Text Generation Experiments with RNNs

1. Introduction

Recurrent Neural Networks (RNNs) are a class of neural networks particularly effective for sequential data modeling. In this report, we explore the implementation and performance of character-based and word-based RNN models on text data related to Egypt sourced from Wikipedia. Both models are trained to generate text character by character or word by word, respectively. We delve into the dataset used, preprocessing steps applied, model architectures employed, and discuss the obtained results.

2. Dataset Used

The dataset utilized for training the RNN models consists of textual information retrieved from the Wikipedia page on Egypt. This dataset provides a rich source of information about Egypt, covering various aspects including history, culture, geography, and more.

3. Preprocessing

Character-based RNN:

- Initially, the text is retrieved from the Wikipedia page and preprocessed to remove any non-alphabetic characters such as punctuation and digits.
- The text is then converted to lowercase to ensure uniformity.
- Subsequently, redundant whitespaces and endlines are removed, and the unique characters present in the text are identified to construct a vocabulary.

Word-based RNN:

- Similar to the character-based approach, the text is first cleaned by removing non-alphabetic characters.
- The text is then converted to lowercase and tokenized into words.
- Redundant whitespaces and endlines are removed
- The unique words in the text are identified to form the vocabulary for the word-based RNN model.

4. Architectures Used

First Character-based RNN Architecture:

- The model architecture comprises an embedding layer with an embedding dimension of 256, followed by a single-layer Simple RNN with 1024 units, and a dense layer for character prediction.

- The embedding layer transforms the character indices into dense vectors, facilitating better representation.
- The Simple RNN layer processes sequential data and captures dependencies among characters.
- Finally, the dense layer outputs the probability distribution over the vocabulary.

Second Character-based RNN Architecture:

- The second character-based RNN architecture employs a more complex architecture with multiple layers.
- The specific architecture used is as follows: Embedding Layer with an embedding dimension of 256 -> RNN Layer with 1024 units -> RNN Layer with 1024 units -> Dense Layer with softmax activation.
- The embedding layer serves the same purpose as in the first architecture, providing dense representations of characters.
- Two consecutive RNN layers allow for capturing more intricate patterns and dependencies within the sequential data.
- The final dense layer outputs the probability distribution over the vocabulary, facilitating character prediction.

Word-based RNN:

- The word-based RNN architecture includes an embedding layer with an embedding dimension of 256, a Simple RNN layer with 1024 units, and a dense layer.
- The embedding layer converts word indices into dense vectors.
- The Simple RNN layer processes sequential word data, capturing dependencies.
- The dense layer predicts the next word in the sequence.

5. Results

First Character-based RNN Architecture:

- The first character-based RNN model was trained for 40 epochs using the Adam optimizer and achieved a loss of 1.22.
- While the results were not optimal, the generated text exhibited coherence and structure, albeit with occasional inaccuracies and misspellings.
- A sample from the generated output: "The Pyramids w ad athenthe thed alathethexeded the rid coris d aten n atusthes tet ede rist therod tw mond"

Second Character-based RNN Architecture:

- Similar to the first architecture, this model was trained for 40 epochs using the Adam optimizer and achieved a loss of 1.45.
- Despite the increased complexity of the model, possibly intended to capture longer-term dependencies and nuanced patterns, the performance was pretty bad compared to the first architecture.
- The generated text lacked coherence and often produced gibberish, as indicated by the sample: "The Pyramidsribcyrpuahlesdt pqrhysdoiydepklmhxyxvprowazgqclskgucefxsdgpw[UNK]mrfy".

Word-based RNN:

- The word-based RNN model was trained for 10 epochs using the Adam optimizer and achieved a loss of 0.045.
- Generated text demonstrated a reasonable coherence, with sentences resembling natural language, Although the generated words are not harmonically related to each other.
- A sample from the output: "egyptis a referendum during the s giving a surprise attack on january which was invaded egypt has been competitive in the british protectorate"

6. Conclusion

Character-based and word-based RNNs offer distinct advantages and trade-offs in text generation tasks. While character-based models are adept at capturing fine-grained patterns, word-based models leverage semantic information for more coherent outputs. Further experimentation and fine-tuning of model architectures and training parameters could enhance the performance of both approaches in text generation applications.