Optimal Classifier to Predict News Article Categories Using Machine Learning: A Study of Urdu News Pages

Ahmed Hassan
26100308@lums.edu.pk
Lahore University of Management Sciences (LUMS)
Lahore, Pakistan

Muhammad Saqib Bukhari
24060006@lums.edu.pk
Lahore University of Management Sciences (LUMS)
Lahore, Pakistan

Arsalan Hashmi
25100260@lums.edu.pk
Lahore University of Management Sciences (LUMS)
Lahore, Pakistan

Usman Sajid
26100027@lums.edu.pk
Lahore University of Management Sciences (LUMS)
Lahore, Pakistan

Luqman Adil
25100023@lums.edu.pk
Lahore University of Management Sciences (LUMS)
Lahore, Pakistan

Figure 1: A Pakistani local scanning through an Urdu newspaper

Abstract

This project aims to develop and compare machine learning models for classifying Urdu news articles into five distinct categories. The dataset, extracted from five major Urdu news sites: Geo, Jang, Dawn, Samaa, and Express News, contains articles labeled with one of five predefined categories. Extracting relevant content such as titles and paragraphs from the websites posed challenges due to varying HTML tag structures across the sites. Additionally, preprocessing the text data involved removing unwanted characters, such as punctuation, numbers, and English alphabets, as well as eliminating Urdu stopwords sourced from Kaggle.

We implemented and evaluated three classification algorithms — Naive Bayes, Neural Networks, and Logistic Regression—each chosen for its unique characteristics. Naive Bayes is well-suited for text classification tasks, leveraging probabilities for efficient processing. Neural Networks were explored for their ability to capture complex patterns in text data, while Logistic Regression was selected for its simplicity and interpretability. For feature extraction, a Bag-of-Words (BoW) model was employed to convert the text data into a numerical format suitable for machine learning.

The models were evaluated using accuracy, precision, recall, and F1 scores, alongside a confusion matrix, with an 80-20 train-test split. The results provide insights into the efficacy of various machine learning models for text classification in the Urdu language and highlight the strengths and weaknesses of each approach when applied to the categorization of news articles.

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://lums.edu.pk

Keywords

Machine Learning, Text Classification, Urdu News, NLP, Data Scraping

ACM Reference Format:
Ahmed Hassan, Muhammad Saqib Bukhari, Arsalan Hashmi, Usman Sajid, and Luqman Adil. 2024. Optimal Classifier to Predict News Article Categories Using Machine Learning: A Study of Urdu News Pages. In . ACM, New York, NY, USA, 7 pages. https://lums.edu.pk

1 Methodology

In the initial stages of this project, we implemented a web scraping solution to gather news articles from various sources, including *Samaa*, *Jang*, *Geo*, *Dawn*, and *Express*. The goal was to collect a wide range of articles in different categories such as entertainment, business, sports, science-technology, and world news. The collected articles from all sources were combined into a single CSV file, which was cleaned and preprocessed to remove noise, irrelevant information, and duplicates.

The finalized dataset named **cleaned_combined_articles.csv** consisted of **1395** distinct articles all shuffled from 5 different news outlets, all assigned a label from five distinct categories referenced.

Table 1: CSV File Format for Scraped Data

id	title	link	content	gold_label
1	Title	https://...	Sample Content	business
2	$Title_2$	https://...	Another Content	sports
...	...	...	...	...

Unique labels

Categories
Business
Entertainment
International
Science & Technology
Sports

To train and evaluate our machine learning models, we split the data into an 80-20 train-test split. This ensured that the models were trained on 80% of the data while the remaining 20% was reserved for testing their performance.

We applied three different machine learning models for the classification task: Multinomial Naive Bayes (MNB), Logistic Regression (LR), and an Artificial Neural Network (ANN). The models achieved respective accuracies of 97.03%, 95.8%, and 87.5%. A bar graph (see Figure 2) visualizes the comparison of these accuracies.

Based on the evaluation results, we chose the Multinomial Naive Bayes model as our final model due to its highest accuracy and simplicity in implementation.

