LLMs as Compression Models

Ryan Hardesty Lewis
rl869@cornell.edu

Abstract

We explore the application of fine-tuned language models, like BERT, for text compression and decompression. By using pre-trained language models and masked language modeling, we develop a system that can efficiently compress text data while maintaining the ability to reconstruct the original text with high accuracy. Our approach involves fine-tuning a BERT model on a masking task and using the trained model to predict masked tokens during decompression. We evaluate our system's performance against traditional compression methods like gzip and discuss the trade-offs between compression ratio and reconstruction accuracy. Preliminary experiments show promising results, with our system achieving higher compression ratios than gzip on short text sequences. We also discuss future directions for improving the system, such as exploring different masking strategies and incorporating additional compression techniques.

1. Motivation

Recently, the New York Times has sued OpenAI, claiming that ChatGPT could spit out entire articles of theirs, thus infringing copyright [4]. OpenAI responded by claiming that the Times had cherry-picked prompts and even inserted parts of their articles to try and get ChatGPT to spit out the rest and the idea that ChatGPT, an inherently generative and random text model, would memorize entire sections of data piece for piece was ridiculous [9]. The sentiment shared by OpenAI is somewhat correct, as generative models are meant to be used to "generate", i.e. create new information, not simply resuscitate trained data. However, researchers from Google Deepmind found similar results, creating adversarial attacks against ChatGPT and other generative models that could spit out training data verbatim [7].

These recent developments encroach on the idea of compression- if we could give a language model a simple prompt like "Othello", would it be possible to simply spit out the entire work of Shakespeare? In doing this, the language model would become the tool of text compression, having stored almost perfect representations of probablistic Shakespeare data in a smaller latent space, thus creating a predictive compression tool. Researchers have long considered the tasks of prediction and compression to be intertwined [6], as one can liken Huffman's encoding algorithm to predicting the smallest byte representations of common characters. Our work aims to expand on the scope of this idea, showing how generative models can indeed be used to improve compression performance compared to traditional compression techniques.

The rapid growth of synthetic data has created a pressing need for efficient compression techniques, particularly in the domain of natural language processing. While traditional compression algorithms like gzip have been widely used, they often struggle to achieve high compression ratios on short text sequences. Recent advancements in pre-trained language models, such as BERT [3], have shown remarkable success in capturing the underlying patterns and semantics of natural language. This presents an opportunity to leverage these models for predictive text compression.

The motivation behind our work is to develop a compression system that can effectively reduce the size of text data while preserving the essential information necessary for reconstruction. By fine-tuning BERT models on our data, we aim to achieve higher compression ratios compared to traditional methods. We note the issue of compounding error in language models [8], as we might prompt "Othello" and have the output devolve into something completely unrelated given a long enough time window.

To combat this, we adopt an approach similar to the New York Times in giving segments of the original data verbatim. This is a masking technique, where we feed in a noisy version of an article, with missing words that must be reconstructed by infilling the data. OpenAI has themselves claimed that training models to infill data is a superior method for learning efficient representations in language processing [1]. In this way, we can constructively denoise the compressed "noisy" data via prediction, which has implications for various applications, such as efficient storage and transmission of text data, error correction in lost network packets, as well as reducing the memory footprint of large-scale language models.

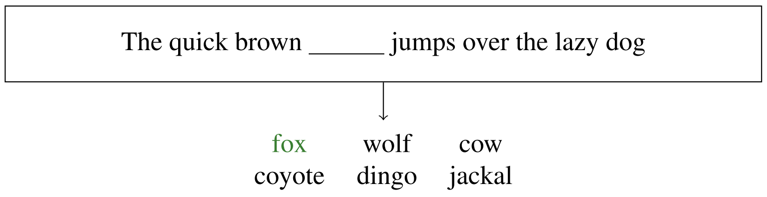

Figure 1. Masked language model architecture. The correct word "fox" is highlighted in green among similar probabilistic words presented for the masked blank.

