mathematics

Article

Contextual Urdu Lemmatization using Recurrent Neural Network Models

Rabab Hafeez [1], Muhammad Waqas Anwar [1], Muhammad Hasan Jamal [1], Tayyaba Fatima [1], Julio César Martínez Espinosa [2,3,4], Luis Alonso Dzul López [2,3,5], Ernesto Bautista Thompson [2,3,6] and Imran Ashraf [7,*]

1 Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan
2 Higher Polytechnic School, Universidad Europea del Atlántico, Isabel Torres 21, 39011 Santander, Spain
3 Department of Project Management, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
4 Department of Project Management, Universidade Internacional do Cuanza, Cuito EN250, Bié, Angola
5 Project Management, Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
6 Fundación Universitaria Internacional de Colombia Bogotá, Bogota 111311, Colombia
7 Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea
* Correspondence: imranashraf@ynu.ac.kr

Abstract: In the field of natural language processing, machine translation is a colossally developing research area that helps humans communicate more effectively by bridging the linguistic gap. In machine translation, normalization and morphological analyses are the first and perhaps the most important modules for information retrieval (IR). To build a morphological analyzer, or to complete the normalization process, it is important to extract the correct root out of different words. Stemming and lemmatization are techniques commonly used to find the correct root words in a language. However, a few studies on IR systems for the Urdu language have shown that lemmatization is more effective than stemming due to infixes found in Urdu words. This paper presents a lemmatization algorithm based on recurrent neural network models for the Urdu language. However, lemmatization techniques for resource-scarce languages such as Urdu are not very common. The proposed model is trained and tested on two datasets, namely, the Urdu Monolingual Corpus (UMC) and the Universal Dependencies Corpus of Urdu (UDU). The datasets are lemmatized with the help of recurrent neural network models. The Word2Vec model and edit trees are used to generate semantic and syntactic embedding. Bidirectional long short-term memory (BiLSTM), bidirectional gated recurrent unit (BiGRU), bidirectional gated recurrent neural network (BiGRNN), and attention-free encoder–decoder (AFED) models are trained under defined hyperparameters. Experimental results show that the attention-free encoder-decoder model achieves an accuracy, precision, recall, and F-score of 0.96, 0.95, 0.95, and 0.95, respectively, and outperforms existing models.

Keywords: neural networks; natural language processing; inflectional morphology; derivational morphology

MSC: 68T50

check for updates

Citation: Hafeez, B.; Anwar, M.W.; Jamal, M.H.; Fatima, T.; Espinosa, J.C.M.; López, L.A.D.; Thompson, E.B.; Ashraf, I. Contextual Urdu Lemmatization using Recurrent Neural Network Models. *Mathematics* **2023**, *11*, 435. https://doi.org/10.3390/math11020435

Academic Editors: Nebojsa Bacanin and Catalin Stoean

Received: 29 November 2022
Revised: 5 January 2023
Accepted: 8 January 2023
Published: 13 January 2023

1. Introduction

In today's modern world, almost every job involves the use of computers, resulting in the production of a massive amount of data that needs to be processed and analyzed by computers [1]. Natural language processing (NLP) is the computerized study of human languages that have naturally evolved over time. NLP and machine translation (MT) are fields that are constantly evolving and helping to bridge the linguistic gap between individuals. To comprehend the development of MT algorithms, along with a strong vocabulary, the knowledge of different modules including morphological analysis and normalization is also required. In the morphological analysis of a language, the morphological structure of different words is analyzed while in information retrieval applications, stemming and lemmatization are the preprocessing steps [2]. Stemming is the process of removing prefixes and/or suffixes from an inflectional or derivational word to obtain the stem as an output [3]. However, in several Semitic languages, such as Arabic, Hebrew, Syriac, and Urdu, infixes can also be found in many words. An infix is a morpheme that is positioned between the base/root word, following a specific morphological balance/pattern. There are no prefixes or suffixes attached to such words, although they do contain infixes. Urdu is one of the languages with many infixes. This implies that the stemming procedures in such languages should consider stemming the infixes as well [4].

In semantics, the objective of lemmatization is to group the various inflected forms of a word to break them down into a common form and analyze them as a basic term. It involves the use of vocabulary and morphological analysis of words, intending to remove inflectional endings from the words and return their base or dictionary form i.e., a lemma (plural lemmas or lemmata). For example, in the English language sit, sits, and sitting are all forms of the verb sit; therefore, sit is the lemma of all these words. Lemmatization is the most used approach in any IR application, including indexing and searching. A few studies on IR systems for the Urdu language have shown the effectiveness of lemmatization instead of stemming [5]. However, lemmatization techniques for resource-scarce languages such as Urdu have not received much attention [6].

There are very few existing studies on Urdu lemmatization and all these studies use a rule-based approach [7]. These studies lack the ability to handle noun identification and while lemmatizing proper nouns are also lemmatized, which changes the meaning and the context of the actual word. Moreover, these studies do not handle stop words that are removed from the corpora upon lemmatization. Additionally, these studies do not deal with loan words, inflectional and derivational morphemes, and diacritized Urdu words. Although many studies apply machine learning and neural network models for lemmatization in various languages, yielding higher performance, to the best of our knowledge, no existing study applies these techniques to the Urdu language. In this study, we present an improved lemmatization algorithm based on recurrent neural network (RNN) models for the Urdu language. The proposed model deals with the detection of proper nouns by generating semantic embeddings and lemmatizing accordingly, if required. Additionally, the model successfully handles the lemmatization of inflectional and derivational Urdu morphological words. The proposed model also successfully handles around 344 stop words during lemmatization as well as diacritized Urdu words and loan words. In a nutshell, we make the following main contribution:

- An Urdu lemmatization algorithm based on four RNN models, namely BiLSTM, BiGRU, BiGRNN, and AFED, is presented.
- Lemmatization is performed to find the accurate and legitimate lemma from the two Urdu corpora.
- Semantic and syntactic embeddings are created using the Word2Vec model and edit trees are generated from the word lemma pairs available in the input file.
- The proposed approach successfully handles proper nouns, inflectional and derivational morphemes, stop words, loan words, and diacritized Urdu words.
- Accuracy, precision, recall, and F-score evaluation measures are used to evaluate the performance of the proposed approach.

The rest of the paper is organized as follows. Section 2 presents the literature review. Section 3 discusses the background of Urdu language morphology. Section 4 discusses the proposed methodology followed by evaluation and discussion on results in Section 5. Section 6 concludes the paper.

2. Related Work

Various studies have been conducted on the lemmatization of resource-scarce languages. In this section, we discuss some of the existing studies based on the techniques used in these studies for lemmatization and their findings.

A rule-based approach is used for the lemmatization of the Hindi language in [8]. Dictionary and rule-based lemmatization is used for lemmatization of Mongolian language [9] which is an enhanced version of the lemmatization method developed by Khaltar et al. [10]. Suhartono et al. [11] develop an algorithm for the lemmatization of Bahasa Indonesia based on the enhanced confix stripping stemmer. Boudchiche et al. [2] present a hybrid approach to Arabic lemmatization that assigns a single lemma to each Arabic word and then takes the context of that word into account. A statistical task based on the hidden Markov models is used for lemmatization. For a labeled corpus of around 500000 words, the lemmatizer identifies the correct lemma with an accuracy of 94.45%. Plisson et al. [12] use a rule-based approach for word lemmatization in the Slovene language.

Freihat et al. [13] perform lemmatization on the Arabic language using a machine-learning approach and dictionary-based lemmatization. Both techniques are used separately and yield an accuracy of 95%. Results show that with the use of a dictionary of lemmatization, the underlying pipeline of NLP gets an additional boost. Chakrabarty et al. [14,15] implement a neural lemmatizer to extract the lemma from the Bengali corpus that outperforms a simple Bengali lemmatizer with the cosine similarity margin of 1.37%. Putz et al. [16] conduct a structural study on the morphological and neural-based approach to German language lemmatization. Kondratyuk et al. [17] evaluate the bidirectional RNN model on languages such as Arabic, German, and Czech and show a higher lemmatization accuracy than part-of-speech tagging and state-of-the-art techniques.

Jabbar et al. [5] present a survey on different techniques and algorithms developed for stemming and lemmatization of Urdu and similar languages and evaluate them for an Urdu corpus. The results show that the lemmatization approach is better as compared to the stemming techniques. Gupta et al. [7] present a rule-based lemmatizer for the Urdu language that subtracts the suffix from the main word and adds some other related words or information to obtain the desired meaning. The proposed approach decreases the time complexity, and when applied to a test dataset of thousand words, extracts the correct “lemma" with an accuracy of 90.30%. Humayoun et al. [18] use rule-based lemmatization for the Urdu language and achieve an accuracy of 65.20% for segmented corpus.