Table 2: Model Accuracy ranges

Model	Accuracy	Variation
Naive Bayes Classifer	97.6%	Low
Artificial Neural Network	87.5%	Low
Logistic Regression	95.8%	High

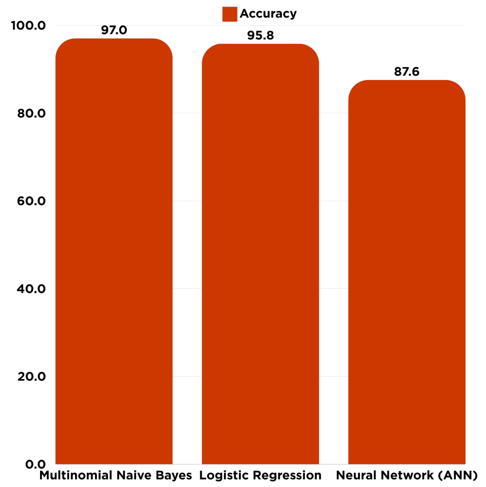

Figure 2: Comparison of accuracies for MNB, LR, and ANN models.

1.1 Web Scraping Implementation

The scraping was done using Python and libraries such as `requests`, `BeautifulSoup`, and `pandas` for extracting and storing the data.

This code scrapes articles from the news websites by iterating through different categories, extracting article titles, links, and content.

2 Data Preprocessing

After the data was scraped, we proceeded to clean the articles by removing unwanted characters, handling missing data, and normalizing text. The text was tokenized, stopwords were removed, and the data was vectorized into a format suitable for machine learning models.

2.1 Removing Non-Urdu Characters and Stopwords

Since the dataset is completely in Urdu, it was essential to remove any non-Urdu characters, including English alphabets, special characters, and digits. Additionally, stopwords are common words that do not contribute significant meaning to the text, such as **aur, hai, etc**. Hence using a dataset from Kaggle, stopwords were removed from the content and title columns. This step reduced noise in the text data and enhances the quality of the features used for analysis.

2.2 Data Normalization

Normalization involves converting text data to a consistent format. In this step, we stripped any leading or trailing whitespaces from the content and title columns. This ensured that the text data was uniformly formatted, which was important for subsequent text processing tasks.

2.3 Exploratory Data Analysis (EDA)

To understand the distribution and patterns in the data, we perform exploratory data analysis. This includes visualizing the distribution of articles by category and generating a word cloud for the article content.

2.3.1 Distribution of Articles by Category. A count plot was materialized to visualize the distribution of articles across different categories. This helped us in understanding the representation of each category in the dataset.

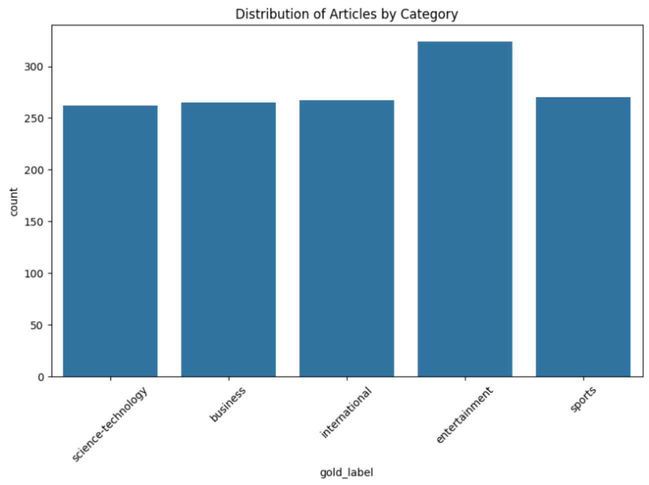

Figure 3: Distribution of Articles by Category

2.3.2 Word Cloud for Article Content. A word cloud is generated to visualize the most frequent words in the article content. This helps in understanding the prominent themes in the dataset. The word cloud provides a visual representation of the most common words, with the size of each word indicating its frequency.

While the wordcloud is technically inaccurate given that Urdu script is read backwards and current libraries cannot properly plot it, it still manages to give a ballpark representation of word distribution

3 Machine Learning Models

We applied three different machine learning models to classify the scraped news articles into predefined categories:

3.1 Multinomial Naive Bayes (MNB)

The Multinomial Naive Bayes model was used for its simplicity and effectiveness in text classification problems, particularly in cases where the feature (word frequency) is discrete. The model performed exceptionally well, achieving an accuracy of 97.03%.

Figure 4: Word Cloud for Article Content

The MNB model applies Bayes' theorem, which is mathematically expressed as:

$$P(C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(w_i|C_k)}{P(X)} \tag{1}$$

where:

- $P(C_k|X)$: Posterior probability of class C_k given the input X.
- $P(C_k)$: Prior probability of class C_k.
- $P(w_i|C_k)$: Probability of word w_i occurring in class C_k.
- $P(X)$: Evidence, which is a normalizing constant.

