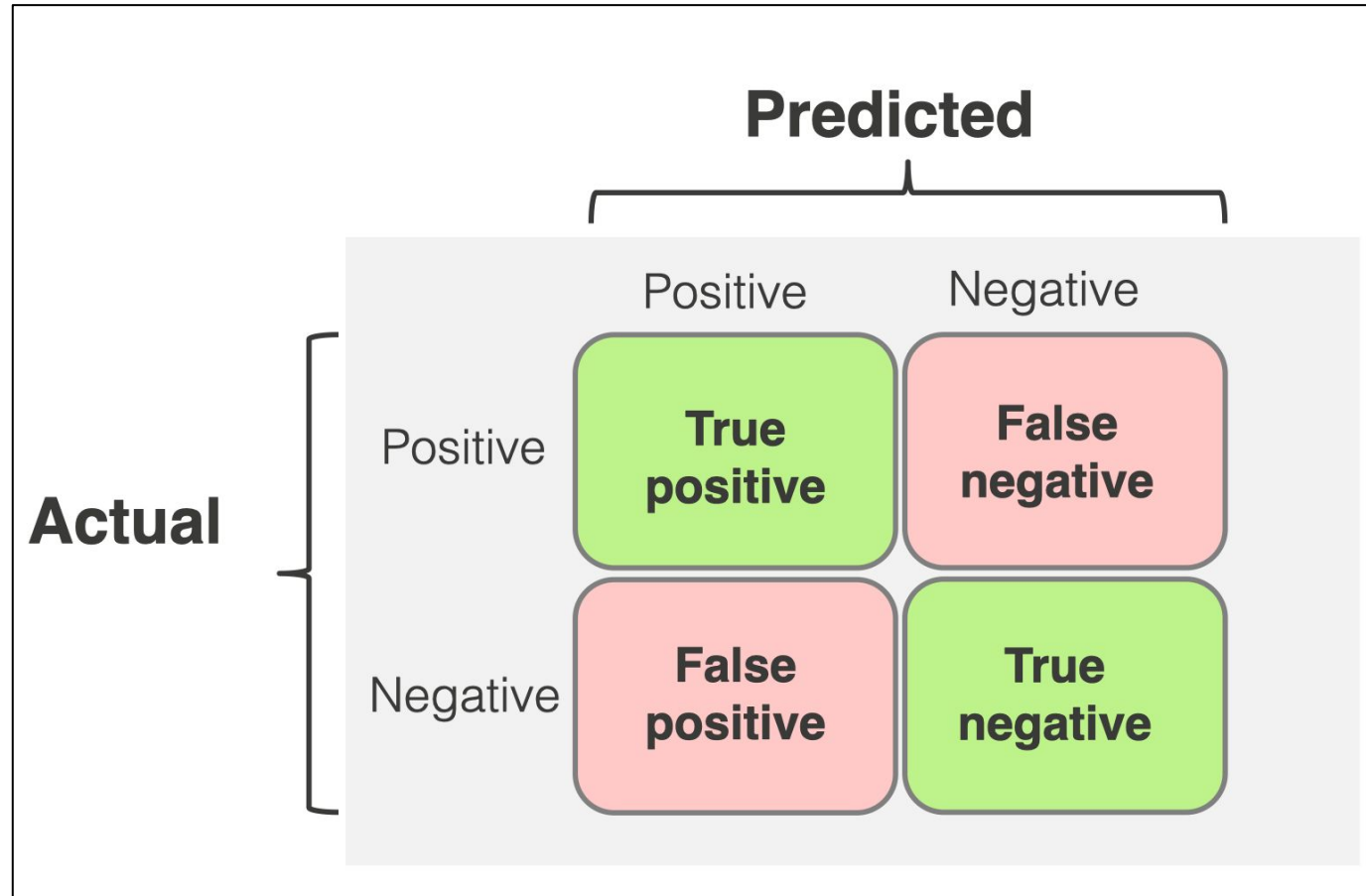
A classifier that always predicts "ham" would still achieve an accuracy of 90%, but it would miss all spam emails.

Therefore, accuracy alone does not provide insights into what the classifier gets wrong.





# Confusion Matrix



Source





# Confusion Matrix

- **True Negatives (TN):** Ham emails correctly classified as ham.
- **False Negatives (FN):** Spam emails incorrectly classified as ham.
- **False Positives (FP):** Ham emails incorrectly classified as spam.
- **True Positives (TP):** Spam emails correctly classified as spam.





