IR U 4

Introduction to Text Classification

Text Classification is the process of automatically assigning predefined categories or labels to text documents. It is a major task in Natural Language Processing (NLP) and Information Retrieval systems. The goal is to analyze the content of a document and determine the class it belongs to based on patterns learned from training data.

Key Concepts

- **Supervised learning:** The model learns from a labeled dataset.
- **Document representation:** Text is converted into numerical vectors using bag-of-words, TF-IDF, or embeddings.
- **Learning algorithms:** Models such as Naive Bayes, KNN, SVM, and neural networks are used.

Process

1. **Preprocessing:** Tokenization, stop-word removal, stemming, normalization.
2. **Feature Extraction:** Converting text into vectors.
3. **Training:** Model identifies patterns using labeled examples.
4. **Prediction:** New documents are assigned the most likely class.

Applications

- Email spam detection
- Sentiment analysis
- News categorization
- Document indexing and filtering

Advantages

- Scales to large datasets
- Automates manual classification
- Provides quick, consistent decisions

Naive Bayes Model for Text Classification

The **Naive Bayes model** is a probabilistic text classification technique based on **Bayes' theorem**. It classifies a document by calculating how likely it belongs to each class, given the words it contains. The model is called "naive" because it assumes that all words in a document occur **independently** of each other, which simplifies computation.

How Naive Bayes Works in Text Classification

1. The system first learns from a **labeled dataset** where each document has a known category.
2. It calculates how often each word occurs in documents of each class.
3. For a new document, the model evaluates how strongly its words indicate each class.
4. The class that gets the highest overall probability is selected as the predicted label.

Why Naive Bayes Works Well

- Text data is high-dimensional; Naive Bayes handles thousands of features easily.
- The independence assumption works surprisingly well in practice.
- It is very fast and efficient for both training and classification.

Applications

- Spam detection
- Sentiment analysis
- Topic classification
- News categorization

Advantages

- Extremely fast and scalable
- Requires small training data
- Performs well even when data is noisy
- Easy to implement

Limitations

- Fails to capture relationships between words (context)
- Independence assumption is unrealistic but practical

K-Nearest Neighbor (KNN) for Text Classification

K-Nearest Neighbor is a simple, instance-based learning method used for classifying text documents. Unlike other algorithms, KNN does not build an explicit training model. Instead, it classifies a new document based on how similar it is to previously labeled documents.

Working of KNN in Text Classification

1. Convert documents into numerical vectors using representations like TF-IDF.
2. When a new document arrives, compute its similarity to all existing documents using cosine similarity or distance measures.
3. Select the **k nearest documents** (neighbors) that are most similar.
4. Assign the new document to the class that occurs most frequently among these k neighbors.

Strengths

- Very simple and intuitive
- No training phase; the algorithm learns directly from stored data
- Works well for multi-class classification
- Effective for smaller datasets and non-linear decision boundaries

Weaknesses

- Slow classification process since it compares with every document
- Requires large memory to store all training documents
- Sensitive to noise and irrelevant features
- Performance depends heavily on the value of **k** and similarity metric

Applications

- Document recommendation
- Topic classification
- Basic categorization tasks in small systems

KNN is widely used as an easy-to-understand baseline model and performs well when data is clean and properly represented.

Spam Filtering

Spam filtering is a text classification task used to automatically detect and block unwanted email messages. It is one of the earliest successful applications of machine learning in text classification.

Purpose of Spam Filtering

Spam messages often contain phishing attempts, malware links, fake promotions, and advertising content. Filtering helps protect users and ensures inbox cleanliness.

How Spam Filtering Works

1. Collect a dataset of **spam** and **non-spam** emails.
2. Extract features such as word occurrences, sender address patterns, suspicious URLs, and email structure.
3. Train a classification model such as Naive Bayes, Support Vector Machine, KNN, or neural networks.
4. Apply the trained model to new emails to classify them as spam or legitimate.

