Seq2Seq Language Translator

Neural Machine Translation using Sequence to Sequence Architecture

Mehdi Ait Abdelouahab
Institut National des Postes et Télécommunications

May 4, 2025

Abstract

This documentation provides a comprehensive overview of the LSTM-Language-Translator project, which implements a neural machine translation system using Long Short-Term Memory (LSTM) networks in a sequence-to-sequence architecture. The system is designed to translate text between different language pairs using parallel sentences from the ready_df.csv dataset. This document covers the theoretical background, system architecture, implementation details, training process, evaluation metrics, and usage instructions for the translation model.

Contents

1 Introduction

1.1 Project Overview

The LSTM-Language-Translator project is a neural machine translation (NMT) system designed to translate between English and Darija (Moroccan Arabic). Using Long Short-Term Memory (LSTM) networks in a sequence-to-sequence architecture, the system captures the complex linguistic patterns and relationships needed for accurate translation between these two languages.

Neural machine translation represents a significant improvement over traditional statistical methods by learning translations from parallel sentences without requiring explicit linguistic rules. This project demonstrates how recurrent neural networks can effectively model the sequential nature of language data, particularly for a language pair with significant structural differences.

1.2 Objectives

The main objectives of this project are:

- To implement an encoder-decoder LSTM architecture with attention for translating between English and Darija
- To process and utilize parallel language data from the ready_df.csv dataset
- To train a model capable of bidirectional translation between English and Darija
- To evaluate the translation quality using appropriate metrics
- To provide a usable interface for generating translations

1.3 Key Features

- Sequence-to-sequence LSTM-based architecture with attention mechanism
- Preprocessing pipeline for text normalization and tokenization
- Support for bidirectional translation (English $\rightarrow$ Darija and Darija $\rightarrow$ English)
- Word embedding representation of vocabularies

- Training with teacher forcing for improved convergence
- Inference module for generating translations

1.4 Dataset Description

The project utilizes a parallel corpus of English-Darija sentence pairs stored in a CSV file called ready_df.csv. This dataset was created through a careful data cleaning process from the Hugging Face Atlaset dataset, which contains a variety of multilingual and multimodal data with Darija content.

The dataset includes sentence pairs across various domains, including:

- Moroccan history and cultural information
- Educational content
- Casual conversation
- Technical information
- Tourism-related content

This diversity helps the model learn translation across different contexts and domains.

2 Theoretical Background

2.1 Neural Machine Translation

Neural Machine Translation (NMT) is an approach to machine translation that uses neural networks to predict the likelihood of a sequence of words. Unlike traditional phrase-based translation systems that require separate components for language modeling and translation rules, NMT systems learn to translate directly from source to target language examples.

The fundamental advantage of NMT is its ability to learn continuous representations of words and sentences, capturing deeper linguistic features and relationships than discrete, symbolic approaches.

2.2 Sequence-to-Sequence Architecture

The sequence-to-sequence (Seq2Seq) model is a type of model architecture commonly used in NMT. It consists of two main components:

- **Encoder**: Processes the input sequence and compresses the information into a context vector (or a sequence of vectors)
- **Decoder**: Takes the context vector and generates the output sequence

This architecture is particularly well-suited for problems where input and output sequences can have different lengths, as is often the case in translation.

2.3 Long Short-Term Memory (LSTM) Networks

LSTMs are a special kind of recurrent neural network (RNN) designed to address the vanishing gradient problem that can affect standard RNNs when learning long-term dependencies. LSTMs include memory cells that can maintain information for long periods, as well as gates that control the flow of information:

- **Forget gate**: Decides what information to discard from the cell state
- **Input gate**: Updates the cell state with new information
- **Output gate**: Controls what information to output based on the cell state

3 System Architecture

3.1 High-Level Architecture

The LSTM-Language-Translator uses a encoder-decoder architecture with the following major components:

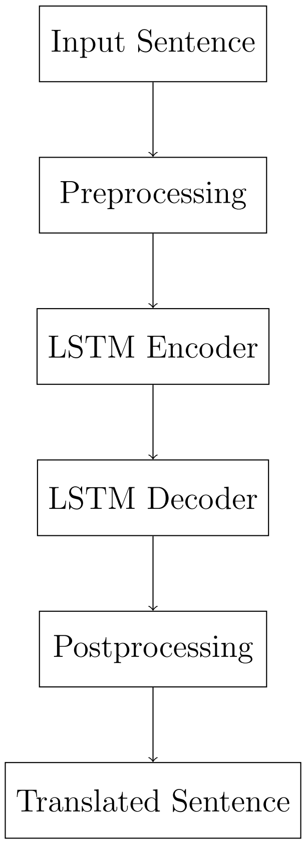

Figure 1: High-level architecture of the LSTM-Language-Translator system

3.2 Encoder Architecture

The encoder processes the input sentence to create a fixed-dimension context vector that represents the meaning of the source sentence. It consists of:

- An embedding layer that converts word indices to dense vectors
- A multi-layer LSTM network that processes the embedded sequence
- Dropout layers to prevent overfitting

The encoder generates a sequence of hidden states that represent the input sentence at different positions, which are later used by the attention mechanism to focus on relevant parts of the input.

3.3 Attention Mechanism

A critical component of the translation system is the attention mechanism, which allows the decoder to focus on different parts of the source sentence when generating each word in the translation. This addresses the limitation of encoding an entire sentence into a fixed-length vector.

The attention mechanism works as follows:

1. For each decoding step t, calculate attention scores between the current decoder hidden state s_t and all encoder hidden states $h_1, h_2, ..., h_n$
2. Normalize these scores using softmax to get attention weights $\alpha_{t,i}$
3. Compute the context vector c_t as a weighted sum of encoder hidden states: $c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$
4. Combine the context vector with the decoder state to make the next word prediction

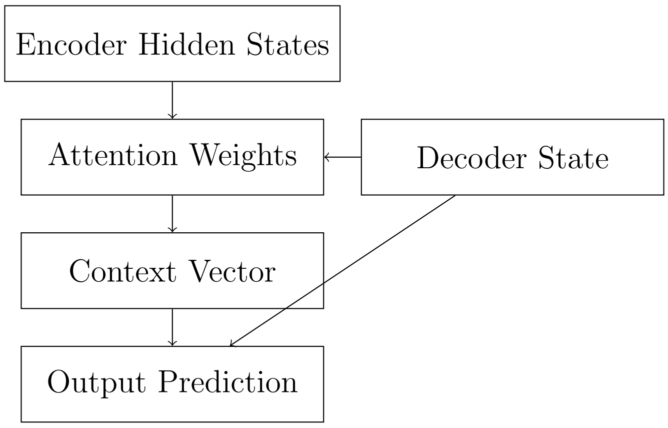

Figure 2: Attention mechanism in the translation model

Mathematically, the attention weights can be calculated as:

$$e_{t,i} = score(s_{t-1}, h_i) \tag{1}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{n} \exp(e_{t,j})} \tag{2}$$

Where *score* can be implemented in several ways:

- Dot product: $score(s, h) = s^T h$
- General: $score(s, h) = s^T W h$
- Concatenation: $score(s, h) = v^T \tanh(W[s; h])$

This project implements the attention mechanism to significantly improve translation quality, especially for longer sentences.

3.4 Decoder Architecture

The decoder generates the target translation word by word, using:

- An embedding layer for target language words
- A multi-layer LSTM network initialized with the encoder's final state
- An attention layer that computes context vectors based on encoder outputs
- A dense output layer with softmax activation to predict the probability distribution over target vocabulary

During training, the decoder uses teacher forcing, where the ground truth target word from the previous time step is fed as input to predict the next word. During inference, the decoder uses its own previous predictions as input for subsequent steps.

4 Data Processing

4.1 Dataset Description

The project uses the ready_df.csv dataset, which contains parallel sentences in source and target languages. Each row in the dataset represents a translation pair, with sentences aligned between languages.

