perplexity

OK. Now I'll start writing the code in the notebook. I'll provide you the 'hi' language directory inside 'dakshina_dataset_v1.0'. 'hi' which has 3 subdirectories - 'native_script_wikipedia', 'lexicons', 'romanized'. I'll provide you files from the last 2 directories so that you have complete understanding of how to input the data into the notebook according to the problem statement (still attached) and the questions. There are total 12 language directories like 'hi' with their codes mentioned in the readme file with the same files inside.

There are 3 files in the lexicons directory -

1. hi.translit.sampled.train.tsv (I've converted them to text and attached here)
2. hi.translit.sampled.dev.tsv (I've converted them to text and attached here)
3. hi.translit.sampled.test.tsv (I've converted them to text and attached here)

Every other file I attach is from romanized directory. Visualize the file structure of my project folder here as well (remember, I'm writing everthing systematically in a jupyter notebook for all questions one by one. Also, each subsequent question might be connected to the previous questions so please code like that). My notebook's name is "Assignment 3.ipynb"

Question 1 (15 Marks):
Build a RNN based seq2seq model which contains the following layers: (i) input layer for character embeddings (ii) one encoder RNN which sequentially encodes the input character sequence (Latin) (iii) one decoder RNN which takes the last state of the encoder as input and produces one output character at a time (Devanagari).

The code should be flexible such that the dimension of the input character embeddings, the hidden states of the encoders and decoders, the cell (RNN, LSTM, GRU) and the number of layers in the encoder and decoder can be changed.

(a) What is the total number of computations done by your network? (assume that the input embedding size is m, encoder and decoder have 1 layer each, the hidden cell state is k for both the encoder and decoder, the length of the input and output sequence is the same, i.e., T, the size of the vocabulary is the same for the source and target language, i.e., V)

(b) What is the total number of parameters in your network? (assume that the input embedding size is m, encoder and decoder have 1 layer each, the hidden cell state is k for both the encoder and decoder and the length of the input and output sequence is the same, i.e., T, the size of the vocabulary is the same for the source and target language, i.e., V)

Okay, I understand the problem. You're tasked with building a transliteration model using a sequence-to-sequence RNN architecture. The model will take a Romanized word (e.g., "ghar") as input and output its Devanagari equivalent (e.g., "घर"). You'll be using the Dakshina dataset, specifically the lexicon files containing word pairs.

Here's a visualization of your project folder structure based on your description:

```
<project_folder>/
├── Assignment 3.ipynb
└── dakshina_dataset_v1.0/
    ├── bn/                            # Bengali
    │   ├── lexicons/
    │   ├── native_script_wikipedia/
    │   └── romanized/
    ├── gu/                            # Gujarati
    │   └── ...
    ├── hi/                            # Hindi
    │   ├── lexicons/
    │   │   ├── hi.translit.sampled.train.tsv
    │   │   ├── hi.translit.sampled.dev.tsv
    │   │   └── hi.translit.sampled.test.tsv
    │   ├── native_script_wikipedia/
    │   └── romanized/
    │       ├── hi.romanized.rejoined.dev.native.txt
    │       ├── hi.romanized.rejoined.dev.roman.txt
    │       ├── hi.romanized.rejoined.test.native.txt
    │       ├── hi.romanized.rejoined.test.roman.txt
    │       ├── hi.romanized.split.validation.edits.txt
    │       └── ... (other files as per the dataset structure)
    ├── kn/                            # Kannada
    │   └── ...
    ├── ml/                            # Malayalam
    │   └── ...
    ├── mr/                            # Marathi
    │   └── ...
    ├── pa/                            # Punjabi
    │   └── ...
    ├── sd/                            # Sindhi
    │   └── ...
    ├── si/                            # Sinhala
    │   └── ...
    ├── ta/                            # Tamil
    │   └── ...
    ├── te/                            # Telugu
    │   └── ...
    └── ur/                            # Urdu
        └── ...
```

DA6401 - Assignment 3 - CE21B097

Question 1

(a) Total number of computations

The total number of computations (approximated as Floating Point Operations - FLOPs) in one forward pass of your network, with the given assumptions (input embedding size m, 1 layer encoder/decoder, hidden state size k, sequence length T, vocabulary size V), can be broken down as follows:

Assumptions for this calculation:

- Cell type: Vanilla RNN.
- Input character embedding dimension: m.
- Encoder: 1 layer, hidden state dimension k.
- Decoder: 1 layer, hidden state dimension k, input character embedding dimension m.
- Sequence length (input and output): T.
- Vocabulary size (source and target): V.
- Computations for embedding lookups are generally excluded as they are memory operations, or $O(1)$ per lookup after setup.
- Matrix-vector multiplication Wx (where W is $p \times q$ and x is $q \times 1$) is $2pq$ FLOPs. Bias additions are p FLOPs. Element-wise activations are p FLOPs.

