Deep Learning-based Bilingual Lemmatizer for Urdu and Punjabi

Malaika Basharat
University of Gujrat, Pakistan
Email: malaikabdev@gmail.com

Muhammad Umar
University of Gujrat, Pakistan
Email: umarwaris450@gmail.com

Zafar Mahmood
University of Gujrat, Pakistan
Email: zafar.mehmood@uog.edu.pk

***Abstract*—Strong Natural Language Processing (NLP) systems for low-resource languages are becoming more and more necessary as data-driven technologies advance quickly. This paper offers a thorough neural framework for lemmatising Punjabi and Urdu, two Indo-Aryan languages with rich morphology but limited resources. The wide range of orthographic variety and inflectional diversity seen in these scripts has proven difficult for conventional rule-based and dictionary-driven techniques to represent. We create a character-level sequence-to-sequence model that uses recurrent neural architectures (SimpleRNN, LSTM, GRU, and BiLSTM) to learn lemma transformations straight from unprocessed word forms without the need for custom rules in order to overcome these difficulties. With 89,660 training and 22,416 testing samples, the bilingual corpus has 112,077 distinct word–lemma pairings (48,642 Punjabi and 63,435 Urdu entries). Two temporal alignment techniques; pre-padding and post-padding were thoroughly assessed to look at how they affected sequence modelling and convergence. Post-padding significantly improves performance by maintaining suffixal information, which is essential to Indo-Aryan morphology, according to experimental results. With the greatest accuracy (97.82%) and F1-score (0.9846) across all architectures, the BiLSTM demonstrated the effectiveness of bidirectional context modelling in identifying morphophonemic regularities. Without explicit transliteration, the model successfully learns sub-character correspondences between the Shahmukhi (Punjabi) and Perso-Arabic (Urdu) scripts, according to qualitative assessments employing confusion matrices. This study provides a computational framework for morphological processing in low-resource environments that is based on linguistics and reproducible, in addition to its empirical success. To make the codebase and trained model more accessible to academics and developers, it is publically available as an open-source Python pip package. Future research will investigate morpho-syntactic cues and transformer-based structures to enhance generalisation and adaptation even more. Finally, by connecting linguistic understanding with deep learning for practical uses including machine translation, information retrieval, and sentiment analysis, this work develops inclusive and scalable natural language processing (NLP) for South Asian languages.**

***Index Terms*—Urdu NLP, Punjabi NLP, Lemmatization, Morphologically Rich Languages, Deep Learning, LSTM, BiLSTM, Recurrent Neural Networks (RNN), Low-Resource Languages, Bilingual Lemmatizer, Shahmukhi Script, Corpus Development, Urdu-Punjabi Text Processing, Sequence-to-Sequence Learning, Language Technology for South Asia, Neural Morphological Analysis**

I. Introduction

A key component of natural language processing (NLP) pipelines, lemmatization is the act of mapping inflected word forms to their dictionary base form. It is essential for tasks ranging from machine translation and text analytics to information retrieval (IR) and search. By normalizing surface word variants to a common root, lemmatization can greatly enhance textual consistency and improve downstream performance in systems such as IR and text classification [1], [2] . In morphologically rich languages, lemmatization has been shown to outperform simpler stemming approaches, which merely strip affixes, because lemmatization properly handles complex inflections and irregular forms [3], [4] . Studies on low-resource and agglutinative languages (e.g., Amazigh, Bengali, Indonesian) have similarly found that using lemmatizers yields better retrieval accuracy and linguistic coherence than stemmers [5], [6] [7] . However, despite its importance, lemmatization for many low-resource languages remains under-explored [5] [8] . Until recently, the focus in such languages often centered on basic normalization or rule-based stemming, with limited attention to robust lemmatization models [9], [10] [1] . Developing accurate lemmatizers for these languages is challenging due to scarce annotated resources and the complex morphology involved [11], [12] [5] , yet it is increasingly recognized as vital for improving search, summarization, and translation in diverse languages [13] [14]. Urdu and Punjabi are prime examples of under-resourced Indo-Aryan languages that present unique linguistic and technical challenges for lemmatization. Both languages are morphologically rich and exhibit extensive inflectional variation. Urdu, for instance, uses suffixes (and occasional infixes) to mark gender, number, case, and tense, resulting in a large space of word forms per lemma [15], [16] [17] . Punjabi shares similar linguistic phenomena; as a closely related language, it also conjugates verbs for tense/aspect and declines nouns for number and grammatical case. This richness means a single lemma (root) can produce dozens of distinct word forms in context [18], [19] [9] . A lemmatizer for such languages must discern subtle phonological and orthographic changes (e.g. vowel elongation, consonant modifications) that occur during inflection [11], [12] [7] . The challenge is amplified by the Shahmukhi script used to write Urdu and Pakistani Punjabi. Shahmukhi is a Perso-Arabic script which is written right-to-left and, like Arabic, omits short vowel markings in ordinary text. The absence of diacritics leads to numerous homographs and ambiguities—different words can share the exact written form, complicating the retrieval of the correct lemma without context [20], [18] [19] . Moreover,

spelling in Shahmukhi is less standardized than in some Latin-script languages, and there are script-specific issues such as contextual letter forms and optional ligatures [20] [5] . Punjabi in Shahmukhi has received relatively little NLP attention compared to its Gurmukhi-script counterpart, partly due to lack of digitized corpora and standardized orthography [18] [19] . These linguistic characteristics make automatic lemmatization of Urdu and Punjabi a non-trivial problem: the system must handle rich morphology, disambiguate homographs using context (e.g., part-of-speech or neighboring words), and cope with noisy or inconsistent spellings in real-world text.

Another obstacle is the paucity of existing lemmatization tools and datasets for Urdu and Punjabi. Urdu has historically relied on rule-based morphological analyzers or stemmers [10], [1] [3]. Notable early work includes finite-state morphological analyzers for Urdu [15] [21] and rule-based lemmatizers that strip known affixes using handcrafted rules [10]. While these approaches can handle regular morphology, they often struggle with exceptions and require substantial linguistic expertise to develop. Recent studies have started to explore machine learning for Urdu lemmatization: for example, a dictionary-based lemmatizer was proposed by Shaukat et al. (2023) using a lexicon lookup combined with heuristics, and Hafeez et al. (2023) applied neural sequence models (BiLSTMs, GRUs, encoder-decoders) to Urdu, reporting significant accuracy gains [1] [2] . These efforts, however, remain monolingual and their methods may not scale easily to other low-resource languages. For Punjabi (especially Shahmukhi Punjabi), the situation is even more acute. To our knowledge, there is no prior published lemmatizer dedicated to Shahmukhi-script Punjabi. Most Punjabi text processing research has focused on the Gurmukhi script or on simpler tasks like word stemming [9] [22]. Only recently have resources begun to emerge: a Universal Dependencies treebank for Punjabi has been introduced [19], and initial studies on Shahmukhi Punjabi have addressed tasks like named entity recognition and part-of-speech tagging using deep learning [23] [24] . These works underscore the scarcity of tools for Shahmukhi text and often note that cross-script and cross-lingual approaches could be beneficial. Indeed, Urdu and Punjabi share a considerable portion of lexicon and grammatical structure (Punjabi is sometimes described as "Urdu-like" in linguistic surveys [18]), suggesting that a bilingual approach to lemmatization might allow the two languages to compensate for each other's data scarceness. Yet, no prior study has attempted a unified lemmatizer for both languages, nor explored leveraging their similarities in a joint model.

In this paper, we present a novel deep learning-based bilingual lemmatizer for Urdu and Punjabi (Shahmukhi script) that addresses the above challenges. Our approach is the first to train a single model to perform lemmatization in both languages. We construct a new mixed corpus combining Urdu and Shahmukhi Punjabi word forms with their lemmas. The Urdu portion is derived from a comprehensive lexical resource originally curated by Hussain [15] [21] , containing thousands of inflected forms from her Urdu morphological analyzer, while the Punjabi portion consists of a newly created dataset of Punjabi words manually annotated with their lemmas. By merging these, we obtain a bilingual training set that enables the model to learn cross-lingual patterns (for example, many Punjabi words of Persian/Arabic origin share roots with Urdu). We train and evaluate four recurrent sequence-to-sequence architectures—SimpleRNN, LSTM, GRU, and BiLSTM—on this bilingual dataset. The models were implemented in TensorFlow/Keras using Adam optimizer and sparse categorical cross-entropy loss, mapping a character sequence of the word to a character sequence of the lemma [11], [7]. In recent years, such RNN-based architectures (often enhanced with attention or transformers) have achieved state-of-the-art results in lemmatization and morphological inflection tasks across many languages [11] [12]. Our work is inspired by these advances: for instance, Bergmanis and Goldwater's Lematus model applied an encoder-decoder LSTM to context-sensitive lemmatization in high-resource languages[11], and Chakrabarty et al. (2017) used a two-stage BiGRU for lemma restoration in context [6]. We adapt and extend such neural approaches to a low-resource, bilingual setting, exploring how a unified model can learn the morphological rules of both Urdu and Punjabi simultaneously.

A key technical insight from our experiments is the impact of sequence padding strategy on model performance. Unlike many implementations that pad input sequences at the beginning (pre-padding) by default, we pad at the end of sequences (post-padding) and find that this post-padding yields consistently better accuracy across all tested architectures in our lemmatization task. Intuitively, post-padding preserves consistent alignment of word-final morphology across the batch, which may help the network more easily attend to suffixes (particularly important for suffix-dominant morphology). This observation corroborates best practices noted in some sequence modeling research—e.g., recommendations to use post-padding for RNNs to avoid diluting the signal of trailing characters [25] [26] —yet it is seldom discussed in the lemmatization literature. During evaluation, we ignore padding tokens when computing metrics to ensure fair comparison across sequence lengths. This work thus provides the first empirical comparison of pre- and post-padding strategies for Urdu and Punjabi lemmatization. By reporting this finding, we provide empirical evidence that padding strategy is a key factor for maximizing performance in character-level NLP models.

A. Key Contributions

Developing an accurate lemmatizer for morphologically rich and low-resource languages such as Urdu and Punjabi is a non-trivial task. Both languages exhibit extensive inflectional morphology, complex gender–number–case systems, and orthographic variability in the Shahmukhi script, making it extremely challenging for machine learning models to learn consistent morphological rules [15], [18], [19]. The absence of large-scale annotated corpora and the lack of standardized orthography further compound the problem [11], [12]. Unlike high-resource languages such as English or German, where

rich lexical resources and universal dependencies treebanks exist, Urdu and Punjabi remain underrepresented in most multilingual NLP datasets [13], [5]. Designing a lemmatizer that generalizes well under these constraints required both linguistic insight and deep learning innovation.

Our key contributions are summarized as follows:

1) **First bilingual lemmatizer for Urdu and Shahmukhi Punjabi.** We present, to our knowledge, the first deep learning-based system trained jointly on Urdu and Punjabi lemmatization. While prior work has addressed Urdu lemmatization using rule-based or dictionary lookup methods [1], [2], no prior study has explored a unified bilingual neural model leveraging the shared Indo-Aryan morphology of these languages.
2) **Creation of a new annotated corpus.** We developed a high-quality, bilingual morphological dataset by combining (i) a large lexical resource from Hussain's Urdu morphological analyzer [15], containing thousands of inflected forms, and (ii) a newly curated Punjabi dataset manually annotated with lemmas across all major verb, noun, and adjective classes. The Punjabi portion was built from scratch, involving extensive manual linguistic verification and cross-checking of inflectional patterns. This corpus, containing full lemma–inflection mappings, constitutes the most comprehensive dataset for these languages to date.
3) **Model innovation and comparative analysis.** We implemented and compared four sequence-to-sequence architectures—SimpleRNN, LSTM, GRU, and BiLSTM—trained in TensorFlow/Keras with the Adam optimizer and sparse categorical cross-entropy loss on the bilingual dataset to compare their ability to learn character-level morphological patterns. According to our tests, the Bidirectional LSTM model performs best overall and captures contextual dependencies that are crucial for morphology that is heavy on suffixes.
4) **Padding strategy and morphological alignment.** Because Urdu and Punjabi have a suffix-dominant morphological structure, we found that post-padding input sequences (as opposed to the widely utilised pre-padding) consistently produce superior accuracy and faster convergence. For the larger community researching RNN-based NLP models, this finding offers a useful empirical insight [25], [26].
5) **Bridging linguistic and computational gaps.** Our bilingual method shows that Punjabi and Urdu, two languages that have traditionally been taught separately, may successfully support one another in morphological learning. The shared lexical roots and parallel grammatical patterns allowed our BiLSTM to learn generalized transformations across both languages, offering a scalable path for future multilingual lemmatizers for under-resourced South Asian languages.