Techniques Used

- **Naive Bayes filters** (most common due to speed and accuracy)
- **Rule-based filtering** using keywords and patterns
- **Blacklists and whitelists**
- **Machine learning-based classifiers**
- **Heuristic patterns** (e.g., excessive symbols, promotional language)

Common Indicators of Spam

- Words like “FREE”, “WIN”, “LIMITED OFFER”
- Suspicious sender domains
- Embedded tracking links or malware URLs
- Strange formatting or repeated characters

Advantages

- Protects users from phishing and fraud
- Reduces risk of malware infection
- Saves time by keeping inbox clean
- Highly accurate when trained with quality data

Limitations

- Spammers constantly change strategies
- False positives may block important emails
- Requires continuous updating and retraining

Spam filtering remains a crucial real-world application of text classification and forms a core part of email services today.

Support Vector Machine (SVM) Classifier

Support Vector Machine (SVM) is a powerful supervised learning algorithm widely used in text classification because it performs extremely well in high-dimensional spaces. Text data often contains thousands of features (words), and SVM is specifically designed to handle such situations.

Basic Principle

SVM aims to find the **best separating boundary**, known as a *hyperplane*, that divides documents of different classes.
The key idea is to choose the hyperplane that **maximizes the margin**, i.e., the distance between the hyperplane and the nearest data points from each class (called support vectors).
A larger margin generally leads to better generalization and classification accuracy.

Why SVM Works Well for Text

1. Text features are high-dimensional and sparse—SVM handles this efficiently.
2. It is robust to noise and irrelevant features.
3. SVM can model both linear and non-linear patterns using kernel functions.
4. It produces strong classification accuracy even with limited training data.

Key Characteristics

- Uses only the most informative points (support vectors) to define the decision boundary.
- Can handle multi-class classification with appropriate extensions.
- Avoids overfitting by maximizing the margin.

Advantages

- Very high accuracy for text classification tasks.
- Good performance for large feature spaces such as TF-IDF vectors.
- Works well when classes are well separated.
- Effective for spam filtering, sentiment analysis, and topic categorization.

Limitations

- Training can be slow on extremely large datasets.
- Requires careful tuning of parameters such as C and kernel type.
- Less interpretable compared to simpler models.

SVM remains one of the most preferred models for text classification due to its reliability, robustness, and high performance.

Vector Space Classification Using Hyperplanes

The Vector Space Model converts documents into numerical vectors where each dimension corresponds to a term (word) or feature. Classification in this space involves separating these vectors using **hyperplanes**.

Concept of Hyperplanes

A hyperplane is a linear decision boundary that divides the feature space into regions corresponding to different classes.
Text classification models like SVM, perceptron, and logistic regression rely on hyperplanes to determine class membership.

How It Works

1. Convert documents into vectors using representations like Bag-of-Words or TF-IDF.
2. Each document becomes a point in a high-dimensional space.
3. The model identifies a hyperplane that best separates documents of different categories.
4. Documents fall on one side or the other depending on their features.
5. New documents are classified based on their position relative to the hyperplane.

Advantages

- Works very well for high-dimensional text data.
- Provides a clean geometric interpretation of classification.
- Can handle thousands of features without complexity issues.
- Forms the theoretical backbone of many modern classification algorithms.

Applications

- Spam vs non-spam classification
- Topic identification
- Sentiment classification
- Document clustering and categorization

Significance

Hyperplane-based classification is the foundation of modern linear classifiers and enables efficient, scalable text classification systems used in search engines and NLP applications.

3. Kernel Function (9 Marks)

Kernel functions are mathematical tools used to extend linear classifiers like SVM to solve **non-linear classification problems**. They allow the classifier to operate in a higher-dimensional space **without explicitly computing the coordinates** in that space. This is known as the **kernel trick**.

