TextEconomizer: Enhancing Lossy Text Compression with Denoising Transformers and Entropy Coding

By

Mahbub E Sobhani
012-241-0011

Submitted in partial fulfilment of the requirements
of the degree of Master of Science in Computer Science and Engineering

July 2, 2025

Department of Computer Science and Engineering
United International University

Approval Certificate

This thesis titled **TextEconomizer: Enhancing Lossy Text Compression with Denoising Transformers and Entropy Coding** submitted by **Mahbub-E-Sobhani**, Student ID: **012-241-0011**, has been accepted as **Satisfactory** in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on 'Date of approval'.

Board of Examiners

1.

_____ **Supervisor**

Dr. Dewan Md. Farid
Professor, Department of Computer Science & Engineering
United International University.

2.

_____ **Co-Supervisor**

Dr. Riasat Azim
Assistant Professor, Department of Computer Science & Engineering
United International University.

3.

_____ **Head Examiner**

Dr. Muhammad Nomani Kabir
Professor, Department of Computer Science & Engineering
United International University.

4.

_____ **Examiner-I**

Dr. Md. Motaharul Islam
Professor, Department of Computer Science & Engineering
United International University.

5.

_____ **Examiner-II**

Mohammad Mamun Elahi
Assistant Professor, Department of Computer Science & Engineering
United International University.

6.

_____ **Ex-Officio**

Name of the Ex-Officio
Designation
Affiliation

Declaration

This is to certify that the work entitled **TextEconomizer: Enhancing Lossy Text Compression with Denoising Transformers and Entropy Coding** is the outcome of the research carried out by me under the supervision of **Dr. Dewan Md. Farid, Professor, Department of Computer Science & Engineering** and under the co-supervision of **Dr. Riasat Azim, Assistant Professor, Department of Computer Science & Engineering**

Mahbub-E-Sobhani
MSCSE Program
Student ID: 012-241-0011
Dept. of Computer Science and Engineering
United International University
Dhaka, Bangladesh

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

Name of the supervisor
Designation
Affiliation

Abstract

Lossy text compression reduces data size while preserving core meaning, making it well-suited for tasks like summarization, automated analysis, and digital archives where exact fidelity is less critical. Despite the dominance of transformer-based models in language modeling, the integration of context vectors and lossless entropy coding into Seq2Seq text generation remain underexplored. A key challenge lies in identifying the most informative context vectors from the encoder output and incorporating entropy coding into the transformer framework to enhance storage efficiency while maintaining high-quality outputs, even in the presence of noisy text. Previous studies have primarily focused on near-lossless token generation, often overlooking space efficiency. In this paper, we introduce TextEconomizer, an encoder-decoder framework paired with a transformer neural network. This framework utilizes its latent representation to reduce variable-sized inputs by 50% to 80%, without prior knowledge of dataset dimensions. Our model achieves competitive compression ratios by incorporating entropy coding, while delivering near-perfect text quality, as assessed by BLEU, ROUGE, and semantic similarity scores. Notably, TextEconomizer operates with approximately 153 times fewer parameters than comparable models, achieving a compression ratio of 5.39× without sacrificing semantic quality. Additionally, we evaluate our framework by implementing an LSTM-based autoencoder, commonly used in image compression, and by integrating advanced modules within the transformer architecture as alternatives to conventional techniques. Our autoencoder achieves a state-of-the-art compression ratio of 67× with 196 times fewer parameters, while our modified transformer outperforms the autoencoder with a 263-fold reduction in parameters. The TextEconomizer framework significantly surpasses existing transformer-based models in balancing memory efficiency and high-fidelity outputs, marking a breakthrough in lossy compression with optimal space utilization. Additionally, we propose another transformer-based method named RejuvenateFormer for text decompression, addressing prior issues by harnessing a new pre-processing technique and a lossless compression method. Our meticulous pre-processing technique, incorporating the Lempel-Ziv-Welch algorithm, achieves compression ratios of 12.57, 13.38, and 11.42 on the BookCorpus, EN-DE, and EN-FR corpora, thus showing state-of-the-art compression ratios compared to other deep learning and traditional approaches. Furthermore, the RejuvenateFormer achieves a BLEU score of 27.31, 25.78, and 50.45 on the EN-DE, EN-FR, and BookCorpus corpora, showcasing its comprehensive efficacy. In contrast, the pre-trained T5-Small exhibits better performance over prior state-of-the-art models.

Acknowledgements

All the praises and thanks are to Allah, the most gracious and the ever merciful. HE has given me the strength, courage, and patience to carry out this work.

Then, this thesis represents a great deal of time and effort not only on my part but also on the part of my supervisor, Prof. Dr. Dewan Md. Farid and Dr. Riasat Azim.

I am also thankful to my thesis committee members for providing insightful and constructive comments to improve the quality of this thesis.

Then thanks to family and friends.

Publication List

The main contributions of this research are either published or accepted or in preparation in journals and conferences as mentioned in the following list:

Journal Articles

1. Under Review

Conference Papers

1. An Enhanced Text Compression Approach Using Transformer-based Language Models. IEEE TENSYMP 2024

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an overview of the research field of text compression. It outlines the motivation driving this work, and defines the specific problems addressed. This chapter further details the scope and limitations of the research, lists the key contributions, and concludes with an organization of the thesis.

1.1 Overview

- Text compression is a crucial component of data compression, focusing on reducing the volume of textual data while preserving its necessary informational content. This process is crucial for optimizing storage, bandwidth, and computational efficiency in the face of ever-increasing digital information.

- The field is broadly divided into two categories: lossless compression, which allows for perfect reconstruction of the original data, and lossy compression, which achieves higher compression ratios by sacrificing some non-essential details. While traditional algorithms like Lempel-Ziv-Welch (LZW) [5] and Huffman Coding [6] have been prominent in lossless compression, recent advancements have seen a significant shift towards using deep learning and transformer-based models for both lossy and lossless tasks.

- Core concepts of this study includes encoder-decoder frameworks, which form the basics of sequence-to-sequence tasks. Transformer models like BERT [7] and LLaMA [8] have become dominant in natural language processing, and autoencoders, which are neural networks designed to learn efficient data codings in an unsupervised manner.

- Our thesis addresses a particular problem within this domain by focusing on two innovative approaches to enhance text compression. The first approach introduces a highly parameter-efficient, lossy compression framework using denoising transformers. The second approach proposes a novel pre-processing technique for lossless

compression that is incorporated with a transformer-based model for accurate decompression.

1.2 Motivation

- The research is motivated by the exponential growth of digital textual data, which presents significant challenges in storage and transmission efficiency. With an estimated 18.7 billion texts transmitted daily worldwide and 95% of businesses moving to digital documentation [9], the need for robust and efficient text compression solutions is more critical than ever.

- This work is driven by the practical needs of numerous real-world applications. Lossy text compression is particularly valuable for digital archives [10], chatbots [11], email archiving [12], and web search engines [13], where preserving the core meaning is more important than exact textual fidelity. Such technologies enable fast data retrieval and more efficient communication while managing large-scale textual data.

- A primary technological motivation is the limitation of current transformer-based models. These neural network models are powerful, but these models are often characterized by massive parameter counts, leading to high memory complexity and computational costs. This makes their implementation challenging in resource-constrained environments. Furthermore, the integration of efficient pre-processing techniques with classic lossless compression algorithms and modern transformer-based restoration remains an underexplored area with significant potential for improvement.

1.3 Problem Statement

The research presented in this thesis addresses two primary problems in the domain of text compression:

- **The High Memory and Computational Complexity of Transformer-Based Models:** State-of-the-art transformer models, while effective, often struggle with managing memory intricacy due to their large number of parameters. A key challenge lies in developing a framework that can achieve competitive lossy compression ratios and maintain high-quality text reconstruction while operating with significantly fewer parameters than the existing models. This gap highlights the need for more efficient model architectures that do not sacrifice performance due to size.

- **Inefficient Integration of Pre-processing and Decompression Techniques:** There is a lack of a simple yet effective approach that optimally integrates efficient text pre-processing with existing lossless compression algorithms and transformer-based models for text restoration. The potential of combining context vector filtering

and lossless entropy coding within transformer frameworks remains underexplored, presenting a gap in achieving maximum storage efficiency while preserving output quality.

1.4 Scope and Limitations

- The scope of this research is focused on developing and evaluating novel frameworks for the compression and decompression of monolingual English text. It encompasses both lossy and lossless compression paradigms. The study involves the implementation and benchmarking of several neural network architectures, including a custom denoising transformer (TextEconomizer), a lightweight variant with modern components (LLaMAFormer), an LSTM-based autoencoder [14], and a transformer model for text restoration (RejuvenateFormer). The evaluation is performed on multiple benchmark datasets, including PwC [15], WMT14 [16], WMT19 [17], and BookCorpus [18], using a couple of metrics such as BLEU, ROUGE, BERTScore, Perplexity, and compression ratio.

- The research is primarily focused on the English language and does not explore multilingual compression or compression for low-resource languages, which are identified as areas for future work.

- A known limitation is that the experiments were conducted on subsets of the full datasets due to computational resource constraints. Furthermore, the TextEconomizer model is highly efficient, it incurs a negligible reduction in n-gram precision compared to larger, state-of-the-art models. In some experiments, the heavily pre-trained T5-Small baseline model demonstrated superior performance, highlighting the effectiveness of large-scale pre-training.

1.5 Research Contributions

- The original contributions of this thesis are as follows:
  - **New Models, Methods, and Frameworks**
    - Our proposed TextEconomizer, a novel and unified encoder-decoder framework for lossy text compression. This framework uniquely integrates a denoising transformer with the strategic filtering of salient context vectors (termed Kizuki vectors) and lossless entropy coding to achieve high compression ratios with a model that is approximately 153 times more parameter-efficient than existing comparable models.
    - Developed a straightforward yet highly effective pre-processing technique by removing vowels and used alongside the Lempel-Ziv-Welch (LZW) algorithm and this process achieves exceptional lossless compression ratios.

This is further enhanced by the RejuvenateFormer model, specifically prepared for precise text decompression.

- **Enhanced Algorithms and Improved Accuracy**
  - Developed a sophisticated and pragmatic text-noising process tailored to create a wide range of realistic text distortions, which enables the neural models to learn effective denoising and restoration strategies.
  - Achieved state-of-the-art memory compression ratios of up to 67× with an LSTM-based autoencoder on the PwC dataset and up to 13.38× with the LZW-based pre-processing method on the WMT14 EN-DE dataset, significantly outperforming prior deep learning and traditional compression methods.
- **Implementation of a Novel System**
  - Implemented and benchmarked multiple architectures within the proposed frameworks, including LLaMAFormer, a lightweight 50M parameter model that integrates modern components like Rotary Positional Encoding (RoPE) [19] and SwiGLU [20] activation, demonstrating the versatility and efficiency of the core design.
- **Unique Findings and Theoretical Insights**
  - Empirically demonstrated that a large fraction of encoder-generated context vectors can be discarded (using as little as 20-50% of the most salient vectors) with only a marginal impact on the final output quality. This proof provides a practical path to significant computational savings in transformer models.
  - We established that combining a simple, strategic pre-processing step with traditional lossless compressors can produce superior compression ratios compared to more complex, end-to-end neural compression techniques.

1.6 Organization of the Thesis

- **Chapter 1: Introduction.** This chapter introduces the research topic, motivation, problem statement, contributions, and structure of the thesis.
- **Chapter 2: Literature Review.** This chapter presents a comprehensive review of existing work in text compression, including traditional algorithms, neural network-based methods, and the evolution of transformer architectures.
- **Chapter 3: Proposed Methodology.** This chapter details the architectural design and theoretical underpinnings of the proposed frameworks, including TextEconomizer, the Kizuki vector selection mechanism, the LZW-based pre-processing technique, and the RejuvenateFormer model.

- **Chapter 4: Experimental Analysis.** This chapter describes the datasets, data pre-processing steps, evaluation metrics, and hyperparameter configurations used for the experiments. Furthermore, the quantitative and qualitative results, including ablation studies and direct comparisons with state-of-the-art models, to validate the effectiveness of the proposed methods are thoroughly discussed.
- **Chapter 5: Conclusion and Future Work.** This chapter summarizes the key findings and contributions of the thesis and discusses potential directions for future research.

Chapter 2

Background and Literature Review

Text compression is a key component of data compression that reduces the volume of textual data while retaining its core informational content. This process can be either lossless, which allows for complete data recovery, or lossy, which sacrifices certain details to achieve higher compression ratios. The exponential growth of digital information has created significant challenges in storage and transmission efficiency [9], particularly in contexts where exact textual reproduction is not mandatory. Lossy text compression offers numerous practical applications, providing effective solutions for non-critical documents and archiving [10], where exact preservation of every detail is unnecessary. This approach is useful for internal reports, outdated document versions, corporate archives [21], and educational institutions, where retaining key information is crucial, but minor errors or formatting loss are not detrimental to usability. Libraries, digital archives [22], and universities can compress extensive collections, such as research papers or public domain books, saving storage space while ensuring continued accessibility. Furthermore, industries like chatbots [11], email archiving [12], and web search engines [13] benefit from lossy text processing, enabling quicker retrieval and facilitating communication [23]. Thus, lossy text compression emerges as a valuable and efficient tool for managing large-scale textual data, optimizing both storage and retrieval performance.

