Research Paper Classification System

Anusha Dayanand
ad22bj@fsu.edu
Florida State University

Usha Kiran Mayee Vangeepuram
uv22@fsu.edu
Florida State University

Tazmeen Alam
ta22u@fsu.edu
Florida State University

I. Introduction

In response to the escalating volume of research papers across diverse fields, this study introduces a novel classification system designed to facilitate efficient paper retrieval and categorization for users. The system employs advanced techniques, utilizing abstracts as a basis to extract representative keywords. Leveraging Latent Dirichlet Allocation (LDA), it identifies underlying topics within the papers. Further enhancing this approach, the study employs Term frequency-inverse document frequency (TF-IDF) values, implementing the K-means clustering and hierarchical clustering algorithms to group papers with shared subjects. This systematic classification method significantly streamlines the intricate process of organizing research papers, concurrently improving accessibility for users in search of pertinent content.
Within the K-means clustering algorithm, the study strategically applies the Elbow method to pinpoint the optimal number of clusters, optimizing the grouping process. Hierarchical clustering creates tree-like structures called dendrograms based on data point proximity, by merging and splitting clusters recursively to help visualize relationships and similarities between data points. Methods such as the Elbow Method, Silhouette Analysis, or Dendrogram Cutting determine the optimal number of clusters to ensure that the chosen number of clusters provides meaningful and interpretable results. Additionally, the study uses Silhouette coefficient, Sum of squared errors (SSE) and Sum of squared distances (SSD) as validation metrics for assessing the effectiveness of the clustering results.
In essence, this comprehensive classification system develops as a strong response to the growing difficulty of handling the growing amount of research. It profoundly improves the user experience by offering a systematic and accessible framework for identifying and categorizing relevant scholarly content, in addition to organizational benefits.

II. Literature Review

The number of digital text documents in the last 20 years has grown exponentially which puts immense importance on classifying the documents into similar groups or categories based on their content (Trstenjak et. Al, 2013). It is humanly tedious to classify the documents hence, automatic methods have been developed over the years to improve the speed and efficiency of automated document classification. Text classification is one of the problems of text mining which is an interdisciplinary field that draws on information retrieval, data mining, machine learning, statistics, and conceptual linguistics (Trstenjak et. Al, 2013)
The classification of science into different disciplines has a long history, and it is a fundamental step for demarcating research areas, helping information scientists categorize publications into specific fields (Zhang et. al, 2022). The primary purpose of a classification system is to delineate and define these research areas clearly in a meaningful and responsible way particularly while comparing research across diverse scientific fields. Over the recent past years, researchers have identified numerous ways to categorize academic publications. One of these branches categorizes journals, particularly emphasizing the classification of individual papers within a journal under its associated discipline while another uses a machine learning approach that works at the article level rather than its source of publication. They observed that journal-level classification systems have two main limitations: they offer only a limited amount of detail, and they have difficulties with multidisciplinary journals (Singh et. al, 2020).
One of the solutions to classification problems at an article level is the text-based algorithmic approach. Researchers employed many approaches to compare and comprehend publications such as articles or papers. They examined the titles, summaries, and topic matter of these publications. They either utilized a combination of word analysis and language structure or attempted to organize articles using machine learning techniques that learn from examples, with a concentration on information in titles and summaries. Both studies sought to improve how humans organize and comprehend information by employing algorithms to discern patterns within textual data (Zhang et. al, 2022).
Keyword searches within the academic information system are used to identify potential subjects while excluding fewer common phrases. However, this technique has limitations in terms of morpheme classification. To solve these challenges, one solution is to create a dictionary that concentrates on the researcher's specialized keywords, particularly those that are uncommon. Hence, when looking for literature, the latent Dirichlet allocation (LDA) can be applied as a useful paper-writing tool to classify the searched papers by category based on automatically provided keywords. The qualities and efficiency of new technology terms in recent trend articles based on LDA modeling can be demonstrated using this approach (Kim & Kim,2022). Word-cloud analysis is advised for trend analysis, which includes summarizing gathered academic articles, finding new technology trends, measuring word frequency, and determining keyword similarity.

III. Methodology

A. Traditional TD-IDF

The frequency with which a feature word appears in the document collection determines its weight in the TF-IDF algorithm. Although the TF-IDF algorithm takes into account the relationship between feature words and the volume of texts in which they appear, it ignores the differences in how feature words are distributed across different categories, which is detrimental to improving classification accuracy.

