Classification of Urdu News Articles

Rayan Khan
Abdul Rafey
Shehryar Kashif
Raffay Musadiq

Abstract

This study investigates the effectiveness of various machine learning models for multi-class text classification of Urdu news articles from renowned Pakistani media organizations such as ARY, Geo, Jang, Express and Dunya News. After scraping 1500 articles from the websites of these media outlets, models such as Multinomial Naïve Bayes (MNB), Neural Networks, Logistic Regression, and Random Forest were evaluated for their ability to classify Urdu content into distinct categories. MNB achieved the highest accuracy, excelling with sparse, high-dimensional data. Neural Networks demonstrated strong performance by capturing complex patterns, while Logistic Regression performed moderately. Random Forest, however, struggled with data sparsity. The results emphasize the need to align model selection with data characteristics, offering a practical approach for efficient text classification in resource-constrained settings.Hence, this research contributes to enhancing automated news categorization systems for Urdu-speaking audiences, addressing a significant gap in existing literature and technology.

ACM Reference Format:
Rayan Khan, Abdul Rafey, Shehryar Kashif, and Raffay Musadiq. 2018. Classification of Urdu News Articles. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Urdu is the 10th most spoken language in the world, with an estimated number of 230 million native speakers (Wikipedia). It has many similarities with Hindi owing to the common foundation of Sanskrit and Persian. Today, it is the official language of Pakistan, the 5th most populous country in the world. Despite being a widespread means of communication for millions of people, Urdu has not gotten its deserved attention from machine learning engineers. Although there have been attempts to rectify this in recent years with the creation of Urdu libraries, there is still significant work left to be done for a seamless process of textual mining and classification of Urdu media. The existing research in this domain employs complicated models and techniques utilizing Deep Learning (LSTM) and embeddings that are difficult to replicate for ordinary media organizations (UMT). Other papers have attempted to resolve this problem but lack data provenance, have ambiguous methodologies, and lack actionable insights for future research (LGU).

Thus, the main aim of this paper is to test various robust Machine Learning models that classify Urdu news articles into their correct categories (such as business and entertainment) with clear and comprehensive descriptions of data provenance and methodology techniques with simpler Machine Learning models without compromising on accuracy. This would not only aid ordinary news media organizations to deploy and train their own models that are resource-optimized but also facilitate non-machine learning engineers with limited training to conduct future research in this domain on their own.

2 Methodology

2.1 Data Pre-processing, Preparation, and Splitting

Before training the model, we cleaned the data to make it consistent and remove any unnecessary noise. We got rid of missing values as they made up less than 2% of the instances in our dataset. We created a custom list of Urdu stopwords, which included common words to help filter out unimportant tokens. Each article was broken down into individual words, changed to lowercase, and stripped of non-letter characters. Stop words were removed from the processed text to focus on the most relevant content, which became our new training data. The gold label served as the target label. The dataset was split into training and testing sets in 80:20 ratio. Next, we built a vocabulary using the Bag-of-Words class, which scanned all processed sentences, identified unique words, and assigned each word a unique index in sorted order.

2.2 Exploratory Data Analysis

Before we start to create models based on the data, it is important to explore understand its data distributions and patterns. Firstly, we look at Fig.1, which represents the distribution of article categories within our dataset. We see that the entertainment, sports, business, and world classes are quite balanced with similar occurrences in the dataset. However, we see that the science-technology class seems to be slightly underrepresented. This signals to us that there might be a risk of slight bias in the model where it favours the majority class, and that we should judge evaluation metrics keeping this in mind.

Secondly, we explore the lengths of the articles in our dataset. Fig. 2 shows us that most articles have a length of 100 - 250 words. Fig. 3 shows the average length of articles in each category. We see that entertainment articles tend to be the longest, averaging more than 400 words per article. The shortest articles tend to be of science-technology, which average about 200 words per article.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