Image compression harnessing neural networks, especially through transformer, has gained prominence [24], meanwhile myriad approaches have been discovered to reduce textual volume while preserving salient information. Existing research underscores the efficacy of transformer-based models, including BERT [25], LLaMA [26], and ALBERT [27], LSTM [28], in maintaining contextual integrity during decompression across diverse linguistic landscapes. In particular, architectural modifications such as the shared encoder have shown strong performance [29]. Furthermore, chasing the trend of fine-tuning domain-specific pre-trained models and incorporating the LoRA [30] technique has also resulted in praiseworthy results [15]. Cross-lingual augmentation strategies enhance the capabilities of transformer models for languages with diverse resource availability, an aspect

that has not been investigated in the work of [31]. Notwithstanding these advancements, recent studies have encountered limitations. The study conducted by Huang et al. [32] leveraged the operational procedure of arithmetic coding and incorporated it with GPT for lossless text compression, lacking a quest for harnessing general-purpose compressors for an extensive comparison of compression ratios. Additionally, it is important to note that non-autoregressive decoding may not flawlessly recover the original text, and iterative inspection of meaning preservation might be time-consuming [33]. Ge at al. [15] employed a LoRA-configured Llama-2-7b for context compression, albeit with augmented parameters, exacerbating the already prodigious parameter count characteristic of contemporary LLMs. Meanwhile, Qin et al. [34] executed an autoencoding task generating dynamic text segments with residual connections, achieving a mere tenfold compression—a ratio deemed insufficient. Furthermore, Wang et al. [35] provided an autoencoding model trained to reconstruct input texts by means of a combination of token embeddings as a bottleneck, a strategy susceptible to overfitting, while the compression ratio r is lower. Strengthening the fixed-size bottleneck remains a pivotal challenge for Transformer-based large language models (LLMs) due to their intrinsic self-attention mechanism. While prior research ([35, 36, 37]) has explored text compression in LLMs, they frequently grapple with the challenge of mitigating memory intricacy. Despite their capacity for fixed-size latent spaces, LSTM-based autoencoders frequently produce inadequate results in autoregressive decoding tasks. Using these orthodox approaches, we tackle the issue of memory complexity from an alternative standpoint: employing lossy text compression. Subsequently, the domain of text compression has become widespread attention of research, attracting significant interest and contributing to the development of innovative methodologies and datasets. Our extensive study aims to summarize key findings and showcase the evolving landscape of text compression methodologies across different language contexts, including lossy [25, 27, 29, 33] and lossless [26, 31, 32] techniques.

2.1 Transformer-based methods

The advent of text compression has seen the employment of transformer-based methods where GPT, Llama, BART, and even a single-layer transformer have been utilized. Among lossy techniques, Li et al. [25] introduced an innovative method to compress English text by masking less important words and then restoring them using a Transformer-based model, while Rae et al. [36] refined the Transformer architecture [38] by employing a compressive-memory-based approach that consolidates past activations rather than discarding them. In addition, Zhang et al. [39] proposed a new text compression method called L3TC, which combines learning-based probabilistic modeling with efficient architecture by using an RWKV language model [40] instead of a traditional transformer, along with an outlier-aware tokenizer and high-rank reparameterization to achieve high compression rates; however, the study lacks empirical validation on compression ratio over different batch sizes, might struggle with low-frequency words, and has not been thor-

oughly tested for resilience with large-scale data. Furthermore, Li et al. [27] proposed an approach aimed at extracting the core information of the input text by enhancing the text encoding of transformer models with text compression through two distinct strategies: Explicit Text Compression (ETC) and Implicit Text Compression (ITC), the latter producing compressed text features without generating an explicit text sequence by using a non-autoregressive decoder. Moreover, Elias et al. [41] introduced MultiTok, a variable-length tokenization method motivated by the Lempel-Ziv-Welch (LZW) algorithm, which dynamically compresses duplicative phrases into multi-word tokens to achieve up to 2.5× faster training with 30% less data while maintaining performance comparable to BERT and GPT-2 standards. In a related vein, Qin et al. [34] introduced an interesting methodology that leverages the BART [42] encoder to produce highly significant dynamic text segments, termed "NUGGET," by distilling the logits through a feed-forward network. Additionally, our study has revealed that lossless text compression techniques have yielded exceptional results by integrating diverse approaches with transformers; for example, Valmeekam et al. [26] proposed a method that uses the LLM to predict the next token in a text sequence based on a window of past tokens, and Huang et al. [32] introduced a technique that employs the GPT model to calculate probability distributions for each token, thereby representing the entire text with a single number, while Deletang et al. [43] compared predictive models and lossless compressors, recommending the use of large self-supervised language models for compression. Among context-compression-based methods, Chevalier et al. [44] introduced AutoCompressors—a novel strategy that adapts pre-trained language models such as OPT [45] and Llama-2—to compress long contexts into consolidated summary vectors that are generated consecutively and accumulated across segments, thus enabling efficient processing of long documents while retaining essential information. In a similar context, Ge et al. [15] proposed an innovative approach that leverages the capabilities of the Llama model [8] to generate pertinent memory slots through the teacher-forcing mechanism for both autoencoding and machine translation tasks. Furthermore, Mu et al. [46] introduced a technique known as "gisting," which involves training a language model to condense prompts into smaller sets of "gist" tokens, achieving up to 26 times compression, a 40% reduction in floating-point operations (FLOPs), and a 4.2% improvement in processing speed, while maintaining good output quality with minimal loss. Qin et al. [47] introduced DODO, a dynamic context compression method for decoder-only language models that reduces self-attention costs by depicting text with a variable number of hidden states per layer, thereby maintaining strong performance in language modeling, question-answering, and summarization, and serving as both an autoregressive model and a context compressor for subsequent tasks.

2.2 Neural network-based methods

Within LSTM-based autoencoding tasks, Malireddy et al. [37] introduced an indicator vector to signify the presence or omission of each word, thereby filtering less pertinent

words. Concurrently, Tissier [48] proposed an autoencoder-based model that condenses real-valued embedding vectors into fixed-length binary representations, while [49] enhanced storage efficiency by decomposing the embedding layer through matrix factorization and employing lower-rank matrices. Expanding these autoencoding frameworks, Prato [28] developed a recursive autoencoder that encapsulates entire sentences into single embeddings to optimize decodability, rather than merely enriching latent representations. In contrast, Farsad et al. [50] introduced an LSTM-based architecture for joint source-channel coding, embedding sentences into semantic spaces to preserve information and outperform traditional methods under low bit allocations and error rates. Parallelly, in lossless compression-focused research, Goyal et al. [51] leveraged recurrent neural networks with arithmetic coding for lossless compression, achieving near-ideal results on synthetic datasets and surpassing Gzip on real-world data. Further advancing neural compression techniques, Lara et al. [52] proposed an innovative text compression method combining deep neural networks with Canonical Huffman Codes, reducing message size by up to 30% compared to conventional algorithms. Expanding the application scope of such frameworks, Geleta et al. [53] introduced a variational autoencoder (VAE)-based model for population genetics, compressing single nucleotide polymorphism (SNP) data losslessly while generating synthetic genomes that preserve genetic diversity patterns.

2.3 Denoising Autoencoder-based methods

Recent research has demonstrated the versatility of denoising autoencoders (DAEs) in enhancing natural language processing tasks through diverse architectural and methodological innovations. Shen et al. [54] proposed extending adversarial autoencoders (AAEs) with a denoising objective to improve latent space geometry, introducing the Denoising Adversarial Autoencoder (DAAE), which reconstructs original sentences from perturbed versions to encourage the encoder to map semantically similar texts closer in the latent space. Lopez et al. [55] developed a three-phase framework combining denoising autoencoders with contrastive learning, alongside a novel data augmentation method for supervised contrastive learning to mitigate dataset imbalance; their approach adapts pre-trained Transformer models for classification tasks, enhancing performance by aligning representations with the target data distribution before fine-tuning. Another innovative approach, SUNDAE, introduced by Savinov et al. [56], employs step-unrolled denoising autoencoders to iteratively refine token sequences, achieving state-of-the-art results in machine translation and competitive performance in unconditional language modeling without relying on autoregressive architectures. Furthermore, Freitag et al. [57] explored unsupervised text generation from structured data by framing structured inputs as noisy targets, leveraging sequence-to-sequence DAEs to reconstruct fluent sentences; their method, which introduces controlled noise during training, proved effective across multiple languages and domains without requiring labeled data.

2.4 Neural network-based image compression methods

We extended our in-depth exploration into image compression, finding that neural network-based compressors produce comprehensive outcomes when compressing images. Building on this, VAE-based techniques have seen notable advancements: Kingma et al. [58] efficiently transformed latent variable models into lossless compression methods by leveraging the Bits-Back with Asymmetric Numeral Systems (BB-ANS) framework proposed by Townsend et al. [59]. In parallel developments within quantization-based approaches, Wang et al. [60] introduced Integer-Only Discrete Flows (IODF), which utilize 8-bit quantization and learnable binary gates to eliminate redundant convolution filters during inference. Similarly, Theis et al. [61] advanced rate-distortion optimization through compressive autoencoders (CAEs), achieving performance rivaling JPEG 2000 via differentiable quantization and entropy estimation. Expanding the scope of hybrid architectures, Liu et al. [24] proposed a Parallel Transformer-CNN Mixture (TCM) block, combining CNNs for local features and transformers for non-local features, while enhancing entropy modeling via a channel-wise auto-regressive model with a Swin Transformer-based attention module. Further innovations in spatial adaptation include the work of Zou et al. [62], which integrated window-based attention blocks into CNN and transformer models to optimize bit allocation in high-contrast regions. Moreover, Bai et al. [63] demonstrated a transformer-based framework that enables direct classification from compressed features by replacing ViT's patchify stem with a CNN encoder, enhancing both reconstruction and downstream task accuracy. Finally, Lu et al. [64] bridged architectural efficiency and performance by integrating Swin Transformers [65] with convolutions, achieving state-of-the-art compression with fewer parameters. Collectively, these studies highlight the dynamic growth of learned image compression, driven by synergies between quantization, entropy modeling, and hybrid architectures.

Transformer-based techniques, though superior in autoregressive decoding evaluation, encounter memory constraints due to a large number of parameters. Despite these successes, the use of latent representations within transformers for text compression is still largely unexplored, and there has been limited research on filtering context vectors before passing them to the decoder. To the best of our knowledge, no existing work proposes a denoising transformer framework for autoencoding that eliminates noise, utilizes a filtered latent space based on contextual coherence to generate high-fidelity outputs, and incorporates lossless entropy coding to enhance compression. This distinctive combination establishes the novelty of our approach to addressing these gaps.

Chapter 3

Proposed Method

We propose a seq2seq encoder-decoder framework for English autoencoding that seamlessly integrates any neural network-based method while confirming memory efficiency and high-quality output. We assessed our framework through a tripartite approach. We first implemented our transformer-based model, optimized to select the most salient context vectors during training, termed TextEconomizer. Second, we replaced conventional transformer components with Rotary Positional Encoding, RMSNorm, and SwiGLU activation functions to create LLaMAFormer, which similarly targets the most critical context vectors. Third, we constructed an LSTM-based autoencoder to assess the performance of sequential methods in autoencoding tasks. All three approaches leverage our innovative, pragmatic text-denoising process to guide the autoencoding workflow. The complete English autoencoding framework is illustrated in Fig. 3.1. The subsequent sections delineate the problem formulation and the components of our proposed TextEconomizer and RejuvenateFormer. Moreover, we propose RejuvenateFormer, another seq2seq encoder-decoder transformer-based model, which is superior in compression ratio evaluation, utilizing more straightforward pre-processing steps, harnessing entropy coding. The complete model architecture is depicted in Fig. 3.2.

3.1 TextEconomizer

Our model is a tweaked version of the [38] style sequence-to-sequence (seq2seq) transformer architecture, specifically designed for English text compression and autoencoding tasks. The preference for a vanilla transformer architecture was inspired by the success of recent transformer-based models, including BART, LLaMA, and GPT, which have achieved state-of-the-art results on autoencoding benchmarks such as WMT19 and the PWC test set. Our model includes a stack of six encoder and decoder blocks, where individual blocks contain self-attention mechanisms and feedforward neural networks. Below, we provide detailed descriptions of the encoder and decoder blocks.

