אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Introduction to Deep Learning (364-2-1071)

Final Project Report

Fine-Tuning Transformer-Based Language Models for Contextual Story Understanding

Adi Yogev Golbari and Or Chen

Abstract

This project focuses on enhancing transformer-based language models, specifically BERT (Devlin et al., 2018), to improve their ability to understand story endings. We adapted a pre-trained model to work with specific datasets, enabling it to better evaluate story coherence and select the most suitable ending. The original code, sourced from an existing repository, utilized a simplified version of GPT (Radford et al., 2018) with a basic tokenizer and achieved an accuracy of ~86%. In our work, we experimented with different configurations, adjusted hyperparameters, and explored novel training techniques. Our goal was to demonstrate how fine-tuning these models can significantly enhance their text understanding and analytical capabilities. Ultimately, we achieved a substantial improvement in performance, reaching an accuracy of approximately 90%, primarily by transitioning to a more advanced model and incorporating additional training data (Li et al., 2019). **GitHub Repository**

1. Introduction

1.1 Background

Transformer-based models like GPT and BERT have revolutionized natural language processing (NLP) by achieving state-of-the-art results in tasks like text generation and question answering (Radford et al., 2018; Devlin et al., 2018). These models are pre-trained on large datasets, learning language patterns and relationships, which allows them to handle many tasks without requiring extensive labeled data or custom architectures (Radford et al., 2018).

Pre-trained models can be fine-tuned on smaller, specific datasets to excel in specialized tasks, such as understanding stories and evaluating their flow (Li et al., 2019). However, adapting these models to specific tasks can be challenging and requires careful adjustments, such as optimizing tokenization and training settings (Devlin et al., 2018).

This project focuses on improving the ability to understand story contexts and evaluate narrative coherence effectively.

1.2 Objectives

The primary objectives of this project are:

1. To fine-tune a transformer-based language model for the task of evaluating narrative coherence in contextual stories.
2. To investigate the impact of architectural and hyperparameter modifications on model performance.

1.3 Approach

This project builds upon a modify fork of the OpenAI transformer-based fine-tuning framework.
The steps taken include:

- □ **A/B Testing Methodology:** Changes were introduced iteratively, one at a time, to isolate their impact on model performance.
- □ **Hyperparameter Optimization:** Experimented with optimizer, learning rates, batch sizes, and dropout rates to find the optimal settings.
- □ **Model Transition:** Switched from a simplified GPT architecture to a pre-trained BERT model for improved performance.
- □ **Dataset Integration:** Incorporated additional data to the original dataset to enhance training.
- □ **Evaluation:** Assessed performance using validation and test accuracy metrics to measure the effectiveness of each modification.

2. Related Work

2.1 Domain Overview

Language models have changed how we solve natural language processing (NLP) tasks by making it possible to generate and understand text with near-human fluency (Radford et al., 2018; Devlin et al., 2018). Pre-trained transformer models like GPT and BERT are now essential tools in modern NLP due to their ability to leverage attention mechanisms for understanding complex relationships in text (Vaswani et al., 2017). However, while these models work well for general tasks, adapting them to specific challenges—like understanding and evaluating story coherence—requires careful fine-tuning to handle the unique details of the task (Li et al., 2019)

2.2 Relevant Techniques and Models

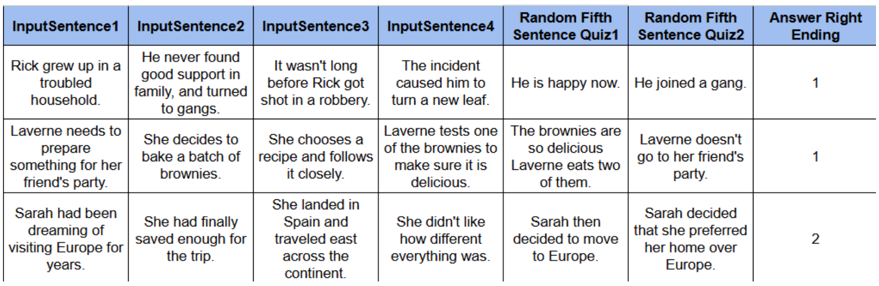
Several methods have been developed to fine-tune language models for specific tasks.
These include:

- □ **Tokenization Techniques**: Using the right tokenization methods helps represent text more effectively, improving the model's performance and ability to generalize.
- □ **Model Architectures**: Transformer-based models like GPT and BERT are powerful because their attention mechanisms allow them to understand long-term relationships in text.
- □ **Training Strategies**: Fine-tuning approaches, such as transfer learning and pretraining on task-specific data, have been shown to significantly boost performance for specific tasks.
- □ **Story Cloze Test (SCT)**: evaluates a model's ability to choose the correct ending for a story from two options. This task demands deep language understanding and commonsense reasoning. Early models struggled to match human performance (~95%), but transformer-based approaches like GPT and BERT have significantly improved results, showcasing the effectiveness of pretraining and fine-tuning frameworks.