Building upon these challenges and motivations, our work aims to bridge the linguistic and computational gap in Urdu and Punjabi language processing through a bilingual, deep learning-based lemmatization framework.

II. Morphology of Urdu and Punjabi

Building on the linguistic challenges outlined in the Introduction, this section describes the core morphological processes in Urdu and Punjabi that motivate our lemmatizer design. Morphology is the study of the internal structure of words and the ways they change form to express grammatical meaning. Both Urdu and Punjabi are morphologically rich Indo-Aryan languages: nouns inflect for number, gender, and case; verbs inflect for person, gender, number, tense, aspect, mood, and (in Punjabi) honorific level. Prior work on Urdu morphology has noted that a single lemma can yield dozens of surface forms [15], [21], and our own rule-based expansion confirms that even a single verb root can systematically generate large families of inflected forms. This high level of surface variation makes lemmatization non-trivial.

A. Urdu Morphology

Urdu morphology is influenced by multiple sources (Arabic, Persian, Turkish, and Indo-Aryan strata), leading to productive inflectional and derivational patterns. Figure **??** illustrates how a single lemma such as اُبل (/ubal/, "to boil") can produce numerous attested surface forms. Each outward arrow corresponds to a different grammatical realization (e.g., tense, voice, causativity, or agreement). This explosion of forms is exactly what a lemmatizer must learn to collapse back to a canonical base form.

1) Urdu Nouns: Urdu nouns (اسم) inflect for number, gender, and case. Nouns are traditionally divided into common nouns (اسم نکرہ) and proper nouns (اسم معرفہ). Proper nouns typically act as their own lemmas and are not lemmatized further, but common nouns undergo rich inflection.

The following are examples of common surface regularities:

- Singular **masculine** nouns frequently finish in ا (alif), ہ (he), or ع (ain).
- Plural **masculine** nouns usually ends with ان (aan), ات (aat), or ے (e).
- Singular **feminine** nouns most commonly ended up with ی (i).
- Plural **feminine** nouns end with ا (a), ان (aan), وں (on), or یں (een).

Examples:

- حشرہ (hashra – "insect")
- سوالات (savaalaat – "questions")
- بانسری (bansuri – "flute")
- ڈالیاں (daaliyaan – "branches")

These patterns show that predictable suffix alterations are frequently involved in gender inflection and pluralisation.

2) Urdu Verbs: Urdu verbs (فعل) are substantially more complex. Verbs inflect for:

- tense/aspect (habitual, perfective, progressive),
- voice/causativity (base, causative, indirect or double causative),

- agreement with gender and number,
- politeness/honorific level.

A single root can produce well over a dozen different word forms, even when periphrastic constructions are disregarded. For instance, beginning with the lemma اُبل (/ubal/, "to boil"), we may observe:

- **Infinitive (base):** اُبلنا (ubal-naa, "to boil")
- **Direct causative:** اُبالنا (ubaal-naa, "to cause [something] to boil")
- **Indirect / double causative:** اُبلوانا (ubalwaanaa, "to have [someone] cause [something] to boil")

These causative alternations are productive in Urdu and are crucial for lemmatization, because surface tokens like اُبلوانا and اُبالنا should both map back to the same underlying lemma.

a) Rule-based surface form expansion.: To bootstrap training data, we implemented a rule-driven generator for Urdu verbs. In our preprocessing notebooks, we defined functions that, given a base Urdu verb, automatically produce a fixed set (typically 16) of surface realizations by systematically attaching tense/person/number suffixes. Conceptually:

- For each combination of grammatical *person* (e.g., first vs. non-first), *number* (singular vs. plural), and *tense/aspect* (e.g., present vs. past), we append the appropriate suffix to the stem.
- We generate forms not only for the base stem but also for its causative and indirect causative variants (e.g., اُبالنا, اُبلوانا).

This produces a controlled grid of inflected forms that reflect how Urdu verbs are realized in real text. Each generated form is then paired with the same lemma in our dataset (the canonical root form). This rule-based expansion is how we obtained large verb coverage despite limited annotated data.

b) Stem ending classes.: During this generation step, we observed that Urdu verbs fall into two broad stem classes that take different suffixes:

- **Consonant-final stems**, i.e., stems that do *not* end with ا، و، ی، ے (alif, waw, ye, bari ye).
- **Vowel-final stems**, i.e., stems that *do* end in one of those characters.

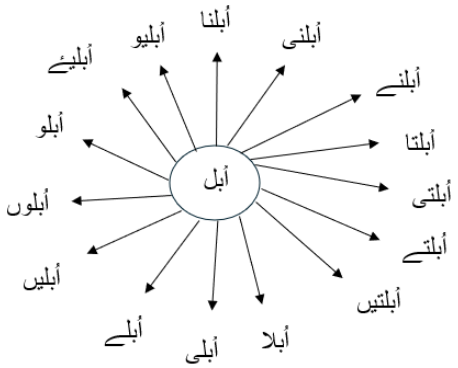

Figure 2: Urdu Morphology

Consider an Urdu lemma اُبل (/ubal/ – boil), which can produce multiple forms.

We explicitly separated these two groups in preprocessing, because the set of legal person/number suffixes differs depending on whether the stem ends in a vowel-like sound or a consonant stop. The following table illustrates this split:

Table I: Urdu Verb Stem Classes

Consonant-final stems	Vowel-final stems
پہچان (pehchaan – "recognize")	ھچکچا (hichkacha – "hesitate")
چور (chor – "steal")	نماؤ (numaao – "show off")
سمجھ (samajh – "understand")	الجھا (uljhaa – "confuse")

This vowel-final vs. consonant-final distinction is important because it determines which suffixes can attach without violating Urdu phonotactics. Our dataset-generation pipeline (the same pipeline used to build our training CSV) encodes this distinction: we automatically generated surface forms separately for each class, then flattened those forms into `(word, lemma)` pairs for model training.

B. Punjabi Morphology

Punjabi (in Shahmukhi script) exhibits similarly rich morphology, but with its own typological quirks. Punjabi and Urdu are closely related and share much vocabulary, but Punjabi shows strong case marking on nouns and highly grammaticalized politeness distinctions in verbs.

Figure **??** illustrates the spread of inflected forms radiating from a single Punjabi lemma. As in Urdu, many of these forms differ only by a few characters at the end of the word. For a lemmatizer, those small suffix changes must still resolve back to one canonical lemma.

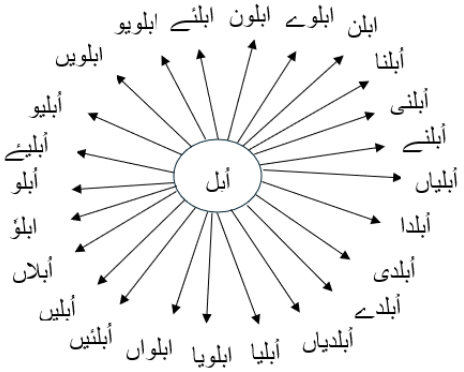

Figure 3: Punjabi Morphology

1) Punjabi Nouns: Punjabi nouns inflect for number (singular/plural), case (nominative, vocative, ergative, accusative/oblique), and gender (masculine/feminine). For example, consider گھوڑا (ghora – "horse"):

- **Nominative (sg/pl):** گھوڑا / گھوڑے
- **Vocative (sg/pl):** گھوڑیا / گھوڑیو

- **Ergative (sg/pl):** گھوڑے / گھوڑیاں
- **Accusative/Oblique (sg/pl):** گھوڑے / گھوڑیاں

Two observations are relevant for lemmatization: (1) many surface forms differ only by a short case/number suffix; (2) some plural/oblique forms are identical across cases (e.g., گھوڑیاں appearing in multiple roles). Our model therefore needs to learn to strip these endings and recover the base nominal form.

2) Punjabi Verbs: Punjabi verbs also demonstrate substantial morphological richness. Verb forms vary by tense/aspect (habitual, progressive, perfective), number, gender, and—critically—honorific level and person. Honorific morphology introduces surface forms that would never appear in Urdu but are very common in Punjabi.

For example, for the verb رنگنا (rungna – "to color"), the habitual aspect yields:

- Masculine singular (1st person): رنگدا (rungda)
- Masculine singular (2nd person, honorific): رنگو (rungo)
- Masculine singular (3rd person): رنگئیں (rungẽ / runge)
- Feminine singular (1st person): رنگدی (rungdi)
- Feminine singular (2nd person, honorific): رنگن (rungn)
- Feminine plural / polite plural: رنگئے (rungye)

These forms reflect *both* agreement (gender/number) and the levels of politeness. From a modeling point of view, Punjabi verb morphology produces a large number of morphologically near-duplicate forms that end in slightly different suffixes. The same is true of Urdu: small, word-final, character-level changes encode important grammatical information.

a) Rule-driven enumeration of Punjabi forms.: Just like with Urdu verbs, we systematically constructed Punjabi verb surface forms using a set of suffix rules and later exported them to spreadsheets. These sheets were then merged into a cohesive set of (lemma, word) pairs. This level of detail captures a broad range of Punjabi inflectional forms, including patterns of honorifics and plural forms that are rarely seen in the raw, unedited crawling text. For the sake of brevity, we will only provide the most dominant patterns here, while a complete account of all the constructed forms is available in Appendix **??**.

Table VIII captures the entire morphology of the Punjabi verb ابل ('to boil') and its forms produced by our system.

Punjabi verbs are morphologically rich, and this is reflected in the variety of forms that result from the combination of different tenses, aspects, genders, numbers, persons, and honorifics.

As detailed in Appendix **??** (Table VIII), we give all the forms of the Punjabi verb ابل ('to boil') that relate to tense, gender, number, person and honorifics including the entire morphological paradigm.

This paradigm illustrates how a single Punjabi verb can yield over a hundred inflected variants, highlighting the extreme morphological richness of the language and the motivation for a character-level lemmatizer.

C. Implications for Lemmatization

The patterns above have two major implications for our system design:

1) **Character-level modeling is necessary.** Many distinctions between forms are expressed as short, often non-segmenting suffixes. Token-level or stem-based approaches struggle when a single lemma explodes into dozens of forms that differ by 1–2 characters. Our model therefore operates purely at the character level, learning to map an inflected surface form back to its lemma one character at a time.
2) **Systematic suffix rules can bootstrap supervision.** Because Urdu and Punjabi inflection is highly regular within stem classes (e.g., consonant-final vs. vowel-final stems in Urdu verbs, or honorific suffixes in Punjabi verbs), we were able to script generators that produce full paradigms of plausible surface forms for each lemma. We then flatten these generated paradigms into paired training examples for the lemmatizer. This hybrid pipeline (linguistic rules → generated forms → supervised pairs) is how we produced large volumes of `word` → `lemma` data despite working in low-resource settings.