B. Improved TD-IDF

The existing discipline of TD-IDF is used as a reference in an updated IDF approach to distinguish feature words that occur in several disciplines with differing frequency and have various meanings. To increase the confidence in the outcomes of the TF-IDF algorithm, separate the meanings of feature words in various subjects using the current subject categorization. Using the improved TD-IDF algorithm, it is possible to improve the accuracy of the categorization of documents by establishing an association between feature words and the quantity of texts while ignoring the variation in feature words i.e. different feature words having different meanings in different domains.

C. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) stands out as a widely favored topic model where the topic word distribution within the LDA model is employed to compute topic features for words, enhancing the quality of text features. This involves filtering out words with less prominent features. LDA represents a probabilistic extension of both Latent Semantic Indexing (LSI) and Probabilistic Latent Semantic Indexing (PLSI) which facilitates large-scale automatic mining of topics within text . However, the model's effectiveness exhibits some instability. The improved LDA-based classification employs refined Linear Discriminant Analysis to reduce feature dimensions. Subsequently, the Extreme Learning Machine algorithm is employed as the classifier in this academic framework.

This study incorporates a range of supervised learning algorithms, including Random Forest and AdaBoost, alongside multi-label classification techniques utilizing Support Vector Machines (SVM) and Naïve Bayes. Furthermore, unsupervised learning methodologies, specifically K-means and hierarchical clustering, are employed for data categorization, all of which will be elaborated upon in the subsequent sections of this paper.

D. Support Vector Machine

The Support Vector Machine (SVM) is a machine learning algorithm for tackling two-class pattern recognition problems. The approach works in a vector space, looking for a decision surface that best divides data points into two classifications. SVM employs a non-linear mapping to translate the input data into a higher-dimensional space when the data is not linearly separable. The separation is then accomplished by identifying a hyperplane in this new space, which is aided by a kernel function that computes vector similarities. The One-Against-the-remainder strategy is often used for multi-class classification, considering each class as a distinct group against the remainder for binary classification. This method creates hyperplanes to identify each class from the others, so tackling the multi-class recognition difficulty. The dimensionality of individual words is approximately ten times greater than that of multi-word expressions, and SVM exhibits the capability to efficiently handle input spaces with high dimensions. In the context of categorization problems, it is observed that Least Square Twin Support Vector Machine (LS-TWSVM) outperforms other SVM variants, by demonstrating superior generalization ability and computational speed.

E. Random Forest

Decision trees based on recursive partitioning is an unstable classification model as the first splitting variable is sensitive to changes i.e. a small change in the learning data has a significant impact on the subsequent structure of the tree. Even though the combination of multiple decision trees helps overcome the instability of the model, too many similar decision trees affect the accuracy of classification adversely. On the contrary, increasing the diversity of the trees can improve the accuracy and Random Forest algorithm achieves this goal through training dataset randomization. By using random extraction and substitution, RF creates numerous fresh training data subsets. Since each training subset is created at random, it is expected that each tree's development in RF will be independent of the others. Once every tree in the RF is built, the classification results can balance the effect of training data and create stability in the RF. The classification results from RF are proven to be accurate.

F. Adaboost

Boosting, a prominent ensemble method, enhances learning accuracy in machine learning but faces two key challenges: adjusting the training set for effective implementation of weak classifiers and combining these trained weak classifiers to form strong classifiers. To tackle these issues, the AdaBoost (Adaptive Boosting) algorithm was introduced by Freund and Schapire. It operates by dynamically adjusting weights, optimizing the training set for weak classifier training, and iteratively combining their predictions. AdaBoost has emerged as a representative and effective method in ensemble learning, contributing significantly to predictive capabilities and model robustness.

G. K-means Clustering

Clustering methods are commonly employed for categorizing a dataset into groups of similar data, making them applicable in various domains such as marketing, pattern recognition, web mining, and social network analysis. Due to its ease of use, effectiveness and higher computation speed, K-means clustering is the most preferred algorithm as by reducing the distances between each data item and the centroid of its related cluster, it divides a dataset of N items into k different subsets. Grouping similar research papers into clusters could be effectively achieved by extracting keywords from paper abstracts and utilizing TF-IDF and LDA schemes. These schemes concentrate on evaluating the importance of each publication, and as a result, the system may perform better in terms of clustering and have higher F-score values.