Purpose of Kernel Functions

- Many classification problems cannot be separated by a straight hyperplane.
- Kernel functions transform data into a new space where the classes become linearly separable.
- This transformation is done implicitly, making the computation efficient.

Common Kernel Types

1. **Linear Kernel**
    - Most commonly used in text classification
    - Works very well with high-dimensional sparse text features
    - Fast and produces high accuracy
2. **Polynomial Kernel**
    - Captures interactions between terms
    - Useful for modeling more complex relationships
3. **Radial Basis Function (RBF) Kernel**
    - Handles non-linear decision boundaries
    - Very powerful for complex datasets
4. **Sigmoid Kernel**
    - Related to neural network activation functions
    - Good for specific applications

Advantages of Kernel Functions

- Allow flexible and powerful classifiers.
- Enable SVM to handle complex, non-linear patterns.
- Avoid explicit mapping to huge feature spaces, reducing computation cost.
- Work extremely well for tasks where a linear classifier is insufficient.

Limitations

- Some kernels require parameter tuning.
- Incorrect kernel choice can reduce accuracy.
- Computational cost increases for large datasets.

Significance

Kernel functions extend the power of classifiers and make SVM suitable for a wide range of NLP problems. They are critical for modeling real-world text data which often contains non-linear patterns.

1. Clustering vs Classification (9 Marks)

Clustering and classification are two major techniques in text mining, but they differ fundamentally in purpose, method, and type of data they operate on.

1. Definition

Classification (Supervised Learning)

Classification is a supervised technique where each document is assigned a predefined class label. The system learns from labeled training data and uses this knowledge to classify new documents.

Examples: spam vs ham emails, sentiment classification, news categorization.

Clustering (Unsupervised Learning)

Clustering groups documents into clusters based on similarity without using any labeled data. The algorithm discovers natural groupings or hidden patterns in the data.

Examples: grouping similar news articles, topic discovery, document organization.

2. Training Requirement

- **Classification:** Requires *labeled data*. The model is trained with examples where categories are known.
- **Clustering:** No labels needed. The algorithm infers structure from raw documents.

3. Output

- **Classification:** Predicts a specific class label for each document.
- **Clustering:** Produces clusters (groups) of similar items but does not assign predefined labels.

4. Algorithms

- **Classification:** Naive Bayes, SVM, KNN, Decision Trees.
- **Clustering:** K-means, Hierarchical Clustering, EM, Gaussian Mixture Models.

5. Use Cases

Classification is used when:

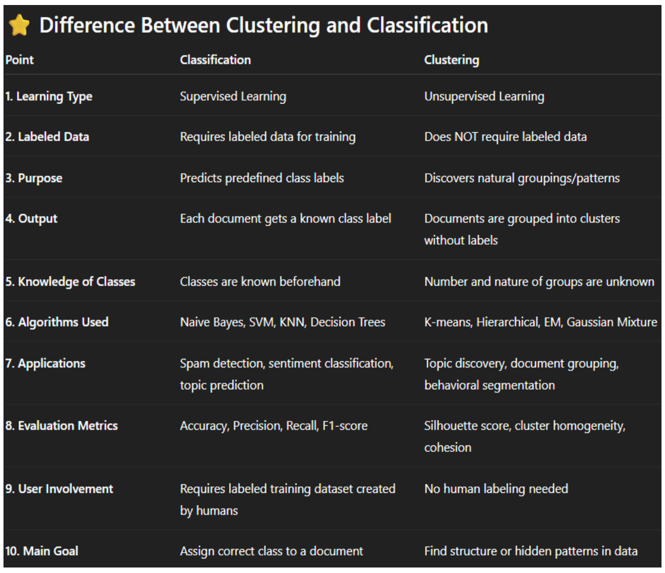
- Categories are known
- Goal is prediction
- Labeled datasets exist

Clustering is used when:

- Structure is unknown
- Goal is pattern discovery or grouping
- Labels are unavailable or expensive to get

6. Evaluation

- **Classification:** Evaluated with accuracy, precision, recall, F1-score.
- **Clustering:** Evaluated with silhouette score, cluster cohesion, or domain knowledge.