4.2 Data Preprocessing Pipeline

The preprocessing pipeline includes the following steps:

Algorithm 1 Data Preprocessing Pipeline

```
 1: Input: raw parallel sentences from ready_df.csv
 2: Output: processed data ready for model training
 3: procedure TextNormalization(text)
 4:     Convert text to lowercase
 5:     Remove special characters and extra whitespace
 6:     Normalize punctuation return normalized text
 7: end procedure
 8: procedure Tokenization(normalized_text)
 9:     Split text into individual tokens (words/subwords) return tokens
10: end procedure
11: procedure VocabularyCreation(all_tokens)
12:     Count frequency of each token
13:     Create vocabulary with tokens exceeding minimum frequency
14:     Add special tokens: PAD, UNK, SOS, EOS
15:     Assign unique index to each token in vocabulary return vocabulary
    lookup tables
16: end procedure
17: procedure Sequencing(tokenized_sentences, vocabulary)
18:     Convert tokens to indices using vocabulary
19:     Add start and end sequence tokens
20:     Pad sequences to uniform length return indexed sequences
21: end procedure
```

4.3 Vocabulary Building

The vocabulary is constructed from the training data, with these key considerations:

- Words are sorted by frequency
- A frequency threshold is applied to limit vocabulary size
- Special tokens are added: PAD for padding, UNK for unknown words, SOS for start of sequence, and EOS for end of sequence
- Each word is assigned a unique integer index

5 Model Implementation

5.1 Key Hyperparameters

The model's performance is influenced by several hyperparameters:

Hyperparameter	**Typical Value**
Embedding dimension	256
LSTM hidden dimension	512
Number of LSTM layers	2
Dropout rate	0.2
Batch size	64
Learning rate	0.001

Table 1: Key hyperparameters of the LSTM-Language-Translator model

5.2 Loss Function and Optimization

The model is trained using:

- **Loss Function**: Categorical cross-entropy to measure the difference between predicted and actual word distributions
- **Optimizer**: Adam optimizer with a customizable learning rate and gradient clipping to prevent exploding gradients

6 Training Process

6.1 Training Methodology

The training process follows these steps:

1. Split the dataset into training, validation, and test sets
2. Prepare input-output pairs for the sequence-to-sequence model
3. Initialize model weights randomly
4. For each epoch:
    - Process mini-batches of training data
    - Apply teacher forcing (use ground truth target words as input)
    - Calculate loss and update weights through backpropagation
    - Evaluate performance on validation set
    - Save model checkpoints
5. Select the best model based on validation performance

6.2 Teacher Forcing

Teacher forcing is a technique used during training where the ground truth target word from the previous time step is fed as input to predict the next word, rather than using the model's own predictions. This approach helps stabilize training and allows for faster convergence.

6.3 Handling Variable-Length Sequences

The model handles variable-length input sequences through:

- Padding shorter sequences with a special PAD token
- Using mask layers to ignore padded positions during loss calculation
- Implementing dynamic unrolling of the LSTM cells during inference

7 Inference and Translation Generation

7.1 Translation Process

The translation process follows these steps:

Algorithm 2 Translation Inference Process

```
1: Input: Source sentence
2: Output: Translated sentence
3: procedure TRANSLATE(source_sentence)
4:     preprocessed_input ← Preprocess(source_sentence)
5:     encoder_input ← ConvertToIndices(preprocessed_input)
6:     states ← EncoderModel(encoder_input)
7:     target_seq ← CreateEmptyTargetSequence()
8:     target_seq[0] ← SOS_token
9:     translated_sentence ← [ ]
10:    stop_condition ← False
11:    while not stop_condition do
12:        output_tokens, h, c ← DecoderModel(target_seq, states)
13:        sampled_token_index ← argmax(output_tokens[0, 0, :])
14:        sampled_token ← IndexToWord(sampled_token_index)
15:        if sampled_token = EOS_token or length(translated_sentence)
   >max_length then
16:            stop_condition ← True
17:        else
18:            Add sampled_token to translated_sentence
19:            target_seq ← sampled_token_index
20:            states ← [h, c]
21:        end if
22:    end while
        return JoinTokens(translated_sentence)
23: end procedure
```

