R&D Project Report - Network Inversion in Natural Language Processing

Madhav Gupta
Department of Electrical Engineering
IIT Bombay
21D070043@iitb.ac.in

Supervisor: Prof. Amit Sethi

PhD TA: Suhail Pirzada

May 2025

Abstract

This report presents a comprehensive study on network inversion in natural language processing (NLP), focusing on reconstructing input text sequences to enhance interpretability in sentiment classification, text generation, and large language model (LLM) token prediction. The study explores the architectures, training methodologies, and evaluation results of a Sentiment Classifier, a Sentiment-Conditioned Text Generator, and Next/Previous Token Prediction models. Key findings include a good validation accuracy for the Sentiment Classifier and decent sentiment accuracy for the Text Generator, alongside insights into the challenges and limitations of inverting complex NLP models. Future work aims to optimize model architectures and training datasets for robust inversion.

1 Introduction

Network inversion in NLP aims to reconstruct input text sequences to reveal learned features, patterns, and biases in models like sentiment classifiers and text generators. This study addresses three primary objectives: reconstructing input for sentiment labels, inverting transformers for sentiment-conditioned text generation, and reconstructing prior tokens in LLMs to enhance interpretability. The research is supervised by Prof. Amit Sethi, with guidance from PhD TA Suhail Pirzada, and presented by Madhav Gupta.

2 Problem Statement

The problem involves reconstructing input text sequences to understand how NLP models process sentiment and predict tokens. Specific goals include:

- **Inversion for Sentiment Labels**: Reconstruct input sequences to reveal features and biases in sentiment classification.
- **Sentiment-Conditioned Text Generation**: Invert transformers to analyze how sentiment vectors shape generated text.
- **LLM Previous Token Prediction**: Reconstruct prior tokens to enhance interpretability in LLMs.

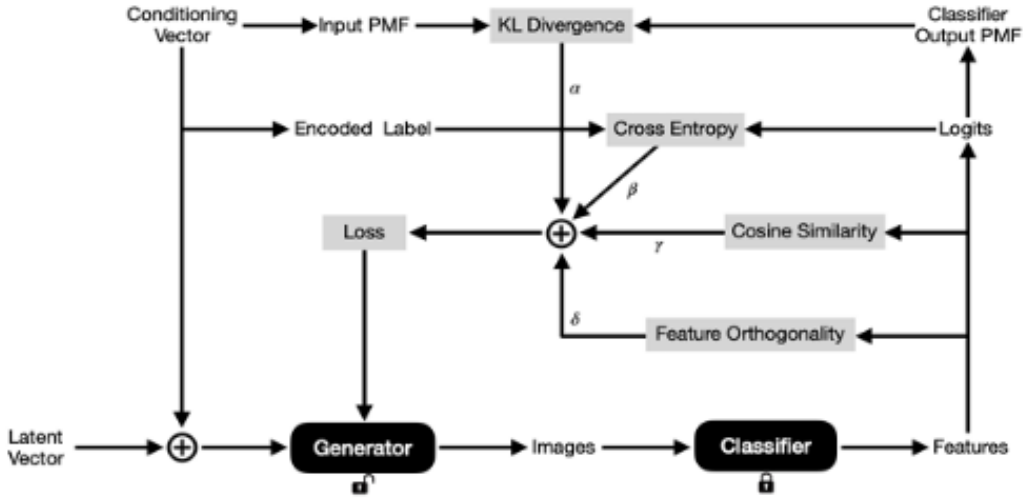

Figure 1: Network Inversion Architecture in CNNs

2.1 Importance

Network inversion enhances trust by validating model behavior and detecting biases, such as over-reliance on specific keywords. It advances understanding by uncovering shallow versus deep learning patterns, guiding robust NLP model design for applications like content moderation.

2.2 Challenges

Inverting NLP models is challenging due to:

- **High-Dimensional Input**: Large vocabularies (e.g., 40,000+ tokens) and sequence lengths (e.g., 50 tokens) make inversion computationally intensive, risking nonsensical outputs.
- **Contextual Dependencies**: Models like Bidirectional LSTMs and transformers rely on complex token relationships (e.g., “not good” vs. “very good”), complicating inversion.