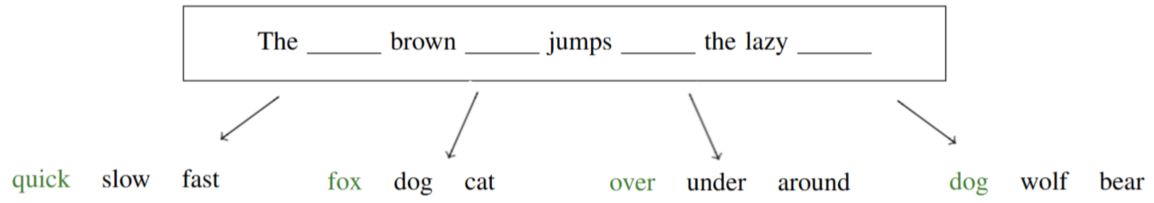

Figure 2. Masked language model architecture with multiple words masked. Each blank has multiple word options, making prediction exponentially harder as more blanks are filled.

2. Method

The masked language modeling (MLM) task, popularized by models like BERT [3], has become a standard approach for pretraining large language models. In MLM, a percentage of input tokens are randomly replaced with a special `<mask>` token, and the model is trained to predict the original tokens based on the surrounding context. Figure 1 illustrates a simple example of the MLM task, where the goal is to predict the masked word ”fox” from a set of contextually relevant candidates. This training objective encourages the model to learn representations that capture the syntactic and semantic relationships between words.

While the traditional MLM approach has been highly successful in pretraining language models for various downstream tasks, it can also be adapted for the purpose of text compression. By strategically masking tokens that are highly predictable given the surrounding context, we can obtain a compressed representation of the text that maintains its essential information. Figure 2 demonstrates this concept with multiple masked tokens, highlighting the potential for using MLM as a foundation for text compression.

Building upon this idea, we propose a novel text compression method that leverages the predictive power of masked language models. Our approach aims to achieve high compression ratios while preserving the ability to reconstruct the original text with minimal loss of information. The trade-off between compression ratio and reconstruction quality can be controlled by adjusting the threshold for masking tokens based on their predicted probabilities. A higher threshold will mask more tokens, yielding higher compression at the cost of potentially more reconstruction errors. By replacing the placeholders with predicted tokens, we reconstruct an approximation of the original text.

Our approach to text compression involves fine-tuning a BERT model on a masking task and using the trained model for compression and decompression. The process can be broken down into the following steps:

1. **Fine-tuning BERT:** We start by fine-tuning a pre-trained BERT model on a masked language modeling task. Given a text corpus, we randomly mask a certain percentage of tokens and train the model to predict the original tokens based on the surrounding context. This allows the model to learn the underlying patterns and relationships within the text.

2. **Compression:** To compress a given text, we tokenize it and randomly mask a subset of the tokens based on a predefined masking probability. The masked tokens are replaced with a special mask token, and the resulting sequence represents the compressed text. The compression ratio can be controlled by adjusting the masking probability.

3. **Decompression:** During decompression, we feed the compressed text into the fine-tuned BERT model. The model predicts the original tokens for the masked positions based on the surrounding context. We replace the mask tokens with the predicted tokens to reconstruct the original text.

To recall how Huffman-esque compression algorithms work, we note that common characters are transformed into smaller byte sequences as they appear more frequently, and thus smaller frequent representations reduce size. In a similar way, our choice of effectively noising our data by taking out random words and instead introducing a masking token

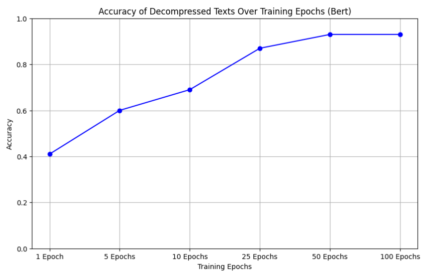

Figure 3. Reconstruction accuracy of BERT model across training epochs

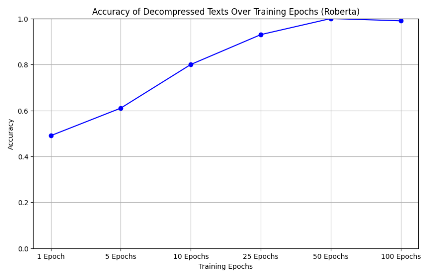

Figure 4. Reconstruction accuracy of RoBERTa model across training epochs

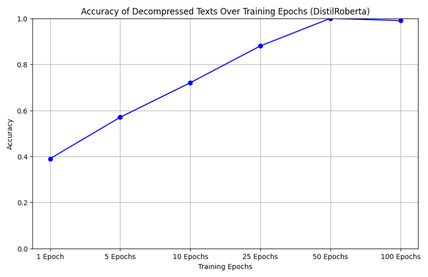

Figure 5. Reconstruction accuracy of DistilRoBERTa model across training epochs

lets us have a much smaller representation, due to the missing words, as well as a single token that can indicate a spot that needs to be predicted by our reconstruction model. We can even further effectively compress our data size by not just masking random words to compress the data, but by masking words and characters that are less frequent in the Huffman encoding. Since we store the noised data in a similar way, we effectively always have a better compression ratio as compared to non-predictive methods as we always store less information, not including model size.

