Automated Spell Checking in Nepali Texts Using LSTM and BiLSTM

Jagdish Bhatta[1] Kripa Shrestha[2] Nawaraj Paudel[3*]

[1, 2, 3] Central Department of Computer Science and IT, Tribhuvan University, Nepal

*** Correspondence Email:** *nawarajpaudel@cdcsit.edu.np*

Keywords:
Spell Checking, Nepali Texts, Natural Language Processing, LSTM, BiLSTM

Received: 11 November 2024
Revised: 8 December 2024
Accepted:29 December 2024

ISSN: 3102-0763 (Print)
3102-0771 (Online)

Abstract

Spell checking remains a fundamental requirement in natural language processing (NLP) applications ranging from word processors to chat bots. For Nepali texts, checking the spelling is crucial for Nepali grammar. This study includes using LSTM and BiLSTM models for the task of spell checking for Nepali texts. The dataset used consists of articles from online Nepali newspapers spanning various topics. Both of the models are trained to identify and correct spelling errors within a supervised learning framework. This study evaluates their performance using standard metrics including accuracy and BLUE score. The results demonstrate that the LSTM model achieved a character-level accuracy of 65.40% and a BLEU score of 0.4871 while the BiLSTM model has 75.12% character-level accuracy and a 0.5537 BLEU score respectively. The BiLSTM model outperforms the LSTM model in terms of accuracy and performance as well.

1. Introduction

Spell checking is a fundamental component of natural language processing and computational linguistics, aimed at detecting and correcting orthographic errors in written text. Automated spell checkers are crucial tools used in various applications like word processors, search engines, and messaging platforms. In languages like Nepali, which have relatively limited digital resources compared to high-resource languages, developing accurate spell checkers is especially important. Machine learning models based on recurrent neural network (RNN) are getting integral solutions for language processing. Long Short-Term Memory (LSTM) is a kind of recurrent neural network (RNN) that uses memory cells and gates to process sequential data and retain long-term relationships. BiLSTM (Bidirectional LSTM) is an expansion of LSTM that process data both forward and backward, enabling the model to include information from the past and the future.

Spell checking in morphologically rich languages like Nepali presents unique challenges due to complex grammar, phonetic variations, and keyboard-based typographical errors. Traditional rule-based and statistical methods often struggle with context-aware corrections. Deep learning approaches, particularly LSTM-based models, have shown promise in learning character-level patterns for error detection and correction. This study implements and analyses LSTM and BiLSTM models for the task of spell checking in Nepali text. The dataset used consists of articles from online Nepali newspapers,

spanning topics such as politics, sports, economics, and education. The objective of this study is to determine which of the RNN model is more effective in identifying and correcting spelling errors by analyzing their ability to understand context and structure in the Nepali language.

2. Literature Review

Long Short-Term Memory (LSTM) networks, introduced in (Hochreiter & Schmidhuber, 1997) are a type of recurrent neural network (RNN) designed to address the vanishing gradient problem in traditional RNNs. LSTMs use gating mechanisms (forget, input, and output gates) to retain long-term dependencies in sequential data, making them effective for tasks like machine translation, speech recognition, and spell checking.

Bidirectional LSTMs (BiLSTMs), an extension proposed in (Graves & Schmidhuber, 2005) process sequences in both forward and backward directions, capturing contextual information from past and future states. This makes BiLSTMs particularly useful in natural language processing (NLP) tasks where full-sequence context is crucial.

In a study on Nepali spell checking, authors have implemented a Nepali Spell Checker using GRU-based sequence-to-sequence (Seq2Seq) model to correct misspelled Nepali words for the health domain. They have trained the model for 10 epochs and resulted an accuracy of 75.11% (Prasain et al., 2022).

The authors in (Sharma et al., 2023) has built a multilingual spell checker which is used in search in adobe products. The spell checker handles English, French, and German queries. Their model consists of a suggester module and a ranker module. The suggester based on symmetric delete algorithm suggests possible corrected tokens while the ranker based on MLP rakgs the suggestions. Their model has outperformed Aspell and Neuspell with accuracy of above 80% for all three languages.

Authors in (Thapa et al., 2025) introduced Prasta Nepali, a system that uses transformer-based deep learning models to tackle error detection and correction in Nepali grammar. They have used encoder-decoder framework to compute the contextual relationships. Their findings with accuracy of 90.45% and (BLEU) scores of 0.9037, 0.8170, 0.7838, and 0.7694 for 1-gram to 4-gram demonstrated the significance of the transformer models for grammar checking in Nepali language.

