NLPashto: NLP Toolkit for Low-resource Pashto Language

Ijazul Haq, Weidong Qiu, Jie Guo, Peng Tang
School of Cyber Science and Engineering,
Shanghai Jiao Tong University, China

***Abstract*—In recent years, natural language processing (NLP) has transformed numerous domains, becoming a vital area of research. However, the focus of NLP studies has predominantly centered on major languages like English, inadvertently neglecting low-resource languages like Pashto. Pashto, spoken by a population of over 50 million worldwide, remains largely unexplored in NLP research, lacking off-the-shelf resources and tools even for fundamental text-processing tasks. To bridge this gap, this study presents NLPashto, an open-source and publicly accessible NLP toolkit specifically designed for Pashto. The initial version of NLPashto introduces four state-of-the-art models for Spelling Correction, Word Segmentation, Part-of-Speech (POS) Tagging, and Offensive Language Detection. The toolkit also includes essential NLP resources like pre-trained static word embeddings, Word2Vec, fastText, and GloVe. Furthermore, we have pre-trained a monolingual language model for Pashto from scratch, using the Bidirectional Encoder Representations from Transformers (BERT) architecture. For the training and evaluation of all the models, we have developed several benchmark datasets and also included them in the toolkit. Experimental results demonstrate that the models exhibit satisfactory performance in their respective tasks. This study can be a significant milestone and will hopefully support and speed-up future research in the field of Pashto NLP.**

***Keywords*—NLP; text processing; word segmentation; POS tagging; BERT, LLMs; Pashto; low-resource languages; CRF; CNNs; RNNs**

I. Introduction

Pashto is an Indo-European language primarily spoken in Afghanistan and Pakistan. Written in the Perso-Arabic script, Pashto is also cursive, written right-to-left (RTL), and the letters take on different forms depending on their position in the word. Despite being the native language of a large population, NLP research in Pashto is still very rare, and sophisticated tools can hardly be found even for basic text-processing tasks, such as word segmentation and spelling correction. Additionally, the tools developed for other languages are not sufficient for Pashto text processing due to the complex and unique morphology of the language. Pashto is not a standardized language, lacking any golden rules to impose a uniform way of writing. For instance, it is a good practice to add space after each word in typing (writing with a keyboard), but in some cases, space omission is acceptable for human readers, which creates an issue for NLP applications. Without space between words, a naive NLP algorithm will consider the whole string as a single word. Due to this inconsistency in writing, any arbitrary Pashto text, whether on social media, news websites, or even books, is noisier compared to any other language.

This study aims to address the challenges in Pashto text processing and develop essential resources and state-of-the-art (SOTA) models for preliminary NLP tasks. All these resources and models, along with the benchmark datasets, are packaged in a toolkit named NLPashto, publicly available on GitHub and PyPi hub. The objective of toolkit is to enhance re-usability, avoid reinventing the wheel, and provide a single point of entry for further research in Pashto NLP. The initial prototype of NLPashto includes static word embeddings (Word2Vec, fastText and Glove) and the first monolingual Pashto BERT, pre-trained on our custom-developed Pashto text corpus of 15 million words. The toolkit includes four SOTA models, three of which are general-purpose tools for basic text processing: spelling correction, word segmentation, and POS tagging, while the fourth model is for offensive language detection.

Most NLP algorithms require the input text split into individual units before processing. Two baseline techniques commonly used for converting text into tokens/words: whitespace tokenization and lexicon-based word segmentation. In whitespace tokenization, words are separated by whitespaces (spaces, tabs, or line breaks), commonly used in Western languages like English. On the other hand, languages that do not have spaces between words, such as Chinese and Japanese, use word segmentation, which is typically more complex because it involves identifying the boundaries between words and deciding which character belongs to which word. However, none of these techniques is perfect for the Pashto language. In Pashto, unlike English, the ”space” is not consistently used for word separation and is not a reliable word delimiter, and unlike Chinese, space is a part of writing and cannot be completely ignored. To handle these limitations, we have developed two specialized machine learning models, one for spelling correction and the other for word segmentation. The spelling correction model can be used to identify the proper position of spaces in the text, remove extra spaces, or insert spaces where required. Once the spaces are corrected, we can use the baseline whitespace tokenizer to convert the text into tokens. The word segmentation model can be used in applications that need to split the text into ”full” words rather than space-delimited tokens. For example, a NER application may need to extract ”whole” words from text like ”New York” rather than ”New” and ”York”. Therefore, we have developed a specialized word segmenter for the Pashto language that will not break the compound words, such as واره خواره (dispersed) or چیان مظاهره (protesters).

POS tagging is also a fundamental text pre-processing task that involves assigning a grammatical category to each word in a sentence, such as a noun, verb, adjective, or adverb. POS information is very helpful for AI models better understand the language, as they can learn more about the grammatical structure of the text, which could improve models' ability to generate coherent and grammatically correct text. It is one of the earliest types of annotation performed on corpora and is still used, for example, BNC [1], Brown Corpus and LCMC. POS tagging is not intuitive, as a particular word can have different tags based on the context. For example, in the sentence وريکي واوره غرونو په (it is snowfall on the mountain), the word واوره is a noun, while in the sentence واوره خبره زما (listen to me), واوره means "listen," which is an imperative verb. For automatic POS tagging, we have developed a machine learning-based POS tagger and included it in the toolkit.

The rise of social media has led to an increase in the dissemination of offensive language, which has a profound negative impact on the targeted individuals and the community. The sheer volume of content posted everyday, makes the manual removal of offensive content by human moderators unfeasible. Therefore, automated NLP systems for detecting offensive content have become essential. Significant research has been dedicated to this area in other languages, such as English [2], Chinese [3], Arabic [4], [5], [6], [7], Hindi [8], and German [9], to name a few. However, for the Pashto language there is no such research work available. In this study, we have developed a SOTA AI model for Pashto offensive language detection, trained on a dataset of tweets manually categorized as "offensive" and "not-offensive."

