Pashto Poetry Attribution using Deep Learning Techniques

Ali Muhammad Asad
Computer Science
Habib University
Karachi, Pakistan
aa07190@st.habib.edu.pk

Shayan Shoaib Patel
Computer Science
Habib University
Karachi, Pakistan
sp07101@st.habib.edu.pk

Sadiqah Mushtaq
Computer Engineering
Habib University
Karachi, Pakistan
sm07152@st.habib.edu.pk

Lyeba Abid
Computer Engineering
Habib University
Karachi, Pakistan
la07309@st.habib.edu.pk

Dr. Abdul Samad
Associate Professor, CS
Habib University
Karachi, Pakistan
abdul.samad@sse.habib.edu.pk

Sandesh Kumar
Computer Science
Habib University
Karachi, Pakistan
sandesh.kumar@sse.habib.edu.pk

I. Introduction

Poetry has been a cornerstone of cultural and literary heritage across the globe, as it is one of the most popular ways to express emotions, thoughts, and experiences. Pashto is one of the major languages spoken in Afghanistan and Pakistan, with 50% of the Afghan population speaking Pashto, and 15% of the Pakistani population speaking Pashto [1] [2]. As such, it also holds a significant place in the rich tapestry of South Asian literature with its first recorded poetic works believed to be dating back to the 8th century with the works of Amir Kror Suri (a warrior poet) [3]. There have also been other notable poets such as Khushal Khan Khattak, Rahman Baba, Ghani Khan, and Hamza Baba who have contributed to the Pashto literary tradition, and are revered for their works. However, despite its rich history and cultural significance, Pashto remains a low-resource language in the context of computational linguistics, with limited availability of annotated data and tools for natural language processing (NLP) apart from translated works. This gap poses challenges for preserving and promoting Pashto literature on a global scale. Thus, the attribution of poetic works to specific poets can sometimes be ambiguous, due to oral traditions, and lack of proper documentation.

In recent years, deep learning techniques have gained substantial traction in the field of language processing, text classification, and poet attribution. Poet attribution refers to the process of identifying the poet who authored a specific poem. This process is crucial for preserving the cultural heritage of a language, as well as for academic research and literary studies. While researchers have explored poetry attribution in various languages such as English, Hindi, Urdu, Arabic, Persian, and Gujrati, Pashto poetry has not been extensively studied in this context. This gap could also be attributed to the lack of resources, and documentation in Pashto. This paper aims to bridge this gap by introducing deep learning based approaches to classify Pashto poetry and attribute it to specific poets, which can also prevent misinformation and mis-attribution due to the lack of proper documentation available. This paper also aims to curate the first Pashto poetry dataset of 13 poets (old and relatively newer poets) with about 12000 couplets in total, which will be made publicly available to the research community.

II. Related Work

Poet attribution using deep learning and machine learning has been explored in various languages previously as well.

A study on poet attribution for Urdu Ghazals [4] employed machine learning, deep learning, and transformer-based techniques on a dataset of 18,609 couplets from 15 notable Urdu poets, covering both the 18th-19th and 19th-20th centuries. The distinct word usage and styles across these periods benefited the model in identifying patterns. The study tested four machine learning models — Logistic Regression, Random Forest, Naive Bayes, and SVM — using label encodings and TF-IDF for feature extraction. They also experimented with four deep learning models: Flatten, LSTM, GRU, and 1D-CNN, utilizing one-hot encoding and tokenization with a 300-dimensional embedding layer. The best performance came from transformer models: Bert and roBerta achieved 80.32% and 79.52% accuracy, respectively, outperforming deep learning methods. Among traditional machine learning models, SVM and Logistic Regression achieved 64% and 60% accuracy, surpassing deep learning approaches, where LSTM performed best at 59.96%. The study notes class imbalance as a challenge but emphasizes the superior performance of transformer models in poet attribution tasks.

Another study was conducted on the classification of Persian poetry based on the poet's era [5]. The authors used the Persian Hafez's Ghazal dataset which contained 496 Persian Ghazals. The authors categorized Persian Ghazals based on the poetic era in which they were written, with the classification focusing on distinguishiing Hafez's ghazals

by the time period or era associated with the poetry using sequential learning architectures. Word embeddings were used, an important technique in NLP, and deep learning classification models. In addition, they also used Bag Of Words (BOW) - a baseline statistical model for textual feature extraction, and Latent Dirichlet Allocation (LDA) for feature extraction. They also used Distributed Memory (DM) for vectorization, and concatenated Distributed BOW with DM. Then they trained Machine Learning (ML) and Deep Learning (DL) models including Random Forest, Logistic Regression, LSTMs, GRU, and Bi-LSTM. For evaluation, they used accuracy, F1 score, precision, and recall to evaluate the performance of their models. The ML models were compared to SVM, which performed better in terms of accuracy but not F1 score. In the DL models, LSTM gave highest accuracy of 77.8% and the highest F1-score of 76.6% on the Persian dataset.