Shashank Singh and Shailendra Singh presented a novel model HINDIA to handle the spelling errors of the Hindi language. The HINDIA is developed using the attention-based encoder–decoder bidirectional recurrent neural network (BiRNN) using long short-term memory cells. The model identifies erroneous words in input and replaces with most probable correct word. HINDIA performs significantly well with precision 0.86, recall 0.72, f-measure 0.78 and accuracy 0.80 (Singh & Singh, 2021).

In (Devkota et al., 2016), the authors have used decision tree with Damerau-Levenshtein distance method for spell checking of Romanized Nepali texts from the agricultural domain. They have integrated the spell checker for building the decision support system for Nepalese farmers. The authors have computed the confidence interval calculations for 100 sample size with confidence level of 95% and confidence interval of 8.12%.

A research for contextual spell correction of Nepali texts is done by authors in (Luitel et al., 2024). They have trained an autoregressive language model based on word-based vocabulary with around

350,000 tokens with two error models namely Constant Distributive (CD) and Brill and Moore (BM) and five language models KnLM-2, NepaliBert, deBerta, Trans and Trans-word. It is observed that the BM model better captures the errors than CD in terms of word accuracy and word error rate.

Authors in (Phukan et al., 2024) have used Long-short-term memory (LSTM) and Bidirectional Long-short-term memory (BiLSTM), to accurately detect misspelled words by analyzing the context of each word inside a sentence. The models are trained using a dataset containing Assamese sentences that include both correctly and incorrectly spelled words. The LSTM and BiLSTM resulted an accuracy of 92.72% and 94.59% of accuracy.

3. Methodology

Below figure outlines the research methodology using LSTM (Long Short-Term Memory) and BiLSTM (Bidirectional LSTM) models.

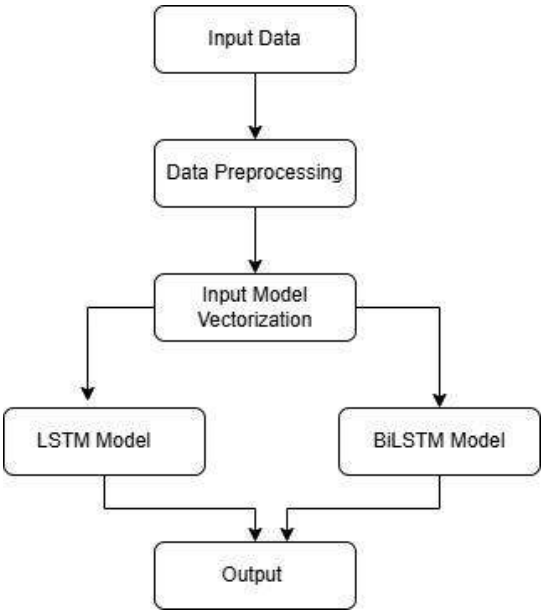

Figure 1: Methodology for spell checking using LSTM and BiLSTM

The methodology for Nepali spell correction using LSTM/BiLSTM involves preprocessing the input data by cleaning and filtering Nepali words, followed by character-level tokenization with special start/end tokens. Noisy data is generated by introducing common typos errors and random noise. The sequence-to-sequence model consists of an encoder (LSTM or bidirectional LSTM) and a decoder (LSTM with softmax output), trained using RMSprop optimizer and evaluated using character accuracy, word accuracy, and BLEU score. Training includes early stopping and learning rate reduction, with results visualized through loss/accuracy curves. The model is saved for inference,

where it corrects noisy inputs by predicting character sequences auto regressively until the end token is generated. Both LSTM and BiLSTM variants follow this pipeline, with BiLSTM offering enhanced context awareness.

3.1. Dataset Description

Source data was acquired through a public GitHub (Bhattarai, 2018) repository featuring a comprehensive collection of Nepali news articles from Nagarik and Setopati, covering primary domains like politics, sports, economics, and education with full Devanagari script preservation. The total number of words consists of 2803723 and out of them the number of unique words are 200431.

3.2. Data Preprocessing

The data preprocessing involves several critical steps to ensure the model can effectively learn and correct spelling errors. It includes generating a unigram frequency list from a large Nepali text corpus. Using a custom tokenizer that extracts Devanagari-script words via regular expressions, the words from multiple .txt files are processed and their frequencies are counted using Python's Counter. This frequency data is then compiled into a CSV file sorted in descending order.

