Research Paper Classification: A Neural Network Approach with Feature Selection and TF-IDF

JENIYA SULTANA and SU JUNG CHOI*, Missouri State University, USA

With the advancement of computer and information technology, numerous research papers are being published every day. As new research fields are continuously created, categorizing research papers into traditional categories does not accurately reflect the categories anymore. As a result, improper or ambiguous categories lead users to have a hard time finding the research papers they are looking for. Moreover, another problem lies in the research topics themselves when the topics do not exactly fall into a single category. Many papers cover multidisciplinary topics, making it harder to classify without considering all of those categories.

In order to overcome the challenges with research paper classification, it is key to accurately classify research papers according to their content. It will increase user convenience and provide improvement in the fields of research. Hence, in this project, we propose to build a research paper classification system. Generally, there are two possible approaches to classifying research papers: one is content-based classification, and the other one is metadata-based classification. Previous works show that content-based classification yields better results than metadata-based classification. However, since not all research papers are publicly available, the content of the papers is not always accessible. On the other hand, metadata such as author names, titles, abstracts, keywords, etc., are publicly available. Hence, to utilize more data available online, a metadata-based approach is more feasible. In this project, we focused on the title and abstract of the papers. By utilizing the feature selection technique, we reduced the number of features, which led to efficient training. Then, we produced TF-IDF vectors for text representation and used a neural network model for the classifier. The model achieved a classification accuracy of 77.78% and a precision of 81% on the test set. Such results leave room for improvement in future work, which underscores the importance of continued exploration and refinement using different classification methodologies as well as incorporating advanced text representation techniques.

1 RELATED WORK

As technology continues to advance, the number of research studies being conducted in the fields of computer and information technology is also increasing. In order to make it easier to retrieve relevant information and make sense of the vast amount of data available, researchers are working to categorize related research works. The goal is to create a system that can accurately classify research works based on their content and subject matter, making it easier for researchers to find the necessary information.

In [7], authors proposed a research paper classification system that utilizes TF-IDF (Term frequency-inverse document frequency) and LDA (Latent Dirichlet allocation) scheme. In this work, they used only the abstract data to classify the research paper. The reason behind this is

Authors' address: Jeniya Sultana, jeniya1378@missouristate.edu; Su Jung Choi, sujung123@live.missouristate.edu, Missouri State University, Springfield, Missouri, USA.

that the abstract is one of the most essential parts that describes the gist of the paper. Generally, users tend to read the abstract first to identify the research direction before reading the whole paper. As a result, the core concept is concisely written in an abstract. Hence, the authors proposed this approach by considering only abstract data. The authors constructed keyword sets of the top 10, 20, and 30 keywords in this approach. They also identify topics using LDA. Three kinds of topic sets are extracted, consisting of 10, 20, and 30 topics, respectively. Later, they calculated TF-IDF to find out what keywords are important in each paper. Finally, they used k-means clustering algorithm to classify the texts into similar clusters. In order to determine cluster number, this paper utilized elbow scheme. They combined the top frequent keywords and topics extracted from LDA.

Not only for online research works but efforts have been made to classify physical paper-based research studies as well. In [4], authors have focused on the importance of text classification in organizing research papers, both online and offline. It proposes a classification model utilizing Optical Character Recognition (OCR) to digitize physical copies of papers, followed by supervised term weighting schemes and machine learning algorithms for classification. Data is gathered from local schools and online sources, and preprocessing techniques, including OCR and text labeling, are applied. Supervised term weighting schemes are used to assign weights to terms, and machine learning algorithms like Naive Bayes and Support Vector Classifier (SVC) are employed for classification. Results indicate that the model trained with SVC performed consistently well.

However, for any classification task, the way data is represented as numerical vectors plays a crucial role. In NLP (Natural Language Processing), common techniques such as BoW (Bag of Words) and TF-IDF are used. These techniques learn the representation from the documents rather than relying on pre-trained knowledge. In [5], authors explored the efficacy of pre-trained models such as Word2Vec in representing data into numerical vectors. This work presents a research paper classification system based on the combination of Word2Vec, network modeling, and community discovery algorithms. They utilized about 3.2 million paper abstracts in total for training the Word2Vec model. Approximately 1.7 million paper abstracts were obtained from the ArXiv dataset, and 1.5 million were from the AMiner dataset. 250 papers were selected for testing, comprising 50 papers from each of five different fields. To find similarities between papers, they chose the titles and abstracts of the papers, as these should give a general idea of the paper. After preprocessing the data, they deployed the Skip-Gram model within Word2Vec for training. In Skip-Gram, the input is the target word, and the model predicts the words surrounding the target word as output. Then, they modeled the network between papers based on cosine similarity and linked papers with a similarity score equal to or greater than the set threshold value, which is 0.7. The community discovery algorithm is commonly used to find communities in social networks. A community indicates a group of nodes that have many links connected to nodes within their group and have fewer links connected to nodes outside their group. In this study, a community indicates papers that cover a similar topic. They adopted the Girvan-Newman algorithm, which is a well-known algorithm

for community discovery. The algorithm finds communities by removing edges with high edge betweenness values from the network. They used common evaluation metrics, including accuracy, precision, recall, and F-measure. The results show that the proposed classification system achieved an average F-measure score of 75% across the five categories.

Other than the representation of text, research has been conducted largely for the classification method. The work in [2] aims to automatically classify research papers into three main fields: science, business, and social science, using text classification and supervised machine learning techniques. The study creates a balanced dataset of abstracts from these fields and employs four classification techniques: SVM (Support Vector Machines), Naïve Bayes, KNN (K-Nearest Neighbor), and Decision Tree algorithms. Text preprocessing techniques such as tokenization, stemming, and removing stop words are applied, and two vectorization methods, TF-IDF and BoW are used for text representation. The results of the classifiers are compared, considering both vectorization methods. The TF-IDF method is preferred over the count vectorizer (which is a part of BoW), as it assigns more importance to uncommon words, preserving semantic information. Results show that SVM achieves better performance with TF-IDF, with an F1 score of 89% and an accuracy of 88%.