This morphological analysis directly informed both our dataset construction (Section V) and our model training (Section VII). In particular, it explains why Urdu and Punjabi benefit from a bilingual, character-level neural lemmatizer rather than purely rule-based or purely word-level systems.

III. Related Work

Morphological analysis and lemmatization have evolved significantly, from rule-based systems to modern multilingual neural architectures. Given the morphological richness of Urdu and Punjabi, prior techniques—whether finite-state, statistical, or neural—have faced specific challenges due to data scarcity, orthographic variability, and the absence of standardized corpora. This section reviews the key approaches to lemmatization and morphological analysis, with particular emphasis on methods relevant to low-resource and Indo-Aryan languages.

A. Traditional Rule-Based and Lexicon Approaches

Early morphological analyzers relied on hand-crafted lexica and finite-state rules. Classic two-level morphology used finite-state transducers (FSTs) to encode stems and affix rules [27], [28]. For Indo-Aryan languages, numerous such systems were constructed. For example, a Sanskrit analyzer used a lexicon-plus-affix approach to strip endings [**?**]. Paradigm-driven analyzers have been applied in South Asian languages [29], [30]. Dictionary-based FST analyzers were also implemented for Hindi [31], [32]. A similar paradigm approach for Dogri nouns achieved up to 90% F1 using hybrid rule and machine-learning methods [33]. These systems can be precise, but they require exhaustive lexicons and extensive handcrafted rules, which makes them difficult to scale to morphologically

rich and low-resource languages like Urdu and Punjabi, where orthographic and inflectional variation is high.

B. Statistical and Machine-Learning Methods

Before deep learning, researchers used statistical classifiers or CRFs to predict lemmas and features [34]. Such methods generalize beyond explicit rules but require annotated data [32], [34]. However, they often underperform on morphologically rich, low-resource languages compared to modern neural approaches [35], [36]. While statistical models capture generalization beyond rules, they still rely heavily on annotated corpora—resources that remain scarce for Urdu and Punjabi.

C. Neural Sequence Models (RNN/LSTM/BiLSTM)

Recurrent neural architectures have become standard for morphology. Encoder–decoder RNNs, particularly LSTMs with attention, dominate inflection and lemmatization tasks [35], [36]. LSTM-based lemmatizers for Bangla achieve about 95.8% character accuracy [7], while BiLSTM models for Urdu reach around 96% accuracy [37]. The Gujarati GujMORPH corpus supports a BiLSTM baseline with 89% accuracy [38]. Neural architectures have also supported Malayalam [39] and Sinhala [40]. These models learn morphological transformations directly from data, performing well when at least moderate amounts of annotated data are available. These architectures directly inform our bilingual lemmatizer design, as we similarly employ character-level sequence-to-sequence RNN variants (SimpleRNN, LSTM, GRU, BiLSTM) to map inflected surface forms back to lemmas.

D. Transformers and Pretrained Models

Transformer architectures and pretraining have significantly advanced morphological analysis. Byte-level models such as ByT5-Sanskrit achieve state-of-the-art results for Sanskrit segmentation and lemmatization [41]. BanglaT5 fine-tuned on a large dataset reports 94.4% accuracy [42]. A multilingual universal lemmatizer leveraging contextual features outperformed baselines across 52 languages [43]. Joint tagger–lemmatizer frameworks enhance morphological prediction [44]. Transformer-based approaches have also proven successful for Kannada [45]. Multilingual models like mT5 and BERT-Te have demonstrated strong cross-lingual performance. Despite their power, transformer models require large-scale, high-quality annotated corpora and benefit from standardized orthography. Such resources are not yet available in sufficient volume or script coverage for Shahmukhi-script Urdu and Punjabi. As a result, recurrent sequence models remain more practical for our setting.

E. Cross-Lingual and Transfer Methods

Cross-lingual transfer has proven effective in low-resource morphology. Attention-based inflection generation models achieved 15% accuracy improvements through transfer [46]. Neural factor-graph taggers sharing features across languages improved tagging accuracy [47]. Multi-task RNNs have been shown to transfer morphological knowledge effectively [48]. Encoder–decoder architectures trained on related languages enhanced lemmatization performance [49]. Multilingual T5 models improved gender–number–person tagging by about 7% [50]. These results confirm that typologically related languages benefit from shared multilingual representations [51], [52], supporting our hypothesis that Urdu and Punjabi—closely related Indo-Aryan languages—can benefit from a shared bilingual lemmatization framework.

F. Morphological Analysis in South Asian Languages

Recent work targets Indo-Aryan and Dravidian languages. For Hindi and Urdu, both dictionary-based and neural approaches exist [31], [37]. Neural lemmatizers such as BaNeL and BanglaLem exceed 90% accuracy for Bangla [7], [42]. Marathi remains supported by finite-state analyzers [30]. Gujarati benefits from neural segmentation models [38], while Oriya morphology uses paradigm-based approaches [29]. Neural and rule-based methods are applied for Malayalam and Sinhala [39], [40]. Tamil's ThamizhiMorph provides an open-source FST [53], and transformer-based models address Telugu and Kannada [54], [45]. Despite progress, many South Asian languages remain low-resource. In such cases, unsupervised segmentation approaches are effective [55], [56].

Overall, research has evolved from finite-state and dictionary-based methods [27], [28] to neural architectures [35], [36]. Cross-lingual and multilingual transformers alleviate data scarcity issues [46], [50]. For South Asian languages, hybrid combinations of rule-based systems and neural models continue to deliver the best performance across morphological analysis tasks [37], [7], [45].

In summary, while substantial progress has been made across South Asian languages using both rule-based and neural methods, no prior study has explored a unified bilingual lemmatizer for Urdu and Punjabi in Shahmukhi script. Building on these developments, our work integrates linguistic rule awareness—through dataset generation—with deep learning-based character-level sequence modeling, addressing both data scarcity and morphological complexity in these two closely related languages.

IV. Methodology

This work explores character-level lemmatization for Urdu–Punjabi words using four recurrent neural network (RNN) architectures: SimpleRNN, LSTM, Bidirectional LSTM (BiLSTM), and GRU. The approach involves data preprocessing, sequence encoding, model training, and evaluation. All models were implemented in Python using TensorFlow [**?**] and scikit-learn [57].

A. Dataset Preparation

A publicly available dataset titled *Urdu–Punjabi Merged Dataset* [58] was used in this study. The dataset was created and released by Malaika Basharat and Umar Waris on the Kaggle platform in 2024 to facilitate research on cross-lingual lemmatization tasks. It contains two columns: `Word` and

Lemma, where each record represents a pair (w_i, l_i) corresponding to a surface word and its lemma. The dataset was loaded using the pandas library [59] and divided into training and testing subsets in an 80:20 ratio using the `train_test_split()` function from scikit-learn [57]. A fixed random seed was applied to ensure reproducibility across experiments.

B. Character-level Tokenization and Padding

Character-level tokenization was applied to both words and lemmas using TensorFlow Keras' `Tokenizer` with `char_level=True`. The resulting vocabulary size was denoted as V. Each word or lemma sequence was converted to integer indices and padded to a fixed maximum length T_{max}:

$$\begin{aligned} X &= \text{pad\_sequences}(\text{Tokenizer}(W)), \\ Y &= \text{pad\_sequences}(\text{Tokenizer}(L)) \end{aligned} \tag{1}$$

where W and L denote lists of input words and lemmas respectively. Padding ensures that all sequences have equal length, enabling batch training. The target labels Y were reshaped to $(N, T_{\text{max}}, 1)$ to match the requirements of the `sparse_categorical_crossentropy` loss.

C. Model Architectures

Each model begins with an embedding layer that maps each input token index $x_t \in \{1, 2, \ldots, V\}$ to a dense vector:

$$e_t = Ex_t, \quad E \in \mathbb{R}^{V \times d} \tag{2}$$

where $d = 64$ is the embedding dimension. This sequence $\{e_t\}_{t=1}^{T_{\text{max}}}$ is then fed into a recurrent layer, followed by a TimeDistributed dense layer with softmax activation for character prediction.

1) Simple RNN: The SimpleRNN updates its hidden state h_t as:

$$h_t = \tanh(W_h h_{t-1} + W_x e_t + b_h) \tag{3}$$

and predicts the output y_t:

$$\hat{y}_t = \text{softmax}(W_y h_t + b_y) \tag{4}$$

This structure captures short-term dependencies but struggles with long-term context due to vanishing gradients [60].

2) Long Short-Term Memory (LSTM): To address this limitation, LSTM introduces gating mechanisms to control information flow:

$$f_t = \sigma(W_f e_t + U_f h_{t-1} + b_f) \quad \text{(forget gate)} \tag{5}$$

$$i_t = \sigma(W_i e_t + U_i h_{t-1} + b_i) \quad \text{(input gate)} \tag{6}$$

$$o_t = \sigma(W_o e_t + U_o h_{t-1} + b_o) \quad \text{(output gate)} \tag{7}$$

$$\tilde{c}_t = \tanh(W_c e_t + U_c h_{t-1} + b_c) \quad \text{(cell candidate)} \tag{8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(cell update)} \tag{9}$$

$$h_t = o_t \odot \tanh(c_t) \tag{10}$$

where $\sigma(\cdot)$ denotes the sigmoid activation, and $\odot$ represents element-wise multiplication.

3) Bidirectional LSTM (BiLSTM): The BiLSTM processes the sequence in both forward and backward directions:

$$\overrightarrow{h_t} = \text{LSTM}_f(e_t, \overrightarrow{h_{t-1}}) \tag{11}$$

$$\overleftarrow{h_t} = \text{LSTM}_b(e_t, \overleftarrow{h_{t+1}}) \tag{12}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{13}$$

Concatenating the two hidden states allows the model to capture both past and future context at each timestep.

4) Gated Recurrent Unit (GRU): The GRU simplifies the LSTM by merging the forget and input gates:

$$z_t = \sigma(W_z e_t + U_z h_{t-1}) \quad \text{(update gate)} \tag{14}$$

$$r_t = \sigma(W_r e_t + U_r h_{t-1}) \quad \text{(reset gate)} \tag{15}$$

$$\tilde{h}_t = \tanh(W_h e_t + U_h(r_t \odot h_{t-1})) \tag{16}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{17}$$

This structure retains efficiency while addressing vanishing gradients [61].

5) Output Layer: The TimeDistributed dense layer applies a softmax function at each timestep:

$$P(y_t \mid h_t) = \text{softmax}(W_o h_t + b_o) \tag{18}$$

producing the probability distribution over the vocabulary for each output position.

D. Loss Function and Optimization

The loss function used was sparse categorical cross-entropy:

$$\mathcal{L} = -\frac{1}{N T_{\text{max}}} \sum_{i=1}^{N} \sum_{t=1}^{T_{\text{max}}} \log P(y_{i,t} \mid x_i) \tag{19}$$

where N is the number of samples, T_{max} the sequence length, and $P(y_{i,t} \mid x_i)$ the predicted probability for the correct character. Optimization was performed using the Adam optimizer [62] with default parameters.

E. Evaluation Metrics

After training for 20 epochs with batch size 128 and validation split 0.2, performance was assessed using Accuracy, Precision, Recall, and F1-score computed via scikit-learn. The confusion matrix was also generated for qualitative assessment. The F1-score served as the primary selection metric for determining the best-performing model.

F. Visualization and Analysis

Training and validation curves were plotted for both accuracy and loss using Matplotlib [63]. Bar charts compared model-level metrics, and the best model's confusion matrix was visualized to assess prediction consistency.

This framework systematically compares multiple RNN-based architectures on character-level sequence modeling for lemmatization. It emphasizes reproducibility, interpretability, and rigorous quantitative evaluation.

