Machine Learning-based News Classification in Big Data Environments

1st Thi-Hien-Hoa Le
FPT University
Ha Noi, Vietnam
hoalth@fpt.edu.vn

2nd Thanh-Dao Duong
Vietnam - Korea College Of Technology In Bacgiang
Bac Ninh, Vietnam
thanhdao92@gmail.com

3rd Thi-Ha Le
Faculty of Information Technology, Hung Yen University of Technology
Hung Yen, Vietnam
lethiha2805.cs@gmail.com

4th Viet-Anh Le
Faculty of Information Technology, Hung Yen University of Technology
Hung Yen, Vietnam
levietanh1202@gmail.com

5th Van-Quyet Nguyen*
Faculty of Information Technology, Hung Yen University of Technology
Hung Yen, Vietnam
quyetict@utehy.edu.vn

* *Corresponding author*

***Abstract*—This paper investigates the application of machine learning algorithms for news topic classification in distributed computing environments. To address the challenges posed by the scale and complexity of online news data, we propose a Spark-based framework that combines TF-IDF vectorization with four popular classifiers: Logistic Regression, Decision Tree, Random Forest, and Linear SVM. We conduct experiments on a real-life Vietnamese news dataset, comprising diverse topical categories and linguistic variations. The evaluation considers both execution performance and predictive accuracy, providing insights into the trade-offs between scalability and classification effectiveness. Experimental results show that Logistic Regression consistently outperforms other algorithms, achieving the highest F1-score of 0.9563 and offering the best balance between accuracy and computational efficiency. In contrast, Random Forest exhibited the highest computation cost and limited scalability. These findings demonstrate the feasibility and effectiveness of deploying classical machine learning models for news classification in big data environments, and provide practical guidance on model selection and cluster configuration for Spark-based learning pipelines.**

Keywords—News Classification, Big Data Analytics, Scalable Text Processing, Text Mining

I. Introduction

News classification plays a pivotal role in the management and organization of information, particularly in the context of the massive volume of news articles published daily on digital platforms. Without clear categorization, the retrieval and navigation of relevant information become increasingly difficult, forcing users to process an overwhelming amount of unstructured content from diverse sources [1]. Effective classification enables readers to quickly access content aligned with their interests and supports the development of advanced information retrieval systems by enhancing search precision based on specific topical categories [2]. The applications of news classification are multifaceted. A notable example is its integration into personalized news recommendation systems, which leverage classification to suggest articles tailored to individual user preferences [3]. Additionally, it plays a significant role in content moderation and misinformation detection, aiding in the identification and suppression of fake news [4].

There are several studies that have explored this domain, particularly in leveraging machine learning algorithms such as Naïve Bayes and Support Vector Machines (SVM) [5], which have demonstrated improved efficiency in text processing tasks. More recently, deep learning approaches have shown superior performance in classifying news articles from large-scale, heterogeneous data sources [6, 7]. Convolutional Neural Networks (CNNs), in particular, have been widely adopted for text classification and natural language processing tasks [6], while transformer-based models such as BERT have significantly enhanced the semantic understanding of text, resulting in improved classification accuracy [7].

However, as the volume and velocity of news data continue to grow exponentially, traditional machine learning approaches face substantial limitations in terms of scalability and processing efficiency. News classification increasingly deals with high-dimensional text features and large-scale datasets, posing computational challenges in both training and inference, especially in real-time applications such as news aggregation and recommendation systems [8]. To address these issues, distributed frameworks like Apache Spark [9] have become popular due to their in-memory computation and scalability. The use of Spark is further motivated by the availability of scalable libraries such as MLlib, which offer distributed implementations of classical learning algorithms (e.g., Logistic Regression, Decision Tree, and SVM) optimized for cluster environments [10].

In this paper, we investigate the application of machine learning algorithms for news classification within a distributed Spark environment, focusing on both execution performance and predictive accuracy under varying system configurations. This dual-perspective evaluation aims to provide practical insights into the trade-offs between scalability and model effectiveness in large-scale text classification tasks. Specifically, we evaluate four widely-used algorithms, namely Logistic Regression, Decision Tree, Random Forest, and Linear SVM, as implemented in the Spark MLlib framework.

