Classification of Urdu News Articles: Challenges and Insights in a Low-Resource Language

Ahmed Wali
26100264@lums.edu.pk

Muhammad Ismail Humayun
25020267@lums.edu.pk

Hamza Sherjeel
26100361@lums.edu.pk

Yamsheen Saqib
26100379@lums.edu.pk

Shanzay Omer
26100202@lums.edu.pk

Abstract

This study addresses the classification of Urdu news articles into five categories: Entertainment, Business, Sports, Science-Technology, and International. Leveraging a dataset scraped from seven major Urdu news websites, we employed a robust methodology encompassing data preprocessing, vocabulary curation, feature transformation, and the implementation of three machine learning models: K-Nearest Neighbors (KNN), Naïve Bayes, and a custom-designed neural network.

Our approach incorporates visualization techniques like PCA to analyze data distributions and ensures a comprehensive evaluation of model performance. We also present two variations of our models, demonstrating how they generalize to lengthy articles. Furthermore, we explore the limitations posed by the low-resource nature of the Urdu language, applying embeddings to address these challenges. However, our findings indicate that embeddings alone did not significantly enhance performance, underscoring the urgent need for further development and research in this domain.

Keywords

Urdu, Classification, Machine Learning

ACM Reference Format:
Ahmed Wali, Muhammad Ismail Humayun, Hamza Sherjeel, Yamsheen Saqib, and Shanzay Omer. 2024. Classification of Urdu News Articles: Challenges and Insights in a Low-Resource Language. In *Proceedings of Fall 2024 (CS 437 Machine Learning Group Project).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The primary objective of this project was to develop an effective machine learning model for classifying Urdu articles into five predefined categories: Entertainment, Business, Sports, Science-Technology, and International. A corpus of 5841 articles was collected from various sources, processed, and analyzed to train and evaluate multiple machine learning models. The project involved extensive preprocessing, feature transformation, and experimentation with three machine learning models: K-Nearest Neighbors (KNN), Naïve Bayes, and a Neural Network implemented from scratch. This report details the methodology employed, the findings of the study, and an evaluation of the strengths and limitations of each model.

CS 437 Machine Learning Group Project, LUMS,

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

2 Methodology

The dataset consisted of 5841 Urdu articles sourced from ARY, Dunya, Express, Geo, Jang, BBC, and Dawn. After preprocessing and removing null entries, the corpus was reduced to 5841 articles. The vocabulary contained 51,074 unique Urdu words, which were sorted by frequency. Annotators manually labeled stopwords and non-meaningful words, supported by a list of 517 Urdu stopwords from Kaggle. The final vocabulary, comprising 49,869 meaningful words, was stored in a JSON file. A Cohen's Kappa score of 0.664 confirmed the reliability of the annotations.

2.1 Data Collection

We collected data from diverse and credible Urdu news sources to build a robust and representative dataset for training and evaluation purposes. The training dataset, comprising data from ARY, Dunya News, Express, Geo, and Jang, totals 5943 rows with a combined size of 18.75 MB. The label distribution across five categories—Sports, Science-Technology, International, Entertainment, and Business—ensures balanced representation for effective model training. Additionally, two external test datasets were curated to simulate real-world scenarios: the Dawn dataset, which closely resembles the training distribution, and the BBC dataset, characterized by longer articles and a more varied distribution. The Dawn dataset contains 265 rows and is 1.34 MB in size, while the BBC dataset consists of 1177 rows and has a file size of 12.14 MB. Together, these datasets enable comprehensive evaluation and benchmarking of our models. For sources like ARY, where direct scraping was challenging, we utilized a custom script with randomized delays to collect data, ensuring authenticity and compliance. All datasets and corresponding scripts are included in the data/raw-datasets folder to ensure transparency and reproducibility.

The combined training dataset consists of 5943 rows, distributed as shown in Table 2.

