DLS/C5/WEEK3/Assignment 1 /SI

Neural_machine_translation_with_attention_v4a

Neural Machine Translation

Welcome to your first programming assignment for this week!

- You will build a Neural Machine Translation (NMT) model to translate human-readable dates ("25th of June, 2009") into machine-readable dates ("2009-06-25").
- You will do this using an attention model, one of the most sophisticated sequence-to-sequence models.

This notebook was produced together with NVIDIA's Deep Learning Institute.

Important Note on Submission to the AutoGrader

Before submitting your assignment to the AutoGrader, please make sure you are not doing the following:

1. You have not added any *extra* `print` statement(s) in the assignment.
2. You have not added any *extra* code cell(s) in the assignment.
3. You have not changed any of the function parameters.
4. You are not using any global variables inside your graded exercises. Unless specifically instructed to do so, please refrain from it and use the local variables instead.
5. You are not changing the assignment code where it is not required, like creating *extra* variables.

If you do any of the following, you will get something like, `Grader not found` (or similarly unexpected) error upon submitting your assignment. Before asking for help/debugging the errors in your assignment, check for these first. If this is the case, and you don't remember the changes you have made, you can get a fresh copy of the assignment by following these instructions.

Table of Contents

Packages

In [1]:
```
from tensorflow.keras.layers import Bidirectional, Concatenate, Permute, Dot, Input, LSTM, Multiply
from tensorflow.keras.layers import RepeatVector, Dense, Activation, Lambda
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model, Model
import tensorflow.keras.backend as K
import tensorflow as tf
import numpy as np

from faker import Faker
import random
from tqdm import tqdm
from babel.dates import format_date
from nmt_utils import *
import matplotlib.pyplot as plt
%matplotlib inline
```

1 - Translating Human Readable Dates Into Machine Readable Dates

- The model you will build here could be used to translate from one language to another, such as translating from English to Hindi.
- However, language translation requires massive datasets and usually takes days of training on GPUs.
- To give you a place to experiment with these models without using massive datasets, we will perform a simpler "date translation" task.
- The network will input a date written in a variety of possible formats (*e.g. "the 29th of August 1958", "03/30/1968", "24 JUNE 1987"*)
- The network will translate them into standardized, machine readable dates (*e.g. "1958-08-29", "1968-03-30", "1987-06-24"*).
- We will have the network learn to output dates in the common machine-readable format YYYY-MM-DD.

1.1 - Dataset

We will train the model on a dataset of 10,000 human readable dates and their equivalent, standardized, machine readable dates. Let's run the following cells to load the dataset and print some examples.

In [2]:
```
m = 10000
dataset, human_vocab, machine_vocab, inv_machine_vocab = load_dataset(m)
```
```
100%|████████████| 10000/10000 [00:00<00:00, 23073.33it/s]
```

In [3]:
```
dataset[:10]
```

Out[3]:
```
[('9 may 1998', '1998-05-09'),
 ('10.11.19', '2019-11-10'),
 ('9/10/70', '1970-09-10'),
 ('saturday april 28 1990', '1990-04-28'),
 ('thursday january 26 1995', '1995-01-26'),
 ('monday march 7 1983', '1983-03-07'),
 ('sunday may 22 1988', '1988-05-22'),
 ('08 jul 2008', '2008-07-08'),
 ('8 sep 1999', '1999-09-08'),
 ('thursday january 1 1981', '1981-01-01')]
```

You've loaded:

- `dataset`: a list of tuples of (human readable date, machine readable date).
- `human_vocab`: a python dictionary mapping all characters used in the human readable dates to an integer-valued index.
- `machine_vocab`: a python dictionary mapping all characters used in machine readable dates to an integer-valued index.

- `machine_vocab`: a python dictionary mapping all characters used in machine readable dates to an integer-valued index.
  - **Note**: These indices are not necessarily consistent with `human_vocab`.
- `inv_machine_vocab`: the inverse dictionary of `machine_vocab`, mapping from indices back to characters.

Let's preprocess the data and map the raw text data into the index values.