1. Encoder RNN:
For each of the T timesteps, a vanilla RNN cell performs:

- Input transformation ($W_{xh}x_t$): $2mk$ FLOPs.
- Hidden state transformation ($W_{hh}h_{t-1}$): $2k^2$ FLOPs.
- Summing transformations and adding bias ($W_{xh}x_t + W_{hh}h_{t-1} + b_h$): $2k$ FLOPs (k for the first sum, k for bias addition).
- Activation function (e.g., tanh): k FLOPs.
- Total per encoder step: $2mk + 2k^2 + 3k$ FLOPs.
- Total for encoder over T steps: $T \times (2mk + 2k^2 + 3k)$ FLOPs.

2. Decoder RNN (including output layer):
The decoder also operates for T timesteps. For each step:

- **Decoder RNN Cell** (similar to encoder, taking embedded previous character y_{t-1} of dimension m as input):
  * Input transformation ($W_{yh}y_{t-1}$): $2mk$ FLOPs.
  * Hidden state transformation ($W_{hh}^{dec}h_{t-1}^{dec}$): $2k^2$ FLOPs.
  * Summing transformations and adding bias: $2k$ FLOPs.
  * Activation function: k FLOPs.
  * Total per decoder RNN cell step: $2mk + 2k^2 + 3k$ FLOPs.
- **Output Layer** (projects decoder hidden state to vocabulary size):
  * Linear transformation ($W_{hy}h_t^{dec}$): $2kV$ FLOPs.
  * Adding bias (b_y): V FLOPs.
  * Softmax activation: Approximately $4V$ FLOPs (V exponentials, V-1 additions for sum, V divisions).
  * Total per decoder output layer step: $2kV + 5V$ FLOPs.
- Total per decoder step (RNN cell + output layer): $(2mk + 2k^2 + 3k) + (2kV + 5V)$ FLOPs.
- Total for decoder over T steps: $T \times (2mk + 2k^2 + 3k + 2kV + 5V)$ FLOPs.

Total Computations for the Network:
Summing the encoder and decoder computations:
Total = $T \times (2mk + 2k^2 + 3k) + T \times (2mk + 2k^2 + 3k + 2kV + 5V)$
Total = $T \times (4mk + 4k^2 + 6k + 2kV + 5V)$ FLOPs.

If using LSTM or GRU cells, the number of computations would be higher due to their more complex internal structures. For an LSTM, the $4mk + 4k^2 + 6k$ part would be roughly quadrupled. For a GRU, it would be roughly tripled.

(b) Total number of parameters

The total number of parameters in your network, with the same assumptions, is:

1. Encoder Side:

- Input Embedding Layer (source language characters to m-dim vectors): $V \times m$ parameters.
- Encoder RNN Cell (vanilla):
  * Weight matrix for input (W_{xh}): $m \times k$ parameters.
  * Weight matrix for hidden state (W_{hh}): $k \times k$ parameters.
  * Bias (b_h): k parameters.
  * Total for encoder RNN cell: $mk + k^2 + k$.

2. Decoder Side:

- Input Embedding Layer (target language characters to m-dim vectors): $V \times m$ parameters.
- Decoder RNN Cell (vanilla):
  * Weight matrix for input (W_{yh}): $m \times k$ parameters.
  * Weight matrix for hidden state (W_{hh}^{dec}): $k \times k$ parameters.

  * Bias (b_h^{dec}): k parameters.
  * Total for decoder RNN cell: $mk + k^2 + k$.

- Output Layer (maps hidden state k to vocabulary size V):
  * Weight matrix (W_{hy}): $k \times V$ parameters.
  * Bias (b_y): V parameters.
  * Total for output layer: $kV + V$.

Total Parameters for the Network:
Summing all components:
Total =
$(Vm)_{\text{enc\_embed}} + (mk + k^2 + k)_{\text{enc\_rnn}} + (Vm)_{\text{dec\_embed}} + (mk + k^2 + k)_{\text{dec\_rnn}} + (kV +$
Total = $2Vm + 2(mk + k^2 + k) + kV + V$
Total = $2Vm + 2mk + 2k^2 + 2k + kV + V$ parameters.

If using LSTM cells, each RNN cell's parameters ($mk + k^2 + k$) would be multiplied by approximately 4. For GRU cells, they would be multiplied by approximately 3. This would significantly increase the $2mk + 2k^2 + 2k$ portion of the total parameter count. For example, with LSTM cells, the total would be $2Vm + 8(mk + k^2 + k) + kV + V$.

⁂