Toolkit development is an ongoing process, and we are actively working on NLPashto to make it more inclusive, though the progress we have already achieved can be summarized as follows:

We developed a *Pashto text corpus* of around 15 million words and used it to pre-train the *Static Word Embeddings* for Pashto. We developed the first monolingual *Pashto BERT* from scratch. We developed benchmark datasets and used them to develop four SOTA models for *Spelling Correction*, *Word Segmentation*, *POS Tagging*, and *Offensive Language Detection*. Finally, we packaged all the resources, benchmark dataset, and pre-trained models in a *Toolkit* and distributed them publicly on GitHub and PyPi hub to facilitate and speed up future research in this domain.

II. Related Work

Natural language toolkits have been utilized in the research and development of various NLP applications. One of the most widely known NLP toolkits is the Natural Language Toolkit (NLTK) [10], which is a Python library providing a comprehensive suite of tools and resources for NLP tasks, including tokenization, POS tagging, NER, sentiment analysis, and more. Another popular NLP toolkit is CoreNLP [11], which offers a set of core NLP tools similar to NLTK, such as tokenization, POS tagging, parsing, and NER. It also encompasses advanced tools like coreference resolution, relation extraction, and sentiment analysis.

In recent years, language-specific NLP toolkits have gained prominence, offering SOTA tools, datasets, and pre-trained models for specific languages. An example of such toolkits is FudanNLP [12] for the Chinese language. FudanNLP employs statistics-based and rule-based methods to tackle various NLP tasks, including word segmentation, POS tagging, NER, dependency parsing, anaphora resolution, and time-phrase recognition. For Urdu, a sister language of Pashto, [13] developed the UNLT toolkit, which includes three preliminary NLP tools: word tokenizer, sentence tokenizer, and part-of-speech tagger. The word tokenizer utilizes a morpheme-matching algorithm combined with a stochastic n-gram model. The toolkit addresses space-omission through back-off and smoothing characteristics, and space-insertion is handled using a lexicon-based look-up technique. The POS tagger is based on HMM entropy-based stochastic and lexicon-based look-up techniques. InaNLP [14] is a natural language toolkit for the Indonesian language, which integrates several NLP modules, such as tokenization, sentence splitter, POS tagger, NER, syntactic parser, and semantic analyzer, with most of the models being rule-based. [15] developed a toolkit named CAMeL for the Arabic language. CAMeL provides tools for preliminary NLP tasks, including morphological analysis, dialect identification, and NER, with support for various Arabic dialects. DaNLP [16] is another toolkit for low-resource languages, specifically designed for Danish. It contains pre-trained models for NER, POS tagging, coreference resolution, and sentiment analysis and also includes benchmark datasets and static word embeddings. IceNLP, developed by [17], is an NLP toolkit for the morphologically complex Icelandic language. It encompasses essential pre-processing tools, such as tokenizer, sentence segmenter, rule-based taggers, finite-state parser, and morphological analyzer. For Vietnamese, [18] have developed VnCoreNLP, a toolkit that provides solutions for preliminary NLP tasks such as word segmentation, POS tagging, NER, and dependency parsing. Lastly, [19] developed an NLP toolkit for the Bengali language, which includes supervised machine learning models for tokenization, POS tagging, and NER, as well as other resources like word embeddings.

III. Challenges in Pashto Text Processing

Pashto is not a standardized language, and there are no golden rules for the proper usage of space in the writing system, which leads to two typical spelling errors, space omission, and space insertion error. Besides that other challenges in Pashto text processing include non-standardized transliteration and homograph ambiguity.

A. Space-omission Errors

A space-omission error occurs when the space between two words is ignored, causing the words to merge into a single string. For example, the phrase داميز (this table) has two words, دا (this) and ميز (table), but the space between the words is omitted, which is perfectly readable to a human reader and thus it is not considered to be a typo in Pashto; however, an NLP system will interpret and process the phrase as a single word.

B. Space-insertion Errors

A space-insertion error occurs when a "useless" space is inserted within a word, splitting it into two or more (possibly meaningless) parts. For example, the words خبريال (reporter) and خبر يال look very similar, but the latter one has an extra space between the two ligatures خبر and یال. A human reader may consider it correct, but an NLP application treat each ligature as a separate word. An example sentence in Fig. 1, highlights the issues of both the space-omission and space-insertion errors.

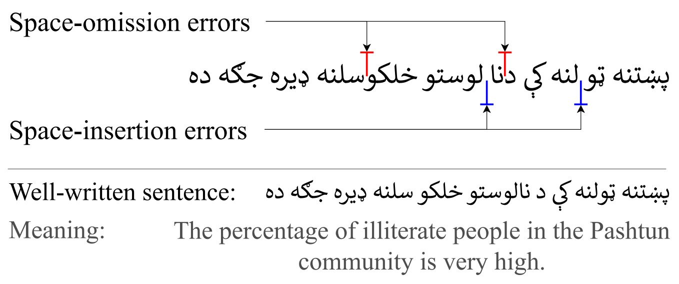
Space-omission errors
پښتنه ټولنه کې دنا لوستو خلکوسلنه ډیره جګه ده
Space-insertion errors
Well-written sentence: پښتنه ټولنه کې د نالوستو خلکو سلنه ډیره جګه ده
Meaning: The percentage of illiterate people in the Pashtun community is very high.

Figure 1. Example of space-omission and space-insertion errors.

C. Non-standardized Transliteration

Transliteration is the process of converting the characters of one writing system into another, preserving the pronunciation of original words. Translitration is very common in Pashto, but there are no standard rules to enforce one common spelling for foreign words transliterated into Pashto. The spelling of transliterated words usually depends on the choice of the writer (translator or transliterator), resulting in several variants of spellings for one word. For instance, the word "Coronavirus" has more than ten variants in our Pashto text corpus, e.g., کرونا وائرس، کورونا وائرس and کروناوایرس.

D. Homograph Ambiguity

Homograph ambiguity is a common issue, not only in Pashto but in many languages, which occurs when a word has more than one meaning. It can create ambiguity in the language, as the meaning of a homograph may be unclear based solely on its spelling. For example, the word "tear" in English can refer to a "liquid from the eyes" or "to rip apart forcefully". Similarly, in the Pashto sentence, هغه تور کوټ اغوستی (he is wearing a black jacket), the word تور is an adjective, means "black", while in the sentence د دې تور هغوی په ګاونډي هیواد لګوي (they blame the neighboring country for it), the word تور refers to "blame", which is a Common Singular Masculine Noun.

