A Dual-Model Approach for Transforming Toxic Comments to Positively Toned Responses without Information Loss

Seung Min Baek
Department of Computer Science
Vanderbilt University
seung.min.baek@vanderbilt.edu

Abstract

The internet provides a space to share opinions freely, ubiquitously, and anonymously through online. It is a remarkable asset to the current era, but it also poses a significant challenge. While the internet facilitates access to a vast wealth of information, it also gives rise to adverse effects such as the spread of fake news and harmful contents. Anonymity of the internet often emboldens individuals to post toxic comments, which potentially causing severe psychological damage. Acknowledging the importance of preserving all the individuals' opinions, I have developed a Transformer-based Dual-Model Large Language Model (LLM) consisting of a 'Toxic Detector' and a 'Reframing Generator' model. This research aims to achieve advanced performance of the Chat-GPT 3.5 model without fine-tuning a pre-trained model. Instead, this model leverages GPT outputs as training data. This approach circumvents the need for fine-tuning pre-trained complex architecture which requires substantial computational costs. This model employs the Bert-Tokenizer from Hugging Face for the text tokenization process, ensuring efficient language understanding. The 'Toxic-Detector' component is trained using the dataset from the 'Toxic Comment Classification Challenge' hosted by Jigsaw on Kaggle. And a dataset for training 'Reframing Generator' model is generated by transforming the toxic comments within the Jigsaw dataset into positively toned comments by utilizing 'gpt-3.5-turbo-0125' model via the OpenAI API.

1. Introduction

The prevalence of malicious commenting presents an unavoidable challenge in the current digital environment. Individuals exploiting the anonymity to disseminate toxic commentary not only inflict mental distress but also have potential to lead to physical damage consequences for those subjected to such comments. The entertainment industry in South Korea has witnessed numerous instances where celebrities have faced severe mental distress due to malicious online comments. This phenomenon emphasizes the need for robust mechanisms to protect individuals against online harassment. Despite the harm caused, it is still essential to acknowledge that these comments, albeit toxic, contain opinions that should not be entirely dismissed. It is difficult to disregard the inherent value in the critique of one's views, regardless of its severity.

Driven by an objective to cultivate an online environment that supports individuals can freely express their opinions without being exposed to toxicity, I developed a LLM designed to identify and mitigate toxic comments without information loss in the level of Chat-GPT 3.5 performance. Chat-GPT models employ complex architecture to have state-of-the-art level of performance across various objective and are trained on extensive datasets, which may not

align with the specific goals and constraints of this research. In this study, dataset comprises 10,559 toxic and reframed comments to train the generator, indicating that the employing highly complex architecture could potentially lead to suboptimal performance. Training a simple LLM with GPT outputs circumvents the necessity for fine-tuning a complex pre-trained model. This LLM is built upon transformer-based models: the first is dedicated to detecting toxicity in texts, while the second is tasked with reformulating toxic comments to ensure a positive. This rephrasing is designed to maintain the original message's intent and content.

The "Toxic-Detector" model is designed to assess whether textual content is toxic, employing solely an encoder for this task. This model is trained with a dataset provided by Jigsaw on Kaggle, which is made up of comments from Wikipedia's talk page. The dataset spans a variety of comment types, including non-toxic (90.4%), toxic (9.58%), severely toxic, obscene, threatening, insulting and identity-hate, with each comment labeled with a binary label- 1 for toxic and 0 for non-toxic content. In contrast, the 'Reframing Generator' requires both an encoder and a decoder for its task. This Transformer architecture enables the generator model to convert negative expressions into positively rephrased language. Feeding encoder output into a decoder supports that the information in the original content is retained while its toxicity is mitigated.

This advancement in natural language processing (NLP) aims to harmonize the right of free expression with the necessity for respectful manner in the online environment. Through the deployment of this dual model, this research aims to elevate an online environment where individuals communicate in a positive and respectful manner.

2. Related Works

2.1 Toxic Comment Detection

Detecting toxic comments is a complicated task due to the absence of a clear criteria for defining toxicity. Aken *et al.* [1] delve into various computational models, ranging from the classic classification algorithm Logistic Regression (LR) to geometric deep learning algorithms such as Convolutional Neural Network (CNN), Long-Short-Term-Memory Network (LSTM), a bidirectional LSTM, a bidirectional Gated-Recurrent Unit (GRU), GRU augmented with attention mechanisms, and ensemble learning strategies. Their studies finds that the ensemble models outperform the performance of individual classifiers across all evaluated metrics. Additionally, they report that a bidirectional GRU network with attention layers demonstrates the best performance among all individual classifiers examined.