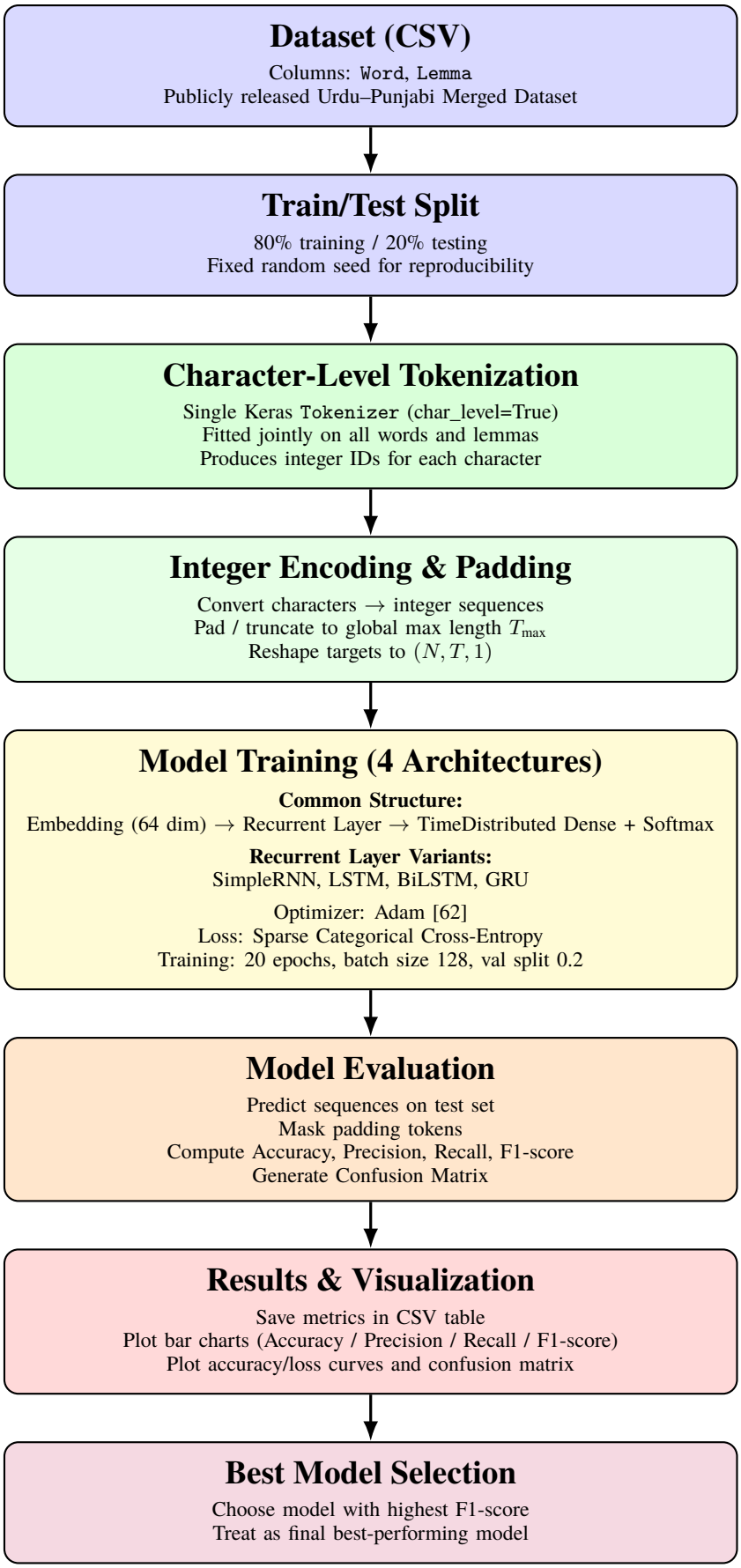

Figure 4: Full-page workflow diagram of the proposed character-level lemmatization pipeline, showing data preparation, preprocessing, model training, evaluation, and model selection steps.

V. Dataset Construction

The bilingual morphological corpus utilized in this work stems from the lemmatization task of the aligned Urdu and Punjabi resources constructed for bilingual morphological corpus. The Urdu portion comes from the noun and verb paradigms in the finite-state morphological analyzer data of Sara Hussain from 2004 [64] while Punjabi lexicon was constructed from linguistic corpora, digital dictionaries as well as Punjabi dictionaries and lexicons developed manually.

A. Dataset Composition

The very first dataset had around 275,000 records, 175,936 of which were Punjabi and just under 100,000 were Urdu. The words were paired and aligned with their respective lemmas to build the pairs (w_i, l_i) such that w_i denotes the inflected word and l_i the corresponding canonical lemma. In order to meet quality standards, the data needed to be pre-processed since the elements were aligned across two different languages.

B. Preprocessing and Cleaning

We noted during our word processing that duplicating word forms was a complication that was serious in nature. In our raw dataset, entries seemed remarkably similar, particularly with respect to inflected forms where only superficial orthographic features (punctuation, diacritics, case marking, etc.) differentiated forms. Such a high degree of word form similarity is problematic in the context of model training as the model is likely to consider such entries as distinct. This phenomenon is referred to as overfitting, whereby a model is said to overfit a dataset if it learns false associations or develops biases to certain word forms, resulting in the undermining of the model's generalizability.

To resolve the issue, duplicates were deleted in line with the following principles:

- **Identical Word Forms:** Pairs of word–lemmas that were considered near duplicates apart from spelling variations or punctuation were classified as duplicates and removed.
- **Minimizing Redundancy:** Redundant duplicates were removed in order to make certain that learned patterns do not consist of redundant and underrepresented patterns which can counter the model's training.

The final dataset contains **112,077 unique word-lemma pairs** of which **63,435 are in Urdu** and **48,642 in Punjabi**. This dataset was then randomly divided into 80% for training and 20% for testing to facilitate the thorough testing of lemmatization models.

C. Data Generation Using Morphological Rules

The dataset generation process was guided by predefined morphological rules for each language. Morphological paradigms for verbs and nouns were implemented programmatically to generate surface forms from canonical lemmas. For example, the *Punjabi non-past tense* and *Urdu verb inflectional* paradigms were derived using the rules presented in Table X and Table XII, respectively. These paradigms define the morphological transformations necessary to capture tense, aspect, person, and number variations.

D. Manual Verification and Annotation

Every generated word–lemma pair was checked and verified for both grammatical and morphological accuracy. For the purpose of dataset enhancement, part-of-speech (POS) tags were added wherever they were warranted. Manual review was particularly necessary for explaining or clarifying ambiguous lemmas and overlapping affix patterns within and across the two languages.

VI. Data Preprocessing

The preprocess stage transforms the raw Urdu-Punjabi word–lemma pairs into a formatable and numerable set for sequence-to-sequence learning. This stage validates the dataset and incorporates useful practices of artificial neural networks to ensure integration in model training step- achieving the most practice alignment of neural morphological processing [65], [66], [67], [68].

A. Data Loading and Splitting

The dataset used in this work is the publicly available *Urdu–Punjabi Merged Dataset* [58], which contains two columns: `Word` and `Lemma`. Each record represents a tuple (w_i, l_i) corresponding to a surface word and its canonical lemma.

The dataset was read using the pandas library [59] and divided into training and testing subsets using the `train_test_split()` function from scikit-learn [57], with an 80:20 ratio and a fixed random seed for reproducibility:

$$D = D_{\text{train}} \cup D_{\text{test}}, \quad |D_{\text{train}}| = 0.8|D|, \quad |D_{\text{test}}| = 0.2|D|.$$

Using a fixed random seed is standard for reproducible machine learning experiments [57].

B. Character-level Tokenization

For sequence modeling at the character level, both words and lemmas were tokenized using the Keras `Tokenizer` configured as:

`char_level=True`, `filters=None`, `lower=False`.

This approach recognizes every character distinctively as a token and observes case sensitivity.

Character-level modeling is dominantly used in the characterization of morphologically rich or resource-poor languages due to their aptitude to capture intricate patterns of inflection and resolve the out-of-vocabulary issue [65], [66].

To promote a single character set used in both input and output sequences, the token romizer was a joint fit to the aggregate of all training words and lemmas. This approach is standard in neural morphological transduction tasks [67], [68].

Following fitting, every lemma and word string was converted into a sequence of integer indices. The vocabulary size V was obtained as the summation of the unique set of characters with one extra padding token.

C. Sequence Padding and Length Normalization

Since the number of words in both Urdu and Punjabi can differ in size, all sequences were truncated or padded to the computed common maximum length T_max which is given by,

$$T_{\max} = \max\left(\max_{x \in X} |x|, \max_{y \in Y} |y|\right),$$

for the sequences $|x|$ and $|y|$ are the input (word) and output (lemma) sequences, respectively. To record the effect of model alignment and model learning, two padding strategies were used.

The preprocessing in this case was performed using Keras as shown in the following code.

```
pad_sequences(..., maxlen = T_{max},
padding = 'pre', truncating = 'pre')

pad_sequences(..., maxlen = T_{max},
padding = 'post', truncating = 'post')
```

Pre-padding inserts PAD symbols at the beginning of each sequence, aligning informative characters to the right, while post-padding appends PAD tokens at the end. Both approaches are commonly adopted in character-level and sequence-to-sequence models such as Keras and TensorFlow [67], [68], [69]. Empirically, post-padding preserves suffix cues, which are linguistically salient for Indo-Aryan languages like Urdu and Punjabi where inflectional morphology is predominantly realized at word endings [70], [71].

D. Target Reshaping

To align with the requirements of Keras' `sparse_categorical_crossentropy` loss, the target arrays were reshaped to include an additional singleton dimension:

$$Y \in \mathbb{R}^{N \times T_{\max} \times 1},$$

where N denotes the number of samples. The padding token was assigned index 0 (`PAD_INDEX = 0`), and these positions were masked during evaluation to prevent bias in metric computation. Masking padded timesteps during evaluation is standard in character-level sequence modeling tasks [67].

E. Rationale for Duplicate Removal

Prior to tokenization, duplicate entries were removed to ensure that the model learned from unique, linguistically valid mappings rather than redundant patterns. Since many inflected forms in Urdu and Punjabi share identical surface realizations (e.g., orthographically identical forms across genders or cases), such duplicates would bias the model toward memorization rather than generalization. Filtering identical word–lemma pairs aligns with best practices in morphological inflection datasets, where only unique mappings are retained to avoid data imbalance [67]. This ensured a cleaner dataset containing approximately 112,000 unique word–lemma pairs, providing more balanced and representative training data.

This preprocessing pipeline guarantees reproducibility and consistent character-level input representation across all model variants.

VII. Model Training

Following the preprocessing pipeline described in Section VI, the cleaned and tokenized word–lemma pairs were transformed into padded integer sequences and used to train four recurrent neural network architectures: SimpleRNN, LSTM, Bidirectional LSTM (BiLSTM), and GRU. All models were implemented using the TensorFlow/Keras framework [**?**] and executed on a CPU backend. The theoretical formulations of the recurrent architectures are provided in Section **??**; this section focuses on the practical training configuration, optimization, and experimental setup.

A. Architecture Definition

Each model follows a consistent sequence-to-sequence architecture consisting of three main components:

- **Embedding Layer:** Converts discrete character indices into dense 64-dimensional continuous vectors. The embedding matrix $E \in \mathbb{R}^{V \times 64}$ is learned jointly with the model parameters.
- **Recurrent Layer:** Encodes the temporal dependencies between characters using one of four recurrent variants—SimpleRNN [72], LSTM [60], GRU [61], or BiLSTM [73]. Each recurrent unit contains 64 hidden dimensions and returns sequences to maintain alignment with character-level outputs.
- **Decoder Layer:** A `TimeDistributed(Dense(V, softmax))` layer projects each timestep's hidden state to a probability distribution over the vocabulary size V, producing the predicted character at each position.

All models are constructed with the same embedding and decoder architectures, thus leading to performance difference ensue solely from the type of recurrent units and not from the overall size and configuration of the models.

