2.2 Text Classification

Text classification is the process of automatically assigning predefined categories or labels to text data. It is widely used in applications such as spam detection, sentiment analysis, topic categorization, and document classification.

Steps in Text Classification:

1. **Text Preprocessing:**
   Converts raw text into a structured form suitable for classification. This involves steps like:
   - Tokenization: Breaking text into individual words or tokens.
   - Stop-Word Removal: Removing common, non-discriminatory words like "the," "is," "and."
   - Stemming/Lemmatization: Reducing words to their root forms (e.g., "running" → "run").
2. **Feature Extraction:**
   Text is transformed into a numerical format that models can process. Common methods include:
   - **Bag of Words (BoW):** Represents text as a frequency count of words.
   - **TF-IDF (Term Frequency-Inverse Document Frequency):** Weighs word importance based on frequency and rarity across documents.
   - Word embeddings (e.g., Word2Vec, BERT): Captures semantic meaning.
3. **Feature Selection (Core Component):**
   Identifying the most relevant features (words or phrases) from the text to improve classification accuracy and reduce computational complexity.

2.2.1 Feature Selection in Text Classification

Why Feature Selection is Critical?

- **High Dimensionality:** Text data often contains thousands of unique words, many of which are irrelevant or noisy.
- **Improved Model Performance:** Selecting relevant features enhances discriminative power and reduces overfitting.
- **Reduced Computational Cost:** By focusing only on key features, training and inference become faster.

Common Feature Selection Methods:

1. **Statistical Measures:**

    - **Gini Index:** Evaluates the discriminative power of a word based on how it splits classes.
    - **Information Gain:** Measures the reduction in entropy when a word is used as a feature.
    - **Mutual Information:** Quantifies how much a word and a class label are dependent.
    - **Chi-Square (χ^2):** Tests the independence of a word and a class, with higher values indicating stronger association.
2. **Preprocessing Techniques:**
    - **Stop-Word Removal:** Eliminates words that appear frequently across all classes but provide little discriminatory power.
    - **Stemming:** Consolidates different forms of a word (e.g., "study," "studied") into a single feature.
3. **Dimensionality Reduction:**
    - **Latent Semantic Indexing (LSI):** Transforms text into a lower-dimensional space while retaining significant relationships.
    - **Principal Component Analysis (PCA):** Identifies patterns and reduces dimensions by finding principal axes of variation.
    - **Supervised LSI (SLSI):** Modifies LSI to incorporate class labels, enhancing class-specific feature reduction.
4. **Domain-Specific Filtering:**
   Features may be selected based on their importance in a specific domain, such as keywords in legal or medical documents.

Statistical Measures for Feature Selection

Statistical measures help evaluate the relevance and discriminative power of features (e.g., words) in text classification. These methods leverage mathematical properties to determine the relationship between words and class labels, focusing on their ability to distinguish between classes effectively.

---

1. Gini Index

The **Gini Index** quantifies the discriminative power of a word by measuring how well it splits class labels.

Definition

For a word w, let $p_i(w)$ represent the conditional probability that a document belongs to class i, given that it contains w. The Gini Index $G(w)$ is calculated as:

$$G(w) = \sum_{i=1}^{k} p_i(w)^2$$

Where k is the total number of classes.

Properties:

- $G(w)$ ranges from $1/k$ to 1.
- **Higher** $G(w)$: Indicates stronger discriminative power. If all documents containing w belong to a single class, $G(w) = 1$.
- **Lower** $G(w)$: Indicates a more uniform distribution of w across classes, with minimal discriminative power. If w is evenly distributed across all classes, $G(w) = 1/k$.

Use Case:

The Gini Index is useful in determining whether a word strongly favors one class or is weakly associated across multiple classes.

- **What is it?**
  It measures how “pure” a word is in splitting classes. If a word perfectly belongs to one class (e.g., "goal" always means sports), it has a high Gini Index.
- **Example:**
  Imagine 10 documents:
- "Sports": 6 documents contain "goal."
- "Politics": 4 documents contain "goal."

Gini Index checks how well "goal" splits the classes. If "goal" appeared only in "sports," it would perfectly separate the classes and have the best Gini Index.

2. Information Gain (IG)

Information Gain measures how much a word reduces uncertainty (entropy) in class labels.

