Institute of Technology of Cambodia
Department of Applied Mathematics and Statistics

LSTM-based Khmer Text Style Transfer Using Representation Learning

Subject: Natural Language Processing (NLP)

Name of Students	ID
PEL Bunkhloem	e20201314
PHETH Soriyuon	e20210674
PHOEUN Rajame	e20211748
PHOEURN Kimhor	e20210823
PHORN Sreypov	e20210166
YIN Sambat	e20210138

Lecturers: **Dr. KHON Vanny (Course)**
Mr. TOUCH Sopheak (TP)

Academic Year 2025 - 2026

Table of Contents

1 Problem Statement

The task is formulated as a **sequence-to-sequence style transfer problem**, where an input sentence in normal Khmer is transformed into its corresponding royal-style sentence while preserving semantic content.

Let:

- $X = (x_1, x_2, \dots, x_T)$ denote the input sequence in normal Khmer
- $Y = (y_1, y_2, \dots, y_{T'})$ denote the output sequence in Royal Khmer
- f_θ denote the encoder–decoder LSTM model parameterized by θ

The conditional probability of generating the target sequence given the input is modeled as:

$$P(Y \mid X; \theta) = \prod_{t=1}^{T'} P(y_t \mid y_1, \dots, y_{t-1}, X; \theta)$$

The optimal parameters θ^* are obtained by maximizing the log-likelihood over the labeled dataset $\mathcal{D}$:

$$\theta^* = \arg\max_\theta \sum_{(X,Y)\in\mathcal{D}} \log P(Y \mid X; \theta)$$

This formulation treats Khmer style transfer as a **representation learning problem**, where the encoder learns latent representations of the input sentence that can be decoded into the royal linguistic register.

1.1 Explanation of Terms and Notation

- X: Input sequence of tokens representing normal Khmer text
- Y: Target sequence of tokens representing Royal Khmer text
- x_t: The t-th token in the input sequence
- y_t: The t-th token in the output sequence
- T, T': Lengths of the input and output sequences

- $\mathcal{D}$: Manually labeled dataset of paired normal–royal Khmer sentences
- θ: Trainable parameters of the encoder–decoder LSTM model
- f_θ: Sequence-to-sequence function learned by the model
- h_t: Encoder hidden state representing contextual information at time step t

2 Pre-training

2.1 Data Description

Data is scraped from:

- Sources: Diverse online Khmer articles (news, blogs, social media, wikipedia).
- Technique: Selenium.

2.2 Data Cleaning and Character-level Processing

To prepare the text for character-level modeling, the following preprocessing steps were applied:

1. Removing unwanted characters: Zero-width spaces (`\u200b`, `\u200c`, `\u200d`), formatting marks (`\ufeff`), and specific Khmer punctuation (។, ៗ, ៖) were removed.
2. Removing Latin characters and digits: Ensures the model focuses purely on Khmer script.
3. Character normalization: Certain Khmer characters were mapped to reduce redundancy and noise (e.g., ឝ ឝ, ឞ ម).
4. Whitespace trimming: Extra spaces were removed to produce clean character sequences.

2.3 Building Vocabulary

The model operates at the character level, treating each Khmer character as an individual token. - Vocabulary mapping: Characters are converted into numerical indices using a dictionary (stoi). Example: {' ': 0,'ក': 1,'ខ': 2, ...}

- Special tokens are added to support sequence modeling:
  - `<sos>`: Start of sequence
  - `<eos>`: End of sequence

  – `<pad>`: Padding
  – `<unk>`: Unknown characters

This step converts all text sequences into numerical sequences suitable for PyTorch training.

2.4 Padding and Batch Preparation

To enable efficient batch processing:

- Sequences within a batch are padded to the same length using the token.
- PyTorch's pad_sequence function ensures that all input-target pairs in a batch are aligned.

2.5 Architecture Overview

The project adapts a character-level Long Short-Term Memory (LSTM) autoencoder for unsupervised pre-training on Khmer text. The architecture implements an encoder-decoder (Seq2Seq) framework where the encoder compresses input sequences into fixed-dimensional context vectors and the decoder reconstructs original sequences from these compressed representations. An embedding layer handles the mapping of discrete character indices to continuous vector representations.

2.5.1 Encoder-Decoder Architecture

Encoder Function

The bidirectional LSTM encoder processes embedded character sequences to produce condensed representations. Formally:

$$(h_T^{\text{enc}}, C_T^{\text{enc}}) = f_{\text{enc}}(e_1, e_2, \ldots, e_T)$$

where $h_T^{\text{enc}}, C_T^{\text{enc}} \in \mathbb{R}^{256}$ encapsulate the complete input sequence information.

