Assignment 2

RNN-Based Text Generation (Total: 50 Points)

Objectives:

Implement a simple RNN for text generation to deepen your understanding of how recurrent neural networks can be used to model sequences and generate text based on learned patterns.

1. RNN Model Implementation (10 Points)
   - Implement a basic Recurrent Neural Network model from scratch using PyTorch or TensorFlow. Your model should include an embedding layer, at least one RNN layer, and a fully connected layer for output. Refer to the "Recurrent Neural Networks (RNN)"section of the lectures for guidance on the architecture.
   - Use the "Long Short-Term Memory RNNs (LSTMs)"section as a reference to enhance your model with LSTM cells to improve its ability to capture long-term dependencies in text.
2. Dataset Preparation (10 Points)
   - Select a small text dataset for training your model. This could be a collection of poems, song lyrics, or any text of your choice. Preprocess the data by tokenizing the text into sequences and converting them into numerical format suitable for training your RNN.
3. Training (10 Points)
   - Train your RNN model on the prepared dataset. Aim to optimize the model to predict the next word in a sequence based on the given context. Adjust hyperparameters such as learning rate, number of epochs, and hidden layer dimensions to improve performance.
4. Text Generation (10 Points)
   - Once trained, use your model to generate text. Start with a seed sentence or word, then predict the next word using your model. Append the predicted word to your text and use the updated sequence as the new input to generate the next word. Repeat this process to generate a text of at least 100 words.
5. Analysis (10 Points)
   - Analyze the generated text. Discuss how well your model captures the style and coherence of the chosen dataset. Reflect on the performance of the basic RNN model versus the LSTM-enhanced version. Consider the effects of different hyperparameters on the quality of the generated text.

Seq2Seq Machine Translation with Attention (Total: 50 Points)

Objectives:

Implement a sequence-to-sequence model with attention to perform machine translation between two languages (e.g., English to French). This task will help you understand how conditioned generation works in the context of translating sequences from one domain to another, leveraging the power of LSTMs and attention mechanisms.

1. Model Architecture (10 Points)
    - Construct a seq2seq model comprising an encoder and a decoder. Both the encoder and decoder should use LSTM layers to effectively capture the temporal dependencies in the input and target sequences.
    - Incorporate an attention mechanism between the encoder and decoder to improve the model's ability to focus on relevant parts of the input sequence during translation, as discussed in the "Attention Mechanisms"section of the lectures.
2. Dataset and Preprocessing (10 Points)
    - Choose a bilingual corpus as your dataset (e.g., a collection of English-French sentence pairs). Perform necessary preprocessing steps, including tokenization, converting text to sequences of integers, and padding sequences to a uniform length.
3. Training (10 Points)
    - Train your seq2seq model on the preprocessed dataset. The goal is to minimize the difference between the predicted translation output by the decoder and the actual target sentence in the dataset. Experiment with different hyper-parameters, such as the number of LSTM units, learning rate, and batch size, to optimize your model's performance.
4. Translation and Evaluation (10 Points)
    - Use your trained model to translate a set of sentences from the source language to the target language. Evaluate the quality of your translations using a suitable metric, such as BLEU (Bilingual Evaluation Understudy) score, to quantitatively measure how your translations compare to a set of reference translations.
5. Analysis (10 Points)
    - Discuss the impact of the attention mechanism on the translation quality. Compare the performance of your model with and without attention using both qualitative examples and quantitative metrics. Reflect on how different architectural choices and hyper-parameters affected the outcomes.

Submission Guidelines:

- Submit all code in a ZIP file, including Jupyter Notebooks, and a detailed PDF report with written explanations and visualizations.
- Clearly label each part and question in your submissions.
- Deadline: Feb. 21, 2024

Expectations:

- **Code Quality and Functionality:** Code should be well-organized, commented, and functioning as intended. The use of Python and relevant libraries should demonstrate a good grasp of the tools.
- **Analysis and Interpretation:** Provide a detailed written analysis of your models' performance. This includes insights into the design decisions of your RNN or Seq2Seq models, the impact of LSTM units and attention mechanisms, and how these choices influenced the outcomes of your tasks. Discuss the strengths and limitations of your models, supported by concrete examples from your results (e.g., text generation snippets or translation pairs).
- **Visualization:** Include visual representations to illustrate your models' behavior or performance, such as loss curves over training epochs or attention heat maps for translation tasks. Visualizations should be well-designed, with clear labels and titles, to effectively communicate your findings. Interpret these visualizations in your report, explaining what they reveal about your model's learning process or attention patterns.
- **Adherence to Guidelines:** Submissions should follow the provided guidelines, including format, labeling, and adherence to the deadline.