To evaluate the performance of our compression system, we compare it against traditional compression methods like gzip. We measure the compression ratio, which is the ratio of the compressed size to the original size, as well as the reconstruction accuracy, which indicates how closely the decompressed text matches the original text.

3. Experimental Analysis

We conducted preliminary experiments to assess the effectiveness of our BERT-based text compression system. We used a pre-trained BERT model (bert-base-cased) and fine-tuned it on a masked language modeling task using a small dataset of text sequences. Figure 3 shows how we trained the model for multiple epochs, ranging from 1 to 100, to observe the impact on reconstruction accuracy.

We generally observe that increasing the number of fine-tuning epochs generally led to higher reconstruction accuracy. This suggests that the model's ability to predict masked tokens improves with more training, which can turn our lossy compression into lossless compression if we are willing to make a tradeoff with time. We note the time taken with our tiny sample datasets are shown in Table 1.

Dataset Size	Training Runtime	Training Loss	Epochs
5,349 bytes	208 s	0.346	100
165 bytes	17 s	0.354	100

Table 1. Training time and loss for different dataset sizes.

From the table, we can observe that the larger dataset (5,349 bytes) required a significantly longer training time (208.3681 seconds) compared to the smaller dataset (165 bytes, 17.0485 seconds). However, the larger dataset also achieved a slightly lower training loss (0.3464) compared to the smaller dataset (0.3542). This means that increasing the dataset size can lead to better model performance, but at the cost of increased training time.

We wanted to see how the choice of model affects accuracy over epochs and used optimized versions of BERT in RoBERTa and DistilRoBERTa. RoBERTa is a more performant version of BERT, while DistilRoBERTa is a small model that had RoBERTa's knowledge distilled into it. Figures 6 and 7 show that the choice of model significantly affects the accuracy and training time. More recent models, like RoBERTa, can train faster and achieve higher accuracy with fewer epochs compared to BERT.

We can attribute some of these differences to improvements in tokenization algorithms per model. A model's tokenizer plays a crucial role in how it processes and understands text. RoBERTa uses a byte-level BPE tokenizer, which can handle out-of-vocabulary words better than BERT's WordPiece tokenizer. This allows RoBERTa to have a more robust understanding of the text, leading to better performance in the reconstruction task.

The masking algorithm itself can also be tuned to get better results, as it is akin to how Huffman chooses which parts of the dataset to mask for compression. We note that a random masking algorithm does not perform the best, as it can mask certain tokens that are hard to predict, like commas, periods, or transition words. We demonstrate that by switching to a more deterministic masking algorithm, using Spacy to discriminate nouns, verbs, and adjectives, we achieve a lossless compression with much fewer training epochs, as in Figure 6.

After fine-tuning, we applied our compression and decompression methods to Wikipedia texts (File 1, 2, 3). We varied the masking probability to control the compression ratio and always evaluate on long-trained models (100+ epochs) that showcase a 100% accuracy to present results from lossless compression. Our results from BERT masking 15% of our text is available in Figure 7, while masking 50% of text is seen in 8. We see that the more our model is able to mask, the more effective we can compress the data.

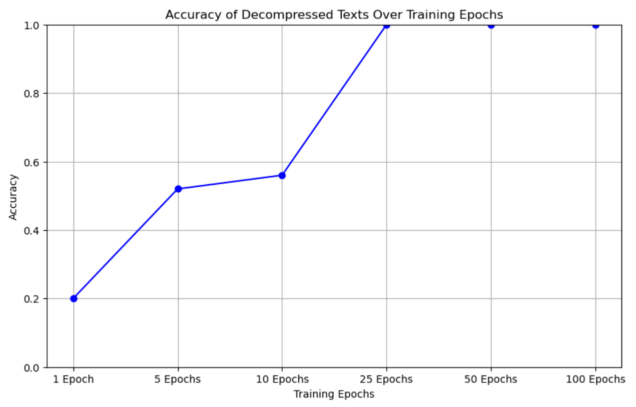

Figure 6. BERT reconstruction accuracy improves with a deterministic masking algorithm targeting nouns and verbs

However, we must note that the inverse holds true for time training and accuracy, as a 50% mask will not achieve the same accuracy as a 15% mask with the same training time, so must be trained for a much longer set of epochs (1000+ epochs). This inherent trade-off is a large downside of the model as it comes to new and unique data, but we believe that with the advent of large language models and in-context learning, the amount of fine-tuning needed to reconstruct such data decreases dramatically.