Decoder Function

The decoder LSTM, initialized with encoder states, generates reconstructed sequences:

$$(h_t^{\text{dec}}, C_t^{\text{dec}}) = f_{\text{dec}}(e_{t-1}, h_{t-1}^{\text{dec}}, C_{t-1}^{\text{dec}})$$

with initial conditions $(h_0^{\text{dec}}, C_0^{\text{dec}}) = (h_T^{\text{enc}}, C_T^{\text{enc}})$. Teacher forcing utilizes ground truth previous characters during training to stabilize learning.

2.5.2 Mathematical Formulation of LSTM Architecture

The LSTM architecture addresses the vanishing gradient problem in the traditional recurrent neural networks (RNN) through gating mechanisms that regulate information flow. At each timestep t, the LSTM cell computes:

Gating Functions

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget Gate)} \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input Gate)} \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output Gate)} \end{aligned}$$

Memory Cell Operations

$$\begin{aligned} \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Candidate State)} \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t && \text{(Cell State Update)} \\ h_t &= o_t \odot \tanh(C_t) && \text{(Hidden State Output)} \end{aligned}$$

Here, σ denotes the sigmoid activation function constraining outputs to $[0, 1]$, tanh produces outputs in $[-1, 1]$, and $\odot$ represents element-wise multiplication. The forget gate f_t modulates retention of previous cell state C_{t-1}, while the input gate i_t regulates incorporation of new candidate values $\tilde{C}_t$. The output gate o_t controls exposure of the cell state to subsequent layers.

3 Fine-Tuning

3.1 Dataset: Normal to Royal Transfer

3.1.1 Parallel Corpus

- **Source:** Scraped from Wikipedia, Fresh News, and various Khmer articles.
- **Volume:** 793 high-quality cleaned sentence pairs.
- **Format:**
  - **Input:** Normal Khmer (សាមញ្ញ)
  - **Target:** Royal Khmer (រាជស័ព្ទ)

3.1.2 Data Structure Example

normal	royal
លោកបានដើរកាត់តាមឆ្នេរសមុទ្រ	ព្រះអង្គស្ដេចយាងកាត់តាមឆ្នេរសមុទ្រ

3.1.3 Preprocessing

Text Cleaning

- **Special Chars:** Stripped `\u200b` (ZWSP), `\u200c` (ZWNJ).
- **Punctuation:** Removed traditional markers (៙, ៚, ៖, ។).
- **Script Filter:** Used Regex to retain only Khmer range (`\u1780-\u17FF`).
- **Normalization:** Whitespace trimming and collapsing.

Tokenization & Splitting

- **Level:** Character-level mapping (`stoi`).
- **Wrapping:** Added `<sos>` and `<eos>` to every sequence.
- **Splitting Strategy:**
  - **Train:** 80%
  - **Validation:** 10%
  - **Test:** 10%

3.1.4 Character Numericalization

To process the script, characters are mapped to indices:

Token	Index	Category
	0	Whitespace
ក	1	Consonant
ខ	2	Consonant
`<sos>`	80	Start of Sequence
`<pad>`	0	Padding

Padding Logic: All sequences in a batch are aligned to the length of the longest sentence using the `<pad>` token to ensure matrix compatibility during PyTorch training.

3.2 Adaptation of Model Architecture

The original character-level autoencoder is adapted into an **attention-based Seq2Seq architecture** tailored for Normal-to-Royal Khmer style transfer.

3.2.1 Encoder Reuse

- The encoder is an LSTM trained on a large Khmer corpus.
- It captures fundamental grammatical and structural properties of the language.
- Encoder weights are reused and initially frozen to prevent catastrophic forgetting.

3.2.2 Decoder and Style Adaptation

- A new LSTM decoder is initialized with a separate embedding layer.
- The decoder is initialized using the final hidden and cell states of the encoder.
- This initialization provides a compact semantic representation of the Normal Khmer input.

3.2.3 Attention Mechanism

An attention layer is incorporated to handle:

- Long-range dependencies
- Complex mappings where a single Normal word may correspond to multiple Royal terms

The attention mechanism allows the decoder to dynamically focus on relevant parts of the source sentence at each decoding step.

3.2.4 Teacher Forcing

A teacher forcing ratio of **0.5** is used during training. At each decoding step:

- 50% of the time, the decoder uses the ground-truth character.
- 50% of the time, it uses its own previous prediction.

3.3 Mathematical Description of the Model

3.3.1 Encoder Representation

Given a Normal Khmer input sequence:

$$x = (x_1, x_2, \dots, x_n)$$

the encoder produces hidden states:

$$h_t^{enc} = \text{LSTM}_{enc}(E(x_t), h_{t-1}^{enc})$$

where $E(x_t)$ is the embedding of the t-th character.

3.3.2 Attention Mechanism

For each decoding step t, attention weights are computed as follows.

