From Classical to Contemporary: Evolutionary Analysis & Classification of Urdu Poetry

Noor Fatima[1] **Hasan Faraz Khan**[1] **Irfan Ahmad**[1,2]
g202427440@kfupm.edu.sa g202427420@kfupm.edu.sa irfan.ahmad@kfupm.edu.sa
[1]Information and Computer Science Department, KFUPM, Dhahran, Saudi Arabia
[2]SDAIA-KFUPM Joint Research Center for AI, Dhahran, Saudi Arabia

Abstract

Automatic classification of literary text by historical era can support literary analysis and reveal stylistic evolution. We study this problem for Urdu poetry across three eras, classical, modern, and contemporary. We introduce a new dataset of 10,026 four-line Urdu poetry segments collected from online archives (Rekhta and UrduPoint) and labeled by era. To handle Urdu's script and orthographic variability, we apply standard preprocessing, including Unicode normalization and removal of diacritics and non-Urdu characters. We benchmark a range of approaches, from traditional machine learning classifiers to deep learning models, including fine-tuned Urdu BERT-style transformers. To assess generalization, we evaluate under two regimes: (i) a standard stratified random split and (ii) a stricter author-disjoint split that ensures poets do not overlap between training and test sets. On the random split, the best traditional models achieve about 70-73% accuracy, suggesting era-related stylistic cues are learnable. However, performance drops to roughly 58-60% under the author-disjoint split, highlighting the difficulty in generalizing across unseen poets and the possibility of overestimating performance via author-specific leakage. Notably, fine-tuned transformers do not surpass simpler TF-IDF-based baselines, indicating that era cues may be subtle and that data limitations constrain more complex models.

Keywords: Urdu NLP; historical text analysis; era classification; Urdu poetry; authorship attribution.

1 Introduction

Language evolves under historical, social, and technological pressures. Urdu (Fig. 1), with a deep literary tradition and widespread use in South Asia, has undergone notable shifts in structure, vocabulary (Abbas et al., 2022), and expressive style, from classical poetry and prose to contemporary digital communication. Urdu poetry spans centuries (Kanwal et al., 2019) and is commonly grouped into classical, modern, and contemporary eras, which differ in diction, imagery, and thematic emphasis. Classical poetry (18th-early 19th century) is often associated with ornate, Persian-influenced language and courtly patronage, while late-19th-century reformist currents and colonial-era modernism encouraged clearer diction and socially grounded themes. In the 20th century, modern poets increasingly moved away from abstract romance and mysticism, and contemporary poetry expanded further through forms such as the *nazm* and movements like *Nai Shaeri*, embracing freer verse and new imagery (Parekh, Accessed 15 Feb 2025). These shifts suggest discernible era-level differences: classical *ghazals* tend toward archaic or highly Persianized vocabulary and stylized metaphors, whereas modern and contemporary works often employ simpler language, novel imagery, and more direct social commentary (Rekhta, 23 Feb 2025).

وہ ہر دل عزیز ہے ہر پھر کے تیرے کوچہ میں کرتے ہیں ہم مقام
نہ ہوا تھا سو ہوا غمزۂ شوخ سیہ مست ترا مژگاں سے قتل عاشق پہ
اک ٹھکانہ نہیں کہ تجھ سے کہیں اے خدا درد دل ہے بخشش دوست

Figure 1: The Urdu Script.

Despite these distinctions, automatically classifying poetry by era is challenging. Poetic language is highly figurative, and Urdu adds practical NLP difficulties as a morphologically rich, Perso-Arabic script language with orthographic variation and segmentation ambiguity, in a relatively low-resource ecosystem (Hassan et al., 2024). Moreover, era cues can overlap with author cues: poets may intentionally adopt older diction or traditional forms, so models risk learning poet-specific signatures rather than era-level patterns (Aslam et al., 2025).

Most computational work on Urdu poetry has

Proceedings of the 2nd Workshop on NLP for Languages Using Arabic Script (AbjadNLP 2026), pages 182–191

focused on authorship attribution, often achieving high accuracy by exploiting stable authorial fingerprints (Khan et al., 2023). Era classification is a different problem: it requires capturing broader stylistic trends that generalize across poets, and naïve evaluation can overestimate performance if poets overlap between training and testing. To address this gap, we (1) curate an era-labeled Urdu poetry dataset from credible repositories, (2) benchmark traditional machine learning and deep learning approaches (Shahid et al., 2024), and (3) evaluate generalization using both a random split and an author-disjoint split that tests on unseen poets. This comparison helps distinguish models that learn era-specific patterns from those that primarily recognize poets seen during training (Lal et al., 2020).