We also created two external test datasets to simulate real-world scenarios:

- **Dawn Dataset:** This dataset closely resembles the training dataset distribution. It contains 265 rows and is 1.34 MB in size. The label distribution is summarized in Table 3.

Table 1: Summary of Training Dataset

Source	File Size (MB)	Rows	Label Distribution
ARY	13.59	4526	Science-Technology: 1220, Entertainment: 957, International: 910, Business: 720, Sports: 719
Dunya News	0.01	42	International: 18, Sports: 12, Business: 6, Entertainment: 6
Express	0.99	500	Entertainment: 100, Business: 100, Sports: 100, Science-Technology: 100, International: 100
Geo	0.72	379	International: 199, Sports: 60, Entertainment: 60, Business: 60
Jang	3.41	496	Sports: 496

Table 2: Combined Training Dataset Summary

Label	Count
Sports	1387
Science-Technology	1320
International	1227
Entertainment	1123
Business	886
Total Rows	**5943**
File Size (MB)	**18.75**

- **BBC Dataset:** This dataset differs in distribution and contains longer articles, ideal for evaluating model generalization. It has 1177 rows and is 12.14 MB in size. The label distribution is summarized in Table 4.

Table 3: Dawn Dataset Summary

Label	Count
International	170
Business	69
Science-Technology	23
Entertainment	3
Total Rows	**265**
File Size (MB)	**1.34**

Scraping Strategy. For sources like ARY, where scraping directly was challenging, we first scraped article links and then scripted the data collection process with random delays to mimic human visitors. All datasets and code are included in the 'data/raw-datasets' folder for reproducibility.

Table 4: BBC Dataset Summary

Label	Count
Entertainment	240
Sports	240
Science-Technology	240
International	234
Business	223
Total Rows	**1177**
File Size (MB)	**12.14**

2.2 Data Cleaning

To clean the data, we followed a structured process involving multiple steps. The first step was to remove irrelevant or unnecessary data points, such as non-Urdu words, multiple spaces, and punctuation marks. After this, we processed the data to generate a clean, usable vocabulary for further analysis. Below is a summary of the data cleaning steps:

2.2.1 Removing Non-Urdu Words and Multiple Spaces. We started by cleaning the data and dropping missing values. We removed non-Urdu words, extra spaces, punctuation marks, and converted multiple spaces into single spaces.

Table 5: Data Cleaning Summary (Step 1)

Action	Result
Initial Row Count	5943
Rows After Dropping Missing Values	5841
Rows Dismissed Due to Missing Values	102
Final Row Count After Cleaning	5841
Unique Labels in 'gold_label'	5
Counts per Label	Sports: 1375 Science-Technology: 1285 International: 1188 Entertainment: 1117 Business: 876

2.2.2 Vocabulary Creation. Next, we built the vocabulary by processing the cleaned dataset. This step resulted in a vocabulary JSON file containing 51,071 unique Urdu words. The total number of words processed, including duplicates, was 2,164,726.

Table 6: Vocabulary Summary (Step 2)

Action	Result
Unique Words in Vocabulary	51,071
Total Words Processed (Including Duplicates)	2,164,726

2.2.3 Top 10 Most Common Words. The vocabulary was then saved as a JSON file in the data/eda directory for later use.

2.2.4 Stopwords Identification. Next, we identified 517 stopwords from a Kaggle dataset for Urdu stopwords. The dataset used for stopword extraction can be found at [LINK A]. After applying the stopwords, we cleaned the dataset further, ensuring the text was free from common, non-informative words.

The final vocabulary contained 51,074 unique Urdu words after removing stopwords.

2.3 Stopwords Annotation

Since we suspected that there could be many other stopwords that might contribute to our dataset—considering the Kaggle dataset was limited to 517 words—we decided to manually annotate additional stopwords. To begin, we ordered the vocabulary by frequency and randomly removed 200 words to calculate Cohen's Kappa and introduce overlaps between the annotators.