The main contributions of this work are as follows: (1) we assess execution efficiency across different Spark cluster configurations; (2) we analyze predictive performance on a real-world Vietnamese news dataset; and (3) we identify an effective Spark configuration that balances system resource

979-8-3315-7790-2/25/$31.00 ©2025 IEEE

utilization and classification quality. These findings offer actionable guidance for deploying scalable and efficient news classification pipelines using Apache Spark.

II. Related Work

News topic classification has been widely studied in natural language processing and information retrieval due to its practical value [1]. Early studies focused on classical supervised models such as Decision Trees, Random Forests, Logistic Regression and Support Vector Machine (SVM) combined with feature extraction techniques like TF-IDF, n-grams, and word embeddings [11, 12]. These approaches have shown robust and interpretable results across different languages and datasets.

With the rapid expansion of online news, scalable computing frameworks like Apache Spark have become essential for handling large-scale classification tasks [13]. The integration of classical algorithms into Spark pipelines, along with advanced feature engineering, has been shown to significantly reduce training time and improve accuracy when working with massive datasets. In particular, Spark distributed architecture enables efficient parallel processing, making it possible to train and evaluate algorithms on data volumes that would otherwise be unmanageable on single machines.

To address the limitations of individual algorithms, recent work has increasingly adopted hybrid and ensemble techniques for news classification, including approaches such as stacking, bagging, boosting and the combination of machine learning with neural networks [14, 15]. These strategies generally enhance scalability and robustness, although some studies have noted that the improvements in accuracy can come at the cost of reduced interpretability and increased computational requirements. Despite these trade-offs, the use of classical algorithms on distributed platforms like Spark remains popular, offering a practical balance between efficiency, transparency, and ease of integration into large-scale news processing systems [16].

Overall, recent advances show that combining established machine learning models with scalable distributed frameworks continues to be an effective strategy for news topic classification, especially in the context of ever-growing news datasets and real-world deployment requirements.

III. News Classification in Big Data Environments

A. Data Collecting and Pre-processing

Data Collecting. To construct a reliable dataset for news classification, we collected articles from three prominent Vietnamese online newspapers: *Dan tri*, *Tien phong*, and *Tuoi tre*. Each article was extracted with four main attributes: Title, Abstract, Content, and Topic Label. The selected categories cover a diverse range of topics, including *Society*, *World*, *Sports*, *Health*, *Entertainment*, *Education*, and *Law*.

The web scraping process was automated to retrieve a substantial number of articles per category. The resulting raw dataset consisted of 123,436 records across 5 columns, encompassing metadata and textual content. The variety of sources ensured heterogeneity in writing style, vocabulary, and topic coverage, which is essential for robust model training and evaluation in a real-world classification setting.

Data Pre-processing. The preprocessing pipeline was designed to handle both the scale and the noisiness of raw web data. The following steps were performed to prepare the dataset for model training:

Step 1: Label Cleaning and Normalization. Topic labels were initially inconsistent due to differences in spelling, formatting, and encoding. We normalized them to a standardized form using regular expressions and mappings (e.g., “the-gioi” to “World”, “phap-luat” to “Law”).

Step 2: Noise Filtering. Non-news entries and incorrectly labeled data were filtered out. Several label anomalies (e.g., “a gil”, “pape luat”) were identified and removed. Duplicate articles were detected and discarded based on content similarity.

Step 3: Text Cleaning and Tokenization. Text fields were cleaned by removing special characters, HTML tags, and irrelevant whitespace. We used User-Defined Functions (UDFs) in PySpark to tokenize the text, split sentences into words, and generate lists of tokens.

Step 4: Stopword Removal. Common stopwords that do not contribute to classification were removed to reduce noise in the feature space.