H. Hierarchical Clustering

Hierarchical clustering, an effective technique in document organization, establishes a hierarchy structure for document collections, facilitating efficient browsing and navigation. This structure aids in concealing irrelevant information, enhancing user experience but faces challenges in computational cost and memory usage, particularly when handling a large volume of daily documents. To address these issues, a scalable hierarchical clustering framework that leverages topic frequency in documents, offers a more efficient and memory-friendly solution. Rather than creating an NxN similarity matrix for hierarchical tree construction, the scalable framework utilizes a low-dimensionality frequency matrix to represent the root cluster. This matrix is recursively split as the hierarchy descends, substantially reducing memory requirements. The implementation relies on distributed computing architecture, enabling scalability to handle a growing number of documents based on available resources. The clustering method involves three phases: binary tree construction, branch breaking, and meta-clustering for graphical representation. Metrics including Identity, Similarity, Entropy, and Bin Similarity drive cluster formation.

IV. Implementation

A. Dataset

In this project, our objective is to conduct experiments using diverse datasets, primarily focusing on research papers. The research paper abstracts are a critical feature of the datasets we picked for our investigation. These abstracts are succinct descriptions of the studies' key contributions, methodology, and findings.

Two key datasets that we plan to utilize for our experiments are as follows:

1) Dataset 1: NSF Research Award Abstracts dataset from UCI : This dataset was obtained from the UCI Machine Learning resource, which is a well-known for machine learning datasets. It contains abstracts of research publications related to NSF (National Science Foundation) research grants. The abstracts cover a wide range of topics and provide insight into various scientific pursuits.

2) Dataset 2: Research Paper Abstract dataset from Kaggle: This dataset, sourced from Kaggle, offers a rich variety of themes and domains, contributed by a diverse community, making it a valuable tool for researchers and practitioners.

B. Data Preprocessing

Data preprocessing involves cleaning, transforming, and organizing the data to enhance its quality and make it suitable for further processing. We performed data preprocessing to enhance the quality of textual data by removing the stopwords, punctuations, converting all words to lowercase and tokenizing the data. This preprocessing is particularly useful for refining the text data to improve subsequent analyses and model training.

C. Word Clouds

The text mining algorithm that uses LDA to identify significant themes or topics within a large collection of documents is based on constructing a word-to-id mapping, transforming tokenized text data into a bag-of-words corpus and then training the statistical model of LDA designed to discover abstract ”topics” that occur in a collection of documents. Word clouds are visual representations of text data, where the size of each word indicates its frequency or importance in the corresponding topic.

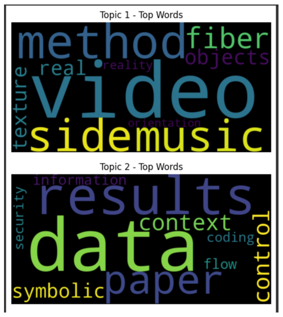


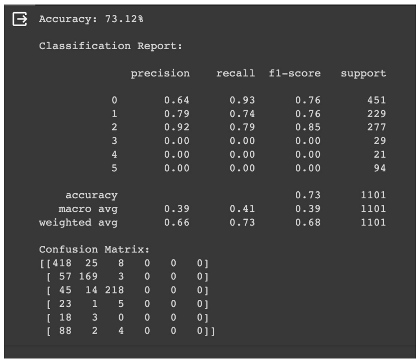
Fig. 1.

For example from Fig. 1:
Topic 1 seems to be about video and music, indicated by words like ”video”, ”music”, ”real”, ”objects”, ”fiber”, ”texture”, and ”orientation”. This suggests that the topic may be related to multimedia, virtual reality, or video processing where these terms are relevant.
Topic 2 appears to be related to data and results, as shown by prominent words like ”data”, ”results”, ”information”, ”context”, ”security”, ”symbolic”, and ”paper”. This indicates that the topic might be concerned with data analysis, research findings, information security, or academic publications where such terms are common.