Our contributions include an empirical analysis of which features and models, ranging from lexical baselines to transformer-based methods, are most effective for era classification in a low-resource literary setting, and a benchmark dataset for Urdu poetic-era classification to support reproducible future work (Khan et al., 2024). The remainder of the paper is organized as follows: Section 2 reviews related work; Section 3 describes the dataset, preprocessing, models, and evaluation; Sections 4 and 5 present results and discussion; and Section 6 concludes with future directions.

2 Related Work

Computational prediction of a text's literary era is often framed as diachronic (temporal) classification or stylistic periodization, where labels correspond to long-term shifts in language use rather than topical categories. In such settings, discriminative cues are frequently subtle, reflecting changes in diction, stylistic conventions, and broader cultural-literary trends, making the task inherently different from topic-driven classification. A closely related line of research is computational stylometry and authorship analysis, especially in poetry, where models can exploit stable author-specific lexical and stylistic signatures. While this literature provides useful methodological insights (e.g., feature sensitivity and stylistic markers), it also highlights an important caveat for era prediction: strong performance can arise from learning poet-specific signals rather than era-level characteristics when evaluation splits permit author overlap. For this reason, prior stylometric findings are most informative here as guidance on split design and generalization testing rather than as direct solutions to era labeling.

Language evolution is examined (Grifoni et al., 2016) as a growing research area leading to various computational models. Several studies have explored key NLP tasks, including preprocessing, tokenization, part-of-speech (POS) tagging, and named entity recognition, contributing to information retrieval and text classification applications. We find no prior work that directly studies diachronic/ era-based modeling for Urdu poetry (Daud et al., 2017). With the rapid growth of digital content, structured access to Urdu text remains a challenge, particularly in news classification (Rasheed et al., 2018). Retrieving precise information from vast unstructured web data is particularly difficult for Urdu due to word sense ambiguity (WSA) (Shoaib et al., 2023).

Urdu remains comparatively low-resource, with much of the prior work focusing on foundational datasets and general-purpose tools rather than literary-era modeling. For example, efforts include Urdu corpora creation and standardization (Becker and Riaz, 2002; Naseer et al., 2021), the development of Urdu NLP toolkits (Shafi et al., 2023), and general infrastructure work supporting tasks such as tagging and transliteration (Mukund et al., 2010). In addition, Urdu text classification has been studied in non-literary settings (Ali and Ijaz, 2009; Asim et al., 2021).

The most directly comparable prior work comes from Arabic and Persian poetry, where historical-period prediction has been formulated as supervised text classification across well-defined eras, closely aligning with our Urdu setting. Arabic studies explicitly classify poems into major periods (e.g., Pre-Islamic, Umayyad, Abbasid, Andalusian) and benchmark alternative feature representations and learning algorithms for era labeling (Abbas et al., 2019), while also highlighting the practical difficulty of separating genuine period effects from stylistic overlap—especially around transitional movements. In Persian, computational literature has long emphasized stylistic analysis, and more recent deep-learning work has explored chronology/era prediction using learned representations (e.g., embeddings) and sequential architectures on poetic forms such as ghazals (Orabi et al., 2020; Makhoul Sleiman et al., 2024). Collectively, this multilingual line of work provides the closest methodological parallel for Urdu poetry era classification: historically grounded labels, poetic text as input, and reliance on stylistic/lexical cues under

careful evaluation protocols.

Literary-era classification in Urdu requires generalization beyond authorial fingerprints and genre conventions; therefore, evaluation protocols that permit poet or poem overlap across splits can substantially overestimate apparent "era learning." This motivates our emphasis on stricter split design and interpreting confusions as stylistic overlap rather than mere mislabeling (urduhack, Accessed 4 Feb 2025). Prior Urdu NLP has often focused on foundational corpora and general-purpose pipelines. We draw on that ecosystem primarily for preprocessing and modeling infrastructure, while our main contribution is a novel dataset for Urdu poetry era classification that combines historical and contemporary sources to enable more robust evaluation; to our knowledge, no comparable compilation currently exists for studying diachronic variation in Urdu poetry.