In [6], authors propose a model named DANN (Deep Attentive Neural Network) for classifying scholarly papers into subject categories based solely on their abstracts. This approach is crucial for bibliometric studies, domain knowledge extraction, and enhancing digital library search engines. The model is trained on nine million abstracts from the WoS (Web of Science) using a schema covering 104 subject categories. The DANN architecture consists of two bidirectional recurrent neural networks followed by an attention layer. It outperforms baselines by achieving a micro-F1 measure of 0.76, with individual subject category F1 scores ranging from 0.50 to 0.95. The study emphasizes the importance of retraining word embedding models to maximize vocabulary overlap and highlights the effectiveness of the attention mechanism. Comparisons between different architectures and text representations reveal that combining word vectors with TF-IDF yields superior results compared to character and sentence-level embedding models. The study addresses challenges such as imbalanced samples and overlapping categories, suggesting potential mitigation strategies. However, this study acknowledges limitations such as relying on journal-level rather than article-level classifications in WoS and the potential for changes in WoS's classification schema over time. They also note that their BoW model does not capture sequential information effectively, especially after removing stop words.

The study in [12] addresses the challenge of classifying academic papers into specific fields of knowledge accurately. It utilizes data from various paper elements such as abstracts, titles, keywords, and additional expert input to identify existing subfields and journals exclusive to each subfield. By extracting frequent terms specific to each subfield, the study aims to optimize classification procedures, thereby facilitating more accurate assignment of papers to their respective subfields. First of all, it gathers metadata of papers published in journals related to Physics, including those not categorized by popular databases like Web of Science (WoS).

This dataset serves as the basis for classification. Then, it identifies exclusive journals. This involves a procedure based on WoS categories and expert consultation to confirm exclusive journals. Then, the system extracts labels and calculates weights based on their frequency and specificity to each subfield. The core concept in this study is the SCF (Subfield Compatibility Factor). The system calculates the compatibility of each paper with each subfield using this SCF metric. The SCF is determined by the dot product of label weights and paper occurrences, with coefficients optimized through an iterative process. Finally, in the classification step, the system assigns each paper to the subfield with the highest SCF value. This process ensures papers are classified based on their content rather than predefined database categories.

In [11], the authors pointed out that the existing methods 1) do not employ a semantic model for text representation, 2) rely on the frequency of terms rather than the context and semantics, and 3) rely on domain experts or experimenting with arbitrary values for similarity threshold values when it comes to multi-label classification. So, they proposed the Word2Vec model to address the limitations of previous approaches. Also, in terms of finding threshold values, they proposed the method of conducting a rigorous analysis of the dataset. The datasets from two different journals were used—1,460 research publications from the J.UCS (Journal of Universal Computer Science) and 86,116 research publications from the ACM (Association for Computing Machinery). The proposed system used cosine similarity to compute the similarity between the test document and the average similarity score of individual categories, and then, based on the highest score, the document was categorized into that class. They utilized features based on publicly available metadata of titles, keywords, and general terms, and they evaluated the system's performance for every possible combination of these features. The results showed that in single-label classification (SLC), 'title' metadata had a higher average accuracy, while for multi-label classification (MLC), 'keywords' metadata had a higher average accuracy. But in both cases, the performance of double metadata (i.e., the combination of the title and keywords) was superior to the single feature. Moreover, the average accuracy of the proposed approach for SLC and MLC was 0.86 and 0.81, respectively, which outperformed two state-of-the-art techniques.

As traditional single-label classification is not suitable for papers labeled with multiple classes, the authors in [9] proposed a multi-label K-Nearest Neighbor (ML-KNN) algorithm for the multi-label classification of research papers. Unlike traditional KNN classification, ML-KNN uses the maximum a posteriori (MAP) principle to decide whether the sample belongs to a class label or not, and it applies the Bayesian formula to calculate the posterior probability by considering the same cases in the entire dataset. They utilized the dataset from Kaggle, which we also use in this paper. They selected the first 10,000 papers from the dataset and preprocessed the data using lemmatization and TF-IDF. Five evaluation metrics were utilized, including 1) hamming loss, 2) one-error, 3) coverage, 4) ranking loss, and 5) average precision. Except for average precision, all the other metrics indicate better performance when the value is close to 0 (i.e., the smaller the value, the better the performance). The authors consider hamming loss and average precision to be relatively more important indicators than the rest.

Moreover, both Euclidean distance and cosine similarity were compared, and in all cases, Euclidean distance gives better results. The experiments show that the ML-KNN algorithm performs better than the traditional multi-label learning algorithms—Decision Tree, Rank-SVM, and classical KNN—in all five evaluation criteria, which proves the effectiveness of ML-KNN.

