Asian Institute of Technology (AIT)

Artificial Intelligence: Natural Language Understanding (NLU)

Assignment 3 (A3)

Submitted To:

Dr. Chaklam Silpasuwanchai

Assistant Professor

Computer Science and Information Management

Asian Institute of Technology

chaklam@ait.asia

Submitted By:

Tisa Bajracharya

st126686

GitHub Link: https://github.com/bajratisa/NLP_assignment

English to Nepali Neural Machine Translation

This assignment focuses on building a neural machine translation system that translates English text into Nepali. By implementing sequence-to-sequence (Seq2Seq) models with different attention mechanisms from scratch, we explore how neural networks learn to align and translate languages with different grammatical structures (SVO vs. SOV).

1. Scope of Assignment

The goal of this assignment is to create an effective English-to-Nepali translation system. Unlike rule-based translation, this system uses deep learning to learn translation patterns from data. It includes training custom Seq2Seq models, experimenting with General vs. Additive Attention mechanisms to handle long sentences, and deploying a user-friendly web interface for real-time translation.

2. Training Data

We used the sharad461/ne-en-parallel-208k dataset for training. This is a collection of approximately 208,000 parallel English-Nepali sentence pairs, hosted on Hugging Face, suitable for supervised neural machine translation.

- **Citation:** Sharad461. (n.d.). *ne-en-parallel-208k*. Hugging Face Datasets. https://huggingface.co/datasets/sharad461/ne-en-parallel-208k
- **Preprocessing:** We cleaned the text by lowercasing English sentences and tokenizing both languages. We built custom vocabularies to convert words into unique integer IDs and added special tokens (<sos>, <eos>, <pad>, <unk>) to manage sequence boundaries and unknown words.

3. Workflow

The assignment follows a step-by-step path from raw text to a working web application:

1. **Preprocessing:** We loaded the dataset, tokenized the English and Nepali text, and built vocabularies to convert text into numerical tensors usable by PyTorch.
2. **Model Implementation:** We implemented a Sequence-to-Sequence (Seq2Seq) architecture from scratch. This included an Encoder (GRU), a Decoder (GRU), and two distinct Attention mechanisms:
   - **General Attention:** Calculates alignment using a dot product between decoder and encoder states.
   - **Additive Attention:** Uses a learned feed-forward neural network to calculate alignment scores.

3. **Training:** We trained both models side-by-side using the Cross Entropy Loss function and the Adam optimizer, utilizing "Teacher Forcing" to stabilize training.
4. **Evaluation:** We evaluated the models by comparing Training and Validation Loss/Perplexity (PPL) and visualizing Attention Heatmaps to interpret how the models aligned words between languages.
5. **Application:** Finally, we built a web app using Dash that takes a user's English input, translates it to Nepali using the best-performing model (Additive), and displays an attention heatmap.

4. Libraries Used

- **PyTorch:** The primary engine used to build, train, and execute the neural networks (Encoder, Decoder, Attention) from scratch.
- **Dash & Plotly:** Used to build the interactive web interface and visualize the Attention Heatmaps dynamically.
- **Datasets (Hugging Face):** Used to easily download and manage the large parallel corpus.
- **Seaborn/Matplotlib:** Used during the training phase to plot learning curves (Loss/Perplexity) and static attention maps.
- **NumPy:** Used for handling numerical operations and processing attention matrices.

5. Models Used: Pros & Cons

Model	**Advantages**	**Disadvantages**
General Attention (Luong)	**Computationally Efficient:** Uses simple dot products, making it faster to train and run inference.	**Less Expressive:** Can struggle with complex alignment relationships compared to non-linear methods.
Additive Attention (Bahdanau)	**Higher Accuracy:** Uses a learned neural network to calculate alignment, handling complex word-order changes (like English SVO to Nepali SOV) better.	**Slower Training:** More parameters and computations per step make it slower to train than dot-product attention.

6. Conclusion

This assignment demonstrates the power of Attention mechanisms in Neural Machine Translation. By testing custom General and Additive attention models, we observed that Additive Attention generally provided better convergence and handled the structural differences between English and Nepali more effectively. The final web application successfully interfaces with the trained model, providing real-time translations and offering interpretability through attention heatmaps, fulfilling the goal of "making our own" translation system.