IV. Text Corpus and Benchmark Datasets

This project involves the development of word embeddings and a BERT model. Training these models requires a substantial corpus of text that should be diverse and representative of the entire language. However, Pashto is a low-resource language, where electronic textual content, large-scale corpora, and well-organized datasets are hard to find. Therefore, we developed a Pashto text corpus, for which we collected text from four primary sources: news websites, Wikipedia articles, books, and Twitter tweets. The corpus size is over 15 million tokens, with an average token length of 3.6 characters.

A. Dataset for Spelling Correction and Word Segmentation Models

To develop the dataset for spelling correction, the first step was to obtain raw sentences. We used our Pashto text corpus, which consisting around 400K sentences. Our goal was to annotate each sentence for explicit word-boundary information. However, manually annotating such a large dataset was not feasible. So, we initially employed a lexicon-based approach to mark word boundaries.

Lexicon-based segmentation is an intuitive technique often used to divide text into words [20], [21]. It involves scanning a sequence of input characters and matching them against a lexicon of words. If the sequence is found in the lexicon, it is considered a word. To ensure matching the longest possible sequence, a variant of the lexicon-based approach called the Longest Matching (LM) algorithm is used. The LM is a greedy algorithm that strives to find the longest sequence. Since the LM algorithm starts the search from the beginning and moves forward, it is also called Forward Longest Matching (FLM). Another variant of the LM algorithm performs the search in the backward direction, known as Reverse Longest Matching (RLM). Sometimes, both FLM and RLM are combined to form Bidirectional Longest Matching (BMM).

To annotate our corpus, we incorporated the BLM technique with a small modification. Instead of looking up sequences of characters, we looked up sequences of tokens obtained from whitespace tokenization. A space in Pashto can either be a word delimiter or part of the word, where the purpose of annotation was to discriminate these spaces and mark them with explicit labels. We used the label "B" for word delimiters and "S" for the spaces that were part of the words. In the first round, after passing all 400K sentences through the lexicon-based model, 95K sentences were "fully" annotated, where no out-of-vocabulary (OOV) tokens were found. The "partially" annotated sentences, where at least one token was OOV, were further processed. The OOV tokens were extracted, manually inspected, and added to the lexicon if they were valid Pashto words. These partially annotated sentences once again passed through the lexicon-based segmenter, where some more sentences were fully annotated. This process was repeated several times with an updated lexicon each time. Finally, the size of the dataset reached 150K sentences (nearly 4 million words). It is worth mentioning that we used the same dataset for training both the spelling correction and word segmentation models.

B. Part-of-Speech Dataset

To develop the POS dataset, we annotated the spelling correction and word segmentation dataset with POS information. For POS annotation, we initially employed the lexicon-based approach, in which the words in the sentences are looked up one by one in the lexicon and labeled with the corresponding POS tags. It is a context-free approach in which the surrounding information of the taken are not taken into consideration. In the first round, after passing all 150K sentences through the lexicon-based model, 80K sentences were fully annotated, with every word assigned a POS tag. Two example sentences are given in Fig. 2, annotated using this approach. In both sentences, the word تور has been assigned the Singular Masculine Noun tag, though, in the first sentence تور means (black), which is an Adjective.

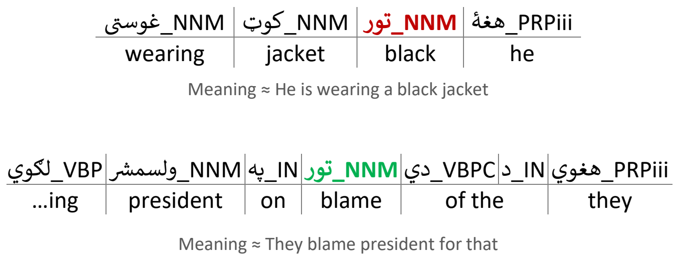
غوسټی_NNM | کوټ_NNM | تور_NNM | هغه_PRPiii
wearing | jacket | black | he
Meaning ≈ He is wearing a black jacket
لګوي_VBP | ولسمشر_NNM | په_IN | تور_NNM | دي_VBPC | د_IN | هغوي_PRPiii
...ing | president | on | blame | of the | they
Meaning ≈ They blame president for that

Figure 2. Example sentences, annotated using the lexicon-based technique.

After the lexicon-based annotation, we randomly selected 10K of the sentences for manual correction. Using a specialized web application we developed, sentences from the database were presented one by one to the annotators (human experts), with each word already assigned a static tag (by the lexicon-based tagger), and the job of the annotators was to verify or change the tags. This way, all 10K sentences were annotated for POS information with 100% (theoretical) accuracy. By analyzing the results of manual correction, we found that changes were made to around 7% of the tags by human annotators. It shows that the lexicon-based POS tagger can achieve an accuracy of 93%.

C. Pashto Offensive Language Dataset

The Pashto Offensive Language Dataset (POLD) is a collection of tweets manually categorized into two classes: ”offensive” and ”not-offensive” (Fig. 3). The first step in creating the POLD dataset was to collect raw tweets in Pashto from Twitter. However, offensive tweets only make up a small portion of overall tweets, therefore, annotating random tweets was inefficient. To increase the size of the offensive class, we used a seed list of offensive words to filter tweets. To minimize bias and maintain diversity in the dataset, firstly, we made the seed list large and inclusive, and secondly, we analyzed many tweets and observed common patterns in offensive tweets, such as the use of second-person pronouns, i.e., تا (you: singular) or تاسو (you: plural), and included these patterns in the seed list. Using the Twitter Search API, we searched for each word and pattern in the seed list and collected nearly 300K raw tweets between January 10 and February 10, 2023.