3 Methodology

A. Proposed System

The proposed system is organized as a text-processing pipeline (Fig. 2) from raw data collection to final classification. It begins by gathering a large corpus of Urdu poems from credible online literary repositories to ensure broad coverage across eras. The raw poems are then passed through a preprocessing module for script normalization and noise reduction. After cleaning, each poem is segmented into four-line textual samples to standardize the input length (Fig. 3). Finally, these uniform segments are fed into classification models that assign each sample an era label (Classical, Modern, or Contemporary). Several design choices in this pipeline are tailored to the task. Using Rekhta (Rekhta, 23 Feb 2025) and UrduPoint (UrduPoint, 23 Feb 2025) is advantageous for data collection because of their literary credibility and extensive poetry. Segmenting each poem into four-line samples provides a balanced context window, roughly a couplet pair, ensuring each sample contains sufficient poetic context while maintaining uniform length across the dataset. The preprocessing stage applies Urdu-specific text normalization: We remove diacritics, normalize character variants, and strip out extraneous punctuation or non-Urdu symbols. These steps reduce orthographic noise and variability so that the classification features reflect the poetry's linguistic patterns rather than artifacts of encoding or script inconsistencies.

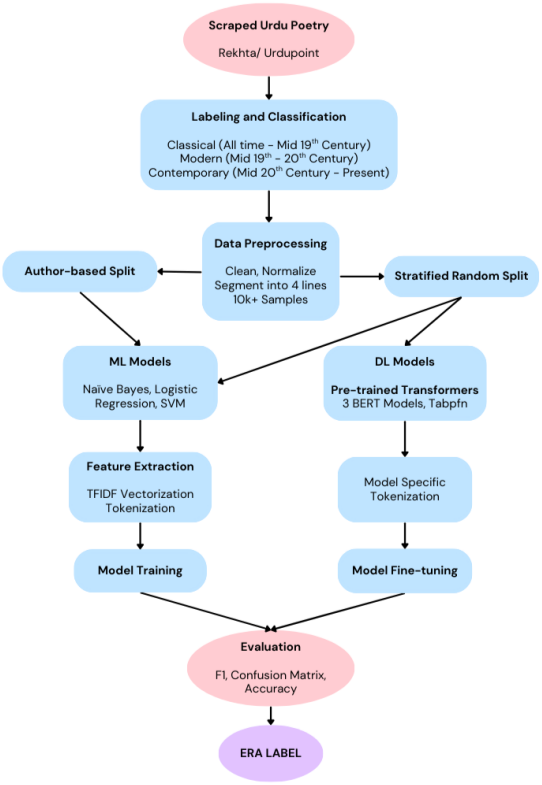

Figure 2: Proposed Pipeline for Urdu Era Classification.

At the classification stage, the system employs two parallel modeling approaches to evaluate the most effective textual features. The first approach is a traditional machine learning pipeline: each four-line sample is transformed into a TF-IDF vector of word features, which is then used to train classifiers such as logistic regression, support vector machines, or Naïve Bayes. Our second approach leverages transformer-based language models such as XLM-R (xlm-roberta-base), mBERT (bert-base-multilingual-cased) and BERT (bert-base-uncased). We fine-tuned them on the poetry samples so that contextual embeddings inform the era prediction. By comparing these models, we can assess whether surface-level lexical cues are sufficient for era identification or if deeper semantic and contextual patterns yield superior performance. This dual strategy thus not only builds a robust classifier but provides insight into the nature of linguistic differences that distinguish classical, modern, and contemporary Urdu poetry.

We additionally explored tabPFN (a tabular probabilistic classifier applied to TF-IDF vectors) and other preliminary variants; they underperformed the reported baselines and are omitted for low performance. We did not include intermediate neural

	ERA	AUTHOR	SNIPPET
0	classical	Meer Hasan	کیوں کر نہ چاہیے اس کو ہر اک جان کی طرح خواہش م...
1	classical	Baqar Agah Vellori	... شاہد غیب ہویدا نہ ہوا تھا سو ہوا حسن پر آپ نے
2	contemporary	Ahmad Faraz	قاصدا ہم فقیر لوگوں کا اک ٹھکانہ نہیں کہ تجھ س...
3	contemporary	Umar Khaleeq Azmi	آرزو میری بھی ہے ساتھ چلوں تیرے مگر اتنے مہنگے...
4	modern	Jigar Moradabadi	پایا جاتا ہے تری شوخیٔ رفتار کا رنگ کاش پہلو م...
5	modern	Allama Iqbal	گرچہ ہے دل کشا بہت حسن فرنگ کی بہار طائرک بلند...

Figure 3: Example poem snippets per class.

baselines (CNN/RNN classifiers) to keep the study focused on (i) strong lexical baselines known to be competitive on small literary datasets and (ii) transformer fine-tuning as the dominant modern paradigm.