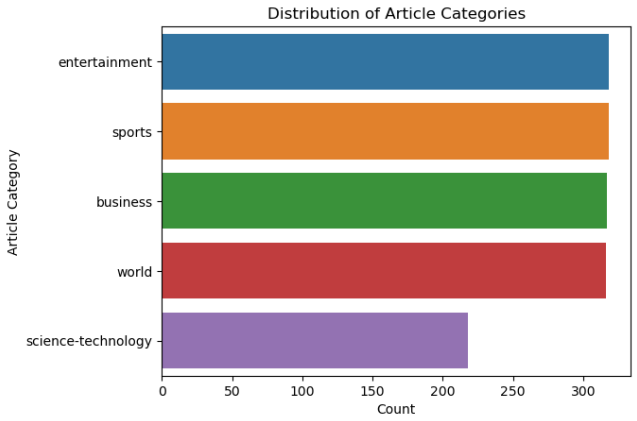

Figure 1: Distribution of Article Categories

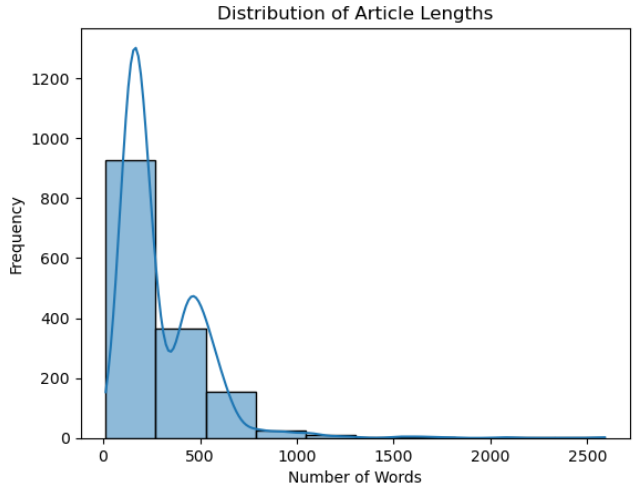

Figure 2: Distribution of Article Lengths

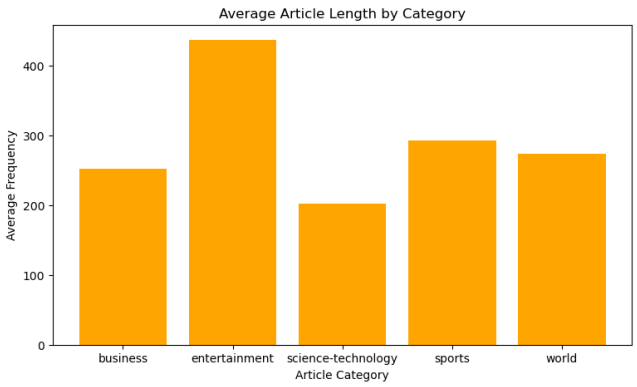

Figure 3: Article Length by Category

2.3 Multinomial Naive Bayes

For our first model, Multinomial Naive Bayes was chosen because it is better in text classification tasks with discrete features, such as word counts from a Bag-of-Words model. MNB's use of class priors and Laplace smoothing ensures robustness against unseen words and class imbalances, and it accurately classifies the content as per their gold labels. MNB is efficient, easy to implement, and works well even with noisy or sparse data. These qualities make it a great choice for tasks like document classification and sentiment analysis. It handles large datasets well and can classify text effectively even with limited information. For MNB, we added Laplace smoothing to handle words that might not appear in the training data. During training, we calculated class probabilities and the likelihood of each word in a class, ensuring no probabilities were zero. The fit function calculates the class priors by counting the occurrences of each class in the target vector. For each class, the method calculates the conditional probability of each word given the class. This is done by summing the word counts for each class and then normalizing by the total word count for that class. This step ensures that every word has a non-zero probability due to the smoothing term. Finally, the predict method calculates the log of the class prior for each class and adds the log of the conditional probabilities of the words in the input sample. This approach avoids underflow errors that occur when multiplying small probabilities. The class with the highest log-probability is selected as the prediction. We further analyze the model performance using evaluation metrics.

2.4 Neural Network

For our second model we chose Neural Network because of its ability to capture complex, non-linear relationships in the data, that makes it ideal for text classification tasks. The layers of the neural network helped turn the scattered Bag-of-Words features into more useful patterns. The identification of these patterns was helpful in correctly classifying the correct genre for the content. For our NN model, initially, labels are encoded using LabelEncoder to convert them into numerical format, as neural networks require numerical labels, which are than converted to tensors train test data. We built a custom neural network class in PyTorch, named TextClassificationNN. The input layer was designed to match the size of our Bag-of-Words vectors. The network featured two hidden layers both using the ReLU activation function to ensure smooth learning and avoid vanishing gradients. For training, we relied on CrossEntropyLoss, a go-to choice for multi-class classification since it combines softmax activation with negative log-likelihood loss and also caters for class imbalances. To optimize the model, we used the Adam optimizer due to its robust nature and its ability to converge faster than traditional optimizers like SGD. By allowing 20 epochs, we provided the model with enough iterations to learn patterns in the data without running the risk of overfitting. We also used a small batch size in data loaders (16). Smaller batch sizes introduce noise into the gradient updates, which can help escape local minima and improve generalization.