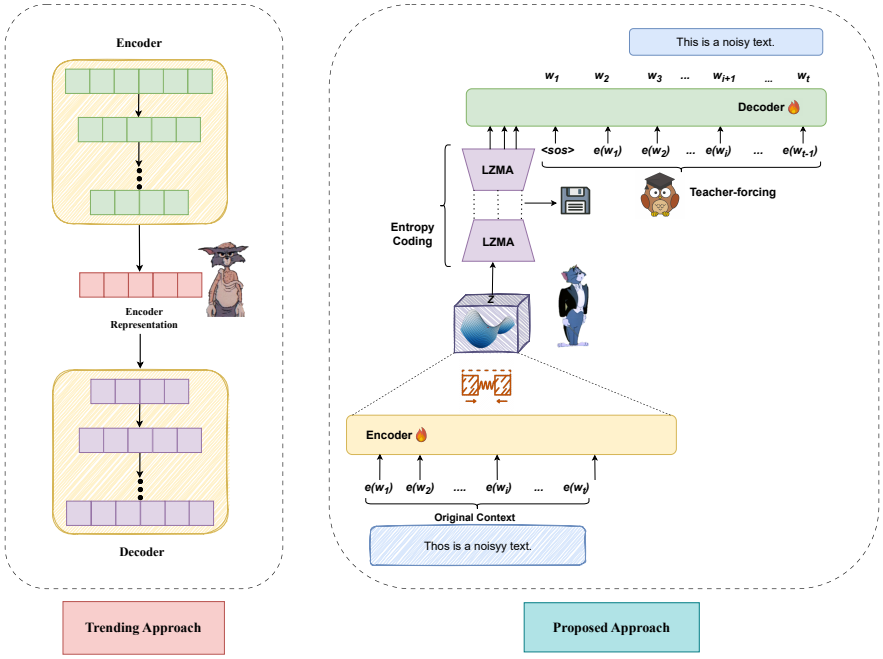

Figure 3.1: **(Left)** Trending Transformer-based approach that generates a latent representation with dimensions identical to the input. **(Right)** Our proposed framework optimally picks a consolidated latent representation for transformer-based methods, while employing a fixed-size latent space for LSTM-based autoencoders, thereby enhancing memory efficiency and preserving salient features. This figure visually distinguishes the structural and functional differences between the standard Transformer-based approach and our advanced framework, highlighting the improvements in computational efficiency, memory utilization, and performance achieved by our method.

3.1.1 Problem Formulation & Overview

Consider input sentence $S = \{S_1, S_2, \ldots, S_{N-1}, S_N\}$, where N denotes the word count. The noise injection process ($\aleph(\cdot)$) adroitly introduces strategic realistic noise through a meticulous automated supervision protocol. Subsequently, we scrutinize the juxtaposition of token pairs, $X_I = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ and $Y_I = \{y_1, y_2, \ldots, y_{k-1}, y_k\}$, where X_I epitomizes the input sequence with multifarious noise types, while Y_I embodies the pristine target sequence. The erroneous sentence X_I is tokenized using a pre-trained tokenizer $\tau(\cdot)$. This process produces a tokenized representation of X, denoted as $X = \{x_{\tau_1}, x_{\tau_2}, \ldots, x_{\tau_{n-1}}, x_{\tau_n}\}$, where x_{τ_i} corresponds to the numerical value of the i-th token if the word is present in the vocabulary. If the word is not present, an $\langle\text{unk}\rangle$ token is used instead. The tokenized sequence is then passed to the encoder $E(\cdot)$, which processes it to generate context vectors for each subword in the sentence. This raises an important query: are these context vectors truly indispensable? To address this question, we conducted a meticulous filtering process, denoted by the $\triangledown$, wherein only the most salient context vectors—referred to as Kizuki vectors, deemed the most significant—are allowed to proceed to subsequent stages. Kizukis are collectively represented as $\mathcal{Z}$. This latent representa-

tion $\mathcal{Z}$ is further subjected to compaction through the Lempel–Ziv–Markov compressor $\mathcal{LC}(\cdot)$ to facilitate parsimonious storage on a hard disk, whereupon memory-related computations are conducted. The Lempel–Ziv–Markov decompressor $\mathcal{LD}(\cdot)$ reconstitutes the compressed representation to a form indistinguishable from $\mathcal{Z}$. The decoder $D(\cdot)$ uses this latent representation, incorporating the teacher forcing concept, to remove the noise and synthesize the correct sentence non-autoregressively during training. This pipeline exhibits seamless integration capabilities with any Transformer-based encoder-decoder architecture. The entire procedure can be encapsulated in the following mathematical formulation:

$$\hat{Y} = D((\mathcal{LD}(\mathcal{LC}(\nabla(E(\tau([X_I]), W^E)))), D_{out}^{t-1}), W^D) \tag{3.1}$$

3.1.2 Encoder

We transform each sentence into a sequence of tokens $X = \{x_1, x_2, \ldots, x_n\}$ and assign discrete values to each subword. Here, n is the sequence length. To ensure consistent input dimensions for every input sequence, we incorporate padding into X. Subsequently, each token x_i is passed through an embedding layer E, which transforms the discrete values into continuous vectors represented by a trainable matrix $E_{x_i} = Embedding(x_i)$. Through backpropagation during training, these embeddings are fine-tuned to minimize the loss. To preserve the positional order of the tokens in a sentence, we incorporate positional encoding PE for each token x_i, denoted as PE_{x_i}. This positional encoding is added to the embedding, resulting in a composite embedding $Z_{x_i} = E_{x_i} + PE_{x_i}$. The composite embedding is then fed into a stack of K identical encoder layers, which are built upon two main components: a position-wise feed-forward network and a multi-head self-attention mechanism. The self-attention mechanism is the key segment of the Transformer architecture. It computes attention scores between all pairs of tokens in the input sequence X using three learnable matrices: query (Q), key (K), and value (V). These matrices capture the contextual dependencies between tokens in the sequence. Specifically, the self-attention mechanism calculates a weighted sum of the hidden states for each token, where the weights are based on the similarity between the token and all other tokens in the sequence. This allows the encoder to focus on the most relevant parts of the input sequence for each token, taking into account the context in which it appears. The mathematical expression of self-attention is defined as follows [38]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.2}$$

Subsequently, the dimension of the keys denoted as d_k, is used to scale the resultant scores for standardizing overall variance and stabilizing gradients. To enhance the model's ability to capture multiple relationships and increase its expressive power, the transformer employs multi-head self-attention. Each head independently computes self-attention, allowing the model to focus on different parts of the input sequence simultaneously. The outputs of all heads are concatenated and linearly transformed to produce the final output.

The self-attention for each head is calculated as:

$$\text{Head}_i = \text{Attention}(Q_i, K_i, V_i) \tag{3.3}$$

where $Q_i = ZW_i^Q$, $K_i = ZW_i^K$, and $V_i = ZW_i^V$ represent the query, key, and value projections for the i-th head. The final multi-head self-attention output is calculated as:

$$MultiHead(Q, K, V) = Concat(Head_1, Head_2, \ldots, Head_h)W^O \tag{3.4}$$

Here, W^O is a learnable weight matrix that merges the outputs of all heads. Following the multi-head self-attention mechanism, the output is passed through a position-wise feed-forward neural network (FFN). This FFN applies a non-linear transformation to the output of the self-attention mechanism, enhancing the model's capacity to capture complex relationships between tokens. Specifically, the FFN consists of two linear transformations with a rectified linear unit (ReLU) activation function in between, introducing non-linearity and enriching the representation of the input sequence. The output of each encoder layer, which integrates contextual information from the multi-head self-attention mechanism and the non-linear transformations of the FFN, is sequentially passed to the next of the K identical encoder layers. This stacking of layers enables the transformer encoder to progressively construct a rich, contextualized representation of the input sequence X, capturing both local and global dependencies that are critical for understanding the meaning and context of each token within the sentence. Ultimately, the final encoder layer produces a sequence of hidden states that effectively encapsulates the semantic and contextual information of each token in the input sentence.

3.1.3 Kizuki Selector

To optimize the decoding process and reduce redundant computations in the cross-attention layer of the decoder, we leverage the fact that contextual token embeddings intrinsically capture the semantics of their surrounding text. By analyzing the attention scores computed between all pairs of token embeddings, we observe that some embeddings have very low attention scores, indicating their minimal contribution to the decoding process. To address this inefficiency, we propose a selective mechanism for identifying the most informative context vectors. Specifically, given a set of contextual token embeddings $Z = \{z_1, z_2, \ldots, z_n\}$, where $z_i \in \mathbb{R}^d$ represents the i-th embedding and n is the total number of tokens, we compute a probability distribution over these embeddings using a softmax function with a dynamic temperature T:

$$p_i = \text{Softmax}(z_i; T) = \frac{\exp(\phi(z_i)/T)}{\sum_{j=1}^{n} \exp(\phi(z_j)/T)}, \tag{3.5}$$

where $\phi(z_i)$ is a scoring function that quantifies the relevance or informativeness of the i-th embedding. The temperature T is dynamically adjusted to optimize token selection based on the sequence length, enhancing both computational efficiency and contextual

relevance. From this probability distribution, we select the top k most informative embeddings, referred to as **Kizuki**, denoted as $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k\}$, where $k = \lceil r \cdot n \rceil$ and r is the reduction ratio ($0 < r \leq 1$). The selection procedure can be represented as:

$$\mathcal{E} = \text{TopK}_k \left(\{ z_i \mid p_i > \theta \} \right), \tag{3.6}$$

where θ is a threshold determined by the top-k selection criterion. These Kizuki vectors are then further compressed leveraging the Lempel-Ziv-Markov chain Algorithm (LZMA) [66] algorithm to enhance memory efficiency, which works by using a sliding window to find repeated patterns in the data and substitute them with shorter codes, then applying a range encoder that calculates how often things appear and gives shorter codes to the common stuff and eventually producing a compressed representation that can be reversed back to its actual form. The compressed representation is then reversed back to the Kizuki vectors, which are then used as Key (K) and Value (V) vectors in the decoder's cross-attention layer $K, V = \mathcal{E}$. This approach ensures that only the most semantically significant embeddings are included in the decoding process, thus reducing computational redundancy while preserving essential contextual information. By focusing on high-impact tokens, Kizuki enhances both computational efficiency and the model's ability to capture crucial contextual relationships during decoding.

3.1.4 Decoder

The target sequence, denoted as $Y = \{y_1, y_2, \ldots, y_m\}$, where m portrays the sequence length, is first processed through an embedding layer. This layer transforms discrete token values into continuous vector representations using the mapping $E_{y_i} = \text{Embed}(y_i)$. To keep positional information within the sequence, a positional encoding PE_{y_i} is incorporated into the token embedding, resulting in the combined representation $Z_{y_i} = E_{y_i} + PE_{y_i}$. The combined embeddings Z_y are thereafter passed through L identical decoder layers. Each decoder layer consists of two primary components: a masked multi-head self-attention mechanism and a position-wise feed-forward network. The masking in the decoder assures that each token can only attend to prior tokens, stopping access to future tokens during the generation process. The masked multi-head self-attention mechanism follows the same computational formulation as the encoder's self-attention mechanism (3.9). This sub-layer enables the decoder to grasp dependencies among tokens within the output sequence, enriching contextual understanding. Following the self-attention mechanism, the decoder applies a cross-attention layer to compute attention over the Kizuki vectors $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k\}$. This multi-head attention layer allows the decoder to attend to the top-k contextualized representations of the input sequence while processing the output sequence, thereby facilitating adequate alignment between input and output tokens. Subsequently, a position-wise feed-forward network is applied to enhance the learned representation through a non-linear transformation. This transformation includes residual connections and layer normalization, consistent with the procedure utilized in the

encoder's feed-forward network. These operations distill the learned representation and contribute to the generation of the final output sequence. During training, the decoder employs a technique known as teacher forcing, wherein the ground-truth token from the earlier time step is fed as input at each decoding step. In contrast, during inference, the decoder produces the output sequence sequentially, predicting the most probable token at each step while considering both the previously generated tokens and the Kizuki vectors. The processed representation undergoes L identical decoder layers, ultimately producing the refined target sequence.

3.2 RejuvenateFormer

RejuvenateFormer is a novel transformer-based architecture designed for text decompression. We developed this to address a significant gap in the text compression field. The lack of an effective approach that integrates lossless compression algorithms with transformer-based methods for text restoration is the key factor to facilitate our research. The model architecture is depicted in Fig. 3.2.

3.2.1 Problem Formulation & Overview

Consider input sentences $S = \{S_1, S_2, \ldots, S_N\}$, where N is the number of character. The vowel remover ($VR(\cdot)$) takes the input sentence and removes the vowels. The Lempel–Ziv–Welch compressor ($LZWC(\cdot)$) takes the sentence and the compressed representation is stored in a hard disk, whereupon calculations regarding length are computed. The Lempel–Ziv–Welch decompressor ($LZWD(\cdot)$) decompresses the compressed representation. Subsequently, consider pair of tokens, $SRC_I = \{src_1, src_2, \ldots, src_n\}$ and $TGT_I = \{tgt_1, tgt_2, \ldots, tgt_k\}$, where SRC_I represents the input sequence that has no vowel, and TGT_I represents the target sequence with vowel restored. Erroneous input sequence SRC_I is tokenized using a subword tokenizer ($T(\cdot)$) and then fed into the encoder ($E(\cdot)$) and generates context vectors of the sentence, denoted as $CV = \{CV_1, CV_2, \ldots, CV_{512}\}$. The decoder ($D(\cdot)$) utilized the context vector CV, alongside the previously decoder generated tokens to generate the correct sentence autoregressively. The pre-processing can integrate seamlessly with any transformers-based encoder-decoder models. The entire procedure can be mathematically abbreviated as follows:

$$SRC_I = LZWD(LZWC(VR(S))) \tag{3.7}$$

$$\hat{TGT} = D((E(T([SRC_I]), W^E), D_{out}^{t-1}), W^D) \tag{3.8}$$

3.2.2 Lempel-Ziv-Welch

Lempel-Ziv-Welch (LZW) compression forms a dictionary of symbols, replaces repeated substrings with dictionary indices, and updates the dictionary during compression. During

decompression, it initializes, reads, retrieves, and updates the dictionary to reconstruct the original data.