B. Preprocessing

All text data underwent thorough preprocessing to handle issues specific to Urdu script and poetry formatting. We normalized Unicode forms and removed diacritical marks (harakat like zabar, zer, pesh), which are often optional and inconsistently used in Urdu. Characters with variant forms were standardized. We also removed any residual HTML entities or extraneous symbols that may have appeared in the scraped text. Next, we stripped punctuation that was not relevant to Urdu poetic text: only Urdu-specific punctuation and introductory sentence terminators were retained. Non-textual elements like page numbers or transliteration artifacts were eliminated using regular expressions.

Additionally, we collapsed multiple whitespace characters and line breaks into single spaces so that each four-line segment became a continuous sequence of text. Importantly, we did not translate or transliterate the Urdu text; all processing was done in the native script. We also did not remove common words or stopwords in Urdu because what counts as a stopword can be era-dependent (for instance, the prevalence of Persian versus native Urdu function words might differ by era). However, the TF-IDF vectorizer (Ramos, 2003) was configured to ignore English stopwords simply as a precaution; in practice, our cleaning removed Latin characters, so this had minimal effect. After cleaning, each text sample was typically a couple of sentences of Urdu poetry in UTF-8 encoding, ready for vectorization or tokenization.

C. Evaluation Strategy

The performance of the classification models was evaluated using standard metrics, including accuracy, precision, recall, and F1-score. Accuracy measured the percentage of correctly classified texts, while precision and recall assessed the models' ability to identify true positives among predicted and actual positives, respectively. The F1-score, a harmonic mean of precision and recall, provided a balanced evaluation of the models' performance. These metrics were used to compare different models' effectiveness and identify areas for improvement. Cross-validation was used to ensure the models were not overfitting. We use weighted F1 (which accounts for the support of each class) as a key evaluation metric alongside overall accuracy since it provides a more informative measure when class sizes are unequal. Given the class imbalance in the author-disjoint split, we additionally report macro-F1, which weights all classes equally and better reflects performance on minority classes. We also plotted confusion matrices to visualize how classes were being mispredicted. These helped identify, for example, if modern-era verses were systematically being confused with classical or vice versa. We could observe the impact of author overlap by comparing the regular split and author-disjoint split results.

4 Experiments and Results

4.1 Dataset Scraping

We compiled a novel dataset, the UrduPoetryEra Corpus (UPEC), spanning three eras (classical, modern, contemporary). Poems were collected from two online repositories: Rekhta and UrduPoint. Using custom web scrapers, we extracted poem text and metadata (e.g., poet name). Each poem was assigned an era label based on the poet's historical period, and the data were organized hierarchically by era and author to support author-based splitting.

4.1.1 UrduPoint

Poems were scraped automatically using Python (`requests + BeautifulSoup`). Poet URLs were collected from the UrduPoint poet listing page via `a.ipoet_box`. For each poet, poem links were gathered from `div#list_items` via `a.ipoem_box`, and poem text was extracted from `div.poem_text.urdu` as plain text (`get_text(separator="\n")`). Crawl-time processing was limited to HTML→text conversion and removal of obvious boilerplate; Urdu-specific normalization was handled later in preprocessing.

Table 1: Dataset statistics (pre-downsampling) by era showing segments, total tokens and unique tokens.

Era	Segments	Total	Unique
Classical	4369	102072	9342
Contemporary	3638	69750	8248
Modern	3342	67356	7551

4.1.2 Rekhta

Rekhta poems were scraped using a `requests.Session()` with a custom User-Agent and polite delays (`sleep()`). Crawling started from the Rekhta poet index (`lang=ur`), extracting poet links from `div.poetColumn`, then following `readFull` pages to collect poem URLs (from `h3.noPoetSubTtl` entries). Poem text was parsed from `input#HtmlRawText` (the embedded `data-html`) and converted to plain text by joining verse `<p>` blocks. Records store `era_label`, `poet`, `poem_title/poem_id`, `source`, `poem_url`, `poem_text_raw`, and (after segmentation) `segment_index/segment_text`.

4.2 Dataset Creation and Split

Poem lengths vary substantially, so we converted poems into fixed-size samples for consistent classification (Table 1). We segment poems into 4-line blocks (two couplets), a meaningful unit in Urdu poetry (especially ghazal) that provides enough local context for diction/metaphor while keeping inputs short for efficient training; this can reduce coherence in nazm/ free verse but controls length and increases samples. We considered whole-poem, 2-line, and sliding-window setups, and chose 4 lines as a balance reducing reliance on any single poem's unique context, encouraging models to focus on stylistic signals. To prevent segment-level leakage, all segments originating from the same poem are kept within a single split (grouped by poem ID/filename). Thus, the random split may allow poet overlap, but never poem overlap.