Table 9: Data Instance

S. No.	Word	Frequency
1	पनि	32287
2	भएको	15574
3	गरेको	13216
4	गर्ने	13254
5	भने	15178

Then, the loading and preprocessing a dataset of Nepali words, specifically focusing on the Devanagari script. The preprocessing step removes any non-Nepali characters and filters out empty entries, ensuring clean input data. Character-level tokenization is then performed, mapping each Nepali character to a numerical index and adding special start (\t) and end (\n) tokens to mark sequence boundaries.

3.3. Spell Checking using LSTM

LSTM (Long Short-Term Memory) is a special type of Recurrent Neural Network (RNN) designed to effectively learn and remember long-range dependencies in sequential data. An LSTM cell processes sequential data using three gates namely Forget, Input, Output and a cell state to control information flow which are defined as;

Forget Gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

Input Gate: $i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$ $\quad \widetilde{C}_t = \tanh (W_c \cdot [h_{t-1}, x_t] + b_c)$

Cell State Update: $C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$

Output Gate: $o_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_o)$ $h_t = o_t * \tanh(C_t)$

Where, x_t is input vector, $h_{t-1}$is previous hidden state, C_{t-1} is previous cell state, W_f, W_i, W_o, W_c are weight matrices for gates, b_f, b_i, b_o, b_c are bias terms, σ is sigmoid activation and tanh is hyperbolic tangent activation

The architecture of the LSTM model consists of encoder, decoder, dense layer and output sequence resulting the corrected word. The model takes Nepali text input and converts characters to numerical values. The encoder LSTM processes the text sequentially from left to right, capturing patterns in the order they appear. The decoder LSTM then uses this information to generate corrected text character by character. Dense layers help to transform the LSTM outputs into final predictions, resulting in the corrected Nepali words.

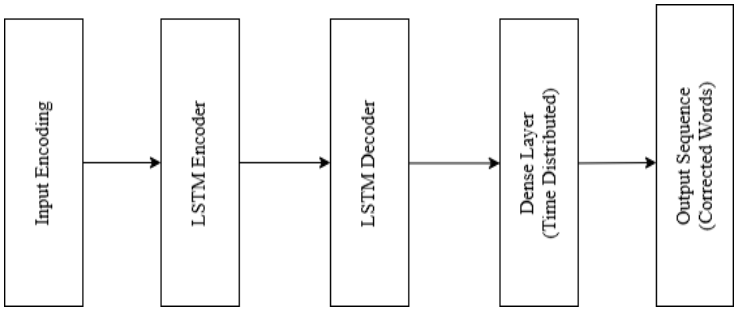

Figure 2: Spell Checking using LSTM Model

The input encoding begins with raw Nepali text characters converted into numerical representations. This is done via one-hot encoding where each character becomes a sparse binary vector. The LSTM Encoder processes the input sequence sequentially, building a hidden state that captures information from the beginning to the end of the word. LSTM Decoder generates corrected characters auto regressively. In the LSTM, the decoder receives only the final forward states from the encoder. The post-LSTM processing typically involves a Time Distributed Dense Layer consisting of 256 units with ReLU that non-linearly transforms each timestep's LSTM outputs and a final Softmax Dense Layer that predicts character probabilities. LSTM models output character probabilities at each timestep, but with key behavioral differences as the LSTM tends to be better at local corrections (fixing single-character typos) but may struggle with longer-distance dependencies in complex Nepali words.

3.4. Spell Checking using BiLSTM

Bidirectional LSTM enhance traditional LSTMs by processing input sequences in both forward and backward directions, allowing the model to capture context from past and future states simultaneously. A BiLSTM consists of two independent LSTMs:

- Forward LSTM: Processes the sequence from left to right (standard LSTM).

  $\overrightarrow{ht} = \text{LSTM}_{\text{forward}}(x_t, \overrightarrow{ht-1})$

- Backward LSTM: Processes the sequence from right to left (reverse order).