Definition

For a word w, let P_i be the global probability of class i, and $p_i(w)$ the conditional probability of class i given w. Let $F(w)$ represent the fraction of documents containing w. The information gain $I(w)$ is computed as:

$$I(w) = -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \sum_{i=1}^{k} p_i(w) \cdot \log(p_i(w)) + (1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w))$$

Properties:

- **High** $I(w)$: Indicates w significantly reduces uncertainty, meaning it has high discriminative power.
- **Low** $I(w)$: Indicates w contributes little to reducing uncertainty.

Use Case:

Information Gain is particularly effective in selecting words that significantly contribute to class separation.

What is it?
It measures how much uncertainty (confusion) is reduced when a word is used to predict a class.

Example:
Suppose without any words, you don't know if a document is about "sports" or "politics." Adding the word "goal" reduces your confusion because you now know it's likely "sports."
The reduction in confusion is the **Information Gain**.

3. Mutual Information (MI)

Mutual Information measures the dependency between a word w and a class i, indicating whether their co-occurrence is higher or lower than expected under independence.

Definition

For a word w and class i, let P_i represent the global probability of class i, $F(w)$ the fraction of documents containing w, and $p_i(w)$ the conditional probability of class i given w. The mutual information $M_i(w)$ is:

$$M_i(w) = \log\left(\frac{p_i(w)}{P_i}\right)$$

Interpretation:

- $M_i(w) > 0$: w is positively correlated with class i.
- $M_i(w) < 0$: w is negatively correlated with class i.

Global MI Measures:

- **Average MI:**

$$M_{\text{avg}}(w) = \sum_{i=1}^{k} P_i \cdot M_i(w)$$

- **Maximum MI:**

$$M_{\max}(w) = \max_i M_i(w)$$

Use Case:

Mutual Information identifies words that strongly correlate with specific classes and can differentiate between classes effectively.

What is it?
It measures how strongly a word (e.g., "goal") and a class (e.g., "sports") are related.

Example:
If "goal" occurs in "sports" documents 90% of the time, it has a high **Mutual Information** with "sports." If "goal" appears equally in all classes, the mutual information is low because it's not specific to any class.

4. Chi-Square (χ^2) Statistic

The **Chi-Square** statistic tests the independence of a word w and a class i. It measures how much the actual co-occurrence of w and i deviates from what is expected under independence.

Definition

For a word w and class i, let n be the total number of documents, $p_i(w)$ the conditional probability of class i given w, P_i the global fraction of documents in class i, and $F(w)$ the fraction of documents containing w. The χ^2 statistic is:

$$\chi_i^2(w) = n \cdot F(w)^2 \cdot \frac{(p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i)}$$

Global χ^2 Measures:

- Average χ^2:

$$\chi^2_{\text{avg}}(w) = \sum_{i=1}^{k} P_i \cdot \chi_i^2(w)$$

- Maximum χ^2:

$$\chi^2_{\text{max}}(w) = \max_i \chi_i^2(w)$$

Use Case:

The Chi-Square statistic is often used in text classification because of its normalized and interpretable values, especially for comparing features across different categories.

What is it?
It tests if a word (e.g., "goal") and a class (e.g., "sports") are independent. A high χ^2 value means the word and class are strongly connected.

Example:
If "goal" mostly appears in "sports" documents, the χ^2 test will show a strong association between "goal" and "sports." If "goal" appears equally in all classes, χ^2 will be small, showing no strong link.

Comparison of Measures

Measure	Focus	Strength	Limitation
Gini Index	Distribution of words across classes	Simple and interpretable	May be biased by class imbalance
Information Gain	Reduction in entropy	Reflects global and local word importance	Computationally expensive for large datasets
Mutual Information	Dependency between word and class	Captures strong correlations	Can overvalue rare words
Chi-Square	Lack of independence	Normalized, suitable for multi-class problems	Assumes sufficient sample size

Key Difference Between Them

- **Gini Index:** Focuses on how a word splits classes.
- **Information Gain:** Focuses on reducing uncertainty.
- **Mutual Information:** Measures the strength of the relationship between a word and a class.
- **Chi-Square:** Tests whether the word and class are dependent.