The raw corpus was imbalanced due to unequal availability across eras. To mitigate bias toward majority classes, we downsampled each era to the smallest post-segmentation class size, capping each class at $n = 3342$ segments. This produced a balanced dataset of 10,026 samples (3,342 per era), using a fixed seed for reproducibility.

We evaluated using two strategies: (a) Random (Stratified) Split and (b) Author-based Split. For the random split, we used a 60/20/20 train/val/test split with stratification by label (Tables 2 and 3); samples from the same poet can appear across splits, providing a standard baseline that may benefit from author overlap. For the author-based split, we grouped samples by poet and ensured no poet appears in more than one split. We first split authors into train/test pools (80/20), stratified by era at the author level, then carved out 25% of training authors for validation. Because poets contribute uneven numbers of segments, the resulting class distributions are naturally imbalanced.

Table 2: Dataset Statistics: Class-wise Distribution Across Random Split

Split	**Classical**	**Modern**	**Contemporary**
Train	2005	2005	2005
Val	669	668	668
Test	668	669	669

Table 3: Dataset Statistics: Class-wise Distribution Across Author Split

Split	**Classical**	**Modern**	**Contemporary**
Train	2434	1990	2187
Val	191	495	595
Test	717	857	560

4.3 Experimental Setup

We benchmarked models from simple baselines to transformer fine-tuning. As traditional baselines, we used Logistic Regression, linear SVM, and Multinomial Naïve Bayes with TF-IDF representations. We tokenized using whitespace/basic punctuation and limited the vocabulary to the top 5,000 terms to control sparsity. No stemming was applied due to Urdu's morphological complexity. Hyperparameters were tuned on the validation set (e.g., L2 + `max_iter`=1000 for LR, $C = 1$ for SVM, default Laplace smoothing for NB).

For deep learning, we fine-tuned multilingual transformer encoders for three-way classification using Hugging Face: XLM-R (Conneau et al., 2020) and BERT. Inputs were tokenized and truncated/padded to length 512. We trained with AdamW and standard text-classification settings (learning rate 3×10^{-5}, batch size 8, three epochs), using early stopping for XLM-R but not for BERT. Transformers were evaluated on the random split only due to computational constraints and their observed limitations on this dataset (Table 4).

Table 4: Hyperparameters and settings used in the experimental design.

Model	Representation / Tokenization	Key hyperparameters (as specified in files)
TF-IDF + Logistic Regression	TF-IDF features; vocabulary capped at 5,000	L2 regularization; `max_iter`=1000
TF-IDF + Linear SVM	TF-IDF features	Linear kernel; `C`=1
TF-IDF + Multinomial Naïve Bayes	TF-IDF features	Laplace smoothing (default additive smoothing)
XLM-R (`xlm-roberta-base`) (random split only)	BPE tokenizer; truncation/padding to fixed length	`learning_rate`=3e-5; train batch size=8; eval batch size=16; epochs=3; early stopping patience=2; best checkpoint selected by weighted F1. `weight_decay`=0.01; `max_length`=512.
BERT (`bert-base-uncased`) (random split only)	WordPiece tokenizer; truncation/padding to fixed length	Epochs=3; no early stopping. *Notebook notes:* `learning_rate`=3e-5; train batch size=8; eval batch size=16; `weight_decay`=0.01; `max_length`=512;

Table 5: Performance on Random Split (A is accuracy, P is precision, R is recall, wF1 is weighted F1 and mF1 is macro F1 score).

Model	A	P	R	wF1	mF1
LR	0.71	0.71	0.71	0.71	0.71
SVM	0.70	0.71	0.70	0.70	0.70
NB	0.73	0.74	0.73	0.73	0.73
XLM-R	0.36	0.27	0.36	0.26	0.26
BERT	0.60	0.61	0.60	0.60	0.61

Table 6: Performance on Author-Disjoint Split TF-IDF.

Model	A	P	R	wF1	mF1
LR	0.58	0.58	0.58	0.57	0.58
SVM	0.57	0.57	0.57	0.57	0.60
NB	0.60	0.62	0.60	0.59	0.58

4.4 Results

4.4.1 Results on Random Split

Table 5 reports performance on the stratified random split. All traditional TF-IDF models achieved around 70% accuracy (well above the 33% chance baseline). Multinomial Naïve Bayes performed best (≈ 0.73 accuracy and 0.73 weighted F1), with LR and SVM close behind (0.70-0.71 F1). Errors were more frequent for the Modern class than for Classical or Contemporary.

In contrast, XLM-R underperformed (0.36 accuracy, 0.26 weighted F1) and tended to over-predict the Classical label, effectively missing Modern examples. This suggests unstable fine-tuning and/or insufficient data for learning subtle era cues in poetry. BERT achieved moderate performance (0.60 accuracy), but still lagged behind TF-IDF baselines, indicating that lexical signals captured by simpler models are strong for this task.