In addition to machine learning and deep learning techniques, our literature review also found transformer-based implementation for research paper classification. In [13], the authors proposed a text classification method using a BERT-based graph convolutional neural network (BERT-GCN). To fine-tune the original BERT model, they incorporated three techniques: span masking, learning rate attenuation, and data augmentation. These techniques enhance the semantic representation of text, accelerate the convergence speed of the model to avoid overfitting, and expand training samples to improve the model's generalization ability. Additionally, GCN is employed to generate a similarity graph based on token representations derived from scientific paper titles. During the classification phase, GCN utilizes nodes to represent words within sentences, with node features comprising token embedding vectors. Through multiple GCN layers, node feature vectors are iteratively updated via message passing and aggregation from neighboring nodes, effectively capturing the structural features from the paper titles. For the experiment, they selected 10,000 papers from CLS, a Chinese scientific literature dataset released by the Alibaba Group, and 34,672 papers from arXiv-2019. The datasets have 13 primary subject categories. They compared the proposed model with four baseline algorithms—TextCNN, TextRNN, DPCNN, and ERNIE. The chosen evaluation metrics were Macro Precision, Macro Recall, and Macro F1. These were computed by calculating the Precision, Recall, and F1 for each class and averaging them. Among the four baseline algorithms, ERNIE performed the best, but BERT-GCN outperformed ERNIE by 3.04% and 5.08% for macro precision rate in the CLS dataset and arXiv-2019 dataset, respectively. Overall, the results show that the three techniques improved the performance of the BERT algorithm significantly. Moreover, the BERT-GCN model achieved 88.32% and 91.09% accuracy for the CLS and arXiv-2019 datasets, respectively, which were further improved by 2.09% and 2.51% from the fine-tuned BERT.

In [8], the authors proposed a metadata-based multi-label classification solution for accurately classifying research articles. They utilized the publicly available arXiv dataset, which contained about 2.3 million records. To streamline the data and address imbalance issues, they narrowed down the dataset to papers from the computer science domain published within the last ten years (since 2012), resulting in a subset of 63,209 records with six output labels, each record labeled with 1-4 categories. The authors pointed out the limitations of RNN and LSTM models: 1) ineffective handling of short- and long-term dependencies and 2) a sequential workflow that impedes parallelization and increases processing time. The transformer model was introduced in this paper to address these issues. Specifically, they employed the Specter method, built on Transformer architecture and pre-trained on SciBERT embeddings—a large corpus of scientific datasets. Specter operates at the document level, employs a bidirectional encoder transformer model with multi-head attention mechanisms, and utilizes a triplet learning objective to

capture semantic and contextual relationships between input documents. As a result, the Specter model achieved robust performance metrics on the test dataset, including precision of 91.06%, accuracy of 84.67%, recall of 89.52%, an F1-score of 90.28%, and a hamming loss of 0.0349. These results showed that the proposed classification model outperformed all other models, including RNN, LSTM, bert-base-uncased, SciBERT, and all-MiniLM-L6-v2.

2 METHODS

Our proposed framework for research paper classification is shown in figure 1.

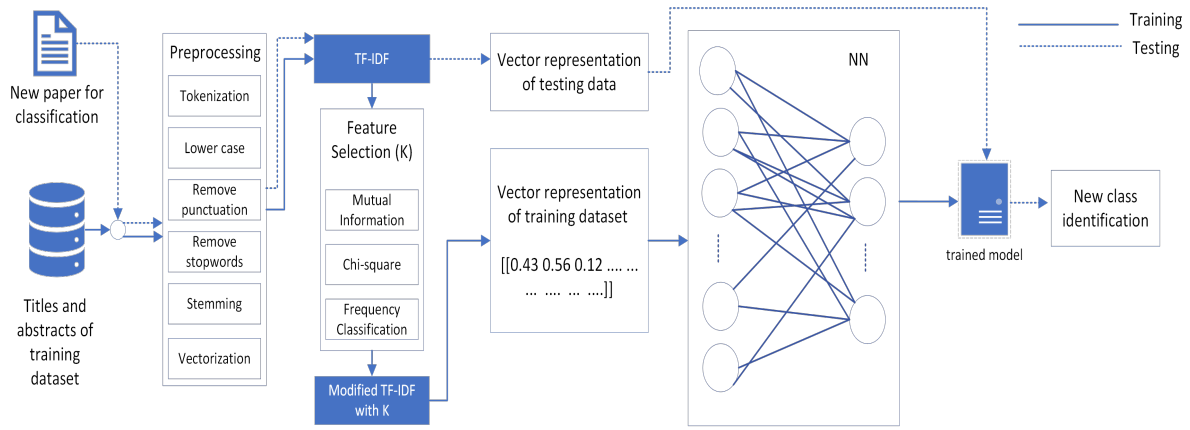

Fig. 1. Proposed framework for research paper classification

2.1 Dataset Description

A multi-label classification dataset obtained from [1] was utilized in this project. The distribution of content in the original dataset is shown in figure 2. It comprises 20,972 data entries and includes fields such as ID, Title, Abstract, and six labels representing six different categories: Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, and Quantitative Finance. Papers within the dataset may belong to multiple categories, and if the paper belongs to a category, it is represented by '1' for that label and '0' otherwise.

In this project, since we decided to focus on multi-class classification, the dataset is filtered to have only the papers that belong to a single category. Then, to address data imbalance while maintaining enough data, we dropped the two categories that have the least amount of papers. Thus, we sampled 6,544 papers in total, containing 1,636 papers per category in four categories: Computer Science, Physics, Mathematics, and Statistics. The sampled dataset we use for the project is shown in the figure 3.