2.5 Multinomial Logistic Regression

For our third model, we turned to implementing a custom Logistic Regression model from scratch. Multinomial Logistic Regression

was selected as it is inherently suited for multi-class classification problems, which aligns perfectly with the task of categorizing news articles into multiple predefined categories such as sports, politics, and entertainment. Its probabilistic nature allows it to model the relationship between the features extracted from text data and the categorical outcomes effectively. Additionally, Logistic Regression is relatively simple to implement and interpret, making it an attractive choice for future researchers who may be exploring machine learning techniques for the first time. Logistic Regression also performs well with high-dimensional data (BOW in our case) typical in text classification tasks due to its reliance on the concept of odds ratios and log-odds, which can provide insights into the significance of various features and help in feature selection.

To implement our model, we built a custom class that harnessed the training features to fit the sigmoid function and then calculated the cross-entroopy loss by using the predicted and actual train labels. To enhance our accuracy and make our findings more robust, we iterated over our training corpus using different learning parameters (0.001,0.01) and number of iterations (3000, 5000) to find the optimal classifier for each class. We also deployed L2 regularization to help us scale down the weights of unimportant features to focus on the more important ones. Once this was done, we tested our multi-class model on the test dataset to predict the probabilities of each category for each every data instance. The category with the highest probability was assigned as the predicted label for the class which was then compared with the actual test labels. This helped us analyze the model's performance using evaluation metrics such as accuracy, macro-average f1, precision and recall.

2.6 Random Forest

For our fourth model, a random forest model utilizing a bag of words strategy was implemented. This model was chosen as it works well against overfitting and works well with high dimensional BOW matrices. Each tree in the forest can take different features into account and learn from them. Considering that the BOW matrix consists of over a thousand features, the random forest is a good candidate. This approach works well with very large datasets and feature intensive projects. Furthermore, the ensemble approach of the model reduces bias and variance.Lastly, Bagging is used for the final prediction by taking a majority vote across all trees.

The maximum depth of the tree was set to 15 to ensure trees capture the right patterns without overfitting to the dataset. While the model could have gone up to a depth of 20-25, it would have diminishing returns before overfitting.

10 trees were selected to balance computational efficiency and ensemble performance. The number of trees could have increased the accuracy, as increasing the trees from 10 to 20 gave an accuracy increase of about 5%. However, computation time skyrocketed due to the model being a custom implementation and as such, not supporting GPU computations using CUDA as one would typically do with premade libraries.

Each tree was trained on 80% of the training dataset (via bootstrapping). This improved generalization and allowed each tree to learn patterns in the data without overfitting to noise.

Gini Impurity was used to determine split points, ensuring that each tree remains as pure as possible at higher depths. The splitting continues till the specified depth is reached. In order to keep runtime down to allow for testing, some accuracy was lost. This includes shortening the depth, the number of trees, and only choosing a certain number of most repeating features. With more computational power and time, accuracy could potentially increase more before running into overfitting. Majority voting aggregated predictions from the ensemble, ensuring that predictions have high accuracy.

3 Results

3.1 Multinomial Naive Bayes:

Accuracy	Precision	Recall	F1 Score
0.9631	0.9625	0.9644	0.9626

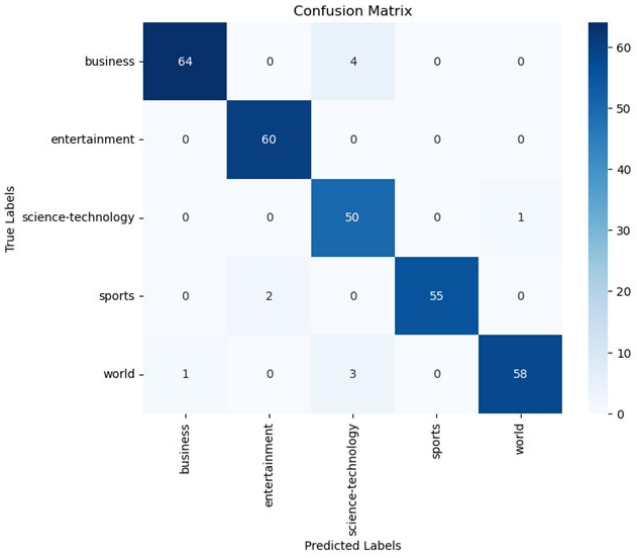

Figure 4: Heat Map for MNB

3.2 Neural Networks

Accuracy	Precision	Recall	F1 Score
0.9564	0.9571	0.9564	0.9563

3.3 Logistic Regression

Accuracy	Precision	Recall	F1 Score
0.9463	0.950	0.94	0.95

3.4 Random Forest

Please find the relevant figures/tables below.