Step 5: Vectorization. We applied TF-IDF Vectorization on the textual fields (Title, Abstract, and Content) to convert the documents into sparse numerical vectors suitable for machine learning algorithms. This representation captures both word importance and frequency across the corpus.

B. Label Encoding and Feature Engineering

To prepare the dataset for supervised machine learning classification, both the target labels and input features required transformation into numerical formats that can be efficiently processed by learning algorithms.

Label Encoding. The original topic labels in the dataset were represented as text strings including “Education”, “Entertainment”, “Law”, “Health”, “World”, “Sports”, and “Society”. Therefore, we performed normalization to standardize these labels and subsequently mapped each distinct topic to a unique integer, ranging from 0 to 6. This label encoding approach enabled efficient training and evaluation of classification models using integer-based loss functions such as cross-entropy.

Feature Engineering. To represent textual input in a machine-readable format, we applied TF-IDF vectorization on the processed text. Rather than using a single field, we concatenated the Title, Abstract, and Content fields to form a comprehensive input for each sample. The resulting TF-IDF matrix provided a sparse, high-dimensional vector for each article, where each dimension corresponds to a unique term weighted by its importance across the corpus. This vectorized representation served as the input feature matrix for all subsequent classification models, ensuring that the semantic and structural richness of the articles was preserved.

C. Machine Learning-based News Classification with Spark

To perform large-scale training and evaluation of news classification models, we utilized Spark MLlib - a scalable machine learning library built for distributed computing. This approach enabled efficient processing of high-dimensional textual features derived from the TF-IDF vectorization step.

We constructed a machine learning pipeline using the following stages:

Stage 1 - Train and Test Split: The dataset was randomly split into 75% training and 25% testing subsets using *randomSplit()* with a fixed seed to ensure reproducibility.

Stage 2 - Model Selection: Four machine learning classifiers were implemented and compared:

- *Logistic Regression*: Trained with multinomial loss, regParam=0.3, elasticNetParam=0, and maxIter=5.
- *Decision Tree*: Configured with a maximum depth of 20 to avoid overfitting.
- *Random Forest*: Built with 100 decision trees and a maximum depth of 20.
- *Linear Support Vector Machine (SVM):* Trained using LinearSVC with regParam=0.1 and maxIter=10.

Each model was trained using the features column (TF-IDF vectors) as input and the encoded *topic_index* column as the prediction target.

We selected Logistic Regression, Decision Tree, Random Forest, and Linear SVM as representative algorithms due to their diversity in learning paradigms and proven effectiveness in text classification tasks. These models balance linear and non-linear approaches, are well-supported in Spark MLlib, and offer meaningful trade-offs between scalability, interpretability, and performance. Their selection enables focused analysis of execution efficiency and classification accuracy in distributed environments.

Stage 3 - Model Evaluation: We employed the MulticlassClassificationEvaluator to assess model performance using four key metrics:

- *Accuracy:* the overall correctness of the model.
- *Precision:* the proportion of correct positive predictions among all positive predictions.
- *Recall:* the proportion of correct positive predictions among all actual positives.
- *F1-Score:* the harmonic mean of precision and recall, providing a balanced measure of performance.

These metrics were computed using the PySpark evaluation module after applying each model to the test dataset.

IV. Experimental Evaluation

A. Experimental Setup

Environment Settings. Our experiments were conducted on a Spark cluster consisting of one master node and two worker nodes. Each node was equipped with 8 GB of RAM and 8 CPU cores.

Datasets. To evaluate the performance of various machine learning algorithms in classifying news topics, we used a collected dataset containing 123,436 samples across seven predefined categories: *Politics*, *World*, *Education*, *Entertainment*, *Law*, *Health*, and *Sports*. The dataset was stored in 24 separate files to simulate realistic partitioning for distributed data processing.

Prior to training, we preprocessed the data by removing duplicate entries, normalizing topic labels, filtering out irrelevant labels, and applying text cleaning procedures. We employed the TF-IDF vectorizer to transform textual content into numerical feature vectors. We split the dataset into training and testing sets using a 75:25 ratio, yielding 72,837 training samples and 24,599 testing samples.