The tweets corpus underwent several pre-processing steps, which involved removing HTML tags, URLs, usernames, and other special characters. Digits in non-Pashto format were normalized to the Pashto format, i.e., 1234 became . Duplicate tweets were deleted, and tweets with less than 10 characters or more than 150 characters were also discarded. The final corpus size dropped to 70K tweets, from that we randomly selected 35K (50%) for manual annotation.

The manual annotation was carried out by a total of five participants, including one of the authors and four paid professionals. Tweets containing any type of offensive language, such as hate speech, cyberbullying, aggression, abuse, or profanity, were assigned the label ”1” (offensive), and the rest (normal or positive tweets) were assigned the label ”0” (non-offensive). Each annotator individually tagged the complete corpus without knowing the decisions of other annotators. The decision regarding the final status of the tweets made by a majority vote. The final POLD dataset consists of 34,400 tweets, with 12,400 labeled as offensive and 22,000 labeled as non-offensive.

V. Word Embeddings

Word embedding is a process that involves mapping words from a vocabulary to vectors of real numbers. The basic idea behind word embeddings is to learn a distributed representation of words based on their co-occurrence in a large corpus of text. A neural network is trained to capture the syntactic and semantic meaning of the words in the text. The network learns to associate words that appear in similar contexts with similar vector representations. The resulting vector representations can then be used as input for other machine learning models. NLP researchers generally prefer to utilize pre-trained word embeddings, typically trained on extensive corpora. However, for Pashto, no pre-trained word embeddings are currently available except fastText. Nevertheless, we pre-trained static word embeddings and a BERT model for the Pashto language and included them in NLPashto.

A. Static Word Embeddings

We used the Pashto text corpus and trained the three popular types of static word embeddings: Word2Vec, fastText, and GloVe, and included them in the toolkit. For all three models, most of the hyper-parameters were kept uniform. The vector size was fixed at 100, the window size at 5, and the minimum count at 2, which is the minimum frequency needed for a word to be included in the final vocabulary. We chose the skip-gram architecture for Word2Vec and fastText and trained each model for 5 epochs. The GloVe model was trained using the GloVe package, while Word2Vec and fastText were trained using the Gensim and fastText Python libraries.

B. Pashto BERT

The recent progress in Large Language Models (LLMs) has revolutionized the field of NLP. Some of the most popular LLMs in use today include BERT [22], GPT [23], XLM-R [24], and RoBERTa [25], to name a few. LLM takes into account the context in which a word appears in a sentence when generating the embeddings, which allows the model

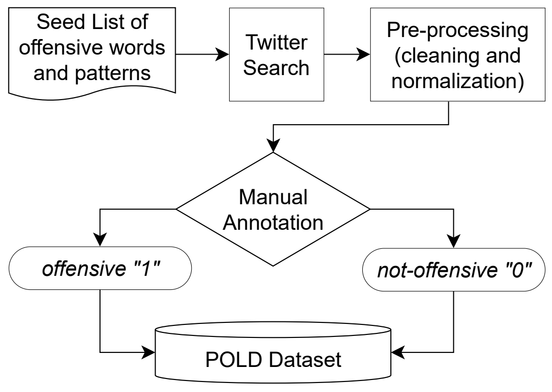
Seed List of offensive words and patterns
Twitter Search
Pre-processing (cleaning and normalization)
Manual Annotation
offensive "1"
not-offensive "0"
POLD Dataset

Figure 3. Procedure of POLD dataset development.

to capture the meaning and nuances of words in different contexts. These models are usually trained on multilingual data and can understand and generate text in several languages. However, for language-specific tasks, monolingual models generally outperform the multilingual models. In this study, we have trained a Pashto monolingual language model and included it in the NLPashto toolkit as well as publicly uploaded it to the Huggingface hub.

The model we utilized for training Pashto BERT is the $\text{BERT}_{\text{Base}}$, which has 12 layers, 768 hidden states, 12 attention heads, and 110M parameters. Training data is our Pashto text corpus of 15 million tokens. To tokenize the input sequences, we used WordPiece tokenizer [26], which is the recommended tokenizer for $\text{BERT}_{\text{Base}}$. We fixed the vocabulary size at 30K words and used special [CLS] and [SEP] tokens at the beginning and end of the sequences, respectively. To enable the model to differentiate between original and padded tokens, we employed an attention mask to generate a vector of 1s and 0s for each input sequence, where 0s indicate the padded tokens and 1s indicate the original ones. The hyper-parameters setup for pre-training is shown in Table I. We implemented the model architecture and training pipeline using PyTorch and the Huggingface transformers library, where training of the model performed on a cloud GPU – NVIDIA Tesla P100 that took nearly 2 hours to complete.

Table I. Hyper-parameters Setup for Pre-training the Pashto BERT Model

Hyper-parameters	**Values**
Batch Size	32
Sequence Length	128
Padding	Post-padded
Learning Rate	1e-4
Linear Warmup Schedule	10K steps
β_1 and β_2	0.9, 0.999
L2 Weight Decay	0.01
Epsilon	1e-8

VI. Models

The initial prototype of the NLPashto toolkit includes four SOTA AI models for (i) Spelling Correction, (ii) Word Segmentation, (iii) POS tagging, and (iv) Offensive Language Detection. The first three are essentially sequence tagging models that involve assigning a label or tag to each element in the sequence of input data. For sequence tagging tasks, NLP researchers use several supervised machine learning algorithms, such as HMM (Hidden Markov Model) [27], [28], RNN (Recurrent Neural Networks) [29], [30], [31], [32], [33], and CRF (Conditional Random Fields) [34], [35], [36], [37]. For various sequence tagging tasks, such as Word Segmentation and POS tagging, the CRF usually outperforms the other models. We have also incorporated CRF for training the three sequence tagging models. On the other hand, the model for offensive language detection is essentially a (binary) sequence classifier, which discriminates the input sequences into two categories, offensive and not-offensive. For that, we fine-tuned our pre-trained Pashto BERT model.

A. Spelling Correction Model

The spelling correction module is aimed to remove the two typical spelling errors, the space-omission, and space-insertion. This model will predict the correct position of space in the text, insert a space where necessary, remove extra spaces from the sequence, and will return the noise-free text with the minimum required spaces.