B. Compilation and Optimization

The loss function was defined as `sparse_categorical_crossentropy` as it allows comparison of predicted and target character indices and obviates the need for one-hot encoding. The optimization was done as per [62] using the Adam algorithm which is common in sequence modeling tasks [**?**], [67] and a learning rate of 0.001. During training, the main metric was accuracy, and additional metrics, which included Precision, Recall, and F1-score, were calculated after training using the scikit-learn library [57].

C. Training Procedure

Each architecture was trained independently using identical hyperparameters (Table II) to ensure comparability. The models were trained for 20 epochs with a batch size of 128 and a validation split of 0.2. All experiments were conducted using a fixed random seed to guarantee reproducibility. Training time and parameter counts were logged automatically for each model configuration.

The workflow was as follows:

1) Initialize model using the corresponding recurrent unit type.
2) Compile using the specified loss and optimizer.
3) Train on preprocessed sequences $(X_{\text{train}}, Y_{\text{train}})$ for 20 epochs.
4) Validate on held-out 20% of the training data.
5) Evaluate the final model on the test set $(X_{\text{test}}, Y_{\text{test}})$ to obtain accuracy, precision, recall, and F1-score.

This standardized pipeline ensured a fair comparison among architectures under identical data and optimization settings.

D. Logging and Reproducibility

Both training and validation phases documented training histories, including recordings of accuracy and loss for each epoch. Assessment results included CSV summaries, for which metrics were visualized in the form of accuracy and loss curves and bar plots created in Matplotlib [63]. All the experimentation was conducted in a Python 3 environment on a machine equipped with an Intel Core i5-1235U (12th Gen) CPU and 24 GB of RAM. To control for stochastic variation and ensure reproducibility across reruns, random seeds were set and maintained throughout the process.

E. Training Configuration

Table II summarizes the hyperparameters and system environment used across all experiments.

This controlled setup enabled a systematic evaluation of four recurrent architectures for Urdu–Punjabi lemmatization. The resulting trained models and evaluation results form the basis for the comparative analysis presented in Section VIII.

VIII. Model Evaluation

After training, each model was evaluated on cross-validation for model selection with respect to a unified evaluation framework coded in Python. The evaluation concerned character-level accuracy, precision, recall, and F1-score, as is common for tasks involving morphological generation and lemmatization [67], [**?**].

A. Evaluation Pipeline

Following training, predictions $\hat{Y}$ were generated for the test set X_{test} using `model.predict()`. For each word, a probability distribution for every character position was generated over the vocabulary. The system defined the predicted token as:

$$\hat{y}_{i,t} = \arg\max_{c \in V} P(y_t = c \mid x_i),$$

where V is the shared character vocabulary True and predicted sequences were flattened and masked, eliminating the padding symbols for metric calculation. This guaranteed that the evaluation was performed on valid characters, rather than inflated scores due to padding.

B. Metrics Computation

The function `evaluate_sequence_model()` calculated the following metrics for each architecture:

Accuracy, Precision, Recall, and F1-score.

All metrics were computed at the **character level**, using weighted averaging to handle class imbalance:

$$\text{Precision} = \frac{\sum_i \text{TP}_i}{\sum_i (\text{TP}_i + \text{FP}_i)}, \tag{20}$$

$$\text{Recall} = \frac{\sum_i \text{TP}_i}{\sum_i (\text{TP}_i + \text{FN}_i)}, \tag{21}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{22}$$

The `scikit-learn` evaluation suite was used to perform these calculations [57]. The calculations were organized in a structured DataFrame for ease of reproducibility and analysis, which was then exported to `model_comparison.csv`.

C. Confusion Matrix Analysis

To visualize per-character misclassifications for each model, confusion matrices were created for the flattened, masked predictions:

$$\mathbf{C}_{i,j} = \text{count}(y_t = i \wedge \hat{y}_t = j),$$

with true characters displayed as rows and predicted characters as columns. Such matrices reveal misclassifications where the system seems to confuse certain characters, particularly the pairs of glyphs from Urdu and Punjabi that look the same, as well as issues involving diacritics. For the model that achieved the highest F1-score, the confusion matrix was rendered in Matplotlib for qualitative analysis.

D. Model Comparison and Selection

The architecture: SimpleRNN, LSTM, BiLSTM, and GRU was evaluated meaningfully and uniformly with respect to preprocessing and hyperparameter settings. The results captured included corresponding metrics of accuracy, precision, recall, F1-score, number of learnable parameters, and total training time. The best model was thus selected automatically as:

$$M^* = \arg\max_{m \in \mathcal{M}} \text{F1}(m),$$

with the trained models being the set $\mathcal{M}$.

The chosen model had its performance metrics along with its visualizations (bar plots, loss/accuracy curves, and confusion matrix) all preserved for later reference (Section IX).

E. Reproducibility

In order to achieve reproducibility, all evaluation experiments were conducted using fixed random seeds and a deterministic configuration in TensorFlow. Reproducibility was further ensured through logged and version-controlled evaluation outputs (metric plots, and matrices), which provided complete traceability of the reported results.

To maintain consistent, equitable, and reproducible performance evaluation in all model architectures, the implemented pipeline and evaluation setup were closely aligned.

Table II: Training Configurations.

Parameter	Value	Description
Epochs	20	Number of full training passes
Batch Size	128	Samples per weight update
Validation Split	0.2	Data reserved for validation
Optimizer	Adam	Adaptive Moment Estimation [62]
Learning Rate	0.001	Default TensorFlow setting
Loss Function	Sparse Categorical Cross-Entropy	Token-level objective
Masking	Enabled	Exclude PAD (index 0) tokens
Metrics	Accuracy, Precision, Recall, F1	Character-level metrics
Framework	TensorFlow/Keras	Python 3 runtime
Hardware	Intel Core i5-1235U (12th Gen), 24 GB RAM	Trained using TensorFlow CPU backend

IX. Results

The assessment was performed independently to study the impacts of the temporal alignment gap on convergence, generalization, and character-level accuracy in the different padding configurations. This section demonstrates the results comparison for the various recurrent architectures (SimpleRNN, LSTM, GRU, BiLSTM) considering the two sequence-alignment techniques, **pre-padding** and **post-padding** . In an effort to control the impacts of the padding scheme and the design of the network for each experiment, the configurations, datasets, and training conditions (see Section II) were held constant.

A. *Pre-padding Results*

Table III and Figure 5 analyse performance of models when padding tokens are added *before* the sequences. Rather than faster convergence, models were slower to converge, and there were instances of gradient propagation saturation as a consequence of early timesteps. Of all the variants, the **BiLSTM** model still achieved the highest accuracy and F_1-score, providing evidence of its capacity to capture contextual information bidirectionally.

However, the performance gap between LSTM and GRU remained moderate, suggesting that even unidirectional models can partially capture morphological dependencies when trained sufficiently. Figure 5 (top) plots accuracy over epochs, showing that the BiLSTM and GRU models exhibit smoother learning trajectories, while SimpleRNN stagnates early due to vanishing gradients. The lower panel of Figure 5 presents the confusion matrix for the BiLSTM model, illustrating sparse off-diagonal errors. Most misclassifications occur between morphologically similar inflections (e.g., Urdu plurals ات/ں) or rare Punjabi honorific verb forms. This behavior aligns with prior findings that unidirectional recurrent networks struggle with long-range dependencies in morphologically rich languages [35], [74]. Under the **pre-padding** configuration, all models exhibited slower and less stable convergence, with higher residual validation loss across epochs, confirming the inefficiency of front-aligned padding for character-level lemmatization.

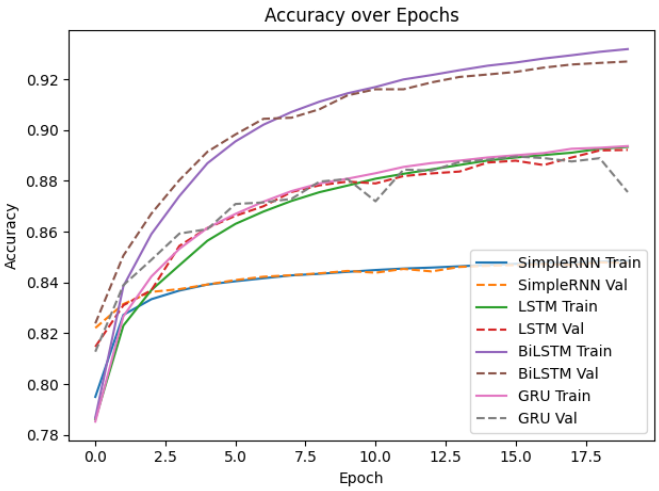

Figure 5: Training and validation accuracy across epochs under the **pre-padding** configuration. The BiLSTM shows superior convergence among recurrent models, though overall performance is lower than post-padding due to early truncation effects.

Figure 7 illustrates the training and validation loss curves for all recurrent architectures under the pre-padding setup. Across all models, loss values decrease gradually but remain noticeably higher than their post-padding counterparts, indicating slower convergence. The SimpleRNN demonstrates the highest residual loss, suggesting inadequate temporal credit assignment due to vanishing gradients at earlier timesteps. Both LSTM and GRU show steady, monotonic improvement, yet their validation losses plateau prematurely, reflecting in-

Table III: Model performance under **pre-padding** strategy.

Model	Acc.	Prec.	Rec.	F_1	Params	Time (s)
SimpleRNN	0.4767	0.5462	0.4767	0.5075	15,222	290.98
LSTM	0.6283	0.6893	0.6283	0.6559	39,990	542.90
GRU	0.5851	0.6277	0.5851	0.6040	31,926	577.29
BiLSTM	**0.7399**	**0.7591**	**0.7399**	**0.7485**	**76,470**	**867.30**

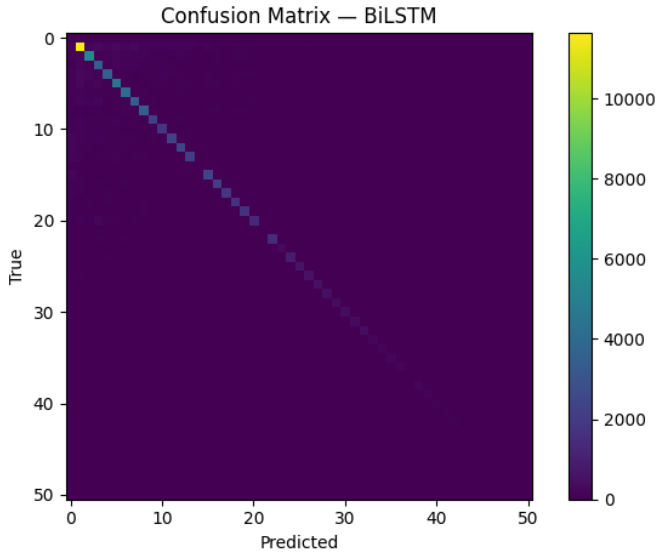

Figure 6: Character-level confusion matrix for the **BiLSTM** model under the **pre-padding** configuration. While the matrix is largely diagonal, minor off-diagonal dispersion indicates occasional misalignment in character prediction due to leading zero-padding, resulting in slightly lower performance compared to post-padding.

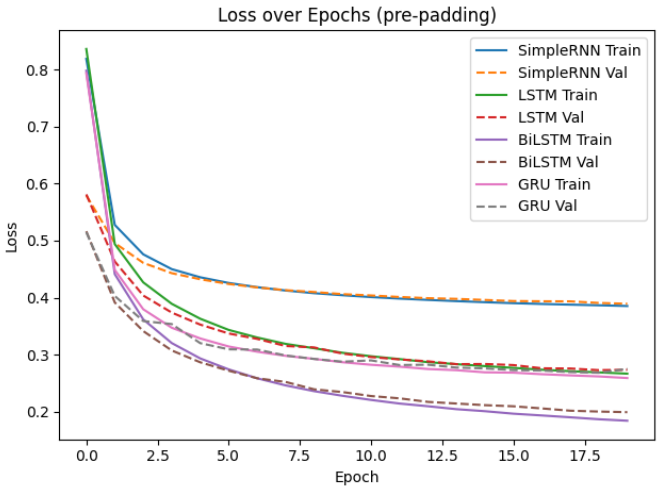

Figure 7: Loss over epochs (pre-padding).