⭐ **Difference Between Clustering and Classification**

Point	Classification	Clustering
1. Learning Type	Supervised Learning	Unsupervised Learning
2. Labeled Data	Requires labeled data for training	Does NOT require labeled data
3. Purpose	Predicts predefined class labels	Discovers natural groupings/patterns
4. Output	Each document gets a known class label	Documents are grouped into clusters without labels
5. Knowledge of Classes	Classes are known beforehand	Number and nature of groups are unknown
6. Algorithms Used	Naive Bayes, SVM, KNN, Decision Trees	K-means, Hierarchical, EM, Gaussian Mixture
7. Applications	Spam detection, sentiment classification, topic prediction	Topic discovery, document grouping, behavioral segmentation
8. Evaluation Metrics	Accuracy, Precision, Recall, F1-score	Silhouette score, cluster homogeneity, cohesion
9. User Involvement	Requires labeled training dataset created by humans	No human labeling needed
10. Main Goal	Assign correct class to a document	Find structure or hidden patterns in data

Partitioning Methods

Partitioning methods are a family of clustering techniques that divide a dataset into a fixed number of clusters. Each document is assigned to exactly one cluster based on similarity to a central representative.

1. Basic Idea

Partitioning methods treat clustering as an optimization problem. The dataset is partitioned into **k clusters**, and the algorithm tries to:

- Minimize within-cluster distance (documents in the same cluster are similar)
- Maximize between-cluster distance (different clusters are well separated)

The number of clusters **k** is usually user-defined.

2. Steps in Partitioning Methods

1. **Choose k**, the number of clusters.
2. **Initialize cluster centers** (randomly or using heuristics).
3. **Assign documents** to the nearest cluster based on a similarity/distance measure.
4. **Update centroids** based on newly assigned points.
5. Repeat until the cluster assignments stabilize.

Distance is often computed using **cosine similarity** for text data.

3. Common Partitioning Algorithms

(a) K-Means Clustering

- Most widely used partitioning algorithm.
- Uses mean value of documents as the cluster center.
- Repeats assignment and update steps until convergence.

(b) K-Medoids / PAM

- Uses actual data points (medoids) instead of means as cluster centers.
- More robust to outliers than k-means.

(c) CLARANS (Clustering Large Applications)

- Designed for large datasets.
- Combines sampling with k-medoids for scalability.

4. Strengths of Partitioning Methods

- Simple and easy to implement
- Efficient for large datasets
- Produces compact and meaningful clusters
- Works well when clusters are spherical and of similar size

5. Limitations

- Requires the user to specify the number of clusters (k).
- Sensitive to initial cluster centers.
- Not suitable for clusters with irregular shapes.
- Can get stuck in local optima.

6. Applications

- Text document grouping
- Customer segmentation
- Image clustering
- Topic discovery
- Web page grouping

1. K-Means Clustering for Text Clustering

K-means is one of the most popular algorithms used for **text clustering**, where the goal is to group similar documents together. It is a **partitioning-based** method that divides documents into *k* clusters, each represented by a **centroid**, which is the average vector of documents in that cluster.

How K-Means Works in Text Mining

1. **Choose k**, the number of clusters (topics).
2. **Initialize centroids** randomly from the document vectors.
3. **Assignment Step:**
   Assign each document to the nearest centroid using **cosine similarity**, which works well for text.
4. **Update Step:**
   Recompute each centroid as the mean TF-IDF vector of documents in that cluster.
5. **Repeat** the two steps until assignments stabilize.

Key Features for Text

- Represents documents as high-dimensional vectors (TF-IDF).
- Minimizes distance between documents and cluster centroid.
- Efficient for large collections like news articles, webpages, tweets.

Advantages

- Simple and fast even for millions of documents.
- Easy to implement and understand.
- Suitable for text datasets with well-defined topical clusters.

