Transformers and Attention-Based Deep Networks Assignment - 1

Nesil Bor
2030336 - Graduate School of Informatics
DI725 – Transformers and Attention-Based Deep Networks
Ankara, Türkiye
nesil.bor@metu.edu.tr

***Abstract*—This study explores the application of transformer-based deep networks for sentiment analysis of customer service conversations, classifying interactions into positive, negative, or neutral categories. Two approaches were implemented: a custom transformer model trained from scratch using a modified nanoGPT architecture, and a fine-tuned GPT-2 model leveraging pre-trained weights. The dataset, comprising customer-agent interactions, was pre-processed and analyzed to understand sentiment distribution. Both models were trained and evaluated using accuracy and loss metrics, with experiments tracked via Weights & Biases (WANDB) and version-controlled on GitHub. The fine-tuned GPT-2 model achieved a test accuracy of 46.67%, while the custom model reached 66.67%, highlighting trade-offs between training from scratch and transfer learning. These findings underscore the adaptability of transformers for sentiment analysis tasks.**

***Index Terms*—nanoGPT, finetune, GPT2, sentiment analysis**

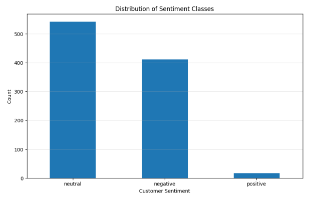
I. Introduction

Sentiment analysis of customer service interactions is critical for automating quality assessment in corporate settings. This assignment investigates transformer architectures, a cornerstone of modern natural language processing, to classify sentiments in customer-agent dialogues. Two models were developed: a custom transformer based on nanoGPT, trained from scratch, and a fine-tuned GPT-2 model utilizing pre-trained weights. The objective was to compare their performance on a three-class sentiment classification task (positive, negative, neutral) while adhering to clean coding practices, version control, and experiment tracking. This report details the methodology, results, and insights gained from these experiments.

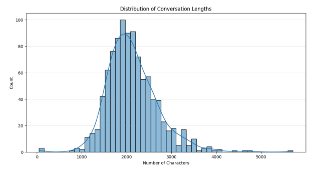
II. Dataset

The dataset for this study consists of customer service interactions from train.csv and test.csv, containing 970 and 30 samples, respectively, sourced from the same directory as the processing script. It includes 11 features: issue_area, issue_category, issue_sub_category, issue_category_sub_category, customer_sentiment, product_category, product_sub_category, issue_complexity, agent_experience_level, agent_experience_level_desc, and conversation. The target, customer_sentiment, is mapped as 0 (negative), 1 (positive), or 2 (neutral). The training set is imbalanced with 411 negative, 542 neutral, and 17 positive samples, while the test set is balanced with 10 samples per class. No missing values were found. For modeling, only conversation and sentiment were retained, with training data split into 80% training (776 samples) and 20% validation (194 samples) using stratified sampling. Text was cleaned by removing prefixes (e.g., "Agent:", "Customer:"), special characters, and excess whitespace, then tokenized using the GPT-2 BPE tokenizer (tiktoken) with a maximum length of 512 tokens. Processed data was saved as .bin files for tokens and .pkl files for labels in a processed directory.

Exploratory Data Analysis (EDA) informed feature selection. Numerical features, conversation_length (character count) and word_count (word count), were derived from conversation, showing strong sentiment correlation via boxplots—longer conversations and higher word counts vary across sentiment classes. Categorical features were assessed using bar plots and Chi-Square tests: issue_area (p = 4.70e-31) and issue_category (p = 6.59e-172) exhibited significant sentiment dependency $p < 0.05$, while product_category (p = 0.14) and agent_experience_level (p = 0.05) showed weak or no correlation. Consequently, modeling focused on conversation, leveraging its text dynamics over other features with minimal impact. For the EDA phase, a notebook file named Exploratory_Data_Analysis is also created and included in the images.



(a) Distribution of Sentiment Classes



(b) Distribution of Conversation Lengths

Fig. 1: A few examples from EDA part

III. Modeling

The custom nanoGPT model and fine-tuned GPT-2 model differ fundamentally in architecture, training approach, and resource utilization. The custom model, built from scratch, uses a lightweight configuration (5 layers, 4 heads, 384 embedding dimensions) tailored for the sentiment task, with all parameters (approximately 14.5M) trained on the dataset. It employs a batch size of 16 and a learning rate of 1.8e-4, optimized with AdamW and cosine decay, training for up to 1000 iterations with early stopping (patience=2). Conversely, the fine-tuned GPT-2 model leverages pre-trained weights from the 124M-parameter "gpt2" variant (12 layers, 12 heads, 768 embedding dimensions), with only the top transformer layer (transformer.h.11) and sentiment head unfrozen, adapting 10% of its parameters to the task. It uses a smaller batch size (8) due to memory constraints, a lower learning rate (3e-5), and a 10x higher rate for the sentiment head, reflecting a transfer learning strategy. The custom model prioritizes flexibility and efficiency, while GPT-2 exploits pre-trained linguistic knowledge, though limited fine-tuning restricts its adaptability to the small, imbalanced dataset.

