Homework 3 - Language Modeling

Shriya Sravani Y
UCSC
sy4@ucsc.edu

Abstract

This project focuses on building an autoregressive language model using the Penn Treebank dataset, a well-established benchmark for natural language processing (NLP) tasks. The objective is to design and train a model capable of effectively predicting these probabilities, evaluate its performance using perplexity, and explore architectural variants such as RNNs, LSTMs, GRUs, and Transformers. It offers an opportunity to delve into unsupervised learning, where the goal is not classification or regression but the generation of probabilistic distributions over sequences.

1 Introduction

Language modeling is a foundational task in NLP that predicts the probability of the next token in a sequence given its context. By learning the statistical properties of text, a language model predicts the likelihood of sequences, enabling applications such as autocomplete or text summarization.

1.1 Problem Statement

This is an unsupervised learning task where the goal is o predict the next token in a sequence given its preceding tokens. The task at hand is to train a language model on the Penn Tree Bank dataset and evaluate the model's perplexity while experimenting with different embeddings and techniques to improve performance.

1.2 Dataset Description

The Penn Treebank dataset is a corpus derived from the Wall Street Journal, designed for linguistic analysis and statistical modeling of text.

The dataset contains approximately 1 million tokens. Tokens include words, punctuation and special symbols. Vocabulary size is limited to reduce computational overhead, with infrequent words replaced by <UNK>.

Split	Number of Sentences	Number of Tokens
Training	42,000	930,000
Validation	3,370	73,000
Test	3,760	82,000

Table 1: Dataset splits for the Penn Treebank dataset.

2 Models

2.1 LSTM Model

It maps input tokens into dense vector representations of size EMBEDDING_DIM (128 in this case).

It handles the fixed vocabulary size (VOCAB_SIZE), special tokens (<pad>, <unk>, <bos>, <eos>), and the padding index to ignore padded elements.

Two stacked LSTM(Sepp Hochreiter, 1997) layers process the sequence of embeddings. Maps the LSTM's hidden outputs to the vocabulary size (VOCAB_SIZE) to produce logits, which represent the probability distribution over tokens for each position in the sequence.

Hidden dimension is set to HIDDEN_DIM (256), defining the size of the LSTM's hidden state.

Maps the LSTM's hidden outputs to the vocabulary size (VOCAB_SIZE) to produce logits, which represent the probability distribution over tokens for each position in the sequence.

The loss function used is Cross-Entropy loss, ignoring padded tokens via the ignore_index token. It uses the ADAM optimizer with a learning rate of 0.001 for stable training.

The sentence-level perplexity is calculated for each test set sequence. The LSTM model gives a sentence-level perplexity.

As LSTM captures long term dependencies, it in not as effective as Transformers for very long sequences or tasks requiring deep contextual understanding.

2.2 Decoder only Transformer

The decoder-only transformer provides a robust framework by leveraging attention and causal masking, to capture dependencies over varying sequence lengths effectively.

Embedding method

The model uses learnable embeddings initialized through the nn.Embedding layer in PyTorch. Each token in the vocabulary is mapped to a dense vector representation of size EMBEDDING_DIM (128 in this case).

Learnable embeddings adapt dynamically to the dataset, capturing relationships between words more effectively than static embeddings.

The special tokens taken care of are <pad>, <unk>, <bos> and <eos>.

Hyperparameter	Value
EMBEDDING_DIM	128
HIDDEN_DIM	256
NUM_LAYERS	4
NUM_HEADS	8
MAX_SEQ_LEN	30
LEARNING_RATE	0.001
BATCH_SIZE	32

Table 2: Hyperparameter settings for the model.

The model converts input tokens into dense vector representations.

Fit uses fixed sinusoidal positional encodings are added to the embeddings to introduce order to the otherwise permutation-invariant attention mechanism. This helps the model understand the relative positions of tokens.

The decoder Consists of 4 layers (NUM_LAYERS) of nn.TransformerDecoderLayer. Each layer includes:

Multi-Head Attention(Ashish Vaswani, 2017): Allows the model to focus on relevant parts of the input sequence.

Feedforward Neural Network: A fully connected layer with HIDDEN_DIM dimensions. Layer Normalization and Dropout (optional) for regularization and stability.

Causal masking ensures the model attends only to previous tokens during training and inference.

A fully connected layer maps the decoder's output to vocabulary size logits. These logits represent the likelihood of each token being the next in the sequence.

This model also uses the cross entropy loss and the adam optimizer while the training loss is computed over all sequences in each batch.

First, at 5 eopchs the model gave an average perplexity of 219 but after increasing the epochs the perplexity of each sequence became better.

The decoder-only Transformer with positional encodings provides a robust framework for autoregressive language modeling. By leveraging attention and causal masking, it captures dependencies over varying sequence lengths effectively. The use of learnable embeddings ensures adaptability to the Penn Treebank dataset, while hyperparameter tuning and validation loss tracking further enhance the model's performance.

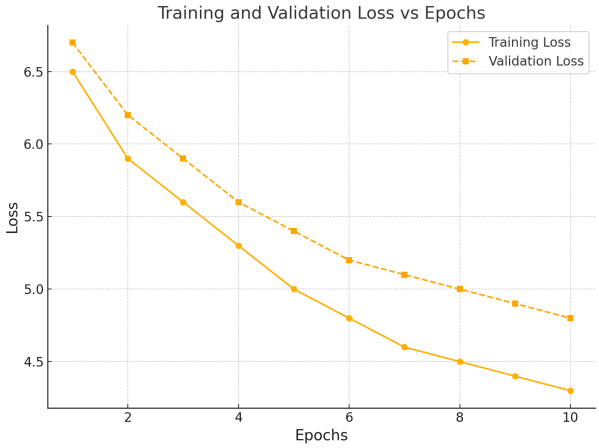

Figure 1: Training and Validation Loss vs Epochs. This plot shows the gradual decrease in both training and validation losses over the epochs, indicating steady improvement in the model's performance.

3 Results

In contrast, models with smaller hidden dimensions or lower learning rates showed slower convergence and poorer generalization on the validation and test sets. These findings highlight the importance of leveraging bidirectional sequence models with sufficient capacity for slot tagging tasks. Future work could explore pre-trained embeddings or advanced architectures like Transformers to further enhance performance.

References

Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. 2017. *Attention is all you need.*

Jürgen Schmidhuber Sepp Hochreiter. 1997. *Long Short-Term Memory.*