All experiments were implemented in PySpark and executed in Spark Standalone mode, enabling distributed computation and efficient handling of large-scale text data.

B. Experimental Results

To evaluate the effectiveness of applying machine learning algorithms to the task of news classification in a distributed environment, we conducted experiments from two main perspectives: (1) the execution performance of machine learning algorithms under varying computing configurations of a Spark cluster, and (2) the predictive performance of machine learning models on the news categorization task.

Execution Performance Evaluation. By leveraging its capability for parallel processing across multiple machines and cores, Apache Spark offers robust support for large-scale data preprocessing, model training, and evaluation. Among various configuration parameters, the number of executors in a Spark cluster plays a critical role in determining system performance. To assess the impact of executor parallelism and identify an optimal setting for our infrastructure, we conducted experiments with executor counts ranging from 2 to 12. Each executor was assigned 1 GB of RAM and 1 CPU core. For every machine learning model, we recorded execution times across three stages: data preprocessing, model training, and model testing. Each model was evaluated under all six executor configurations.

We evaluated preprocessing performance by varying the number of Spark executors from 2 to 12 on our dataset. It is important to note that the data preprocessing process was identical across all four algorithms. The results, presented in Fig. 1, represent the average preprocessing time across 24 experimental runs involving the four models. According to the results, increasing the number of executors significantly improved preprocessing efficiency, reducing execution time from 405 seconds with 2 executors to 175 seconds with 12 executors, which corresponds to a 56.8% reduction in execution time. The largest improvements occurred between 2 and 6 executors, with limited gains beyond due to scheduling overhead and core saturation. A configuration of 8 to 10 executors provides an effective balance between performance and resource use.

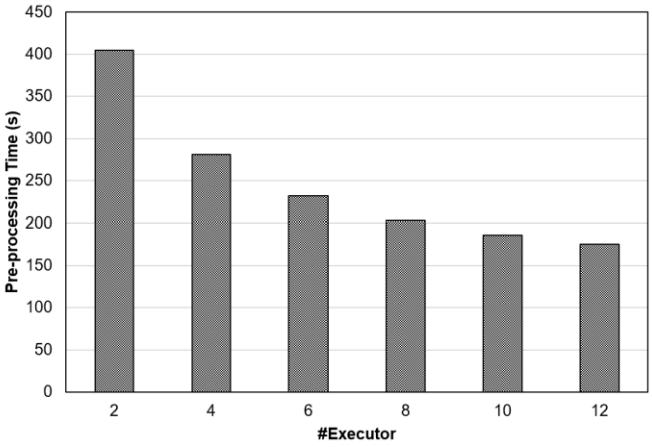

Fig. 1. Pre-processing performance under different executor configurations

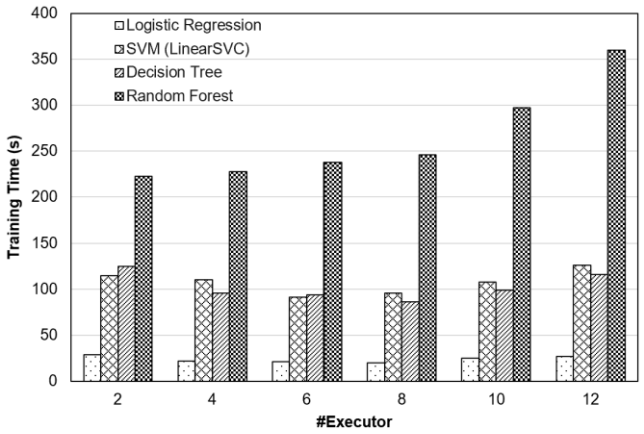

Fig. 2. Comparison of training times for different machine learning algorithms under varying executor configurations

