Department of Computing

CS 818: Large Language Models (LLMs)

Assignment 02

Seq2Seq models for Neural Machine Translation

Instructor

Prof. Dr. Muhammad Moazam Fraz

A 02: Seq2Seq models for Neural Machine Translation

Learning Outcome

(To be added)

Tools / Software Requirements

Python (>= **3.7**), **IDE:** PyCharm (preferred) or VSCode

Contents

1) Objectives

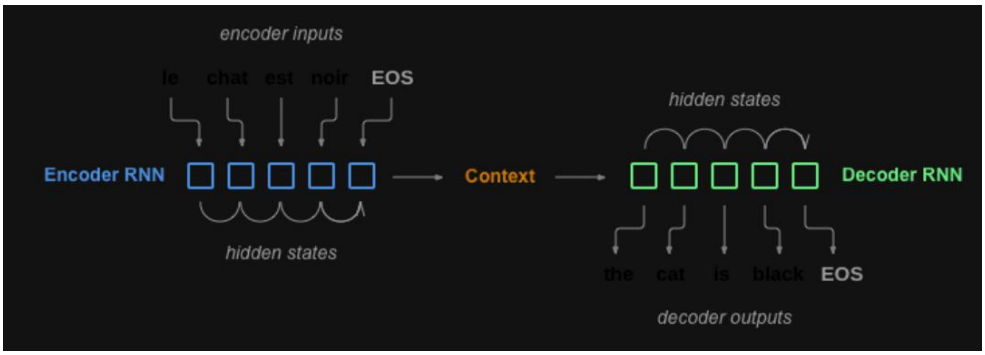
The objective of this lab is to understand and implement a sequence-to-sequence (seq2seq) neural architecture for machine translation, focusing on the use of encoder-decoder networks to handle variable-length input and output sequences. Students will also apply attention mechanisms, particularly Bahdanau attention, and experiment with different datasets, pretrained embeddings, and architectural variations to gain deeper insights into model performance and behavior.

2) Core Ideas

- **Seq2Seq Modeling** enables end-to-end translation by encoding a source sentence into a fixed-length vector and decoding it into a target sentence.
- **Encoder-Decoder Structure** uses two RNNs: one to process the input (encoder) and one to generate the output (decoder).
- **Context Vector** represents the compressed information from the encoder used to initialize the decoder.
- **Attention Mechanism**, like Bahdanau attention, enhances translation by allowing the decoder to focus on relevant parts of the input at each step.
- This lab emphasizes the practical application of these components in building a functional Neural Machine Translation (NMT) system.

3) Overview of Sequence-to-sequence Modelling

Sequence-to-sequence modelling is a type of neural network architecture that is designed to handle variable-length input and output sequences. It consists of two main components: an encoder and a decoder. The encoder takes in a sequence of input data, such as a sentence, and encodes it into a fixed-length vector. The decoder then takes in the encoded vector and generates a sequence of output data, such as a translation of the input sentence. The encoder and decoder are typically implemented using recurrent neural networks (RNNs) or variants of RNNs such as long short-term memory (LSTM) networks, or gated recurrent units (GRUs).



Neural Machine Translation

Among many applications of the sequence to sequence models is the Neural Machine Translation (NMT) is an approach to automatically translating text from one language to another using deep neural networks.

Core Idea

NMT treats translation as a sequence-to-sequence learning task — meaning it takes a sentence in one language (source) and generates a sentence in another language (target) using a single, unified neural model.

Basic Architecture

A seq2seq model for machine translation relies on a two-part architecture: an encoder and a decoder. Here's a breakdown of their functionalities:

A seq2seq network, also known as Encoder Decoder network, is a model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence. Unlike sequence prediction with a single RNN, where every input corresponds to an output, the seq2seq model frees us from sequence length and order, which makes it ideal for translation between two slanguages.