We then divided the corpus amongst two annotators by assigning alternate words to each, asking them to flag words as either meaningful, not meaningful, or stopwords.

To make the annotation process easier and error-free, we developed a software tool for the annotators to quickly and accurately annotate the words. Below is a dummy image of the tool interface:

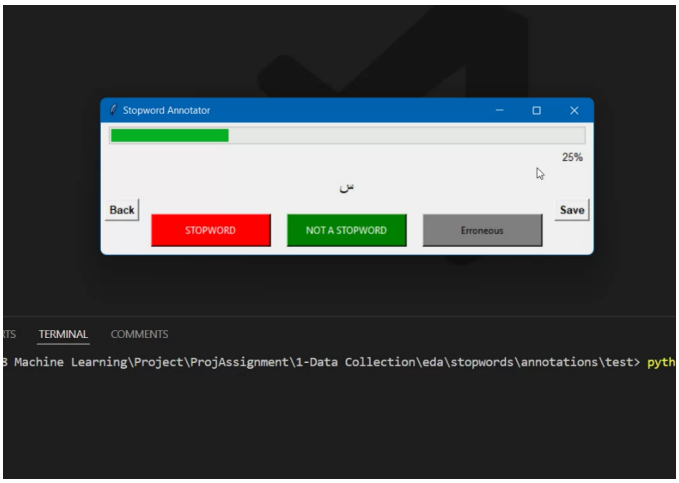

Figure 1: Annotator's Software Tool for Stopwords Annotation

After gathering the annotations, we obtained the following distributions for the two annotators:

All the annotations can be found in the 'data/eda' folder, which is included with this report.

2.3.1 Estimating Cohen's Kappa for Inter-Annotator Agreement. Since we had 200 overlapping words for annotation, we estimated Cohen's Kappa to assess inter-annotator agreement.

Overlapping Words: 200

Annotator 1: 199 words labeled as '0' (not meaningful), and 1 word labeled as '1' (meaningful). **Annotator 2:** 198 words labeled as '0' (not meaningful), and 2 words labeled as '1' (meaningful).

Agreement and Disagreement in Overlap:

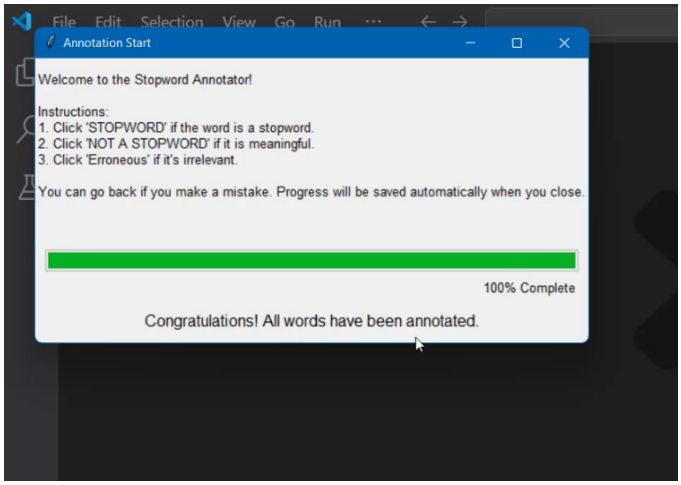

Figure 2: Annotator's Software Tool for Stopwords Annotation

Table 7: Stopwords Annotation Distribution

Annotator	Stopwords	Not Meaningful	Meaningful
Annotator	1		
	242		
		145	
			24933
Annotator	2		
	661		
		38	
			24936

- Agreement on '0': Both annotators labeled 198 words as '0'. - Agreement on '1': Both annotators labeled 1 word as '1'. - Disagreement: Annotator 2 labeled 1 additional word as '1', but Annotator 1 labeled it as '0'.

