Supervised Machine Learning for Urdu Text Classification

Asfandiyar Safi
25020221@lums.edu.pk
LUMS
Lahore, Pakistan

Shahmir Qazi
shahmirqazi5@gmail.com
LUMS
Lahore, Pakistan

Raza Hamid
26100319@lums.edu.pk
LUMS
Lahore, Pakistan

Ibrahim Khan
26100261@lums.edu.pk
LUMS
Lahore, Pakistan

Abstract

The growing accessibility of online content in regional languages has amplified the need for effective and automated classification methods. In this study, we present a supervised machine learning approach for categorizing Urdu text from online newspaper sources into predefined categories. Urdu, a language rich in morphology and structure, poses unique challenges in natural language processing, including preprocessing and feature extraction. To address these challenges, we collected and preprocessed a dataset of Urdu newspaper articles, ensuring proper handling of stopwords, punctuation, and tokenization. Our methodology involved training three distinct machine learning models, each leveraging the preprocessed textual data paired with their respective labels. We evaluated the models on their ability to accurately predict the category of unseen Urdu text, focusing on metrics such as precision, recall, and F1-score. By comparing the performance of these models, we identified the most effective approach for Urdu text classification within the constraints of our dataset. This work contributes to the development of computational tools for Urdu text classification, paving the way for more personalized and efficient information retrieval systems in regional languages. The results highlight the potential of machine learning in addressing linguistic diversity and enhancing accessibility to categorized information for Urdu-speaking audiences.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; *Supervised learning by classification.*

Keywords

Urdu NLP, Text Classification, Supervised Machine Learning, Naive Bayes, Softmax Classifier

ACM Reference Format:
Asfandiyar Safi, Shahmir Qazi, Raza Hamid, and Ibrahim Khan. 2024. Supervised Machine Learning for Urdu Text Classification. In *Proceedings of LUMS Machine Learning Course Project (Course Project).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Course Project, December, 2024, Lahore, Pakistan

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The exponential growth of online regional content has underscored the need for advanced natural language processing (NLP) techniques tailored for underrepresented languages. This report explores a machine learning-based approach for the classification of Urdu text, drawn from online newspaper sources, into predefined categories. Unlike widely studied languages, Urdu presents unique challenges due to its complex grammatical structures, rich morphological features, and unique script. The primary objective of this study is to address these challenges by creating a robust supervised learning framework. The project involves collecting a dataset of Urdu news articles, preprocessing the text to prepare it for analysis, and applying various machine learning models to classify the text. By examining the performance of multiple algorithms, we aim to identify the most effective model for categorizing Urdu text. Through this effort, we hope to advance the field of Urdu NLP, enabling practical applications such as automated news categorization, personalized content delivery, and improved information retrieval for Urdu-speaking audiences.

2 Naive Bayes Model

2.1 Introduction to the Theorem

To begin our project, we implemented a simple and efficient model to establish a baseline for predicting our gold labels. The first model utilized was the Bernoulli Naive Bayes model, a probabilistic approach grounded in Bayes' Theorem. Named after Thomas Bayes, an 18th-century philosopher and statistician, Bayes' Theorem provides a mechanism to calculate the probability of an event based on prior knowledge of conditions related to the event. The formula for Bayes' Theorem is expressed as:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{1}$$

$P(h|D)$ is Posterior probability of hypothesis given data, $P(D|h)$ is the Likelihood of data given hypothesis, $P(h)$ is the prior probability of hypothesis, and $P(D)$ is Marginal probability of data.