4.4.2 Results on Author-Disjoint Split

Under the author-disjoint setting (Table 6), performance dropped for all models, reflecting the difficulty of generalizing to unseen poets. Naïve Bayes achieved 60.2% (weighted F1 ≈ 0.59), followed by Logistic Regression (~ 0.58 F1) and SVM (~ 0.57 F1). This split is also class-imbalanced due to uneven poet productivity (Table 3), which further skews learning (e.g., Modern underrepresented in training but overrepresented in test). We did not fine-tune transformers on this split due to their underperformance on the random split and resource constraints.

5 Discussion

To better understand the behavior of the model, we examined the confusion matrices of the best traditional baseline (Naïve Bayes) and BERT in each setting (Fig. 4). On the random split, Naïve Bayes struggled most with the Modern class: only about 67% of Modern verses were correctly identified, with many being mislabeled as Classical or Contemporary. In contrast, Classical and Contemporary were more stable (roughly 71-74% recall). The dominant error was Modern→Classical, followed by Modern→Contemporary, whereas direct Classical↔Contemporary confusion was relatively rare. This suggests Modern poetry functions as a stylistic "middle ground" that overlaps with both extremes.

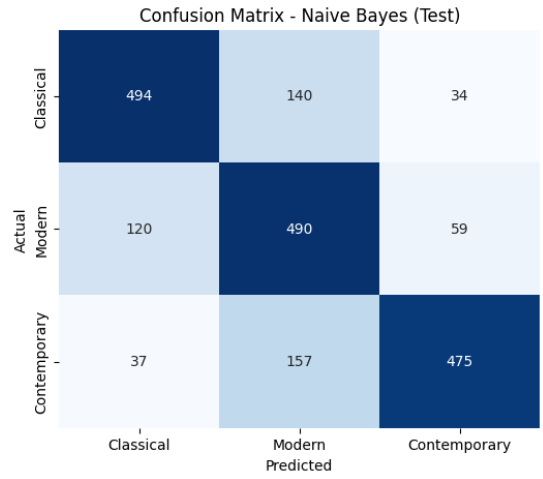

Figure 4: Confusion Matrix of Naïve Bayes with random split

Because the stratified random split permits poet overlap, it likely captures poet-specific lexical signatures that correlate with era labels. We therefore treat random-split scores as an optimistic upper bound. The author-disjoint split more directly tests era-level generalization to unseen poets, and the 13-15 point drop indicates that a substantial fraction of the signal learned under random splitting is poet-specific rather than era-specific. Naïve Bayes strongly over-predicted the Classical label: Modern recall dropped to 37% (Fig. 5), meaning nearly two-thirds of Modern poems were mislabeled, predominantly as Classical. Classical recall remained high (82%), but its precision fell to ∼53%, reflecting many false Classical predictions. Contemporary was less affected (about 69% correct), with most remaining errors occurring between Contemporary and Modern. When poet-specific cues were removed, the classifier defaulted to Classical for ambiguous cases, likely because it learned strong lexical/formal signals associated with older styles.

For BERT on the random split, the model did not collapse into a single class, but it still failed to cleanly separate eras, again with the weakest recall for Modern. Many Modern poems were predicted as Classical or Contemporary, indicating that the model did not acquire reliable cues for this transitional era. This diffuse confusion is consistent with using an English-pretrained transformer on Urdu poetry, where script and stylistic conventions differ from the pretraining domain; nevertheless, it still captured partial signals and outperformed XLM-R in our experiments. Across models, Modern-era poetry was consistently the hardest class to generalize, while Classical and Contemporary were identified more reliably.

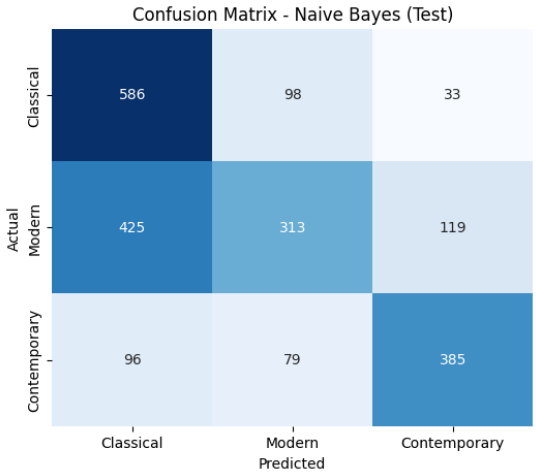

Figure 5: Confusion Matrix of Naïve Bayes with author-based split

6 Conclusions and Future Work

This study investigated automatic era classification for Urdu poetry, bridging computational linguistics and literary analysis. We constructed a balanced dataset of four-line segments from credible archives covering Classical, Modern, and Contemporary eras. Models were evaluated using both a stratified random split and a stricter author-disjoint split to test generalization to unseen poets. Traditional TF-IDF baselines, especially Multinomial Naïve Bayes, performed best achieving strong results on the random split and remaining comparatively stable under author-disjoint evaluation. Deep learning results were mixed: BERT showed moderate performance despite lacking Urdu-specific pretraining,