2.2.2 Decision Tree Classifiers:

A **Decision Tree** classifier is a machine learning algorithm that splits the data into smaller, more homogeneous groups based on certain conditions. These conditions typically involve testing the presence or absence of words in text data. The process of splitting continues until the data reaches a point where no further useful splits can be made, or a stopping criterion is met (e.g., a minimum number of records in a leaf node).

In text classification, **decision trees** are used to categorize documents based on words or phrases. Each internal node in the tree represents a test condition on an attribute (typically the presence or absence of words), and each leaf node contains a classification label (the predicted category for the document).

How Decision Trees Work in Text Classification:

1. **Recursive Partitioning**:
    - Starting at the root, the tree divides the data space based on certain predicates (conditions). For text, these predicates are usually conditions on whether certain words appear or not in the document.
    - At each step, the decision tree chooses the word that best divides the data into classes.
    - The process continues until the leaf nodes are pure (i.e., most documents in the leaf node belong to the same class) or a stopping condition is met.
2. **Overfitting and Pruning**:
    - Decision trees tend to overfit if they grow too large. This means they may perform well on training data but poorly on unseen data.
    - To avoid this, **pruning** is applied: some branches (nodes) of the tree are removed if they don't improve the model's ability to classify new data.
    - The pruning process is guided by comparing the performance of the tree on a separate validation set.
3. **Splitting Criteria**:
    - **Single Attribute Splits**: At each node, the decision tree looks for the word or phrase that best divides the data. Criteria like **Gini Index** or **Information Gain** can help decide which word to split on.
    - **Multi-Attribute Splits**: Instead of splitting based on a single word, the tree can also use the similarity between documents and word clusters. Documents are grouped based on their similarity to a set of words, and then split according to these similarities.
4. **Types of Splits**:
    - **Single Attribute Split**: This involves selecting one word from the document and using it to split the data into two groups (e.g., does the document contain the word "football"?).

    - **Similarity-based Multi-attribute Split**: This uses the similarity between a document and a group of words or concepts (word clusters). The document is then partitioned into groups based on its similarity to the word clusters.
    - **Discriminant-based Multi-attribute Split**: This technique uses **discriminants** like **Fisher's Discriminant** to split the data in a way that maximizes the separation between different classes.

2.2.3 Rule-based Classifiers

Rule-based Classifiers are closely related to Decision Trees, with both approaches using a set of conditions or rules to classify data. However, while Decision Trees create a hierarchical partitioning of the data space, rule-based classifiers allow for more flexibility, including overlaps and multiple rules firing for a given test instance. Below is a detailed explanation of rule-based classifiers and their relation to Decision Trees in text classification:

1. Rule-based Classifiers Overview

- In a **rule-based classifier**, the data space is represented by a set of rules. Each rule consists of two parts:
    - **Left-hand side (LHS)**: A condition or a set of conditions on the features (e.g., presence of specific words in a document).
    - **Right-hand side (RHS)**: The class label associated with the condition.
- A test instance is classified based on which rules are satisfied. If multiple rules are satisfied, the predicted class is determined by considering the class labels of the satisfied rules. A variety of ranking methods are used to resolve conflicts, such as picking the class label with the highest confidence.

2. Key Concepts

- **Support**: Refers to the absolute number of training instances that satisfy both the LHS and RHS of a rule. A rule with higher support (e.g., satisfied by 50,000 out of 100,000 documents) is considered more important.
- **Confidence**: Refers to the conditional probability that the RHS of the rule is satisfied, given that the LHS is satisfied. This measures the strength of the rule.

These two metrics are common in rule-based classifiers and are used to evaluate the significance and reliability of the rules.

3. Types of Rules

- **Simple Conditions**: For text data, rules are often represented as conjunctions (AND conditions) of terms that must all be present in the document. These rules focus on the presence of terms (e.g., "Honda" and "Toyota" ⇒ "Cars").

- **Avoidance of Union in Rules**: In text classification, using union (OR conditions) in a rule is rare because it can lead to overly broad and less informative rules. For instance, the rule "Honda ∪ Toyota ⇒ Cars" is typically split into two separate rules: "Honda ⇒ Cars" and "Toyota ⇒ Cars."