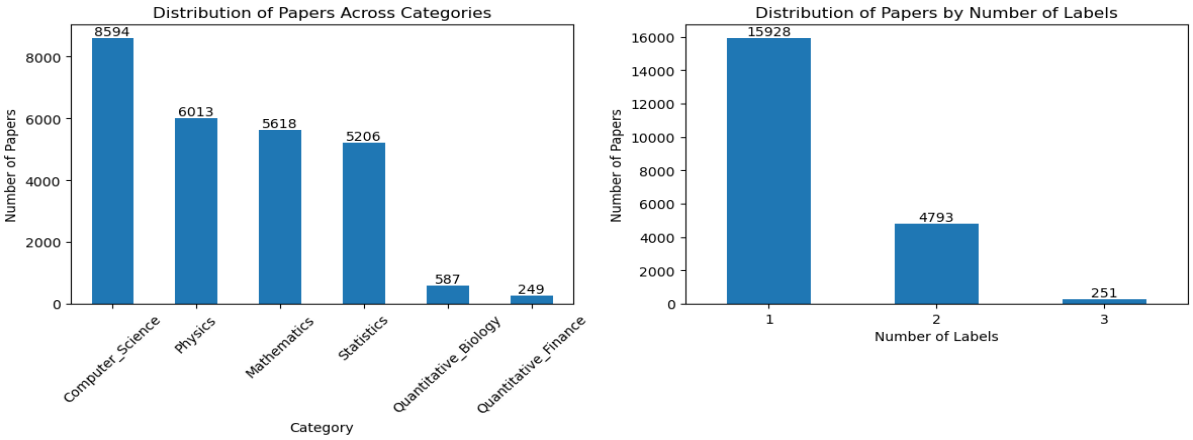

Fig. 2. Original Data Distribution

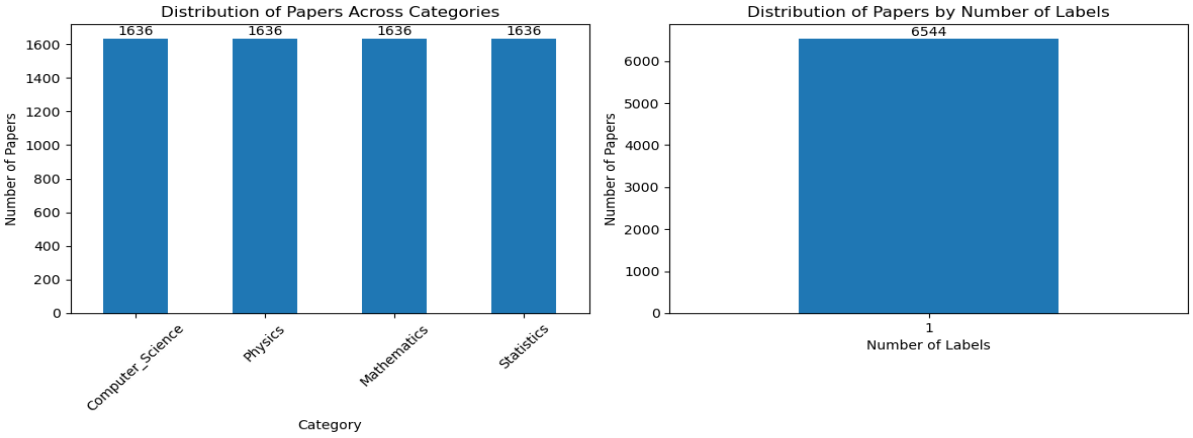

Fig. 3. Sampled Data Distribution

2.2 Data Preprocessing

In order for the dataset to be used in training the model, several data preprocessing steps are required. Each title and abstract undergo tokenization and conversion to lowercase. Subsequently, any punctuation present is removed, and common stop words such as ‘the’, ‘be’, and ‘of’ are eliminated, as they do not contribute significantly to the context. Additionally, stemming is applied to obtain the base form of each token. Finally, the TF-IDF method was employed to transform the words into vectors.

2.3 Data Representation using TF-IDF

In order to analyze the underlying patterns in the dataset and train a classifier, it is necessary to convert textual data into numerical vectors. For this purpose, we utilize TF-IDF representation. TF-IDF is a classical and widely used method in natural language processing (NLP) for text representation. Term frequency (TF) [3] is defined as the number of times a term appears in the document. It is used to give weight to the term based on its frequency in the document. However, term frequency does not consider actual relevance since it treats every term equally important, which is often not the case. Therefore, to overcome this limitation, inverse document frequency (IDF) was introduced. IDF can be calculated as $\log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$, where df_t is the document frequency, representing the number of documents in the collection in which a term t appears, and N is the total number of documents in the collection. Then, TF-IDF is calculated by multiplying the TF of a term by its IDF. This will reduce the TF weight of a term that simply has a high frequency in the collection. We extracted the labels from the dataset, prepared the TF-IDF vector representation, and saved them as numpy array files for future use.

2.4 Model Building: Neural Network

For the classification, we aimed to identify important phrases and relationships between words. In order to accomplish this, based on our literature review, we found deep neural networks to be best suited for this project. However, when it comes to selecting the most appropriate neural network architecture, the decision is heavily influenced by the task and domain at hand, as well as the linguistic structure of the text. For instance, as per the authors in [10], feed-forward neural networks perceive input text as a BoW or TF-IDF, RNN (Recurrent Neural Network) models are effective at capturing word sequences, CNN (Convolutional Neural Network) models are capable of recognizing patterns, LSTMs (Long Short Term Memory neural network) can handle sentences of varying lengths and suitable for analyzing longer texts, attention mechanisms can detect associated words, Siamese networks perform well for text-matching, and graph neural networks are suitable for the graph structure of a text. For our project, the most important feature is considered as the keywords and associated frequency that are present in the title and abstract, not their location in the sentence. Hence, it suggests that a neural network, especially a fully connected neural network, would be a suitable model to be used with TF-IDF representation.

In this project, we built a feedforward neural network as a sequence of layers, starting with an input layer, followed by three hidden layers, and ending with an output layer. Each layer contains neurons, which perform computations on the input data and pass the results to the next layer. As the neural network is feedforward in nature, information flows in one direction, from input to output. At the output of each layer, we applied activation functions. These functions introduce non-linearity to the model, enabling it to learn complex patterns in the data. We used the ReLU (Rectified Linear Unit) activation function at the output of each hidden layer and the softmax activation function at the final output layer to produce

probability distributions over the classes. During training, we applied dropout regularization and early stopping techniques to prevent overfitting. The dropout randomly deactivates a fraction of neurons during training, forcing the network to learn more robust features and reducing its reliance on specific neurons. On the other hand, by early stopping, training is halted when the model does not improve significantly after a specific number of epochs. We observed the validation loss with a patience of 7 so that if the performance does not improve significantly for seven consecutive epochs, then the training process will stop. In addition to accuracy, we incorporated other evaluation metrics such as precision, recall, and the F1 score to assess the model's performance comprehensively.