whereas multilingual XLM-R failed to converge effectively. Overall, the findings suggest that in low-resource literary settings, careful preprocessing and simple lexical models can outperform larger pretrained transformers. The author-disjoint results further indicate that random splits may overstate performance by capturing author-specific cues, and that imbalance in author-based splits particularly harms Modern-era recognition. These observations highlight the importance of evaluation design and model parsimony for robust literary NLP.

Future work will prioritize improving generalization under author-disjoint evaluation and mitigating imbalance from uneven author contributions via author-level stratification constraints, resampling, or cost-sensitive learning. Incorporating richer stylistic features (e.g., meter-related patterns, rhetorical structure, and semantic representations) may better capture diachronic shifts beyond lexical cues. On the deep learning side, domain adaptation continued pretraining on large-scale Urdu poetic text and more robust fine-tuning may yield stronger era-sensitive representations. Finally, expanding the corpus and testing cross-source robustness would further strengthen the dataset as a benchmark for computational literary analysis.

7 Limitations

Despite demonstrating that era classification of Urdu poetry is feasible, several limitations should be considered.

7.1 Dataset scope

Although the corpus was sourced from credible repositories [1] [2] , it is restricted to four-line segments and a balanced subset created via downsampling. This facilitates controlled comparison across eras but may remove naturally occurring diversity, and era labels based on poet-era association may not fully capture stylistic overlap (e.g., poets adopting older forms). This may affect both generalization and ecological validity. As a mitigation, future work will evaluate imbalance-aware alternatives (e.g., class-weighted objectives or focal loss) and report results under the original distribution in addition to balanced benchmarks.

[1] https://rekhta.org/

[2] https://www.urdupoint.com/

7.2 Author-disjoint imbalance and generalization challenges

The author-based split reduces leakage but introduces class imbalance due to uneven poet productivity and unequal numbers of poets per era. This likely contributed to the performance drop, especially for Modern, and limits generalization to unseen poets.

7.3 Feature and modeling constraints

TF-IDF features capture lexical frequency but not deeper stylistic cues such as meter, rhetoric, or semantic abstraction. Transformer experiments were also limited in training budget, and XLM-R did not converge well, suggesting that stronger domain adaptation (e.g., Urdu-poetry-specific pretraining) may be needed to surpass lexical baselines.

7.4 Evaluation boundaries

While accuracy, weighted F1, and confusion matrices are informative, we do not include expert qualitative validation or cross-corpus robustness tests (e.g., train on one source, test on another), which would strengthen claims about stylistic generalization. These limitations align with the observed Modern-era confusions and the Classical overprediction under author-disjoint evaluation. Accordingly, we view our results as a strong baseline and benchmark for Urdu poetic era classification rather than a definitive solution.

Acknowledgments

The authors would like to thank Saudi Data and AI Authority (SDAIA) and King Fahd University of Petroleum & Minerals (KFUPM) for supporting this work through SDAIA-KFUPM Joint Research Center for Artificial Intelligence grant number JRC-AI-CAI02563.

References

Mourad Abbas, Mohamed Lichouri, and Ahmed Zeggada. 2019. Classification of arabic poems: from the 5th to the 15th century. In *International Conference on Image Analysis and Processing*, pages 179–186. Springer.

Syed Zain Abbas, Arif ur Rahman, Abdul Basit Mughal, and Syed Mujtaba Haider. 2022. Urdu news article recommendation model using natural language processing techniques. *Preprint*, arXiv:2206.11862.

Abbas Raza Ali and Maliha Ijaz. 2009. Urdu text classification. In *Proceedings of the 7th international*

conference on frontiers of information technology, pages 1–7.

Muhammad Nabeel Asim, Muhammad Usman Ghani, Muhammad Ali Ibrahim, Waqar Mahmood, Andreas Dengel, and Sheraz Ahmed. 2021. Benchmarking performance of machine and deep learning-based methodologies for urdu text document classification. *Neural Computing and Applications*, 33:5437–5469.