- We will set Tx=30
  - We assume Tx is the maximum length of the human readable date.
  - If we get a longer input, we would have to truncate it.
- We will set Ty=10
  - "YYYY-MM-DD" is 10 characters long.

.]:
```
Tx = 30
Ty = 10
X, Y, Xoh, Yoh = preprocess_data(dataset, human_vocab, machine_vocab, Tx, Ty)

print("X.shape:", X.shape)
print("Y.shape:", Y.shape)
print("Xoh.shape:", Xoh.shape)
print("Yoh.shape:", Yoh.shape)
```

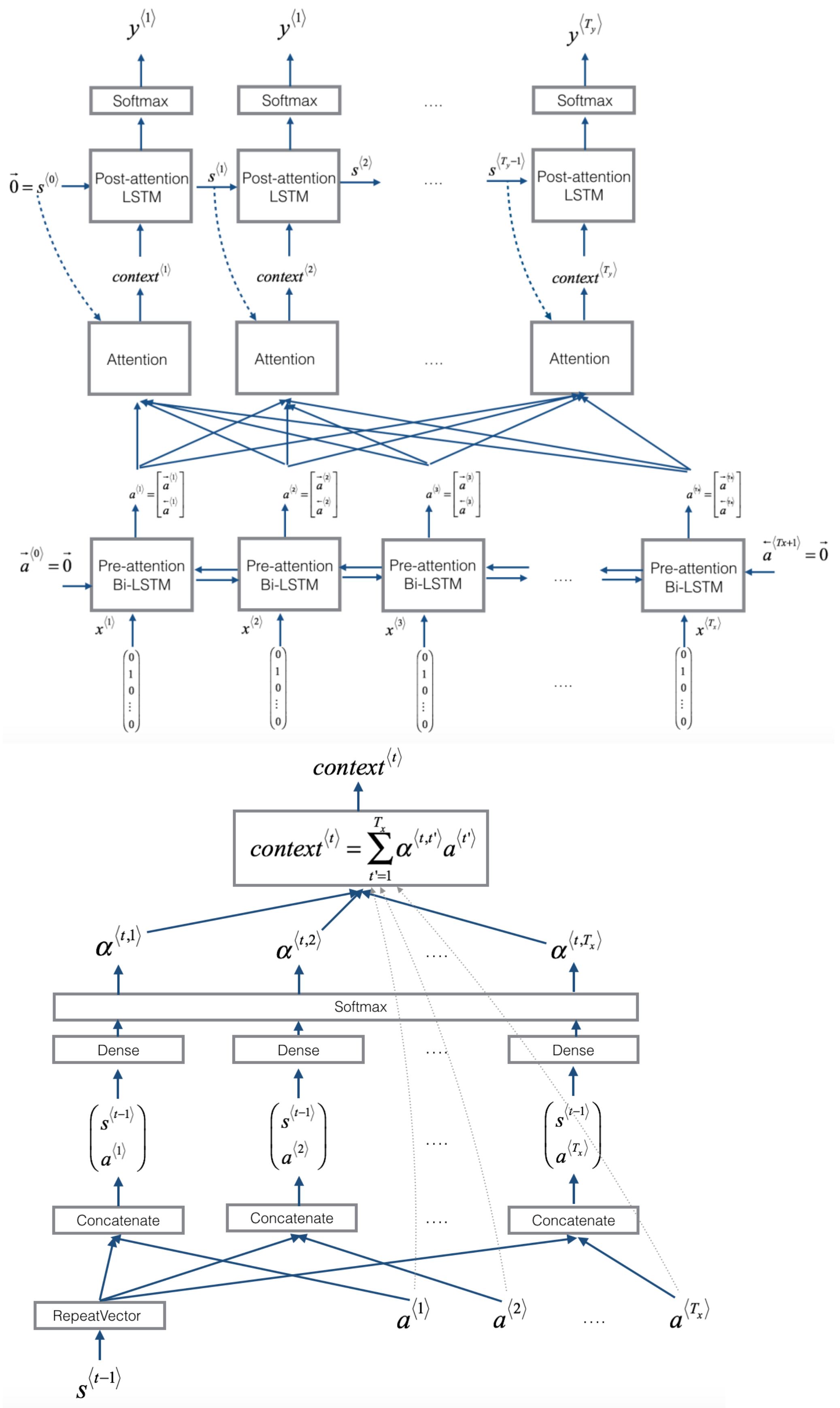
2 - Neural Machine Translation with Attention

- If you had to translate a book's paragraph from French to English, you would not read the whole paragraph, then close the book and translate.
- Even during the translation process, you would read/re-read and focus on the parts of the French paragraph corresponding to the parts of the English you are writing down.
- The attention mechanism tells a Neural Machine Translation model where it should pay attention to at any step.