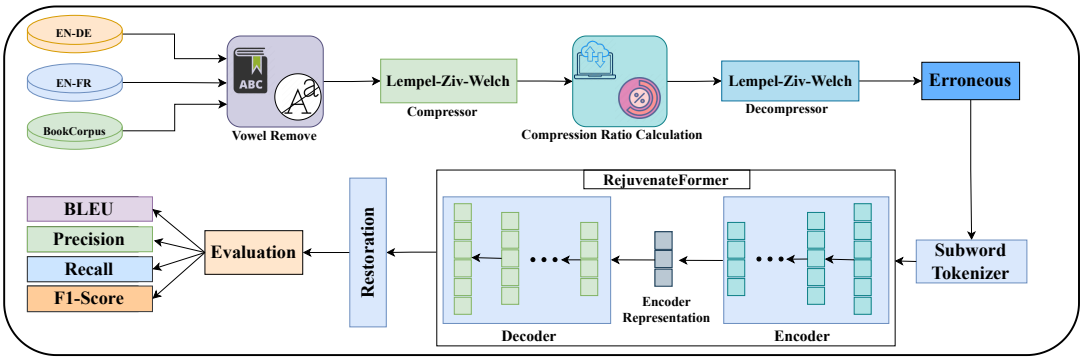

Figure 3.2: **(Top)** Each corpus undergoes vowel removal, and the compression ratio is calculated using the compressed representation. The text is then reverted to its earliest form without vowels. **(Bottom)** After tokenization, the RejuvenateFormer is trained on each corpus, to proficiently generate expected outcomes.

3.2.3 Vowel Removal and Compression

In our text processing pipeline, the function $(VR(\cdot))$ removes vowels from input sentences, producing a text devoid of vowels. Subsequently, the Lempel-Ziv-Welch compressor $(LZWC(\cdot))$ generates a compressed representation of the text, which is then used to calculate the length optimization ratio. Lastly, the compressed representation is decompressed using the $(LZWD(\cdot))$, restoring the original text without vowels, which is then fed into our transformer model to recover the vowels.

3.2.4 Encoder

We transform each sentence into a sequence of tokens $X = \{x_1, x_2, \ldots, x_n\}$ and assign discrete values to each subword. Here n is the sequence length. We confirmed the same input dimension to every input sequence by incorporating padding to X_i. Subsequently, each token x_i went through an embedding layer E to transform the discrete values to continuous vectors which represent a trainable matrix $Ex_i = Embedding(x_i)$. Through backpropagation during training, these metrics are fine-tuned to minimize the loss. To preserve the positional order of the tokens in a sentence we incorporated positional encoding PE for each token x_i, denoted as PEx_i with the embedding. This composite embedding, represented as $Zx_i = Ex_i + PEx_i$ is then fed into the stack of K identical encoder layers, which are built upon two main components: a position-wise-feed-forward network and a multi-head-self-attention layer. Furthermore, the self-attention mechanism calculates an attention score for each token by looking into all other tokens in a sequence X using three learnable vectors, query (Q), key (K), and value (V) for capturing the contextual dependencies in a sequence of tokens. The mathematical expression of self-attention is defined

as follows [38]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.9}$$

The position-wise-feed-forward mechanism introduces non-linearity to the encoder through two linear transformations by incorporating ReLU, which is a non-linear activation function for enhancing the obtained representation. The output of each encoder layer is propagated sequentially to K identical layers which output a rich contextualized representation of the input sequence X covering local and global dependencies.

3.2.5 Decoder

The target sequence, $Y = \{y_1, y_2, \ldots, y_m\}$, where m is the sequence length, undergoes through an embedding layer $Ey_i = Embed(y_i)$ for converting tokens with discrete values into continuous vectors. For preserving positional information of tokens in a sequence, positional encoding PEy_i in blended with embedding Ey_i, and the resulting embedding is $Zy_i = Ey_i + PEy_i$. This combined embedding is passed through L identical decoder layers. The decoder layers are composed of two main components: a position-wise-feed-forward layer and a masked multi-head-self-attention layer. Masking is incorporated in the decoder so that it can not generate the current token by attending the future tokens. Following this distinction, the computational process of masked-multi-head-self-attention follows the same equation (3.9) as the encoder's muti-head-self-attention. Furthermore, the position-wise-feed-forward network enhances the learned representation by following the same procedure as the encoder's position-wise feed-forward network. The acquired learned representation undergoes through L identical decoder layers and produces the fine-graded target sequence.

Chapter 4

Experimental Analysis

4.1 Dataset

We outline the datasets used in our experiments, highlighting their fundamental characteristics. These datasets have been meticulously curated to uphold semantic coherence and syntactic diversity, ensuring their exemplary quality for research purposes. For the autoencoding task, we included only the columns containing English text in our experiments.

WMT14 English-French (EN-FR) [16] The WMT14 dataset is a widely-used benchmark for machine translation, originally containing approximately 36 million sentence pairs from sources such as Europarl v7, Common Crawl, and News Commentary. Due to computational limitations, we used a subset of the EN-FR corpus: 600K training pairs, 3K validation pairs, and 3K test pairs. Despite its reduced size, this subset remains effective for evaluating translation quality.

WMT14 English-Germany (EN-DE) The dataset contains a total of 1.5M source-target pairs after our thorough pre-processing. We separated the corpus into training (100K), validation, and test sets, keeping newstest2013 as the validation set and newstest2014 as the data in the test set. Accordingly, the resulting training, validation, and test sets comprise 100000 (100K), 3000 (3K), and 3003 (3k) source-target pair,s respectively.

WMT19 English-Chinese (EN-ZH) [17] The WMT19 dataset provides a diverse EN-ZH parallel corpus. We selected a subset of 600K training pairs and 3.98K test pairs. This subset ensures a robust evaluation of translation performance while maintaining topic coverage.

Prompt-With-Context (PWC) [15] The PWC dataset assesses models in context-dependent settings. It has approximately 242K training instances and 18.1K test instances. To enhance robustness testing, we systematically added noise to the original sentences, simulating real-world noisy input scenarios.

BookCorpus [18] The BookCorpus dataset comprises text from books, offering diverse linguistic styles and contexts. We introduced synthetic noise to simulate corrupted data

and partitioned the dataset into 1M training pairs and 20K test pairs. This setup evaluates model generalization in autoencoding tasks.

4.2 Data Preprocessing

From the WMT19 corpus, we extracted the English sentences from the Chinese-to-English pairs. We also obtained English text from WMT14's English-to-French pairs. For the PwC dataset, we isolated the answer column. Since BookCorpus is monolingual, no extraction was needed. This systematic approach produced an English-centric corpus across all datasets. In our text preprocessing, we delineate a comprehensive character set containing 80 frequently occurring English characters, denoted as $DC = \{DC_1, DC_2, \ldots, DC_{80}\}$. This set is augmented by 14 frequently occurring punctuation marks in English, represented as $PM = \{PM_1, PM_2, \ldots, PM_{14}\}$, and a space character SP. The resulting amalgamated set of 95 English characters is defined as $C = \{DC + PM + SP\} = \{C_1, C_2, \ldots, C_{95}\}$. Subsequently, we consider each sentence in our corpus, denoted as $S = \{S_1, S_2, \ldots, S_N\}$, where N illustrates the total number of characters in the sentence. We then employ an iterative technique, examining each character $S_i \in S$ and systematically eradicating any character not present in our predefined character set C. This meticulous preprocessing assures a standardized and sophisticated textual dataset for subsequent neural network processing. Furthermore, for RejuvenateFormer beneath the WMT14 dataset, we consider English-to-German and English-to-French language pairs. We only extracted the column containing English text from both the dataset. In the BookCorpus dataset, it was not necessary to do this procedure as it has a single column containing English text. Then, we identified 5 vowel letters, in both capital and small letters, that appear in the English text. These vowel letters are denoted as $VC = \{VC_1, VC_2, ..., VC_{10}\}$. Next, we take every sentence indicated as $S = \{S_1, S_2, ..., S_N\}$, where N indicates the number of characters in the sentence. We traverse through each of the characters $S_i \in S$ and remove any character that is present in the character set VC. By following the above process, the dataset has been constructed in source-target pairs. In this source-target pair, the source sentence has no vowel letters, while the target sentences are corrected versions with missing vowel letters.

4.3 Data Augmentation

We introduce a sophisticated noise injection technique to introduce controlled linguistic variability. This process operates on the premise that each sentence constitutes a finite set of lexical units, denoted as $S = \{W_1, W_2, \ldots, W_{N-1}, W_N\}$ where N is the length of the sentence such that $N \in Z^+$. The proposed noisy text corruption process aims to forge an altered version of the input sentence S that preserves its semantic meaning while incorporating realistic linguistic perturbations. The process commences with identifying named entities $NE = \{e_1, e_2, \ldots, e_m\}$ utilizing a Named Entity Recognition

(NER) model. Subsequently, the sentence undergoes identifying part-of-speech (POS) tags $T = \{(W_i, t_i), (W_2, t_2), \ldots, (W_N, t_N)\}$ are assigned. Auxiliary verbs $V_a \subseteq S$ are probabilistically omitted with a probability P_{aux}, constrained to words where $W_i \in V_a$ and $t_i \in V^*$, where V^* encompasses all verb forms. Consequently, we obtain the modified set $\tilde{S} = \{\tilde{W}_1, \tilde{W}_2, \ldots, \tilde{W}_{M-1}, \tilde{W}_M\}$. The corruption process is controlled by a normally distributed corruption probability $p_c \sim \mathcal{N}(\mu_p = 0.6, \sigma_p^2(\sigma = 0.1))$, which is bounded by a maximum corruption threshold $p_{max} = 0.5$ to ensure the degree of alteration remains within acceptable limits. Given p_c, we determine the number of words to be corrupted as $k = \lceil M \times p_c \rceil$. These words are carefully chosen to avoid consecutive corruptions, preserving the sentence structure. For each chosen word $\tilde{W}_i$, the corruption method is selected based on POS tags and whether $\tilde{W}_i \in NE$ or critical nouns and verbs $C = \{c_1, c_2, \ldots, c_l\} \subset \tilde{S}$. Specifically, if $\tilde{W}_i \notin NE$ and $t_i \in \{P_{nouns}, P_{verbs}, P_{adj}\}$, a contextual synonym $W_i^{'}$ is generated using a masked language model with a $p_{mlm} = 0.5$ probability, unless $W_i^{'} = \tilde{W}_i$. Alternatively, spelling augmentation is propagated $p_{spelling} = 0.3$ of the time, while random word substitution is employed in the remaining $p_{sub} = 0.2$ of cases. Words that do not fall into the categories of nouns, verbs, or adjectives are subjected to typographical alterations through character interchanges or replacements. Punctuation corruption is then applied with a $p_{punc} = 0.2$ probability per word, eradicating existing punctuation or introducing new punctuation marks. This meticulous process pinnacles in a corrupted sentence $S^{'}$ that mimics natural language errors, echoing common mistakes while preserving the essential meaning of the original sentence S.

4.4 Corpus Statistics

In our curated English corpus, characterized by intended noise injection, we have selected $\approx$ 246K, 600K, 600K, and 1M source-target pairs from the PwC, WMT14, WMT19, and BookCorpus datasets, respectively, due to resource limitations. Within these pairs, the source sentences undergo a meticulous text noise technique, while the target sentences serve as pristine, noise-free counterparts. To achieve this, we systematically introduced perturbations into each sentence, as outlined in Section 4.3. Furthermore, the corpus exhibits the following linguistic statistics: PwC demonstrates a minimum of 1, a maximum of 180, and a mean of 35.52 words per sentence; WMT14 ranges from 1 to 72 words, averaging 21.68; WMT19 spans from 1 to 137 words, with a mean of 12.67; and BookCorpus holds sentences varying from 2 to 150 words, averaging 15.13 words per sentence.

4.5 Hyperparameters

The hidden dimension is kept as 512 through all the encoder and decoder layers for maintaining consistency. To maintain the model's depth and capacity the number of neurons is kept at 2048 for the feed-forward layer and a 0.1 dropout ratio is applied to prevent overfitting. To maintain the efficient computation and non-linearity over the network

we have incorporated ReLU. The model went through 50 epochs with 5×10^{-5} learning rate incorporating AdamW optimizer. We incorporated the categorical cross-entropy loss function for the optimization process, which leads the model towards desired translations.