1) Features for Spelling Correction Model: For modeling the spelling correction task, the “character” was considered as the basic text unit (like Chinese), and Pashto text was formalized as a series of characters and intervals as shown in Eq 1.

$$C_1 I.C_2 I.C_3 I...IC_n \quad for \;\; I \epsilon \{J, S\} \tag{1}$$

where C means a character, and I means an interval between the two characters, where the interval can be a space or ”none”. Each character in the dataset was assigned with one of the two tags: an “S” if the character is followed by a space or a “J” otherwise, where ”J” stands for “Joined”. Features for training the spelling correction model are the target character itself and n-grams of characters before and after the target character, based on which the CRF algorithm predicts whether the character is followed by a space or not. A summary of the features for the spelling correction model is as follows:

- The target character C
- Check if C is the first character in the sentence, last on neither
- n-grams of characters before and after C, where the value of n ranges between 1 and 4

2) Experimental Results: For the experiment, we used the sklearn-crfsuite library in Python. Hyper-parameters were tweaked for optimal results, where both c_1 and c_2 were set to 0.1, representing L_1 and L_2 regularization, respectively. The L-BFGS method was selected as the training algorithm. The dataset of labeled characters was converted into a CRF-friendly dataset of features. 80% of the dataset was used for training, and the remaining portion was used to test the model.

The spelling correction model achieved an F1-score of 99.16% with an accuracy of 99.35%. The contribution of different pairs of n-grams and their combined effect is given in Table II. The model performed its best for n-gram in the rage between 1 and 4, which is very logical considering the average token length of 3.6 characters. By utilizing 4-grams before and after the target character, the model captures information from a total of 9 characters. Overall the model’s performance is quite satisfactory, making it useful in practical applications.

Table II. Contribution of the Previous and Next n-grams and their Combined Effect in Model’s Performance, in Terms of F1-score (%)

n-grams	Previous	Next	Combined
1	82.42	88.18	91.22
2	88.25	93.78	96.99
3	92.86	96.61	98.87
4	94.61	97.44	**99.16**

B. Word Segmenter

Similar to the spelling correction model, the word segmenter is also a sequence tagging model based on the CRF algorithm. However, the word segmenter considers the token-level information instead of character-level information. A token is the basic unit for feature extraction, which can be the character in Chinese [31] and Javanese [34], syllable in Burmese [38], or character cluster (KCC) in Khmer [39]. In the proposed Pashto word segmenter, a token is a space-delimited "string" of characters, which can be a single character, such as د (of), or a string of any arbitrary length like افغانستان (Afghanistan), which has nine characters.

1) Features for Word Segmenter: For modeling the word segmentation task, the token was considered as the basic unit, and Pashto text was formalized as a series of tokens and intervals as shown in Eq. 2.

$$T_1 I.T_2 I.T_3 I...IT_n \quad for\ T \epsilon \{S, B\} \tag{2}$$

where T means a token, and I means Interval (or space) between two tokens. The space can be a word-delimiter or a separator between two ligatures of a compound word. All the tokens in the dataset were assigned one of the two tags, an "S" if the token is followed by a "simple" space or a "B" if the token is followed by a word-breaker (word boundary). Pashto is a language with a rich morphology, and words are inflected to express various grammatical and syntactic information. These inflections are mostly exhibited in the form of prefixes and suffixes of the words, which are very informative and, in most cases, enough for locating the word boundary. The features for the word segmenter include these morphological attributes and context information of the token. Following is the list of features, finalized after trial and error:

- The *token*
- Length of the *token*
- One and two characters prefixes and suffixes of the *token*
- All the above features for previous and next token
- Three-characters prefix and suffix of the *token*
- Is *token* first in the sentence, last or neither
- Is *token* numeric
- Previous and next tokens up to two places

2) Experimental Results: The contribution of various features and their combined effect on the model's performance is presented in Table III. The results show that without context information (Previous and Next tokens), the model achieves an F1-score of 79.87%. However, combining the features of surrounding tokens increases the F1-score to 96.81%. It demonstrates the model's ability to predict the boundary of the word, even if it has not been encountered before, thereby overcoming OOV errors that occur in the baseline lexicon-based segmentation approach.

C. Part-of-Speech Tagger

Similar to the previous two models, the POS tagger is also modeled as a sequence tagger based on the CRF algorithm. However, the dataset for the POS tagger has 10K sentences, which is comparatively smaller. Unlike the lexicon-based word segmenter, the lexicon-based POS tagger yields a very high error rate (around 5%), leads to a laborious and time-consuming manual correction phase. To extend the size of the dataset with reasonable speed, we adapted an iterative approach where the model training and dataset development were carried out in parallel in several rounds, as shown in Fig. 4.

In the first round, we used the POS dataset of 10K sentences to train the initial prototype of the model. This model was then used to annotate another chunk of 10K sentences, followed by a manual correction phase add then added to the dataset. This process was repeated for several rounds, where each round has basically two phases, an automatic POS assignment, and manual correction. In each round, the amount of training data was increased by an amount of 10K sentences, and consequently, the accuracy of the model increased as well. With the reduction in the error-rate of the model, the burden on human annotators reduced as well, and the manual correction phase became less time-consuming.

1) Tagset: A Tagset is a list of POS labels/tags used to indicate the part-of-speech of each word in a text corpus. In this study, we used the tagset proposed in [40], which follows the naming convention similar to the Penn Treebank [41] that is one of the commonly adopted conventions by various corpora. Our tagset has a total of 38 tags, which is very concise and pragmatic and enough to encompass all the words. The disagreement of the researchers is respected, and a non-tagged version of the corpus is also included in the toolkit, which can be tagged using any preferred tagset.