Although many studies applying machine learning and neural-based lemmatization to various languages yield higher performance, to the best of our knowledge, no existing study applies these techniques to the Urdu language. Table 1 summarizes different lemmatization approaches used for various resource-scarce languages, along with the benefits and drawbacks of each approach.

Table 1. Summary of different lemmatization approaches applied for various resource-scarce languages.

Techniques	Benefits	Drawbacks	Language
Statistical task-based using hidden Markov models	Higher accuracy as the model has the ability to learn complex patterns and dependencies existing between inflected forms of a word and its lemma.	Requires large-annotated training data, which is not readily available, to learn the probabilistic relationships between the different word forms and their lemmas.	Arabic [2]
	Efficiently handles large datasets, making them well-suited for lemmatization tasks where the goal is to process a large amount of text.	Agnostic of the context in which a word appears, hence unable to accurately lemmatize words having multiple possible lemmas depending on the context.	
	Language-agnostic which helps build a lemmatizer for any language provided there is enough training data available.	Based on probabilistic models of word form and lemma relationships, hence unable to handle unusual or unexpected word forms, leading to errors/inaccuracies in lemmatization.	
Rule-based approach	Higher accuracy as it is based on pre-defined rules, list of lemmas and their corresponding inflected forms.	Lacks generalization, as it is dependent on the list of rules that requires comprehensive knowledge of the language.	Hindi [8] Mongolian [9,10] Bahasa Indonesia [11] Slovene [12] Urdu [7,18]
	Does not require model training like ML-based techniques.	Error-prone to rare or unusual inflections because of predefined rules. Therefore, should be used in conjunction with other techniques, such as stemming.	
	Customizable to specific languages/domains, allowing for greater control over the stemming or lemmatization process.	It is time-consuming as it requires defining a predefined list of rules and a set of suffixes for each language.	
		Agnostics to the context of the word and cannot accurately lemmatize words that may have different meanings in different contexts.	
		Has limited coverage as it is unable to handle a wide range of inflections and derivations, leading to inaccurate lemmatization for words that are not covered by the rules.	
		Not adaptable to new words or changes in the language.	
		Unable to capture the full range of variations and complexities of a language, leading to lower stemming and lemmatization accuracy.	
Machine and deep learning based approach	Neural networks can learn complex patterns and relationships in data, leading to improved accuracy in lemmatization.	Neural network-based lemmatization can be complex and resource-intensive, requiring significant computational power and memory.	Bengali [14,15] German [16] Arabic, German, Czech [17]
	Well-suited to train and test large data, hence practical choice for lemmatization on even very large corpora.	Neural network-based lemmatization requires a large amount of annotated training data to learn the patterns and relationships between word forms and lemmas. If enough annotated data are not available, the accuracy of the lemmatization may be limited.	
	Can consider the context in which a word appears, leading to improved accuracy of lemmatization for words with multiple possible lemmas depending on the context.	Dependent on the choice of hyperparameters.	
	Able to handle a wide range of inflections as it learns from the data rather than relying on predefined rules.	Less practical choice for tasks where computational resources are limited.	
	Can generalize well to new data, as it is able to learn patterns in the data rather than memorizing specific examples.		
Hybrid approach	Useful in cases where rule-based or machine and deep learning approach alone is not sufficient to accurately stem a wide range of words.	More complex to implement and understand, and requires more computational resources than a single technique, as it involves combining multiple techniques.	Arabic [13] Urdu [5]
	Often achieves higher accuracy than a single technique, as it takes advantage of the strengths of multiple techniques.	Less practical choice for tasks where computational resources are limited.	
	More robust since it can use the dictionary to derive a word's lemma even if the word is not seen during training.		
	Can be customized to handle specific inflections or words by adding them to the dictionary.		

3. Morphologically of Urdu Language

The Urdu language is a combination of a few other languages including Arabic, Turkish, Persian, and Hindi. All these languages are characterized by their complex morphological structure. This results in wide linguistic variations in phonology and grammar of the language causing difficulties in the development of a lemmatization algorithm. Studies on Urdu morphology show that a single Urdu lemma can have approximately 57 distinct forms. Consider the Urdu lemma "پڑھ" -/parh/ (read), from which multiple variants can be generated, as shown in Figure 1.

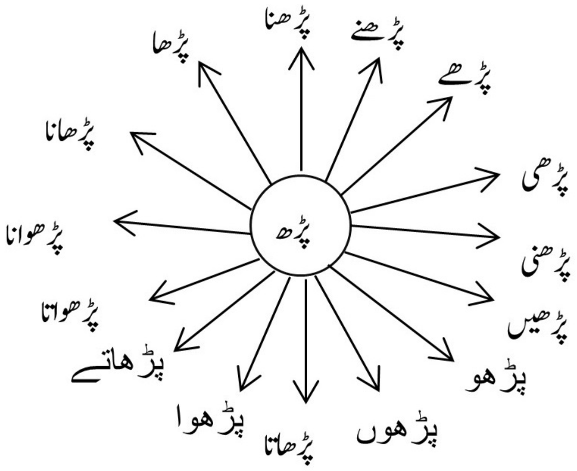


Figure 1. The many distinct forms of the Urdu lemma "پڑھ" -/parh/ (read).

The words "پڑھانا" -/parhana, "پڑھوانا" -/parhwana, "پڑھواتا" -/parhwata, "پڑھاتي" -/parhatay, "پڑھوا" -/parhwa, "پڑھاتا" -/parhata translate to the English word "teach" in various contexts while the words "پڑھوں" -/parhoon, "پڑھو" -/parho, "پڑھں" -/parhain, "پڑھنی" -/parhni, "پڑھی" -/parhi, "پڑھي" -/parhay, "پڑھني" -/parhanay, "پڑھنا" -/parhana, "پڑھا" -/parha translate to the English word "read" in various contexts.

3.1. Vowels in Urdu

The Urdu language is written from right to left and consists of around 48 consonants known as Urdu characters, 10 long vowels, 6 short vowels, and 4 diacritics. A few examples are shown in Table 2. These characters need to be handled correctly to get the correct lemma out of such words.

Table 2. Examples of long vowels, short vowels, and diacritics in Urdu Morphology.

Long Vowel		Short Vowel		Diacritic	
Symbol	Example	Symbol	Example	Symbol	Example
اُ	اُو -/O	اَ	پَاک -/Pak	ں	مں -/Maine
اي	آي -/Aye	ئ	ولئ کامل -/Wali-e-Kaamil	ء	إسماعيل -/Ismail
اِ	اِی -/E	ئے	روئي زمں -/Ru-e-Zameen	آ	بنْت -/Bint
او	آو -/Aao	اُ	حُسن -/Husn	آّ	مدرسّہ -/Madrasa

3.2. Spellings in Urdu

The spellings (ہجی) of words in the Urdu language are often modified to account for features such as gender, number, tense, and case. For example, in the following sentence, the word "لکھائی" -/likhaee/ (handwriting) is the inflected form of lemma"لکھ" -/likh/

(write). It uses the inflectional affix "+ئی" to change the grammatical form of the word from verb to noun and gender from masculine to feminine. Similarly, the word "خوبصورت" -/khoobsurat/ (beautiful) is the inflected form of the lemma "خوب" -/khoob/ (good) and affix "+صورت", respectively.

مری لکھائی بہت خوبصورت ہي
(My handwriting is very beautiful)

3.3. Inflections and Derivations in Urdu

In the Urdu language, the gender of a word, i.e., masculine/feminine, and the number, i.e., singular/plural are recognized by inflections which are represented by the case, i.e., direct, oblique, and vocative whereas derivations are the words added to some other categories such as nouns, adjectives, or verbs to make a new word, i.e., their forms and types are derived from some other words [19]. A few of such inflections and derivations are discussed in Table 3. Such properties of Urdu morphology cause many complications in the development of contextual lemmatization algorithms.

Table 3. Morphological-based discussion of a few inflectional and derivational words in the Urdu language.

Word	Type	Discussion
پانی ، حجازی	Inflectional	Both end with "ی", making them feminine but in reality, they are masculine in nature.
بچاؤ ، بہاؤ	Derivational	These masculine abstract nouns are the resultant of causative verbs when their roots are attached with "ؤ".

3.4. Loan Words in Urdu

Since the Urdu language is a mixture of many languages, the words whose origin belongs to other languages are known as loan words [20]. The inclusion of loan words in the vocabulary of the Urdu language makes its morphology more complicated. Common loan languages include Arabic, Hindi, Persian, Turkish, and English. A few examples of loan words or borrowed words are shown in Table 4.

Table 4. A few examples of load words used in the Urdu language.