We note that without fine-tuning on the data that we evaluate against, our baseline remains around 0.2%, or equivalent to around one epoch of training during a fine-tuning. The preliminary results showed that our system achieved higher compression ratios compared to gzip on both short text sequences as well as longer sequences. We note that the compression gains from using a language model are rather insignificant for single byte-size text sequences, where both gzip and our compression algorithm create larger files on account of short sequences being a failure case for Huffman. However, we do note that even within this failure case, the masking of tokens, and thus, their semi-deletion, still allows our method to outperform gzip due to the method's inherent nature of having less unique tokens overall.

4. Discussion

To increase the adoption of natural language in text compression systems, we see several potential paths to increase the efficiency of our current approach:

- **Masking Strategy:** Instead of random masking or noun masking, we can likely use something that we can probabilistically say is more accurate. For example, we can choose mask words based on their prediction probability in the model so far, so words we can easily guess with our current model are more likely to be masked. This would definitively lead to better compression than the current semi-random noun picking.

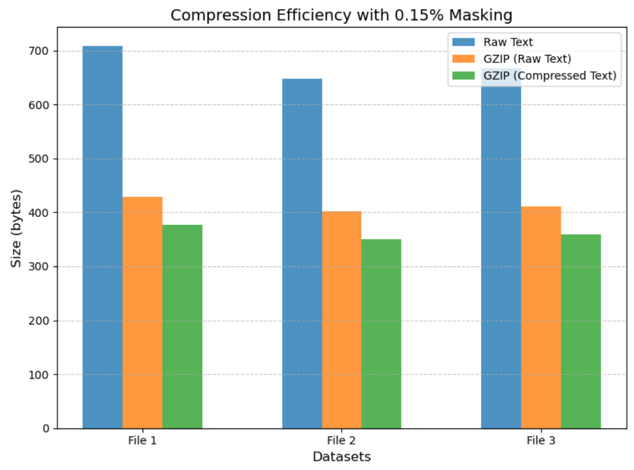

Figure 7. A 15% mask shows slight improvements over gzip

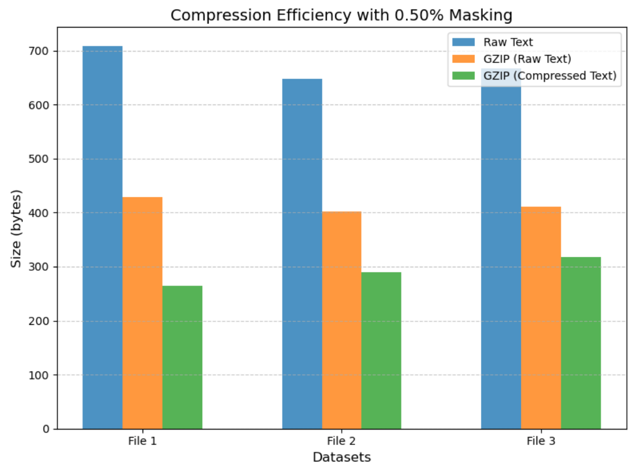

Figure 8. A 50% mask allows almost a 2x reduction in size

- **Latent Space Compression:** We could try to get a better latent space representation, which is usually done by the tokenizer. We can likely use a different tokenizer, like GPT-2, but note that current implementations do not support masking and need to be slightly modified to support our approach. This is akin to how we saw slightly changing to RoBERTa's better tokenizer increased performance.

- **Scaling to Longer Sequences:** Our current experiments focused on short text sequences from Wikipedia. We should make a model that can handle longer sequences and evaluate its performance on larger datasets, but note training time is a significant barrier.

- **Integration with Other Techniques:** We want to see if we can combine the current gzip and BERT system with other compression techniques, such as entropy coding or dictionary-based methods, to double down on our overall compression efficiency.