We also came across a study that classified Gujrati Poetry based on emotions using deep learning techniques [6]. The author collected the first Gujrati poetry dataset of more than 300 different Gujrati poems, and called it the "Kavan" dataset that represented the different "Rasa's" emotions. They further used the NLTK library in Python for tokenization and labelling of the data. Poems were used as input one by one, with each word compared to a metadata based on the "Navarasa" concept, returning values between 0 and 8 for the emotions. They implemented a Deep Learning classifier model, and achieved an 87.62% accuracy.

In addition to classification and attribution, another study aimed to perform a comparative analysis on various machine learning models for Urdu poet attribution [7]. They collected a total of 1563 poems from 5 famous Urdu poets. They also tokenized the data using Python's NLTK library, however, they did not remove stop words as they believed that such functions are important in modeling the style of the author. They repeated various experiments on various models repeatedly with different parameters, to conclude that SVM performed the best over most configurations with the highest accuracy of 88.7% and an average accuracy of 81%, with Naive Bayes at a close second with the highest accuracy of 84% and an average accuracy of 77%. They also implemented LSTM models, however, the highest accuracy they received was 45.78%. They concluded that with over 100 samples per poet and more than 2 lines per sample, one could achieve good results in poet attribution.

Regarding Urdu language we found another research on the poet attribution on urdu authorship of poets [8]. They collected a total of 11406 couplets from nonsocial Urdu websites of mainly 3 different Urdu poets. They developed various models for classification including SVM, Multilayer Perceptron (MLP), Multinomial Naive Bayes, and Multinomial MLP Pre-Trained Word2Vec model. SVMs achieved the highest accuracy and F1-score of 82.85% and 82.67%.

Another similar work was done on identification of Urdu Ghazals using SVM [9]. The authors collected a total of 3967 couplets from 4 poets, and generated a total of one million tokens from the dataset with a vocabulary size of 6427. They also created a Term-Document Frequency (TDF) matrix with rows representing couplets and columns representing unique terms. For feature selection, they used Chi-Square and L1-based techniques to select the best features. The Chi-Square evaluates feature importance based on statistical independence, while L1-based selection focuses on non-zero regression coefficients. They selected a total of 5 models; Naive Bayes, Decision Trees, SVMs, KNNs, and Random Forest. Overall, SVMs performed the best out of all the models with an accuracy of 72%, closely followed by Naive Bayes with an accuracy of 70%.

We also found a study that used ML approaches for authorship attribution in Arabic poetry [10]. They established various characteristic of Arabic poetry such as *Meter (Wazn)* and *Rhyme (qafiya)*, and curated a corpus of Arabic poetry of 73 poets with 18646 Qasidah's for training that amount upto 1856436 words. They used SVMs and Naive Bayes for classification, using Chi-Square and Information Gain for feature selection. They came up with 6 features corresponding to character, word length, sentences length, first word length, meter, and rhyme. They achieved the highest accuracy of 98.86% on SVMs when all features were used, while on average SVM had an accuracy of 87.4%. They also implemented a Naive Bayes model which had the highest accuracy of 98.86% as well with only two features, and an average accuracy of 90.68%.

Apart from the aforementioned literature, various other works were also consulted that were relevant to our research. One study we came across was the first to work on Pashto text classification using language processing techniques for single and multi-label analysis [11]. They constructed a dataset of Pashto documents including Sports, History, Health, Scientific, Cultural, Economic, Political, and Technology based - although poetry wasn't included. They used DistilBERT-base-multilingual-cased model, Multilayer Perceptron, SVMs, KNNs, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Random Forest, and Logistic Regression models. They got a 94% accuracy using MLP classification and TFIDF feature extraction. DistilBERT - a multilingual model which was not pretrained on Pashto still was able to achieve 66.31% accuracy, thus, the authors conclude that this still gave promising results, and that the model could be further improved by developing a tokenizer specifically tailored for Pashto.

III. Methodology

A. Dataset

Since no existing dataset for Pashto Poetry existed, we collected our dataset from scratch. A major challenge in the data collection aspect was the lack of digitalization and proper documentation on Pashto poets. While we were able to find some poetry in blogs and social sites, most of the poetry had been translated, pdf forms, or in apps for more famous poets such as Rahman Baba, Hamza Baba and Khushal Khan Khattak. In addition, poetry was also

found as just a couplet or two in the form of images on social media. Thus, our data collection included searching for poetry online through blogs, articles, and sites including Hamari Web, Wordpress, and Rekhta. We also collected pdf books of poets that we passed through an OCR and then cleaned out. Since there were also mobile apps available for certain Pashto poets, we extracted from mobile apps and converted it into textual form, or took screenshots of the poetry and passed it through the OCR to get the text.