Loan Language	Loan Words			
Persian	نمائندگان	ہم نفس	دانشور	دستِ بازو
Arabic	مساجد	مفتاح	علماِ دین	لا قانونیت
English	کالم (Column)	انجن (Engine)	سائکل (Cycle)	ہیٹر (Heater)
Turkish	امداد	بدترین	مرکب	مراسم

3.5. Nouns and Verbs in Urdu

In the Urdu language, a noun's lemma is its singular masculine form without diacritics. For example, lemma of the noun "مُعّلِمَاتْ" -/mualimaat/ (female teachers), is " معلم" -/mualim/ (male teacher). However, the lemma of a verb in Urdu is its root form without any gender information e.g., the lemma of the verb "لکھوایا" -/likhwaya/ (to make someone write) is "لکھ" -/likh/ (write).

Lemmatization and stemming can be tricky in the Urdu language because of ambiguities in a few words [4], e.g., the resulting lemma for the words "مہمان" -/mehmaan/

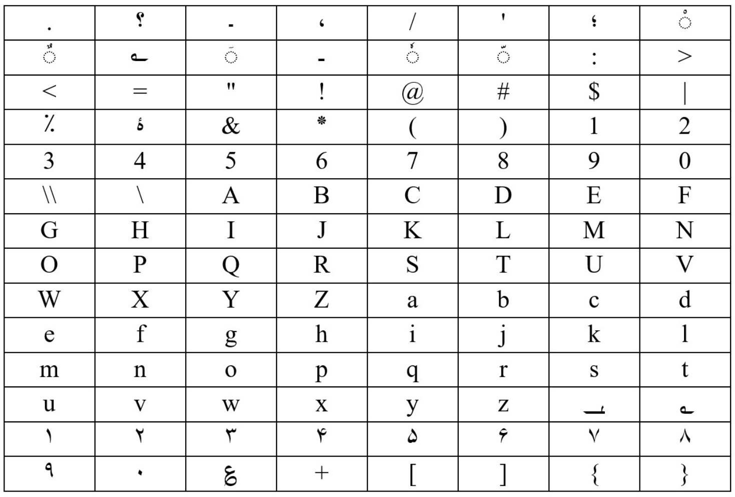
(guest),"مہم جو" -/muhim jo/ (adventurer), and "مہمات" -/muhmaat/ (adventures) is the word "مہم" -/muhm/ (adventure), which is correct for the word "مہمات" -/muhmaat/ (adventures) and "مہم جو" -/muhim jo/ (adventurer) but not for the word "مہمان" -/mehmaan/ (guest) because "مہمان" itself is a root word. Similarly, for the word "ناراضی" -/narazi/ (outrage), lemmatization provides "راض" -/raz as the lemma by chopping the morpheme "نا" -/na considering it a prefix and "ی" considering it a postfix, resulting in an incorrect lemma. However, the correct lemma for the word "ناراضی" -/narazi/(outrage) is "ناراض" -/naraz/ (angry), which gives the accurate dictionary meaning of the word and related morphological properties [5].

4. Proposed Methodology

4.1. Corpus Selection and Preprocessing

For this study, we perform lemmatization on two Urdu corpora, i.e., the Urdu Monolingual Corpus (UMC) [20] and the Universal Dependencies Corpus of Urdu (UDU) [21,22], which comprise 5,464,575 and 5,130 sentences, respectively. Sentences are annotated in the ten-column CoNLL-U format.

Preprocessing of both UMC and UDU datasets involves multiple different steps. The initial step is sentence splitting. For sentence splitting in the Urdu language, full stop "ختمہ (-)" and question mark "سوالیہ(؟)" are called sentence boundaries. Data redundancy is minimized and only unique occurrences are kept. Arabic and Quranic quotes as well as English translations of a few Arabic phrases are removed manually. The Western and Eastern Arabic numerals available in the sentences are removed as they are not required to be lemmatized. The segmented sentences of both datasets are tokenized. The control and space characters are detected and replaced by "\n" so that each token appears on a new line. The sentence boundary operators are replaced with an empty line so that the end of the previous sentence and the start of the new sentence can be figured out. Additional noise including brackets, colons or semi-colons, invalid UTF-8 characters, and punctuation marks is also discarded. Figure 2 shows the characters that were removed.



.	؟	-	،	/	'	؛	◌ۡ
◌ؑ	ے	◌ٰ	-	◌ؒ	◌ّ	:	>
<	=	"	!	@	#	$	|
٪	ۂ	&	٭	(	)	1	2
3	4	5	6	7	8	9	0
\\	\	A	B	C	D	E	F
G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V
W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l
m	n	o	p	q	r	s	t
u	v	w	x	y	z	ـــ	ـە
۱	۲	۳	۴	۵	۶	۷	۸
۹	۰	ع	+	[	]	{	}

Figure 2. Noise removed while dataset preprocessing.

4.2. Experimental Design and Setup

The proposed model requires the creation of initial data files including the parameters files. Two different embeddings are used to achieve higher accuracy. One is semantic embedding W_e^{sem} and the other is syntactic embedding W_e^{syn}. The model does not rely on human-defined features or a morphological tag set, but it requires a gold lemma based on a training dataset. An input training file has words $w_t^{iS_i}$ and their respective lemma $l_t^{iS_i}$ tab separated. After every sentence, there should be a "\n" to get the count of all sentences $\sum S^t$.

The Word2Vec model file [23,24] is created to retrieve the semantic information of a word W_e^{sem} from the context words. There are two different approaches to get the word vectors out of corpora with the vocabulary of size N. The continuous bag of words (CBOW) model is used to get the W_e^{sem} from the context words. To calculate the vector representation, it takes the context terms to the left w_{i-2}, w_{i-1} and right w_{i+1}, w_{i+2} of the word w_i taken as input. This word window can be adjusted to increase or decrease the size of the local context associated with the word embedding. A negative sampling is preferred to get around 200-dimensional word vectors. Figure 3 shows the feed-forward neural network of CBOW–Word2Vec.

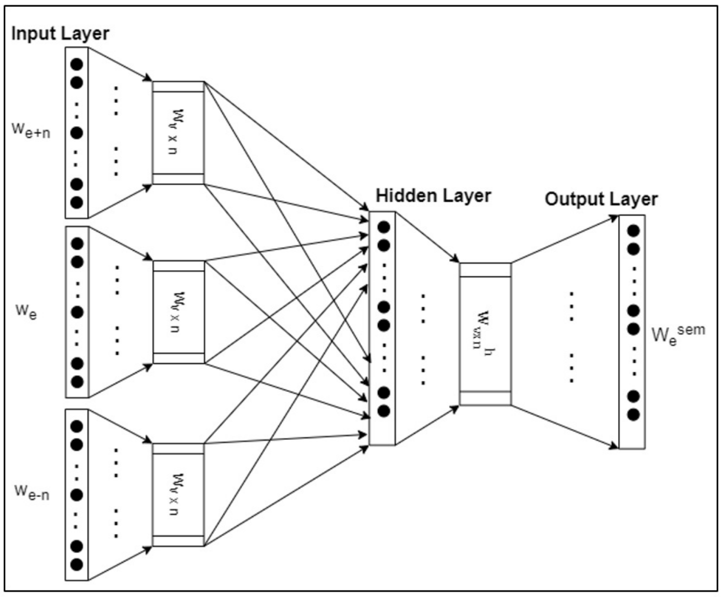


Figure 3. CBOW model for generating W_e^{sem}.

A parameter file is created to set up the model parameters. The first parameter is the cell type of the character-level network. It can be any of the models we are working on, i.e., BiLSTM, BiGRNN, or BiGRU. The second parameter should be the maximum word length in a number of characters. For words with more characters than this value, the excess characters will be truncated. In the proposed model, we consider a sample of 25 characters per word. Words shorter than 25 characters are padded with null characters at the end whereas, for the longer words, excess characters are truncated. The third parameter includes the number of neurons in the cell of the character-level network. The embedding using the Word2Vec file name should be given as the fourth parameter. The fifth parameter should be the cell type of the second-level recurrent network which does the edit tree classification. The number of neurons in the cell of the edit tree classifier is given as the sixth parameter. The number of epochs for the training of the network is the seventh parameter. Epochs E can vary according to need and can be set to any integer value. For this study, the epoch value is set to 50. The eighth parameter gives the batch size for the training of the network. A training dataset can be divided into one or more batches. The ninth parameter is the divide file factor. For large corpora, generating the data matrices (class annotation matrices, applicable edit trees matrices, char vector matrices, etc.), which will be generated while running the code, takes too much time. To reduce the time, we fragment the matrices part by part and then dump them in the disk and reload them part by part. To evaluate the lemmatization results on a test file, the gold lemma of each word is created. The format of this file is like the training file. The words in corpora are refined and according to فروز الغات، اردو لائبریری، اردو لغات Urdu dictionaries, the accurate lemma of the words is copied for further evaluation purposes.