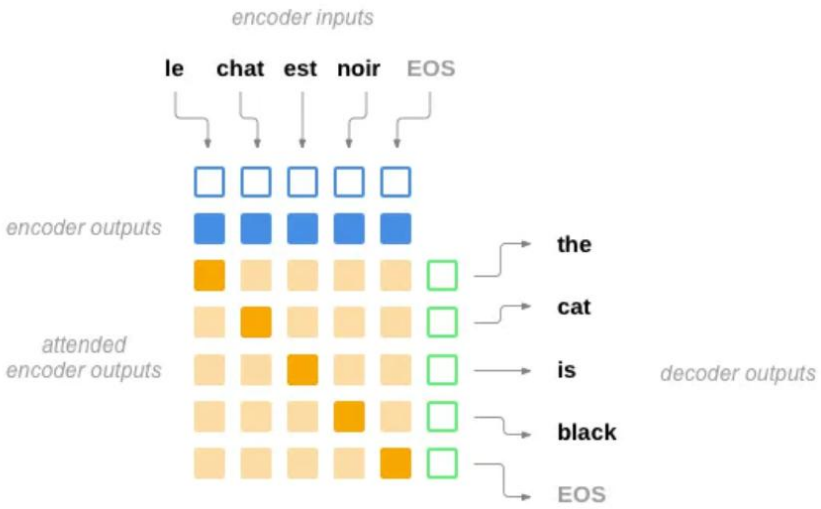
Example

Consider the sentence:-

- `Je ne suis pas le chat noir` →`I am not the black cat`

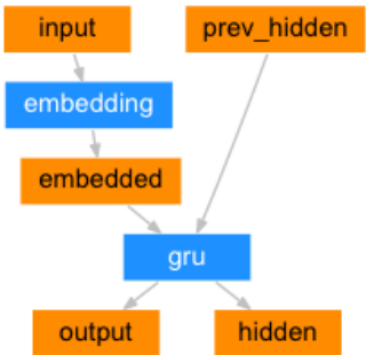
Most of the words in the input sentence have a direct translation in the output sentence, but are in slightly different orders, e.g. `chat noir` and `black cat`. Because of the `ne/pas` construction there is also one more word in the input sentence. It would be difficult to produce a correct translation directly from the sequence of input words.

With a seq2seq model the encoder creates a single vector which, in the ideal case, encodes the meaning of the input sequence into a single vector, a single point in some N dimensional space of sentences.



Encoder

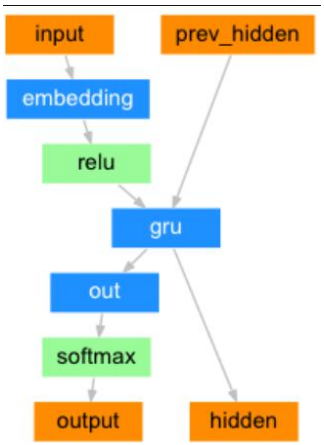
The encoder of a seq2seq network is a RNN that outputs some value for every word from the input sentence. For every input word the encoder outputs a vector and a hidden state, and uses the hidden state for the next input word.



Decoder

The decoder is another RNN that takes the encoder output vector(s) and outputs a sequence of words to create the translation. In the simplest seq2seq decoder we use only last output of the encoder. This last output is sometimes called the *context vector* as it encodes context from the entire sequence. This context vector is used as the initial hidden state of the decoder.

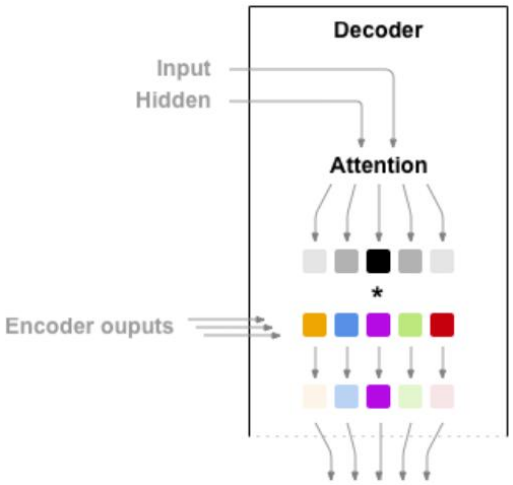
At every step of decoding, the decoder is given an input token and hidden state. The initial input token is the start-of-string <SOS> token, and the first hidden state is the context vector (the encoder's last hidden state).