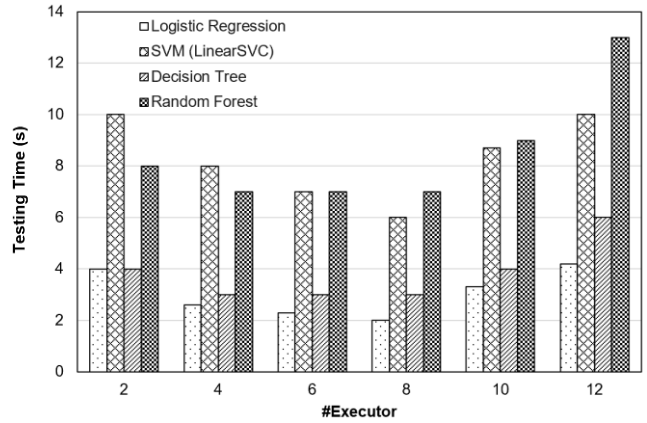

Fig. 3. Comparison of testing times for different machine learning algorithms under varying executor configurations

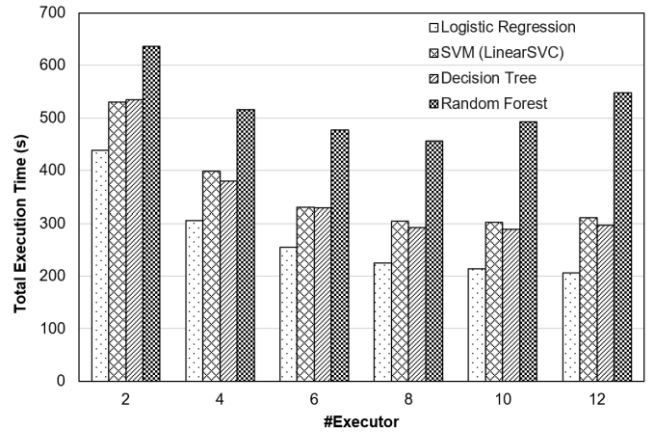

Fig. 4. Comparison of total execution times for different machine learning algorithms under varying executor configurations

We evaluated the end-to-end execution time of four machine learning models. The performance results for training, testing, and total execution time (including preprocessing time) are presented in Fig. 2, Fig. 3, and Fig. 4, respectively. Among the models, Logistic Regression consistently demonstrated the best overall efficiency, with the total execution time decreasing from 438 seconds (2 executors) to 204 seconds (12 executors), reflecting a 53.4% reduction. Similarly, Decision Tree also scaled effectively, improving from 534 to 297 seconds, equivalent to a 44.4% decrease. LinearSVC showed moderate improvement, reducing from 530 to 317 seconds (40.2%), but remained computationally heavier than Logistic Regression and Decision Tree. In contrast, Random Forest exhibited the least scalability, with only a 14.3% decrease in total time (from 636 to 545 seconds), and even experienced performance degradation beyond 10 executors due to increased overhead in coordinating tree ensembles.

It is worth noting that the 10-executor configuration provided a near-optimal balance across models. At this level, preprocessing time was reduced by 54%, and training and inference for most models reached or approached their minimum values. Increasing the executor count beyond this threshold yielded marginal or negative gains, especially for complex models like Random Forest. These findings emphasize that while executor scaling can significantly reduce execution time, the extent of improvement is model-dependent. For medium-scale distributed workloads, using 10 executors offers a practical trade-off between speed, scalability, and system efficiency.

TABLE I. COMPARISON OF PREDICTIVE PERFORMANCE ACROSS CLASSIFICATION MODELS

Models	Accuracy	Precision	Recall	F1-score
Logistic Regression	**0.9555**	**0.9561**	**0.9555**	**0.9563**
SVM (LinearSVC)	0.9528	0.9526	0.9528	0.9528
Decision Tree	0.9112	0.9139	0.9112	0.9124
Random Forest	0.9382	0.9400	0.9382	0.9395

Predictive Performance Evaluation. Table 1 presents the predictive performance of four machine learning models on the news classification task. Among them, Logistic Regression achieved the highest overall performance, with an F1-score of 0.9563, followed closely by LinearSVC (0.9528). Random Forest attained relatively high precision (0.9400) but lower overall effectiveness (F1-score: 0.9395). In contrast, Decision Tree showed the lowest performance across all metrics (F1-score: 0.9124). These results indicate that Logistic Regression is the most reliable model for this task, offering both high accuracy and balanced precision-recall trade-off.