IV. Evaluation

Model performance was assessed using accuracy and cross-entropy loss, suitable for the three-class sentiment classification task (negative, neutral, positive). Accuracy measured overall correctness, while loss evaluated model confidence, chosen for their interpretability and alignment with classification goals. A confusion matrix detailed class-specific performance, critical given the training set's imbalance (411 negative, 17 positive). Inference scripts evaluated the test set (30 samples), logging predictions and probabilities with temperature scaling (logits/2) to refine softmax outputs. Training and validation were tracked via Weights & Biases (WANDB), with early stopping (patience=2) halting the custom nanoGPT at 500 iterations and GPT-2 fine-tuning concluding at 950 iterations. Both models excelled at negative sentiment detection but struggled with positive samples, reflecting data skew. The custom model outperformed GPT-2, suggesting better task adaptation, while GPT-2 faced transfer learning limitations.

V. Results

The custom nanoGPT model achieved a test accuracy of 66.67% (20/30 correct) via 'sample.py', with a best validation loss of 0.3159 and accuracy of 91.25% at 400 iterations, predicting 9/10 negative, 7/10 neutral, and 4/10 positive samples; training stopped at 600 iterations (loss 0.3234, accuracy 93.75%). The fine-tuned GPT-2 model, evaluated with 'sample_finetune.py', recorded a test accuracy of 46.67% (14/30 correct), with a best validation loss of 0.6004 and accuracy of 75.00% at 900 iterations, identifying 7/10 negative, 7/10 neutral, and 0/10 positive samples after 950 iterations. Fig. 1 plots validation accuracy and loss, showing nanoGPT's faster convergence. WANDB logs confirmed stability, with inference outputs (e.g., [0.65, 0.25, 0.10] for negative) aiding interpretability. Test evaluation used a balanced set (10 samples per class).

VI. Discussion

Fig. 2 illustrates nanoGPT's rapid rise to 93.75% validation accuracy and 0.3159 loss by 400 iterations, stopping at 600,

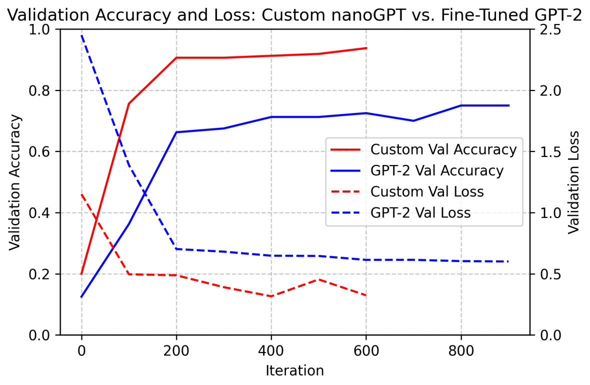


Fig. 2: Validation accuracy and loss curves for custom nanoGPT and fine-tuned GPT-2.

versus GPT-2's slower climb to 75.00% and 0.6004 by 900 iterations, highlighting nanoGPT's efficiency. GPT-2's constrained adaptation underscores transfer learning challenges on small datasets. Full training thus excels over partial fine-tuning when data is skewed and limited, though both struggle with positive class generalization.

The custom model outperformed the fine-tuned GPT-2 model in test accuracy, suggesting that training from scratch can better adapt to specific datasets, especially with imbalanced classes. However, GPT-2's lower performance may stem from limited fine-tuning (only the last layer and head adjusted) and a smaller training set relative to its pre-training corpus. The custom model's higher accuracy on negatives reflects the dataset's bias, while GPT-2's neutral bias indicates retained general language patterns. Future improvements could include data augmentation for neutral samples, deeper fine-tuning, or hyperparameter tuning via WANDB sweeps.

VII. Conclusion

This assignment demonstrated the efficacy of transformer models for sentiment analysis, with a custom nanoGPT model outperforming a fine-tuned GPT-2 model on a small, imbalanced dataset. Both approaches highlight the flexibility of transformers, though training from scratch proved more effective here. The process reinforced the importance of EDA, pre-processing, and experiment tracking. Future work could explore larger datasets and advanced fine-tuning strategies to enhance performance. You can access my code and files from: https://github.com/adigew/DI725-transformer-sentiment-analysis. Also you can access my WANDB results from these links: https://wandb.ai/adigew-middle-east-technical-university/nanoGPT-sentiment?nw=nwuseradigew https://wandb.ai/adigew-middle-east-technical-university/nanoGPT-sentiment-gpt2?nw=nwuseradigew

References

[1] A. Karpathy, "NanoGPT," GitHub, 2023. [Online]. Available: https://github.com/karpathy/nanoGPT