The collected data was then passed through an extensive data cleaning pipeline which involved both manual and automated cleaning. The data was first cleaned manually by removing such texts which could not be automated such as prefaces or index from books, headers or footers, authors notes, footnotes, and other such irrelevant text. Since that text was also in the same script, automating this was impossible, hence first, the data was manually inspected for such discrepencies. Then the data was passed through a data cleaning pipe where first the english characters, and other irrelevant symbols were removed including numbers, special characteres, etc, followed by removing those lines where only 1 or 2 words were present as they were mostly the titles of the poems (this was also extensively cleaned manually before), followed by removing the duplicate entries in the data, followed by removing the empty lines. Then a script would return the number of couplets in the data. The image given below describes the complete data pipeline.

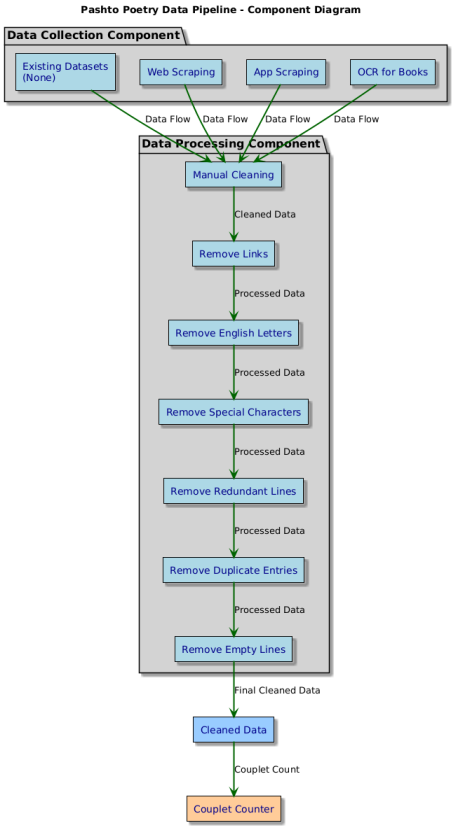


Figure 1: Data Cleaning Pipeline

All the data was then cleaned and sorted to get just the Pashto poetry of each poet. All in all, we were able to collect 27,607 couplets from a total of 23 poets. The chosen poets also show a good distribution in the sense that some poets are of historic ages such as Rahman Baba, Khushal Khan Khattak, and Hamza Baba, while there are also recent poets such as Ghani Khan, and Khatir Afridi. Thus, the poetry would have varying styles, themes, word usage, dialects, and vocabulary which would help in training a robust model as the patterns would be more diverse.

The table below shows the number of couplets collected for each poet.

No.	Poet/Shayari	Couplets Count
1.	Abbasin Yousuf	1875
2.	Ajmal Khattak	780
3.	Allama Abdul Hai	2380
4.	Aziz Mazerwal	30
5.	Ghani Khan	1399
6.	Hamza Baba	1177
7.	Javed Ahmedzai	561
8.	Karan Khan	979
9.	Khaliq Ziari	40
10.	Khatir Afridi	1436
11.	Khushal Khan Khattak	4257
12.	Matiullah Turab	1192
13.	Mumtaz Orakazi	1645
14.	Munir Jan	1169
15.	Naeem Ahmed	856
16.	Rabia Mumtaz	572
17.	Rahman Baba	2202
18.	Rehmat Shah	1534
19.	Sahib Shah Sabir	1051
20.	Shabbir Khan Durrani	1056
21.	Shakir Orakzai	569
22.	Salim Riaz	232
23.	Shoaib Khan Khattak	615
Total		**27,607**

Table I: Poets and their couplets count

We also decided to remove 3 of the poets for our models with the lowest poet count - mainly Aziz Mazerwal, Khaliq Ziari, and Salim Riaz. The final dataset that we trained our models on consisted of 20 poets then.

B. Machine Learning Models

From our extensive literature review, we found that machine learning models have been widely used for classification and attribution tasks. Thus, we decided to employ several machine learning models to classify Pashto poetry. Our main machine learning pipeline includes first label encoding the dataset, followed by extracting features from the text data based on TF-IDF Feature Extraction. TF-IDF Feature Extraction works by first tokenizing the text data, followed by converting the text data into a matrix of token counts, and then normalizing the token counts. The TF-IDF value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. This helps in reducing the importance of common words in the text data. For our models, we extracted 10000 features. The TF-IDF vectors were then split into 80-20 train-test split. Since our dataset was heavily

imbalanced, we decided to go for a weighted train-test split, where the train-test split was done in such a way that the distribution of the classes in the train and test set was the same, thus, each class had the same ratio of samples in the train and test set. Finally, those TF-IDF vectors were then passed through several machine learning models including Random Forest, Support Vector Machine, Logistic Regression, and XGBoost, as they all were supported by the literature. The diagram below explains the complete machine learning pipeline.

Figure 2: Machine Learning Pipeline

We used python's "Pandas" library for creating the dataframes, and for labelling the data for the 20 poets. We defined our own weighted train-test split function to ensure that the distribution of the classes in the train and test set was the same. Python's "TfidfVectorizer" was used for the TF-IDF feature extraction. Lastly, we used the "Scikit-learn" library for the machine learning models, and the "XGBoost" library for the XGBoost model.