As for considering the impact of model size, we do not see too much of a need to include it for comparison in this result or future results. This is because a baseline BERT model does not change in size even after fine-tuning, meaning the words were already accounted for within the model weights, with only slight changes occurring during fine-tuning to the existing latent representation.

There are similar veins of research that we would like to compare our model to, as compression using other forms of neural networks, like RNNs, is not unheard of. For instance, next-token prediction has been researched before and deemed to be performant at almost 2x against gzip [5]. However, the lack of properly open-sourced implementations or a standard baseline dataset in other NN-compression papers makes it rather difficult to compare our approach against theirs, other than just comparing the rough compression ratio against that of gzip.

We also see a potential outlet in applying the same prediction modeling technique to similar domains to text, like images. We could train a model that knows an initial image's representation, then noise (mask) over the parts of the image that we can predict easily, such as a celebrity's face or the texture pattern of a wall, and reconstruct that with the trained model. Recent work in the area admits similar results from Google Deepmind, with researchers seeing that they could use language models to similarly compress data in other domains, like images [2].

Delétang et al. [2] also look at the relationship between model size, dataset size, and compression performance. They show empirically that for a given test dataset, there is an optimal model size that balances the improved compression from more parameters with the cost of storing those parameters. Increasing the model size beyond this optimum actually worsens the adjusted compression rate. This connects to our analysis of the inherent trade-off between masking percentage, training time, and reconstruction accuracy. Together, these results highlight the importance of considering model efficiency, not just absolute performance, for practical deployment of language model-based compressors in real-world scenarios.

5. Conclusion

We have presented a compression technique based on the fine-tuning of language models like BERT. By using masked language modeling and noise infilling, our system shows promising results in compressing and reconstructing text data in both lossy and lossless compression. Our preliminary experiments show higher compression ratios compared to traditional methods like gzip on short text sequences and indicates a promising future, where a machine itself could ingest data like a language model, condense it into a latent representation, and a user might retrieve this data with something as small as a single token.

However, this work is still much a work in progress, and there are still several ways to get a higher compression ratio. Better masking strategies, latent spaces, alongside other general improvements can make language models a new state of the art in compression techniques.

The New York Times lawsuit, alongside this research, suggest how general improvements in language modelling can be used to potentially compress data efficiently and serve as a glimpse of a possible future where available compute and storage can be traded off against one another.

References

[1] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle, 2022.

[2] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression, 2024.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Michael M. Grynbaum and Ryan Mac. The times sues openai and microsoft over a.i. use of copyrighted work. *The New York Times*, 2023. Accessed: 31 March 2024.

[5] Michael Herrera and Kasey Luo. Lossless neural text compression, 2023. Available: `https://web.stanford.edu/class/cs224n/reports/custom_116635402.pdf`.

[6] Shay Moran and Amir Yehudayoff. Sample compression schemes for vc classes. *J. ACM*, 63(3), jun 2016.

[7] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023.

[8] J. O' Neill and D. Bollegala. Analysing dropout and compounding errors in neural language models, 2018.

[9] The New York Times Company v. Microsoft Corporation, OpenAI, Inc., et al. Memorandum of law in support of openai defendants' motion to dismiss. `https://tmsnrt.rs/3Ve68rO`, February 2024.

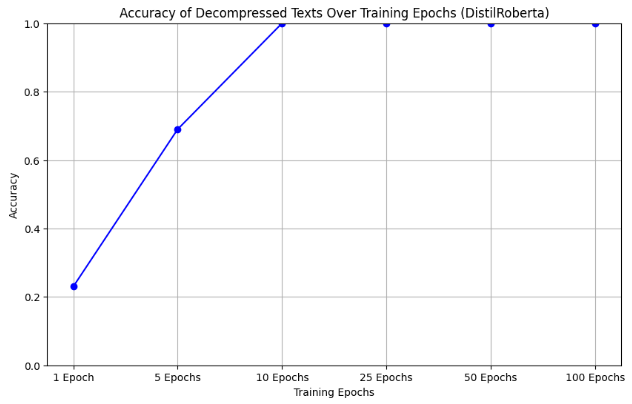

Figure 9. DistilRoBERTa reconstruction accuracy also improves with a deterministic masking algorithm targeting verbs