4.6 Baselines

- **T5-Small [67]** Text-to-Text Transformer (T5) is a language model developed by Google. T5-Small is the smaller variant of T5 consisting of (≈70M) parameters where the base version is of (220M) parameters. The T5-Small was created aiming to maintain good performance with a smaller number of parameters.

4.7 Performance Evaluation

We evaluated our model's performance in the autoencoding task by removing noise from text and calculating BERTScore (4.3), BLEU (4.4), ROUGE, and Perplexity scores. Additionally, we evaluated the performance of our approach with compression ratio (**Original Length/Compressed Length**).

4.7.1 BERTScore. [1]

The BERTScore calculates the semantic similarity of two pieces of text by calculating the cosine similarity of their embedding tokens. This metric outputs precision, recall, and f1 score, and their equations are as follows:

$$Recall_{\text{BERT}} = \frac{1}{N} \times \sum_{1}^{N} \left(\frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \right) \tag{4.1}$$

$$Precision_{\text{BERT}} = \frac{1}{N} \times \sum_{1}^{N} \left(\frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \right) \tag{4.2}$$

$$F1_{BERT} = \frac{1}{N} \times \sum_{1}^{N} (2 \times \frac{P_{BERT} \times R_{BERT}}{P_{BERT} + R_{BERT}}) \tag{4.3}$$

where $\mathbf{x}_i^\top \hat{\mathbf{x}}_j$ is the cosine similarity between two pieces of text and N is the total number of sentence pairs. Recall_BERT matches the scores of reference and candidate text's each token whereas, Precision_BERT calculates a matching score between candidate and reference text and F1_BERT is the harmonic mean of Recall_BERT and Precision_BERT.

4.7.2 BLEU. [2]

The BLEU metric estimates the quality of candidate text by assigning precision scores to n-grams and comparing them with one or more reference texts. Scores range from 0 and

100, where a higher score denotes better results. The mathematical formula for BLEU is as follows:

$$\text{BLEU} = \text{BP} \times e^{\sum_{n=1}^{N}(w_n \cdot log p_n)} \tag{4.4}$$

Here, The Brevity Penalty (BP) punishes shorter predictions. N is the maximum n-gram length. w_n are the weights for n-gram precision, and $\log p_n$ is the logarithm of n-gram precision in the candidate text.

4.7.3 ROUGE [3]

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a widely used metric for evaluating text generation tasks, particularly in summarization and machine translation.

- ROUGE-1 measures unigram (single-word) overlap between the generated and reference translations, providing a basic lexical similarity assessment.
- ROUGE-2 focuses on bigram (two-word sequence) overlap, capturing short-range contextual consistency.
- ROUGE-L evaluates the Longest Common Subsequence (LCS), which considers word order and fluency, making it more aligned with human judgment than simple n-gram overlap metrics.

By analyzing different granularities of text similarity, ROUGE provides a comprehensive measure of text generation quality.

4.7.4 Perplexity [4]

Perplexity is the exponential of cross-entropy loss, reminiscing how uncertain the model is about the test set, computed as $\text{PPL} = e^{H(p,q)}$, where $H(p, q)$ is the cross-entropy between true and predicted distributions. Lower PPL indicates confident, fluent predictions; higher PPL suggests uncertainty or poor generalization. It is widely used to evaluate language models for natural sequence generation.

4.8 Ablation Study

In this thorough ablation study, we investigate three pivotal dimensions of model efficiency—corpus size, leveraging Kizuki vectors, and integrating traditional and trendy transformer architectural components—across our proposed framework, TextEconomizer, alongside LLaMAFormer, a lightweight variant of transformer-based encoder-decoder method that integrates key architectural components from LLaMA. In this model, we applied RMSNorm immediately after the embedding layer and before rotary positional encoding of the query (Q) and key (K) vectors, effectively replacing absolute positional encoding. Residual connections were introduced after the multi-head attention layers to enhance training

stability, and the feed-forward network now employs SwiGLU [20] instead of the conventional ReLU activation. With all traditional Layer Normalizations replaced by RMSNorm for improved computational efficiency, LLaMAFormer retains the core flow of TextEconomizer. Subsequently, as a baseline, we adopt an Autoencoder model that mirrors the implementation of [68], substituting the original bidirectional RNN with a bidirectional Gated Recurrent Unit (Bi-GRU), which is an integral component of our framework, and evaluating exclusively on English autoencoding tasks. Another key focus of our study is to assess the role of attention in sequence-to-sequence (seq2seq) learning by comparing performance metrics with and without the attention mechanism in the Autoencoder.

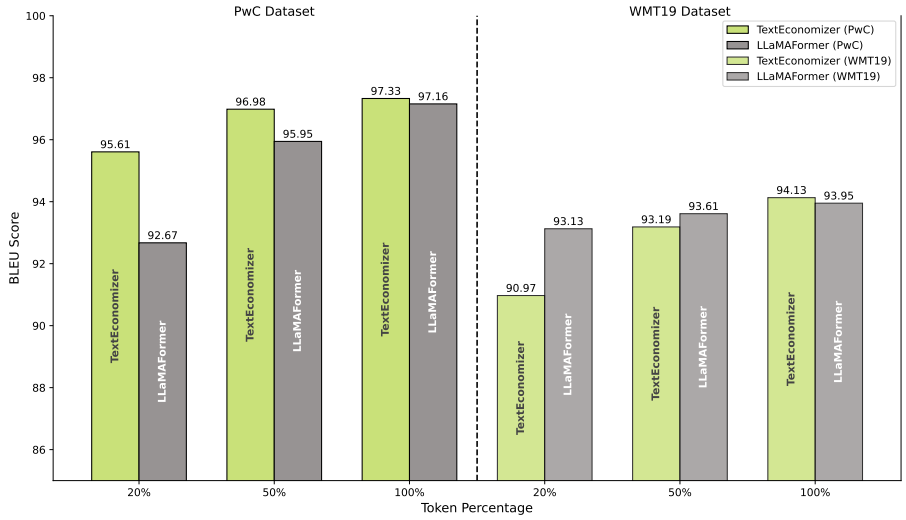

Figure 4.1: Comparative analysis of BLEU scores for TextEconomizer and LLaMAFormer across different token utilization tiers (20%, 50%, 100%) on the PwC and WMT19 datasets, where higher scores signify improved translation fidelity.

Empirical results on the WMT19 and PwC datasets demonstrate that TextEconomizer invariably outperforms both the Autoencoder baseline and LLaMAFormer across several evaluation metrics, highlighting the consequence of our architectural preferences and providing insights into optimizing efficiency in modern neural autoencoding systems. First, as illustrated in Fig. [4.1, 4.2, 4.3], we evaluated the efficacy of token reduction by conducting experiments with three configurations—20%, 50%, and 100% Kizuki vectors usage—across two datasets (PwC and WMT19). The results indicate that TextEconomizer achieves highly competitive scores even with substantial token reductions, highlighting its ability to focus on the most relevant portions of the encoder output. For instance, at 20% Kizuki vector usage, TextEconomizer achieves a BLEU score of 95.61 on the PwC dataset, which is only marginally inferior to its full-token counterpart. Similarly, on the WMT19 dataset, the 20% configuration yields a BLEU score of 90.97, maintaining potent performance despite passing only a fraction of the Kizukis to the decoder. These findings are further corroborated by the BERTScore and ROUGE-L metrics, where TextEconomizer consistently outperforms or closely matches the performance of LLaMAFormer under similar

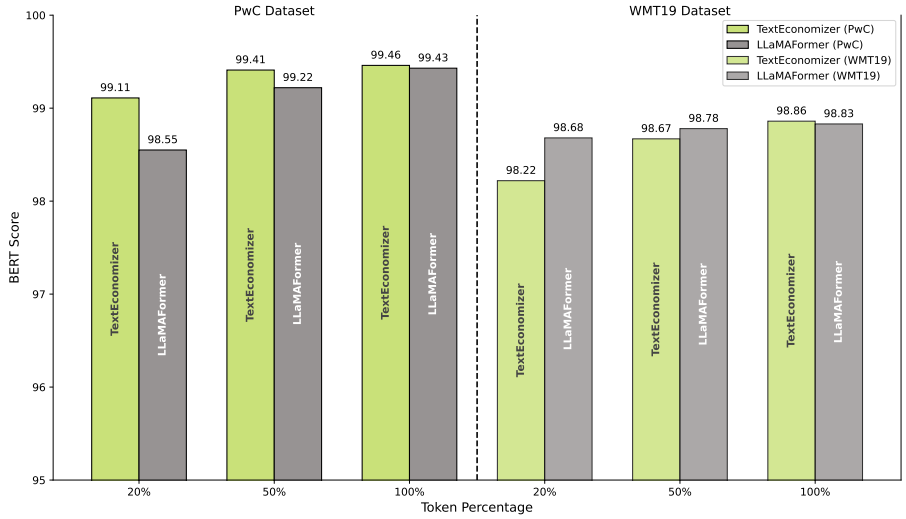

Figure 4.2: A juxtaposition of BERT scores between TextEconomizer and LLaMAFormer indicates subtle disparities in semantic fidelity across token usage levels (20%, 50%, 100%) on the PwC and WMT19 datasets, with higher values denoting refined contextual acuity.

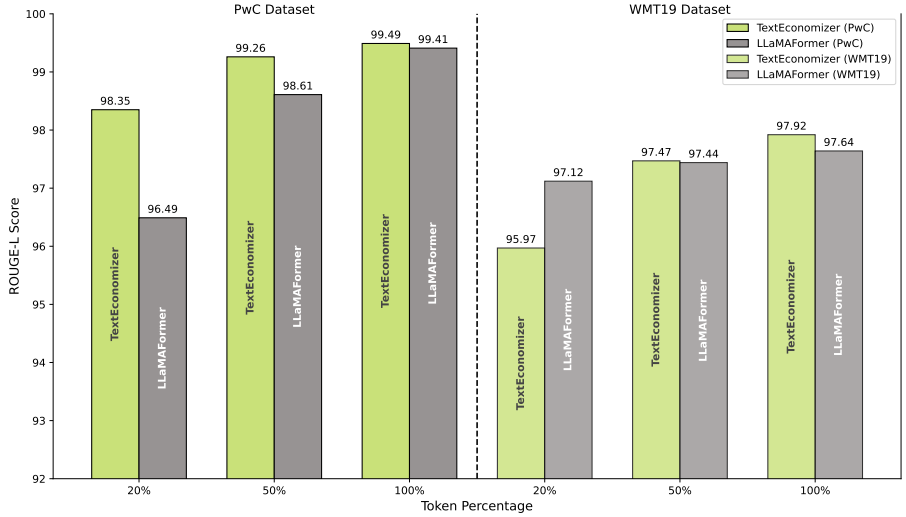

Figure 4.3: ROUGE-L metrics, elucidating the adeptness of TextEconomizer and LLaMAFormer in preserving extended sequence coherence, span token usage gradations (20%, 50%, 100%) on the PwC and WMT19 datasets, with augmented scores echoing superior alignment.

token constraints. Moreover, when augmenting Kizuki vector usage to 50%, TextEconomizer exhibits near-optimal performance, achieving a BLEU score of 96.98 on the PwC dataset and 93.19 on WMT19. Notably, these scores are within 0.35 and 1.04 points of the full-token configuration, respectively, while significantly reducing computational overhead. This suggests that our framework effectively prioritizes the most salient information from the encoder output, thereby enhancing the efficiency of the cross-attention mechanism without compromising quality.

The perplexity (PPL) results, portrayed in Fig. 4.4, further validate the efficacy of

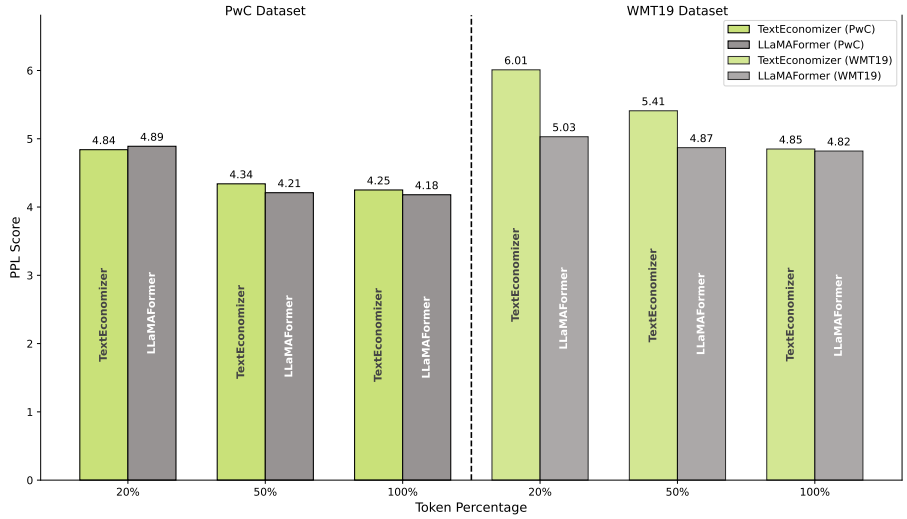

Figure 4.4: Perplexity trends for TextEconomizer and LLaMAFormer across token usage levels (20%, 50%, 100%) on both PwC and WMT19 datasets, with lower scores indicating enhanced language modeling.