3.1.1 Laplace Smoothing. To address the zero-frequency problem, Laplace smoothing was applied. The conditional probability of a word w_i given class C_k is calculated as:

$$P(w_i|C_k) = \frac{\text{count}(w_i, C_k) + \alpha}{\sum_{j=1}^{V} \text{count}(w_j, C_k) + \alpha V} \tag{2}$$

Here:

- $\text{count}(w_i, C_k)$: Frequency of word w_i in class C_k.
- V: Vocabulary size.
- α: Smoothing parameter (typically set to 1 for Laplace smoothing).

3.1.2 Prior Calculation. The prior probability for each class C_k is computed as:

$$P(C_k) = \frac{\text{count}(C_k)}{\sum_{j=1}^{N} \text{count}(C_j)} \tag{3}$$

where:

- $\text{count}(C_k)$: Number of instances in class C_k.
- N: Total number of instances across all classes.

The dataset consisted of news articles with their respective titles and content. To prepare the data for modeling, we concatenated the title and content of each article into a single text field. This combined text field served as the input feature for the model.

```
X['text'] = X['title'] + " " + X['content']
```

3.2 Feature Extraction

For feature extraction, we used a custom Bag of Words (BoW) class to create a vocabulary from the training data and convert each text into a count vector. This approach helped in converting the text data into numerical features that could be used as input to the machine learning model.

3.3 Model Training

The training process involved the following steps:

(1) **Vocabulary Creation:** The BoW class was used to create a vocabulary from the training data.
(2) **Count Vector Creation:** Each text was converted into a count vector based on the vocabulary.
(3) **Class Prior Calculation:** The prior probabilities of each class were calculated based on their frequencies in the training data.
(4) **Word Probability Calculation:** The conditional probabilities of each word given a class were calculated using Laplace smoothing.

The Multinomial Naive Bayes (MNB) classifier was chosen for this task due to its effectiveness in handling text data and its simplicity. The MNB classifier is based on Bayes' theorem and assumes that the features follow a multinomial distribution. It is particularly useful for discrete data, such as word counts.

3.4 Model Evaluation

To evaluate the performance of the model, we used metrics such as accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the model's performance in predicting the labels. The confusion matrix was also used to visualize the performance of the model across different classes.

Table 3: Model Evaluation Metrics

Metric	Score
Accuracy	0.9704
Precision	0.9676
Recall	0.9707
F1-score	0.9681

Table 4: Confusion Matrix

	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	55	0	0	2	0
Class 2	0	66	0	0	1
Class 3	1	1	47	2	0
Class 4	0	0	0	40	0
Class 5	0	0	0	1	54

3.5 Conclusion

The Multinomial Naive Bayes classifier, combined with the Bag of Words feature extraction, provided an effective approach for predicting labels in the dataset. The model achieved satisfactory performance metrics, demonstrating its suitability for text classification tasks. Future work may involve exploring more advanced models and feature extraction techniques to further improve the prediction accuracy.

4 Logistic Regression (LR)

Logistic Regression is another popular choice for binary and multi-class classification problems. It also showed strong performance, with an accuracy of 95.8%. Logistic Regression models the probability of a certain class based on the features of the input text.

The probability of an instance X belonging to a class C_k is modeled as:

$$P(C_k|X) = \frac{1}{1 + e^{-z}} \tag{4}$$

where:

- $z = w^T X + b$, the linear combination of the feature vector X, weights w, and bias b.

For multi-class classification, the Softmax function is used to generalize the logistic regression model:

$$P(C_k|X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

where:

- $z_k = w_k^T X + b_k$, representing the linear score for class C_k.
- K: Total number of classes.

4.0.1 Loss Function. The model is optimized using the Cross-Entropy Loss, defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log P(C_k|X_i) \tag{6}$$

where:

- N: Number of training instances.
- $y_{i,k}$: Binary indicator (1 if instance i belongs to class C_k, 0 otherwise).
- $P(C_k|X_i)$: Predicted probability for class C_k for instance i.

The model parameters w and b are learned using optimization techniques such as Gradient Descent or its variants.

4.0.2 Regularization. To prevent overfitting, Logistic Regression incorporates regularization. The L2 regularization term (Ridge penalty) is expressed as:

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \frac{\lambda}{2} \sum_{j=1}^{M} w_j^2 \tag{7}$$

where:

- λ: Regularization strength.
- M: Number of features.
- w_j: Weight for feature j.

4.1 Feature Extraction

For feature extraction, we used a custom Bag of Words (BoW) class to create a vocabulary from the training data and convert each text into a count vector. This approach helped in converting the text data into numerical features that could be used as input to the machine learning model.