2.5 Feature Selection

Since we have 27,490 unique terms in the dataset, not all of the terms add values in determining their category. Having too many features may decrease the performance of the classifier by adding noise. Feature selection becomes handy in such cases. Feature selection is the process of selecting a smaller set of terms that are significant in the training dataset and using only those to further train the classifier. It helps to reduce the dimensionality of the dataset. However, finding the optimal K that gives the best performance requires experimenting with different K values. We need to consider the balance between efficiency and performance. Thus, we deployed three utility measures–mutual information, chi-square test, and ANOVA F-value–that are commonly used for feature selection. To experiment with different k values, we defined the range of values for k to be evenly spaced 20 values from 100 to 27,000.

Mutual information quantifies the degree to which the presence or absence of a term contributes to the accurate classification of the class label. On the other hand, x^2 (Chi-squared) is used to evaluate the independence between a feature and the target variable in a dataset. Lastly, ANOVA F-value is an analysis of variance between the feature values and the class labels. All three methods help to find the significant features that are most relevant to a specific class. Figure 4 shows the performance of three methods based on the F1 score. Based on this result, we found that when k is around 10,000, it returns the best F1 scores, which is about 80% for ANOVA F-Value. Mutual Information shows the worst performance among the three methods. So, we chose the optimal K to be 10,000 and opted for Chi-Squared because the difference between the results using ANOVA F-value and Chi-Squared was insignificant during our experiment.

2.6 Model Training and Testing

After finding the optimal k, in order to train and test the model, the dataset was split into training data and testing data with a ratio of 80:20. By selecting only 10,000 features based on the chi-squared, the selector object was created and was fitted to training data. We set the Neural Network's parameters for learning rate as 0.0005, batch size as 32, and number of epochs as 30. Also, the early stopping callback was defined with the validation loss to

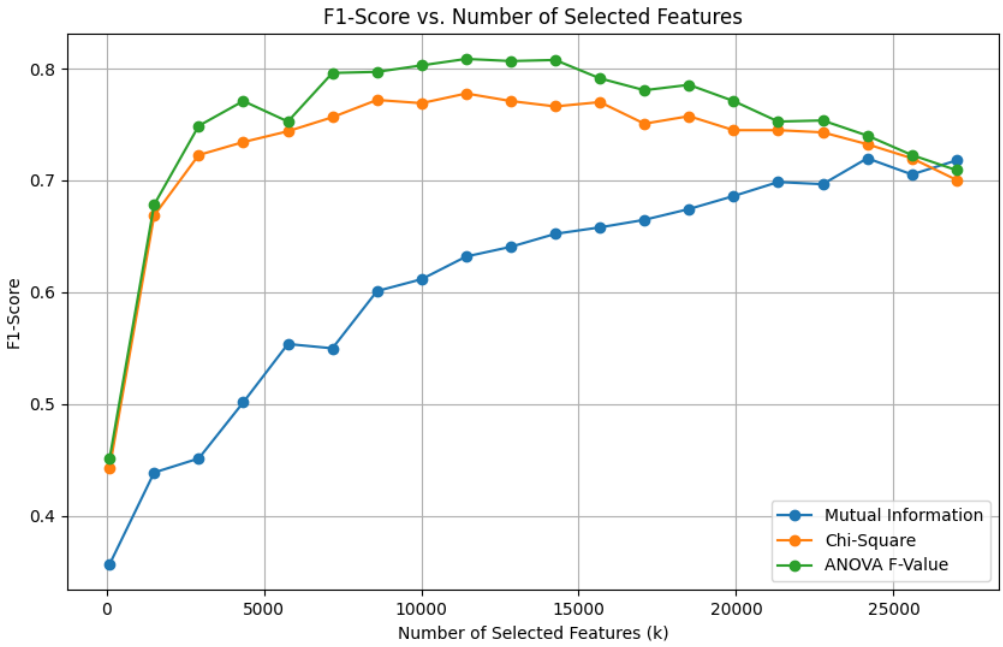

Fig. 4. Feature Selection

be monitored so that the training can stop when the validation loss does not improve for 7 consecutive epochs. When the model was fitted with the training set, it also split again into training and validation sets with a ratio of 80:20 so that we could observe the performance for both training and validation sets. This helps us to monitor the performance of the model on unseen data and its generalization ability during the training process. In the figure 5, it presents the architecture of the model. Regularization is applied to the weights of each layer using L2 regularization with a regularization parameter set to 0.001. To prevent overfitting during training, a dropout rate of 0.5 is applied after each hidden layer. Dropout randomly sets a fraction of input units to zero. We used four Dense layers; the ReLU activation function was used for the first three layers, and the last layer used the softmax activation function. Then the model was compiled with the loss function of 'categorical_crossentropy', which is suitable for multi-class classification, and Adam optimizer with a learning rate of 0.0005.

2.7 Performance Metrics

To evaluate the performance of our model, we considered the following metrics: accuracy, F1, and loss.