V. Experimental Results

A. Random Forest

Random forest was implemented on **Dataset (2)**.
The model's **Accuracy** is 73.12%, which means that the model



```
Accuracy: 73.12%

Classification Report:

              precision    recall  f1-score   support

           0       0.64      0.93      0.76       451
           1       0.79      0.74      0.76       229
           2       0.92      0.79      0.85       277
           3       0.00      0.00      0.00        29
           4       0.00      0.00      0.00        21
           5       0.00      0.00      0.00        94

    accuracy                           0.73      1101
   macro avg       0.39      0.41      0.39      1101
weighted avg       0.66      0.73      0.68      1101

Confusion Matrix:
[[418  25   8   0   0   0]
 [ 57 169   3   0   0   0]
 [ 45  14 218   0   0   0]
 [ 23   1   5   0   0   0]
 [ 18   3   0   0   0   0]
 [ 88   2   4   0   0   0]]
```

Fig. 2.

correctly predicted the class of 73.12% of the samples in the test set.
The classification report is a performance evaluation metric that gives a deeper insight into the accuracy of a classifier. The classification report for each class are as follows:

- **Precision** values for each class:
  Class 0: 0.64 (64% of instances predicted as class 0 are actually class 0)
  Class 1: 0.79 (79% precision means that 79% of instances predicted as class 1 are actually class 1)
  Class 2: 0.92 (92% precision indicates a very high reliability in predictions for class 2)
  Classes 3, 4, and 5 have a precision of 0.00, indicating that there were no correct predictions for these classes, or they were not predicted at all.
- **Recall** values:
  Class 0: 0.93 (93% of actual class 0 instances were correctly predicted)
  Class 1: 0.74 (74% of actual class 1 instances were correctly predicted)
  Class 2: 0.79 (79% of actual class 2 instances were correctly predicted)
  Classes 3, 4, and 5 have a recall of 0.00, which means that the model failed to identify any of the actual instances of these classes correctly.
- **F1-Score** are as follows:
  Class 0: 0.76
  Class 1: 0.76
  Class 2: 0.85
  Classes 3, 4, and 5 have an F1-Score of 0.00, indicating a poor performance for these classes.
- **Support** is the number of actual occurrences of the class in the dataset. It is the actual number of samples in the class that should have been predicted. The support values are:
  Class 0: 451
  Class 1: 229
  Class 2: 277
  Class 3: 29
  Class 4: 21
  Class 5: 94

The **overall accuracy** of the model across all classes is 73.12%, but this accuracy can be misleading, especially when the data is imbalanced which means that if certain classes dominate the dataset, a high accuracy might not reflect the model's performance across all classes, which seems to be the case here with the model performing poorly on classes 3, 4, and 5.
The **macro average** gives equal weight to each class, which highlights the poor performance on the minority classes, as shown by the macro average F1-Score of 0.39.
The **weighted average** metric takes his takes into account the support for each class. It gives a weighted average of the precision, recall, and f1-score, where each class's metric is weighted by the number of true instances in each class. This provides a measure of quality that reflects the prevalence of each class in the data. A better average F1-Score of 0.68 is seen, since it is influenced more by the majority classes.
Confusion Matrix shows the number of correct and incorrect predictions broken down by each class. Each row of the matrix corresponds to a true class, while each column corresponds to a predicted class. The diagonal from top left to bottom right (418, 169, 218, 5, 0, 0) shows the number of correct predictions for each class (0 to 5). The off-diagonal elements show the number of incorrect predictions, e.g., the model predicted 25 instances of class 0 as class 1, 8 instances of class 0 as class 2, and so on.
From these metrics, we can infer that the model is good at predicting classes 0, 1, and 2, but it fails to predict classes 3, 4, and 5 which drastically affects its overall performance. This could be due to a class imbalance, lack of representative features for these classes in the model, or not enough instances of these classes in the training data. Steps should be taken to improve the model's ability to classify the underperforming classes, possibly by gathering more data for these classes, applying resampling techniques to address class imbalance, or reevaluating the features used for training the classifier.

