DI725 - Assignment 1: Training and Fine-tuning GPT Models for Sentiment Classification

İbrahim Ethem Deveci
Cognitive Sciences
METU Informatics Institute
Ankara, Turkey
ethem.deveci@metu.edu.tr

***Abstract*—This report outlines the work for Assignment 1 of the *Transformers and Attention-Based Deep Networks* course, focusing on enhancing familiarity with transformer architectures. It covers the preprocessing of a dataset, modifications to GPT-based models for sentiment classification, and the results of training NanoGPT from scratch and fine-tuning GPT-2.**

GitHub: GitHub repository of the assignment.

WandB: WandB profile

***Index Terms*—transformers, sentiment classification, GPT-2, NanoGPT, fine-tuning, training**

I. Introduction

This assignment explores the application of transformer-based architectures to sentiment classification through two approaches: (1) training a NanoGPT model from scratch and (2) fine-tuning a pretrained GPT-2 model. Section II covers the dataset and preprocessing pipeline. Section III details the architectural adjustments, training procedures, and integration of classification components. Sections IV and V present the evaluation metrics and results comparing both approaches. Finally, Section VI discusses their relative strengths and limitations.

II. Dataset

The dataset used in this assignment consists of customer service conversations with two primary columns: `conversation` and `customer_sentiment`. The dataset includes a training set with 970 entries and a test set with 30 entries. In this section, the observations obtained from the expalaratory data analysis and preprocessing strategies are presented.

A. Explaratory Data Analysis

The explaratory data analysis revealed that each dataset contains 11 columns, with no missing or duplicated values, but the training set exhibited a severe class imbalance, with positive sentiment comprising only 0.02% of instances—potentially impairing model performance. A Chi-square test revealed a significant correlation between issue_area and sentiment, while no such relationship was found for product_category or issue_complexity. The conversation and customer_sentiment columns were selected for modeling, and the training set was split into training and validation subsets before preprocessing.

B. Preprocessing

This study employs a targeted preprocessing strategy to clean and structure the dataset by isolating customer dialogue and eliminating extraneous content. Key steps include lowercasing to standardize tokens, removing text within brackets and parentheses, and filtering out agent responses and non-conversational elements. This focused approach enhances the relevance of the input for sentiment analysis, ensures compatibility with models like NanoGPT and GPT-2, and improves training efficiency by reducing token length and emphasizing customer emotion.

III. Modeling

A. Modifications

Architecture Modifications: The original language modeling head was replaced with a linear classifier head that maps the final token representation to sentiment classification logits. The number of output classes was set to three, and cross-entropy loss was used to optimize the model for classification. To ensure compatibility, a pre-trained GPT-2 model for sequence classification was loaded, avoiding key mismatches during weight initialization.

Training Pipeline Modifications: The training pipeline was modified to support both NanoGPT and GPT-2, with updated dataloaders reflecting their distinct tokenization methods. The get_batch function was adapted accordingly. Wandb integration enabled performance tracking, and early stopping was introduced to prevent overfitting. These adjustments optimized the training process for sentiment classification.

B. Training and Fine-tuning

NanoGPT was trained from scratch, whereas GPT-2 was fine-tuned, each using custom configurations aligned with their respective architectural parameters, including *n_layer*, *block_size*, *n_head*, *n_embd*, *dropout*, and *max_iters*. The *init_model* setting controlled whether the model was initialized from scratch or fine-tuned.

Both models were periodically evaluated on the validation set, employing early stopping based on validation loss. Checkpoints were saved upon improvement, and training metrics such as loss and learning rate were logged using WandB.

For NanoGPT, hyperparameters including *n_layer*, *n_head*, *n_embd*, and *learning_rate* were varied to analyze their effect

on performance, with *dropout* fixed at 0.0. In contrast, GPT-2 fine-tuning used a reduced *block_size* to mitigate CUDA memory constraints. Although only one configuration was tested for GPT-2 due to computational cost, it offers informative contrast against training from scratch.

IV. Evaluation

Accuracy provides an overall measure, while **class-wise accuracy** highlights performance across different sentiment categories, identifying potential misclassifications. The **F1 score** balances precision and recall, offering a more comprehensive evaluation, particularly for imbalanced datasets. The **confusion matrix** visualizes actual vs. predicted classifications, revealing where the model fails. Together, these metrics provide a thorough assessment of model performance on sentiment classification for both NanoGPT and GPT-2.

V. Results

TABLE I
ACCURACIES OF NANOGPT AND GPT-2

Model	Accuracy
NanoGPT (from scratch)	50.0%
GPT-2 (fine-tuned)	40.0%

TABLE II
CLASS-WISE ACCURACY FOR NANOGPT AND GPT-2

Class	NanoGPT	GPT-2
Positive	0.0%	0.0%
Neutral	90.0%	50.0%
Negative	60.0%	70.0%

TABLE III
F1-SCORE FOR NANOGPT

Class	Precision	Recall	F1-score
Positive	0.00	0.00	0.00
Neutral	0.39	0.90	0.55
Negative	0.86	0.60	0.71

TABLE IV
F1-SCORE FOR GPT-2

Class	Precision	Recall	F1-score
Positive	0.00	0.00	0.00
Neutral	0.33	0.50	0.40
Negative	0.47	0.70	0.56

The accuracy results indicate that NanoGPT, trained from scratch, outperforms fine-tuned GPT-2 with 50.0% versus 40.0%, respectively. However, both models exhibit low overall accuracy, highlighting difficulties in generalizing for sentiment classification. NanoGPT performs well on neutral sentiment (90.0%) but struggles with positive sentiment (0.0%), while GPT-2 shows similar trends, with moderate performance on negative sentiment (70.0%) but poor classification of positive sentiment (0.0%).

TABLE V
CONFUSION MATRIX FOR NANOGPT

True \Predicted	Positive	Neutral	Negative
Positive	0	10	0
Neutral	0	9	1
Negative	0	4	6

TABLE VI
CONFUSION MATRIX FOR GPT-2

True \Predicted	Positive	Neutral	Negative
Positive	0	7	3
Neutral	0	5	5
Negative	0	3	7

F1-scores reveal that NanoGPT excels in identifying negative sentiment (0.71), with high precision (0.86) and moderate recall (0.60), but performs poorly on positive sentiment (0.00). GPT-2's performance is also weak for positive sentiment (0.00) and neutral sentiment (0.40), with moderate performance for negative sentiment (0.56).

Confusion matrices show that both models predominantly classify instances as neutral, with significant misclassification of negative and positive sentiments. This suggests a bias toward neutral and negative sentiment and shows the challenges of sentiment analysis with imbalanced datasets.

The WandB report for NanoGPT training: WandB NanoGPT.

The WandB report for GPT-2 fine-tuning: WandB GPT-2.

VI. Discussion

The poor performance in positive sentiment classification is primarily due to the dataset imbalance, with positive instances comprising only a small fraction, making it difficult for both models to accurately identify positive sentiment. A more balanced dataset could improve model generalization across all sentiment categories.

Additionally, the preprocessing approach may have influenced performance, and further experiments with different strategies could offer valuable insights into optimizing model configurations.

VII. Conclusion

This assignment compared two transformer-based approaches for sentiment classification: training a custom NanoGPT model from scratch and fine-tuning a pretrained GPT-2 model. Both models struggled with positive sentiment classification due to class imbalance, but GPT-2, despite limited training iterations, showed competitive performance, suggesting that fine-tuning can be as effective as training from scratch.