EE 769 Project: Next Word Prediction

1stHarshwardhan Kiran Waghchoure
Department of Electrical Engineering
Indian Institute of technology Bombay
Mumbai, India
e-mail: harshkw2@gmail.com

2nd Siddhant Batra
Department of Electrical Engineering
Indian Institute of technology Bombay
Mumbai, India
e-mail: batrasiddhant10@gmail.com

3rd Ashish Kedari
Department of Electrical Engineering
Indian Institute of technology Bombay
Mumbai, India
e-mail: akedari1052@gmail.com

4th Aareen Aher
Environment Science and Engineering Department
Indian Institute of technology Bombay
Mumbai, India
e-mail: 20d180004@iitb.ac.in

1. Introduction

Next word prediction is a crucial task in natural language processing (NLP). The aim of this project is to build a next word prediction model using Long Short-Term Memory (LSTM) and evaluate its performance. We also used pre-trained model of GPT-1. The model is trained on a large corpus of text data and the goal is to predict the next word in a given sentence. This report provides an overview of the project and presents the results and evaluation of the model.

2. Background

Next Word Prediction is a well-known problem in natural language processing. It involves the prediction of the next word in a given sequence of text. This task has applications useful such as speech recognition, text generation, and machine translation.

Recurrent Neural Networks (RNNs) are commonly used for next word prediction. However, these have limitations when dealing with long sequences, such as memory loss and vanishing gradients. Long Short-Term Memory (LSTM) models were introduced to overcome these challenges.

LSTM models are a type of RNN that can learn long-term dependencies in sequences. They use a gated mechanism to selectively forget or remember information from previous time steps, enabling them to maintain a stable memory over long sequences.

3. Appraoch

The objective of this project is to develop a Next Word Prediction model using LSTM. We will train the model on a large corpus of text data and evaluate its performance on a separate test set.

To begin, we will preprocess the text data and convert it into a numerical format suitable for feeding into the LSTM model. We will then build and train various LSTM models with different configurations, such as varying the number of LSTM layers, hidden units, and sequence lengths.

To assess the model's performance, we will use perplexity, which measures the average surprise of the model in predicting each word in the test set. Lower perplexity values indicate better model performance.

We will also compare our LSTM models' performance with other models, such as Markov models and pre-trained language models such as GPT-1 and ALBERT. We will use perplexity to evaluate the performance of these models and compare their relative strengths and weaknesses.

4. Methodology

The first step in building the next word prediction model is to pre-process the text data. The dataset used for this project is taken from kaggle.The text data is preprocessed by tokenizing the words and removing any special characters or numbers. The preprocessed text data is then split into training and testing datasets. We are using the LSTM model,that we trained and a pre-trained GPT-1 model. Then we compared the reulst of the same using training loss.

4.1. GPT-1

The GPT-1 (Generative Pre-trained Transformer) model is a type of neural network architecture that was developed by OpenAI in 2018. It is used for generating high-quality text based on input prompts and is widely regarded as a significant breakthrough in the field of natural language processing.

The GPT-1 model is made up of 12 transformer layers and has a total of 117 million parameters. It was trained on a massive corpus of text data, including the entire English Wikipedia, using an unsupervised pre-training approach.

The pre-training process of the GPT-1 model involves two stages. In the first stage, the model is trained to predict the next word in a given sequence of text. This is done by masking out a random subset of the input tokens and training the model to predict the missing tokens based on the context of the surrounding words.

In the second stage, the model is fine-tuned on a specific downstream task, such as sentiment analysis or language translation. During fine-tuning, the parameters of the model are adjusted to optimize performance on the specific task, using a supervised learning approach.

The transformer-based architecture of the GPT-1 model is particularly well-suited for sequence-to-sequence tasks, such as language modeling. It overcomes the limitations of previous recurrent neural network (RNN) models by using an attention mechanism that allows the model to focus on different parts of the input sequence depending on their relevance to the current output. This makes the model more efficient and effective at learning long-term dependencies in sequences.