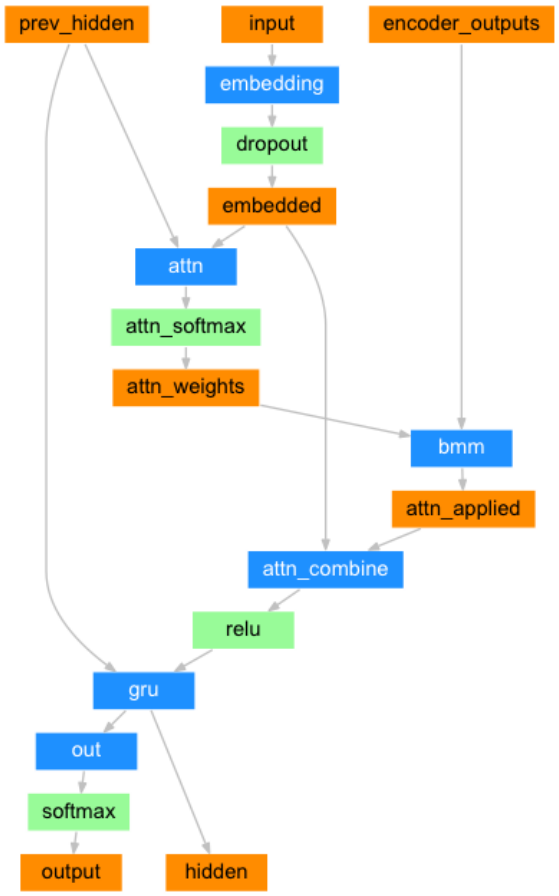
Attention Decoder

If only the context vector is passed between the encoder and decoder, that single vector carries the burden of encoding the entire sentence.

Attention allows the decoder network to *focus* on a different part of the encoder's outputs for every step of the decoder's own outputs. First, we calculate a set of attention weights. These will be multiplied by the encoder output vectors to create a weighted combination. The result should contain information about that specific part of the input sequence, and thus help the decoder choose the right output words.



Attention weights are calculated with another feed-forward layer attn, using the decoder's input and hidden state as inputs. Because there are sentences of all sizes in the training data, to actually create and train this layer we have to choose a maximum sentence length (input length, for encoder outputs) that it can apply to. Sentences of the maximum length will use all the attention weights, while shorter sentences will only use the first few.



Bahdanau Attention

Bahdanau attention, also known as additive attention, is a commonly used attention mechanism in sequence-to-sequence models, particularly in neural machine translation tasks. It was introduced by Bahdanau et al. [1] in their paper titled Neural Machine Translation by Jointly Learning to Align and Translate. This attention mechanism employs a learned alignment model to compute attention scores between the encoder and decoder hidden states. It utilizes a feed-forward neural network to calculate alignment scores. However, there are alternative attention mechanisms available, such as Luong attention, which computes attention scores by taking the dot product between the decoder hidden state and the encoder hidden states. It does not involve the non-linear transformation used in Bahdanau attention.

4) Tasks

Task 1:
Your task is to apply the given sequence-to-sequence model architecture and helper functions (such as Lang, tensorFromSentence, trainIters, etc.) to a different dataset. The goal is to test the model's adaptability across various types of input-output language mappings beyond the provided English–French translation pairs.

You may choose from the following alternatives:

- **Another language pair** (e.g., Spanish–English, Urdu–English)
  Download a bilingual corpus and preprocess it similarly to the given dataset format (eng-fra.txt).
- **Human → Machine commands**
  For example:
  - Input: "Turn off the lights."
  - Output: "device.off('lights')"
    This could simulate smart home or IOT instructions.
- **Chat → Response pairs**
  Build a dataset where one line is a user message and the next is an appropriate chatbot reply.
- **Question → Answer pairs**
  Prepare a dataset of factual questions and their answers (e.g., from a quiz dataset or knowledge base)

Task 2:

In the second task, you are suggested to replace the embeddings with pretrained word embeddings such as word2vec or GloVe.

Task 3:

In the third task, you have to experiment with the modifications in the given architecture. You may try adding more layers, more hidden units, and more sentences. Compare the training time and results. Report your results and discuss the findings.

Note:- For the details of the tasks and basic structure of the code, please refer to the attached notebook.

5) References

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