Initially, the edit trees are constructed that map a word to its given lemma. All the extracted unique edit trees are associated with the surface word lemma pairs provided in the training file. Here the requirement is that for each test word, the correct edit tree should be identified by the model so that the exact lemma is generated. The words available in the training corpora are then annotated by their corresponding edit trees. A sample example of the edit tree is shown in Figure 4. It shows the lemma pair of an inflected noun "ترجمانی" -/tarjamani/(representation). In the right edit tree, the root node stores the length of prefix "تر" (2) and suffix "انی" (3), whereas the left edit tree visualizes what each node corresponds to. The model uses the right edit tree.

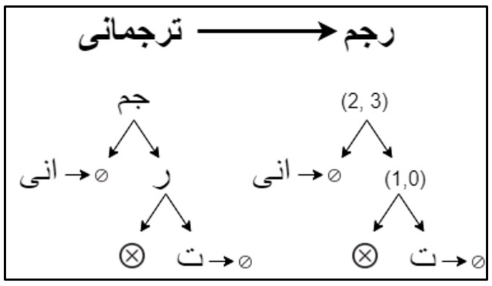


Figure 4. Edit tree example of a word lemma pair of an inflected noun "ترجمانی" -/tarjamani/ (representation).

For W_e^{syn} of a contextual word, each word becomes an array of the length of 25 characters and is represented as a one-hot encoded vector. This vector then becomes the input of the embedding layer. To get the syntactic structure of an input word w_in for the dimensional vector v_d, a sequence of characters $c_1, c_2, ..., c_s$ is defined from the alphabet character class C_a, where s represents the word length and a single character c_i is defined as one hot encoded vector shown as $1v_i$. Next, the embedding layer is specified as:

$$L_c = \mathbb{R}^{v_E \times |C_a|} \tag{1}$$

where each $1v_i$ is projected to a dimensional vector V_E. Using Equation (1), the projected vector P_{ci} corresponding to each character c_i is obtained using Equation (2), which is a matrix multiplication operation specified by the dot operator.

$$P_{ci} = L_c.1v_i \tag{2}$$

These generated sequences of projected vectors are given as input to the LSTM cell that computes the state sequences $t_1, t_2, ..., t_m$ using the given Equations (3) to (7) of input gate i_t, forget gate f_t, and output gate o_t. In the following given equations, weight matrices are denoted by X, Y, and Z, where b is the bias parameter and $\odot$ is the element-wise Hadamard product and σ denotes the sigmoid function.

$$f_t = \sigma(X_f x_t + Y_f t_{m-1} + Z_f c_{t-1} + b_f) \tag{3}$$
$$i_t = \sigma(X_i x_t + Y_i t_{m-1} + Z_i c_{t-1} + b_i) \tag{4}$$
$$o_t = \sigma(X_o x_t + Y_o t_{m-1} + Z_o c_{t-1} + b_o) \tag{5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(X_c x_t + Y_c t_{m-1} + b_c) \tag{6}$$
$$t_m = o_t \odot tanh(c_t) \tag{7}$$

Then the GRU gives the rules for updating which are followed by Equation (8). In the GRU model, two gates are mainly used, update gate u_t and reset gate g_t.

$$t_m = (1 - u_t) \odot t(m-1) + u_t \odot tanh(X_t x_t + Y_h(g_t \odot t_{m-1}) + b_t) \tag{8}$$

The sequence of projected character P_{ci} is given by $p_{c1}, p_{c2}, ..., p_{cm}$. The forward and backward networks of given BiLSTM and BiGRU computed the state sequences $t_1^f, t_2^f, ..., t_m^f$ and $t_m^b, ..., t_1^b$ for the given projected character sequences, respectively. So, finally, the syntactic embeddings for a word w are given by merging both final states of the two sequences. It is denoted by W_e^{syn} and given by the following relation.

$$W_e^{syn} = [t_1^b, t_m^f] \tag{9}$$

The composite representation of both W_e^{sem} and W_e^{syn} embeddings is denoted by W_e^{com} and given by

$$W_e^{com} = [W_e^{sem}, W_e^{syn}] \tag{10}$$

The next step is to feed the sentence-wise data to the neural network model set up in the parameter file. It can either be simple BiGRNN, BiGRU, or BiLSTM. The model is trained to get the edit tree classification tasks. In context-sensitive lemmatization, the local context of words is extracted from these second-level bidirectional models which are in forwarding and backward directions. The inputs of the BiGRNN, BiLSTM, and BiGRU models are given by the $w1_e^{com}, ..., wn_e^{com}$ sequence. The output y_t in BiGRNN is determined by the given equation

$$P(y_t|\{x_d\}_{d \neq t}) = \varphi(W_y^f h_{t-1}^f + W_y^b h_{b+1}^b + b_y) \tag{11}$$

Similarly, when the BiLSTM model is under discussion the basic idea is to process the sequence $x = (x_1, x_2, ..., x_t)$ forward and backward. Therefore, a bidirectional LSTM has a forward and backward part. Their hidden states $\overrightarrow{h}$ and $\overleftarrow{h}$ are connected to the output layer by the FNN. Using the last hidden states, the outcome y_t at the output gate is computed similarly to the following equation.

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h_t} + W_{\overleftarrow{h}y} \overleftarrow{ht} + b_y \tag{12}$$

While BiGRU processes data in both directions with forward and backward hidden layers. Let $\overrightarrow{y}_1^T$ be the forward output of the BiGRU by processing the input sequence x_1^T through $t = 1, 2, ..., T$ and let $\overleftarrow{y}_1^T$ be the corresponding backward output by processing the input sequence in the reverse direction through $t = T, ..., 1$. The output y_1^T of the BiGRU is the stepwise concatenation of the forward and backward output.

$$y_t = (\overrightarrow{y_t}, \overleftarrow{y_t}) \tag{13}$$

For the ith input vector, which is wi_e^{com}, t_i^f gives the forward states where the backward states are given by t_i^b. While incorporating the information from the edit trees into the neural network model, we need to extract the unique edit trees as all the edit trees do not apply to the word. To avoid incompatible substitutions, we need to sort out the subclasses for the training set from the class labels. So here the model will learn how to get the applicable mass of the unique edit tree. Let us say, set T contains the distinct edit trees, and set A contains the applicable edit trees. Given that the element a_j is an application for w_i, A_i is the set that holds the final classification of forward and backward states using "SoftPlus" as an activation function.

$$I_i = softplus(G_f t_i^f + G_b t_i^b + G_a A_i + b_l) \tag{14}$$

$$f(x) = ln(1 + e^x) \tag{15}$$

To pick the accurate edit tree from the softmax layer, we select the maximum class probability and the class with the highest probability is selected as the right candidate by using the following function in which o_j represents the output of the softmax layer.

$$j = argmax_{j' \in \{1,\ldots,k\} \wedge a_i^{j'} = 1} o_i^{j'} \tag{16}$$

Figure 5 shows the model for the bidirectional gated neural network. There are multiple hyperparameters over which the working of the bidirectional neural network models is dependent. The number of hidden neurons for both layers, the number of epochs and batch size, the optimization algorithm, and loss function are described to get the proper results of the model. Mostly online results of parameters are used to update the weights of bidirectional neural networks. Other than that, the number of epochs and the number of neurons varies. The drop rate is set up to 0.2, epochs are 50, and neurons are 64. In this model, cross_entropy is used as the loss function.

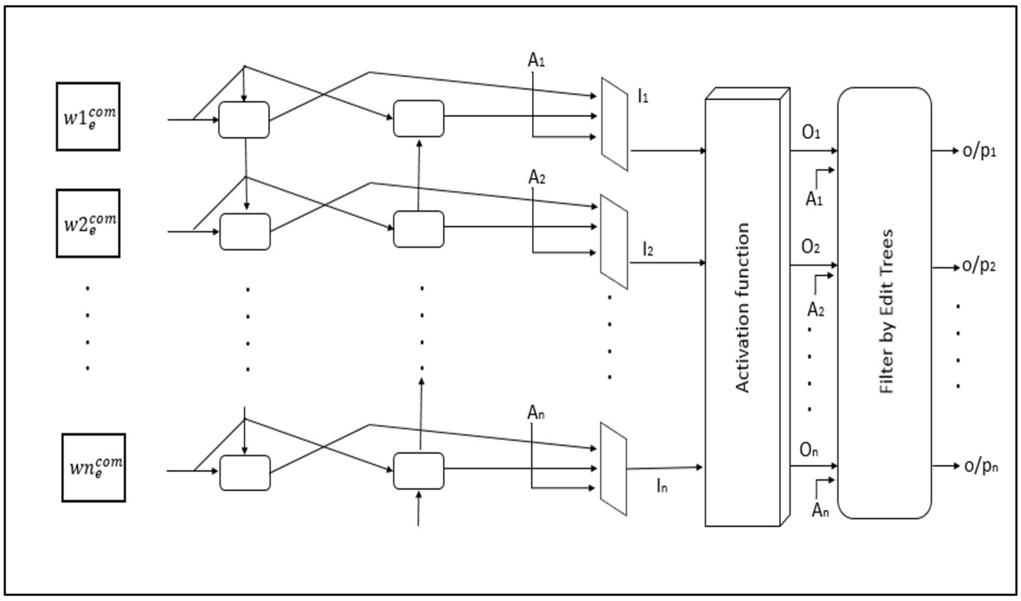


Figure 5. Second-level bidirectional gated neural network model (BiGRNN, BiLSTM, and BiGRU).