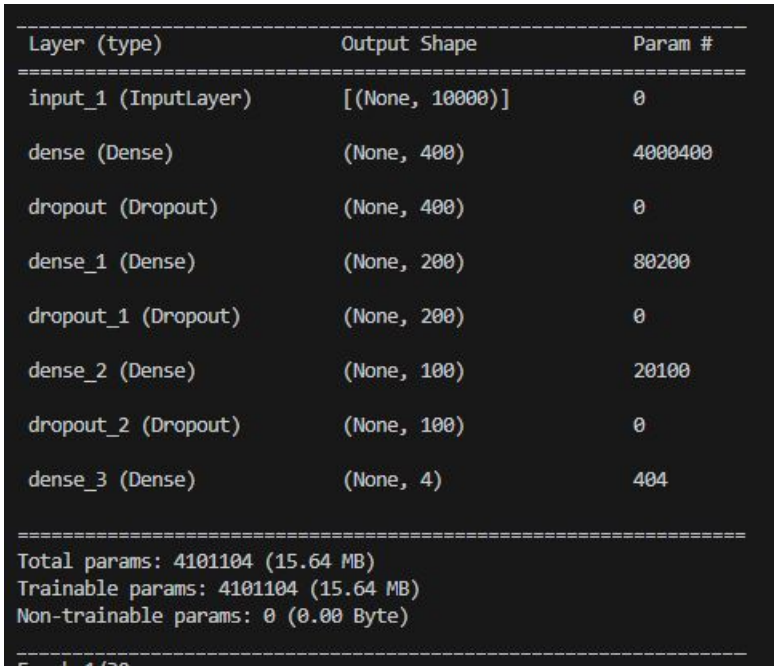

Fig. 5. Neural Network Model Summary

- **Accuracy**: Accuracy measures the proportion of correctly classified instances among the total instances. It is calculated as the ratio of the number of correct predictions to the total number of predictions.
- **F1 Score**: The F1 score is calculated using precision and recall.
  - **Precision**: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive predictions to the sum of true positive and false positive predictions. In other words, precision indicates the proportion of correctly predicted positive instances among all instances predicted as positive.
  - **Recall**: Recall, also known as sensitivity or true positive rate, measures the ability of the model to identify all positive instances. It is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions. Recall indicates the proportion of correctly predicted positive instances among all actual positive instances.

  The F1 score provides a balance between precision and recall and is particularly useful when the classes are imbalanced. It is calculated as

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

- **Loss**: Loss, also known as the cost function, measures the discrepancy between the predicted output and the actual target values. It quantifies how well the model is performing during training by penalizing incorrect predictions.

3 EXPERIMENT AND RESULTS ANALYSIS

During the training phase, we saved the history of the model's performance for both the training dataset and the validation dataset.

Figures 6, 7, and 8 present the performance of the model per each epoch based on accuracy, F1, and loss, respectively. We noticed that the model performs well on the training set while not generalizing well, resulting in lower performance on the validation set. For both Accuracy and F1 scores, while it reaches close to 100% on the training set, it remains below 80% on the validation set.

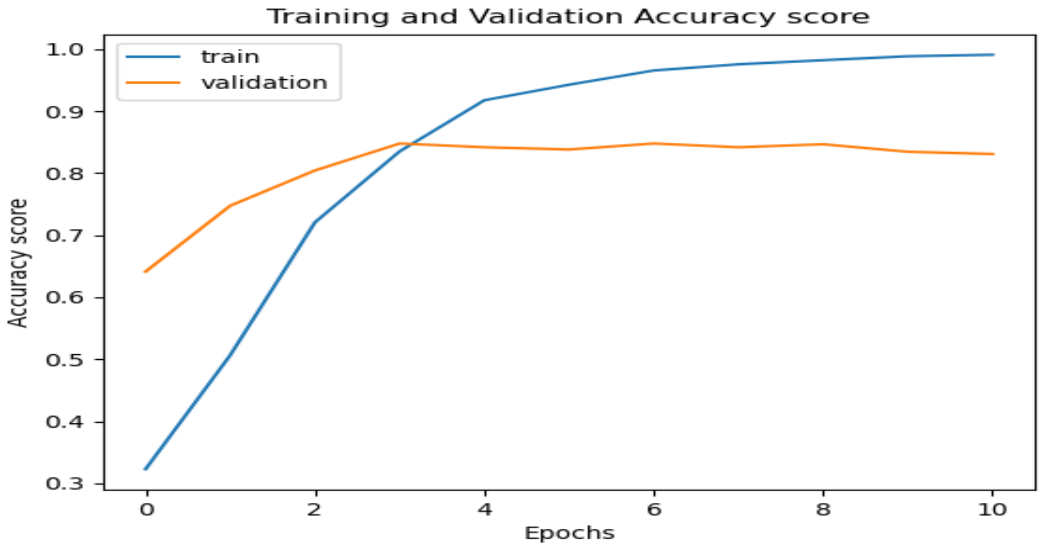

Fig. 6. Training and validation Accuracy Performance

After the training, we evaluate the performance of the model on a new, unseen dataset that we set aside earlier for testing purposes. Figure 9b shows the performance results on the test set when using all features, whereas Figure 9a presents the performance when using selected features based on the value of k. While there are subtle differences between the results, both are fairly similar. This suggests that selecting k features effectively maintains performance while being computationally less expensive to run. The performance with k features includes an accuracy of 77.78%, a precision of 81%, and an error of 1.174.