Limitations

- Requires pre-setting k, the number of topics.
- Sensitive to noisy or outlier documents.
- Clusters may not be meaningful if documents overlap in topics.
- Initialization affects final clusters.

Text Applications

- Topic grouping of news articles
- Clustering search engine results
- Document organization in digital libraries

2. Agglomerative Hierarchical Clustering for Text

Agglomerative hierarchical clustering is a **bottom-up text clustering method** that builds a hierarchical structure called a **dendrogram**. It is highly useful for exploring document similarity relationships.

How It Works

1. Begin with **each document as a separate cluster**.
2. Compute similarity between all pairs using cosine similarity (best for text).
3. **Merge the two most similar clusters**.
4. Update similarity matrix.
5. Repeat merging until all documents form a single tree-like structure.

Linkage Methods

- **Single linkage**: merges closest pair of documents.
- **Complete linkage**: merges clusters based on their farthest documents.
- **Average linkage**: averages distances.
- **Ward's method**: minimizes variance in merged clusters.

Advantages

- No need to specify number of clusters; choose by cutting the dendrogram.
- Reveals hierarchical and nested document groups.
- Useful when studying how documents relate at multiple levels.

Limitations

- Computationally expensive for large text datasets.
- Once two clusters merge, it cannot be undone.
- Sensitive to noisy text or very short documents.

Text Applications

- Organizing research papers
- Building topic hierarchies
- Visualizing similarity between text documents

3. Expectation-Maximization (EM) Algorithm for Text Clustering

EM is a **probabilistic clustering method** which performs **soft clustering**, meaning each document belongs to clusters with certain probabilities. This captures the fact that documents may relate to multiple topics.

Two-Step Process

1. E-Step (Expectation Step)

- Compute the probability that each document belongs to each cluster.
- This uses current cluster parameters (means, variances).

2. M-Step (Maximization Step)

- Update cluster parameters using the probabilities calculated in E-step.
- Refines cluster centroids and spreads.

These steps repeat until there is no significant improvement.

Why EM is Good for Text

- Handles overlapping topics (e.g., a sports-business article).
- Assigns partial membership, which reflects real document behavior.
- More flexible than k-means because clusters can take different shapes.

Advantages

- Produces soft, probabilistic clusters.
- Useful for complex and ambiguous document sets.
- Works well when documents may belong to multiple topics.

Limitations

- Slower and more computationally expensive than k-means.
- Sensitive to initialization; may converge to local optima.
- Often uses k-means to get initial cluster positions.

Text Applications

- Clustering mixed-topic articles
- Soft topic assignment in news or blogs
- Document classification with overlapping themes

4. Mixture of Gaussians Model (GMM) for Text Clustering

A Gaussian Mixture Model assumes that text documents are generated from multiple **Gaussian distributions**, each representing a different cluster or topic. It is one of the most powerful probabilistic clustering methods.

Key Concepts for Text

- Each cluster corresponds to a Gaussian distribution in vector space.
- Documents have **probable membership**, not fixed membership.
- The EM algorithm estimates parameters for each Gaussian.

How GMM Works

1. Initialize Gaussian parameters (mean, covariance, weight).
2. **E-Step:** Compute probability of each document belonging to each Gaussian cluster.
3. **M-Step:** Update the Gaussian parameters using these probabilities.
4. Repeat until convergence.

Advantages

- Can model **elliptical or unevenly shaped** clusters, unlike k-means.
- Handles overlapping document groups naturally.
- Provides a full probabilistic interpretation of text clusters.
- More flexible than k-means because each Gaussian can have its own shape.

Limitations

- Computationally expensive for large text datasets.
- Requires assuming text data fits a Gaussian distribution.
- Sensitive to initialization.

Text Applications

- Topic modelling
- Grouping articles by complex themes
- Anomaly detection in textual logs or emails
- Speaker/document identification