Kizuki vectors. On the PwC dataset, TextEconomizer achieves a PPL of 4.34 at 50% Kizuki vector usage, closely identical to the full-token PPL of 4.25. Likewise, on WMT19, the 50% configuration produces a PPL of 5.41, closely matching the full-token PPL of 4.85. These observations underscore the robustness of TextEconomizer in leveraging reduced context vectors to maintain high-quality predictions. In addition to these observations, LLaMAFormer exhibits distinctive advantages in high-sparsity regimes on the WMT19 dataset. At 20% Kizuki vectors, it acquires a BLEU score of 93.13, outperforming TextEconomizer's 90.97 under the same settings while utilizing 1.72× fewer parameters. In contrast, its full-token PPL (4.82) marginally exceeds TextEconomizer's 4.85 on WMT19, suggesting stronger fluency in certain autoencoding configurations. However, TextEconomizer dominates in token efficiency on PwC, with a +2.94 BLEU gain over LLaMAFormer at 20% Kizuki vectors. Furthermore, as depicted in Table 4.1, scaling the PwC corpus

Table 4.1: The influence of corpus size PwC on the performance of our proposed Autoencoder.

Method	**Corpus Size**	Inference			
		BLEU	**BERT Score**	**R-L**	**PPL**
Autoencoder	100K	87.30	97.27	93.69	22.44
Autoencoder	200K	91.97	98.41	96.59	17.92
Autoencoder	242K	95.75	99.27	98.85	11.26
Autoencoder (No Attention)	242K (PwC)	7.87	79.78	19.29	948.85
Autoencoder (No Attention)	600K (WMT19)	31.16	85.68	46.72	240.02

size from 100K to 242K instances in the Autoencoder neural network significantly enhances performance across all metrics. The full 242K corpus achieves a BLEU score of 95.75 and a PPL of 11.26, outperforming smaller corpora. This trend aligns with findings from [69], confirming that larger training data volumes improve linguistic coherence and task-specific accuracy. To rigorously evaluate architectural distinctions, we compared the Autoencoder to our TextEconomizer with an equivalent parameter count. The transformer achieved superior results, including a BLEU score of 97.43, a BERT Score of 99.48, and a ROUGE-L score of 99.54, accentuating its advantages over sequential architectures like LSTMs. Notably, ablating the attention mechanism—a core component of both architectures—harshly decreases performance (e.g., PwC BLEU drops to 7.87), underscoring its paramount role in contextual alignment. While the Autoencoder model lags behind transformer-based architectures like TextEconomizer and LLaMAFormer in task-specific metrics, its fixed-size bottleneck layer offers a special advantage in memory efficiency. By compressing inputs into a fixed-size latent space, the Autoencoder reduces computational overhead greatly, making it a practical choice for scenarios prioritizing resource conservation over state-of-the-art performance. This trade-off emphasizes the importance of architectural flexibility: transformer-based models excel in quality-driven applications, whereas the Autoencoder provides a lightweight alternative for environments with stringent memory limitations. Synthesizing these insights, TextEconomizer emerges as a pragmatic choice for resource-constrained deployments, with a minimal 1.72 BLEU drop on PwC at 50% Kizuki vectors. Likewise, LLaMAFormer exhibits robustness in high-sparsity regimes, particularly on WMT19. Both models validate token reduction as a scalable strategy for English autoencoding tasks, provided attention mechanisms remain consistent, hovering computational efficiency with output quality. The Autoencoder, while lower in metrics (e.g., 95.75 BLEU vs. 97.33), offers better memory efficiency and is suitable for resource-conserving deployments, emphasizing the choice between quality and footprint. Additionally, Table 4.2 shows the impact of corpus size on Rejuvenate-

Table 4.2: The influence of corpus size (EN-DE) on the performance of RejuvenateFormer.

Method	**Corpus Size**	Inference			
		BLEU	**PR**	**RE**	**F1**
RejuvenateFormer	30K	23.07	0.8714	0.9012	0.8859
RejuvenateFormer	50K	25.12	0.8720	0.9071	0.8890
RejuvenateFormer	100K	27.31	0.8717	0.9139	0.8921

Former's performance. In this extensive study, we used three large-scale datasets but the correlation between the size of the corpus and the effectiveness of the model performance has been shown for the WMT14 EN-DE language pair dataset. Due to computational

constraints, we limited our experiment to a smaller number of sentence pairs. Surprisingly, the corpus with 100K instances outperformed those with 30K and 50K instances. Conversely, the corpus with 30K instances showed the least significant results, while the 50K instances corpus delivered moderate outcomes. This tendency was seen in all three datasets, signifying that larger corpus sizes lead to enhanced performance [70].

4.9 Comparison with State-of-the-Art Methods

In this section, we compare the performance of our TextEconomizer with state-of-the-art models that showcase advanced techniques in English autoencoding and language modeling. NUGGET [34] employs a BART encoder incorporating with a feed-forward network to distill logits and extract "nuggets"—concise text segments that minimize memory usage while keeping a high compression ratio. In contrast, ICAE [15], developed by Microsoft, builds on the Llama-2-13b architecture by integrating teacher forcing and an additional 70 million parameters, generating memory slots that improve both compression efficiency and autoencoding performance. Meanwhile, Google's T5 offers a range of scalable variants, with T5-Small (approximately 70 million parameters) delivering remarkable results despite its reduced complexity.

We scrutinize the performance of TextEconomizer, which incorporates a meticulous noise process across four corpora. In autoencoding tasks, judiciously selected Kizuki vectors for text generation reveal that TextEconomizer overtakes several transformer-based architectures in memory compression and conservation. Conventional transformers, which are especially reliant on self-attention, are hindered in their ability to narrow the bottleneck without incurring significant information loss. To substantiate this claim, we modified a Vaswani-style transformer by reducing its feed-forward layer fourfold and incorporating additional residual connections; however, this variant suffered from extreme overfitting and unendurable information loss. In contrast, while NUGGET effectively extracts succinct segments from the encoder output, it does so at almost double the parameter cost of our model. Moreover, our observations further reveal that not all encoder-produced vectors uniformly capture the complete sentence context. Additionally, the implementation of noise injection as detailed in [57] did not yield significant improvements in our experimental settings. In contrast, as shown in Table 4.3, TextEconomizer achieves a compression ratio of 3.86× on the PwC corpus while maintaining competitive BLEU scores.

With only 86 million parameters, it outperforms T5-Small by providing a 153× reduction in parameter count compared to ICAE and a 1.88× reduction compared to NUGGET. Additionally, it invariably achieves BERT scores above 99.4 and ROUGE scores exceeding 98, despite encountering a modest BLEU drop of 1.67 to 2.47 points when compared to state-of-the-art methods. Further validation on WMT14 and BookCorpus depicted in Table 4.4 demonstrates that TextEconomizer excels in all evaluation metrics except for compression ratio, where our autoencoder significantly outperforms all other methods. Although the autoencoder compresses the latent space more aggressively, its perplexity is

Table 4.3: The comparison of the quantitative performance of various existing methods across PwC and WMT19 datasets. In this table, **r** denotes the memory compression ratio, while **Θ** symbolizes no memory savings.

Method	**#Params.**	**Token%**	PwC							WMT19						
			BLEU	**BERT Score**	**R1**	**R2**	**R-L**	**PPL**	**r**	**BLEU**	**BERT Score**	**R1**	**R2**	**R-L**	**PPL**	**r**
ICAE	13.13B	100%	**99.8**	–	–	–	–	9.50	4×	–	–	–	–	–	–	–
NUGGET	161M	10%	–	–	–	–	–	–	–	**99**	–	–	–	–	28.10	10×
T5-Small	70M	100%	38.29	93.58	66.92	65.12	66.89	1.04	Θ	50.15	94.55	75.99	73.40	75.95	1.07	Θ
Our Proposed Methods																
TextEconomizer	86M	100%	97.33	99.46	99.49	98.78	99.49	4.25	1.32×	94.13	98.86	97.92	95.84	97.92	4.85	1.14×
Autoencoder	67M	100%	95.75	99.28	98.86	97.66	98.85	11.26	**67**×	91.94	98.41	96.65	93.73	96.61	22.25	**33**×
LLaMAFormer	**50M**	50%	95.94	99.22	98.61	97.05	98.61	4.21	3.86×	93.61	98.78	97.45	94.87	97.44	4.87	3.08×
TextEconomizer	86M	50%	96.98	99.41	99.27	98.33	99.26	4.34	3.86×	93.18	98.67	97.47	95.12	97.47	5.41	3.08×

Table 4.4: The comparison of the quantitative performance of various existing methods across WMT14 and BookCorpus datasets. In this table, **r** denotes the memory compression ratio, while **Θ** symbolizes no memory savings.

Method	**#Params.**	**Token%**	WMT14							BookCorpus						
			BLEU	**BERT Score**	**R1**	**R2**	**R-L**	**PPL**	**r**	**BLEU**	**BERT Score**	**R1**	**R2**	**R-L**	**PPL**	**r**
T5-Small	70M	100%	60.05	95.45	81.52	79.31	81.52	1.16	Θ	73.37	97.04	92.16	88.47	92.15	1.03	Θ
Our Proposed Method																
Transformer	86M	100%	94.08	98.93	97.84	95.51	97.84	4.90	1.25×	90.58	97.92	93.89	87.11	93.86	5.24	1.12×
Autoencoder	67M	100%	91.72	98.39	96.39	93.37	96.37	21.52	**33**×	88.81	97.61	92.90	85.15	92.86	12.33	**25**×
LLaMAFormer	**50M**	50%	93.60	98.74	97.51	94.94	97.50	5.04	3.07×	89.92	97.82	93.52	86.37	93.48	5.31	3.10×
TextEconomizer	86M	50%	94.18	99.06	98.25	96.42	98.25	4.95	3.07×	90.30	97.88	93.71	86.75	93.66	5.33	3.10×

markedly higher than that of our TextEconomizer, which indicates lesser confidence in the token prediction. Remarkably, by passing 20% of the Kizuki vectors to the decoder's cross-attention layer and applying LZMA entropy coding, the compression ratio on PwC improved by 1.39 points over ICAE, with similar excellent results on WMT14 (5.07), WMT19 (5.08), and BookCorpus (5.10), all with insignificant quality loss. Furthermore, our TextEconomizer produces superior bits-per-character (bpc) scores—WMT14: 1.5106×10^{-1}, WMT19: 4.5817×10^{-1}, PwC: 3.5102×10^{-1}, and BookCorpus: 2.9544×10^{-1}—which are significantly lower than those of the autoencoder, thereby demonstrating more efficacious data representation and more precise token prediction. In sum, TextEconomizer offers a perspicacious harmony between rigorous memory compression and high-fidelity text generation, rendering it an invaluable asset for various downstream tasks.

In parallel, we examined our Autoencoder variant that reduces the initial dimensionality into a fixed-size latent space, which is then further compressed using general-purpose lossless compressors such as LZMA [66], GZIP [71], ZLIB [72], and ZSTD [73]. Our rigorous tests revealed that LZMA consistently achieved the highest compression ratios—67× on PwC, 33× on both WMT19 and WMT14, and 25× on BookCorpus. Moreover, it surpasses transformer-based lossless compressors such as GPT-AC [32] and TRACE [31] in compression efficiency on the BookCorpus dataset, while exhibiting only minor degradation in text quality for sequences where $\ell < 1000$. This approach significantly optimizes memory usage, reducing storage requirements by 32GB to 44GB per epoch across datasets.

Table 4.5: The juxtaposition of the compression ratio of different existing methods across various corpora.

Dataset	Corpus Size	Vowel Remove (Ours)					GPT	TRACE
		LZW	LZMA	GZIP	GLIB	AC		
BookCorpus	6.7M	**12.57**	4.583	3.603	3.575	2.907	10.55	4.49
EN-DE	1.5M	**13.38**	5.257	3.959	3.927	2.947	–	–
EN-FR	1.6M	**11.42**	4.656	3.643	3.626	2.953	–	–

Table 4.6: The comparison of the quantitative performance of different existing methods across different datasets. Here, PR denotes precision, RE denotes Recall, and F1 denotes the average F1-score.