B. AdaBoost

Adaboost was implemented on **Dataset (2)**.
base_classifier = DecisionTreeClassifier(max_depth=1)
DecisionTreeClassifier is a classifier that bases decisions on a series of questions about data point features, commonly implemented through the scikit-learn library's DecisionTreeClassifier class. The parameter max_depth=1 indicates that the trees will be very simple, consisting of a single decision node plus two leaf nodes. Such a tree is also known as a decision stump and is weak by itself in making accurate predictions.
AdaBoost works by creating a series of weak learners, which are improved iteratively. In this case, the weak learner is a decision tree classifier (DecisionTreeClassifier) with a max_depth of 1, which means each tree is a stump (one root node with two leaf nodes). This simplicity is typical for boosting algorithms, where the ensemble's power comes from

combining many weak learners.

ada_classifier = AdaBoostClassifier(base_classifier)
AdaBoost which stands for Adaptive Boosting, is an ensemble method that amalgamates multiple weak learners, typically decision stumps, to create a robust learner. It operates by sequentially fitting weak learners on modified data versions and combines their predictions through a weighted majority vote or sum to yield the ultimate prediction. Implemented through the AdaBoostClassifier class in scikit-learn, it enables the specification of the base classifier. The algorithm iteratively enhances the model by prioritizing previously misclassified samples, contributing to its adaptability and effectiveness.
For AdaBoost, the two main hyperparameters are:

- **n_estimators:** This refers to the number of weak learners to use. In the context of AdaBoost, more estimators usually mean a more complex model, as the algorithm will iteratively add learners to correct the mistakes of the previous ones. The grid search will try 50, 100, and 200 trees. However, adding too many learners can lead to overfitting, especially if the learning rate is not tuned properly.
- **learning_rate:** This parameter shrinks the contribution of each classifier. There is a trade-off between 'learning_rate' and 'n_estimators'. A smaller learning rate means that the method can take more steps to converge and may require more estimators, while a larger learning rate hastens the boosting process at the risk of overshooting and poor generalization. The grid search will test learning rates of 0.01, 0.1, and 1.0.

Grid Search: GridSearchCV is a method used to perform hyperparameter optimization. It tries out all possible combinations of parameters defined in param_grid, evaluates each combination using cross-validation, and determines the best set. cv=5 indicates that a 5-fold cross-validation is used, which means the training set is split into 5 smaller sets, and the model is trained and validated 5 times.
Evaluation (F-score): The F-score on the test set is reported to be 0.24 which is not a very high score, suggesting that the model is not strong. The model may not be performing well in terms of balancing precision and recall for the positive class which could be due to an imbalance in the dataset, non-informative features, or hyperparameters that are not optimal for the given data beyond those tested in the grid search.

C. Naïve Bayes

naive_bayes_model = GaussianNB()
Gaussian Naive Bayes classifier from Python's scikit-learn library, represents a straightforward probabilistic classifier that applies Bayes' theorem under the assumption of feature independence. The reported **accuracy** of the Naïve Bayes model is 79.01% indicating that the model is relatively effective for the given classification task hence, thereby depicting strong performance.

D. SVM

svm_model = SVM(kernel="rbf",degree=2)
SVM classifier from Python's scikit-learn library, represents a straightforward probabilistic classifier that applies Bayes' theorem under the assumption of feature independence. The reported **accuracy** of the SVM model is 79.01% indicating that the model is relatively effective for the given classification task hence, thereby depicting strong performance. The SVM classifier is often effective for high dimensional datasets and Non-linear decision boundaries.

E. K-means

K- means clustering was implemented on **Dataset (1)**.
Assumptions:
Number of clusters =25
The SSE value was initially high when we began with 15 and 20 clusters. To mitigate this, we increased the number of clusters, resulting in a reduction of the SSE value.

In unsupervised learning, K-means clustering is a method used to divide a set of data points into K =25 clusters, where each data point belongs to the cluster with the nearest mean. n_clusters=num_clusters specifies the number of clusters to form as well as the number of centroids to generate. The random_state=42 parameter ensures that the algorithm's random number generation for centroid initialization is reproducible; that is, running the algorithm multiple times will produce the same results if given the same input data and parameter settings.

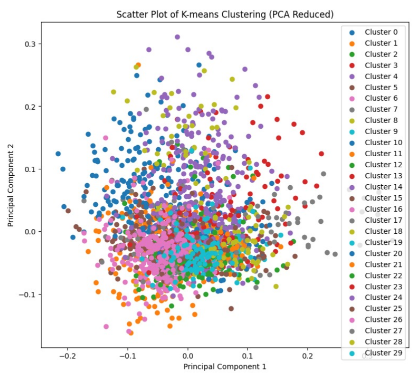


Fig. 3.