2.1 - Attention Mechanism

In this part, you will implement the attention mechanism presented in the lecture videos.

- Here is a figure to remind you how the model works.
  - The diagram on the left shows the attention model.
  - The diagram on the right shows what one "attention" step does to calculate the attention variables $\alpha^{\langle t, t' \rangle}$.
  - The attention variables $\alpha^{\langle t, t' \rangle}$ are used to compute the context variable $context^{\langle t \rangle}$ for each timestep in the output ($t = 1, \ldots, T_y$).



$y^{\langle 1 \rangle}$
$y^{\langle 1 \rangle}$
$y^{\langle T_y \rangle}$
Softmax
Softmax
Softmax
....
$\vec{0} = s^{\langle 0 \rangle}$
Post-attention LSTM
$s^{\langle 1 \rangle}$
Post-attention LSTM
$s^{\langle 2 \rangle}$
....
$s^{\langle T_y - 1 \rangle}$
Post-attention LSTM
$context^{\langle 1 \rangle}$
$context^{\langle 2 \rangle}$
$context^{\langle T_y \rangle}$
Attention
Attention
....
Attention
$a^{\langle 1 \rangle} = \begin{bmatrix} \vec{a}^{\langle 1 \rangle} \\ \overleftarrow{a}^{\langle 1 \rangle} \end{bmatrix}$
$a^{\langle 2 \rangle} = \begin{bmatrix} \vec{a}^{\langle 2 \rangle} \\ \overleftarrow{a}^{\langle 2 \rangle} \end{bmatrix}$
$a^{\langle 3 \rangle} = \begin{bmatrix} \vec{a}^{\langle 3 \rangle} \\ \overleftarrow{a}^{\langle 3 \rangle} \end{bmatrix}$
$a^{\langle T_x \rangle} = \begin{bmatrix} \vec{a}^{\langle T_x \rangle} \\ \overleftarrow{a}^{\langle T_x \rangle} \end{bmatrix}$
$\vec{a}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
....
Pre-attention Bi-LSTM
$\overleftarrow{a}^{\langle Tx+1 \rangle} = \vec{0}$
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$
$\begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$
....
$context^{\langle t \rangle}$
$context^{\langle t \rangle} = \sum_{t'=1}^{T_x} \alpha^{\langle t,t' \rangle} a^{\langle t' \rangle}$
$\alpha^{\langle t,1 \rangle}$
$\alpha^{\langle t,2 \rangle}$
....
$\alpha^{\langle t,T_x \rangle}$
Softmax
Dense
Dense
....
Dense
$\begin{pmatrix} s^{\langle t-1 \rangle} \\ a^{\langle 1 \rangle} \end{pmatrix}$
$\begin{pmatrix} s^{\langle t-1 \rangle} \\ a^{\langle 2 \rangle} \end{pmatrix}$
....
$\begin{pmatrix} s^{\langle t-1 \rangle} \\ a^{\langle T_x \rangle} \end{pmatrix}$
Concatenate
Concatenate
....
Concatenate
RepeatVector
$a^{\langle 1 \rangle}$
$a^{\langle 2 \rangle}$
....
$a^{\langle T_x \rangle}$
$s^{\langle t-1 \rangle}$

Figure 1: Neural machine translation with attention

Here are some properties of the model that you may notice:

Pre-attention and Post-attention LSTMs on both sides of the attention mechanism

- There are two separate LSTMs in this model (see diagram on the left): pre-attention and post-attention LSTMs.
- *Pre-attention* Bi-LSTM is the one at the bottom of the picture is a Bi-directional LSTM and comes *before* the attention mechanism.
  - The attention mechanism is shown in the middle of the left-hand diagram.
  - The pre-attention Bi-LSTM goes through T_x time steps
- *Post-attention* LSTM: at the top of the diagram comes *after* the attention mechanism.
  - The post-attention LSTM goes through T_y time steps.
- The post-attention LSTM passes the hidden state $s^{\langle t \rangle}$ and cell state $c^{\langle t \rangle}$ from one time step to the next.