Method	#Params.	WMT14 EN-DE				WMT14 EN-FR				BookCorpus			
		BLEU	PR	RE	F1	BLEU	PR	RE	F1	BLEU	PR	RE	F1
BBFNMT [29]	**230.3M**	29.37	–	–	–	42.52	–	–	–	–	–	–	–
T5-Small	60.51M	41.44	0.93169	0.91312	0.92211	34.57	0.9224	0.9039	0.9129	63.61	0.9703	0.9539	0.9618
RejuvenateFormer	63.23M	**27.31**	**0.8717**	**0.9139**	**0.8921**	**25.78**	**0.8691**	**0.9131**	**0.8903**	**50.45**	**0.9384**	**0.9527**	**0.9454**

Notably, the Autoencoder is 196× smaller than the best-performing transformer-based alternative while incurring only a 4% disparity in quality performance. The empirical findings demonstrate that the Autoencoder surpasses NN-based lossless compressors in terms of memory efficiency. Building on these insights, our lightweight neural network variant, LLaMAFormer, achieves prominent performance over the Autoencoder across all datasets and surpasses TextEconomizer on a couple of metrics in the WMT19 dataset—all while operating with a minimal 50M parameters. This demonstrates that a streamlined transformer-based architecture can effectively process injected noise to denoise text and generate high-quality outputs. Moreover, We evaluated the results of our pre-processing technique on all three corpora. We tested the significance of removing vowels in compressing text using various general-purpose lossless compressors such as LZMA, LZW, GZIP [71], ZLIB [72], and Arithmetic Coding (AC) [74]. Our rigorous experiment found that LZW compresses text 13.38×, 12.57×, and 11.42× on EN-DE, BookCorpus, and EN-FR corpus, respectively, and establishes itself as the state-of-the-art in compression ratio by outperforming GPT-based approach [32], TRACE [31], and other compressors. Table 4.5 demonstrates that removing vowels significantly improves the compression ratio and lowers the length incorporating compressed representation. In contrast, we presented the quantitative performance of different Transformer-based methods in Table 4.6. Our proposed model, RejuvenateFormer, shows favorable performance across all three corpora in small-scale experiments, being 3.6× small in terms of parameter count. In contrast, the pre-trained T5-Small performed significantly well over BBFNMT and RejuvenateFormer across all evaluation metrics and corpora, except for the EN-FR dataset with only 40K sentence pairs and 30 epochs. This highlights the effectiveness of larger contextual models

when integrated with our proposed text pre-processing strategy. However, we did not consider the scores for each sentence separately; instead, we calculated the overall BLEU and BERTScore by evaluating all the sentences in a corpus.

Our extensive qualitative experiments present three complementary approaches to efficient text processing. TextEconomizer achieves a 5.39× compression ratio while maintaining competitive BLEU scores through Kizuki vectors, requiring only 86M parameters. The LLaMAFormer variant, incorporating LLaMA architectural components, further reduces the parameter count to 50M while maintaining strong performance, particularly excelling on the WMT19 dataset. Finally, our Autoencoder approach demonstrates exceptional compression capabilities, achieving up to 67× compression ratios on the PWC dataset and significant memory savings across all tested corpora. Together, these approaches showcase different strategies for balancing model efficiency with generation quality, offering practical solutions for resource-constrained NLP applications.

4.10 Qualitative Results

The qualitative performance of ICAE, T5 Small, Autoencoder, LLaMAFormer, and TextEconomizer has been depicted in Table 4.7 and 4.8, effectively showcasing the superiority of our Transformer-based TextEconomizer for the lossy autoencoding task. The examples provided in the table encompass complex sentences, numerical data and special characters, interrogatives, negations, proper nouns, dates, technical terms, idiomatic expressions, wordplay, and rhyme. The narrative of sentences progresses seamlessly from anticipation to connection, then transition, and finally pinnacles in an unexpected twist, thereby highlighting the robustness of our qualitative analysis. Notably, the results reveal that T5 Small struggles with punctuation and long sentences, whereas ICAE often substitutes words with synonyms—occasionally generating redundant sequences that, while semantically uniform, can be unnecessarily lengthy. In contrast, the Autoencoder exhibits punctuation challenges, particularly at sentence boundaries, yet unfailingly maintains the intended meaning and avoids incoherent expressions. For instance, as seen in the fourth example, it tends to replace double quotes with single quotes and sometimes redundantly repeats words (e.g., "investment investment"). LLaMAFormer, on the other hand, occasionally introduces spaces within email addresses and faces difficulties when reconstructing lengthy names of people. The 50% vector reduction variant tends to insert extraneous punctuation after years (e.g., converting "2017" to "2017:") and sometimes expands contractions (e.g., "they've" to "they have"), while the 20% vector reduction variant may replace words but not inappropriate and omit single quotes surrounded within double quotes.

Our proposed model, TextEconomizer, occasionally omits punctuation in quotes or within sentences containing dense punctuation, yet it consistently maintains the core semantic content—a critical factor in lossy text compression. Additionally, TextEconomizer sometimes skips apostrophes in contractions (e.g., rendering "she would" as "she'd"

Table 4.7: The qualitative effectiveness of various transformer-based methods in contrast to TextEconomizer. **Red** denotes ignored/ wrong/ extra words/ characters.

(Input)	reid and partner alfie hewett came from a set down to beat the french pair stephane houdet and nicolas peifer 4-66-1 7-6(8-6).
(ICAE)	reid and partner alfie hewett came from a set down to beat the french pair stephane houdet and nicolas peifer 4-66-1 7-6(8-6).
(T5 Small)	reid and partner alfie hewett came from a set down to beat the french pair stephan
(Autoencoder)	reid and partner alfie hewett came from a set down to beat the french pair stephane houdet and nicolas peifer 4-66 -6-8 (8-6)
(LLaMAFormer)	reid and partner alfie hewett came from a set down to beat the french pair stephane houdet and nicolas peifer 4-66-1 7-6(8-6).
(TextEconomizer)	reid and partner alfie hewett came from a set down to beat the french pair stephane houdet and nicolas peifer 4-66-1 7-6(8-6).
(Input)	experimentally, we comprehensively compare the behavior of icl and explicit fine-tuning based on real tasks to provide empirical evidence that supports our understanding. the results prove that icl behaves similarly to explicit fine-tuning at the prediction level, the representation level, and the attention behavior level.
(ICAE)	experimentally, we comprehensively compare the behavior of icl and explicit finetuning based on real tasks to provide empirical evidence that supports our findings . the experimental evidence proves that icl behaves like us to the same extent . prediction at the explicit finetuning level, the representation level, and the attention behavior level.
(T5 Small)	experimentally, we comprehensively compare the behavior of icl and explicit fine-tuning
(Autoencoder)	experimentally, we comprehensively compare the behavior of icl and explicit fine-tuning based on real tasks to provide empirical evidence that supports our understanding. the results prove that icl behaves similarly to explicit fine-tuning at the prediction level, the representation level, and the attention behavior level.
(LLaMAFormer)	experimentally, we comprehensively compare the behavior of icl and explicit fine-tuning based on real tasks to provide empirical evidence that supports our understanding. the results prove that icl behaves similarly to explicit fine-tuning at the prediction level, the representation level, and the attention behavior level.
(TextEconomizer)	experimentally, we comprehensively compare the behavior of icl and explicit fine-tuning based on real tasks to provide empirical evidence that supports our understanding. the results prove that icl behaves similarly to explicit fine-tuning at the prediction level, the representation level, and the attention behavior level.
(Input)	i was eagerly preparing for my trip to paris, france, where the eiffel tower stands tall, to attend a conference on machine learning, a subset of artificial intelligence, scheduled from september 10 to september 12, 2023. i have not read that book yet-the one recommended for the conference-but i plan to do so soon, perhaps during the flight. back home, despite the heavy rain, which had been pouring since morning, the football match continued as scheduled, much to the delight of the fans, and i couldn't help but follow the updates online. the company that sponsors the team had just announced that its revenue increased by 15% last quarter, reaching $10 million, a significant milestone, thanks to their cutting-edge ai innovations-some of which would be showcased at the conference. on the plane to paris, i overheard a fellow passenger ask, 'what is the capital of australia, and why is it not sydney?'-prompting me to recall that it's canberra, designed specifically as the capital, unlike the more famous sydney. unfortunately, when i arrived in paris, i felt under the weather and decided to stay in my hotel, missing the first day of the conference. as i rested, a quirky thought crossed my mind: 'time flies like an arrow; fruit flies like a banana,' and i smiled, hoping to recover quickly for the remaining days.
(T5 Small)	i was eagerly preparing for my trip to paris, france : where the eiffel tower stands tall, to attend a conference on machine learning, a subset
(Autoencoder)	i was eagerly preparing for my trip to paris, france, where the eiffel tower stands tall to to a a conference on machine learning, a subset of artificial intelligence, scheduled from september 10 to september 12, 2023. i have not read that book yet-the one recommended for the conference-but i plan to do so soon, perhaps during the flight back home. despite the heavy rain, which had been pouring since morning, the football match continued as scheduled, much to the delight of the fans, and i couldn help help follow follow the updates online. the company that sponsors the team had just announced that its revenue increased by 15% last quarter, reaching $10 million, a significant milestone, thanks to their cutting-edge ai innovations-some of which would be showcased at the conference. on the plane to paris, i overheard a fellow passenger ask', what is the capital of australia, and why is it not sydney'-prompting me to recall that it canberra, designed specifically as the capital, unlike the more famous sydney. unfortunately, when i arrived in paris, i felt under the weather and decided to stay in my hotel, missing the first day of the conference. as i rested, a quirky thought crossed my mind', time files like an arrow, fruit flies like a banana, and and i am, hoping to recover quickly for the remaining days.
(LLaMAFormer)	i was eagerly preparing for my trip to paris, france, where the eiffel tower stands tall, to attend a conference on machine learning, a subset of artificial intelligence, scheduled from september 10 to september 12, 2023. i have not read that book yet-the one recommended for the conference-but i plan to do so soon, perhaps during the flight back home. despite the heavy rain, which had been pouring since morning, the football match continued as scheduled, much to the delight of the fans, and i couldn't help but follow the updates online. the company that sponsors the team had just announced that its revenue increased by 15% last quarter, reaching $10 million, a significant milestone, thanks to their cutting-edge ai innovations-some of which would be showcased at the conference. on the plane to paris, i overheard a quirky sydney. unfortunately, when i arrived in paris, i felt under the weather and decided to stay in my hotel, missing the first day of the conference. as i rested, a quirky thought crossed my mind'. time flies like an arrow fruit flies like a banana', and i smiled quickly for the conference. on the plane to paris, i rested a quirky thought-to recover quickly for the remaining days.
(TextEconomizer)	i was eagerly preparing for my trip to paris, france, where the eiffel tower stands tall, to attend a conference on machine learning, a subset of artificial intelligence, scheduled from september 10 to september 12, 2023. i have not read that book yet-the one was recommended for the conference-but i plan to do so soon, perhaps during the flight back home. despite the heavy rain, which had been pouring since morning, the football match continued as scheduled, much to the delight of the fans, and i couldn't help but follow the updates online. the company that sponsors the team had just announced that its revenue increased by 15% last quarter, reaching $10 million, a significant milestone, thanks to their cutting-edge ai innovations-some of which would be showcased at the conference. on the plane to paris, i overheard a fellow passenger ask', what is the capital of australia, and why is it was not sydney's-prompting me to recall that it's canberra, designed specifically as the capital, unlike the more famous sydney. unfortunately, when i arrived in paris, i felt under the weather and decided to stay in my hotel, missing the first day of the conference. as i rested, a quirky thought my mind: time flies like an arrow, fruit flies like a banana', and i smiled, hoping to recover quickly for the remaining days.

Table 4.8: The qualitative effectiveness of various transformer-based methods in contrast to TextEconomizer. **Red** denotes ignored/ wrong/ extra words/ characters.

(Input)	sarah found a $50 bill on the street and excitedly shouted, “i’m going to save this!” ten minutes later, she walked out of the store with $75 worth of things she didn’t need, proudly calling it an “investment.”
(T5 Small)	sarah found a $50 bill on the street and excitedly shouted , “ i’m going
(Autoencoder)	sarah found a $50 bill on the street and excitedly shouted, ‘ i’m going to save this! ’ ten minutes later, she walked out of the store with $75 worth of things she didn’t need, proudly calling it an investment investment.
(LLaMAFormer)	sarah found a $50 bill on the street and excitedly shouted“, is going to save this, minor ten minutes later. she walked out of the store with $75 worth of things she didns t need, proudly calling it an “investment investment.
(Transformer)	sarah found a $50 bill on the street and excitedly shouted, “i’m going to save this ?” ten minutes later, she walked out of the store with $75 worth of things she didn’t need, proudly calling it an “investment.”
(Input)	tiny toes and button nose, a bundle of joy soon to expose!
(T5 Small)	tiny toes and button nose, a bundle of joy soon to expose!
(Autoencoder)	tiny toes and button nose, a bundle of joy soon to to be seen .
(LLaMAFormer)	tiny toes and button nose : a bundle of joy soon to expose
(TextEconomizer)	tiny toes and button nose : a bundle of joy soon to expose!
(Input)	in some cases the number is 120,000,130,000.
(T5 Small)	in some cases the number is 120,000,130,000.
(AutoEncoder)	in some cases the number is 120,000,130,000.
(LLaMAFormer)	in some cases the number is 120,000,130,000.
(TextEconomizer)	in some cases the number is 120,000,130,000.

without the apostrophe), yet the 50% Kizuki vector variant exhibits improved accuracy in tackling punctuation within quotations, albeit with minor punctuation substitutions. With 20% Kizuki vector variant, nevertheless, misses single-digit numbers in lengthy sentences and, in cases with consecutive commas, alters base-form verbs to past participles rarely. Subsequently, we evaluated the qualitative performance of T5-Small and RejuvenateFormer. Our proposed model showed superior performance in both training and inference times compared to T5-Small in a variety of sentence conditions. We found that while T5-Small was accurate in restoring short sentences with a low number of missing vowels, it struggled with longer sentences that had more missing vowels. However, due to its training on a larger corpus, T5-Small outperformed RejuvenateFormer in terms of the overall score, benefiting from a wider linguistic context. In contrast, our RejuvenateFormer model demonstrated its competitive performance by accurately restoring sentences with a higher number of missing vowels in different sentence structures.