Agreement Breakdown:

$$\text{Agreement: } 198 \text{ (on 0)} + 1 \text{ (on 1)} = 199$$

$$\text{Total disagreements: } 200 - 199 = 1$$

1. Observed Agreement (p_0):

$$p_0 = \frac{\text{Number of agreements}}{\text{Total number of overlapping words}} = \frac{199}{200} = 0.995$$

2. Expected Agreement (p_e):

To calculate the expected agreement, we use the marginal probabilities.

Marginal Probabilities:

For Annotator 1:

$$p_1(0) = \frac{199}{200} = 0.995, \quad p_1(1) = \frac{1}{200} = 0.005$$

For Annotator 2:

$$p_2(0) = \frac{198}{200} = 0.99, \quad p_2(1) = \frac{2}{200} = 0.01$$

Expected Agreement (p_e):

$$p_e = (p_1(0)\cdot p_2(0)) + (p_1(1)\cdot p_2(1)) = (0.995\cdot 0.99) + (0.005\cdot 0.01) = 0.9851$$

3. Cohen's Kappa (κ):

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{0.995 - 0.9851}{1 - 0.9851} = \frac{0.0099}{0.0149} \approx 0.664$$

Summary of Results:

- Observed Agreement (p_0): 0.995 - Expected Agreement (p_e): 0.9851 - Cohen's Kappa (κ): 0.664

Interpretation: A Cohen's Kappa value of 0.664 indicates substantial agreement between the annotators after accounting for chance.

2.4 What does the data say about the model

To visualize the dataset, we had to get into numbers. For that, we used the 'fasttext.cc' embedding model called `cc.ur.300.vec.gz` in its vector form. It takes a sentence and converts it into a 300-dimensional embedding, preserving the semantic meaning of the sentence. After running the embeddings on the text, we then performed PCA and t-SNE analysis of the data.

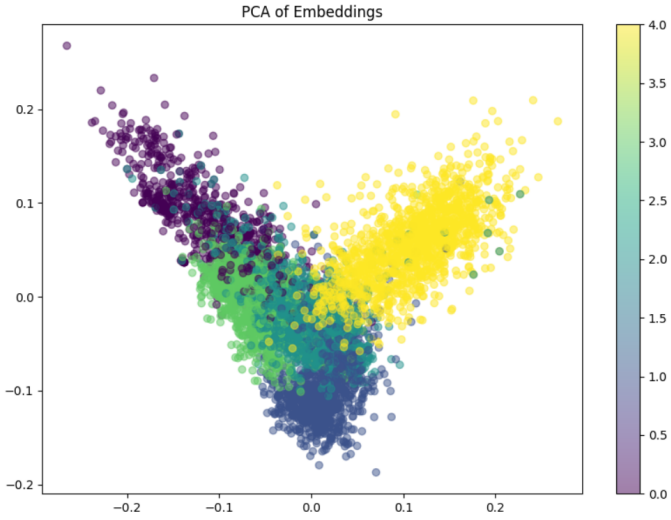

Figure 3: PCA Analysis of the Dataset

As can be seen in the images, the PCA analysis shows visible features across which the words can be grouped together. The t-SNE analysis clearly shows clusters, thus giving us the opportunity to use nearest neighbor methods to cluster the words effectively. Later, we will demonstrate that nearest neighbor methods are actually the best ones for this task.

2.5 Feature Transformation

Two types of feature extraction methods were employed in this project. The first was a Count Vectorizer, which encoded each article into a sparse matrix by marking word occurrences. The second was a TF-IDF vectorizer, which captured the importance of words across documents. For neural networks, embeddings from FastText were used to represent words in a dense vector space.