1) Random Forest: Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. Random Forest is a bagging technique and is an ensemble of decision trees. We trained our Random Forest model on 100 esimators, with no max-depth defined. We tried hyperparameter tuning of the Random Forest model as well with various parameters, however, they did not improve the model performance.

2) Support Vector Machine: Support Vector Machine is a supervised machine learning algorithm that can be used for both classification or regression challenges. From the literature, we found that SVM has been widely used for text classification tasks. We experimented with the linear kernel, rbr kernel with gamma set to scale and C set to 1 as this was the best performing hyperparameters for our dataset.

3) Logistic Regression: Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. We used the Logistic Regression model on 1000 iterations, we also implemented a liblinear solver, and a newton-cg solver with a C value of 10.

4) XGBoost: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. We used the XGBoost model with 100 n estimators, a max depth of 3, a learning rate of 0.1, gamma set to 0.1, eval metric set to logloss, and a scale pos weight of 1, as this was the best performing hyperparameters for our dataset.

C. Deep Learning Models

We also experimented with deep learning models for Pashto poetry classification. Deep Learning models are inspired by the structure and function of the brain, in which the neural networks work in a similar way to the human brain, updating and learning their weights and biases based on the input data. As such, deep learning models have been quite powerful, and robust in their ability to learn complex patterns in the data. Therefore, they have also been widely used for text classification tasks.

We used the Keras library with TensorFlow backend for our deep learning models. Our deep learning pipeline included first labelling and one-hot encoding the input data, followed by tokenizing the text data, and then padding the text data to a fixed length. We then used an embedding layer to convert the text data into a dense vector of fixed size. Those embeddings were then passed through our deep learning models. The figure below illustrates the complete deep learning pipeline.

Figure 3: Deep Learning Pipeline

We used the "Pandas" library for creating the dataframes. The "Label Encoder" was used for label encoding the 20 poets in our dataframes. We used our previous weighted train-test split function for splitting the data into train, validation and test sets. We used an 75-5-20 train-validation-test split. The remaining methodology remains the same for tokenization, padding, and embedding layers. We used the "Keras" library for the deep learning models, and the "TensorFlow" library for the backend.

We initially thought of using Bi-LSTM, Bi-GRU, and CNN models for our deep learning pipeline. However, after various experimentation and hyperparameter tuning, we found that those deep learning models were not performing well on our dataset, mostly overfitting on the training data due to which we were not getting a desired validation or test accuracy as expected. Therefore, after experimenting with the Bi-LSTM model, we decided to move onto other Deep Learning models instead.

1) Bi-LSTM: For the Bi-LSTM model, we used 3 Bi-LSTM layers with 64, 32, and 16 units respectively. We also used a dropout layer with a dropout rate of 0.3 after each Bi-LSTM layer. We used the softmax activation function for the output layer. We used the Adam optimizer with a learning rate of 0.001, and sparse categorical cross-entropy as the loss function. We used early stopping with a patience of 10 epochs. We trained the model for 20 epochs with a batch size of 32. We also experimented with various hyperparameters for the Bi-LSTM model, including the number of units, the number of layers, the dropout rate, the learning rate, and the batch size. We also implemented a simple LSTM model, however, the Bi-LSTM model performed better than

the simple LSTM model, but even that was overfitting on the training data. We suspect the overfitting was due to the small dataset size, and similar features in the poetry of different poets. Therefore, we decided not to move on with more traditional deep learning models and instead move on to transformer models instead.

D. Transformer Models

Tranformers are a type of deep learning model aimed at solving natural language processing, and other sequence-based tasks such as text classification, generation, and translation. Transformers have been widely used for text classification tasks due to their ability to learn long-range dependencies in the data. They utilize the power of attention, and encoder decoder architecture. As from our literature review, transformer based models can be seen to perform well in the recent papers for text classification tasks, we decided to experiment with transformer models for Pashto poetry classification. We experimented with three transformer models; roBERTa, DistilBERT, and Meta Llama.

The methodology remains the same as for the Deep Learning models, since transformers are essentially deep learning models. We used the "Pandas" library for creating the dataframes. The "Label Encoder" was used for label encoding the 20 poets in our dataframes. We used our previous weighted train-test split function for splitting the data into train, validation and test sets. We used an 80-10-10 train-validation-test split. The remaining methodology remains the same for tokenization, padding, and embedding layers where needed. We used the "Hugging Face" library for the transformer models, and the "TensorFlow" library for the backend.

1) roBERTa: roBERTa is a robustly optimized BERT approach, which is a transformer-based model. It is a variant of BERT, and has been pre-trained on a large corpus of text data. We used the "Hugging Face" library for the roBERTa model.

2) DistilBERT: DistilBERT is a distilled version of BERT, which is a transformer-based model. It is a smaller version of BERT, and has been pre-trained on a large corpus of text data. We used the "Hugging Face" library for the DistilBERT model.