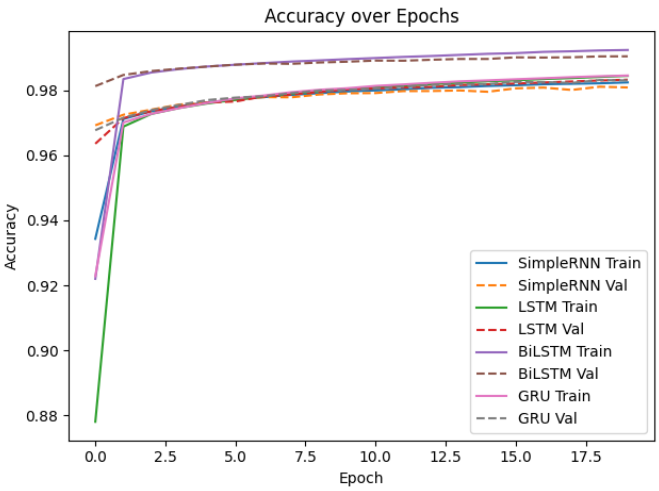

Figure 8: The post-padding setup complete on all recurrent models. Of all the models the BiLSTM achieves the fastest convergence and the highest validation accuracy.

complete long-range dependency learning. Among all configurations under pre-padding, the BiLSTM has the least overall loss; however, the distance between the training and validation curves indicates slight overfitting.

As these trends suggest, the front-aligned padding disrupts the flow of gradients through the first recurrent steps, which negatively impacts the model's efficacy in sequence modeling at the character level.

All architectures show significant pass epochs losses and convergence stabilization depicted in Figure 10. BiLSTM's validation loss is minimum and train and validation loss curves nearly perfectly overlap resulting in no overfitting and strong generalization capability. Even though GRU provides efficient convergence, it validation loss settles slightly higher than BiLSTM due to minor context retention long-term trade-offs. SimpleRNN, in contrast to other architec-

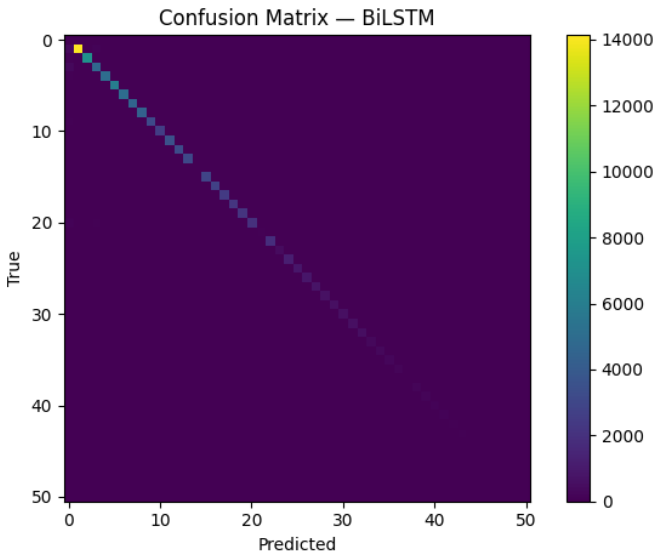

Figure 9: Character-level confusion matrix for BiLSTM model under post-padding settings. The results show a considerable length diagonal which highlights strong agreement between predicted and true lemma characters. This result shows strong sequential alignment which indicates minimal confusion between classes.

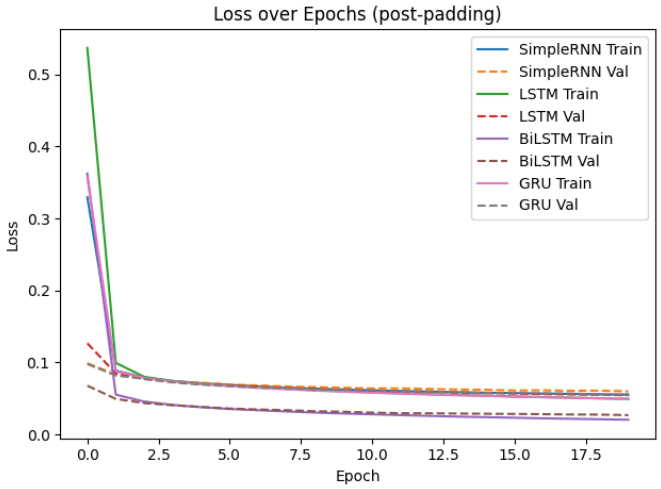

Figure 10: Training/validation loss for all architectures under **post-padding**.

tures, continues to converge slowly and eventually halts which reiterates it's unsuitability for morphologically rich, variable length sequences. Apart from this, which is encouraged, the post-padding configuration provides quicker and more secure optimization for Indo-Aryan character-level morphologically modeling suffixing.

B. *Post-padding Results*

The **post-padding** configuration yielded substantial improvement across all models, as summarized in Table IV and Figure 8. By appending padding tokens *after* the sequence, models preserved prefix information and avoided truncated context during temporal updates. As a result, convergence was faster and validation accuracy stabilized within the first five epochs.

Figure 10 demonstrates clear improvement: the BiLSTM's accuracy curve rises steeply in the first three epochs and reaches near-saturation by epoch ten, while its confusion matrix is highly diagonal with minimal dispersion. The stronger performance of post-padding validates that preserving sequential integrity improves backpropagation efficiency and temporal alignment, especially in suffix-heavy scripts.

To analyze convergence behavior, Figure 10 shows the training and validation loss across epochs for all recurrent architectures under the post-padding configuration. All models exhibit rapid initial loss reduction within the first two epochs, stabilizing after approximatel ten epochs. Among them, the BiLSTM achieves the lowest and most consistent validation loss,confirming its superior generalization capability.

C. *Comparative Discussion*

Across both padding schemes, the **BiLSTM consistently outperforms** other models in F_1-score and accuracy, confirming the utility of bidirectional context modeling for lemmatization in morphologically complex languages. Although the GRU achieves competitive accuracy, its recall remains slightly lower, indicating it occasionally underfits rare morphological variants. Training times scale approximately with parameter count ($O(U^2)$) as shown in Tables III and IV, reflecting the higher computational demand of bidirectional architectures. Nonetheless, BiLSTM provides the most stable convergence and generalization, justifying its selection for the final production model. Overall, these results demonstrate that the *post-padding* sequence alignment strategy significantly enhances model performance, and that bidirectional recurrent networks are best suited for Urdu–Punjabi lemmatization due to their ability to capture prefix and suffix dependencies in parallel temporal directions. With respect to the data, the BiLSTM with post-padding obtained an F1-score of 0.9846, which is an about 0.6% relative improvement over the GRU's 0.9825 and 3.5% over the unidirectional LSTM's 0.9796. Compared to the pre-padding configuration, all architectures show substantial performance gains, with BiLSTM improving from 0.7485 to 0.9846—a 31.5% relative increase. These margins confirm that both bidirectionality and suffix-preserving alignment jointly enhance character-level generalization.

X. Discussion

Neural sequence models are much better than rule-based and dictionary-based lemmatizers for Urdu and Punjabi, which are Indo-Aryan languages with complicated morphology.

BiLSTM models being the best models can be attributed to the importance of estimating bidirectional contextual dependencies necessary to model suffixal morphology and the inflectional variability across word forms [35], [74].

Table IV: Model performance under **post-padding** strategy.

Model	Acc.	Prec.	Rec.	F_1	Params	Time (s)
SimpleRNN	0.9715	0.9900	0.9715	0.9803	15,222	292.99
LSTM	0.9696	0.9907	0.9696	0.9796	39,990	479.53
GRU	0.9758	0.9897	0.9758	0.9825	31,926	547.25
BiLSTM	**0.9782**	**0.9913**	**0.9782**	**0.9846**	**76,470**	**867.40**

Deep learning approaches, in contrast to stitched-together approaches based on morphology and lexicon rules or, for example, lexicon rule [64], [75], generalizes to unseen tokens better and manual effort for the former is dominantly less compared to the latter.

Using character-level recurrent models in distributed representation learning is a widely used technique to learn to morphologically and orthographically encode the hidden layers of a model and learn the overlapping ortho-phoneticity [68], [76].

Despite strong overall model performance, there are still some unresolved linguistic problems.

For compound agglutinative structures, and for code-mixed cases where contextual cues are weak or where languages overlap, errors are frequent [70], [77].

Expanding the training corpus and adding more auxiliary signals of part of speech and morphology tags will be the focus for adding to unsupervised data to likely improve performance and abstraction on sequence length with proposed transformer-like models [78], [79].

Subword-based training and unsupervised data will likely help with inflection coverage and irregular inflectional forms in the model.

These observations confirm the computational efficiency of recurrent architectures. More importantly, they indicate the potential of these architectures to generalize across morpho-phonemic variation. This indicates a significant linguistics consequence.

Linguistic Consequence:

Linguistically, the results confirm the morphological and orthographic similarity of Urdu and Punjabi, two languages of the Indo-Aryan family, and their historical connection [80], [81]. The performance of the BiLSTM is noteworthy, suggesting that recurrent architectures can capture the implicit learning of sub-character shared correspondences between the Perso-Arabic (Urdu) script and the Shahmukhi (Punjabi) script without any transliteration and alignment to guide the learning.

This opens directions for extending the current framework to other Indo-Aryan under-resourced languages, like Saraiki, Hindko, and Kashmiri, which have limited morphological parallel resources [58], [70].

The research enables us to broaden the contribution beyond modeling accuracy, especially for the NLP ecosystem of low-resourced South Asian languages.

For instance, integration of the trained lemmatizer to downstream tasks of lemmatization like machine translation, information retrieval, and sentiment analysis will be made possible by our release of the lemmatizer as a Python `pip` package.

This study demonstrates, for the first time, that advanced deep neural architectures can learn to lemmatize languages morphologically rich and low-resourced, and thereby, advances the field of computational morphology. This provides a new benchmark for cross-lingual generalization within the Indo-Aryan languages.

Appendix A
Tokenizer Character Mapping

Table V: Examples of Urdu suffix generation during morphological analysis.

Root Word	Derived Word	Suffix
دھمکا	دھمکایا	یا
ورغلا	ورغلائی	ئی
آبدوز	آبدوزوں	وں

Table VI: Urdu Verb Groups: Consonant-Ending vs. Vowel-Ending Forms.

Consonant-ending Verbs	Vowel-ending Verbs
پہچان (*pehchaan*, "recognize")	ہچکچا (*hichkacha*, "hesitate")
چور (*chor*, "steal")	نماؤ (*numaao*, "show off")
سمجھ (*samajh*, "understand")	الجھا (*uljhaa*, "confuse")

Appendix B
Punjabi Verb Forms

Appendix C
Urdu Verb Forms

Table VII: Complete mapping of characters to indices used by the tokenizer.

Index	Character	Index	Character
1	ا	2	و
3	ی	4	ن
5	ر	6	ک
7	ل	8	ه
9	ت	10	د
11	م	12	س
13	ب	14	ں
15	پ	16	ٹ
17	چ	18	گ
19	ج	20	ہ
21	ے	22	ڑ
23	ئ	24	ڈ
25	ش	26	ف
27	ق	28	ُ
29	خ	30	ح
31	ع	32	ز
33	ص	34	ط
35	ِ	36	آ
37	ؤ	38	غ
39	ض	40	ِ
41	ٰ	42	ث
43	ذ	44	ظ
45	ً	46	ء
47	ٔ	48	ٗ
49	ّ	50	ژ
51	ـئ	52	ّ
53	ة		

Table VIII: Punjabi Verb Forms — Infinitive and Habitual Variants of ابل (“to boil”).