4.2 Model Training

The Logistic Regression classifier was chosen for this task due to its simplicity and effectiveness in handling text data. The training process involved the following steps:

(1) **Initialization:** The model parameters (weights and bias) were initialized.
(2) **Forward Propagation:** The linear combination of input features and weights was passed through a sigmoid activation function to obtain the predicted probabilities.
(3) **Loss Calculation:** The cross-entropy loss was calculated to measure the difference between the predicted probabilities and the actual labels.
(4) **Backward Propagation:** The gradients of the loss with respect to the model parameters were calculated and used to update the parameters using gradient descent.
(5) **Iteration:** The forward and backward propagation steps were repeated for a specified number of epochs.

```
mod = LogisticRegressionScratch(lr=0.1, epochs=1000)
mod.fit(train_x, train_y)
```

4.3 Model Evaluation

To evaluate the performance of the model, we used metrics such as accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the model's performance in predicting the labels. The confusion matrix was also used to visualize the performance of the model across different classes.

Table 5: Model Evaluation Metrics

Metric	Score
Accuracy	95.85%

Table 6: Classification Report

Class	Precision	Recall	F1-score	Support
0	0.97	0.99	0.98	70
1	0.94	0.99	0.96	76
2	0.94	0.94	0.94	66
3	0.96	0.92	0.94	50
4	0.99	0.95	0.97	75
Accuracy	0.96			
Macro avg	0.96	0.96	0.96	337
Weighted avg	0.96	0.96	0.96	337

4.4 Conclusion

The Logistic Regression classifier, implemented from scratch and combined with the Bag of Words feature extraction, provided an effective approach for predicting labels in the dataset. The model achieved satisfactory performance metrics, demonstrating its suitability for text classification tasks. Future work may involve exploring more advanced models and feature extraction techniques to further improve the prediction accuracy.

5 Artificial Neural Network (ANN)

Finally, we employed an Artificial Neural Network, which is highly effective for complex tasks like text classification due to its ability to learn intricate patterns in large datasets. The ANN model performed well with an accuracy of approximately 87%.

5.0.1 Architecture. The network consisted of:

- **Input Layer:** Processes the Bag-of-Words feature vectors.
- **Hidden Layer:** Includes 64 neurons with ReLU activation and dropout ($p = 0.7$) to prevent overfitting.
- **Output Layer:** A softmax layer with neurons equal to the number of classes (K), outputting class probabilities.

The forward pass for a hidden layer h is given by:

$$h = \mathrm{ReLU}(WX + b) \tag{8}$$

where:

- W: Weight matrix.
- b: Bias vector.
- $\mathrm{ReLU}(x) = \max(0, x)$: Activation function.

The output probabilities for the classes are computed using the softmax function:

$$P(C_k|X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{9}$$

Optimization was performed using the Adam optimizer, known for its adaptive learning rate and efficient convergence.

5.0.2 Training and Evaluation. The model was trained over multiple epochs with a learning rate of 0.001 and a batch size of 32. Performance was evaluated using accuracy, precision, recall, and F1-score.

5.1 Feature Extraction

For feature extraction, we used the Bag of Words (BoW) model to create a vocabulary from the training data and convert each text into a count vector. The count vectors were then standardized using the StandardScaler to ensure that the features had a mean of 0 and a standard deviation of 1.

5.2 Hyper-parameters:

The following hyper-parameters were selected after much modification and experimentation: **1. Input dimensions:** 14865 Since a bag-of-words approach is utilized, a vocabulary vector is generated for each piece of content. **2. Loss Function:** Since the problem at hand is a multi-classification task, BCE does not fit. To account for classifying the text into one of five categories, cross entropy has to be used.

3. Learning Rate: 0.0001 Picking a larger learning rate for this use-case consistently lead to over-fitting and plateauing. This is an indication that the model has found a false minima during gradient descent.

4. Number of Epochs: 250 Similar to the learning rate, determining the number of epochs took multiple training rounds. A common problem that arose was that the loss would decrease and then start increasing again. Hence, the number of epochs is very closely tied to the learning rate because the size and number of weight updates are determined by these two factors.

5.3 Model Training

The artificial neural network (ANN) was chosen for this task due to its ability to capture complex patterns in the data. The ANN consisted of an input layer, one hidden layer with ReLU activation, and an output layer. Dropout was applied to the hidden layer to prevent overfitting. The training process involved the following steps:

(1) **Initialization:** The model parameters (weights and biases) were initialized.
(2) **Forward Propagation:** The input features were passed through the network to obtain the predicted probabilities.
(3) **Loss Calculation:** The cross-entropy loss was calculated to measure the difference between the predicted probabilities and the actual labels.
(4) **Backward Propagation:** The gradients of the loss with respect to the model parameters were calculated and used to update the parameters using the Adam optimizer.
(5) **Iteration:** The forward and backward propagation steps were repeated for a specified number of epochs.