2) Features for Part-of-Speech Tagger: The contexts used to predict the POS tag in Pashto are roughly similar to that used for English. These are the surrounding words and word components. Pashto has a similar or maybe richer morphology than English, where words are enriched by various affixes that

Table III. Contribution of Various Feature Sets and their Combined Effect on Model's Performance

Feature set	Accuracy (%)	F1-score (%)
Token	98.12	79.87
Previous	98.31	81.24
Next	98.61	86.34
Combined	99.65	**96.81**

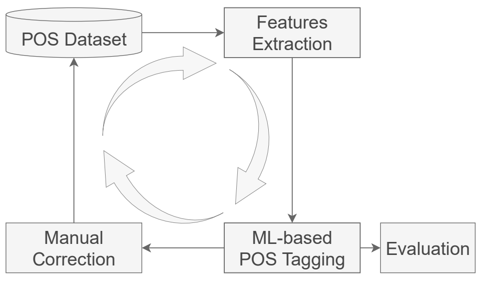

Figure 4. Iterative training of the POS tagger.

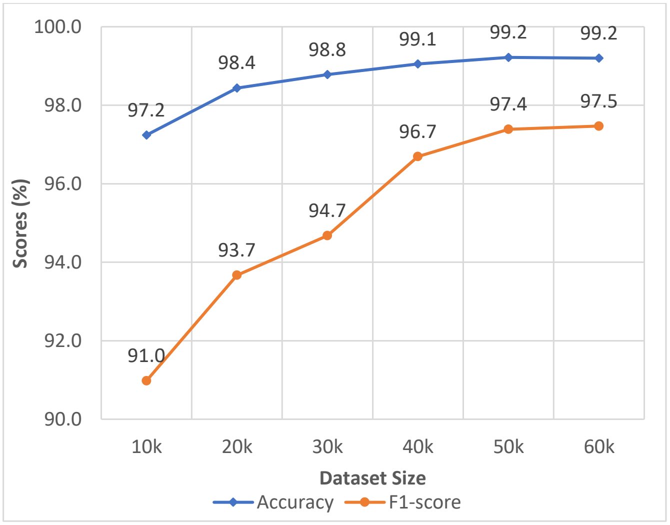

Figure 5. Improvement in performance of the POS tagger with the increasing dataset size. Lx.B represents the lexicon-based segmentation.

can be exploited for feature extraction. For training the POS tagger, we considered the same features as the word segmenter model, which are the morphological components of the token and the neighboring tokens.

3) Experimental Results: For the implementation and training of the POS tagger, we followed the same experimental setup as the previous two models, including the hyper-parameters and ratio of the dataset train and test splits. However, we trained the POS tagger in several rounds, starting from a dataset of 10K sentences and gradually increasing the dataset size until the model achieved significant performance. The model's performance was evaluated after each round, as plotted in the graph in Fig. 5. In the first round, the model secured an accuracy of 97.2% with an F1-score of 91.0%, which was an improvement of 4.2% in terms of accuracy in comparison to the baseline lexicon-based approach, which achieves an accuracy of 93.0%. After six rounds of training, the dataset size reached 60K sentences (nearly 1.5 million words), and consequently, the accuracy of the model reached 99.2% with an F1-score of 97.5%. This performance is significantly better than the baseline.

D. Offensive Language Detection Model

The model for Offensive language detection is based on the transfer learning approach. Conventional machine learning generally involves training a model from scratch using a large dataset, whereas transfer learning uses a pre-trained model as a starting point to solve a new task, called fine-tuning. The pre-trained model we fine-tuned is our Pashto BERT model, which is pre-trained on a generic text corpus; where the purpose of fine-tuning is to adapt it to the specific task of offensive language detection by fine-tuning its parameters on the labeled dataset, POLD.

To our knowledge, no previous research work is available on Pashto offensive language detection. Hence, for the evaluation, we also fine-tuned a multilingual language model, XLM-R [24]. Several other multilingual models are also available, though most of these are missing Pashto, while XLM-R can understand text in 100 languages, including Pashto. Besides transfer learning, we also investigated the classic neural network-based models, such as CNNs and various types of RNNs, using the static word embeddings, Word2Vec, GloVe, and fastText as features.

Table IV. Parameters for Fine-tuning the Transformer Models

Parameters	XLM-R	Ps-BERT
Learning Rate	2e-5	5e-5
Batch Size	16	16
Sequence Length	100	100
$\beta1$, $\beta2$	(0.9, 0.999)	(0.9, 0.999)

1) Fine-tuning BERT models: Fine-tuning a BERT model involves adding a classification layer on top of the model and training it on a specific dataset. For sequence classification, BERT takes the final hidden state of the classification token, identified by [CLS], as the representation of the whole sequence of text. We fine-tuned both the multilingual XLM-R and monolingual Pashto BERT on the task-specific POLD dataset. We tokenized the tweets and added special tokens [CLS] and [SEP] to mark the beginning and end of the sequence, respectively. However, the tokenizers used by the XLM-R and vanilla BERT are different, where XLM-R uses SentencePiece [42] tokenizer, while BERT expects the text to be tokenized by the WordPiece tokenizer. We implemented the models in PyTorch and Huggingface transformers library and used a GPU-facilitated Kaggle platform to conduct the experiments. We used the hyper-parameters given in Table IV and trained each model for 3 epochs.

2) Neural Network-based Models: As the baseline classifiers, we examined the performance of five neural networks, the CNNs, and four types of RNNs (LSTM, Bi-LSTM, GRU, and Bi-GRU), across three types of word embeddings: Word2Vec, fastText, and GloVe, as features. The primary components of our neural network models are the embedding layer, hidden layer, and output layer. The Embedding layer is the first hidden layer, which is a matrix of size pq, where p is the vocabulary size, and q is the sequence length, fixed at 64 tokens. To prevent overfitting, We used a dropout of 0.2. The output layer employs the Sigmoid activation function and Adam optimizer and uses cross-entropy loss to predict the tweet's category.

Besides the upper-mentioned common components, each model has its own adaptation and hyper-parameters setup. For CNNs, we constructed a 1D convolutional layer with 100 filters and a kernel size of 4. The next layer is max-pooling, followed by a dropout layer, and finally the output layer. The LSTM model has one LSTM layer with 100 units and a dropout layer, followed by a classification layer. The same architecture is used for the GRU model also, with the LSTM layer replaced by GRU. To build the Bidirectional LSTM, we construct one Bi-LSTM layer with 100 hidden units. The output vectors are flattened and fed to the classification layer. The Bi-GRU is using the same configuration of Bi-LSTM, except for the first layer which is replaced by the Bi-GRU. We used the batch size of 32 and trained each model for 5 epochs.