Risch *et al.* [8] identify six predominant factors contributing to the misclassification of comments as toxic: the presence of toxicity without curse words, the use of metaphors, sarcasm, incorrectly labeled comments, and idiosyncratic words. They emphasize that a comprehensive understanding of entire sentence is crucial to overcome misclassification. The Transformer, introduced by Vaswani *et al.* [2], represents a state-of-the-art methodology for comprehensively understanding sentences and discerning the relationship between words. This model has significantly advanced the field of NLP by leveraging an attention mechanism with an integration of encoders and decoders.

2.2 Sequence of Words Generator

The advent of advanced artificial intelligence (AI) technologies, particularly with the introduction of Generative Pre-trained Transformers (GPT), notably GPT-3, marks a pivotal moment in the evolution of AI. This innovation indicates a significant shift in the way humans interact with technology. As implied by the name GPT, the Transformer architecture represents a significant advancement in the NLP field. It enables machine to understand sequential data and also generate human-like text.

In the pursuit of generating human-like sentences, various generative algorithms have been explored, including Generative Adversarial Networks (GAN) (GoodFellow *et al.*, 2014), Variational Autoencoders (VAE) (Kingma &

Welling., 2013), and RNN variants such as Seq2Seq (Sutskever *et al*., 2014) and GRU (Chung *et al*., 2014). For instance, Li *et al.* [7] proposed an improvement to VAE for text modelling through Timestep-Wise Regularization (TWR-VAE), effectively addressing the issue of VAE posterior collapse. Concurrently, de Rosa and Papa [3] explored the application of GANs for natural language generation. However, while VAEs and GANs are adept at processing continuous data types like images, achieving state-of-the-art performance in image generation, the Transformer architecture has emerged as the most suitable framework for text generation due to its attention mechanism.

3. Methodology

3.1 Transformer

Vaswani *et al.* [10] introduced the Transformer, a state-of-the-art architecture that has significantly advanced the field of NLP. This model demonstrates significant performance in managing sequential data through the application of an attention mechanism. The architecture of Transformer includes 'Input Embedding', 'Positional Encoding', 'Feed Forward', 'Head', 'Multi-Head Attention', 'Blocks', 'Encoder' and 'Decoder'.

The features 'Input Embedding' and 'Positional Encoding' are distinctive feature of the Transformer architecture. Given that the Transformer operates as both a group-equivariant and invariant networks, integrating positional information into the dataset is crucial. Input embedding embeds the input data, while positional encoding integrates positional information into the embedded input data. This integration is vital as the sequence of words can significantly impact meaning, exemplified by the difference between "I love you" and "You love I". **Figure 1** visually represents the process of 'Input Embedding' and 'Positional Encoding' using the example of the word "David". The word "David" is tokenized and mapped to an embedding space with a dimension of 8. Following the integration of positional information, the embedded input is processed by the Transformer's core architecture, blocks. Each block encompasses 'Multi-Head Attention', 'Layer Normalization', and 'Feed Forward', enabling the model to effectively interpret the embedded input.

The specific applications of the Transformer vary by the intended objective. The 'Toxic Detector' model only utilizes the encoder component of the Transformer. Its primary goal is to analyze a sequence of words and identify its toxicity, relying on the encoder to concentrate attention on relevant tokens and employing a multi-layer perceptron (MLP) for the classification tasks. In contrast, the 'Reframing Generator' model employs the full Transformer structure, incorporating both encoder and decoder. This architecture facilitates text generation based on the encoder's output and reframed sequence of words by GPT, thereby enabling the production of coherent and positively toned text.