3. Materials and Methods

3.1 Data Acquisition

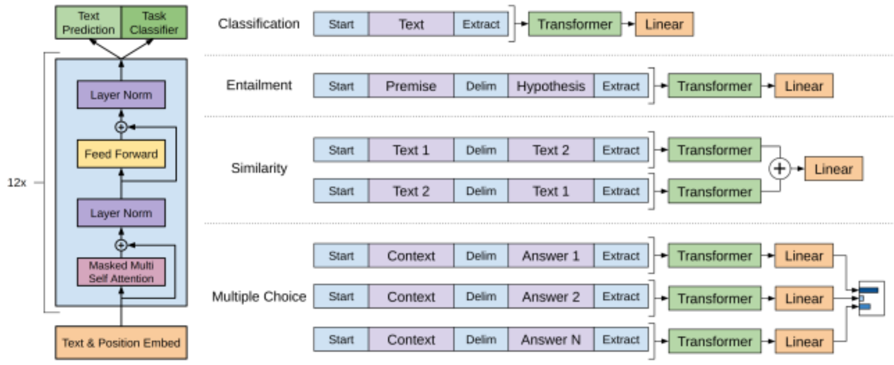
The dataset used for this project was sourced from the**ROCStories dataset**, which includes 1,871 stories. To further enhance the model's training, we added 7,500 more stories generated from Wikipedia, creating a richer and more diverse dataset to boost performance.



InputSentence1	InputSentence2	InputSentence3	InputSentence4	Random Fifth Sentence Quiz1	Random Fifth Sentence Quiz2	Answer Right Ending
Rick grew up in a troubled household.	He never found good support in family, and turned to gangs.	It wasn't long before Rick got shot in a robbery.	The incident caused him to turn a new leaf.	He is happy now.	He joined a gang.	1
Laverne needs to prepare something for her friend's party.	She decides to bake a batch of brownies.	She chooses a recipe and follows it closely.	Laverne tests one of the brownies to make sure it is delicious.	The brownies are so delicious Laverne eats two of them.	Laverne doesn't go to her friend's party.	1
Sarah had been dreaming of visiting Europe for years.	She had finally saved enough for the trip.	She landed in Spain and traveled east across the continent.	She didn't like how different everything was.	Sarah then decided to move to Europe.	Sarah decided that she preferred her home over Europe.	2

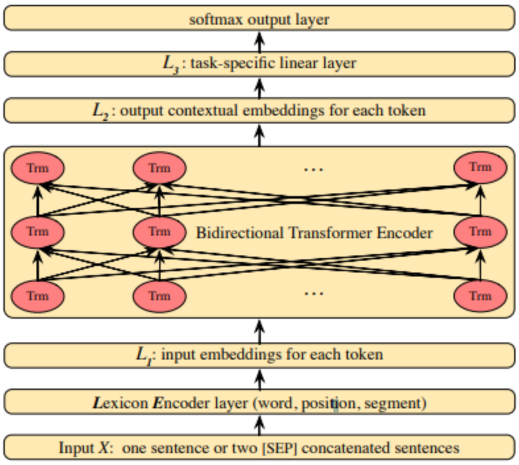
3.2 Model Architectures

Model 1 (Original):



Model 1 uses a simplified GPT-based architecture with preloaded weights, featuring multiple linear transformations and a dense output layer designed for binary classification. Like the figure, it relies on a Transformer backbone for contextual understanding and can handle tasks with multiple input components, such as story context and potential endings. However, it differs from the figure by using a basic tokenizer and lacking additional components like classification heads for more complex tasks or attention visualization tools.

