Homework Assignment #3

The one with the RNNs

Description

This homework assignment focuses on developing your skills to work with recurrent neural networks.
It has two main tasks:

1. Implement a basic RNN Cell and understand the role of **teacher forcing, warm start** and **capturing of the structure of interest in a sequence** on hand of a simple **Sine Wave Dataset**.
2. To **develop, train and test** your own **sequence-to-sequence model** for:
    a. A toy problem of arabic-to-roman numeral conversion
    b. A language translation problem

Task 1 [4p]

The objective of this task is the implementation of a simple RNN cell and to understand how teacher forcing and warm start affect the **prediction capabilities** of an RNN.

The Jupyter notebook attached to this assignment provides the setup for this task. The mentioned TODOs follow this schema:

- TODO 1.1 targets implementing the forward pass for the RNN base cell (the recurrent step which updates $\mathbf{h}_t$ based on $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$)
- TODO 1.2 completes our VanillaRNN implementation by creating the layers that map from input to hidden space and from hidden to output. This is done in the ***step()*** method which performs on timestep in the RNN computation
- TODO 1.3 implements the whole forward step of an RNN by going over the entire input sequence.
    - It asks to implement **teacher forcing during training** (input at next time step *t* is given by the ground truth at time step *t* instead of the prediction made by the network at time step *t-1*)
    - It asks you to implement **warm start during evaluation** - for a **limited amount of timesteps** the network is given ground truth input instead of its own predictions

After implementing the TODOs, you will train the network on the simple task of predicting values from a sine wave.
Several hyperparameters control the learning and generation of sine sequences.

- UNROLL_LENGTH - gives the length of subsequences from the sine wave used during training
- TEACHER_FORCING - controls the use of teacher forcing

- WARM_START - sets the max number of timesteps considered in the warm starting of the RNN when in prediction mode
- NUM_ITERATIONS, LEARNING_RATE and REPORTING_INTERVAL - control the training and reporting process

In your training perform the following experiments:
- Run **with** and **without teacher forcing** using the default parameters
- Keep Teacher forcing on and use a **very small** and a **very large** UNROLL_LENGTH
- Use UNROLL_LENGTH = 62 (maximum), set teacher forcing **off** and use a warm start of only 2. Does the model learn?
- Use an UNROLL_LENGTH of 3, put teacher forcing **on** and use a warm start of 2. Does the model learn?

After performing the experiments answer to the following questions:
1. Difference between teacher forcing and learning on own samples:
    a. What are the pros and cons of teacher forcing?
    b. In which setup is the model struggling to learn and why?
2. How does warm starting affect our training? Why?
3. What happens if the *structure of interest* is much longer than the unroll length?

Task 2 [6p]

This task focuses on the development of a sequence-to-sequence architecture using LSTM or GRU layers.

The typical architecture of a basic sequence-to-sequence model for *language translation* is shown in the Figure 1 below, which shows an unrolled representation of the *encoding* and *decoding* modules for a french-to-english translation of the phrase “Good evening”.