5.4 Model Evaluation

To evaluate the performance of the model, we used metrics such as accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the model's performance in predicting the labels. The confusion matrix was also used to visualize the performance of the model across different classes.

```
model.evaluate()
```

Table 7: Model Evaluation Metrics

Metric	Score
Accuracy	86.67%
Precision	88.74%
Recall	86.63%
F1-score	86.72%

Table 8: Confusion Matrix

	Class 0	Class 1	Class 2	Class 3	Class 4
Class 0	27	0	1	0	0
Class 1	1	21	0	0	0
Class 2	5	0	16	0	1
Class 3	2	0	0	29	0
Class 4	5	2	1	0	24

5.5 Conclusion

The artificial neural network, implemented using PyTorch and combined with the Bag of Words feature extraction, provided an effective approach for predicting labels in the dataset. The model achieved satisfactory performance metrics, demonstrating its suitability for text classification tasks. Future work may involve exploring more advanced models and feature extraction techniques to further improve the prediction accuracy.

6 Conclusion

The combination of web scraping, data cleaning, and machine learning models allowed us to classify news articles with high accuracy. The best results were obtained with the Multinomial Naive Bayes model, which was selected as the final model for its simplicity and highest accuracy among the three models. Future work can involve hyperparameter tuning, expanding the dataset, and exploring deep learning methods for further improvements.

7 Findings

The performance of the three machine learning models varied based on their inherent assumptions, algorithms, and suitability for text data. Below is an analysis of the results:

7.1 Multinomial Naive Bayes (MNB)

The MNB model achieved the highest accuracy of 97.03%. This can be attributed to its assumptions about the independence of features and its suitability for text classification tasks. According to Nigam et al. [1], the MNB classifier is particularly effective for high-dimensional datasets like textual data. The Bag of Words (BoW) feature extraction approach preserved the most relevant information for classification, which significantly contributed to the model's performance.

7.2 Logistic Regression (LR)

Logistic Regression showed strong performance with an accuracy of 95.85%. However, it fell slightly behind MNB due to its inability to naturally leverage the discrete nature of word counts. Despite this, its interpretability and linear decision boundaries make it a suitable choice for multi-class text classification tasks.

7.3 Artificial Neural Network (ANN)

ANNs achieved an accuracy of 86.67%. While ANNs are known for their ability to capture complex patterns, their performance is heavily dependent on large, diverse datasets [2]. The smaller dataset used in this study and limited hyperparameter tuning might have constrained its ability to generalize effectively.

7.4 Visual Comparison of Models

8 Limitations and Conclusion

8.1 Limitations

- **Dataset Size and Diversity:** The dataset was limited in size and diversity, which restricted the ANN's ability to generalize effectively.
- **Feature Engineering:** While effective, the Bag of Words approach does not capture semantic relationships between words. Advanced techniques such as word embeddings could improve the feature representation.
- **Hyperparameter Tuning:** Limited computational resources restricted comprehensive hyperparameter tuning, especially for ANN.

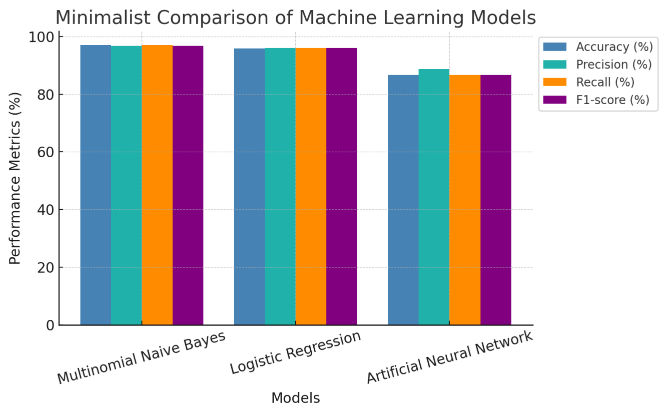

Figure 5: Comparison of Model Accuracies

- **Bias in Data:** The dataset was sourced from a limited number of online news outlets, potentially introducing bias into the model predictions.

References

[1] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39, 2 (2000), 103–134.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2015. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1511.07528* (2015).