Alignment score:

$$e_{t,i} = \text{score}(h_{t-1}^{dec}, h_i^{enc})$$

Attention weights:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{n} \exp(e_{t,k})}$$

Context vector:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i^{enc}$$

3.3.3 Decoder and Output Prediction

The decoder updates its hidden state using the previous output and the context vector:

$$h_t^{dec} = \text{LSTM}_{dec}([E(y_{t-1}); c_t], h_{t-1}^{dec})$$

The probability distribution over the output vocabulary is:

$$P(y_t \mid y_{<t}, x) = \text{Softmax}(W(h_t^{dec} + c_t) + b)$$

where W and b are learnable parameters.

3.3.4 Loss Function

The model is trained using **Cross-Entropy Loss**:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t^* \mid y_{<t}^*, x)$$

4 Experimental Steps

4.1 Pre-training Phase

Two separate Seq2Seq models were trained at the character level using different text sources.

- **Model A (General text):** trained on modern Khmer sentences from news and public text sources.

- **Model B (Folktale text):** trained on classical Khmer folktales.

Parameter	Value
Hidden Size	256
Layers	2
Batch Size	32
Learning Rate	0.001
Optimizer	Adam
Epochs	50
Loss Function	Cross-entropy loss

The general-text pre-trained model demonstrated significantly higher predictive accuracy, indicating a more comprehensive understanding of modern Khmer syntax and vocabulary.

4.2 Fine-tuning Phase

The general-text pre-trained model was fine-tuned on a Khmer style transfer dataset.
Fine-tuning hyperparameters:

Parameter	Value
Learning Rate	0.001
Batch Size	16

Parameter	Value
Epochs	100
Optimizer	Adam
Loss Function	Cross-entropy loss
Special Mechanism	Attention

5 Results

5.1 Quantitative Results

After pre-training, each model's pre-training BLEU score was computed on a held-out test set.

Model	Pre-training BLEU on Test (%)
General Text	**30.1**
Folktale Text	**9.4**

After fine-tuning the **general text** pre-trained model, we test it on the test set. Average BLEU: 40%, here are some examples:

Generated Output	**Actual Output**	**BLEU Score**
ព្រះមហាក្សត្រប្រទានព្រះរាជបន្ទូលថា បាន	ព្រះមហាក្សត្រប្រទានព្រះរាជទ្រព្យជួយរាស្	0.79
ព្រះនាងមិនឱ្យមានព្រះរាជបុត្រពីព្រះនាងមា	ព្រះនាងមិនឱ្យភិលៀងធ្វើព្រះរាជកិច្ចជំនួស	0.73

5.2 Discussion

The experimental results show that the choice of pre-training corpus directly affects fine-tuning performance in Khmer text style transfer.

Pre-training on general modern Khmer text exposed the model to a richer vocabulary, more varied sentence structures, and diverse linguistic patterns. This led to stronger encoder representations that transferred well to the style transfer task. In contrast, pre-training on folktales, while grammatically valid, introduced stylistic biases. Folktale language relies heavily on archaic expressions and repetitive narrative formulas, which reduced the model's ability to generalize to modern royal style constructions. This explains the significantly lower BLEU scores observed with folktale based pre training.

Beyond pre training, several factors limit the model's performance on the fine tuning task itself.

First, the parallel dataset is very small. We have only 793 sentence pairs. This is insufficient for a neural sequence to sequence model to learn the full range of mappings between normal and royal Khmer. As a result, the model depends heavily on frequent patterns in the training data. For example, when the input describes someone speaking or making a statement, the model often defaults to the phrase ព្រះរាជបន្ទូលថា (royally declared that...), even when the correct royal rendering should involve giving aid, traveling, or performing a ritual. Without enough examples of diverse royal usages, the model reuses the most common template it has seen.

Second, the model operates at the character level, generating one Khmer character at a time. While this avoids the challenges of word segmentation in Khmer, it also makes the task more fragile. A single early error, such as choosing the wrong consonant, can steer the entire output toward an incorrect word. For instance, once the model begins generating បន្ទូល instead of ទ្រព្យ, it is locked into that lexical path and cannot recover. This explains why many outputs start with correct royal markers but end in incomplete, malformed, or semantically incorrect phrases.

Third, the use of teacher forcing during training creates a mismatch between training and inference. During training, the decoder sometimes receives the ground truth previous character, but at test time it must rely entirely on its own predictions. If an early prediction is wrong, the error propagates through the rest of the sequence. In some cases, this leads to hallucinated content, such as generating ព្រះរាជបុត្រ (royal child) when the original sentence mentions no such person, simply because the model followed a familiar but inappropriate royal pattern.

In short, the model succeeds at applying surface level royal markers like ព្រះអង្គ or ទ្រង់, but it often fails to preserve the original meaning in a way that is both accurate and culturally appropriate. This limitation is not due to a flaw in the model architecture, but rather to the inherent difficulty of the task given the limited data and the deep linguistic differences between normal and royal Khmer registers.