  $\overleftarrow{ht} = \text{LSTM}_{\text{backward}}(x_t, \overleftarrow{ht+1})$

- Final Output at Time *t*: $h_t = [\overrightarrow{ht} ; \overleftarrow{ht}]$

Where, $\overrightarrow{ht}$ is hidden state from forward pass, $\overleftarrow{ht}$ is hidden state from backward pass and x_t is input vector

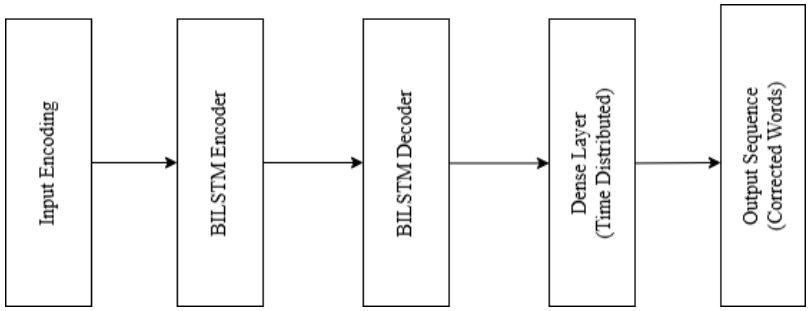

Figure 3: Spell Checking using BiLSTM Model

For BiLSTM model, the input encoding begins with raw Nepali text characters converted into numerical representations. This is done via one-hot encoding where each character becomes a sparse binary vector. The BiLSTM Encoder enhances this by processing the sequence in both forward and backward directions simultaneously. This is particularly valuable for Nepali's agglutinative morphology where prefixes and suffixes both modify word meaning. The BiLSTM decoder receives concatenated forward-backward states, providing richer contextual information. For Nepali words with common infix errors (like swapping "ो" and "ौ"), this bidirectional context helps the decoder make more informed corrections. The post-LSTM processing typically involves a TimeDistributed Dense Layer containing 256 units with ReLU that non-linearly transforms each BiLSTM outputs and a final Softmax Dense Layer that predicts character probabilities. For BiLSTM models, this layer often benefits from the richer encoder representations, requiring slightly fewer to achieve similar performance.

3.5. Performance Metrics

The performance of the LSTM model and BiLSTM model is measured using following metrics:

Character-level Accuracy: It measures the **percentage of correctly predicted characters** in a sequence compared to the ground truth**, position by position.**

$$\textbf{Char Accuracy} = \frac{Number\ of\ correctly\ predicted\ characters}{Total\ characters} \times \mathbf{100\%}$$

BLEU Score (Character-Level): It measures the **percentage of correctly predicted characters** in a sequence compared to the ground truth**, position by position.**

$$\text{BLEU Score} = \text{BP} \times \exp(\sum_{n=1}^{N} w \times \log pn)$$

Where, BP (Brevity Penalty) penalizes prediction shorter than target and *pn* is precision for n-grams.

$$\text{BP} = \begin{cases} 1 \text{ if } |\text{pred}| > |\text{target}| \\ e^{1-\frac{|target|}{|predicted|}} \text{ if } |\text{pred}| \leq |\text{target}| \end{cases}$$

$$pn = \frac{count\ of\ matching\ n-grams}{total\ n-gram\ in\ prediction}$$

Training Accuracy*:* It is the percentage of correct predictions on the training set.

$$\textbf{Training Accuracy} = \frac{Number\ of\ total\ prediction\ on\ training\ set}{Total\ training\ samples}$$

Training Loss*:* It is the average loss computed during training, showing how well the model fits the training data.

$$\textbf{Training Loss} = \frac{Sum\ of\ error\ on\ training\ set}{Total\ training\ samples}$$

Validation Accuracy: It is the percentage of predictions that exactly match the ground truth labels in the validation set.

$$\textbf{Validation Accuracy} = \frac{Number\ of\ total\ prediction\ on\ validation\ set}{Total\ validation\ samples}$$

Validation Loss*:* It measures model error on unseen validation data.

$$\textbf{Validation Loss} = \frac{Sum\ of\ error\ on\ validation\ set}{Total\ validation\ samples}$$

4. Model Implementation

The models were implemented using Python programming language with libraries TensorFlow 2.x, Keras, NumPy, Pandas, Matplotlib, and Scikit-learn. The hyperparameters for the models are set as below;

Table 10: Hyperparameter Setting for the Models

Batch Size	128
Epochs	100
Optimizer	RMSprop
Loss Function	Categorical Crossentropy
Learning Rate	0.001
Encoder Units	256
Decoder Units	256
Activation Function	ReLU (for Dense layers)
Output Activation	Softmax

5. Results and Discussions

After the successful training of the models with batch size of 128 and 100 epochs, character level accuracy and BLEU Score for both LSTM and BiLSTM model is shown below:

Table 11: Character-level Accuracy and BLEU Score of Spell Checking

	Character-level Accuracy	**BLEU Score (Character Level)**
LSTM	65.40%	0.487
BILSTM	75.12%	0.553

The BiLSTM model outperformed LSTM marking improvements of 9.72% and 13.6% percentage change on the character-level accuracy and BLEU Score respectively. This enhancement stems from BiLSTM's bidirectional processing capability, which effectively captures contextual relationships between characters by analyzing sequences in both forward and backward directions. However, both models exhibited limitations when handling non-Nepali text or special characters, as evidenced by their incorrect predictions for inputs.

The models were evaluated with accuracy and loss. Following table and graph plots demonstrates the accuracy and loss of the models;

Table 12: Accuracy and loss of LSTM and BiLSTM Models

Epoch	Training Accuracy		Validation Accuracy		Training Loss		Validation Loss	
	LSTM	BiLSTM	LSTM	BiLSTM	LSTM	BiLSTM	LSTM	BiLSTM
0	0.1738	0.1717	0.1837	0.1729	1.2572	1.2532	1.2709	1.2795
20	0.3127	0.3886	0.2941	0.3886	1.0250	0.9238	1.1143	1.0002
40	0.4218	0.5150	0.3679	0.4742	0.8984	0.7576	1.0306	0.8549
60	0.4809	0.5953	0.4231	0.5953	0.8306	0.6630	0.9615	0.8434
80	0.5309	0.6532	0.4331	0.5248	0.7741	0.5984	0.9462	0.7872
100	0.5736	0.6912	0.4604	0.5350	0.7230	0.5548	0.7230	0.7791