Model 2 (Modify):


softmax output layer
L_3: task-specific linear layer
L_2: output contextual embeddings for each token
Trm
Trm
...
Trm
Trm
Trm
Bidirectional Transformer Encoder
Trm
Trm
Trm
...
Trm
L_1: input embeddings for each token
Lexicon Encoder layer (word, position, segment)
Input X: one sentence or two [SEP] concatenated sentences

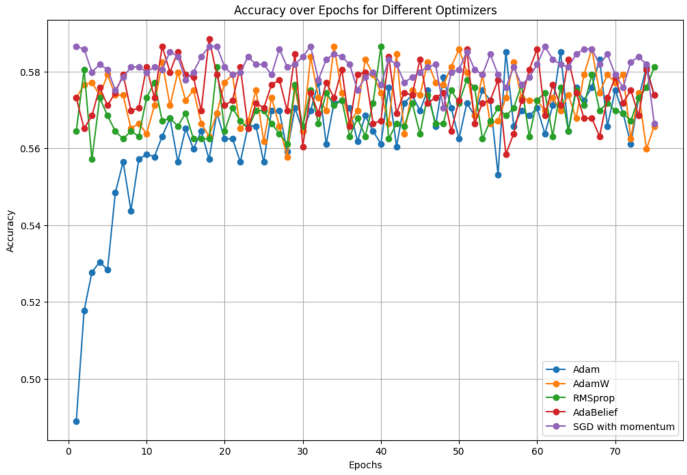
Model 2 uses a BERT-based architecture with a pre-trained transformer backbone, specifically designed for multiple-choice classification. Like the figure, it includes input embeddings for tokens, positions, and segments, a bidirectional transformer encoder for contextual understanding, and a task-specific softmax output layer for selecting the correct story ending. However, it differs slightly from the figure by explicitly handling multiple candidate endings as separate input branches, tailored for the story ending prediction task.

3. Experiments and Results

Disclaimer: *after making several adjustments to run the code locally, we discovered that the accuracy dropped to 50%. We decided to use this as our baseline for all future decision-making and improvements.*

3.1 Experiment 1 - Optimizer Selection

We began by selecting the optimizer with the highest potential for improving model performance. To do this, we evaluated several optimizers, including Adam, AdamW, RMSprop, AdaBelief, and SGD with momentum. Each optimizer was tested to determine its effectiveness in achieving stable and consistent accuracy across epochs.

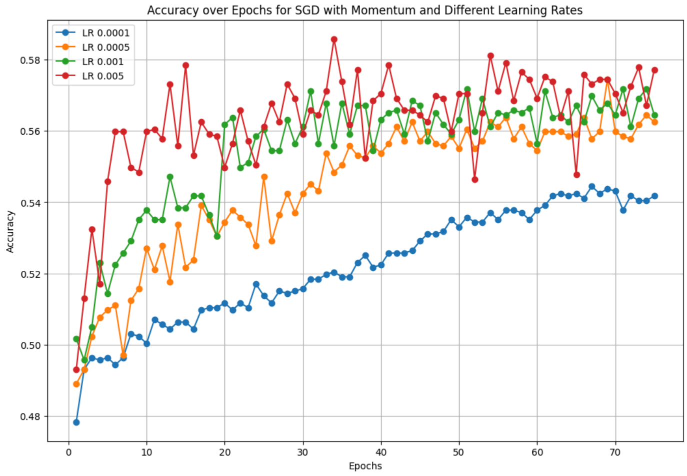


We chose **SGD with Momentum** because it provides stable results, avoids big accuracy swings, and performs as well as other optimizers. Momentum helps smooth out the learning process, making it more reliable and less likely to overfit. Unlike Adam, it takes a steady and controlled approach, which suits our setup well.

Next, we fine-tuned **SGD with Momentum** using an A/B testing approach. We adjusted one parameter at a time, keeping everything else constant, and chose the value that gave the highest accuracy before moving on to test the next parameter. This step-by-step method ensured we found the best settings for the model.

3.2 Experiment 2 - Learning Rate Selection

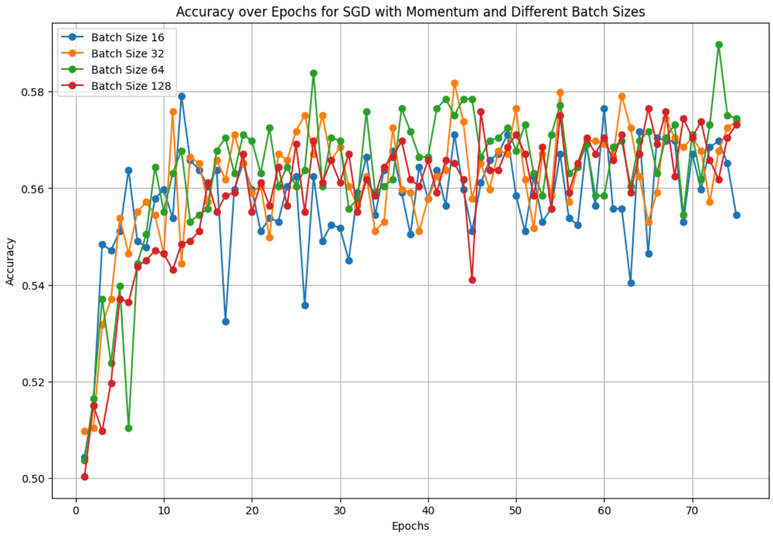
We started by fine-tuning the **Learning Rate** parameter, testing four levels: [0.0001, 0.0005, 0.001, 0.005]. This allowed us to observe how different learning rates impacted the model's performance and identify the optimal value for achieving the best accuracy.