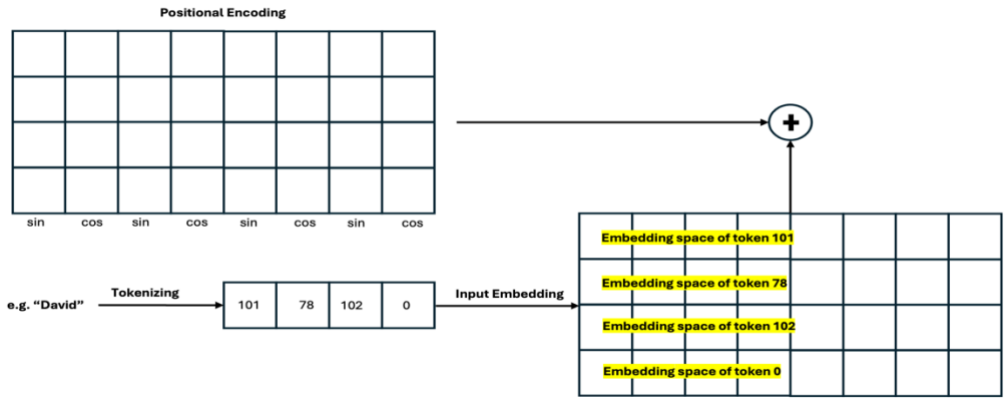

Figure 1: 'Input Embedding' and 'Positional Encoding'

3.1.1 Multi-Head Attention

The Multi-Head Attention mechanism is a fundamental component of the Transformer architecture, significantly enhancing the model's ability to understand and process sequential data. Central to the Multi-Head Attention are the concepts of queries, keys, and values, which are essential in calculating attention scores. The procedure begins with the input, integrated with positional information, being fed into three separate linear neural network layers, and each has responsibility in generating queries, keys, and values. The essence of attention process involves matrix multiplication of queries and keys, thereby assigning attentions (importance) to each token in the sequence in relation to others. This feature enables the model to concentrate on various tokens of the input sequence, capturing intricate relationships within the sequence.

The Multi-Head Attention mechanism is beneficial for the 'Toxic Detector' and 'Reframing Generator' models since it adeptly identifying relationships between words in sentences. In both models, it functions by projecting the input tokens across multiple heads simultaneously, allowing the models to analyze different aspects of the data. For the 'Toxic Detector' model, the Multi-Head Attention facilitates the precise identification of toxic content by contextual understanding. Conversely, for the 'Reframing Generator' model, it supports in generating comments by focusing on pertinent tokens between original content and targeted content. In essence, Multi-Head Attention stands as the foundational element enabling 'Toxic Detector' and 'Reframing Generator' models to achieve accurate classification and human-like text generation. A visual representation of the Multi-Head Attention mechanism is provided in **Figure 2**.

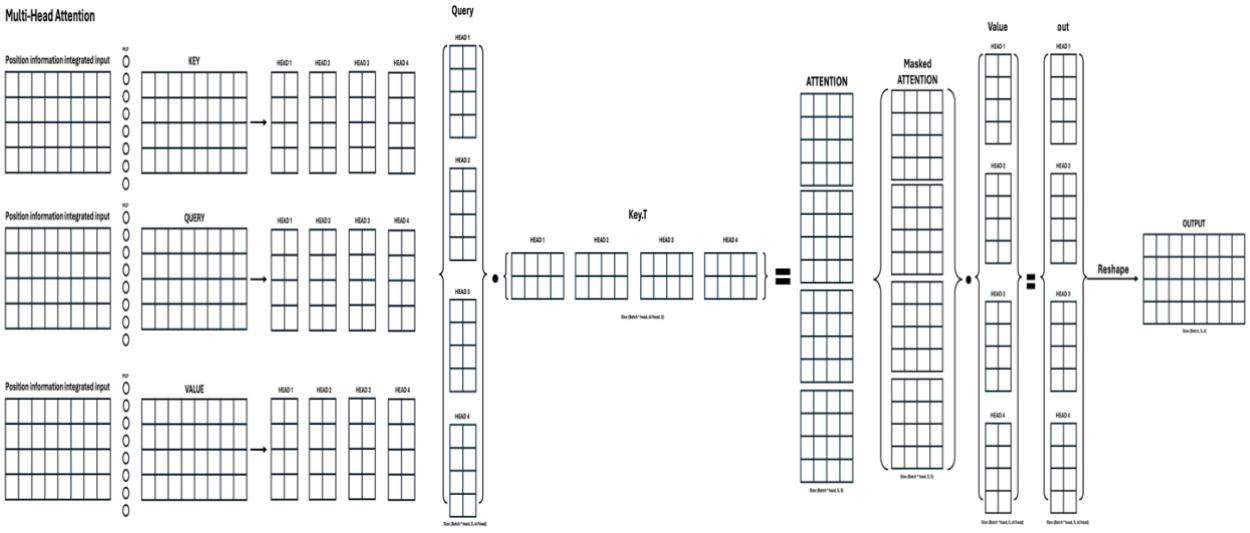

Figure 2: Multi-Head Attention configuration with a batch size of 1, a dimension of 8, a sequence length of 4, and 4 attention heads