3) Experimental Results: We performed a series of experiments investigating various models for the task of Pashto

Table V. Comparison of All the Models for Offensive Language Detection

Model		Accuracy (%)	F1-score (%)
Features	**Classifier**		
Word2Vec	BiGRU	92.33	91.50
	BiLSTM	92.85	92.09
	CNN	90.29	89.08
	GRU	92.94	92.18
	LSTM	93.23	92.52
GloVe	BiGRU	93.40	92.74
	BiLSTM	93.40	92.76
	CNN	92.97	92.24
	GRU	93.43	92.75
	LSTM	93.40	92.78
fastText	BiGRU	93.46	92.82
	BiLSTM	93.49	92.81
	CNN	92.24	91.44
	GRU	93.49	92.82
	LSTM	93.72	93.08
XLM-R		94.48	94.01
Ps-BERT		94.77	**94.34**

offensive language detection. A performance evaluation is presented in Table V. The results exhibit that the transformer models achieve comparatively better performance than the classic neural networks. Among all the models we examined, the fine-tuned monolingual Pashto BERT demonstrates the best performance which yields an F1-score of 94.34% with an accuracy of 94.77%. The XLM-R performed poorly compared to the Pashto BERT, yet better than the neural networks. Concerning the neural network models, the results indicate that the RNNs performed better than CNNs, where the LSTM classifier with fastText embeddings outperforms the other models and achieve an F1-score of 93.08% with an accuracy of 93.72%. In bidirectional RNNs, BiGRU performs the best with fastText features and achieves an F1-score of 92.82% with an accuracy of 93.46%. On the downside, the CNN model with Word2Vec embeddings exhibits the lowest performance.

E. Comparison of Static Word Embeddings

Fig. 6 illustrates the performance comparison of the static word embeddings, using the LSTM classifier. The results show that the fastText achieves the highest F1-score of 93.08% with an accuracy of 93.72%. One reason is that the fastText model uses sub-word tokenization, which is particularly useful for the task of offensive language detection, as the OSN users commonly write half words instead of the full form, or use alteration. For example, on English social media, words like "f*ck, "b!tch", "c#ck", etc., are commonly used, where the same convention is used in Pashto also. This way of writing often leads to OOV errors in Word2Vec and GloVe, while in fastText, if a word is not present in the vocabulary the sub-words might be, which is useful in obtaining representations for altered, misspelled, or half-words.

VII. Conclusions

Pashto is a low-resource language and lakes the basic tools and resources required for NLP. This study aimed to develop SOTA models, benchmark datasets, and other preliminary resources necessary for the research in Pashto NLP. To facilitate the reuse of our findings, we packaged everything in a toolkit called NLPashto and distributed it publicly on GitHub[1] and PyPi[2] hub. The initial prototype of NLPashto consists of three general-purpose models for basic text pre-processing, a spelling correction model, a word segmenter and a POS tagger, and a special-purpose model for detecting offensive language (particularly on social media). Additionally, the toolkit includes three pre-trained static word embeddings: Word2Vec, fastText, and GloVe, as well as a pre-trained monolingual Pashto BERT for dynamic word embeddings. All the SOTA AI models we developed are based on the supervised learning approach, trained on labeled datasets. The toolkit also includes the benchmark datasets we developed for training and evaluating the models. The evaluation results show that our Pashto BERT model outperforms the multilingual XLM-R, even though the corpus used for training the Pashto BERT is much smaller in comparison to the XLM-R corpus. Similarly, all the other models included in the NLPashto toolkit perform quite satisfactorily on their respective tasks and can be used in practical applications. In summary, this is a pioneering study on Pashto NLP, and we hope that our findings and the resources and tools we developed will facilitate and speed up future research in this domain.

Toolkit development is an ongoing process, and we are continuously working to add more modules in the upcoming prototypes of NLPashto, such as NER and Constituency and Dependency Parsing. Apart from that, there are several research areas yet to be explored in Pashto NLP, such as stemming, lemmatization, machine translation, text-to-speech, and speech-to-text.

References

[1] B. Consortium *et al.*, "British national corpus," *Oxford Text Archive Core Collection*, 2007.

[2] S. Khan, M. Fazil, V. K. Sejwal, M. A. Alshara, R. M. Alotaibi, A. Kamal, and A. R. Baig, "Bichat: Bilstm with deep cnn and hierarchical attention for hate speech detection," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, pp. 4335–4344, 2022.

[3] J. Deng, J. Zhou, H. Sun, F. Mi, and M. Huang, "Cold: A benchmark for chinese offensive language detection," *arXiv preprint arXiv:2201.06025*, 2022.

[1]NLPashto on GitHub: https://github.com/ijazul-haq/nlpashto
[2]NLPashto on PyPi Hub: https://pypi.org/project/nlpashto/

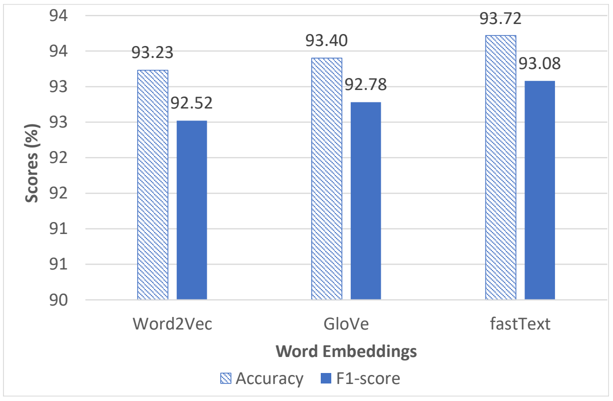

Figure 6. Comparison of the static word embeddings using LSTM classifier.

[4] H. Mubarak, K. Darwish, and W. Magdy, "Abusive language detection on arabic social media," in *Proceedings of the first workshop on abusive language online*, 2017, Conference Proceedings, pp. 52–56.