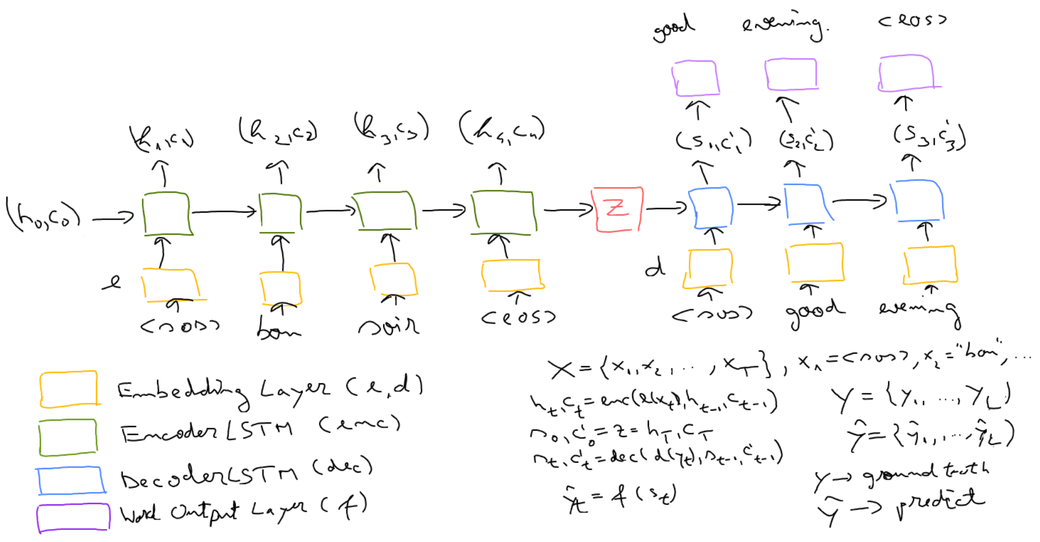

Figure 1. Depiction of a simple language translation seq-2-seq model using single layer LSTMs.

The network comprises text embedding layers (one for the encoder - french words - and one for the decoder - english words).
Encoder and Decoder are implemented as single layer LSTMs.
The Word Output Layer (purple depiction in Figure 1) converts the hidden state of the decoder (**s**$_t$) to a vector the size of the output language vocabulary. It uses a Linear Layer to do this and applies a softmax on top of it to indicate the most likely token.

You have to perform the following subtasks.

Task 2a (4p)

The objective is to implement a seq-2-seq model for the french-to-english translation task.

To do so we use the [Multi30k dataset](). This is a dataset with ~30,000 parallel English, German and French sentences, each with ~12 words per sentence.
To obtain tokenizers for French and English sentences, you can use the [**spaCy**]() small language models for French ("fr_core_news_sm") and English ("en_core_web_sm"). See [here]() an example documentation for how to load and use the language models for tokenization.

Objective 1. Implement and test a **single layer LSTM encoder** and **decoder model** for a seq-to-seq translation model trained on the Multi30k dataset.
Experiment with:

- Using input dropout - adding a dropout layer (with probability 0.1 or 0.5) after the initial text embedding layer
- Different sizes of the hidden dimension (e.g. 256, 512, 1024)
- Different batch sizes (e.g. 128, 256)

Objective 2. Modify the architecture to work in a double-layer encoder and decoder models, as shown in Figure 2.

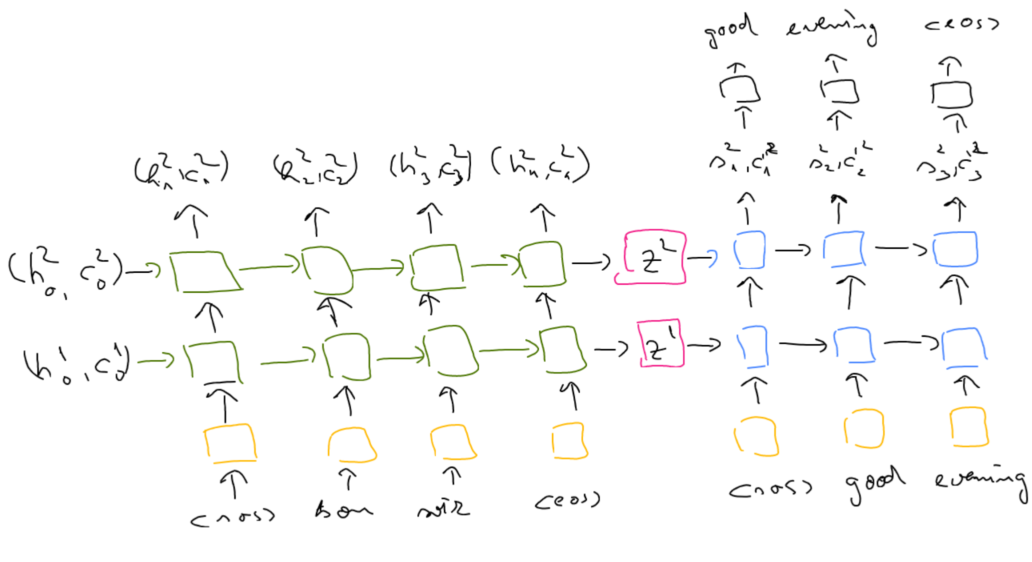

Figure 2. 2-layer LSTM models for encoder and decoder in the seq-2-seq language translation model