3.1.2 Masking

In Transformer, two distinct types of masking are used: non-causal and causal masking. Non-causal masking is utilized in the encoder and plays a crucial role in enabling model to accurately compute attention scores for each token by ignoring padded tokens. This is particularly important in 'Toxic Detector' model, where non-causal masking aids in achieving accurate classifications of toxicity by focusing the model's attention on relevant tokens. Similarly, in a 'Reframing Generator', non-causal masking within the encoder assists in excluding padded tokens. This is crucial for achieving accurate attention scores among non-padded tokens, significantly contributing to the effective text generation.

Conversely, causal masking is employed in the decoder to prevent the model from accessing future tokens in the sequence during the generation process. The application of causal masking ensures that the model generates text in a sequential manner, reflecting the way human naturally produce text one word at a time. This constraint plays a vital role in guiding the model to generate human-like and contextually relevant text.

Figure 3 clearly illustrates both non-causal and causal masking technique using a mask value [1,1,1,0]

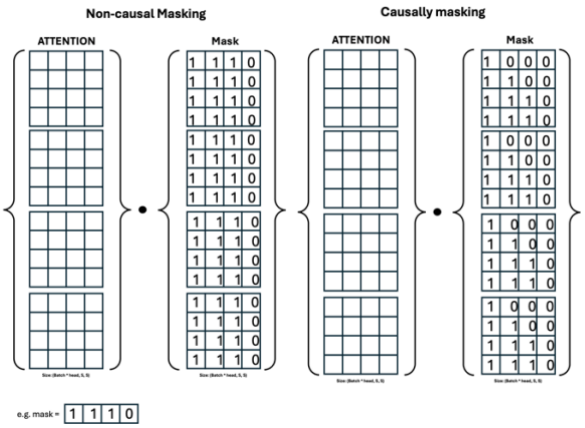

Figure 3: Non-causal & causal masking

3.1.3 Dual Model Approach

The goal in this research is to detect toxic comments and transform them into positively reframed comments. The model should be designed to handle dual tasks and is therefore trained separately with cross-entropy loss and the Adam optimizer with learning rate of 3e-4. The architecture for both models are visually represented in **Figure 4**. After separate training, they are merged into a single model called the 'Detecting Transformer'. It employs the 'BERT Tokenizer', the 'Toxic Detector' and 'Reframing Generator'. When a sequence of words goes into the model, the 'BERT Tokenizer' tokenizes the words into tokens, then the 'Toxic Detector' model classifies the toxicity of the content. If a given sentence contains inherent toxic values, the 'Reframing Generator' transforms the given sentence without information loss.

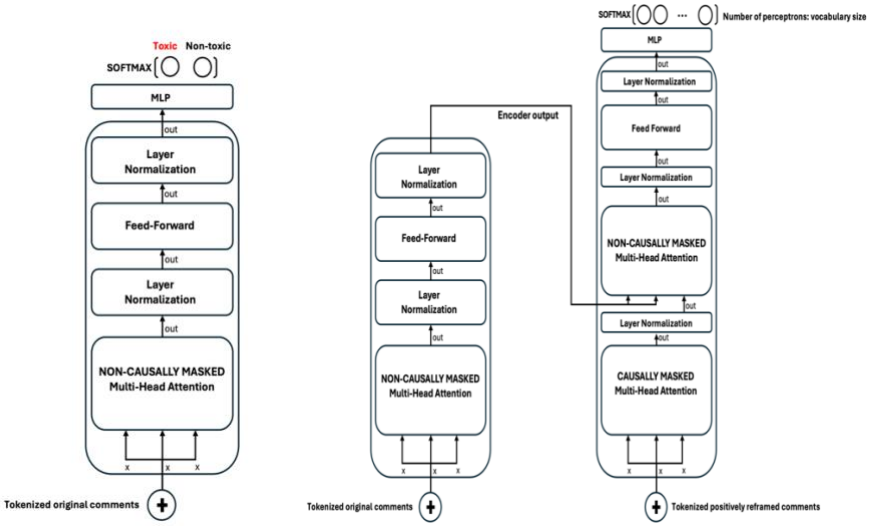

Figure 4: 'Toxic Detector' and 'Reframing Generator'