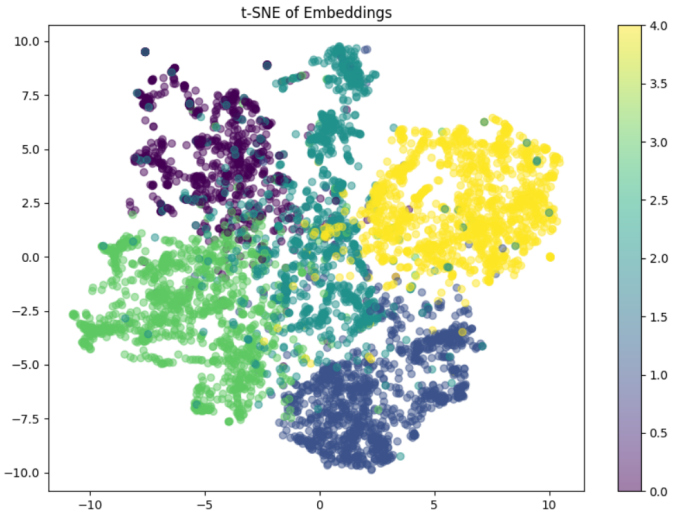

Figure 4: t-SNE Analysis of the Dataset

2.6 Model Selection

Three machine learning models were implemented and evaluated. Each model utilized a distinct methodology for feature representation and classification.

2.6.1 K-Nearest Neighbors (KNN). The KNN model employed a TF-IDF vectorizer, leveraging unigrams, bigrams, and trigrams for feature extraction. Cosine similarity was used as the distance metric. Hyperparameter tuning identified $k = 4$ as the optimal value. The model excelled in classification tasks, particularly on validation data, but faced challenges in generalizing to datasets with a different distribution.

2.6.2 Naïve Bayes. The Naïve Bayes model utilized unigram features to estimate word probabilities within each category. Despite its simplicity, the model provided a strong baseline. It was effective for validation data but struggled with domain adaptation when tested on external datasets like BBC and Dawn.

2.6.3 Neural Networks. A custom neural network was developed using NumPy, demonstrating the potential of handcrafted models. It included hidden layers and used activation functions. Features were extracted using both Count Vectorizers and pre-trained FastText embeddings. While the neural network showed strong performance on the validation dataset, its computational cost and limitations in handling unseen data distributions were significant drawbacks.

3 Findings

The results revealed that KNN achieved the best overall performance on the Dawn test set, with an accuracy of . The neural network outperformed other models on the validation dataset, reaching an accuracy of . However, all models struggled with the BBC test set,

which had a distribution differing significantly from the training data. Notably, the "International" category consistently exhibited lower precision and recall across all models, likely due to label overlap in the training data.

3.1 KNN

We have trained two models to see how well the KNN generalizes to the data. We first cleaned the data, implemented our own `CustomTfidfVectorizer` and the KNN class, and ran the analysis. The vectorizer took the sentences and converted them into sparse vectors in order of the vocabulary, and the `fit` method in the KNN performed the dot product to estimate the cosine distances.

Note: The best k value thus established is $k = 6$, giving both the highest accuracy on the *test set* at around 93% and the least error.