We performed another experiment using the same datasets and the same word embedding files but changing the bidirectional gated neural network model with the encoder–decoder system as shown in Figure 6. The attention-based encoder-decoder model seems to be complex and resource-consuming for lemmatization, so we propose an AFED model that is composed of a three-layered bidirectional LSTM encoder and a three-layered unidirectional LSTM decoder. The AFED model is computationally less expensive and provides a 3% to 5% increase in accuracy (at word level) as opposed to its attention-based counterpart, as discussed in [25].

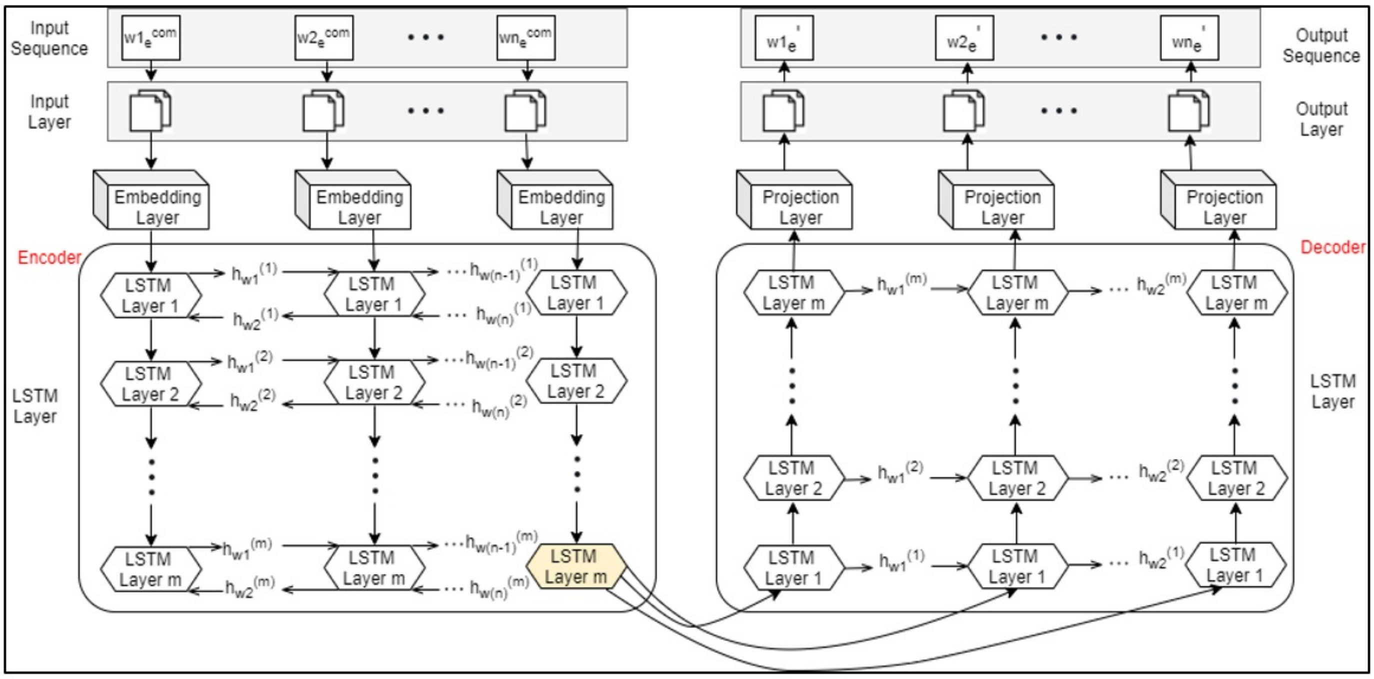
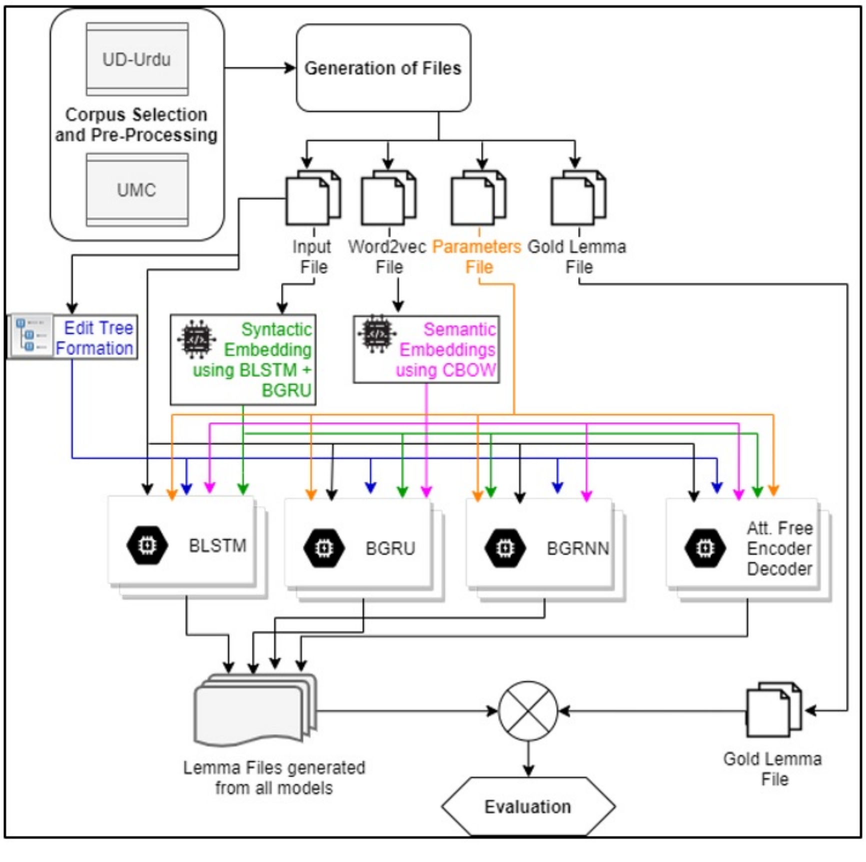


Figure 6. AFED model with a bidirectional LSTM encoder and unidirectional LSTM decoder.

We establish the "gold standard" decoder output based on the alignments which intend to copy as many input characters as possible to the output while increasing the input cursor and producing new symbols only as the last option. AFED provided state-of-the-art performance. The whole experimentation model is shown in Figure 7.



UD-Urdu
Corpus Selection and Pre-Processing
UMC
Generation of Files
Input File
Word2vec File
Parameters File
Gold Lemma File
Edit Tree Formation
Syntactic Embedding using BLSTM + BGRU
Semantic Embeddings using CBOW
BLSTM
BGRU
BGRNN
Att. Free Encoder Decoder
Lemma Files generated from all models
Gold Lemma File
Evaluation

Figure 7. Layman architecture describing contextual Urdu lemmatization using RNN models.

5. Results and Discussions

For Urdu language lemmatization, few hyperparameters were set after multiple trainings of the dataset and standard hyperparameters were found. The maximum word length was set to 25 characters. The number of neurons was 64 and the number of epochs was 50. The batch size was set to 1, where the file divide factor was 2000. Three dense hidden layers were used and the dropout rate was set to 0.2. Softmax was used as an activation function. Adam optimizer was used with a learning rate of 0.001. Categorical cross entropy was used as the loss function. Over these hyperparameters, the model is trained and eventually tested for better results. The ratio of training, testing, and validation dataset is 80%, 20%, and 10%, respectively. Figure 8 shows the loss of training and validation sets for the four models used in this study.