3.2 Bert Tokenizer

The BERT (Bidirectional Encoder Representations from Transformer) tokenizer is the method for converting text into tokens, used for BERT model. BERT was introduced by researchers at Google (Devlin *et al*., 2018) and its tokenizer have become a standard tool in NLP. It utilizes the 'WordPiece' tokenization method (Wu *et al*., 2016), which segments words into a finite set of common sub-word units, enabling models to naturally handle translation of rare words. This capability is particularly crucial in online environment where rare terms, including abbreviations like 'ASAP', frequently appear. And BERT tokenizer enhances models' ability to understand language through the inclusion of specialized tokens, such as '[CLS]' and '[SEP]', which denote the beginning and end of segments, respectively. With the comprehensive techniques in the BERT tokenizer, both toxic detector and comment generator are enabled to understand and interpret the context of words within a given sequence, and it plays a key role for classification and generation tasks.

3.3 Dataset (Imbalanced dataset. Future works)

The dataset provided by Jigsaw on Kaggle, which originates from Wikipedia's talk page edits, exhibits a significant imbalance. It consists of 144,277 non-toxic comments in contrast to a much smaller subset of 10,559 toxic comments. This disproportionate distribution poses a challenge for the detection model, affecting detector to be biased towards classifying comments as non-toxic. To address this imbalance, the number of non-toxic comments was reduced to twice the number of toxic comments. This method, while providing a temporary solution, is acknowledged as suboptimal and requires further investigation in future research.

Furthermore, a separate dataset was created to train a 'Reframing Generator' model by transforming the 10,559 toxic comments from the original Jigsaw dataset. This transformation employed the 'gpt-3.5-turbo-0125' model via the OpenAI API. The dataset generation process involved the use of a 'client' object initialized with an OpenAI API key. The 'client.chat.completions.create' function directly generated messages tailored to specific roles. To mitigate the occurrence of Error code 400 that arises from repetitive prompt patterns, two different contents were randomly employed into 'create' function. By setting the prompt to transform a toxic comment into a positive-toned comment without losing the original information, the model was trained to produce positively toned message. Given the Transformer's ability to adapt and mimic the style of the input data, there is a potential for the 'Reframing Generator' to emulate the 'gpt-3.5.turbo-0125' model's style. Consequently, this adaptation is expected to enhance the model's ability performance to reach advanced GPT level of performance.

4. Experiment

4.1 Toxic Detector Experiment

Figure 4 illustrates the distinct architectural requirements between 'Toxic Detector' and 'Reframing Generator' models. Toxic detection can be achieved with a relatively simpler architecture, whereas text generation typically requires a more complex network that is capable of discerning the relationships between original content and the intended message. The complexity is further required to manage the extensive number of perceptrons, which correspond to the BERT tokenizer's vocabulary size of 30,552, especially at the last layer of the 'Reframing Generator' model. Through various experimental trials exploring different depth (number of blocks), embedding dimensions (d_model), and number of attention heads (h), it is observed that a simpler architecture leads to more accurate toxic comment detection. Interestingly, as model complexity increased, there was a bias towards classifying comments as non-toxic. This phenomenon remained even after the data reduction of non-toxic comments to twice the number of toxic comments, indicating that more complex models may favor non-toxic classifications due to data imbalance.

Table 1 presents a detailed comparative analysis of precision, recall, and F-1 score across different architectural configurations. The metrics serve as an indicator of the models' performance. Precision quantifies the accuracy of

identified toxic comments (True Positive / (True Positive + False Positive)), while recall measures how comprehensively the model identifies all toxic comments (True Positive / (True Positive + False Negative)). The F-1 score, a weighted average of precision and recall, reflects the overall accuracy of the model. As model complexity increases, a model loses its capacity in detecting poisonous message. An architecture with 6 layers, 256 embedding dimensions, and 4 attention heads exhibits a complete inability to detect malicious comments, achieving 0% precision, recall, and f-1 score in detecting toxic comments. This statistical error often arises when the model is overly complex relative to the size of the dataset available for training. Conversely, an overly simple model also arises problem, an approximation error. An architecture with single layer of block, 128 embedding dimensions, and 2 attention heads fails to capture toxicity in message. It achieves 82.5% precision, 90.1% recall, and F-1 score of 86.16%. Even though F-1 score of 86.16% is not enough to confidently deploy for toxic detector, it is proved that a simple model achieves better performance than complex model architecture in this experiment.