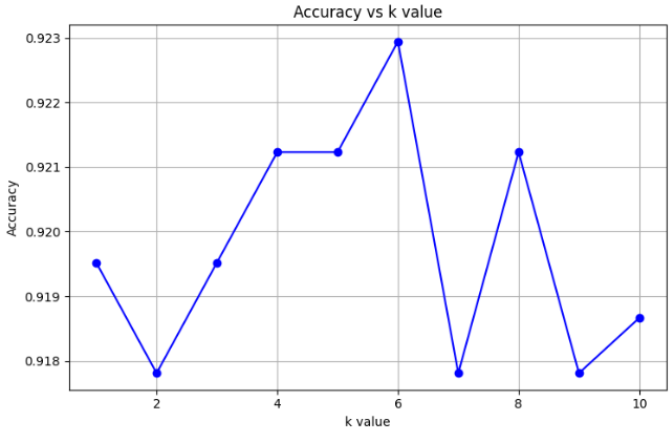

Figure 5: KNN Accuracies on the Test Set

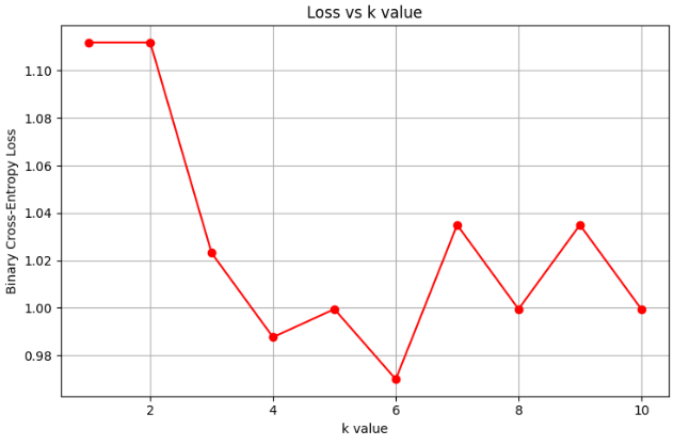

Figure 6: KNN Loss on the Test Set

Best k value: 6 with accuracy: 0.9229 on the internal test set.

External test accuracy on the Dawn dataset: 0.8162.
External test accuracy on the BBC dataset: 0.6211.
These were only conducted to assess how well the model performed in relation to real-world data.

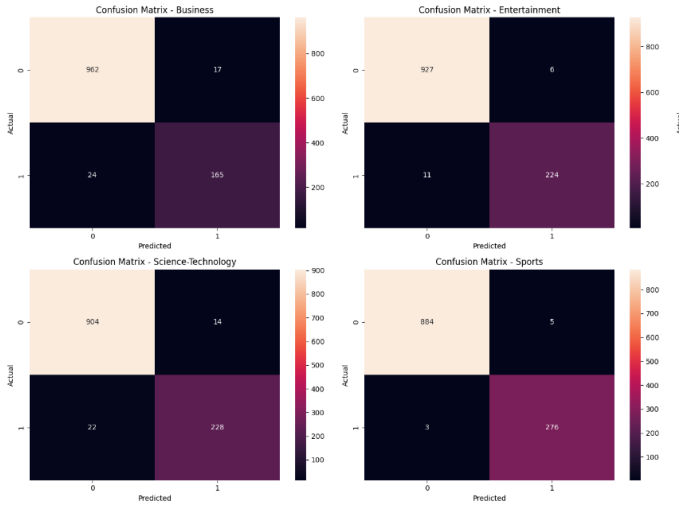

Figure 7: Confusion Matrix on the Internal Test Set

3.2 Improving the KNN Model

After all that, we trained the KNN model on the combined distributions of both datasets (7000 samples) and estimated the external test accuracy on the BBC dataset to be:

Best k value: 4 with accuracy: 0.8685.

3.3 Naive Bayes

We trained two variants of the Naive Bayes model: one using only the internal dataset and another trained on the combined dataset, which included both internal and external data. For both variants, we implemented a custom classifier from scratch.

3.3.1 Internal Dataset Model. For the Naive Bayes model trained solely on the internal dataset, the accuracy on the test dataset was found to be:

Accuracy on Test Dataset:0.9324

External Test Set Results:

- Dawn dataset: 0.8162
- BBC dataset: 0.61498

The model worked best when using unigrams as the features for training.

3.3.2 Combined Dataset Model. For the Naive Bayes model trained on the combined dataset, the accuracy on the test dataset was found to be:

Accuracy on Test Dataset:0.8782

This model also worked best with unigrams as the features.

3.4 Neural Networks

We experimented with three flavors of the neural network model: one using the internal test data, one using the combined dataset, and one using embeddings on the internal test data in hopes of improving performance.

For both the internal and combined models, we used a simple count vectorizer. The architecture consisted of two hidden layers with 256 and 128 neurons each, along with an input and output layer. The batch size was set to 32, and the total number of epochs was 250. The model was built completely from scratch using numpy.

