Next Word Prediction Using RNN based Models

Ahmad Mustafa

***Abstract*—This research paper presents a novel approach to next-word prediction in natural language processing (NLP) using recurrent neural networks (RNNs) and long short-term memory (LSTM) units. The study aims to develop a predictive model capable of generating contextually relevant word predictions based on input text sequences. Leveraging a diverse dataset sourced from online text repositories, the proposed model undergoes extensive training to learn the underlying patterns and relationships within the text corpus. Through comprehensive experimentation and evaluation, the effectiveness and performance of the model are assessed, demonstrating its potential applications in various NLP tasks.**

I. Introduction

Next-word prediction, a fundamental task in NLP, plays a pivotal role in various applications, including text generation, autocomplete, and machine translation. The ability to accurately anticipate the next word in a sequence is essential for enhancing user experience and improving the efficiency of text-based applications. In this context, the development of robust predictive models capable of generating contextually relevant word predictions has garnered significant attention from researchers and practitioners in the field of artificial intelligence.

In this paper, I propose a data-driven approach to next-word prediction using deep learning techniques, specifically RNNs with LSTM units. My research focuses on harnessing the power of neural networks to capture the intricate dependencies and contextual nuances present in natural language text. By leveraging a diverse and extensive dataset sourced from online text repositories, I aim to train a predictive model that can effectively anticipate the next word in a given text sequence.

II. Method

A. Data Acquisition

The text data utilized in this project was sourced from various online text repositories, including online books, articles, and websites. These sources were chosen to ensure a diverse and comprehensive dataset that captures a wide range of linguistic styles and contexts. The dataset comprises a substantial volume of text, encompassing thousands of documents and spanning multiple genres and topics.

B. Data PreProcessing

1) Tokenization: The raw text data was tokenized using TensorFlow's Tokenizer class, which facilitated the segmentation of the text into individual words or tokens. This process involved converting the text into a sequence of integers, where each word in the vocabulary was assigned a unique index.

2) Sequence Generation : Input-output pairs, also known as sequences, were generated from the tokenized text data using a sliding window approach. This involved sliding a window of fixed size over the tokenized text, creating sequences of varying lengths. Each sequence comprised a context (input) and the subsequent word (output), providing the model with contextual information for predicting the next word in a sequence

3) Padding: To ensure uniform length across sequences, padding was applied using TensorFlow's pad_sequences function. This involved appending zeros to sequences shorter than the maximum sequence length, thereby standardizing the length of all sequences. Standardizing the sequence length is crucial for efficient batch processing during model training and inference

III. Model Architecture

A. Overview

In the model architecture section, provide an overview of the recurrent neural network (RNN) and long short-term memory (LSTM) networks used in the project. Highlight their significance in natural language processing tasks, particularly in capturing temporal dependencies and long-range contextual information in sequential data.

B. Model Description

- **Embedding Layer**: This layer converts input word indices into dense vectors of fixed size, allowing the model to learn semantic representations of words.
- **LSTM Layer**: The LSTM layer processes sequential input data and captures temporal dependencies through its gated mechanism, which helps mitigate the vanishing gradient problem.
- **Dense Layer**: The dense layer with softmax activation is used for multiclass classification, predicting the probability distribution over the vocabulary of words.

Model Architecture is depicted in Fig.1 and Fig.2

IV. Training and Performance

A. Model Training

I conducted the training of the next-word prediction model using the acquired dataset and the designed model architecture. The training process involved optimizing the model parameters to minimize the categorical cross-entropy loss function and

Identify applicable funding agency here. If none, delete this.

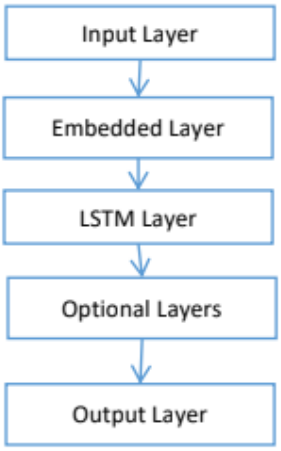

Fig. 1. Architecture Depiction

Layer (type)	Output Shape
embedding_20 (Embedding)	(None, 15, 100)
lstm_19 (LSTM)	(None, 150)
dense_19 (Dense)	(None, 413)

Fig. 2. Model Summary

improve predictive accuracy. The following steps outline the training procedure:

1) Data Preparation: : The dataset was preprocessed to convert text data into a format suitable for training. This involved tokenization, sequence generation, and one-hot encoding of the target words.

2) Model Compilation: The LSTM-based model was compiled using the Adam optimizer and categorical cross-entropy loss function. The model was configured to optimize the loss function during training.

3) Model Training: The compiled model was trained on the preprocessed dataset for a specified number of epochs. During training, the model learned to predict the next word in a sequence based on the input context.

4) Monitoring Training Progress: The training progress was monitored by recording the training loss and accuracy metrics after each epoch. This facilitated the analysis of model performance and convergence over time.

B. Model Performance Evaluation:

Upon completion of training, the performance of the trained model was evaluated using a separate test dataset. The evaluation process aimed to assess the model's ability to accurately predict the next word in a given text sequence. The following metrics were utilized to evaluate model performance:

1) ***Loss Curve****:* A line plot depicting the categorical cross-entropy loss over successive epochs was generated to visualize the training progress. This graph provides insights into the model's convergence and training stability.

2) ***Accuracy Metric****:* The accuracy of the model was calculated based on its predictions compared to the ground truth

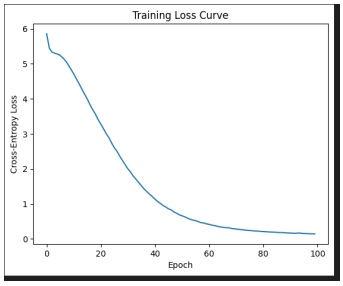

Fig. 3. Cross Entropy Loss Graph

labels in the test dataset. This metric quantifies the proportion of correct predictions made by the model.

Training Loss Curve: As shown in Fig. 3

.

REFERENCES

[1] https://www.geeksforgeeks.org/next-word-prediction-with-deep-learning-in-nlp/
[2] https://medium.com/@ilaslanduzgun/next-word-prediction-using-lstm-with-tensorflow-e2a8f63b613c
[3] https://thecleverprogrammer.com/2023/07/17/next-word-prediction-model-using-python/
[4] K. Elissa, "Title of paper if known," unpublished.
[5] Rianti, Afika and Widodo, Suprih and Ayuningtyas, Atikah and Hermawan, Bima. (2022). NEXT WORD PREDICTION USING LSTM. Journal of Information Technology and Its Utilization. 5. 10.56873/jitu.5.1.4748.