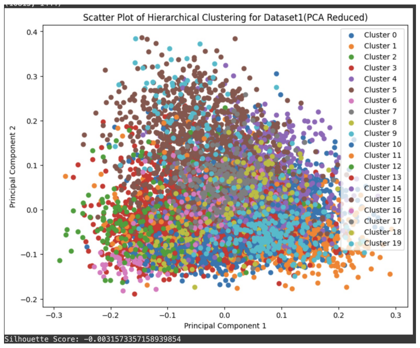
The choice of the number of clusters K is crucial and can greatly affect the SSE. If K is set too low, the SSE will be high because the clusters will be very generalized and contain a wide spread of data points. Conversely, if K is set too high, the SSE could be very low, potentially leading to overfitting where each data point becomes its own cluster. To find an

optimal number of clusters, techniques such as the elbow method are often used, which involve plotting the SSE for a range of K values and looking for a point where the rate of decrease sharply changes.
The SSE is a measure of how close each data point is to its cluster's centroid. It is calculated by summing the squared distances from each data point to its nearest centroid.
SSE value reported is 2296.851190979879.

F. Hierarchical Clustering

Hierarchical clustering was implemented on both **Dataset (1)** and **Dataset (2)**.
Assumptions:
Number of clusters =20
The distribution of data points among the clusters was non-uniform when initiating the clustering process with 5, 10, and 15 clusters. An optimal distribution was achieved with 20 clusters.

Dataset (1): The utilized method involves Agglomerative Clustering, a hierarchical clustering technique that operates in a bottom-up fashion. In this approach, each individual observation initially constitutes a distinct cluster, and successive merging of pairs of clusters occurs as one ascends through the hierarchy.



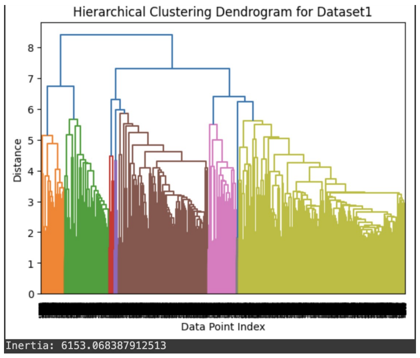
```
Silhouette Score: -0.0031573357158939854
```

Fig. 4.

A Silhouette Score of 0.00315 (Fig. 4) is very close to zero, which suggests that the clusters are overlapping and there is no clear distinction between them. This implies that the clustering structure may have very weak or no substantial structure.
An inertia value of 6153.068 (Fig. 5) suggests that the total squared distance within clusters is somewhat high, indicating that the clusters are not very tight and there is variance within them. This could also hint that the number of clusters chosen may not be optimal, or the data does not cluster well.

Parameters used:

- **Number of Clusters (20):** The number of clusters chosen here are 20 hence, the algorithm will stop combining clusters when there are 20 remaining.
- **Linkage (”ward”):** The linkage criterion determines the metric used for the merge strategy. ”Ward” minimizes the variance of the clusters being merged. It is a popular method because it tends to create more evenly sized clusters, but it may not always provide the best results for all types of data.
- **Metric (”euclidean”):** This is the distance metric used to compute the linkage. ”Euclidean” distance is the straight-line distance between two points in Euclidean space, which is the most common choice of distance metric.

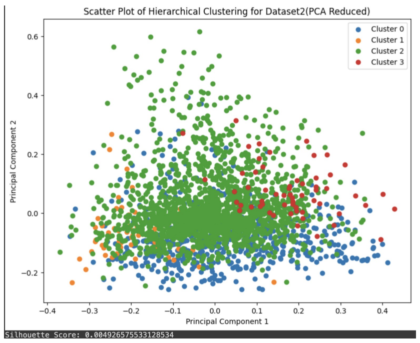


```
Inertia: 6153.068387912513
```

Fig. 5.

Based on the evaluation of the silhouette score and inertia, it appears that the clustering method employed may not effectively capture significant structures within the dataset. This limitation could stem from various factors, including the selection of the cluster count, the characteristics of the data, or the mismatch between the clustering algorithm and the inherent structure of the dataset.
To enhance results, it is advisable to explore alternative configurations such as varying the number of clusters, employing different distance metrics, or considering an alternative clustering algorithm.