3) Meta Llama: We used the Meta Llama 3.2 model, which is collection of multilingual large language models (LLMs) is a collection of pretrained and instruction-tuned generative models in 1B (billion parameters) and 3B (billion parameters) sizes (text in/text out). The Llama 3.2 instruction-tuned text only models are optimized for multilingual dialogue use cases, including agentic retrieval and summarization tasks. They outperform many of the available open source and closed chat models on common industry benchmarks. For our task, we opted for the 1B model. We used the "Hugging Face" library for the Meta Llama model.

E. Bert Based Multilingual Uncased

We also tried the Bert Based Multilingual Uncased model, a pretrained model of 168M (million) parameters, on the top 102 languages with the largest Wikipedia using a masked language modeling (MLM) objective. The MLM model takes a sentence, and randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence. We used the "Hugging Face" library for the Bert Based Multilingual Uncased model.

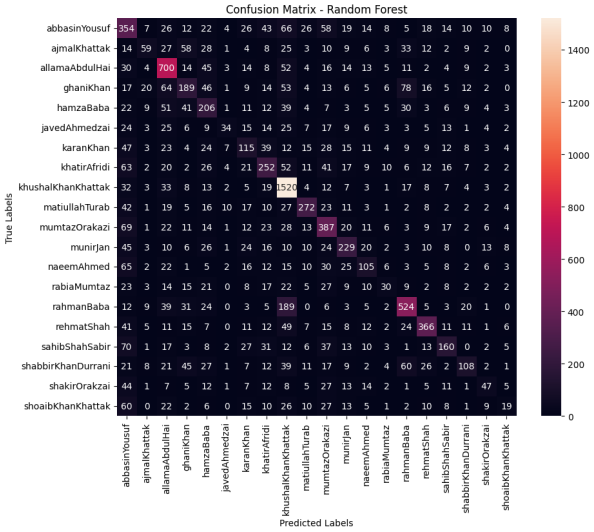
IV. Results

In this section we explore and discuss the results of our experiments with the machine learning, deep learning, and transformer models for Pashto poetry classification. We present the results of our experiments in the following subsections.

A. Machine Learning Models

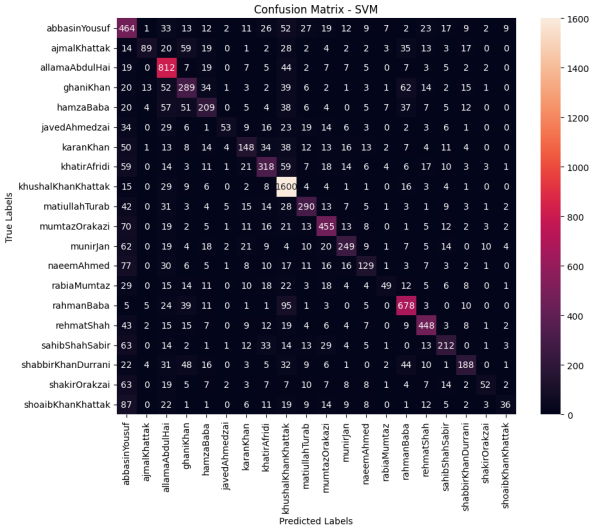
This subsection presents the results of our experiments with the machine learning models. We used the following machine learning models for our experiments: Random Forest, Support Vector Machine, Logistic Regression, and XGBoost. The dataset was divided into 21844 samples for training, and 5465 samples for testing. The models were trained on an 8 Core, 4.90 GHz Intel Core i7-12700 CPU with 16GB RAM.

1) Random Forest: Using 100 estimators, and a max depth of None, the Random Forest model achieved an accuracy of 51.93%, and an F1 score of 0.50. The confusion matrix for the Random Forest model is shown below:

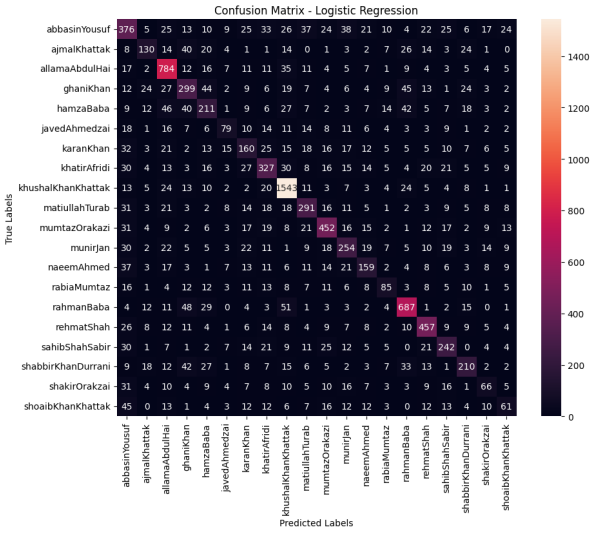


2) Support Vector Machine: With a C value of 1.0, an 'rbf' kernel, and a gamma value of 'scale', the Support Vector Machine model achieved an accuracy of 61.92%, and