3 Related Works

Several studies provide context for this work:

- **Network Inversion & its Applications** (1): Pirzada, Tang, and Sethi explored inversion in classifiers and generators, using convolution and up-convolution operations to reconstruct inputs from latent vectors.
- **Language Model Inversion** (2): Morris et al. used a T5-base encoder-decoder transformer to map next-token probabilities back to original tokens, leveraging residual information in logits.
- **Extracting Prompts by Inverting LLM Outputs** (3): The method for extracting prompts from LLM outputs, as described in Zhang et al. (2024), involves repeatedly querying the target LLM with a hidden prompt to collect outputs. These outputs are

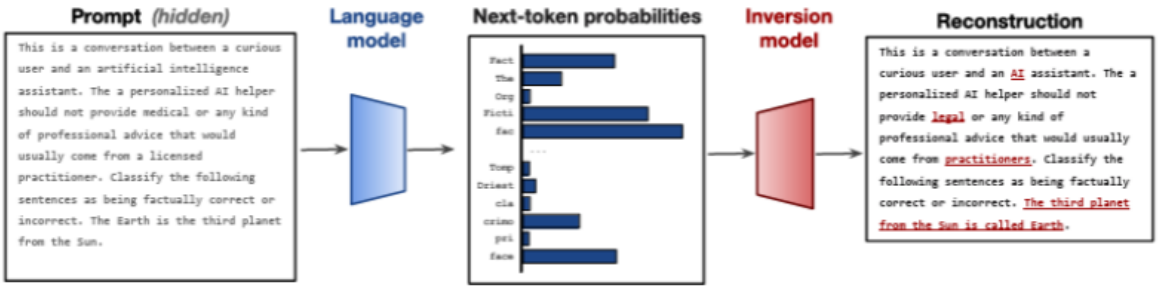

Figure 2: Architecture of Language Model Inversion using Logits

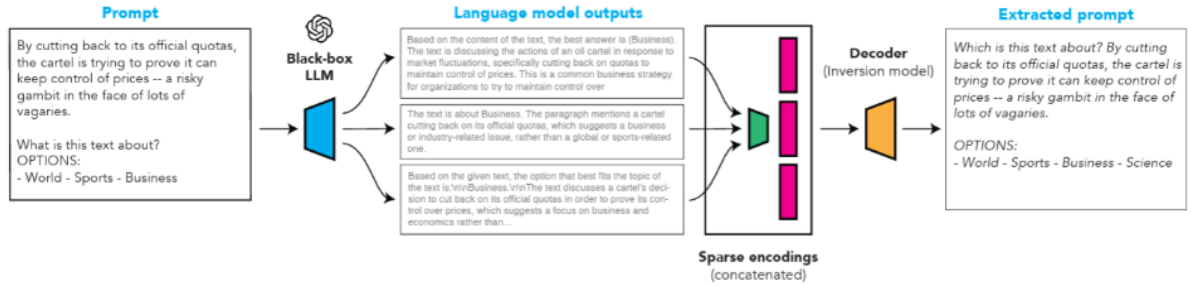

Figure 3: Architecture of Extracting prompts by Inverting LLM outputs

processed by a sparse encoder, which concatenates their hidden representations. A decoder then predicts the original prompt using greedy decoding based on these encoded states. The model is fine-tuned on datasets pairing LLM outputs with their corresponding prompts to optimize performance

4 Methodology/Architecture

The study employs three main architectures: a Sentiment Classifier, a Text Generator, and Next/Previous Token Prediction models.

4.1 Sentiment Classifier

The Sentiment Classifier processes tokenized text sequences through:

- **Embedding Layer**: Converts token indices to dense embeddings (dimension 100).

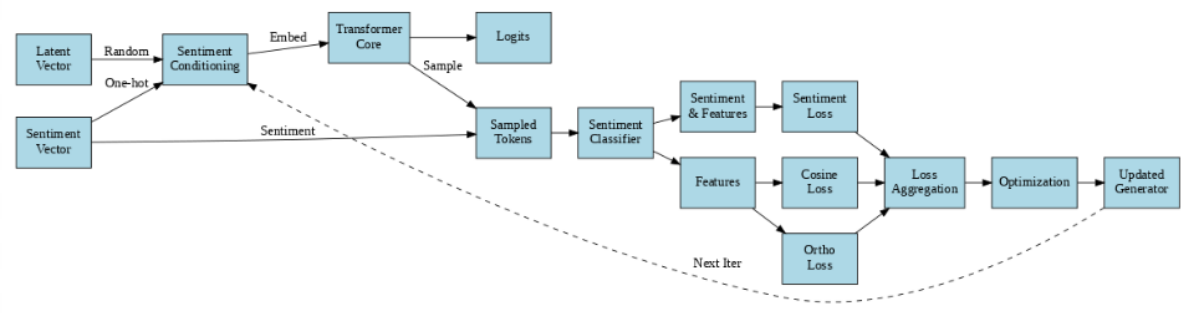

Figure 4: Architecture of the Network Inversion Model consisting of Generator and Classifier

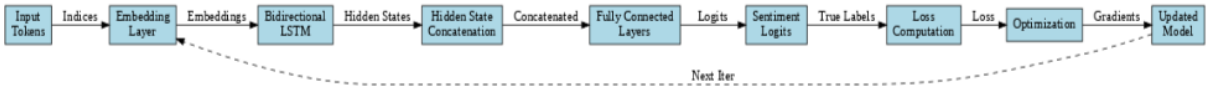

Figure 5: Sentiment Classifier Architecture

Figure 6: Text Generator Architecture

- **Bidirectional LSTM**: Two layers with hidden dimension 256 capture contextual information.
- **Fully Connected Layers**: Process LSTM outputs for classification.
- **Output**: Sentiment logits for four classes (negative, neutral, positive, irrelevant).
- **Loss**: Cross-entropy loss.