7.2 Handling Unknown Words

Unknown words (those not present in the training vocabulary) are handled using:

- Replacement with an UNK token during preprocessing
- Potential post-processing strategies, such as copying source unknown words to the target or using subword tokenization approaches

8 Evaluation and Performance Metrics

8.1 Evaluation Metrics

The model's translation quality is primarily evaluated using BLEU (Bilingual Evaluation Understudy) score, which is the standard metric for machine translation systems. BLEU measures the precision of n-grams between generated translations and reference translations, with a penalty for translations that are too short.

The BLEU score ranges from 0 to 1 (or 0 to 100 when expressed as a percentage), where higher scores indicate better translation quality.

The ready_df.csv dataset provides parallel sentences that serve as ground truth references for calculating the BLEU score during model evaluation.

8.2 Performance Analysis

The performance of the model depends on various factors:

- Size and quality of the training dataset
- Complexity of the language pair
- Model hyperparameters
- Training duration

9 Limitations and Future Work

9.1 Current Limitations

The current implementation has several limitations:

- Limited handling of long sentences due to LSTM memory constraints
- Difficulty with rare words and out-of-vocabulary terms
- Sequential nature of LSTMs limiting parallelization during training
- Single-head attention mechanism may not capture all relevant contextual information
- Performance degradation for language pairs with significant word order differences

9.2 Potential Improvements

Future improvements could include:

- Implementing multi-head attention mechanisms
- Incorporating transformer-based architectures for better parallelization
- Using subword tokenization techniques like BPE or SentencePiece
- Implementing beam search for better translation generation
- Exploring transfer learning approaches with pre-trained language models
- Adding a coverage mechanism to avoid word repetition or omission
- Implementing a copy mechanism for handling proper nouns and rare terms

10 Conclusion

The LSTM-Language-Translator project demonstrates the application of deep learning techniques to the challenging task of machine translation. By leveraging LSTM networks in a sequence-to-sequence architecture, the system can learn to translate between language pairs directly from parallel examples, without the need for explicit linguistic rules.

The documentation provides a comprehensive overview of the theoretical foundations, implementation details, and practical usage of the translation system. While the current approach has certain limitations, it establishes a solid foundation for neural machine translation that can be extended and improved through future work.

11 References

References

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv preprint arXiv:1409.0473.

[2] Luong, M. T., Pham, H., & Manning, C. D. (2015). *Effective approaches to attention-based neural machine translation.* arXiv preprint arXiv:1508.04025.

A Appendix B: Dataset Statistics

Statistic	**Value**
Number of sentence pairs	555,172
Darija token count range	2-1566
English token count range	2-1566
Darija vocabulary size (est.)	30,000-50,000
English vocabulary size (est.)	25,000-45,000
Dataset sources	19+ different sources

Table 2: Dataset statistics for ready_df.csv

B Appendix C: Dataset Sources

Source	**Content Type**
medmac01/moroccan_history_qa	Moroccan history Q&A pairs
learnmoroccan.com	Language learning resources
loecsen.com	Language learning resources
NQ-Open-AYA	General knowledge Q&A
common-crawl	Web text data
auto-math	Mathematical content in Darija
stanford	Educational content
translation-rows	Translated content
wikihow	How-to guides
atlasia/darija_english/stories	Short stories
SoftAge-AI/sft-conversational_dataset	Conversational dialogs
aboutaleb/moroccantourism	Tourism information
flan_v2	General language data
cot	Chain-of-thought reasoning
oasst1	Assistant conversations
lima	Technical content
gpt4_alpaca	Generated content
sharegpt	Dialog data
wizardlm	Educational/instructional content

Table 3: Sources of data in the ready_df.csv dataset