Table 5 shows the overall results of both datasets. The accuracy, precision, recall, and F-score of both datasets were calculated by using the following formulas.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{17}$$

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{20}$$

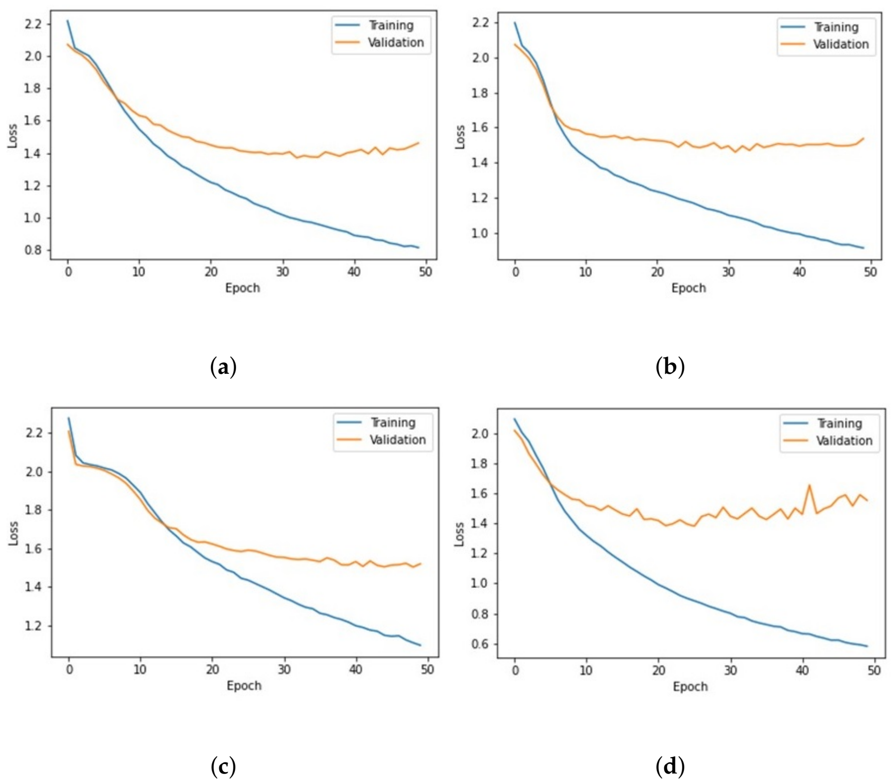


Figure 8. Training and validation loss. (**a**) BiLSTM loss; (**b**) BiGRU loss; (**c**) BiGRNN loss; (**d**) AFED loss.

Table 5. Overall results of both datasets for all models over the given hyperparameters.

Models	UMC Dataset				UDC Dataset			
	Accuracy	**Precision**	**Recall**	**F-Score**	**Accuracy**	**Precision**	**Recall**	**F-Score**
BiLSTM	91.6%	88.4%	86.4%	89.4%	95.4%	91.7%	92.3%	91.3%
BiGRU	92.5%	90.4%	89.9%	91.5%	95.7%	94.1%	93.5%	94.6%
BiGRRN	91.8%	87.8%	88.3%	88.7%	94.7%	91.8%	92.6%	92.1%
AFED	94.3%	91.7%	91.4%	92.6%	96.3%	95.2%	94.9%	95.1%

The evaluated results show that the generation of multiple word embeddings and the formation of edit trees incorporated a lot of word-lemma information, which results in better outcomes. The use of the AFED model is computationally less expensive and provides higher accuracy. When comparing two recently acknowledged models, Lemming [26] and Morfette [27], which use seq2seq and neural networking models, there is an option in Lemming and Morfette to provide an exhaustive set of root words that are used to exploit the dictionary features, i.e., to verify if a candidate lemma is a valid form or not. In contrast, the generation of edit trees and finding the applicable edit tree makes things easier without exploiting any external dictionary. Another neural network model Morpheus [28] performs lemmatization on the UDU dataset with an accuracy of 95.2% whereas the accuracies of the proposed model using BiLSTM, BiGRU, BiGRNN, and AFED are 95.4%, 95.7%, 94.7%, and 96.3%, respectively. Context-sensitive lemmatization by Chakrabarty et al. [29] perform BiLSTM and BiGRU on Bengali, Hindi, Latin, and Spanish languages, and the highest reported accuracy was 86.74% given by BiGRU on the Spanish corpus, whereas the developed model's average accuracy is above 91% for all models.

The literature shows that the stop words were removed from the corpora while lemmatization of the Urdu language in almost all the models in existing studies. The proposed model overall dealt with around 344 stop words during the lemmatization of the UDU corpus and UMC corpus and successfully lemmatized them (see Appendix A). The results show that diacritized Urdu words were successfully lemmatized according to the context of the sentence. Table 6 shows the diacritized Urdu text and its lemmatized result.

Table 6. Example of lemmatized diacritized Urdu corpus.

Words	**Lemma**
شُمَالی (Northern)	شمال (North)
چوٹِیوں (Hills)	چوٹی (Hill)
آبشَاروں (Waterfalls)	آبشار (Waterfall)

The literature shows that during lemmatization, proper nouns were also lemmatized which changes the meaning and context of the actual word. The proposed model deals with the detection of proper nouns with the generation of semantic embeddings and lemmatizes accordingly if required. Consider, for example, the sentence:

پری گل، علی اور پارسا نے امتحانات میں اعلیٰ نمبر حاصل کیے
(Pari Gul, Ali, and Parsa achieved good marks in the exams.)

The proposed model identifies پری گل -/Pari Gul and پارسا -/ Parsa as proper noun, as they are names of persons, and do not lemmatize them to پارس -/Paras or گل -/Gul. Similarly, علی -/Ali and اعلیٰ -/aala/(supreme) resembles syntactically but due to semantic embeddings, the proposed model will recognize علی -/Ali as a noun. Furthermore, loan words are successfully lemmatized using bidirectional neural network models. Table 7 shows successfully lemmatized loan words available in the Urdu Language.

Table 7. Lemmas of loan words in the Urdu language.

Words	Loan Language	In Urdu	Lemma
All ambulances	English	تمام امبولنسس	امبولنس
Am'dad	Turkish	امداد	مدد
Bad'tareen	Turkish	بد ترین	بد تر
Ayaat	Arabic	آيات	آيت

The gated neural network models successfully deal with the lemmatization of the complex inflectional and derivational affixes and other words for the Urdu language. Lemmatization of inflectional words means that the lemma and the original word both belong to a similar category, whereas derivational lemmatization deals with the lemmatization of words whose lemma does not belong to their category. A few examples of the lemmatized inflectional and derivational words are shown in Table 8.

Table 8. A few examples of lemmatized inflectional and derivational words.

Inflectional Urdu Words		Derivational Urdu Words	
Words	**Lemma**	**Word**	**Lemma**
برسات (N)	برس (N)	آبادی (N)	آباد (Adj)
حمایتی (N)	حمایت (N)	حیادار (Adj)	حیا (N)
حلف نامہ (N)	حلف (N)	خاردار (N)	خار (Adj)

Details of all cosine similarities based on baseline models are shown in Table 9. The best accuracy percentage in Lemming and Morfette for 20 languages was 94.3% and the proposed model beats the simple cosine similarity by a 2% margin. Morpheus provides the best accuracy of 94.15% on the UDU dataset, while the proposed neural networking models beat their simple cosine similarity by 1.8%. Similarly, NLP cube [25] deals with almost 73 datasets and for Urdu lemmatization, the best accuracy score is 86.85%, which is outperformed by the proposed model by a simple cosine similarity of around 9.5%. Context-Sensitive NL [29] works on four languages, Hindi, Bengali, Spanish, and Latin, and its baseline model gives a maximum lemmatization accuracy of 87.19% in the Hindi Language, which is outperformed by the proposed model by a simple cosine similarity of around 9.1%.

Table 9. Comparison of cosine similarities of the proposed model with baseline models.

Baseline Model Accuracy	Percentage Improvement
Lemming + Morfette—94.3% [26]	2.0%
Morpheus—94.15% [28]	1.8%
NLP Cube—86.85% [25]	9.5%
Context-Sensitive NL—87.19% [29]	9.1%

6. Conclusions

In this work, a supervised neural-network-model approach is proposed for the development of an Urdu lemmatizer. Two Urdu datasets are preprocessed by going through several preprocessing steps. Lemmatization of both datasets is done using the RNN models. The merger of sequence-to-sequence models is used for the creation of the lemma of the Urdu dialect. Semantic and syntactic embeddings are created using the CBOW–Word2Vec model and edit trees. BiLSTM, BiGRU, BiGRNN, and AFED models are trained under specified hyperparameters and the results of all these models are evaluated using accuracy, precision, recall, and F-score. The results show that the proposed model has the capabilities to deal with many lacking areas of Urdu lemmatization, discussed in the literature, such as handling the loan words, stop words, noun identification, and Urdu words with diacritics. Lemmatization of inflectional and derivational Urdu morphological words is also successfully handled by the proposed model. The incorporation of the AFED model improved the system's performance remarkably by achieving an accuracy, precision, recall, and F-score of 0.96, 0.95, 0.95, and 0.95, respectively. The proposed model outperforms existing lemmatization models and improved the cosine similarity significantly. In future, we plan to apply modern deep learning architectures such as the transformer to further improve the performance of our lemmatization approach for the Urdu language.