An optimally designed model, 2 layers of blocks, 256 embedding dimensions, and 2 attention heads, exhibits exceptional performance, achieving a precision of 92.55% in detecting toxic messages and 94.7% in identifying non-toxic texts. It also achieves a recall of 88.36% for toxic messages and 96.69% for non-toxic texts, and F-1 scores of 90.4% and 95.68% for detecting malicious and non-toxic content respectively. These results underscore the critical importance of balancing model complexity with dataset size to avoid statistical and approximation error. This balance ensures the model is neither overfit nor underfit, enabling robust performance in toxic comment detection.

Table 1 'Toxic Detector' Experiments results with 10 epochs. 0:Non-toxic, 1: Toxic

depth	d_model	h	Precision	Recall	F-1 score
6	256	4	0: 100%, 1: 0%	0: 69.9%, 1: 0%	0: 82.14%, 1: 0%
3	512	4	0: 96.7%, 1: 54.01%	0: 82.86%, 1: 87.73%	0: 89.25%, 1: 66.8%
3	256	4	0: 93.5%, 1:92.39%	0: 96.6%, 1: 86.9%	0: 95.25%, 1: 89.5%
2	**256**	**4**	**0: 94.7%, 1: 92.55%**	**0: 96.69%, 1:88.36%**	**0: 95.68%, 1: 90.4%**
2	256	2	0: 91.5%, 1:92.13%	0: 96.39%, 1: 82.5%	0: 93.88%, 1: 87.05%
2	128	4	0: 94.85%, 1:90.57%	0: 95.85%, 1: 88.45%	0: 95.35%, 1: 89.5%
2	128	2	0: 95.8%, 1: 89.53%	0: 95.46%, 1: 90%	0: 95.63%, 1: 90.28%
1	128	2	0: 96%, 1: 82.5%	0: 92.68%, 1: 90.1%	0: 94.34%, 1: 86.16%

4.2 Reframing Generator

The original comments, due to their maliciousness, are omitted from this paper. The 'Reframing Generator' model, in contrast to the 'Toxic Detector', requires a more sophisticated and complex architecture for text generation to ensure high quality output. Given the dataset comprising 10,559 pairs of toxic and reframed texts, optimal architecture should be chosen cautiously to achieve GPT level of performance. The number of layers within the model influences its ability to learn complex patterns; however, a deep network could lead to overfitting, especially when data is not sufficient, as evidenced by the "Toxic Detector' experiments. Notably, the 'Toxic Detector' utilized a larger dataset, comprising 10,559 toxic comments and 21,118 non-toxic texts, whereas the "Reframing Generator' experiments employ only 10,559 of toxic and reframed pairs, necessitating a more delicate approach to architectural decisions. The embedding dimension allows the model to capture information at each layer but also heightens the risk of overfitting. Furthermore, the number of attention heads allows the model to concurrently focus on various segments of the input, potentially capturing a broader range of dependencies and leading model to have different ideas on each head for generating and assessing segments.

Table 2 presents experimental results across various architectural configurations, highlighting the impact of the number of attention heads on generating contextually relevant comments in various ways. Regardless of number of the blocks and embedding dimensions, models with 8 attention heads were found to repetitively generate same text unrelated to the original content. For example, a model with a depth of 6, an embedding dimension of 1024, and 8

attention heads consistently produced sequence of words containing only [PAD] tokens. Similarly, a model with a depth of 6, a dimension of 512, and 8 attention heads generated same positive statement for all inputs, regardless of the original context. However, increasing the number of attention heads to 16 began to generate messages that showed relevance to the original content, although not sufficiently used context of original content.