The Bernoulli Naive Bayes model operates under the assumption that features (words) are binary—either present or absent in a document. This simplicity makes it computationally efficient and suitable for scenarios where the presence of certain keywords strongly correlates with specific categories. While the Bernoulli

model provided a foundational understanding of probabilistic text classification, the nature of our dataset—where word frequency plays a significant role in determining categories—necessitated the use of a more sophisticated approach. We therefore extended our analysis to the Multinomial Naive Bayes model. This variation accommodates the frequency of words in a document, allowing us to model the probabilistic influence of repeated terms more effectively. The formula for this model is given as:

$$P(X_1, X_2, X_3, \ldots, X_n \mid \text{Class}) = P(X_1 \mid \text{Class}) \cdot P(X_2 \mid \text{Class}) \cdot \cdots \cdot P(X_n \mid \text{Class}) \quad (2)$$

This formulation enabled us to integrate the nuanced patterns of word distribution within the text, enhancing the model's predictive capabilities. Starting with Naive Bayes models not only allowed us to establish a strong baseline but also served as a steppingstone for exploring more complex algorithms, facilitating a comprehensive evaluation of classification approaches.

2.2 Data Processing

To implement the Multinomial Naive Bayes model, we began by preprocessing our data to ensure compatibility with the model's input requirements. The first step involved cleaning the raw text data by removing punctuation, extraneous symbols, links, and other noise. Additionally, we eliminated stopwords—common words such as "a," "an," "the," and "and" which often carry little semantic weight in text classification tasks. This preprocessing step was crucial for reducing noise and improving model accuracy. Following the cleaning process, we employed a Bag of Words (BoW) approach to vectorize the textual data. This technique involves creating a numerical representation of the text by converting each document into a vector based on the frequency of words in the corpus. So, for example, if our entire vocabulary were to be in the form ["the", "cat", "sat", "on", "mat", "is"], while our data's row contained the sentence "the cat is on the mat", then the vector created for this row would come out as [2, 1, 0, 1, 1, 1]. This step is essential for our model, as it not only numericizes the data, but also applies a form of one-hot encoding on our data which helps in the learning and training of our model.

2.3 Implementation

With the preprocessing and vectorization steps complete, we now proceed to implement the Multinomial Naïve Bayes model. The first step involves splitting the dataset into training and testing sets. A 70-30 ratio is used for this purpose, ensuring a sufficiently large dataset for training while reserving enough data for a robust evaluation during testing.

Once data splitting is complete, we move onto the model itself. In the training stage, we fit the Bag of Words vector for each row to its corresponding label, which represents the news categories in our data. After this step, we apply the Multinomial Naïve Bayes formula to the testing data. Specifically, we aim to find the class c that maximizes $P(c|x)$. This can be expressed mathematically as:

$$\hat{c} = \arg\max_c P(c) \prod_{i=1}^{n} P(x_i \mid c) \quad (3)$$

Here, x_i represents the i^{th} word of the sentence, and c denotes the class. We estimate $P(x_i \mid c)$ by counting the occurrences of the i^{th} word in sentences of class c, divided by the total number of words in sentences of class c.

To mitigate underflow errors and simplify computations, we take the logarithm, transforming the product into a sum:

$$\hat{c} = \arg\max_c \log P(c) + \sum_{i=1}^{n} \log P(x_i \mid c) \quad (4)$$

By applying this formula, we compute the predicted class for each row of input data, completing the classification process.

3 Softmax Text Classifier: Algorithm and History

3.1 Introduction

The Softmax Text Classifier is a machine learning model used for multi-class classification problems, particularly in the domain of text classification. It takes advantage of the softmax function, which transforms raw prediction scores (logits) into a probability distribution over classes. The model is often used for classifying text into different categories, such as sentiment analysis, topic categorization, and spam detection. The Softmax Text Classifier follows a standard procedure for training and predicting, which includes three key components: the Softmax function, Cross-Entropy Loss, and gradient-based optimization using Gradient Descent.

3.1.1 Initialization. The algorithm begins by initializing the following parameters:

- **Learning Rate**: Determines how much the weights are updated during each iteration.
- **Epochs**: Specifies the number of iterations the model will go through during training.
- **Weight Matrix (W)**: Randomly initialized with small values.
- **Bias Vector (b)**: Initialized as a zero vector.