In summary, the experiments show that Logistic Regression offers the best trade-off between execution efficiency and predictive performance. It achieved the fastest runtime and highest F1-score (0.9563) among all models. While Decision Tree and LinearSVC showed moderate scalability, Random Forest suffered from limited efficiency. A 10-executor configuration consistently yielded near-optimal performance, making it well-suited for medium-scale distributed learning on Spark.

V. CONCLUSION

This study evaluated the performance of machine learning algorithms for Vietnamese news classification in a distributed Spark-based environment. By evaluating both execution efficiency and predictive accuracy, our experiments demonstrate that Logistic Regression consistently outperforms other models, offering the best trade-off between computational cost and classification performance. In contrast, Random Forest, despite its high precision, incurred the highest training overhead and exhibited limited scalability. The results also indicate that tuning the number of Spark executors plays a critical role in optimizing processing performance, with a properly balanced configuration yielding substantial improvements across all models.

As future work, we plan to extend this study by incorporating deep learning models such as CNNs and transformer-based architectures (e.g., BERT) to explore their effectiveness in distributed settings. Additionally, we aim to investigate streaming and real-time classification scenarios using Spark Structured Streaming to support low-latency news processing applications.

REFERENCES

[1] Ahmed, Jeelani, and Muqeem Ahmed. "Online news classification using machine learning techniques." IIUM Engineering Journal 22.2 (2021): 210-225.

[2] Wu, Meng-Jin, et al. "A study on natural language processing classified news." 2020 Indo–Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN). IEEE, 2020.

[3] Gómez, J., Gönen, M., & Smola, A. J. (2017). Personalized News Recommendation Using Distributed Representations of Articles. ACM Transactions on Information Systems (TOIS), 36(2), 1-26.

[4] Vosoughi, S., Roy, D., & Aral, S. (2018). The Spread of True and False News Online. Science, 359(6380), 1146–1151.

[5] Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Machine Learning: ECML-98. Springer, Berlin, Heidelberg.

[6] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1746-1751.

[7] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019, 4171-4186.

[8] Abushab, M. M., & Baraka, R. S. (2017, December). Large-Scale Arabic Text Classification Using MapReduce. In The 18th International Arab Conference on Information Technology (ACIT '2017).

[9] Zaharia, Matei, et al. "Apache spark: a unified engine for big data processing." Communications of the ACM 59.11 (2016): 56-65.

[10] Meng, Xiangrui, et al. "Mllib: Machine learning in apache spark." Journal of Machine Learning Research 17.34 (2016): 1-7.

[11] JC Miraclin Joyce Pamila, R Senthamil Selvi, P Santhi, and TM Nithya. Ensemble classifier based big data classification with hybrid optimal feature selection. Advances in Engineering Software, 173:103183, 2022.

[12] Hanxu Wang and Xin Li. Chinese news text classification based on convolutional neural network. Journal on Big Data, 4(1):41, 2022.

[13] Chengcheng Hu, Ying Li, Yongbin Wang, and Lin Wu. Analysis of hot news based on big data. In 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), pages 678–681. IEEE, 2018.

[14] Sunagar, P., Kanavalli, A., Nayak, S. S., Mahan, S. R., Prasad, S., & Prasad, S. (2021, March). News topic classification using machine learning techniques. In International Conference on Communication, Computing and Electronics Systems: Proceedings of ICCCES 2020 (pp. 461-474).

[15] M Jahnavi, K Chandana, Priyanka C Nair, and K Dheeraj. Classification of news category using contextual features. In 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), volume 1, pages 1–7. IEEE, 2024.

[16] Wang, J., Ji, F., Liu, B., Wang, N., Yin, H., & Zhang, F. (2020, October). Research on mass news classification algorithm based on spark. In 2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE) (pp. 408-414). IEEE.