Implementing an LLM Text Generator on the Persian Wikipedia Dataset

Reyhaneh Saffar 99222064

- **Dataset in details:**
  Initially, the dataset was loaded, and unnecessary line breaks were removed to form a continuous text sequence. Due to the large size of the dataset, a manageable sample was extracted for subsequent processing. To facilitate the model's ability to learn from the text, unique characters from the sample were identified, resulting in 224 unique characters. These characters were then mapped to numerical indices, creating a character-to-index dictionary. This mapping allowed the text to be converted into numerical format, which is necessary for training neural networks. The numerical representation of the text was transformed into a TensorFlow dataset, with each character encoded as its corresponding index. The dataset was further split into batches, and a function was defined to generate input-target pairs for the model. The input data consisted of sequences of characters, and the target data consisted of the same sequences shifted by one character. This preparation step ensured that the model could learn to predict the next character in a sequence based on the preceding characters. The final dataset was batched appropriately for training, providing input and target pairs in a format suitable for the model.

- **Implementation of model:**
  The process involves the creation and training of a neural network model designed to process a dataset of character sequences. This model consists of an embedding layer, a GRU layer, and a dense layer. The embedding layer transforms the input characters into dense vectors of fixed size, the GRU layer processes these vectors to capture sequential dependencies, and the dense layer generates the output predictions. The model is compiled with an optimizer and a specific loss function suitable for categorical data. During training, a callback is used to save the model weights at each epoch. The model is trained over multiple epochs, during which it iteratively adjusts its parameters to minimize the loss. The output of the training process is a series of loss values indicating the model's performance over each epoch. Initially, the loss is relatively high, reflecting the model's initial state of learning. As training progresses, the loss values decrease steadily, showing the model's improving ability to predict the target sequences. By the end of the training, the loss has significantly reduced, indicating a well-trained model ready for further evaluation or deployment.

  In the following new model with the same architecture is then created and its weights are loaded from the saved file. This ensures that the new model has the same learned parameters as the original model. The new model is built with a specific input shape, and its state is reset to prepare it for generating text sequences. A function is defined to predict and generate a sequence of characters based on a given starting text and a specified number of characters to generate. To generate text, the function first converts the starting text into a numerical format using a predefined character-to-index mapping. It then expands the dimensions of the input to fit the model's expected input shape. The text generation loop begins, where the model predicts the next character based on the current input. The predicted character is appended to the generated text, and the input is updated for the next prediction. This process repeats for the specified number of characters, resulting in a sequence of generated text. The output demonstrates the model's performance by displaying the generated text sequence, which follows naturally from the starting text. For instance, starting with the text "سلام", the model generates an additional sequence of characters, resulting in a plausible continuation of the initial input. This indicates the model's capability to understand and generate text based on learned patterns from the dataset.