4. Decision Trees and Rule-based Classifiers

- **Relation to Decision Trees**: Both Decision Trees and rule-based classifiers aim to model data using conditions on features. In Decision Trees, a path from root to leaf can be seen as a rule, where the predicates along the path form a conjunction. A key difference is that Decision Trees use a hierarchical partitioning, whereas rule-based classifiers allow for overlaps in the decision space.
- **Conversion from Decision Trees to Rules**: A path in a Decision Tree can be transformed into a rule. For example, a decision tree path that tests "Honda" and "Toyota" can be represented by a rule like "Honda ⇒ Cars" or "Toyota ⇒ Cars."

5. Advantages of Rule-based Classifiers

- **Flexibility**: Rule-based classifiers can handle overlaps between rules, making them more flexible than Decision Trees, which have strict partitions. This flexibility allows them to easily adapt to new data.
- **Interpretability**: Rules are highly interpretable, making it easier to understand how decisions are made. This interpretability is particularly beneficial in domains where expert knowledge can be manually encoded into the rules.
- **Handling Sparse Data**: In sparse data (e.g., text data), where many words may not appear in every document, rule-based classifiers perform better because they focus on the presence of specific terms or term combinations.

6. Training and Rule Generation

- **Rule Generation Process**: During the training phase, rules are generated based on conditions like support and confidence. These rules are derived from frequent term combinations or patterns that are associated with a class.
- **Iterative Rule Learning**: One method for rule generation involves iteratively selecting the most confident rule for a particular class, applying it, and removing the corresponding instances from the training set. This process continues until no strong rules can be found.

7. Handling Rule Conflicts

- When multiple rules apply to a test instance, conflicts can arise (i.e., the rules may predict different classes). To resolve this, classifiers often rank the rules based on confidence and select the class that appears most frequently among the top rules.

8. Notable Rule-based Techniques

- **RIPPER**: A well-known rule-based classifier that identifies frequent word combinations related to a particular class. It has been found to perform well in situations with smaller training sets.
- **Sleeping Experts**: This technique generates rules based on sparse phrases in the documents, considering the proximity of words. It assigns weights to rules based on their specificity in the training data and combines the weights of multiple rules during classification.

9. Benefits of Rule-based Classifiers in Text Classification

- **Maintenance and Adaptability**: Rule-based classifiers can be easily updated when new examples are encountered. Adding or modifying rules is straightforward, making them easy to maintain over time.
- **Expert Knowledge Integration**: Domain-specific expert knowledge can be manually added to the rule set, allowing for enhanced classification in specialized areas.
- **Performance**: Techniques like RIPPER and Sleeping Experts have demonstrated excellent performance in various text collections, especially in cases with smaller datasets.

10. Challenges

- **Overlapping Rules**: While rule-based classifiers allow overlaps, managing the conflicts between overlapping rules can be complex.
- **Skewed Feature or Class Distributions**: Like Decision Trees, rule-based classifiers may suffer from misinterpretation if the distribution of features or classes in the dataset is skewed.

Scenario: Classifying documents into two categories: Sports and Technology

Let's assume we have a collection of documents, and we want to classify them into two categories: **Sports** and **Technology**. We use a rule-based classifier where each rule is based on the presence of specific terms in the document.

Training Data:

- Document 1: "The football match was exciting, and the players were incredible."
- Document 2: "The new smartphone has amazing features, including an upgraded camera."
- Document 3: "The tech industry is rapidly evolving, with advancements in AI and robotics."
- Document 4: "I love watching basketball and soccer games with my friends."
- Document 5: "The new AI research paper discusses deep learning algorithms."

Rule Generation:

Based on the training data, we generate a set of rules. We will focus on conjunctions of terms (i.e., the presence of certain words).

1. **Rule 1 (for Sports)**:
   If the document contains the words "football" **AND** "players" ⇒ **Sports**
2. **Rule 2 (for Sports)**:
   If the document contains the words "basketball" **AND** "games" ⇒ **Sports**
3. **Rule 3 (for Technology)**:
   If the document contains the words "smartphone" **AND** "features" ⇒ **Technology**
4. **Rule 4 (for Technology)**:
   If the document contains the words "AI" **AND** "algorithms" ⇒ **Technology**

Test Document:

Document: "The new smartphone has an upgraded camera and a faster processor."

