Neural Machine Translation (English→French) Report

Saksham Madan
IIT BHU, Mathematics & Computing

20 June 2025

Abstract

This report presents an end-to-end Neural Machine Translation pipeline translating English to French. I developed three models: a vanilla Seq2Seq LSTM, an attention-augmented Seq2Seq, and a Transformer fine-tuned from a pretrained checkpoint. It also includes mathematical formulations, key code excerpts, training strategies, challenges and solutions, evaluation metrics (BLEU), comparative tables, and interpretability via attention visualization all in detailed.

1 Introduction

Machine Translation (MT) seeks a function $f : \mathcal{X} \to \mathcal{Y}$ mapping source sentence $x = (x_1, \dots, x_{T_x})$ to target sentence $y = (y_1, \dots, y_{T_y})$. Neural MT uses encoder-decoder architectures with sequence modeling via LSTMs or Transformers.

2 Dataset Preparation

2.1 Source Data

I used the ManyThings English–French dataset. After download and extraction:

Listing 1: Loading and filtering pairs

```
with open('fra.txt','r',encoding='utf-8') as f:
    lines = f.read().splitlines()
pairs = [line.split('\t')[:2] for line in lines if '\t' in line]
random.shuffle(pairs)
pairs = pairs[:200000]
```

2.2 Tokenization and Padding

Define tokenizers and sequence conversion:

Listing 2: Tokenization and padding

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```python
eng_tokenizer = Tokenizer(filters='', lower=True, oov_token='<unk>')
fra_tokenizer = Tokenizer(filters='', lower=True, oov_token='<unk>')

eng_tokenizer.fit_on_texts(eng_sentences)
fra_tokenizer.fit_on_texts(fra_sentences_input + fra_sentences_target) # full
    vocab

eng_sequences = eng_tokenizer.texts_to_sequences(eng_sentences)
fra_input_sequences = fra_tokenizer.texts_to_sequences(fra_sentences_input)
fra_target_sequences = fra_tokenizer.texts_to_sequences(fra_sentences_target)

max_len_eng = max(len(seq) for seq in eng_sequences)
max_len_fra = max(max(len(seq) for seq in fra_input_sequences),
                  max(len(seq) for seq in fra_target_sequences))

encoder_input_data = pad_sequences(eng_sequences, maxlen=max_len_eng, padding=
    'post')
decoder_input_data = pad_sequences(fra_input_sequences, maxlen=max_len_fra,
    padding='post')
decoder_target_data = pad_sequences(fra_target_sequences, maxlen=max_len_fra,
    padding='post')
decoder_target_data = decoder_target_data.reshape(*decoder_target_data.shape,
    1)
```

3 Summary of Models

Model	Layers	Epochs	BLEU Score
Vanilla Seq2Seq (LSTM)	2x BiLSTM + LSTM decoder	30	28.80
Seq2Seq + Attention	1x BiLSTM + AdditiveAttention + LSTM	30	37.12
Transformer (HF fine-tune)	6x Transformer layers	3	56.80

Table 1: Comparison of model architectures, training epochs, and BLEU scores.

4 Part 1: Vanilla Seq2Seq LSTM

4.1 Model Formulation

The vanilla Seq2Seq model uses an encoder LSTM and decoder LSTM:

$$h_t = \text{LSTM}(x_t, h_{t-1}, c_{t-1}), \tag{1}$$

$$s_t = \text{LSTM}(y_{t-1}, s_{t-1}, c'_{t-1}), \tag{2}$$

$$P(y_t|y_{<t}, x) = \text{softmax}(W_o s_t + b_o). \tag{3}$$

Encoder with bidirectional stacking:

Listing 3: Building vanilla Seq2Seq model

```
EMBEDDING_DIM = 100
HIDDEN_UNITS = 256
STACKED_LAYERS = 2
BATCH_SIZE = 256
EPOCHS = 30
LEARNING_RATE = 0.001

# Encoder
enc_in = Input((None,))
emb_enc = Embedding(eng_vocab_size, EMBEDDING_DIM, weights=[
    en_embedding_matrix], trainable=True)(enc_in)
for _ in range(STACKED_LAYERS-1):
    emb_enc = Bidirectional(LSTM(HIDDEN_UNITS, return_sequences=True))(emb_enc)
enc_out, fh, fc, bh, bc = Bidirectional(LSTM(HIDDEN_UNITS, return_state=True))
    (emb_enc)
state_h = Concatenate()([fh,bh])
state_c = Concatenate()([fc,bc])