Author Contributions: Conceptualization, R.H. and M.W.A.; Data curation, M.W.A., M.H.J., and T.F.; Formal analysis, R.H. and M.H.J.; Funding acquisition, J.C.M.E.; Investigation, M.W.A. and L.A.D.L.; Methodology, M.H.J.; Project administration, J.C.M.E. and E.B.T.; Resources, J.C.M.E. and L.A.D.L.; Software, T.F. and E.B.T.; Supervision, I.A.; Validation, E.B.T. and I.A.; Visualization, L.A.D.L.; Writing—original draft, R.H. and T.F.; Writing—review and editing, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the European University of Atlantic.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset and code is available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Included Stop Words

ابھی-/abhi/(now), اپ-/ap, اپنا-/apna/(own), اپنی-/apni/(own), اپني-/apnay/(own), اجنبی-/ajnabi/(stranger), اکثر-/aksar/(often), اگر-/agar/(if), اگرچہ-/agarcha/(although), اگلي-/aglay/(next), از-/iz, اس-/iss/(this), اسی-/issi, اسي-/issay/(it), استعمال-/istimal/(use), البتہ-/albata/(however), الف-/alif, ان-/in, اندر-/andar/(inside), انہوں-/unhon/(they), انہی -/unhi/(their), انہیں-/unhain/(them), اور-/aur/(and), اوپر-/upper/(up), ایسا-/aisa/(such), ایسی-/aisi/(such), ایسي-/aisay/(such), ایک-/aik/(one), اي-/ay/(oh), آ-/aa/(come), آئی -/ai/(came), آئیں-/aain/(come), آئي-/aye/(came), آتا-/aata/(coming), آتی-/aati/(coming), آتي -/aatay/(coming), آخری -/akhri/(last), آس -/aas/(hope), آنا -/aana/(coming), آنی -/aani/(to come), آني-/aanay/(to come), آپ-/aap/(you), آیا-/aya/(came), بار -/bar/(time), باري -/baaray/(about), باوجود -/bawajood/(even so), باہر -/bahar/(outside), بظاہر -/bezahir/(apparently), بعد -/baad/(after), بعض -/ba'az/(some), بغیر -/beghair/(without), بلکہ -/balkay/(rather), بن -/ban/(become), بنا -/bana/(made), بناؤ -/banao/(make it), بند -/band/(closed), بننا -/ban'na/(to be), بھر -/bhar/(full), بھریں -/bharain/(fill up), بھی -/bhi/(too), بہت -/bohat/(very), بیس -/bees/(twenty), بي -/bay, بڑی -/bari/(big), پاس -/paas/(near), پایا -/paya/(found), پر -/per/(on), پوری -/puri/(whole), پھر -/phir/(then), پہلا -/pehla/(first), پہلي -/pehlay/(first), پیچھي -/peechay/(behind), تا -/taa, تاکہ -/ta'akay/(so that), تب -/tab/(then), تجھ -/tujh/(you), تجھي -/tujhay/(you), تحت -/tahat/(under), تر -/tar, تم -/tum/(you), تمام -/tamam/(all), تمہارا -/tumhara/(yours), تمہاری -/tumhari/(yours), تمہاري -/tumharay/(yours), تمہیں -/tumhain/(you), تو -/to/(so), تک -/tak/(up to), تھا -/tha/(was), تھی -/thi/(was), تھیں -/theen/(were), تھي -/thay/(were), تیری -/teri/(yours), جا -/jaa/(go), جاؤ -/jao/(go), جائیں -/jaeen/(go), جائي -/jaye/(go), جاتا -/jata/(goes), جاتی -/jati/(goes), جاتي -/jaatay/(goes), جانی -/jaani/(will go), جاني -/janay/(to go), جب -/jab/(when), جبکہ -/jabkay/(while), جس -/jis/(which), جن -/jin/(those), جنہوں -/jinho/(those who), جنہیں -/jinhain/(whom), جو -/jo/(which), جگہ -/jagha/(place), جہاں -/jahan/(where), جیسا -/jaisa/(like), جیسوں -/jaison/(like), جیسی -/jaisi/(like), جیسي -/jaisay/(like), چکی -/chaki/(mill), چونکہ -/chaunkay/(because), چاہئي -/chahiye/(should), چاہتي -/chahtay/(want), چاہئي -/chahiye/(should), چاہي -/chahay/(whether), حاصل -/haasil/(get), حالانکہ -/halankay/(although), حالاں -/halan, حصہ -/hisa/(share), خالی -/khaali/(empty), ختم -/khatam/(finish), خلاف -/khalaf/(against), خود -/khud/(self), درمیان -/darmiyan/(between), دسترس -/dastaras/(reach), دلچسپي -/dilchaspi/(interest), دکھائیں -/dikhain/(show), دو -/dau/(give), دوبارہ -/dobara/(again), دوران -/dauraan/(during), دوسرا -/dusra/(second), دوسروں -/dusroon/(others), دوسری -/dusri/(second), دوسري -/dusray/(second), دونوں -/dono/(both), دوں -/dun/(give), دی -/di/(gave), دئي -/diyay/(gave), دیا -/dia/(gave), دیتا -/dayta/(gives), دیتی -/dayti/(gives), دیتي -/daytay/(give), دیر -/dair/(late), دنیا -/dunya/(world), دینی -/dayni/(to give), ديني -/daynay/(to give), دیکھو -/dekho/(look), دیں -/dian/(give), دیي -/diay/(gave), دي -/day/(give), ذریعي -/ziryay/(through), ڈالا -/daala/(put), ڈالی -/daali/(put), ڈالي -/daalay/(put), ڈالنا -/daalna/(to put), ڈالنی -/daalni/

(to put), ڈالني -/daalnay/(to put), ڈالیں -/daalain/(put), رکھا -/rakha/(kept), رکھتا -/rakhta/(keeps), رکھتی -/rakhti/(keeps), رکھتي -/rakhtay/(keep), رکھنا -/rakhna/(to put), رکھنی -/rakhni/(to keep), رکھني -/rakhnay/(keep), رکھو -/rakho/(keep), رکھی -/rakhi/(kept), رکھي -/rakhay/(keep), رہ -/reh/(stay), رہا -/raha/(stayed), رہتا -/rehta/(lived), رہتی -/rehti/(lived), رہتي -/rehtay/(live), رہنا -/rehna/(live in), رہنی -/rehni, رہني -/rehnay/(stay), رہو -/raho/(stay), رہی -/rahi/(stayed), رہیں -/rahain/(stay), رہي -/rahay/(stay), زیادہ -/ziada/(more), سا -/saa, ساتھ -/saath/(with), سامني -/saamnay/(in front), سب -/sab/(all), سو -/sau/(sleep), سکا -/saka/(could), سکتا -/sakta/(can), سکتي -/saktay/(can), سی -/si, سي -/say/(from), شان -/shan/(glory), شاید -/shaid/(probably), صرف -/sirf/(only), صورت -/surat/(case), ضرورت -/zarorat/(need), ضروری -/zarori/(necessary), طرح -/tarha/(like), طرف -/taraf/(side), طور -/taur/(as), علاوہ -/ilawa/(except), عین -/ain/(exactly), غیر -/ghair/(non), کافی -/kaafi/(enough), کب -/kab/(when), کبھی -/kabhi/(ever), کچھ -/kuch/(some), کر -/kar/(do), کرتا -/karta/(does), کرتی -/karti/(does), کرتي -/kartay/(do), کرنا -/karna/(to do), کرني -/karnay/(to do), کرو -/karo/(do), کریں -/karain/(do), کري -/karay/(do it), کس -/kiss/(who), کسی -/kisi/(someone), کسي -/kisay/(to whom), کوئی -/koi/(someone), کون -/kon/(who), کونسا -/konsa/(which one), کیا -/kiya/(what), کیسي -/kaisay/(how), کیونکہ -/kyonkay/(because), کیوں -/kyun/(why), کیي -/kiye/(done), کئي -/kiye/(did), کہا -/kaha/(said), کہاں -/kahan/(where), کہہ -/keh/(say), کہیں -/kaheen/(somewhere), کہي -/kahay/(say), کي -/kay, کی -/ki/(they), کا -/ka, کو -/ko, کم -/kum/(less), کہ -/kay/(that), قریب -/kareeb/(near), گا -/ga/(will), گی -/gi/(will), گي -/gay/(will), گویا -/goya/(therefore), گئی -/gai/(she went), گیا -/geya/(he went), گئي -/gaye/(they went), گزشتہ -/guzishta/(previous), لا -/la, لائی -/lai/(brought), لائي -/laye/(brought), لاتا -/lata/(bring), لاتی -/lati/(bring), لاتي -/latay/(bring), لانا -/lana/(to bring), لانی -/lani, لاني -/lanay/(to bring), لایا -/laya/(brought), لو -/Lo/(take), لگ -/lug, لگا -/laga/(felt), لگتا -/lagta/(it seems), لگی -/lagi/(got), لگیں -/lagain/(start), لگي -/lagay/(felt), لهذا -/lehaza/(therefore), لئي -/liye/(for), لی -/li/(took), لیا -/liya/(took), لیتا -/leta/(take), لیتی -/leti/(takes), لیتي -/laytay/(take), لیکن -/laikin/(but), لیں -/lain/(take), لیي -/liye/(took), لي -/lay/(take), مجھ -/mujh/(me), مجھي -/mujhay/(me), مزید -/mazeed/(more), مقابلي -/muqabalay/(competitions), مل -/mil/(get, meet), مکمل -/mukamal/(complete), مگر -/magar/(but), میرا -/mera/(mine), میری -/meri/(mine), میري -/meray/(my), مطابق -/mutabik/
(according to), میں -/main/(I), نا -/na/(no), نہ -/na/(no), نہیں -/nahi/(no), نیچي -/neechay/(below), ني -/nay/(by), ہاں -/haan/(yes), ہر -/her/(all), ہم -/hum/(we), ہمارا -/humara/(ours), ہماری -/humari/(ours), ہماري -/humaray/(ours), ہمیشہ -/humesha/(always), ہوتا -/hota/(would have), ہوتی -/hoti/(would have), ہوتي -/hotay/(would have), ہوتیں -/hoteen/(would have been), ہونا -/hona/(to be), ہوا -/hua/(happened), ہوئي -/huay/(happened), ہوئیں -/hueen/(happened), ہوئی -/hui/(happened), ہو -/ho/(be), ہیں -/hain/(are), ہی -/he, ہوں -/hoon/(am), ہوني -/honay/(to be), ہونی -/honi/(to be), ہونگي -/hongay/(will be), یہیں -/yahin/(here), یہی -/yahi/(this), یہاں -/yahan/(here), یہ -/yay/(this), یقینا -/yaqeenan/(indeed), یعنی -/yaani/(meaning), یات -/yaat, یا -/ya/(or), ہي -/hai/(is), ہیں -/hain/(are), ہی