Experiment with:

- The same aspects as for Objective 1
- Adding dropout layer in between the 1st and 2nd layer of LSTMs in encoder and decoder models
- **Perform a comparison** between using a single layer seq-2-seq model with a higher hidden dimension size (e.g. 512) as opposed to the double layer LSTM model with lower hidden size (e.g. 128).

Objective 3. Implement a seq-2-seq model with a **2-layer encoder model** and a **single layer decoder model** (see Figure 3).

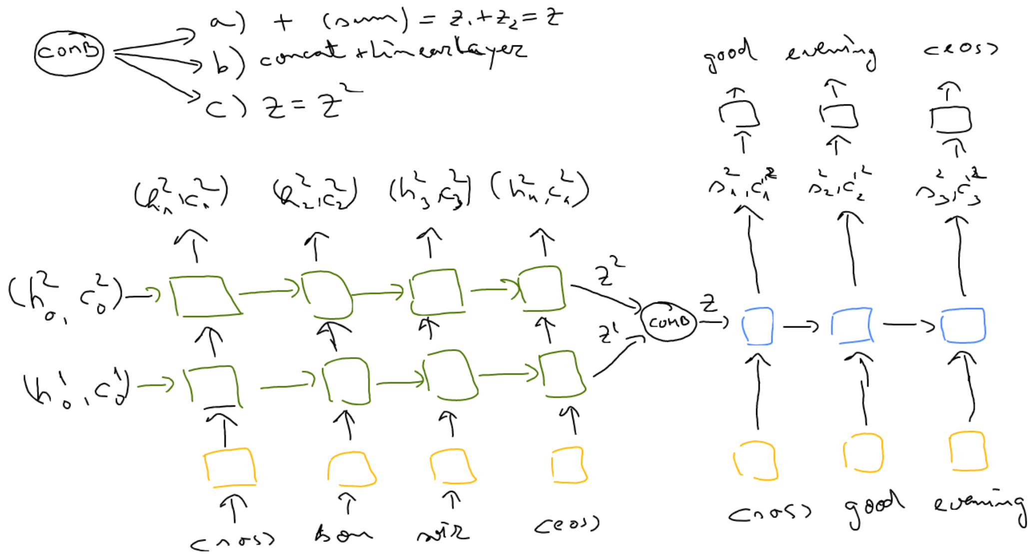

Figure 3. Seq-2-seq language translation model using a 2-layer LSTM encoder and a single layer LSTM decoder.

Experiment with:

- Different ways to combine the encoder *context vectors (z^1, z^2)*
    - Sum them and ensure that the decoder hidden size has the same size as the context vectors
    - Concatenate them and use a LinearLayer based projection to the hidden size of the decoder LSTM model
    - Just use one of the vectors (e.g. z^2)
- **Compare** with the previously mentioned models in terms of performance

Task 2b (2p)

This task requires you to use the single-layer seq-to-seq model from Objective 1 in Task 2a to implement **an arabic to roman numeral translation model.**
Use the **utils.py** file provided in the annex to this assignment to create a dataset consisting of pairs of numerals in the range [1, 2021].

The dataset consists of pairs of character sequences such as:
- [“5”, “1”, “1”] → [“D”, “X”, “I”]
- [“2”, “0”, “0”, “0”] → [“M”, “M”]

Experiment with:
- The size of the text embedding vectors representing each digit
- The hidden dimension size
- The batch size and the use of **shuffled** or **sorted (in terms of output sequence length)** batches
- Provide a qualitative analysis of the results.