# Decoder
dec_in = Input((None,))
emb_dec = Embedding(fra_vocab_size, EMBEDDING_DIM, trainable=True)(dec_in)
dec_out, _, _ = LSTM(HIDDEN_UNITS*2, return_sequences=True, return_state=True)
    (emb_dec, initial_state=[state_h,state_c])
outputs = TimeDistributed(Dense(fra_vocab_size, activation='softmax', dtype='
    float32'))(dec_out)
model = Model([enc_in, dec_in], outputs)
```

4.2 Training

I optimized the loss:

$$\mathcal{L} = -\sum_t \log P(y_t^* | y_{<t}^*, x),$$

using Adam (lr=1e-3), batch size 256, early stopping (patience=3). Dynamic bucketing groups similar-length sequences to reduce padding.

4.3 Results

Beam search decoding (beam-width=3) yields BLEU ≈28.80% on held-out samples.

5 Part 2: Seq2Seq with Additive Attention

5.1 Attention Mechanism

Additive (Bahdanau) attention computes context vectors:

$$e_{t,i} = v^T \tanh(W_1 s_{t-1} + W_2 h_i), \tag{4}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_j \exp(e_{t,j})}, \tag{5}$$

$$c_t = \sum_i \alpha_{t,i} h_i. \tag{6}$$

Listing 4: Integrating additive attention

```
# After decoder LSTM outputs dec_seq
attention = AdditiveAttention()
context = attention([dec_seq, enc_seq])
combined = Concatenate()([dec_seq, context])
outputs = TimeDistributed(Dense(fra_vocab_size, activation='softmax', dtype='
    float32'))(combined)
```

5.2 Training and Results

Same optimizer and loss, with patience=5. BLEU improves to 37.12%.

6 Part 3: Transformer Fine-Tuning

6.1 Transformer Architecture

Self-attention layer:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

Listing 5: Hugging Face fine-tuning setup

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, \
                         Seq2SeqTrainer, Seq2SeqTrainingArguments
max_input_length = 128
max_target_length = 128

def preprocess_function(examples):
    inputs = tokenizer(examples["en"], max_length=max_input_length, padding="
        max_length", truncation=True)
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(examples["fr"], max_length=max_target_length,
            padding="max_length", truncation=True)
    inputs["labels"] = labels["input_ids"]
    return inputs
```

```
model_checkpoint = "Helsinki-NLP/opus-mt-en-fr"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)

training_args = Seq2SeqTrainingArguments(
    output_dir="./transformer-nmt",
    save_strategy="epoch",
    learning_rate=5e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    predict_with_generate=True,
    logging_strategy="epoch",
    fp16=True,
    report_to="none"
)

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
trainer.train()
```

Model used : Helsinki-NLP/opus-mt-en-fr
BLEU reaches 56.80% after fine-tuning.

7 Bonus Task :Attention Visualization

To better understand model interpretability, I visualized attention weights using heatmaps for selected English-French sentence pairs. These visualizations reveal alignment patterns and highlight which source tokens contributed most to each generated target token.

8 Evaluation and Comparisons

Aspect	Part 1	Part 2	Part 3
BLEU Score (%)	28.80	37.12	56.80
Epochs	30	30	3
Model Parameters	37.7M	62M	59.8M

Table 2: Comparison of evaluation metrics and resource usage across parts.