Classification Process:

1. **Check against Rule 1**:
   - Does the document contain both "football" and "players"?
     No, so Rule 1 is not satisfied.
2. **Check against Rule 2**:
   - Does the document contain both "basketball" and "games"?
     No, so Rule 2 is not satisfied.
3. **Check against Rule 3**:
   - Does the document contain both "smartphone" and "features"?
     Yes, Rule 3 is satisfied, so the document is classified as **Technology**.
4. **Check against Rule 4**:
   - Does the document contain both "AI" and "algorithms"?
     No, so Rule 4 is not satisfied.

Final Classification:

Since Rule 3 is satisfied, the document is classified as **Technology**.

Interpretability:

- **Rule 3** indicates that the presence of both "smartphone" and "features" in the document is a strong indicator that the document belongs to the **Technology** category.
- If a new document contains terms like "AI" or "deep learning", it could trigger Rule 4, associating the document with **Technology** as well.

Handling Conflicts:

- If multiple rules were satisfied for a test document (e.g., a document containing both "football" and "smartphone"), then the classifier could use additional criteria such as **confidence** or a **weighted rule set** to resolve conflicts and determine the most likely class.

Advantages:

- **Flexibility**: Rules can overlap, and the system can easily adapt to new terms or classes by adding new rules.
- **Interpretability**: The reasoning behind classification is clear, as it directly correlates with the terms or phrases in the document.
- **Adaptability**: If new training examples are added, new rules can be created without a complete retraining process.

2.2.4 Proximity-based classifiers

Proximity-based classifiers are a family of classification algorithms that rely on distance or similarity measures to assign labels to data points. Here's a breakdown of the key concepts and methods discussed:

Key Concepts

1. **Assumption**: Proximity-based classifiers operate on the assumption that documents or data points of the same class are closer to each other in the feature space, based on measures like the cosine similarity or dot product.
2. **Distance Measures**: These classifiers typically rely on mathematical metrics to quantify the closeness between instances. Commonly used metrics include cosine similarity, Euclidean distance, and others.

Methods for Classification

1. k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors (k-NN) algorithm is a simple, non-parametric, and intuitive classification method. It relies on the assumption that data points of the same class are closer to one another based on some distance metric.

- **Mechanism**:
  - Identifies the k closest training instances to a given test instance.
  - Assigns the majority class among these k neighbors as the class label.
- **Parameter k**:
  - Typically ranges between 20 and 40 depending on corpus size.
- **Variants**:
  - Feature selection or weighting terms based on class relevance can enhance accuracy.
  - Temporal weighting, which decays document importance over time, is another enhancement.

Imagine you're classifying emails as **Spam** or **Not Spam** using text data.

Training Data (Features: Word Frequencies)

Email ID	"Buy"	"Free"	"Meeting"	"Project"	Class
1	3	4	0	0	Spam
2	0	0	3	4	Not Spam
3	2	3	0	0	Spam
4	0	0	4	5	Not Spam

Test Email

The new email contains the following word frequencies: `"Buy": 2`, `"Free": 3`, `"Meeting": 1`, `"Project": 1`.

Steps

1. **Distance Calculation**: Compute similarity (e.g., cosine similarity or Euclidean distance) between the test email and all training emails.
2. **k-NN Decision**: Let k=3. Identify the three closest neighbors based on the distance metric.
3. **Class Assignment**:
    - If the majority of neighbors belong to **Spam**, classify the email as **Spam**.
    - If the majority belong to **Not Spam**, classify it as **Not Spam**.

Example Calculation

Using Euclidean distance:

$$\text{Distance from Email 1 to Test Email} = \sqrt{(3-2)^2 + (4-3)^2 + (0-1)^2 + (0-1)^2} = 1.73$$

Similarly, compute distances for all emails. Select the 3 nearest emails. Suppose they are Email 1(spam), Email 3 (Spam), and Email 4 (Not Spam). The majority class is **Spam**, so the test email is classified as **Spam**.

2. Aggregated Training Data

- **Approach**:
  - Groups documents of the same class into clusters during pre-processing.
  - Summarizes each cluster into a single meta-document (generalized instance).
- **Advantages**:
  - Reduces the computational overhead by summarizing data.
  - Potentially improves accuracy by mitigating outlier influence.

---

3) WHIRL Method

- Performs soft similarity joins using text attributes.
- Documents are matched based on approximate similarity, not exact matches.
- Adaptable for use in nearest neighbor classification.