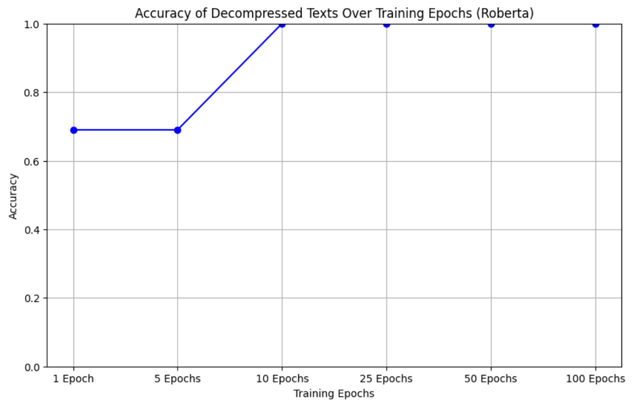

Figure 10. RoBERTa reconstruction accuracy improves dramatically with a deterministic masking algorithm targeting verbs

6. Appendix

All code and datasets for this research are open-sourced at `https://github.com/ryanhlewis/LLMCompression`.

We also include some results that did not fit in the paper. For instance, we show below how other models, such as DistilRoBERTa and RoBERTa showcase similar trends to BERT when Spacy is used to actively discriminate what should and should not be masked based on what a word is. In Figures 9 and 10, we opt to only mask verbs, resulting in incredible performance on Roberta, while DistilBert follows a similar but faster performance to BERT in its general faster trend towards original text equivalence at 100% accuracy. Both models reach 100% accuracy in less than half the typical epochs with a non-random masking technique.

We also present similar 15% and 50% masking graphs for DistilRoBERTa, as by Figures 11 and 12. DistilRoBERTa shows worse compression performance on the 15% masking task, likely attributed to masking common words that gzip had already compressed. However, we note a significant compression increase with a 50% mask, with greater than a 2x compression ratio on the same dataset. The 50% mask, however, comes with the same impediments we described earlier in the paper, as training a model to predict it to a perfect accuracy takes a long time, with more than an hour needed for a few kilobytes, and the time required make such a large reconstruction usually lossy instead of lossless compression. We advocate for future research to search for a Pareto frontier for each of these objectives in finding an ideal compression rate given both time constraints and accuracy constraints. Like language modelling, it seems that compression can be solved with enough compute.

This logic opens questions in the future as to how such a compressor is operated, if like language models, it can only compress and decompress files by sending to a remote powerful server with the ability to predict upon it.

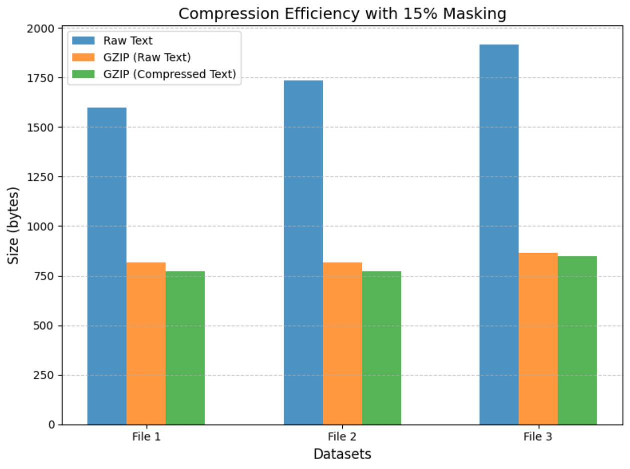

Figure 11. A 15% mask on DistilRoBERTA marginally improves over gzip

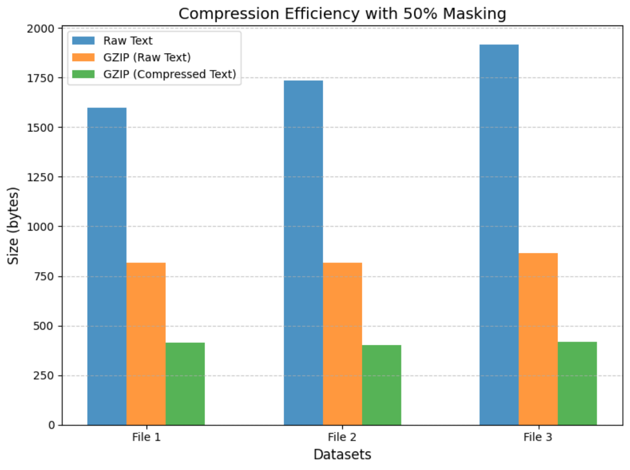

Figure 12. A 50% mask on DistilRoBERTa allows greater than a 2x reduction in size