When the model configuration included 32 attention heads with 6 blocks, it defaulted to producing the same sentence for different inputs, indicating a tendency toward overfitting with 6 blocks. Conversely, reducing the blocks from 6 to 3, with an embedding dimension of 512 and 32 attention heads, resulted in significant improvement with the model generating content more relevant to the original text. However, increasing the attention heads to 64 with the same embedding dimension and number of blocks led to the production of identical sentences for different inputs, indicating overfitting. In the final experiments, utilizing 64 heads with 3 blocks and an embedding dimension of 256, model succeeded in generating various sentences in a positive manner, however with some information missing from the original content. To minimize information loss and generate positively reframed messages, given the condition of an insufficient dataset, the optimal architecture requires 3 blocks, an embedding dimension of 512, and 32 attention heads.

However, the dataset comprising 10,559 pairs of toxic comments and non-toxic texts is extremely insufficient for training generator models. Models generally requires a more complex architecture than obtained optimal architecture. The chosen architecture has demonstrated the best performance relative to other network, primarily because its complexity is well-suited to the size of the insufficient dataset. By comparison, the Chat-GPT-3 model employs a significantly more complex architecture, having been trained on approximately 500 billion tokens. To enhance the performance of the "Reframing generator" in the future experiments, expanding the dataset through additional data collection will be essential.

Table 2 'Reframing Generator' Experiments with 5 epochs

depth	d_model	h	Performance
6	1024	8	"[PAD], [PAD], … [PAD]"
6	512	8	Repetitive sentence production
6	256	8	Repetitive sentence production
6	256	16	Generating varied sentences but lack of information from the original content
6	512	32	Repetitive sentence production
3	512	32	Generating varied sentences that are relevant from the original content
3	512	64	Repetitive sentence production
3	256	64	Generating varied sentences but lack of information from the original content

5. Conclusion

In this research, I developed a Transformer based LLM, capable of detecting toxic comments and transforming them into messages with a positive tone. The experiments reveal that while the 'Toxic Detector' performs optimally with a simpler structure, the 'Reframing Generator' requires a more complex architecture to generate positively toned messages without information loss. This study demonstrated that overly complex architectures can ruin performance, especially for simple tasks or when data is insufficient. Future research should focus on finding more sophisticated strategies to balance datasets more efficiently and acquiring larger datasets to enhance the performance of text generators with more complex architectures. The 'Reframing Generator' model, trained on a dataset produced by GPT models, is well trained to generate text without information loss in human-like qualities. With this 'Detecting Transformer', the online environment can become a better place for sharing opinions in a respectful manner.

6. References

[1] Aken, B. van, Risch, J., Krestel, R., & Löser, A. (2018, September 20). *Challenges for toxic comment classification: An in-depth error analysis*. arXiv.org. https://arxiv.org/abs/1809.07572

[2] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014, December 11). *Empirical evaluation of gated recurrent neural networks on sequence modeling*. arXiv.org. https://arxiv.org/abs/1412.3555

[3] de Rosa, G. H., & Papa, J. P. (2022, December 20). *A survey on text generation using generative Adversarial Networks*. arXiv.org. https://arxiv.org/abs/2212.11119

[4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, Oct 11). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. https://arxiv.org/abs/1810.04805

[5] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June 10). *Generative Adversarial Networks*. arXiv.org. https://arxiv.org/abs/1406.2661

[6] Kingma, D. P., & Welling, M. (2013, December 20). *Auto-encoding variational Bayes*. arXiv.org. https://arxiv.org/abs/1312.6114

[7] Li, R., Li, X., Chen, G., & Lin, C. (2020, November 3). *Improving variational autoencoder for text modelling with timestep-wise regularisation*. arXiv.org. https://arxiv.org/abs/2011.01136

[8] Risch, J., & Krestel, R. (2020, January). toxic comment detection in online discussions. https://www.researchgate.net/publication/338798595_Toxic_Comment_Detection_in_Online_Discussions

[9] Sutskever, I., Vinyals, O., & Le, Q. V. (2014, December 14). *Sequence to sequence learning with Neural Networks*. arXiv.org. https://arxiv.org/abs/1409.3215

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023, August 2). *Attention is all you need*. arXiv.org. https://arxiv.org/abs/1706.03762

[11] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016, October 8). *Google's Neural Machine Translation System: Bridging the gap between human and machine translation*. arXiv.org. https://arxiv.org/abs/1609.08144