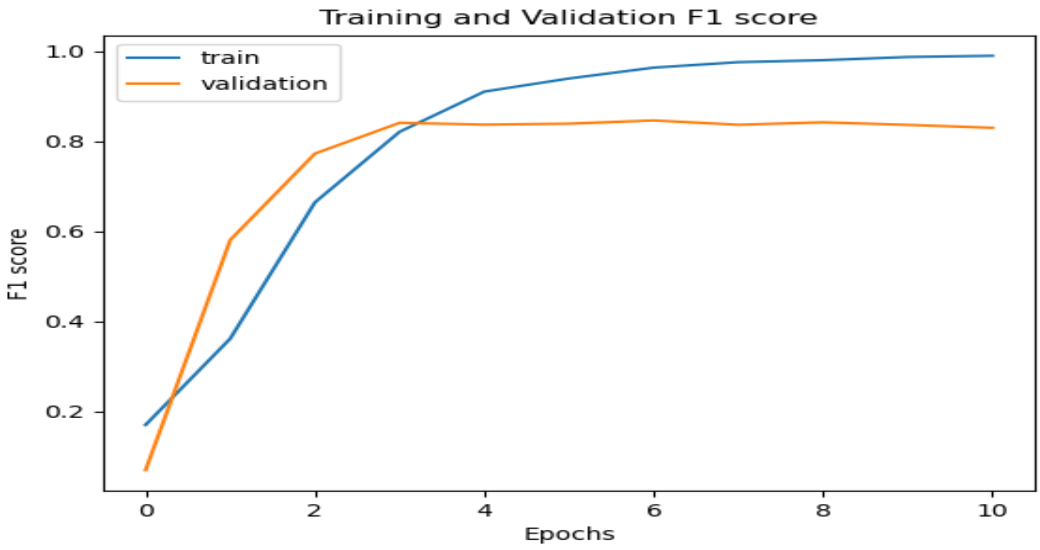

Fig. 7. Training and validation F1 Performance

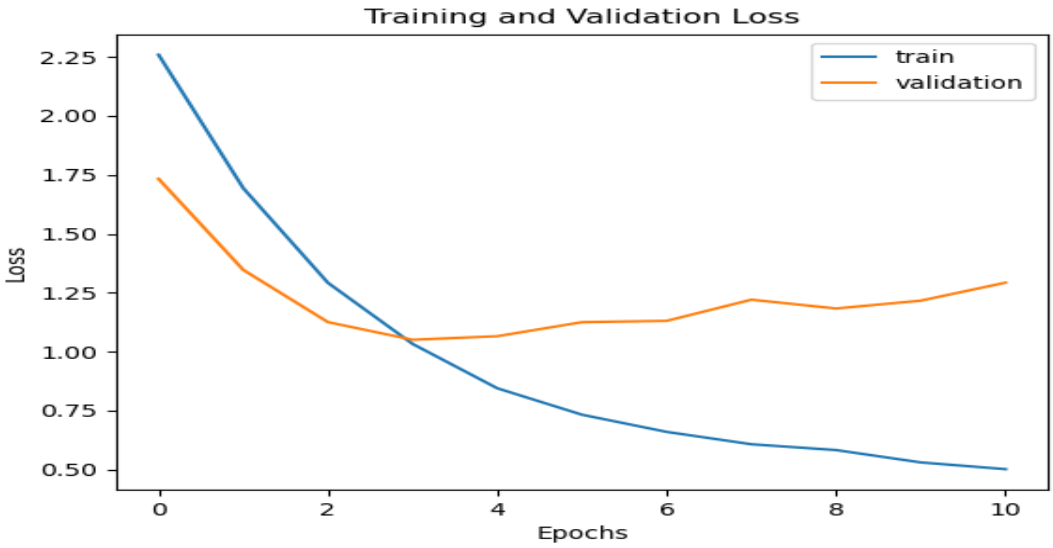

Fig. 8. Training and validation Loss Performance

```
Error on test set: 1.174
Accuracy on test set: 77.778
Precision on test set: 81.003
```

(a) Performance Results on Test Set with selected features

```
Error on test set: 1.214
Accuracy on test set: 77.971
Precision on test set: 80.619
```

(b) Performance Results on Test Set with all features

Fig. 9. Loss and Accuracy on Testing Dataset

The figure 10 displays a confusion matrix, with the x-axis representing predicted labels and the y-axis representing true labels. It illustrates that the majority of predictions align accurately, as highlighted along the diagonal axis. However, certain categories were not as well-captured; notably, the top right and bottom right sections show the highest number of incorrect predictions. These two subject fields were 'Computer Science' and 'Statistics'. This result led us to reasonably infer that the issue stems from overlapping concepts, which often blur the distinction between these categories.