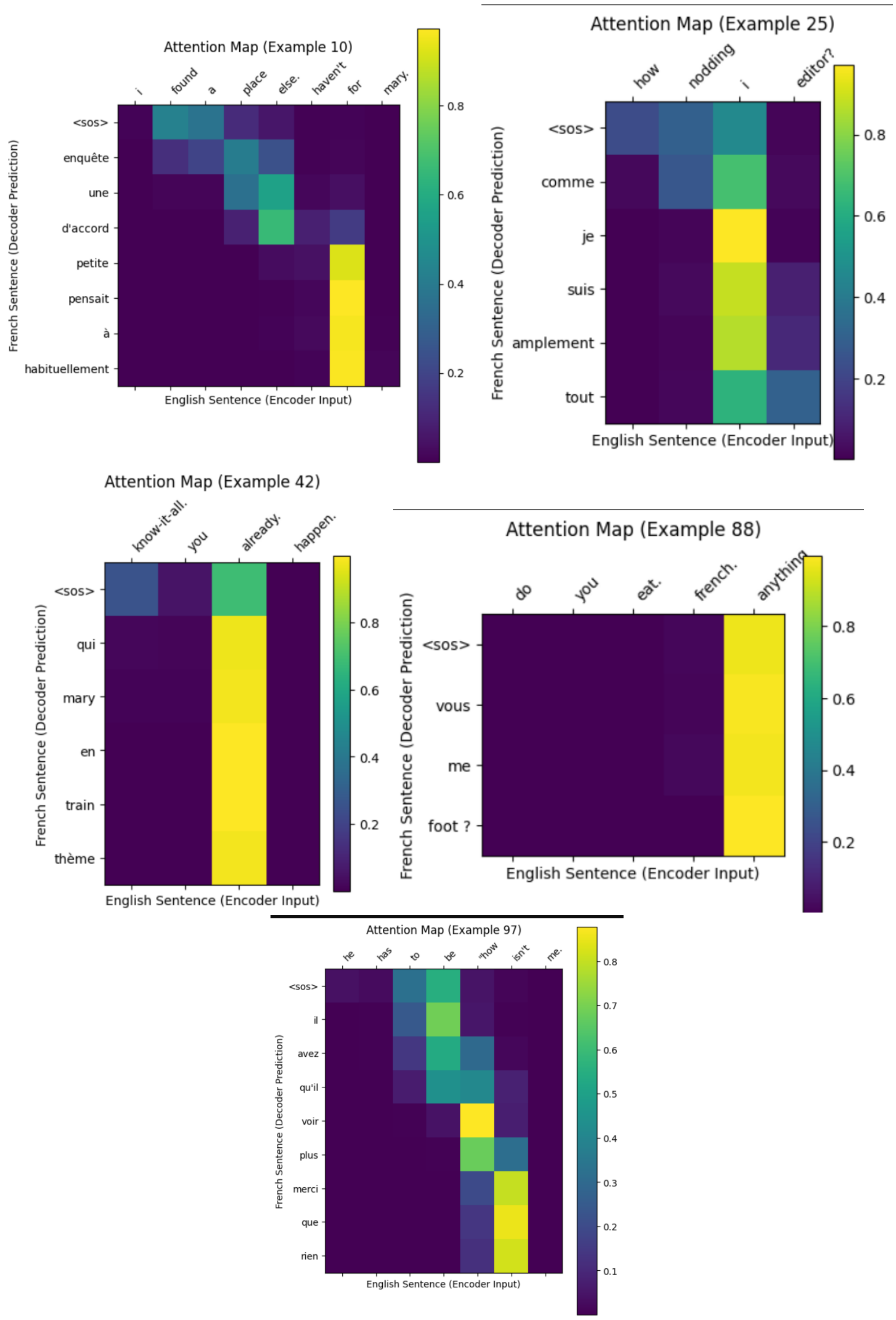

Figure 1: Sample Attention visualizations.

9 Challenges and Solutions

9.1 Data Handling

- **Parsing Errors**: Extra metadata in `fra.txt`. *Fix*: Extract only first two columns via `split(’’)[:2]`.
- **Tokenizer Mismatch**: Keras vs HF tokenizers. *Fix*: Keep separate tokenizers for RNN and Transformer parts.

9.2 Model Training

- **Low BLEU in Part 1** (less than 30%). *Fix*: Added bidirectional layers, initialized encoder with GloVe.
- **Attention Instability** in Part 2. *Fix*: Masked padding in AdditiveAttention, increased hidden units.
- **Transformer** BLEU=0 initially. *Fix*: Re-tokenized dataset with HF tokenizer, corrected label IDs.

9.3 Resource Optimization

- **Long Epochs**: Greater than 3h per epoch on Kaggle. *Fix*: Enabled mixed precision, XLA JIT, and dataset slicing.

10 Future Work

Possible enhancements include multilingual extension, pretrained French embeddings, multi-head custom attention, learning rate schedulers, and detailed error analysis.

11 Download Links

- **Dataset (English–French Pairs):** https://www.manythings.org/anki/
- **GloVe 100d Embeddings:** https://nlp.stanford.edu/data/glove.6B.zip

12 Conclusion

I began with a simple Seq2Seq model and gradually advanced to attention-based and Transformer architectures, boosting BLEU from 28.8% to 56.8%. Along the way, attention visualizations helped me understand model behavior. It helped me get a deeper grasp of neural machine translation through thoughtful design, rigorous training, and meaningful evaluation.