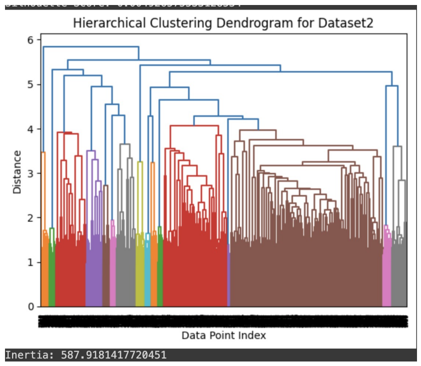
Dataset (2): Similar to Dataset (1), the utilized method involves Agglomerative Clustering.



```
Silhouette Score: 0.004926575533128534
```

Fig. 6.

The value of Silhouette coefficient ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. A



```
Inertia: 587.9181417720451
```

Fig. 7.

Silhouette Score of 0.0049 (Fig. 6) is very low, suggesting that the clusters are not well separated and there is considerable overlap between them. This means that the clusters are not distinct from each other, and the data points might not be grouped in the most meaningful way.
An inertia value of 587.9 (Fig. 7), which is considerably lower than the inertia for dataset1, suggests that the data points are closer to their respective cluster centers. Since inertia depends on the number of clusters and the scale of the data, this lower value could indicate tighter clusters or just reflect the smaller number of clusters compared to dataset1.
Despite the lower inertia observed in hence, typically indicative of improved clustering, the Silhouette Score remains in close proximity to zero, suggesting poorly defined clusters and potential significant overlap. This outcome may be attributed to the inherent distribution and characteristics of the data or the possibility that the ”ward” linkage criterion may not be optimal for this specific dataset.
To enhance clustering outcomes, one could explore variations in the number of clusters, experiment with different linkage criteria (such as ”average,” ”complete,” or ”single”), or consider preprocessing steps like normalization or dimensionality reduction. Additionally, visualization tools like dendrograms can aid in comprehending cluster formation and determining an appropriate number of clusters.

VI. Evaluation Metrics

A. Sum of squared errors (SSE)

Sum of Squared Errors (SSE) is a metric to evaluate the performance of a clustering algorithm by calculating the sum of the squared differences between each observation in a cluster and the cluster’s mean. The objective in clustering is often to minimize SSE. When data points are closer to the centroid of their respective clusters, the SSE is lower, indicating a more compact and well-defined clustering.

B. Silhouette Coefficient

The Silhouette Coefficient is a clustering evaluation metric that quantifies how well-defined and separated clusters are within a clustering configuration. It considers both the distances between clusters and within clusters. The coefficient yields a value between -1 and 1, where higher values indicate more distinct clusters. This metric is widely used in clustering analyses to objectively assess the quality and appropriateness of clustering results.

C. Sum of squared distances (SSD) Inertia

”Sum of Squared Distances” (SSD) refers to a metric used in clustering algorithms to assess the compactness or dispersion of data points within clusters. Specifically in K-means, it is also known as ”within-cluster sum of squares” (WCSS) or ”inertia.” Inertia serves as a metric to assess cluster compactness by computing the sum of squared distances between individual data points and their respective cluster centroid. Lower inertia values indicate a higher degree of compactness, implying more closely grouped data points within clusters.

D. Precision

Precision is a measure of a classifier’s exactness. It measures the accuracy of the positive predictions made by the model and is calculated as the ratio of true positives to the sum of true positives and false positives. A high precision means that an algorithm returned substantially more relevant results than irrelevant ones, while a low precision indicates many false positives (instances incorrectly labeled as positive).

E. Recall

Recall measures a classifier’s completeness. It is the ratio of true positives to the sum of true positives and false negatives, and it measures the model ability to identify true positives relative to the total number of actual positive instances. A high recall means that an algorithm returned most of the relevant results, while a low recall indicates many false negatives (instances that were positive but incorrectly labeled as negative).

F. F Score

The F-Score (or F1 Score) is the harmonic mean of precision and recall and is used to balance the trade-off between these two metrics. It measures how good paper classification is when compared with ground truth. A high F1 Score is an indicator of a robust model with good performance in both precision and recall. The F1 Score tends to favor models that have similar precision and recall.

G. Confusion Matrix

The confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The confusion matrix provides a detailed breakdown of a model’s performance and allows for the calculation of various performance metrics, including precision, recall, and the F1 score. The 2x2 confusion matrix for binary classification consists of:

- True Positives (TP): Correctly predicted positive instances.
- True Negatives (TN): Correctly predicted negative instances.