# Precision

Precision measures the accuracy of positive predictions.

Formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Use Case: Important in applications where false positives are costly (e.g., spam detection).





# Recall

Recall measures the ability to find all relevant instances (true positives).

Formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Use Case: Crucial when missing positive cases is costly (e.g., disease diagnosis).





## F1-Score

Harmonic mean of precision and recall, balancing both metrics.

Formula:

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

Use Case: Useful when both precision and recall are important.



# Let's Practice

## Tutorial 1:

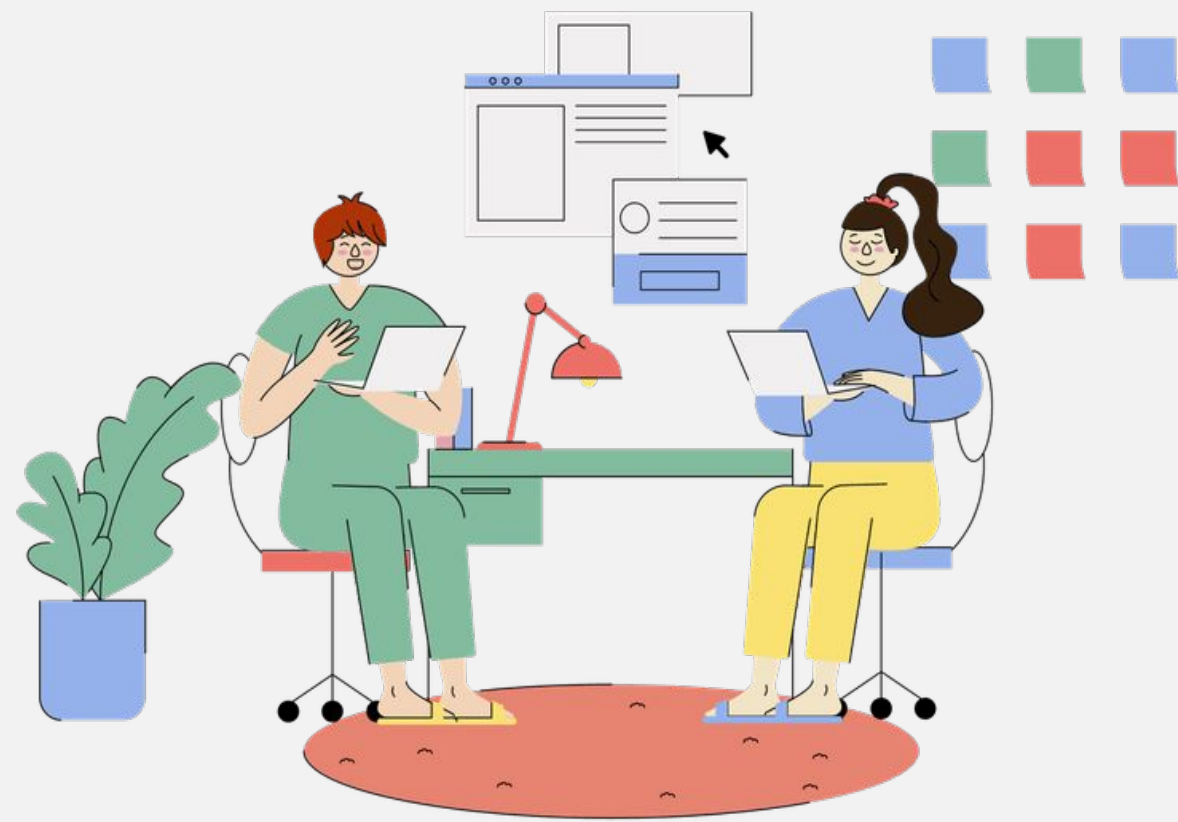
- 7- Introduction to Natural Language Processing/4 - Text Classification with Sequential Models/LAB/Text\_classification\_RNN.ipynb

## Tutorial 2:

- 7- Introduction to Natural Language Processing/4 - Text Classification with Sequential Models/LAB/Text\_classification\_LSTM.ipynb

## Exercise:

- 7- Introduction to Natural Language Processing/4 - Text Classification with Sequential Models/LAB/Text\_classification\_exercise.ipynb





# Thank You



**SDAIA**  
الهيئة السعودية للبيانات  
والذكاء الاصطناعي  
Saudi Data & AI Authority