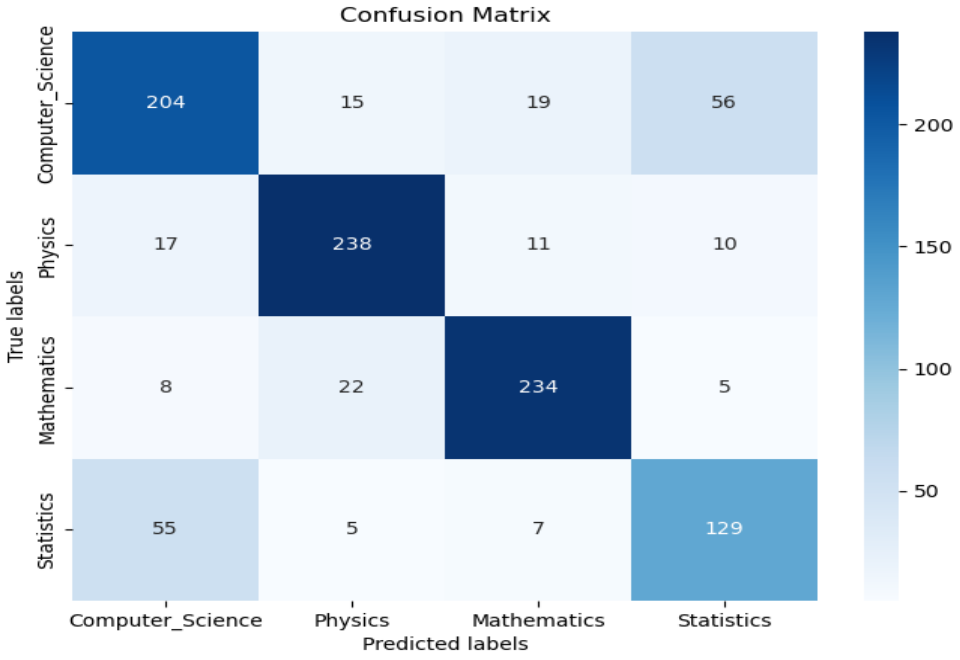

Fig. 10. Confusion Matrix for true labels vs. predicted labels

4 CONCLUSION

In this project, we proposed a research paper classification method using TF-IDF representation and a Neural Network model. By leveraging the title and abstract of each paper, we calculated the TF-IDF score for each document and trained the model to classify documents into appropriate categories. We considered these two metadata as they are publicly available for most research papers and contain important keywords representing the main idea of the paper.

To enhance the efficiency of the training process, we implemented feature selection. By analyzing the performance of mutual information, the chi-square test, and the ANOVA F-value graph, we were able to select 10,000 terms, approximately one-third of the dataset, that achieved optimal results in F1 scores. This reduction in features not only maintained performance but also significantly improved efficiency, as demonstrated by the streamlined training process. Our model achieved an accuracy of 77.78% on the test set.

While our feature selection process optimized the model's efficiency without compromising performance, it also showed potential limitations in accurately capturing category boundaries. Several factors may contribute to the model's performance remaining in the moderate range. Thus, we have realized the benefit of further investigation into the most frequently occurring terms in each class. Considering that 1) the four subject fields—Computer Science, Statistics, Physics, and Mathematics—share many similar vocabularies such as "data analysis," "model," "probability," "experiment," etc., and 2) we utilized only the title and abstract, it is possible that the vector representation may contain highly scored terms across multiple categories. This situation may lead the model to be unable to precisely capture the boundaries between different categories. Particularly, as indicated in figure 4, mutual information kept increasing as the number of features increased. This suggests that each term may not strongly contribute to each category.

Therefore, as an extension to this work, we plan to employ a heuristic method to prioritize terms that are more indicative of specific categories. By observing the distribution of unique keywords within each category and assigning greater weight to those most representative of each class, we expect to improve the model's ability to distinguish between closely related subjects. Additionally, we intend to incorporate other text representation techniques, such as word2vec and GloVe, to compare performance. We also aim to capture the global context of the data and identify the semantic relationships of words, potentially resolving the issue of shared domain keywords. By pursuing these avenues, we expect to enhance the model's performance in future work.

REFERENCES

[1] Accessed: April 22, 2024. Research Paper Classification Dataset. https://www.kaggle.com/datasets/shivanandmn/multilabel-classification-dataset.

[2] Shovan Chowdhury and Marco P Schoen. 2020. Research paper classification using supervised machine learning techniques. In *2020 Intermountain Engineering, Technology and Computing (IETC)*. IEEE, 1–6.

[3] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*. http://nlp.stanford.edu/IR-book/ ISBN: 0521865719.

[4] Rhey Marc A Depositario, Glenn Geo T Noangay, Julian Michael F Melchor, Cristopher C Abalorio, and James Cloyd M Bustillo. 2023. Automated Categorization of Research Papers with MONO Supervised Term Weighting in RECApp. *International Journal of Advanced Computer Science and Applications* 14, 2 (2023).

[5] Esra Gündoğan and Mehmet Kaya. 2020. Research paper classification based on Word2vec and community discovery. In *2020 International Conference on Decision Aid Sciences and Application (DASA)*. 1032–1036. https://doi.org/10.1109/DASA51403.2020.9317101

[6] Bharath Kandimalla, Shaurya Rohatgi, Jian Wu, and C Lee Giles. 2021. Large scale subject category classification of scholarly papers with Deep Attentive Neural Networks. *Frontiers in research metrics and analytics* 5 (2021), 600382.

[7] Sang-Woon Kim and Joon-Min Gil. 2019. Research paper classification systems based on TF-IDF and LDA schemes. *Human-centric Computing and Information Sciences* 9 (2019), 1–21.

[8] Kushal Lahoti, Vrinda Ahuja, and Archana Patil. 2023. SPECTER-Based Transformer Model For Multi-Label Research Paper Classification. In *2023 IEEE Pune Section International Conference (PuneCon)*. 1–7. https://doi.org/10.1109/PuneCon58714.2023.10450144

[9] Shurui Li and Jiechen Ou. 2021. Multi-label classification of research papers using multi-label k-nearest neighbour algorithm. In *Journal of Physics: Conference Series*, Vol. 1994. IOP Publishing, 012031.

[10] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *Comput. Surveys* 54, 3 (2021), 1–40.

[11] Ghulam Mustafa, Muhammad Usman, Lisu Yu, Muhammad Afzal, Muhammad Sulaiman, and Abdul Shahid. 2021. Multi-label classification of research articles using Word2Vec and identification of similarity threshold. *Scientific Reports* 11 (11 2021), 21900. https://doi.org/10.1038/s41598-021-01460-7

[12] Gerson Pech, Catarina Delgado, and Silvio Paolo Sorella. 2022. Classifying papers into subfields using Abstracts, Titles, Keywords and KeyWords Plus through pattern detection and optimization procedures: An application in Physics. *Journal of the Association for Information Science and Technology* 73, 11 (2022), 1513–1528.

[13] Xiaohe Zhang, Xinguo Yu, Xiaoqian Liu, and Xiaopan Lyu. 2023. Scientific Paper Classification by Fusing BERT and GCN. In *2023 International Conference on Intelligent Education and Intelligent Research (IEIR)*. 1–6. https://doi.org/10.1109/IEIR59294.2023.10391239