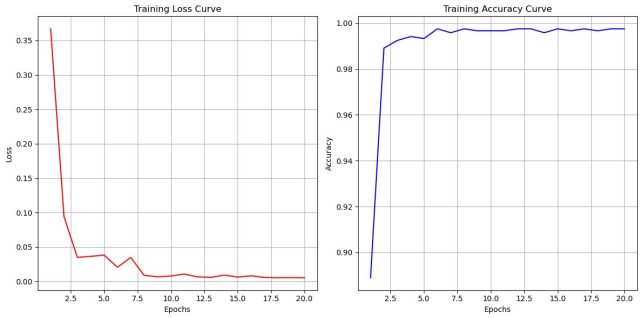

Figure 5: Plot for Accuracy and Loss Curves for NN

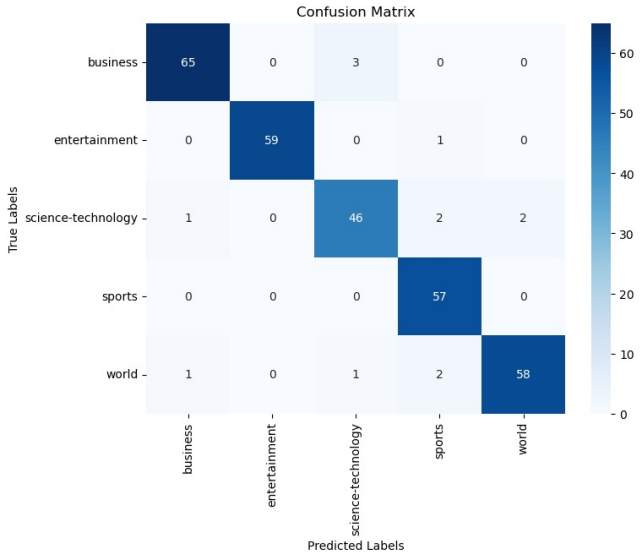

Figure 6: Heat Map for NN

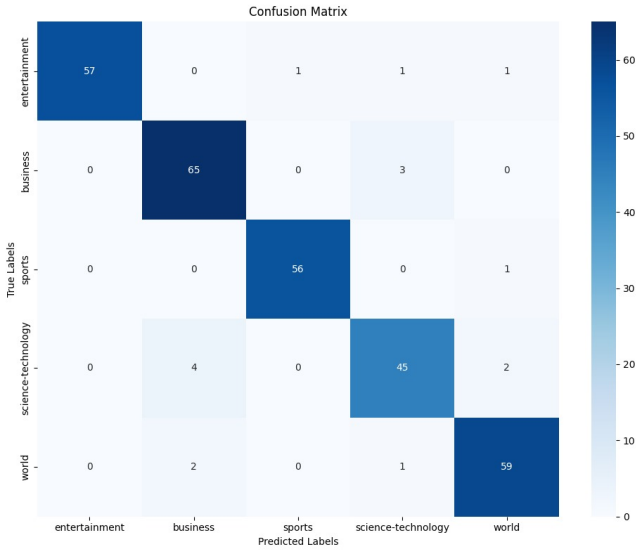

Figure 7: Heat Map for Logistic Regression

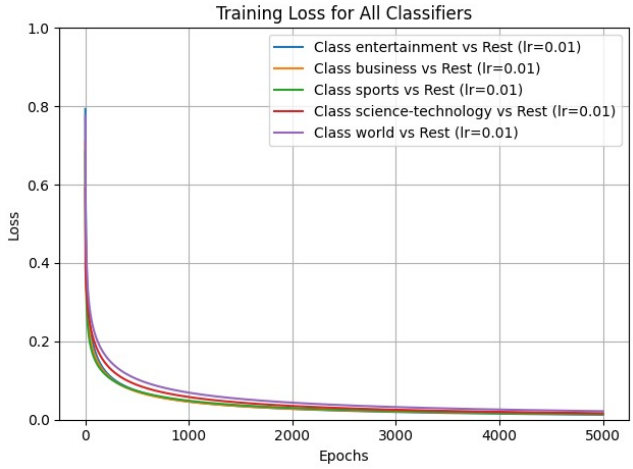

Figure 8: Training Loss for Logistic Regression

Accuracy	Precision	Recall	F1 Score
0.8121	0.8153	0.8066	0.8093

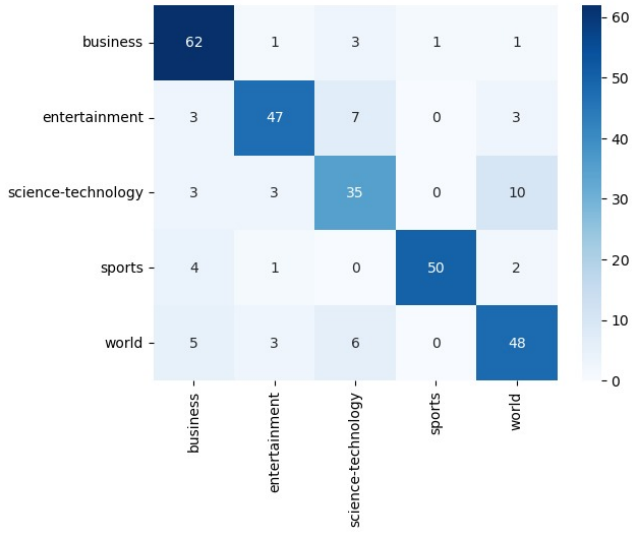

Figure 9: Heat Map for Random Forest

3.5 Commentary on Results

The overall performance of our models underscores the adaptability of various approaches to multi-class text classification. Multinomial Naïve Bayes (MNB) emerged as the most effective, achieving the highest accuracy and balanced metrics across precision, recall, and F1 score. This can be attributed to MNB's assumption of word independence, which simplifies computations and aligns well with the dataset's characteristics—dominated by high-dimensional and sparse text features.

Neural networks, with slightly less performance than MNB, delivered robust results, indicating their potential for capturing complex relationships in the data. However, their need for extensive training

and computational resources points to a trade-off between flexibility and efficiency. Their capacity to model complex, non-linear relationships suggests their suitability for more nuanced datasets, albeit at the cost of increased computational demands and longer training times.

Logistic regression provided moderate results, benefiting from its straightforward design but lacking the sophistication to fully exploit the intricacies of the dataset. Similarly, Random Forest struggled to cope with the sparsity and high dimensionality of textual data, which likely impeded its ability to construct effective decision trees. These results reflect the importance of aligning model complexity and assumptions with the nature of the data.

Overall, the findings suggest that while advanced models like neural networks hold promise, simpler, well-suited models like MNB can achieve high performance efficiently in text-based classification tasks.

4 Conclusion and Limitations