3.4.1 Internal Dataset Model. For the neural network trained on the internal dataset:

- Epoch 240, Accuracy: 0.9679 (training accuracy)
- Neural Network Accuracy: 0.875 (suggests slight overfitting)
- Dawn dataset: Accuracy on new dataset: 0.7094
- BBC dataset: Accuracy on new dataset: 0.5322

3.4.2 Combined Dataset Model. For the neural network trained on the combined dataset:

- Epoch 240, Accuracy: 0.9628 (training accuracy)
- Neural Network Accuracy: 0.8208
- BBC dataset: Accuracy on new dataset: 0.6847
- Dawn dataset: Accuracy on new dataset: 0.8803

3.4.3 Model with Embeddings. For the model using embeddings, we utilized the fasttext.cc embedding model that was used earlier in the PCA analysis. We hoped that these embeddings, which capture the semantic essence of words, would improve performance.

- Epoch 490, F1 Score: 0.9919
- Neural Network Accuracy: 0.8965
- BBC dataset: Accuracy on new dataset: 0.5692

Despite using embeddings, the performance only showed a 2% increase compared to the previous models. This result is further discussed in the limitations section.

3.5 Other Models

These models were tested to compare their performance against our models. The models included Random Forest, Logistic Regression, and Naive Bayes.

3.5.1 Random Forest. The Random Forest model was trained using a count vectorizer with 100 trees and maximum depth. On the training set, the model achieved an accuracy of 0.8349. However, it performed poorly on the BBC external dataset, with an accuracy of only 0.0127.

- Random Forest Accuracy on Train Set: 0.8349
- Random Forest Accuracy on BBC External Dataset: 0.0127

3.5.2 Logistic Regression. The Logistic Regression model achieved an internal test accuracy of 0.9273, but it struggled to generalize to the external BBC dataset, where the accuracy dropped to below 12%.

- Logistic Regression Accuracy on Internal Test Set: 0.9273
- Logistic Regression Accuracy on BBC External Dataset: 0.12

3.5.3 Naive Bayes. Considering the performance of the other models, the winner is Naive Bayes. It achieved an accuracy of 87.5% on the external test dataset and around 92% accuracy on the internal test dataset, outperforming the other models.

- Naive Bayes Accuracy on Internal Test Set: 0.92
- Naive Bayes Accuracy on External Test Set: 0.875

4 Limitations and Conclusion

This study demonstrated the effectiveness of machine learning models in classifying Urdu articles. While the models performed well on the validation and Dawn datasets, their accuracy dropped significantly on the BBC dataset, highlighting challenges in domain adaptation. The imbalanced dataset and potential mislabeling in categories, particularly "International," also contributed to lower performance.

Future work could explore advanced neural architectures such as transformers and transfer learning with pre-trained Urdu language models. Additionally, increasing dataset diversity and addressing label ambiguities could further enhance model performance.

5 Future Considerations

One of the key challenges we encountered during this project was that the currently available embeddings could not generalize well to the model. This indicates that state-of-the-art embedding models are not yet fully developed for the Urdu language. This limitation significantly impacted the model's performance, especially in terms of generalization. As a result, there is a need for immediate action to create and refine embedding models tailored specifically for resource-rich languages like Urdu. Such advancements are crucial to enhance the performance of machine learning models in these underrepresented languages.

Further such projects are required to validate the hypothesis.

Acknowledgments

To Robert, for the bagels and explaining CMYK and color spaces.

References

@miscrtatman2019urdu, author = T. Atman, title = Urdu Stopwords List, year = 2019, howpublished = https://www.kaggle.com/datasets/rtatman/urdu-stopwords-list, note = Accessed: 8-Dec-2024

@miscbojanowski2017fasttext, author = J. Bojanowski and P. Grave and A. Mikolov and A. Joulin and T. Mikolov, title = Enriching Word Vectors with Subword Information, year = 2017, howpublished = https://fasttext.cc/docs/en/crawl-vectors.html, note = Accessed: 8-Dec-2024