The **WHIRL (Weighted Hybrid Integration Rule-based Learner)** method is a proximity-based classification technique that integrates text similarity with nearest neighbor classification. It is primarily designed for text data and performs classification based on **soft similarity joins**.

Key Characteristics of WHIRL:

1. **Soft Similarity Joins**:
   - Instead of requiring exact matches between text attributes, WHIRL allows partial matches using similarity measures like cosine similarity or Jaccard similarity.
2. **Text Representation**:
   - Text documents are represented as vectors, where each term's weight can be derived using methods like TF-IDF.
3. **Nearest Neighbor Classification**:
   - WHIRL finds the nearest neighbors to the test document based on similarity scores, then assigns the label of the closest or majority neighbors.

Steps in WHIRL Method:

1. **Preprocess Text**:
   - Convert training and test documents into numerical vectors using methods like **TF-IDF**.
2. **Calculate Similarities**:
   - Compute the similarity between the test document and training documents using a soft similarity measure (e.g., cosine similarity).
3. **Select Nearest Neighbors**:
   - Identify the kkk-nearest neighbors based on similarity scores.
4. **Classify Test Document**:
   - Assign the class label of the majority neighbors to the test document.

Example: Classifying Articles

Training Data:

Article ID	Text	Class
1	"AI advancements in deep learning."	Technology
2	"New vaccines improve health."	Health
3	"Machine learning for big data."	Technology
4	"Benefits of a balanced diet."	Health

Test Article:

"Deep learning in healthcare."

Step 1: Represent Text as Vectors (TF-IDF):

Using **TF-IDF**, assign weights to terms based on their importance.

Term	Article 1	Article 2	Article 3	Article 4	Test Article
AI	0.5	0.0	0.3	0.0	0.0
advancements	0.7	0.0	0.0	0.0	0.0
deep	0.8	0.0	0.0	0.0	0.6
learning	0.7	0.0	0.8	0.0	0.6
vaccines	0.0	0.9	0.0	0.0	0.0
healthcare	0.0	0.8	0.0	0.0	0.7
diet	0.0	0.0	0.0	0.8	0.0

Step 2: Compute Cosine Similarity:

1. Similarity with Article 1: 0.94
2. Similarity with Article 2: 0.63
3. Similarity with Article 3: 0.91
4. Similarity with Article 4: 0

Step 3: Identify Nearest Neighbors:

Sort by similarity:

1. Article 1 (0.94, Technology)
2. Article 3 (0.91, Technology)
3. Article 2 (0.63, Health)

Step 4: Classify Test Document:

Majority class among the nearest neighbors:

- Technology: 2 votes
- Health: 1 vote

Predicted Class: **Technology**

Advantages of WHIRL:

1. Handles partial matches and similarity-based joins effectively.
2. Well-suited for text classification tasks.
3. Combines benefits of proximity-based and similarity-based classification.

4)) Rocchio Framework

The **Rocchio Method** is a proximity-based classification technique used in text classification. It works by creating a **meta-document (or centroid)** for each class during the training phase. A test document is then classified by comparing its similarity with these meta-documents.

Key Steps in the Rocchio Method

1. **Compute Term Weights for Meta-Documents**:
   - A **meta-document** for each class is created by combining all the training documents of that class.
   - Term weights are calculated using the formula:

$$f_{\text{rocchio}} = \alpha_p \cdot f_p - \alpha_n \cdot f_n$$

where:

- f_p: Frequency of the term in the positive class (class of interest).
- f_n: Frequency of the term in the negative class (other classes).
- α_p, α_n: Weighting parameters, typically $\alpha_p = 1$ and $\alpha_n = 0.5$.

2. Classify Test Document:

- For the test document, compute its similarity (e.g., cosine similarity) with each class's meta-document.
- Assign the class label of the most similar meta-document.

3.Solved Example

Scenario

We are classifying a test document into two categories: **Technology** and **Health**.

- **Training Data**:
  - **Technology Documents**:
    - "AI advancements in machine learning."
    - "New algorithms for big data analysis."
  - **Health Documents**:
    - "Meditation improves mental health."
    - "Health benefits of balanced nutrition."
- **Test Document**:
  - "AI for healthcare advancements."

Step 1: Preprocessing

Tokenize and normalize the text (lowercase, remove stop words, etc.).