The MNB model outperformed others in this study, achieving an impressive 96% accuracy, closely followed by Neural Networks. In contrast, Logistic Regression and Random Forest delivered moderate results, revealing their limitations in effectively managing high-dimensional, sparse data and capturing the intricate patterns necessary for optimal classification.

However, the study does have its limitations that have to be noted. The MNB model assumes that words are independent from each other, which is hardly the case in textual data. Moreover, our study relied on the Bag-of-Words model for text representation, which may overlook semantic relationships between words and the context in which they appear. More advanced techniques like word embeddings (e.g., Word2Vec or fastText) could provide better representations and potentially improve model performance. Lastly, while custom stopwords and preprocessing steps were employed, Urdu's rich lexical morphology and the complexities of its script (e.g., word inflections, compound words) might not have been fully captured in the preprocessing phase.

5 Future Research

In future research, several promising avenues can be explored to further improve the performance of Urdu text classification models.

Cross-Lingual Transfer Learning: We could further explore pre-trained models from languages like Hindi or English and fine-tune them for Urdu. Since these languages share similarities, this approach could improve model performance, especially when resources for Urdu-specific models are limited.

Sentiment Analysis Integration: Integrating sentiment analysis with topic categorization could add depth by analyzing the tone of articles (positive, negative, or neutral) alongside their topics. This would enable more dynamic and insightful content classification.

6 Works Cited

Wikipedia contributors. "List of Languages by Total Number of Speakers." *Wikipedia*, https://en.wikipedia.org/wiki/List_of_languages_by_total_nu Accessed 8 Dec. 2024.

"UMT AIR." *University of Management and Technology*, https://doi.org/10.32350/u air.32.05. Accessed 8 Dec. 2024.

"LGU Journal of Computer Science and Information Technology." *Lahore Garrison University*, https://lgurjcsit.lgu.edu.pk/lgurjcsit/article/view/274/20 Accessed 8 Dec. 2024.