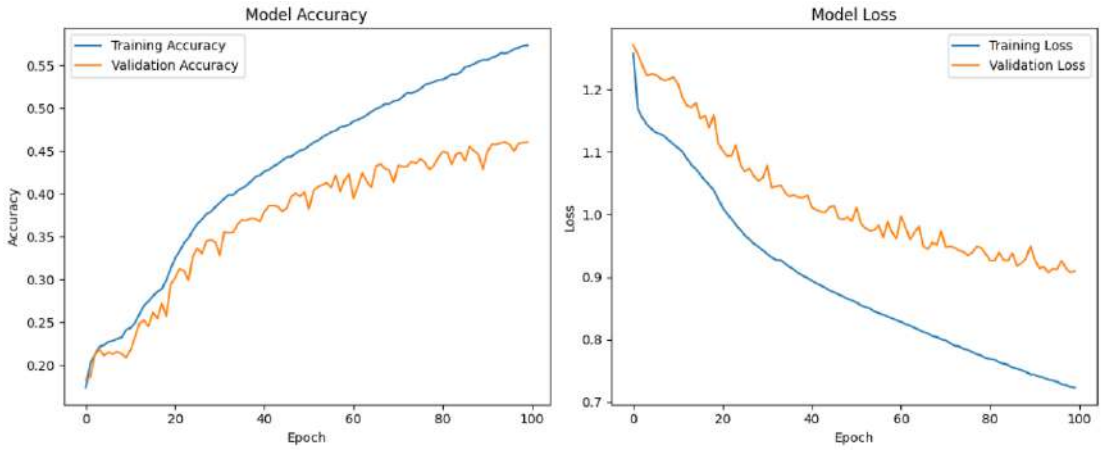

Figure 4: Graph plot of accuracy and loss of LSTM Model

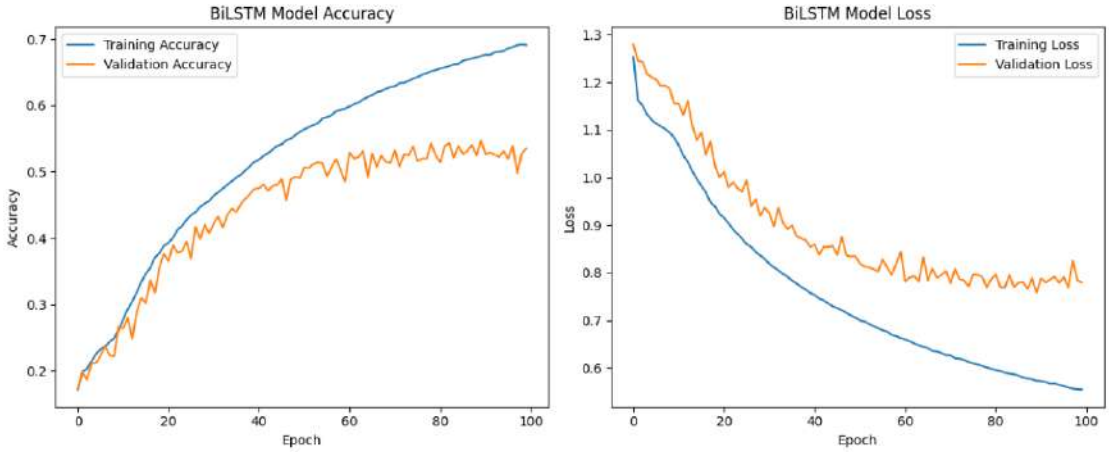

Figure 5: Graph plot of accuracy and loss of BiLSTM Model

The above graphs and findings demonstrate that the BiLSTM model outperforms the LSTM model in terms of accuracy and performance. The LSTM's training metrics further highlighted challenges in generalization, with validation accuracy plateauing at 46.04% compared to its peak training accuracy of 57.56%, suggesting over fitting. In contrast, BiLSTM's more stable performance underscores its robustness, though it demands greater computational resources.

6. Conclusion

The LSTM model demonstrates effective learning capabilities for Nepali text processing, as evidenced by the decreasing training and validation loss curves. However, the plateau in validation loss suggests the model may struggle with complex Nepali word structures. While suitable for simpler Nepali text tasks, the LSTM's unidirectional nature limits its ability to fully capture contextual relationships in longer or morphologically complex words. The BiLSTM model shows stronger promise for Nepali text processing, with training/validation losses converging more smoothly. Bidirectional architecture better handles: Prefixes/suffixes, Long-range dependencies in compound characters and Keyboard-adjacent typos via contextual awareness. For more robust performance, future work could explore attention mechanisms or transformer-based architectures, coupled with an expanded dataset encompassing dialectal variations and common misspellings, to further improve the model's accuracy and generalization across diverse Nepali text processing tasks.

References:

Bhattarai, A. (2018). *GitHub - ashmitbhattarai/Nepali-Language-Modeling-Using-LSTM: Semester Project for NLP course*. https://github.com/ashmitbhattarai/Nepali-Language-Modeling-Using-LSTM

Devkota, B., Adhikar, B., & Shrestha, D. (2016). Integrating romanized Nepali spellchecker with SMS based decision support system for Nepalese farmers. *SKIMA 2015 - 9th International Conference on Software, Knowledge, Information Management and Applications*. https://doi.org/10.1109/SKIMA.2015.7400046

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6). https://doi.org/10.1016/j.neunet.2005.06.042

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8). https://doi.org/10.1162/neco.1997.9.8.1735

Luitel, N., Bekoju, N., Sah, A., & Shakya, S. (2024). *Contextual Spelling Correction with Language Model for Low-Resource Setting*. https://doi.org/10.1109/ICICT60155.2024.10544712

Phukan, R., Neog, M., Goutom, P. J., & Baruah, N. (2024). Automated Spelling Error Detection in Assamese Texts using Deep Learning Approaches. *Procedia Computer Science*, *235*, 1684–1694. https://doi.org/10.1016/J.PROCS.2024.04.159

Prasain, B., Lamichhane, N., Pandey, N., Adhikari, P., & Mudbhari, P. (2022). Nepali Spelling Checker. *Journal of Engineering and Sciences*, *1*(1). https://doi.org/10.3126/jes2.v1i1.58461

Sharma, S., Valls-Vargas, J., King, T. H., Guerin, F., & Arora, C. (2023). Contextual Multilingual Spellchecker for User Queries. *SIGIR 2023 - Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://doi.org/10.1145/3539618.3591861

Singh, S., & Singh, S. (2021). HINDIA: a deep-learning-based model for spell-checking of Hindi language. *Neural Computing and Applications*, *33*(8). https://doi.org/10.1007/s00521-020-05207-9

Thapa, S., Pradhan, A., Kayastha, D., Lawaju, A., Rana, P. B., & Basnet, M. (2025). Prasta Nepali: A Transformer Based Approach for Automated Nepali Grammar (Byakaran) Error Detection and Correction. *2025 International Conference on Inventive Computation Technologies (ICICT)*, 1327–1334. https://doi.org/10.1109/ICICT64420.2025.11004703