Based on the graph, **LR = 0.005** is the best choice. It shows the fastest convergence to high accuracy and maintains stable performance throughout the epochs, outperforming the other learning rates in terms of both speed and overall accuracy.

3.3 Experiment 3 - Batch Size Selection

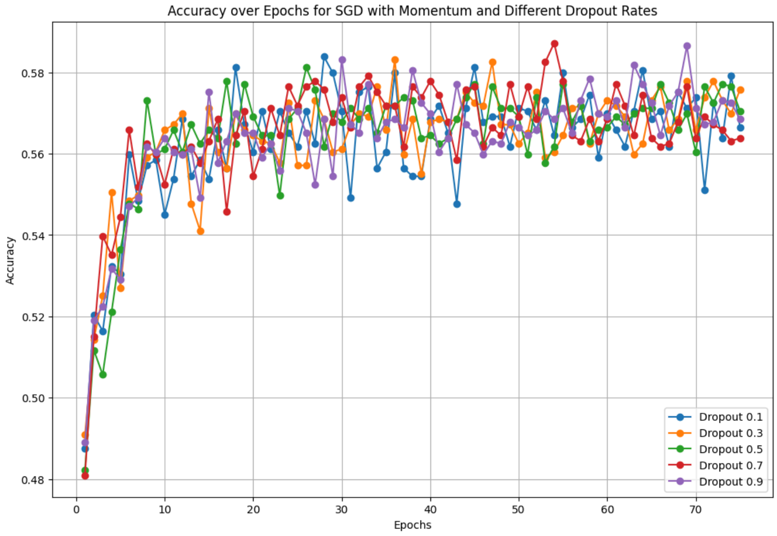
Next, we fine-tuned the **Batch Size** using the best learning rate from the previous step (LR = 0.005). We tested four levels: [16, 32, 64, 128]. This helped us determine the batch size that delivers the highest accuracy while maintaining stable performance over epochs.



Looking at the graph, **Batch Size 64** appears to be the best choice. It provides stable performance with smoother accuracy trends, avoiding the larger fluctuations seen with smaller batch sizes like 16. However, the accuracy remains fairly consistent across the tested batch sizes, showing no significant improvement. This suggests that batch size may not have a major impact on the model's performance in this setup. Nevertheless, we will continue experimenting with other parameters while keeping the batch size at 64 for stability and computational efficiency.

3.4 Experiment 4 - Dropout Rate Selection

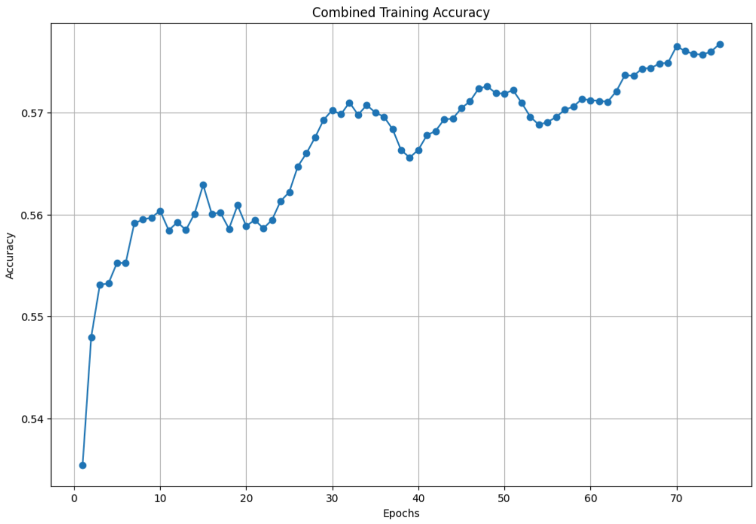
Next, we evaluated different **Dropout Rates**, testing the values [0.1, 0.3, 0.5, 0.7, 0.9].