No.	Word	Lemma	Morphological Features	Person / Hon.
1	ابل	ابل	V, Base	–
2	ابلنا	ابل	V, Inf, Masc, Sg	–
3	ابلنی	ابل	V, Inf, Fem, Sg	–
4	ابلنے	ابل	V, Inf, Fem, Sg	–
5	ابلیاں	ابل	V, Inf, Fem, Pl	–
6	ابلنا	ابل	V, Hab, Masc, Sg	1P
7	ابلنی	ابل	V, Hab, Fem, Sg	1P
8	ابلنے	ابل	V, Hab, Masc, Pl	1P
9	ابلیاں	ابل	V, Hab, Fem, Pl	1P
10	ابلدا	ابل	V, Hab, Masc, Sg	3P
11	ابلنا	ابل	V, Hab, Masc, Sg	2P, Hon1
12	ابلدا	ابل	V, Hab, Masc, Sg	2P, Hon1
13	ابلدا	ابل	V, Hab, Masc, Sg	1P
14	ابلدی	ابل	V, Hab, Fem, Sg	1P
15	ابلدی	ابل	V, Hab, Fem, Sg	2P, Hon1
16	ابلدے	ابل	V, Hab, Masc, Pl	3P
17	ابلدے	ابل	V, Hab, Masc, Sg	1P, Hon2
18	ابلدے	ابل	V, Hab, Masc, Pl	1P
19	ابلدے	ابل	V, Hab, Masc, Sg	2P, Hon2
20	ابلدے	ابل	V, Hab, Masc, Pl	2P
21	ابلدیاں	ابل	V, Hab, Fem, Pl	3P
22	ابلدیاں	ابل	V, Hab, Fem, Sg	1P, Hon2
23	ابلدیاں	ابل	V, Hab, Fem, Sg	2P, Hon2
24	ابلدیاں	ابل	V, Hab, Fem, Sg	3P, Hon2
25	ابلدیاں	ابل	V, Hab, Fem, Pl	1P
26	ابلدیاں	ابل	V, Hab, Fem, Pl	2P
27	ابل	ابل	V, Comd, Sg	2P, Hon1
28	ابلئیں	ابل	V, Comd, Sg	2P, Hon1
29	ابلیں	ابل	V, Comd, Sg	2P, Hon1
30	ابلّو	ابل	V, Comd, Sg	2P, Hon2

Table IX: Punjabi Verb Forms — Command and Past Variants of ابل (“to boil”).

No.	Word	Lemma	Morphological Features	Person / Hon.
31	ابلیو	ابل	V, Comd, Sg	2P, Hon2
32	ابلیا	ابل	V, Comd, Sg	2P, Hon2
33	ابلّو	ابل	V, Comd, Pl	2P
34	ابلاں	ابل	V, Comd, Sg	1P, Hon1
35	ابلیئے	ابل	V, Comd, Sg	1P, Hon2
36	ابلیئے	ابل	V, Comd, Pl	1P, Hon1
37	ابلیا	ابل	V, Past, Masc, Sg	–
38	ابلدی	ابل	V, Past, Fem, Sg	–
39	ابلدے	ابل	V, Past, Masc, Pl	–
40	ابلدیاں	ابل	V, Past, Fem, Pl	–
41	ابل	ابل	V, Past, Masc, Sg	1P
42	ابل	ابل	V, Past, Fem, Sg	1P
43	ابل	ابل	V, Past, Masc, Pl	1P
44	ابل	ابل	V, Past, Fem, Pl	1P
45	ابل	ابل	V, Past, Masc, Sg	2P, Hon1
46	ابل	ابل	V, Past, Masc, Sg	2P, Hon2
47	ابل	ابل	V, Past, Masc, Pl	2P
48	ابل	ابل	V, Past, Fem, Sg	2P, Hon1
49	ابل	ابل	V, Past, Fem, Sg	2P, Hon2
50	ابل	ابل	V, Past, Fem, Pl	2P, Hon2
51	ابل	ابل	V, Past, Masc, Sg	3P, Hon1
52	ابل	ابل	V, Past, Masc, Sg	3P, Hon2
53	ابل	ابل	V, Past, Masc, Pl	3P
54	ابل	ابل	V, Past, Fem, Sg	3P, Hon1
55	ابل	ابل	V, Past, Fem, Sg	3P, Hon2
56	ابل	ابل	V, Past, Fem, Pl	3P

Table X: Punjabi Verb Forms — Non-Past and Honorific Variants of ابل (“to boil”).

No.	Word	Lemma	Morphological Features	Person / Hon.
57	ابلواں	ابل	V, Non-Past, Masc, Sg	1P
58	ابلواں	ابل	V, Non-Past, Fem, Sg	1P
59	ابلواں	ابل	V, Non-Past, Masc, Pl	1P
60	ابلواں	ابل	V, Non-Past, Fem, Pl	1P
61	ابلویں	ابل	V, Non-Past, Masc, Sg	2P
62	ابلئیں	ابل	V, Non-Past, Masc, Sg	2P
63	ابلیں	ابل	V, Non-Past, Fem, Sg	2P
64	ابلو	ابل	V, Non-Past, Masc, Pl	2P
65	ابلیو	ابل	V, Non-Past, Masc, Pl	2P
66	ابلویو	ابل	V, Non-Past, Masc, Pl	2P
67	ابلیا	ابل	V, Non-Past, Masc, Pl	2P
68	ابلویا	ابل	V, Non-Past, Masc, Pl	2P
69	ابلیو	ابل	V, Non-Past, Fem, Pl	2P
70	ابلویو	ابل	V, Non-Past, Fem, Pl	2P
71	ابلیا	ابل	V, Non-Past, Fem, Pl	2P
72	ابلویا	ابل	V, Non-Past, Fem, Pl	2P
73	ابلو	ابل	V, Non-Past, Fem, Pl	2P
74	ابلوے	ابل	V, Non-Past, Masc, Sg	3P
75	ابلئے	ابل	V, Non-Past, Masc, Sg	3P
76	ابلوے	ابل	V, Non-Past, Fem, Sg	3P
77	ابلئے	ابل	V, Non-Past, Fem, Sg	3P
78	ابلون	ابل	V, Non-Past, Masc, Pl	3P
79	ابلن	ابل	V, Non-Past, Masc, Pl	3P
80	ابلون	ابل	V, Non-Past, Fem, Pl	3P
81	ابلن	ابل	V, Non-Past, Fem, Pl	3P
82	ابلون	ابل	V, Non-Past, Fem, Sg	3P, Hon2
83	ابلن	ابل	V, Non-Past, Fem, Sg	3P, Hon2

Table XI: Morphological paradigm of the Urdu verb اُبل (/ubal/, “to boil”) — Infinitive, Habitual, and Past forms.

No.	Lemma	Word Form	Morphological Features / Function	Person
1	اُبل	اُبل	Root / Base stem	–
2	اُبل	اُبلنا	Infinitive (base form)	–
3	اُبل	اُبلنی	Infinitive (feminine / nominalized)	–
4	اُبل	اُبلنے	Infinitive (plural / oblique)	–
5	اُبل	اُبلنین	Infinitive (extended plural)	–
6	اُبل	اُبلتا	Habitual, Masc, Sg	3P
7	اُبل	اُبلتی	Habitual, Fem, Sg	3P
8	اُبل	اُبلتے	Habitual, Masc, Pl	3P
9	اُبل	اُبلتیں	Habitual, Fem, Pl	3P
10	اُبل	اُبلا	Past, Masc, Sg	3P
11	اُبل	اُبلی	Past, Fem, Sg	3P
12	اُبل	اُبلے	Past, Masc, Pl	3P
13	اُبل	اُبلیں	Past, Fem, Pl	3P
14	اُبل	اُبلو	Imperative (neutral / plural)	2P
15	اُبل	اُبلو٘	Imperative (polite / honorific)	2P
16	اُبل	اُبلئیے	Imperative (very polite)	2P
17	اُبل	اُبال	Direct causative stem (“boil [something]”)	–
18	اُبل	اُبالنا	Causative infinitive	–
19	اُبل	اُبالنی	Causative infinitive (fem.)	–
20	اُبل	اُبا لنے	Causative infinitive (pl./oblique)	–
21	اُبل	اُبالنین	Causative infinitive (extended plural)	–
22	اُبل	اُبالتا	Causative, Masc, Sg, Habitual	3P
23	اُبل	اُبالتی	Causative, Fem, Sg, Habitual	3P

Table XII: Morphological paradigm of the Urdu verb اُبل (/ubal/, “to boil”) — Double Causative and Derived forms.

No.	Lemma	Word Form	Morphological Features / Function	Person
24	اُبل	اُبالتے	Causative, Masc, Pl, Habitual	3P
25	اُبل	اُبالتیں	Causative, Fem, Pl, Habitual	3P
26	اُبل	اُبالا	Causative, Past, Masc, Sg	3P
27	اُبل	اُبالو	Causative imperative	2P
28	اُبل	اُبالئیے	Causative imperative (polite)	2P
29	اُبل	اُبلوا	Double causative stem (“make [someone] boil [it]”)	–
30	اُبل	اُبلوانا	Double causative infinitive	–
31	اُبل	اُبلوانی	Double causative infinitive (fem.)	–
32	اُبل	اُبلوانے	Double causative infinitive (pl./oblique)	–
33	اُبل	اُبلوانین	Double causative infinitive (extended plural)	–
34	اُبل	اُبلواتا	Double causative, Masc, Sg, Habitual	3P
35	اُبل	اُبلواتی	Double causative, Fem, Sg, Habitual	3P
36	اُبل	اُبلواتے	Double causative, Masc, Pl, Habitual	3P
37	اُبل	اُبلواتیں	Double causative, Fem, Pl, Habitual	3P
38	اُبل	اُبلوایا	Double causative, Past, Masc, Sg	3P
39	اُبل	اُبلوائیں	Double causative, Past, Fem, Pl	3P
40	اُبل	اُبلواؤ	Double causative imperative	2P
41	اُبل	اُبلوائیے	Double causative imperative (polite)	2P

References

[1] S. Shaukat, M. Asad, and A. Akram, "Developing an urdu lemmatizer using a dictionary-based lookup approach," *Applied Sciences*, vol. 13, no. 8, p. 5103, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/8/5103

[2] R. Hafeez, M. W. Anwar, M. H. Jamal *et al.*, "Contextual urdu lemmatization using recurrent neural network models," *Mathematics*, vol. 11, no. 2, p. 435, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/2/435

[3] M. Fatima *et al.*, "Stemur: An automated word conflation algorithm for urdu," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 20, no. 3, pp. 1–23, 2021.

[4] M. Loponen, K. Järvelin *et al.*, "A dictionary-based lemmatizer for four languages," *Journal of Language Modelling*, vol. 4, no. 1, pp. 1–25, 2016.

[5] A. Wiemerslage *et al.*, "Morphological processing of low-resource languages: Where we are and what's next," in *Findings of the ACL*, 2022, pp. 1398–1410. [Online]. Available: https://aclanthology.org/2022.findings-acl.111

[6] A. Chakrabarty *et al.*, "Context-sensitive lemmatization using two successive bigrns," in *Proceedings of ACL*, 2017, pp. 148–158. [Online]. Available: https://aclanthology.org/P17-2014

[7] M. Islam *et al.*, "Banel: An encoder–decoder based bangla neural lemmatizer," in *Proceedings of LREC*, 2022. [Online]. Available: http://www.lrec-conf.org/

[8] A. Nair and V. Kumar, "A survey on morphological processing for low-resource languages," *Journal of Computational Linguistics and NLP*, vol. 5, no. 3, pp. 112–126, 2021.