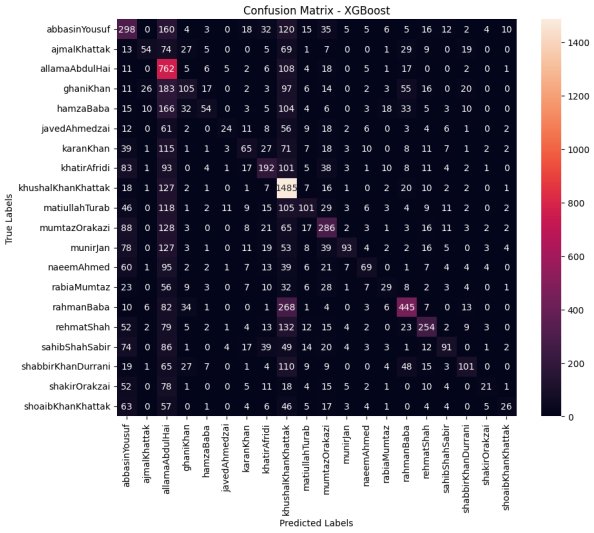
an F1 score of 0.60. The confusion matrix for the Support Vector Machine model is shown below:



3) Logistic Regression: For the Logistic Regression model, our best hyperparameters turned out to be a C value of 10, max iterations of 1000, and a 'newton-cg' solver. This gave an accuracy of 62.88%, and an F1 score of 0.62. The confusion matrix for the Logistic Regression model is shown below:



4) XGBoost: Our best hyperparameters for XGBoost were 100 estimators, a learning rate of 0.1, max depth of 3, gamma value of 0.1, and a log-loss evaluation metric. This gave us an accuracy of 41.67% and an F1 Score of 0.38. The confusion matrix for the XGBoost model is shown below:



B. Deep Learning Models

This subsection presents the results of our experiments with the deep learning models. We used the following deep learning models for our experiments: LSTM and Bi-LSTM. The dataset was divided into 19111 samples for training, 2727 samples for validation, and 5471 samples for testing. They were trained on 2 Tesla T4 GPUs available on Kaggle, with Kaggle's 29GB RAM.

1) LSTM: We experimented with various layers, and hyperparameters for LSTM including Bi-LSTM layers as well. The models were trained on 20 Epochs, with early stopping enabled. We experimented with a simple LSTM model with 2 LSTM layers of 128 and 64 units, respectively. We also added 2 dropout layers, and a dense layer with 64 units. The model was overfitting the training data, hence we continued with different parameters including compiling with an Adam Optimizer with a learning rate of 0.001, and a sparse categorical cross entropy loss, but that as well overfitted with a poor accuracy on the validation and test sets. Our final model was a Bi-LSTM model with a Dense layer with a softmax activation function, compiled using the Adam Optimizer with a learning rate of 0.0001, sparse categorical cross entropy loss, and an early stopping layer. The model achieved an accuracy of 59.63% on the test set. The plots and confusion matrix for the Bi-LSTM Model are shown as follows:

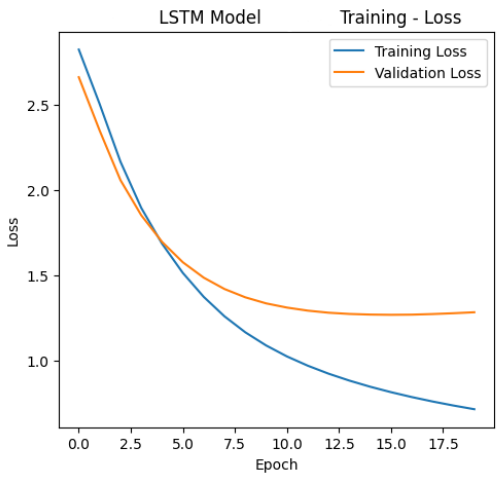


Figure 4: Training Loss for LSTM Model

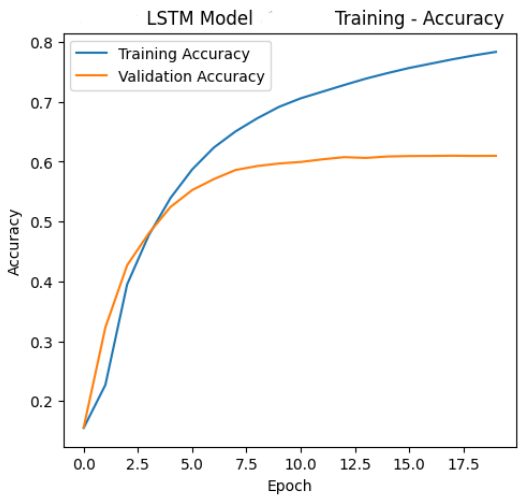


Figure 5: Training Accuracy for LSTM Model

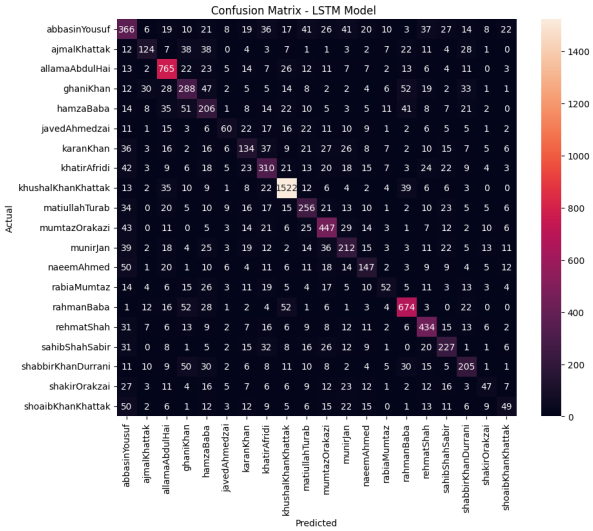


Figure 6: Confusion Matrix for LSTM Model

C. Transformer Models

This subsection presents the results of our experiments with the transformer models. We used the following transformer models for our experiments: roBERTa, DistilBERT, and Meta Llama-3.2-1b. The embeddings generated were trained on 2 Tesla T4 GPUs available on Kaggle, with Kaggle's 29GB RAM.

1) roBERTa:

2) DistilBERT:

3) Meta Llama-3.2-1b: The Meta Llama 3.2 1b parameter model was trained on 20 epochs, with a batch size of 8, learning rate of 0.001, the Adam optimizer, and cross entropy loss function, with an early stopping. The model only achieved an accuracy of 48.88%. The plots and confusion matrix for the Meta Llama-3.2-1b Model are shown as follows:

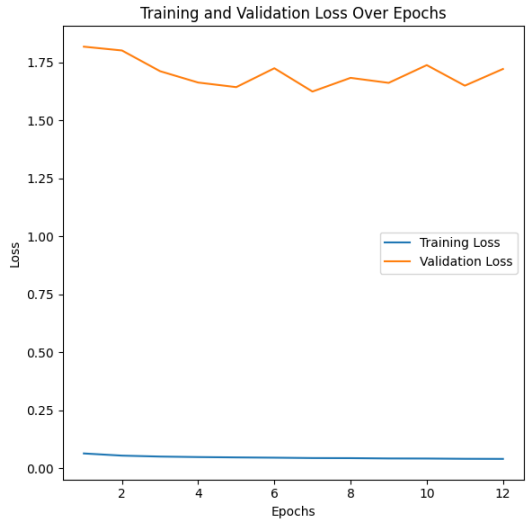


Figure 7: Training Loss for Meta Llama-3.2-1b Model

Figure 8: Training Accuracy for Meta Llama-3.2-1b Model

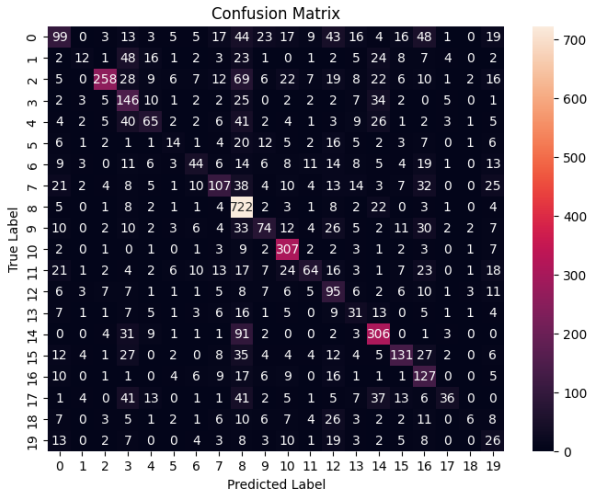


Figure 9: Confusion Matrix for Meta Llama-3.2-1b Model

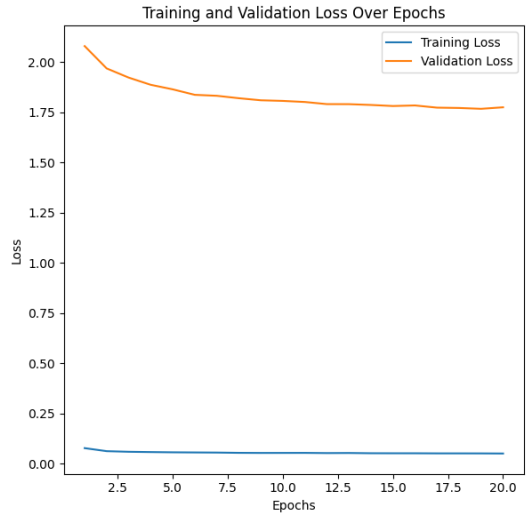


Figure 11: Training Loss for Bert Based Multilingual Uncased Model

D. Bert Based Multilingual Uncased

As above, the model was trained on 20 epochs, with a batch size of 8, learning rate of 0.001, the Adam optimizer, and cross entropy loss function, with an early stopping. The model only achieved an accuracy of 45.61%. The plots and confusion matrix for the Bert Based Multilingual Uncased Model are shown as follows:

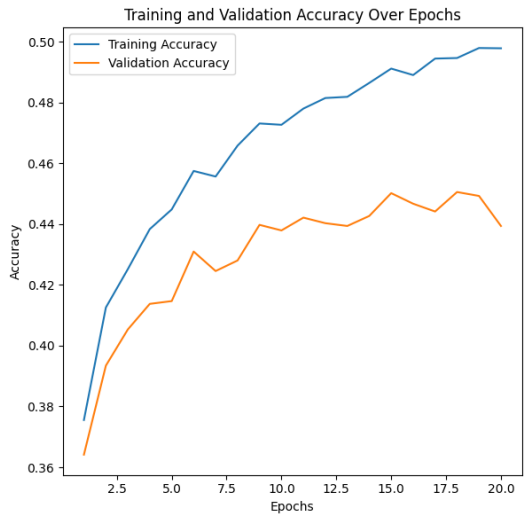


Figure 12: Training Accuracy for Bert Based Multilingual Uncased Model

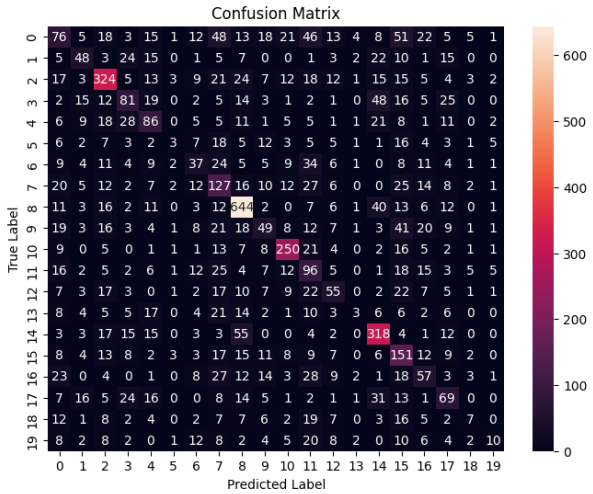


Figure 10: Confusion Matrix for Bert Based Multilingual Uncased Model

E. Summary of Results

This subsection presents a summary of the results of our experiments with the machine learning, deep learning, and transformer models for Pashto poetry classification.

Model	**Accuracy**
Logistic Regression	62.88%
Support Vector Machine	61.92%
Bi-LSTM	59.63%
Random Forest	51.93%
Meta Llama-3.2-1b	48.88%
Bert Based Multilingual Uncased	45.61%
XGBoost	41.67%

Table II: Table of Results

V. Discussion and Analysis

VI. Future Work

References

[1] C. W. Factbook, "Afghanistan." https://www.cia.gov/the-world-factbook/countries/afghanistan/, 2024.

[2] P. B. of Statistics, "Population by mother tongue." https://web.archive.org/web/20220409115251/https://www.pbs.gov.pk/sites/default/files//population_census/census_2017_tables/pakistan/Table11n.pdf, 2021.

[3] A. Aziz, "A brief history of pashto literature," *Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 6, 2019.

[4] I. Siddiqui, F. Rubab, H. Siddiqui, and A. Samad, "Poet attribution of urdu ghazals using deep learning," in *2023 3rd International Conference on Artificial Intelligence (ICAI)*, pp. 196–203, 2023.

[5] J. F. Ruma, S. Akter, J. J. Laboni, and R. M. Rahman, "A deep learning classification model for persian hafez poetry based on the poet's era," *Decision Analytics Journal*, vol. 4, p. 100111, 2022.

[6] B. Mehta and B. Rajyagor, "Gujarati poetry classification based on emotions using deep learning," *International Journal of Engineering Applied Sciences and Technology*, vol. 6, 05 2021.

[7] T. Ahmed and A. Rao, "Poet attribution for urdu: Finding optimal configuration for short text," *Poet attribution for urdu: Finding optimal configuration for short text*, vol. 4, 2021.

[8] M. Dar, "Authorship attribution in urdu poetry," 06 2020.

[9] N. Tariq, I. Ejaz, M. K. Malik, Z. Nawaz, and F. Bukhari, "Idenitification of urdu ghazal poets using svm," *Mehran University Research Journal of Engineering and Technology*, vol. 38, no. 4, 2019.

[10] A. Al-falahi, M. Ramdani, and M. Bellafkih, "Machine learning for authorship attribution in arabic poetry," *International Journal of Future Computer and Communication*, vol. 6, pp. 42–46, 01 2017.

[11] J. Baktash and M. Dawodi, "Enhancing pashto text classification using language processing techniques for single and multi-label analysis," 05 2023.