Based on the graph, We choose a **dropout rate of 0.5.**

3.5 Experiment 5 - Enhancing Training with Additional Dataset

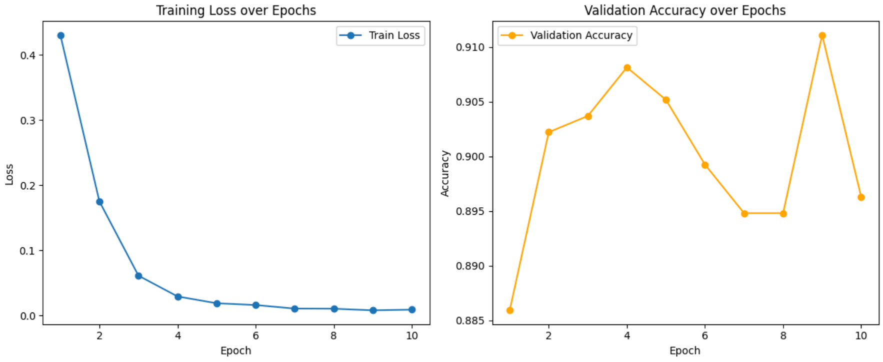
We noticed that tuning the hyperparameters didn't improve the accuracy much, so we decided to try something different. We added an extra dataset to help improve the model. We used the IMDb dataset, which has movie reviews labeled with positive or negative sentiment, to give the model more variety and context. This combined training gave us the following results:



Looking at the results, the combined training shows a steady improvement in accuracy over the epochs, but the overall gain is relatively modest. The curve indicates some fluctuation early on, but it stabilizes as the training progresses, suggesting that the model is learning from the combined datasets. However, the final accuracy improvement isn't very significant, which could indicate that the IMDb dataset, while adding variety, doesn't align closely enough with the sentence completion task to provide a substantial boost. It may be worth exploring other datasets more aligned with sentence structure and completion to achieve better results.

3.6 Experiment 6 - Adopting BERT and Integrating New Dataset

Although the IMDB dataset showed some improvement, it remained far from our goal of achieving an accuracy above 86%. We realized that the IMDB dataset, focused on sentiment labeling, was likely too different from our task of sentence completion. To address this, we extracted 7,500 sentences from a Wikipedia dataset and created a new dataset in the same format as the original, integrating it into our training process alongside the existing data. Additionally, we transitioned to using the BERT model due to its proven ability to capture contextual relationships and handle complex language tasks, making it particularly well-suited for understanding narrative coherence. Despite resource constraints that limited training to 10 epochs, the changes proved impactful, with the final accuracy exceeding 90%.



□ **Training Loss (Left Graph):**
The training loss decreases steadily over the epochs, showing that the model is learning well from the training data. Around the 4th epoch, the training loss stops improving much, meaning further training adds little value.

□ **Validation Accuracy (Right Graph):**
The validation accuracy improves quickly in the first few epochs and reaches its highest point around the 4th epoch. After that, it starts to fluctuate and even drops slightly, which could indicate that the model is overfitting the training data. The model reaches peak performance at 4 epochs, which is typical for fine-tuning pre-trained models like BERT. Beyond this point, further training shows signs of overfitting, with diminishing validation accuracy.
The highest validation accuracy happens around the 4th epoch, making it the best point to stop training and avoid overfitting.

4. Discussion and Conclusions

In this project, we improved story understanding by fine-tuning transformer models, moving from a simplified GPT model to the pre-trained BERT. This transition significantly increased the model's accuracy from an initial baseline of 50% and the original model's 86% to over 90%.

Key Insights:

□ **Better Model, Better Results**: Switching to BERT, known for its ability to understand context, was critical for achieving higher accuracy.

□ **Importance of Relevant Data**: Adding 7,500 Wikipedia sentences in the same format as the original dataset helped the model learn better. In contrast, the IMDb dataset, focused on sentiment, offered limited benefits for this task.

□ **Effective Tuning**: Optimizing settings like learning rate (0.005), batch size (64), and dropout rate (0.5) boosted performance. However, training beyond 4 epochs showed signs of overfitting, making early stopping essential.

Conclusions:

This project shows that fine-tuning pre-trained models like BERT with task-specific data and careful optimization can greatly improve performance. Even with limited training resources, meaningful gains are achievable by focusing on the right data and model.

Future Work:

□ Add more datasets tailored to story understanding.

□ Test advanced models like RoBERTa or T5.

□ Use tools to explain the model’s decisions.

These steps can further enhance performance and bring us closer to human-like story comprehension(~95%).

References

1. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. OpenAI. Retrieved from https://openai.com/research/language-understanding-generative-pre-training

2. Li, Z., Ding, X., & Liu, T. (2019). *Story ending prediction by transferable BERT*. Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology.

3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding*. *arXiv preprint* arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. *arXiv preprint* arXiv:1706.03762. Retrieved from https://arxiv.org/abs/1706.03762