[9] R. Puri, "A rule based approach for lemmatization of punjabi text documents," *International Journal of Computer Applications*, vol. 95, no. 23, pp. 1–4, 2014.

[10] V. Gupta, N. Joshi, and I. Mathur, "Rule-based urdu lemmatizer using suffix removal," *ICT for Sustainable Development*, pp. 381–388, 2016.

[11] T. Bergmanis and S. Goldwater, "Context sensitive neural lemmatization with lematus," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018, pp. 1391–1400. [Online]. Available: https://aclanthology.org/N18-1126

[12] D. Kondratyuk and M. Straka, "Lemmatag: Jointly tagging and lemmatizing for morphologically rich languages," in *EMNLP*, 2018, pp. 4921–4928. [Online]. Available: https://aclanthology.org/D18-1525

[13] N. Zalmout and N. Habash, "Joint diacritization, lemmatization, normalization, and fine-grained tagging of arabic," in *ACL*, 2020, pp. 734–744. [Online]. Available: https://aclanthology.org/2020.acl-main.67

[14] R. Sennrich and B. Haddow, "Linguistic input features improve neural machine translation," *arXiv preprint arXiv:1606.02892*, 2016. [Online]. Available: https://arxiv.org/abs/1606.02892

[15] S. Hussain, "Urdu morphology, orthography and lexicon processing," Ph.D. dissertation, National University of Computer and Emerging Sciences (CRULP), 2004. [Online]. Available: https://www.academia.edu/

[16] T. Ehsan, "A morphological analyzer for urdu verbs," 2016, researchGate preprint. [Online]. Available: https://www.researchgate.net/

[17] R. A. Bhat *et al.*, "The hindi/urdu treebank," in *Universal Dependencies Workshop*, 2017. [Online]. Available: https://universaldependencies.org/

[18] M. Humayoun, A. Ranta, and T. Hallgren, "Developing punjabi morphology, corpus and lexicon," in *LREC*, 2010. [Online]. Available: http://www.lrec-conf.org/

[19] M. Tahir *et al.*, "Adaptation and development of ud for punjabi (shahmukhi)," in *LREC-COLING*, 2024. [Online]. Available: https://aclanthology.org/

[20] M. K. Malik and S. Ahmed, "Punjabi machine transliteration from shahmukhi to gurmukhi script," in *International Conference on Computational Linguistics in Pakistan*, 2006.

[21] N. Durrani and S. Hussain, "Urdu word segmentation," in *NAACL-HLT*, 2010, pp. 528–536. [Online]. Available: https://aclanthology.org/N10-1073

[22] P. Kaur *et al.*, "Stemming algorithms for punjabi language," in *Proceedings of ICACCI (CEUR-WS)*, 2012. [Online]. Available: http://ceur-ws.org/

[23] W. Ahmad *et al.*, "Named entity recognition for punjabi (shahmukhi) using deep learning," in *ACL Workshop on South Asian NLP*, 2023. [Online]. Available: https://aclanthology.org/

[24] A. Khan, "Part-of-speech tagging for punjabi shahmukhi," 2021, researchGate preprint. [Online]. Available: https://www.researchgate.net/

[25] T. Developers, "tf.keras.preprocessing.sequence.pad_sequences," 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences

[26] S. Users, "Padding sequences for rnns: Pre vs post," Data Science Stack Exchange, 2020. [Online]. Available: https://datascience.stackexchange.com/questions/36666/padding-rnn

[27] K. Koskenniemi, "A general computational model for word-form recognition and production," in *Proceedings of the 10th International Conference on Computational Linguistics*, 1984, pp. 178–181. [Online]. Available: https://www.cambridge.org/

[28] K. R. Beesley and L. Karttunen, *Two-Level Morphology: A Computational Model for Word-Form Recognition and Production*. Cambridge University Press, 1992. [Online]. Available: https://www.cambridge.org/

[29] A. Jena *et al.*, "An oriya morphological analyzer using lt-toolbox and paradigm approaches," in *Proceedings of the International Conference on NLP*, 2011. [Online]. Available: https://link.springer.com/

[30] M. Bapat *et al.*, "A finite-state morphological analyzer for marathi," in *Proceedings of ICON*, 2010. [Online]. Available: https://link.springer.com/

[31] P. Goyal and G. S. Lehal, "Hindi morphological analyzer and generator using finite state transducers," *International Journal of Emerging Technologies in Computational and Applied Sciences*, 2009. [Online]. Available: https://www.mdpi.com/

[32] M. Agarwal *et al.*, "Finite-state transducer based morphological analyzer for hindi," in *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*, 2014. [Online]. Available: https://link.springer.com/

[33] P. Gupta and S. Jamwal, "Hybrid rule and ml-based morphological analyzer for dogri nouns," *Journal of South Asian Language Technology*, 2025. [Online]. Available: https://link.springer.com/

[34] D. Malladi *et al.*, "A statistical machine learning approach for hindi lemmatization," *MDPI Applied Sciences*, 2016. [Online]. Available: https://www.mdpi.com/

[35] R. Cotterell *et al.*, "The sigmorphon 2016 shared task—morphological reinflection," in *Proceedings of SIGMORPHON*, 2016. [Online]. Available: https://aclanthology.org/

[36] A. McCarthy *et al.*, "The sigmorphon 2019 shared task: Morphological analysis in context," in *Proceedings of SIGMORPHON*, 2019. [Online]. Available: https://aclanthology.org/

[37] A. Hafeez *et al.*, "A bilstm-based morphological analyzer for urdu," *MDPI Information*, 2023. [Online]. Available: https://www.mdpi.com/

[38] R. Baxi and A. Bhatt, "Gujmorph: A morphological corpus and bilstm baseline for gujarati," *ACL Anthology*, 2021. [Online]. Available: https://aclanthology.org/

[39] B. Premjith *et al.*, "Morpheme segmentation of malayalam words using rnns and grus," in *Proceedings of the International Conference on Computational Linguistics*, 2016. [Online]. Available: https://www.mdpi.com/

[40] M. Ekanayaka *et al.*, "Neural morphological analysis for sinhala using bigrus," *ICTer Journal*, 2023. [Online]. Available: https://icter.sljol.info/

[41] J. Nehrdich *et al.*, "Byt5-sanskrit: A byte-level transformer for sanskrit morphological analysis," in *Proceedings of ACL*, 2024. [Online]. Available: https://arxiv.org/

[42] M. Hasan *et al.*, "Banglat5: A transformer-based lemmatization model for bangla," in *Proceedings of COLING*, 2025. [Online]. Available: https://www.researchgate.net/

[43] J. Kanerva *et al.*, "Universal lemmatizer: A multilingual sequence-to-sequence approach," in *Proceedings of the 58th Annual Meeting of the ACL*, 2020. [Online]. Available: https://www.cambridge.org/

[44] S. Huang *et al.*, "Joint morphological tagging and lemmatization with sequence models," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/

[45] K. Supriya *et al.*, "Transformer-based morphological analysis for kannada using apertium paradigms," in *Proceedings of LREC*, 2025. [Online]. Available: https://www.mdpi.com/

[46] A. Anastasopoulos and G. Neubig, "Pushing the limits of low-resource morphological inflection," in *Proceedings of ACL*, 2019. [Online]. Available: https://aclanthology.org/

[47] C. Malaviya *et al.*, "Neural factor graph models for cross-lingual morphological tagging," in *Proceedings of EMNLP*, 2018. [Online]. Available: https://aclanthology.org/

[48] L. Jin and K. Kann, "Exploring multi-task learning for morphological analysis," in *Proceedings of EMNLP*, 2017. [Online]. Available: https://aclanthology.org/

[49] K. Saunack *et al.*, "Cross-lingual lemmatization for low-resource indian languages," in *Proceedings of COLING*, 2020. [Online]. Available: https://aclanthology.org/

[50] A. Pawar *et al.*, "Multilingual t5 for gender-number-person tagging in indian languages," in *Proceedings of ACL Workshop on South Asian NLP*, 2023. [Online]. Available: https://aclanthology.org/

[51] S. Wu and M. Dredze, "Typology and cross-lingual transfer in morphological learning," *ACL Transactions*, 2019. [Online]. Available: https://cfilt.iitb.ac.in/

[52] A. Lauscher *et al.*, "Multilingual training for morphologically rich languages," *ACL Transactions*, 2020. [Online]. Available: https://cfilt.iitb.ac.in/

[53] K. Sarveswaran *et al.*, "Thamizhimorph: An open-source fst for tamil," *Springer LNCS*, 2021. [Online]. Available: https://link.springer.com/

[54] H. Dasari *et al.*, "Bert-te: A transformer-based contextual morphological analyzer for telugu," in *Proceedings of the ACL Workshop on South Asian Languages*, 2022. [Online]. Available: https://www.mdpi.com/

[55] M. Creutz and K. Lagus, "Unsupervised models for morpheme segmentation," *Computer Speech and Language*, 2007. [Online]. Available: https://aclanthology.org/

[56] K. Lagus and M. Creutz, "Unsupervised morphological segmentation and analysis," *ACL Transactions*, 2007. [Online]. Available: https://aclanthology.org/

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[58] M. Basharat and U. Waris, "Urdu–punjabi merged dataset," Kaggle dataset, 2024, accessed: 2025-10-28. [Online]. Available: https://www.kaggle.com/datasets/malaikabasharat/urdu-punjabi-merged-dataset/data

[59] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[61] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[63] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[64] S. Hussain, "Urdu morphology, orthography and lexicon," *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, 2004.

[65] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," in *Proceedings of EMNLP*, 2015.

[66] M. Ballesteros, C. Dyer, N. A. Smith, and Y. Goldberg, "Improved transition-based parsing by modeling characters instead of words with lstms," in *Proceedings of EMNLP*, 2015.

[67] M. Faruqui, Y. Tsvetkov, G. Neubig, and C. Dyer, "Morphological inflection generation using character sequence to sequence learning," in *NAACL Workshop on Morphological Generation and Paradigm Completion*, 2016.

[68] K. Kann and H. Schütze, "Med: The lmu system for the sigmorphon 2016 shared task on morphological reinflection," in *Proceedings of SIGMORPHON*, 2016.

[69] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 3104–3112.

[70] A. Das *et al.*, "Indicnlp library: Natural language processing for indian languages," 2020, gitHub repository: https://github.com/AI4Bharat/IndicNLP.

[71] M. Akram and S. Hussain, "Urdu morphological analysis and generation using finite-state transducers," *Journal of Quantitative Linguistics*, vol. 28, no. 4, pp. 379–398, 2021.

[72] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[73] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[74] A. Anastasopoulos and G. Neubig, "Cross-lingual transfer in morphological inflection generation," in *Proceedings of ACL*, 2019. [Online]. Available: https://aclanthology.org/

[75] M. G. A. Malik, "Urdu lexicon development and standardization," in *Proceedings of the Conference on Language Technology*, 2006.

[76] A. A. Krizhanovsky and A. V. Linets, "Morphological analysis and generation for inflective languages: A neural network approach," *Procedia Computer Science*, vol. 154, pp. 77–83, 2019.

[77] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The state and fate of linguistic diversity and inclusion in the nlp world," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL, 2020, pp. 6282–6293. [Online]. Available: https://aclanthology.org/2020.acl-main.560/

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008. [Online]. Available: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[79] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*. ACL, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423/

[80] T. Rahman, *Language and Politics in Pakistan*. Oxford University Press, 2011.

[81] H. S. Gill and H. A. Gleason, *Punjabi Grammar and Morphology*. Munshi Ram Manohar Lal, 2019.