- False Positives (FP): Incorrectly predicted positive instances (a.k.a. Type I error).
- False Negatives (FN): Incorrectly predicted negative instances (a.k.a. Type II error).

VII. Conclusion

In constructing our system for categorizing research papers, we harnessed a spectrum of machine learning techniques that spanned supervised learning methods, including Random Forest and AdaBoost, as well as unsupervised techniques such as Hierarchical and K-means Clustering. Additionally, a multi-label classification framework was implemented using Support Vector Machines (SVM) and Naive Bayes. The SVM model attained an accuracy of 79.02%, signifying its relative competence for the classification challenge at hand. Moreover, the Naive Bayes model recorded a comparable accuracy of 79.01%, a testament to its efficacy in managing substantial dataset and in Unsupervised models, Hierarchical clustering demonstrated superior performance, exhibiting a significantly lower Sum of Squared Errors (SSE) compared to the K-means model.

Future Research Directions

Future research in research paper classification should focus on incorporating advanced natural language processing algorithms to deepen semantic comprehension, the development of systems with robust capabilities for cross-disciplinary and multilingual classification, and the implementation of real-time, adaptive models that mitigate algorithmic biases while optimizing computational efficiency, thereby advancing the equitable and efficient distribution of academic knowledge.

The authors would like to thank...

References

1) Chengsheng, T., Bing, X., & Huacheng, L. (2018). The Application of the AdaBoost Algorithm in the Text Classification. 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 1792–1796. https://doi.org/10.1109/IMCEC.2018.8469497

2) Kim, M., & Kim, D. (2022). A Suggestion On The LDA-Based Topic Modeling Technique Based On ElasticSearch For Indexing Academic Research Results. Applied Sciences, 12(6), 3118-. Https://Doi.Org/10.3390/App12063118

3) Kim, S.W., Gil, J.M. (2019). Research paper classification systems based on TF-IDF and LDA schemes. Hum. Cent. Comput. Inf. Sci. 9, 30.

4) Kotouza, M. T., Psomopoulos, F. E., & Mitkas, P. A. (2020). A Dockerized Framework For Hierarchical Frequency-Based Document Clustering On Cloud Computing Infrastructures. JOURNAL OF CLOUD COMPUTING-ADVANCES SYSTEMS AND APPLICATIONS, 9(1), 1–17. Https://Doi.Org/10.1186/S13677-019-0150-Y

5) Saigal, P., & Khanna, V. (2020). Multi-category news classification using Support Vector Machine based classifiers. SN Applied Sciences, 2(3), 458-. https://doi.org/10.1007/s42452-020-2266-

6) Shao, D., Li, C., Huang, C., Xiang, Y., & Yu, Z. (2022). A news classification applied with new text representation based on the improved LDA. Multimedia Tools and Applications, 81(15), 21521–21545. https://doi.org/10.1007/s11042-022-12713-6

7) Singh, P., Piryani, R., Singh, V. K., & Pinto, D. (2020). Revisiting Subject Classification In Academic Databases: A Comparison Of The Classification Accuracy Of Web Of Science, Scopus Dimensions. Journal Of Intelligent & Fuzzy Systems, 39(2), 2471–2476.

8) Trstenjak, B., Mikac, S., & Donko, D. (2014). KNN with TF-IDF based framework for text categorization.

9) Wu, X., Gao, Y., & Jiao, D. (2019). Multi-Label classification based on random forest algorithm for non-intrusive load monitoring system.

10) Xiang, L. (2022). Application of an improved TF-IDF method in literary text classification. Advances in Multimedia. 1–10.https://doi.org/10.1155/2022/9285324

11) Zhang, L., Sun, B., Shu, F., & Huang, Y. (2022). Comparing Paper Level Classifications Across Different Methods And Systems: An Investigation Of Nature Publications. Scientometrics, 127(12), 7633–7651. Https://Doi.Org/10.1007/S11192-022-04352-3

12) Zhang, W., Yoshida, T., & Tang, X. (2008). Text classification based on multi-word with support vector machine. Knowledge-Based Systems, 21(8), 879–886. https://doi.org/10.1016/j.knosys.2008.03.044