- **Technology**:
  - Tokens: `["ai", "advancements", "machine", "learning", "new", "algorithms", "big", "data"]`
- **Health**:
  - Tokens: `["meditation", "improves", "mental", "health", "benefits", "balanced", "nutrition"]`
- **Test Document**:
  - Tokens: `["ai", "healthcare", "advancements"]`

Step 2: Compute Term Frequencies

Term	Frequency in Technology (fpf_pfp)	Frequency in Health (fnf_nfn)
AI	2	0
advancements	1	0
machine	1	0
learning	1	0
new	1	0
algorithms	1	0
big	1	0
data	1	0
meditation	0	1
improves	0	1
mental	0	1
health	0	2
benefits	0	1
balanced	0	1

nutrition	0	1

Step 3: Compute Rocchio Weights

Using $\alpha_p = 1$ and $\alpha_n = 0.5$:

Term	Technology Weight	Health Weight
AI	$1 \cdot 2 - 0.5 \cdot 0 = 2$	$1 \cdot 0 - 0.5 \cdot 2 = -1$
advancements	$1 \cdot 1 - 0.5 \cdot 0 = 1$	$1 \cdot 0 - 0.5 \cdot 1 = -0.5$
machine	$1 \cdot 1 - 0.5 \cdot 0 = 1$	$1 \cdot 0 - 0.5 \cdot 1 = -0.5$
learning	$1 \cdot 1 - 0.5 \cdot 0 = 1$	$1 \cdot 0 - 0.5 \cdot 1 = -0.5$
health	$1 \cdot 0 - 0.5 \cdot 2 = -1$	$1 \cdot 2 - 0.5 \cdot 0 = 2$
meditation	$1 \cdot 0 - 0.5 \cdot 1 = -0.5$	$1 \cdot 1 - 0.5 \cdot 0 = 1$

Step 4: Test Document Vector

For the test document: "AI for healthcare advancements."

- Tokens: `["ai", "healthcare", "advancements"]`

- Vector: [AI: 1, healthcare: 1, advancements: 1, others: 0][

Step 5: Cosine Similarity Calculation

Cosine Similarity Formula:

$$\text{Cosine Similarity} = \frac{\vec{D_{\text{test}}} \cdot \vec{D_{\text{meta}}}}{\|\vec{D_{\text{test}}}\| \cdot \|\vec{D_{\text{meta}}}\|}$$

5.1: Similarity with Technology Meta-Document

- Meta-Document Vector: $[\text{AI: } 2, \text{advancements: } 1, \text{healthcare: } 0]$
- Dot Product:

$$\vec{D_{\text{test}}} \cdot \vec{D_{\text{meta}}^{\text{Technology}}} = (1 \cdot 2) + (1 \cdot 1) + (1 \cdot 0) = 3$$

- Magnitudes:

$$\|\vec{D_{\text{test}}}\| = \sqrt{(1^2 + 1^2 + 1^2)} = \sqrt{3}$$

$$\|\vec{D_{\text{meta}}^{\text{Technology}}}\| = \sqrt{(2^2 + 1^2 + 0^2)} = \sqrt{5}$$

- Cosine Similarity:

$$\text{Similarity} = \frac{3}{\sqrt{3} \cdot \sqrt{5}} \approx 0.775$$

5.2: Similarity with Health Meta-Document

- Meta-Document Vector: $[\text{AI: } -1, \text{advancements: } -0.5, \text{healthcare: } 2]$
- Dot Product:

$$\vec{D_{\text{test}}} \cdot \vec{D_{\text{meta}}^{\text{Health}}} = (1 \cdot -1) + (1 \cdot -0.5) + (1 \cdot 2) = 0.5$$

- Magnitudes:

$$\|\vec{D_{\text{meta}}^{\text{Health}}}\| = \sqrt{((-1)^2 + (-0.5)^2 + (2)^2)} = \sqrt{5.25}$$

- Cosine Similarity:

$$\text{Similarity} = \frac{0.5}{\sqrt{3} \cdot \sqrt{5.25}} \approx 0.126$$

Step 6: Final Classification

- **Technology Similarity**: 0.775
- **Health Similarity**: 0.126

The test document **"AI for healthcare advancements"** is classified as **Technology**, as it has a higher similarity score with the Technology meta-document.