Muhammad Ali Aslam, Khairullah Khan, Wahab Khan, Sajid Ullah Khan, Abdullah Albanyan, and Shabbab Ali Algamdi. 2025. Paraphrase detection for urdu language text using fine-tune bilstm framework. *Scientific Reports*, 15(1):15383.

Dara Becker and Kashif Riaz. 2002. A study in urdu corpus construction. In *COLING-02: The 3rd Workshop on Asian Language Resources and International Standardization*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8440–8451.

Ali Daud, Wahab Khan, and Dunren Che. 2017. Urdu language processing: a survey. *Artificial Intelligence Review*, 47:279–311.

Patrizia Grifoni, Arianna D'Ulizia, and Fernando Ferri. 2016. Computational methods and grammars in language evolution: a survey. *Artificial Intelligence Review*, 45(3):369–403.

Muhammad Hassan, Saad Ahmed, Rohail Qamar, Saman Hina, and Hira Farman. 2024. An nlp approach to predict and suggest next word in urdu typing. *VFAST Transactions on Software Engineering*, 12(4):158–166.

Safia Kanwal, Kamran Malik, Khurram Shahzad, Faisal Aslam, and Zubair Nawaz. 2019. Urdu named entity recognition: Corpus generation and deep learning applications. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(1):1–13.

Asif Khan, Khairullah Khan, Wahab Khan, Sadiq Nawaz Khan, and Rafiul Haq. 2024. Knowledge-based word tokenization system for urdu. *Journal of Informatics and Web Engineering*, 3(2):86–97.

Talha Farooq Khan, Waheed Anwar, Humera Arshad, and Syed Naseem Abbas. 2023. An empirical study on authorship verification for low resource language using hyper-tuned cnn approach. *IEEE Access*, 11:80403–80415.

Madan Lal, Kamlesh Kumar, Asif Ali Wagan, Asif Ali Laghari, Mansoor Ahmed Khuhro, Umair Saeed, Aamir Umrani, and M Ameen Chahjro. 2020. A systematic study of urdu language processing its tools and techniques: A review. *International Journal of Engineering Research & Technology*, 9(12):37–43.

Nariman Makhoul Sleiman, Ali Ahmad Hussein, Tsvi Kuflik, and Einat Minkov. 2024. Automatic era identification in classical arabic poetry. *Applied Sciences*, 14(18):8240.

Smruthi Mukund, Rohini Srihari, and Erik Peterson. 2010. An information-extraction system for urdu—a resource-poor language. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(4):1–43.

Asma Naseer, Tanzeela Shakeel, Kinza Arshad, and Zeenia Ather. 2021. Analysis of corpus development for urdu language. In *2021 international conference on innovative computing (ICIC)*, pages 1–5. IEEE.

Mariam Orabi, Hozayfa El Rifai, and Ashraf Elnagar. 2020. Classical arabic poetry: Classification based on era. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE.

Rauf Parekh. Accessed 15 Feb 2025. Ghazal and modern Urdu poem: two genres, two worlds — dawn.com. https://www.dawn.com/news/1168257. Accessed 15 Feb 2025.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. New Jersey, USA.

Imran Rasheed, Vivek Gupta, Haider Banka, and Chiranjeev Kumar. 2018. Urdu text classification: a comparative study using machine learning techniques. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pages 274–278. IEEE.

Rekhta. 23 Feb 2025. Urdu Poetry, Urdu Shayari of Famous Poets - Rekhta — rekhta.org. https://rekhta.org/. Accessed 23 Feb 2025.

Jawad Shafi, Hafiz Rizwan Iqbal, Rao Muhammad Adeel Nawab, and Paul Rayson. 2023. Unlt: Urdu natural language toolkit. *Natural language engineering*, 29(4):942–977.

Ramish Shahid, Aamir Wali, and Maryam Bashir. 2024. Next word prediction for urdu language using deep learning models. *Computer Speech & Language*, 87:101635.

Umar Shoaib, Laiba Fiaz, Chinmay Chakraborty, and Hafiz Tayyab Rauf. 2023. Context-aware urdu information retrieval system. *Transactions on Asian and Low-Resource Language Information Processing*, 22(3):1–19.

urduhack. Accessed 4 Feb 2025. GitHub - urduhack/urduhack: An NLP library for the Urdu language. It comes with a lot of battery included features to help you process Urdu data in the easiest way possible. — github.com. github.com/urduhack/urduhack. Accessed 4 Feb 2025.

UrduPoint. 23 Feb 2025. Urdupoint.com. https://www.urdupoint.com/. Accessed 23 Feb 2025.