The attention mechanism used in the GPT-1 model weighs the importance of each input token based on its context within the sequence. This allows the model to generate high-quality text that is coherent and contextually relevant.

4.2. LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN). It is designed to overcome the vanishing gradient problem commonly encountered in traditional RNNs. The error function become very small as they propagate through the network, making it difficult to update the weights of the network. This may result in the problem of vanishing gradients.

LSTM introduces a memory cell, that allows the network to selectively remember or forget previous inputs as it processes new inputs. The memory cell is composed of three gates: the input gate, the forget gate, and the output gate. These gates control the flow of information into and out of the memory cell, allowing the network to selectively retain or discard information.

The input gate determines which information from the new input should be stored in the memory cell. It takes as input the current input and the previous hidden state, and produces a value of either 0 or 1 for each element in the memory cell. If the value is 0, the corresponding element is discarded, and if it is 1, the corresponding element is retained.

Similarly, the forget gate determines which information from the previous memory cell should be retained. It takes as input the current input and the previous hidden state and produces a value of either 0 or 1 for each element in the memory cell. If the value is 0, the corresponding element is discarded, and if it is 1, the corresponding element is retained.

Finally, the output gate determines which information from the memory cell should be output as the current hidden state. It takes as input the current input and the previous hidden state and produces a value of either 0 or 1 for each element in the memory cell. The output gate then multiplies the values in the memory cell by the output gate values to produce the current hidden state.

The LSTM network can be trained using backpropagation through time (BPTT), a variant of backpropagation used for training RNNs. BPTT involves unfolding the network over time and computing the gradients of the error function with respect to the weights at each time step. These gradients are then used to update the weights of the network using an optimization algorithm such as stochastic gradient descent (SGD).

In the context of next word prediction, the LSTM network takes as input a sequence of words and outputs the probability distribution over the vocabulary of words for the next word in the sequence. The network is trained to minimize the categorical cross-entropy loss between the predicted and actual probability distributions.

Overall, LSTM is a powerful technique for modeling sequential data and has been applied successfully in various NLP tasks, including next word prediction, machine translation, and sentiment analysis.

5. Results

Model Name	**Train Loss**
LSTM	6.31
GPT-1	4.77

TABLE 1. TRAINING LOSS FOR LSTM AND GPT-1 MODEL

The above table shows the performance comparison of the two models for the task of next word prediction. The LSTM model with sequence length 3, as compared with the GPT-1 model, gives a higher training loss.

6. Challenges faced

Developing an effective Next Word Prediction model using LSTM presented several challenges. One of the primary difficulties was training the models. Due to the size of the text corpus, training the models took a significant amount of time, and tuning hyper-parameters required careful consideration and experimentation.

Processing and cleaning the data was another significant challenge. The text corpus was large and required extensive pre-processing to be suitable for use in the LSTM models. Furthermore, training the models required a substantial amount of computing power.

Selecting the appropriate hyper-parameters for the LSTM models was another challenge. The number of LSTM layers, hidden units, and sequence lengths all played a vital role in determining the model's performance. Fine-tuning these parameters required careful consideration and testing.

Over-fitting was another issue encountered during the training process. To prevent over-fitting, various techniques such as early stopping and dropout were used. However, achieving optimal model performance while avoiding over-fitting was still challenging.

Despite these challenges, we were able to develop effective LSTM models for the Next Word Prediction task. By carefully tuning the hyper-parameters, employing training techniques, and utilizing appropriate data pre-processing methods, we were able to overcome these difficulties and develop models that achieved good performance on the test set.

References

[1] https://www.kaggle.com/datasets/rajeevmore/next-word-prediction-dataset

[2] https://builtin.com/artificial-intelligence/transformer-neural-network

[3] https://www.analyticsvidhya.com/blog/2021/08/predict-the-next-word-of-your-text-using-long-short-term-memory-lstm/

[4] https://medium.com/mlearning-ai/an-illustration-of-next-word-prediction-with-state-of-the-art-network-architectures-like-bert-gpt-c0af02921f17

[5] https://pub.towardsai.net/building-a-lstm-from-scratch-in-python-1dedd89de8fe