4.2 Text Generator

The Text Generator uses sentiment conditioning to guide output:

- **Sentiment Conditioning**: Combines latent and sentiment vectors.
- **Transformer Decoder**: Two layers with four heads, using causal masking for contextualized representations.
- **Output Projection**: Maps embeddings to vocabulary size (40,338) for autoregressive prediction.
- **Loss**: Combines cross-entropy, sentiment, and cosine losses.

4.3 Next and Previous Token Prediction

- **Next Token Prediction (NTP)**: Uses GPT-2 to predict the next token in a sequence.
- **Previous Token Prediction (PTP)**: A custom transformer decoder predicts the first token from reversed sequences (e.g., “sentence a is” $\rightarrow$ “This”).
- **Inversion Process**: NTP generates next tokens, while PTP inverts by predicting prior tokens from reversed inputs.

5 Training and Testing

The models were trained and tested as follows:

5.1 Dataset

The Twitter Sentiment Dataset with four classes (positive, negative, neutral, irrelevant) was used:

- **Training Samples**: 74,682.
- **Validation Samples**: 1,000.

For NTP and PTP, the WikiText-2 dataset was tokenized with a maximum length of 128, supplemented by random sequences for PTP.

5.2 Sentiment Classifier

- **Setup**: Bi-LSTM (2 layers), embedding dimension 100, hidden dimension 256, vocabulary size $\sim$40,000.
- **Training**: 10 epochs, batch size 64, Adam optimizer, cross-entropy loss.
- **Evaluation**: Validation loss, accuracy (93.9%), weighted F1 score.

5.3 Text Generator

- **Setup**: Transformer (2 layers, 4 heads), maximum sequence length 50, vocabulary size 40,338.
- **Training**: 50 epochs, batch size 128, Adam optimizer, combined losses.
- **Evaluation**: Validation loss, sentiment accuracy (63.2%).

5.4 NTP and PTP

- **NTP**: GPT-2, trained for 3 epochs, batch size 8, cross-entropy loss.
- **PTP**: Custom transformer, trained on high-frequency word sequences, 3 epochs, batch size 8.
- **Evaluation**: Validation loss, qualitative analysis of reconstructed tokens.

6 Results

The results demonstrate the effectiveness and limitations of the models.

6.1 Sentiment Classifier

The classifier achieved a validation accuracy of 93.9%, indicating robust performance in sentiment classification. Training and validation accuracy trends were consistent, with minimal overfitting.