3.1.2 Softmax Function. The softmax function is a mathematical function that converts a vector of raw prediction scores (logits) into a probability distribution over multiple classes. The formula for the softmax function is:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (5)$$

Where:

- z_i: The raw score (logit) for class i.
- The denominator: The sum of the exponentials of all logits, ensuring that the output values sum to 1, making them interpretable as probabilities.

3.1.3 Cross-Entropy Loss. To train the model, we use the cross-entropy loss function, which measures how well the predicted probabilities match the true labels. The cross-entropy loss is defined as:

$$\text{Loss} = -\sum_{i=1}^{m} y_i \log(\hat{y}_i) \quad (6)$$

Where:

- $\hat{y}_i$: The predicted probability for the true class of the i-th sample.
- y_i: The true label (1 if the sample belongs to the class, 0 otherwise).
- m: The number of samples.

3.1.4 Model Training (Gradient Descent). During the training process, the model learns by adjusting the weights to minimize the loss function using Gradient Descent. The gradients of the loss with respect to the weights (W) and bias (b) are computed as follows:

$$\frac{\partial L}{\partial z} = (\hat{y} - y) \tag{7}$$

$$\frac{\partial L}{\partial W} = X^T \cdot \frac{\partial L}{\partial z} \tag{8}$$

$$\frac{\partial L}{\partial b} = \sum \frac{\partial L}{\partial z} \tag{9}$$

The weights and biases are updated using the learning rate (η):

$$W := W - \eta \cdot \nabla W \tag{10}$$

$$b := b - \eta \cdot \nabla b \tag{11}$$

Where:

- η: The learning rate, which determines the step size of each update.
- ∇W and ∇b: Gradients of the loss with respect to weights and bias.

This process is repeated for a specified number of epochs, gradually improving the model's accuracy.

3.1.5 Model Prediction. Once the model has been trained, it can make predictions on new, unseen data. The prediction process involves the following steps:

- **Calculate raw scores**: Compute the raw scores (logits) for each class using the learned weights and bias.
- **Apply softmax function**: Transform the raw scores into a probability distribution over all classes.
- **Choose the predicted class**: Select the class with the highest probability as the predicted label.

3.1.6 Result Interpretation. The final output of the model is the class with the highest predicted probability. The model's predictions are mapped back to their respective class labels (e.g., "sports", "business") using a dictionary that associates class IDs with labels.

4 History of the Softmax Function and Text Classification

4.1 Origins of Softmax

The softmax function, first introduced in the 1950s, was developed as a generalization of the logistic function to handle multi-class classification problems. While the logistic function was widely used for binary classification tasks, there was a need for a method to assign probabilities to multiple classes. The softmax function addresses this by transforming raw scores into a probability distribution where the probabilities sum to 1.

The softmax function originated in the fields of decision theory and psychology. Its adoption in machine learning grew significantly in the 1980s, with the rise of logistic regression models and neural networks. These models required a method to handle multi-class outputs effectively, and softmax became the standard tool for converting network outputs into interpretable probabilities.

4.2 Text Classification

Text classification, the task of categorizing text into predefined categories, has a long history. Early efforts in the 1950s relied on rule-based systems and basic pattern recognition. In the 1990s, statistical methods and machine learning algorithms such as Naive Bayes and Support Vector Machines (SVMs) revolutionized text classification, automating the process and making it more scalable.

A major development in this period was the Bag-of-Words (BoW) model, which represented text as a vector of word counts or frequencies. This representation allowed statistical models to classify text effectively, addressing applications such as spam detection and sentiment analysis.

4.3 Softmax and Deep Learning

The rise of deep learning in the 2010s transformed text classification. Deep learning models such as Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) began to outperform traditional machine learning methods in areas like image recognition and natural language processing (NLP).

In these deep models, softmax played a critical role. By converting the raw outputs of a neural network into a probability distribution, softmax enabled accurate multi-class predictions. Deep learning-based text classification models became much more effective and efficient, powering applications such as sentiment analysis, topic categorization, and spam detection.