[5] S. Alsafari, S. Sadaoui, and M. Mouhoub, "Hate and offensive speech detection on arabic social media," *Online Soc. Networks Media*, vol. 19, p. 100096, 2020.

[6] A. Alakrot, L. Murray, and N. S. Nikolov, "Towards accurate detection of offensive language in online communication in arabic," in *International Conference on Arabic Computational Linguistics*, 2018, Conference Proceedings.

[7] M. J. Althobaiti, "Bert-based approach to arabic hate speech and offensive language detection in twitter: Exploiting emojis and sentiment analysis," *International Journal of Advanced Computer Science and Applications*, 2022.

[8] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri, "Benchmarking aggression identification in social media," in *Proceedings of the first workshop on trolling, aggression and cyberbullying (TRAC-2018)*, 2018, Conference Proceedings, pp. 1–11.

[9] J. Risch, A. Stoll, L. Wilms, and M. Wiegand, "Overview of the germeval 2021 shared task on the identification of toxic, engaging, and fact-claiming comments," in *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments*, 2021, Conference Proceedings, pp. 1–12.

[10] E. Loper and S. Bird, "Nltk: The natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, Conference Proceedings, pp. 55–60.

[12] X. Qiu, Q. Zhang, and X.-J. Huang, "Fudannlp: A toolkit for chinese natural language processing," in *Proceedings of the 51st annual meeting of the association for computational linguistics: system demonstrations*, 2013, Conference Proceedings, pp. 49–54.

[13] J. Shafi, H. R. Iqbal, R. M. A. Nawab, and P. Rayson, "Unlt: Urdu natural language toolkit," *Natural Language Engineering*, pp. 1–36, 2022.

[14] A. Purwarianti, A. Andhika, A. F. Wicaksono, I. Afif, and F. Ferdian, "Inanlp: Indonesia natural language processing toolkit, case study: Complaint tweet classification," in *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*. IEEE, 2026, Conference Proceedings, pp. 1–5.

[15] O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash, "Camel tools: An open source python toolkit for arabic natural language processing," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, Conference Proceedings, pp. 7022–7032.

[16] A. B. Pauli, M. Barrett, O. Lacroix, and R. Hvingelby, "Danlp: An open-source toolkit for danish natural language processing," in *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, 2021, Conference Proceedings, pp. 460–466.

[17] H. Loftsson and E. Rögnvaldsson, "Icenlp: a natural language processing toolkit for icelandic," in *INTERSPEECH*, 2007, Conference Proceedings, pp. 1533–1536.

[18] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, "Vncorenlp: A vietnamese natural language processing toolkit," *arXiv preprint arXiv:1801.01331*, 2018.

[19] S. Sarker, "Bnlp: Natural language processing toolkit for bengali language," *arXiv preprint arXiv:2102.00405*, 2021.

[20] R. Rashid and S. Latif, "A dictionary based urdu word segmentation using maximum matching algorithm for space omission problem," *2012 International Conference on Asian Language Processing*, pp. 101–104, 2012.

[21] P. Long and V. Boonjing, "Longest matching and rule-based techniques for khmer word segmentation," *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pp. 80–83, 2018.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[23] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *ArXiv*, vol. abs/2005.14165, 2020.

[24] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Annual Meeting of the Association for Computational Linguistics*, 2019.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019.

[26] M. Schuster and K. Nakajima, "Japanese and korean voice search," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012.

[27] P. Bheganan, R. Nayak, and Y. Xu, "Thai word segmentation with hidden markov model and decision tree," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2009.

[28] X. Yan, X. Xiong, X. Cheng, Y. Huang, H. Zhu, and F. Hu, "Hmm-bimm: Hidden markov model-based word segmentation via improved bi-directional maximal matching algorithm," *Comput. Electr. Eng.*, vol. 94, p. 107354, 2021.

[29] W. AlKhwiter and N. Al-Twairesh, "Part-of-speech tagging for arabic tweets using crf and bi-lstm," *Comput. Speech Lang.*, vol. 65, p. 101138, 2021.

[30] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for chinese word segmentation," in *Conference on Empirical Methods in Natural Language Processing*, 2015.

[31] Y. Jin, S. Tao, Q. Liu, and X. Liu, "A bilstm-crf based approach to word segmentation in chinese," *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 1–4, 2022.

[32] N. Qun, H. Yan, X. Qiu, and X. Huang, "Chinese word segmentation via bilstm+semi-crf with relay node," *Journal of Computer Science and Technology*, vol. 35, pp. 1115 – 1126, 2020.

[33] L. Wang and H. Yang, "Tibetan word segmentation method based on bilstm_ crf model," *2018 International Conference on Asian Language Processing (IALP)*, pp. 297–302, 2018.

[34] D. Tanaya and M. Adriani, "Word segmentation for javanese character using dictionary, svm, and crf," *2018 International Conference on Asian Language Processing (IALP)*, pp. 240–243, 2018.

[35] C. Ma and J. Yang, "Burmese word segmentation method and implementation based on crf," *2018 International Conference on Asian Language Processing (IALP)*, pp. 340–343, 2018.

[36] X. fei Zhang, H. Huang, and Z. Liang, "The application of crfs in part-of-speech tagging," *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, pp. 347–350, 2009.

[37] H. B. Zia, A. A. Raza, and A. Athar, "Urdu word segmentation using conditional random fields (crfs)," in *International Conference on Computational Linguistics*, 2018.

[38] C. Ding, Y. K. Thu, M. Utiyama, and E. Sumita, "Word segmentation for burmese (myanmar)," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 15, pp. 1 – 10, 2016.

[39] V. Chea, Y. K. Thu, C. Ding, M. Utiyama, A. Finch, and E. Sumita, "Khmer word segmentation using conditional random fields," *Khmer Natural Language Processing*, pp. 62–69, 2015.

[40] I. Haq, W. Qiu, J. Guo, and T. Peng, "The pashto corpus and machine learning model for automatic pos tagging," *Language Resources and Evaluation*, 2023.

[41] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Comput. Linguistics*, vol. 19, pp. 313–330, 1993.

[42] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *ArXiv*, vol. abs/1808.06226, 2018.