While training you can see the loss as well as the accuracy on each of the 10 positions of the output. The table below gives you an example of what the accuracies could be if the batch had 2 examples:

True labels 1	1	9	9	5	-	1	2	-	0	4
Predictions 1	1	9	9	5	-	1	0	-	0	5
True labels 1	1	9	6	8	-	0	1	-	0	4
Predictions 2	1	9	7	8	-	0	3	-	0	4
Index	1	2	3	4	5	6	7	8	9	10
Accuracy	1.0	1.0	0.5	1.0	1.0	1.0	0.0	1.0	1.0	0.5

Thus, `dense_2_acc_8: 0.89` means that you are predicting the 7th character of the output correctly 89% of the time in the current batch of data.

We have run this model for longer, and saved the weights. Run the next cell to load our weights. (By training a model for several minutes, you should be able to obtain a model of similar accuracy, but loading our model will save you time.)

```
In [18]: model.load_weights('models/model.h5')
```

3 - Visualizing Attention (Optional / Ungraded)

Since the problem has a fixed output length of 10, it is also possible to carry out this task using 10 different softmax units to generate the 10 characters of the output. But one advantage of the attention model is that each part of the output (such as the month) knows it needs to depend only on a small part of the input (the characters in the input giving the month). We can visualize what each part of the output is looking at which part of the input.

Consider the task of translating "Saturday 9 May 2018" to "2018-05-09". If we visualize the computed $\alpha^{\langle t, t' \rangle}$ we get this:

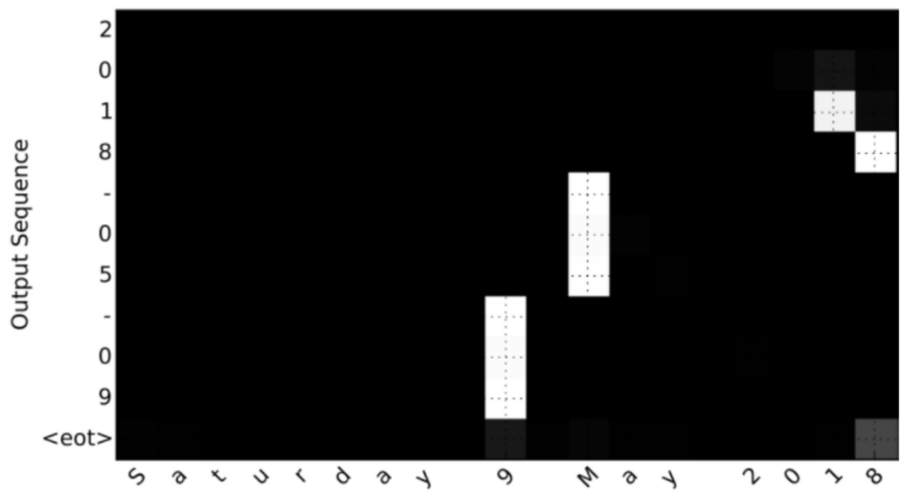


Figure 8: Full Attention Map

Notice how the output ignores the "Saturday" portion of the input. None of the output timesteps are paying much attention to that portion of the input. We also see that 9 has been translated as 09 and May has been correctly translated into 05, with the output paying attention to the parts of the input it needs to to make the translation. The year mostly requires it to pay attention to the input's "18" in order to generate "2018."

3.1 - Getting the Attention Weights From the Network

Lets now visualize the attention values in your network. We'll propagate an example through the network, then visualize the values of $\alpha^{\langle t, t' \rangle}$.

To figure out where the attention values are located, let's start by printing a summary of the model .