Today, the softmax function is integral to deep learning models used in text classification. Its ability to transform raw scores into meaningful probabilities is vital for advancing the capabilities of modern NLP systems.

5 Findings

(1) **Model Performance Metrics**
   (a) The Bernoulli Naive Bayes model achieved an accuracy of 95.61%, with precision, recall, and F1 scores all exceeding 95%. The high consistency across these metrics indicates the model's ability to generalize well and handle class imbalances effectively.
   (b) The Softmax Text Classifier achieved slightly lower metrics compared to the Bernoulli Naive Bayes model, with an accuracy of 94.57%, precision of 94.64%, recall of 94.57%, and an F1 score of 94.59%. While slightly lower, these results still demonstrate strong model performance.

(2) **Confusion Matrix Insights**
   (a) For both models, the confusion matrices revealed high classification accuracy across most categories, with very few misclassifications. This indicates that the models effectively learned patterns in the training data.
   (b) The Bernoulli Naive Bayes model showed fewer misclassifications overall, suggesting it might be better suited for this dataset.

(3) **Feature Representation**

(a) The Bag of Words approach successfully transformed textual data into numerical vectors. This representation allowed both models to process and classify the data effectively.
(b) Vocabulary generation was consistent across training and testing, ensuring no data leakage or inconsistencies in vectorization.

(4) **Strengths and Weaknesses**
(a) **Strength**: The Bernoulli Naive Bayes model's simplicity and efficiency made it highly effective for text classification tasks.
(b) **Weakness**: The Softmax Text Classifier required more computational resources and training time, yet it provided only slightly lower performance metrics.

(5) **Scalability and Generalization**
(a) Both models demonstrated strong potential for scalability to larger datasets, as they maintained high accuracy and balanced metrics without overfitting.
(b) The Softmax Classifier's implementation may require fine-tuning for larger datasets to avoid performance drops due to computational complexity.

6 Recommendations

(1) **Model Selection for Deployment**
(a) Given the higher performance metrics and lower computational requirements, the Bernoulli Naive Bayes model is recommended for deployment in scenarios requiring real-time or resource-constrained environments.
(b) The Softmax Text Classifier could be considered for scenarios where interpretability and probabilistic outputs are prioritized over slight differences in accuracy.

(2) **Optimization Strategies**
(a) For Naive Bayes: Explore feature selection techniques to reduce the dimensionality of the Bag of Words representation, which could further improve performance.
(b) For Softmax Classifier: Implement regularization techniques and optimize hyperparameters like learning rate and epochs to potentially enhance performance.

(3) **Data Enhancements**
(a) Enrich the dataset with more labeled examples to further strengthen the models' generalization capabilities.
(b) Consider using advanced text preprocessing techniques such as stemming, lemmatization, or TF-IDF to improve feature representation.

(4) **Scalability and Maintenance**
(a) Implement a validation set to monitor performance during model training and ensure stability across iterations.
(b) Plan periodic retraining of the models to accommodate changes in data patterns over time.

(5) **Future Development**
(a) Experiment with more complex models like Transformer-based architectures (e.g., BERT) for comparison against simpler models.
(b) Evaluate alternative feature extraction techniques, such as word embeddings, to capture semantic relationships within the text.

7 Conclusion

The study demonstrates the successful implementation and evaluation of two text classification models, Bernoulli Naive Bayes and Softmax Text Classifier. The Bernoulli Naive Bayes model outperformed the Softmax Classifier in accuracy and computational efficiency, making it the preferred choice for most practical applications. However, the Softmax Classifier's slightly lower metrics indicate its potential for tasks requiring a more nuanced probabilistic interpretation.

Both models effectively utilized the Bag of Words approach to represent text data, showcasing the strength of feature engineering in natural language processing. The findings underscore the importance of balancing model complexity and performance to meet specific application needs. Future work can focus on expanding the dataset, exploring advanced preprocessing methods, and experimenting with state-of-the-art models to further enhance classification accuracy and robustness.