Overall, these qualitative remarks suggest that our framework is well-suited for any optimized encoder-decoder neural network, particularly transformer-based models. Specifically, our proposed TextEconomizer not only produces fine-grained outputs with exceptional memory efficiency compared to baseline models but also effectively mitigates diverse textual errors through a meticulous noise injection process. This design choice clarifies occasional punctuation mistakes, which are overshadowed by the model’s ability to maintain the sentence’s core meaning; signifying its achievement in lossy text compression. Moreover, all methods, including T5 Small, reveal proportional efficacy when processing shorter sentences, as evidenced by the final example.

Chapter 5

Conclusion and Future Work

In this study, we present an encoder-decoder framework and a memory-efficient baseline for the task at hand by proposing TextEconomizer—a monolingual transformer that leverages Kizuki vectors and a novel text-noising strategy. TextEconomizer refines the latent representation by strategically filtering its most informative vectors and by using entropy coding algorithms to condense the latent space more efficiently, effectively addressing the complex linguistic intricacies inherent in the task. TextEconomizer outperforms transformer-based baseline methods in terms of parameter and memory efficiency across various corpora, with only a negligible $\approx$2% reduction in N-gram precision compared to the best-performing models. Moreover, the adaptability of our framework enables seamless integration with any encoder-decoder network. We took advantage of this flexibility by incorporating LLaMAFormer, a remarkably memory-efficient transformer variant that demonstrates superior text generation performance compared to our autoencoder. In addition, we integrated a state-of-the-art memory-efficient autoencoder with our sophisticated noisy text processing approach, thereby challenging the notion that autoencoder-based methods are solely adequate for image compression and extending their relevance to text-based tasks. In contrast, this study identified the primary obstacle in text compression and proposed a comprehensive architecture. We introduced a new pre-processing technique that leverages the compatibility of the Lempel-Ziv-Welch algorithm to reduce the length of text, resulting in a state-of-the-art compression ratio on EN-DE, EN-FR, and BookCorpus dataset. Concurrently, we validated the efficacy of our proposed technique on these three large-scale corpora, confirming its compatibility with transformers-based encoder-decoder architecture like pre-trained T5-small and establishing it as a promising method for text restoration tasks. Notably, we introduced a transformer-based text restoration method, meticulously designed to address complex linguistic patterns by harnessing attention mechanisms and positional encoding resulting in promising evaluation scores even in small-scale experiments. Our work opens new avenues for efficient natural language processing in resource-constrained settings. Future research directions include knowledge distillation from multilingual models to enhance our monolingual model, quantization for improved compression outcomes, text compression in low-resource languages

such as Bangla, and large-scale experiments integrating contrastive learning techniques.

References

[1] Tianyi Zhang, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkeHuCVFDr`.

[2] Post Matt. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W18-6319`.

[3] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[4] Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

[5] Mark R Nelson. Lzw data compression. *Dr. Dobb's Journal*, 14(10):29–36, 1989.

[6] Donald E Knuth. Dynamic huffman coding. *Journal of algorithms*, 6(2):163–180, 1985.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[9] Vermont State Safety Highway Office. Worldwide texting statistics, January 2023. URL `https://shso.vermont.gov/sites/ghsp/files/documents/Worldwide%20Texting%20Statistics.pdf`. [Online; accessed 2025-01-31].

[10] Venka Palaniappan and Shahram Latifi. Lossy text compression techniques. In *ICCS 2007: Proceedings of the 15th International Workshops on Conceptual Structures*, pages 205–210. Springer, 2007.

[11] Sam Rahimi. I have been using this compression technique with amazing results for a chatbot platform, 2023. URL `https://medium.com/@samrahimi420/i-have-been-using-this-compression-technique-with-amazing-results-for-a-chatbot-platform-im-b1f425aa361a`. Accessed: 2025-02-23.

[12] R Bhuvaneswari and P R Tharaniesh. Exploring chatgpt for email content compression and summarization. In *2023 4th International Conference on Communication, Computing and Industry 6.0 (C216)*, pages 01–07. IEEE, 2023.

[13] Hugh E. Williams. Compression, speed, and search engines, 2012. URL `https://hughewilliams.com/2012/03/24/compression-speed-search-engines/`. Accessed: 2025-02-23.

[14] Thang Luong, Kyunghyun Cho, and Christopher D. Manning. Neural machine translation. In Alexandra Birch and Willem Zuidema, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `https://aclanthology.org/P16-5005/`.

[15] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

[16] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58, 2014.

[17] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2019 conference on machine translation (WMT19). In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5301. URL `https://aclanthology.org/W19-5301`.

[18] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[19] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[20] Noam Shazeer. Glu variants improve transformer, 2020. URL `https://arxiv.org/abs/2002.05202`.

[21] Jinfang Niu. Corporate archives in the wild. *Global Knowledge, Memory and Communication*, 2023.

[22] Elizabeth R Leggett. *Digitization and digital archiving: A practical guide for librarians*. Rowman & Littlefield, 2020.

[23] Matteo Catena, Craig Macdonald, and Iadh Ounis. On inverted index compression for search engine efficiency. In Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann, editors, *Advances in Information Retrieval*, pages 359–371, Cham, 2014. Springer International Publishing. ISBN 978-3-319-06028-6.

[24] Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed transformer-cnn architectures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14388–14397, 2023.

[25] Mingxiao Li, Rui Jin, Liyao Xiang, Kaiming Shen, and Shuguang Cui. Crossword: A semantic approach to text compression via masking. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9171–9175, 2024. doi: 10.1109/ICASSP48485.2024.10447857.

[26] Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. Llmzip: Lossless text compression using large language models, 2023. URL `https://arxiv.org/abs/2306.04050`.

[27] Zuchao Li, Zhuosheng Zhang, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. Text compression-aided transformer encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3840–3857, 2021.

[28] Gabriele Prato, Mathieu Duchesneau, Sarath Chandar, and Alain Tapp. Towards lossless encoding of sentences. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational*

Linguistics, pages 1577–1583, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1153. URL https://aclanthology.org/P19-1153.

[29] Zuchao Li, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, Zhuosheng Zhang, and Hai Zhao. Explicit sentence compression for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8311–8318, 2020.

[30] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[31] Yu Mao, Yufei Cui, Tei-Wei Kuo, and Chun Jason Xue. Trace: A fast transformer-based general-purpose lossless compressor. In *Proceedings of the ACM Web Conference 2022*, pages 1829–1838, 2022.

[32] Cynthia Huang, Yuqing Xie, Zhiying Jiang, Jimmy Lin, and Ming Li. Approximating human-like few-shot learning with gpt-based compression. *arXiv preprint arXiv:2308.06942*, 2023.

[33] Tao Ge, Heming Xia, Xin Sun, Si-Qing Chen, and Furu Wei. Lossless acceleration for seq2seq generation with aggressive decoding. *arXiv preprint arXiv:2205.10350*, 2022.

[34] Guanghui Qin, Durme Van, and Benjamin. Nugget: Neural agglomerative embeddings of text. In *International Conference on Machine Learning*, pages 28337–28350. PMLR, 2023.

[35] Kexin Wang, Nils Reimers, and Iryna Gurevych. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning, 2021. URL https://arxiv.org/abs/2104.06979.

[36] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

[37] Chanakya Malireddy, Tirth Maniar, and Manish Shrivastava. Scar: Sentence compression using autoencoders for reconstruction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 88–94, 2020.

[38] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[39] Junxuan Zhang, Zhengxue Cheng, Yan Zhao, Shihao Wang, Dajiang Zhou, Guo Lu, and Li Song. L3tc: Leveraging rwkv for learned lossless low-complexity text compression, 2024. URL https://arxiv.org/abs/2412.16642.

[40] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936/.

[41] Noel Elias, Homa Esfahanizadeh, Kaan Kale, Sriram Vishwanath, and Muriel Medard. Multitok: Variable-length tokenization for efficient llms adapted from lzw compression, 2025. URL https://arxiv.org/abs/2410.21548.

[42] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[43] 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. URL https://arxiv.org/abs/2309.10668.

[44] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

[45] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

[46] Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36, 2024.

[47] Guanghui Qin, Corby Rosset, Ethan Chau, Nikhil Rao, and Benjamin Van Durme. Dodo: Dynamic contextual compression for decoder-only LMs. In Lun-Wei Ku, Andre

Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9961–9975, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.536. URL https://aclanthology.org/2024.acl-long.536/.

[48] Julien Tissier, Christophe Gravier, and Amaury Habrard. Near-lossless binarization of word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7104–7111, 2019.

[49] Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 6196–6203, 2019.

[50] Nariman Farsad, Milind Rao, and Andrea Goldsmith. Deep learning for joint source-channel coding of text. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 2326–2330. IEEE Press, 2018. doi: 10.1109/ICASSP.2018.8461983. URL https://doi.org/10.1109/ICASSP.2018.8461983.

[51] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. *arXiv preprint arXiv:1811.08162*, 2018.

[52] Fernando Lara, Jorge Arellano, Miguel Castillo, Luis Topón, and Enrique V. Carrera. Source coding for text transmission using a deep neural network as a lossy compression stage. In *2020 IEEE ANDESCON*, pages 1–5, 2020. doi: 10.1109/ANDESCON50619.2020.9272105.

[53] Margarita Geleta, Daniel Mas Montserrat, Xavier Giro-i Nieto, and Alexander G Ioannidis. Deep variational autoencoders for population genetics. *biorxiv*, pages 2023–09, 2023.

[54] Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi Jaakkola. Educating text autoencoders: Latent representation guidance via denoising. In *International conference on machine learning*, pages 8719–8729. PMLR, 2020.

[55] Alejo Lopez-Avila and Víctor Suárez-Paniagua. Combining denoising autoencoders with contrastive learning to fine-tune transformer models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2032, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.124. URL https://aclanthology.org/2023.emnlp-main.124/.

[56] Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation, 2022. URL https://arxiv.org/abs/2112.06749.

[57] Markus Freitag and Scott Roy. Unsupervised natural language generation with denoising autoencoders. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3922–3929, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1426. URL https://aclanthology.org/D18-1426/.

[58] Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019.

[59] James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

[60] Siyu Wang, Jianfei Chen, Chongxuan Li, Jun Zhu, and Bo Zhang. Fast lossless neural compression with integer-only discrete flows. In *International Conference on Machine Learning*, pages 22562–22575. PMLR, 2022.

[61] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders, 2017. URL https://arxiv.org/abs/1703.00395.

[62] Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17492–17501, 2022.

[63] Yuanchao Bai, Xu Yang, Xianming Liu, Junjun Jiang, Yaowei Wang, Xiangyang Ji, and Wen Gao. Towards end-to-end image compression and analysis with transformers. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 104–112, 2022.

[64] Ming Lu, Peiyao Guo, Huiqing Shi, Chuntong Cao, and Zhan Ma. Transformer-based image compression. In *2022 Data Compression Conference (DCC)*, pages 469–469, 2022. doi: 10.1109/DCC52660.2022.00080.

[65] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[66] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.

[67] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[68] Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[69] Mehedi Hasan Bijoy, Mir Fatema Afroz Faria, Mahbub E Sobhani, Tanzid Ferdoush, and Swakkhar Shatabda. Advancing Bangla punctuation restoration by a monolingual transformer-based method and a large-scale corpus. In Firoj Alam, Sudipta Kar, Shammur Absar Chowdhury, Farig Sadeque, and Ruhul Amin, editors, *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 18–25, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.banglalp-1.3. URL `https://aclanthology.org/2023.banglalp-1.3`.

[70] Mehedi Hasan Bijoy, Mir Fatema Afroz Faria, Mahbub E Sobhani, Tanzid Ferdoush, and Swakkhar Shatabda. Advancing bangla punctuation restoration by a monolingual transformer-based method and a large-scale corpus. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 18–25, 2023.

[71] Peter Deutsch. Gzip file format specification version 4.3. Technical report, 1996.

[72] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. Technical report, Association for Computing Machinery, 1996.

[73] Y. Collet and M. Kucherawy. Rfc 8878: Zstandard compression and the 'application/zstd' media type, 2021.

[74] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.