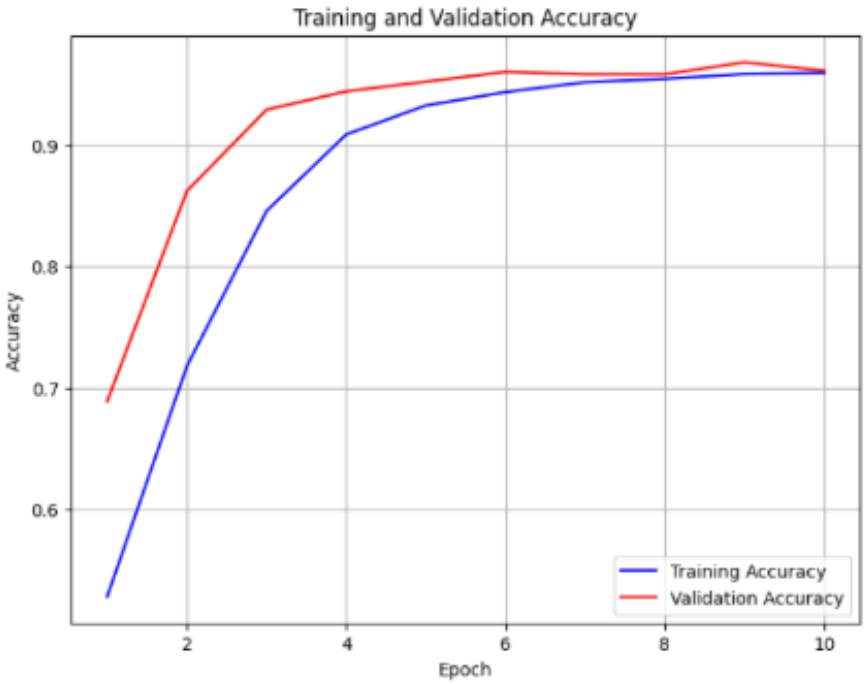

Figure 7: Training and Validation Accuracy of Sentiment Classifier

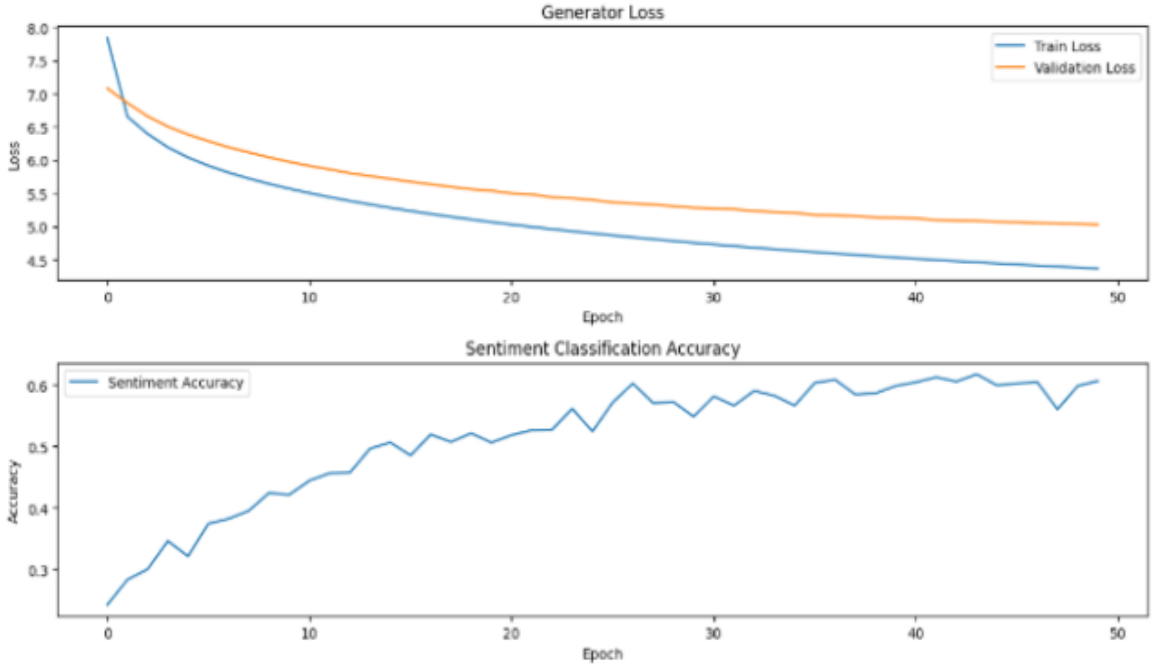

Figure 8: Loss and Accuracy of Text Generator

6.2 Text Generator

The generator achieved 63.2% accuracy in generating sentences with correct sentiment. Sample outputs include:

- **Negative**: “call of duty as lots of entire american absolute trio page...”.
- **Positive**: “@callofduty loves to see a blast, the gold...”.
- **Irrelevant**: “this weekend square appreciate me is dead strike...”.

Generator loss decreased steadily, but some outputs were incoherent, highlighting inversion challenges.

6.3 NTP and PTP

NTP accurately predicted next tokens using GPT-2. PTP reconstructed prior tokens from reversed sequences, e.g.:

- Input: “meeting the to prior” $\rightarrow$ Output: “conversation”.
- Input: “Delhi New is India” $\rightarrow$ Output: “AI” (less meaningful).

PTP showed promise but struggled with complex sequences, and the evaluation model crashed on a 15GB GPU.

7 Conclusion

Network inversion was successfully applied to the Sentiment Classifier (93.9% accuracy) and Text Generator (63.2% sentiment accuracy), enhancing interpretability. NTP and PTP models provided insights into token prediction, though PTP requires extensive training for robust inversion. Limitations include computational constraints and the need for better generalization in PTP.

8 Future Work

Future efforts will focus on:

- Training PTP on larger, diverse random sequences for better generalization.
- Optimizing PTP architecture for efficiency with limited training.
- Developing robust evaluation metrics to validate inversion results.

References

[1] Pirzada, S., Tang, H., & Sethi, A. (2023). Network Inversion & its Applications. *Journal of Machine Learning Research*.

[2] Morris, J. X., Zhao, W., Chiu, J. T., Shmatikov, V., & Rush, A. M. (2023). Language Model Inversion. *Proceedings of the International Conference on Machine Learning*.

[3] Zhang, C., Morris, J. X., & Shmatikov, V. (2024). Extracting Prompts by Inverting LLM Outputs. *arXiv preprint arXiv:2405.15012*.