-/he, ہوں-/hoon/(am), ہوني-/honay/(to be), ہونی-/honi/(to be), ہونگي-/hongay/(will be), واقعی-/waqai/(really), والا-/wala/(about to), والوں-/walon/(those), والی-/wali/(about to), والي-/walay/(the ones), وجہ-/wajah/(reason), وغیرہ-/waghaira/(etc.), وہ-/woh/(they), وہاں-/wahan/(there), وہی-/wohi/(the same), وہیں-/wahin/(right there), وی-/vi, وسي-/waisay/(by the way)

References

1. Sychev, O.A.; Penskoy, N.A. Method of lemmatizer selections in multiplexing lemmatization. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *483*, 012091. [CrossRef]
2. Boudchiche, M.; Mazroui, A. A hybrid approach for Arabic lemmatization. *Int. J. Speech Technol.* **2019**, *22*, 563–573. [CrossRef]
3. Samir, A.; Lahbib, Z. Stemming and lemmatization for information retrieval systems in amazigh language. In Proceedings of the International Conference on Big Data, Cloud and Applications, Kenitra, Morocco, 4–5 April 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 222–233.
4. Fatima, T.; Islam, R.U.; Anwar, M.W.; Jamal, M.H.; Chaudhry, M.T.; Gillani, Z. STEMUR: An Automated Word Conflation Algorithm for the Urdu Language. *Trans. Asian Low-Resour. Lang. Inf. Process.* **2021**, *21*, 1–20. [CrossRef]
5. Jabbar, A.; Iqbal, S.; Khan, M.U.G.; Hussain, S. A survey on Urdu and Urdu like language stemmers and stemming techniques. *Artif. Intell. Rev.* **2018**, *49*, 339–373. [CrossRef]
6. Manjavacas, E.; Kádár, Á.; Kestemont, M. Improving lemmatization of non-standard languages with joint learning. *arXiv* **2019**, arXiv:1903.06939.
7. Gupta, V.; Joshi, N.; Mathur, I. Design and development of a rule-based Urdu lemmatizer. In Proceedings of the Proceedings of International Conference on ICT for Sustainable Development, Ahmedabad, India, 3–4 July 2015; Springer: Berlin/Heidelberg, Germany, 2016; pp. 161–169.
8. Paul, S.; Tandon, M.; Joshi, N.; Mathur, I. Design of a rule based Hindi lemmatizer. In Proceedings of the Third International Workshop on Artificial Intelligence, Soft Computing and Applications, Chennai, India, 13–15 July 2013, Springer Nature: Singapore, 2013; Volume 2, pp. 67–74.
9. Khaltar, B.O.; Fujii, A. A lemmatization method for modern mongolian and its application to information retrieval. In Proceedings of the the Third International Joint Conference on Natural Language Processing: Volume-I, Hyderabad, India, 7–12 January 2008.
10. Khaltar, B.O.; Fujii, A.; Ishikawa, T. Extracting loanwords from Mongolian corpora and producing a Japanese-Mongolian bilingual dictionary. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, 17–21 July 2006; pp. 657–664.
11. Suhartono, D. Lemmatization technique in bahasa: Indonesian. *J. Softw.* **2014**, *9*, 1203. [CrossRef]
12. Plisson, J.; Lavrac, N.; Mladenic, D. A rule based approach to word lemmatization. In Proceedings of the 7th International Multiconference Information Society IS 2004 Ljubljana, Slovenia, 13–14 October, 2004.
13. Freihat, A.A.; Abbas, M.; Bella, G.; Giunchiglia, F. Towards an optimal solution to lemmatization in Arabic. *Procedia Comput. Sci.* **2018**, *142*, 132–140. [CrossRef]
14. Chakrabarty, A.; Garain, U. Benlem (a bengali lemmatizer) and its role in wsd. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP)* **2016**, *15*, 1–18. [CrossRef]
15. Chakrabarty, A.; Chaturvedi, A.; Garain, U. A neural lemmatizer for bengali. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), Portorož, Slovenia, 23–28 May 2016; pp. 2558–2561.
16. Pütz, T.; De Kok, D.; Pütz, S.; Hinrichs, E. Seq2seq or perceptrons for robust lemmatization. an empirical examination. In Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018), Oslo, Norway, 13–14 December 2018; Linköping University Electronic Press: Linköping, Sweden, 2018; pp. 193–207.
17. Kondratyuk, D.; Gavenčiak, T.; Straka, M.; Hajič, J. LemmaTag: Jointly tagging and lemmatizing for morphologically-rich languages with BRNNs. *arXiv* **2018**, arXiv:1808.03703.
18. Humayoun, M.; Yu, H. Analyzing pre-processing settings for Urdu single-document extractive summarization. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), Portorož, Slovenia, 23–28 May 2016; pp. 3686–3693.
19. Alam, M.; ul Hussain, S. Sequence to sequence networks for Roman-Urdu to Urdu transliteration. In Proceedings of the 2017 International Multi-topic Conference (INMIC), Lahore, Pakistan, 24–26 November 2017; IEEE: Piscataway, NJ, ,USA, 2017, pp. 1–7.
20. Jawaid, B.; Kamran, A.; Bojar, O. *Urdu Monolingual Corpus*; LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University: Prague, Czechia, 2014.
21. Bhat, R.A.; Bhatt, R.; Farudi, A.; Klassen, P.; Narasimhan, B.; Palmer, M.; Rambow, O.; Sharma, D.M.; Vaidya, A.; Ramagurumurthy Vishnu, S.; et al. The hindi/urdu treebank project. In *Handbook of Linguistic Annotation*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 659–697.

22. Palmer, M.; Bhatt, R.; Narasimhan, B.; Rambow, O.; Sharma, D.M.; Xia, F. Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In Proceedings of the The 7th International Conference on Natural Language Processing, Dalian, China, 24–27 September 2009; pp. 14–17.
23. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
24. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *2*, 3111–3119.
25. Boroş, T.; Dumitrescu, Ş.D.; Burtica, R. NLP-Cube: End-to-end raw text processing with neural networks. In Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, 31 October–1 November 2018; pp. 171–179.
26. Müller, T.; Cotterell, R.; Fraser, A.; Schütze, H. Joint lemmatization and morphological tagging with lemming. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2268–2274.
27. Chrupala, G.; Dinu, G.; van Genabith, J. Learning Morphology with Morfette. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 28–30 May 2008; European Language Resources Association (ELRA): Marrakech, Morocco, 2008.
28. Yildiz, E.; Tantuğ, A.C. Morpheus: A neural network for jointly learning contextual lemmatization and morphological tagging. In Proceedings of the 6th Workshop on Computational Research in Phonetics, Phonology, and Morphology, Florence, Italy, 2 August 2019; pp. 25–34.
29. Chakrabarty, A.; Pandit, O.A.; Garain, U. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1481–1491.