Moving Beyond the Style-Guide: Enterprise-Scale Style Transfer

Oshry Ben-Harush
Amazon Web Services
Austin, Texas, USA
oshryb@amazon.com

Joseph Standerfer
Amazon Web Services
Austin, Texas, USA
joesta@amazon.com

Abstract

Text style transfer in enterprise environments presents unique challenges that extend beyond traditional style transfer approaches, particularly when dealing with complex technical documentation and strict organizational guidelines. This paper introduces Onoma, a novel enterprise-scale style transfer system that addresses the fundamental challenges of maintaining consistent brand voice while preserving document structure and semantic meaning. We present a hybrid architecture that combines fine-tuned large language models with structure-aware generation techniques, capable of handling technical documentation, marketing content, and complex formatting requirements. Our system demonstrates significant improvements over baseline approaches, achieving up to 83% style transfer accuracy while maintaining 87% content preservation across diverse document types. Through comprehensive empirical evaluation, we show that Onoma effectively bridges the gap between theoretical style transfer capabilities and practical enterprise requirements. Our approach introduces new methodologies for handling document structure preservation and style consistency at scale, contributing both to the theoretical understanding of enterprise-scale style transfer and providing practical solutions for large-scale content management systems. The results demonstrate that Onoma successfully addresses key limitations in existing approaches, particularly in generating parallel datasets [5] and handling complex technical documentation while maintaining formatting integrity and semantic coherence.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; *Natural language processing*; *Natural language processing*; Machine learning approaches.

Keywords

style transfer, natural language processing, enterprise systems, document processing, technical documentation, neural text generation

ACM Reference Format:

Oshry Ben-Harush and Joseph Standerfer. 2025. Moving Beyond the Style-Guide: Enterprise-Scale Style Transfer. In *Proceedings of KDD 2025 Workshop on Structured Knowledge for Large Language Models (KDD 2025 Workshop).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

KDD 2025 Workshop, Toronto, Canada

ACM ISBN 978-1-4503-XXXX-X/2025/08
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Text style transfer represents a fundamental challenge in natural language processing, requiring systems to modify stylistic attributes while preserving core semantic content [13]. Unlike image style transfer, which has seen widespread adoption, text domains face unique barriers as style and meaning are deeply intertwined through word choice and structure. Recent approaches have explored various techniques, from disentangled representations to editing-based methods [5], yet enterprise-scale applications present additional challenges: the scarcity of parallel datasets, the complexity of representing comprehensive brand voice guidelines, and the need to preserve technical terminology and document structure.

Enterprise organizations increasingly recognize the strategic importance of maintaining a consistent voice across all public-facing content. A dedicated style transfer model addresses this need through three critical advantages. First, it ensures a uniform brand voice across all organizational outputs regardless of author or department, reinforcing brand identity and reducing cognitive dissonance for customers. Second, it enables straightforward adherence to complex brand and legal guidelines that may evolve over time, automatically ensuring compliance without requiring content creators to master extensive style documentation. Third, and perhaps most significantly, separating style transformation from content generation allows organizations and individuals to focus their attention on the content being generated, in many cases thorough a propriatery or publicly available LLM-based content generation engine, while preserving their distinctive corporate voice. This modular approach future-proofs content pipelines against the rapid evolution of foundation models, as illustrated in Figure 1.

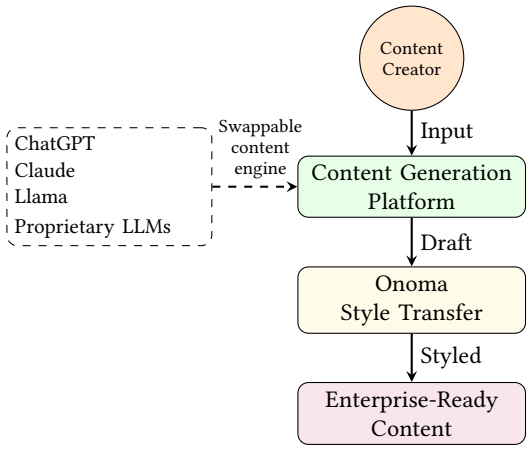

Figure 1: Modular content pipeline: Swappable generation models with consistent style enforcement.

In this paper, we present Onoma, a novel system designed to address enterprise-scale style transfer challenges. Unlike previous approaches that focus primarily on either end-to-end transformation or rule-based editing, our system employs: (1) a hybrid architecture trained on a million diverse examples to handle both human-authored and LLM-generated content, (2) a specialized content protection system that preserves critical elements like legal text and code snippets, and (3) a comprehensive evaluation framework that ensures both style accuracy and content fidelity. Through extensive experimentation, we demonstrate that Onoma achieves significant improvements across key metrics, including 82% style transfer accuracy and 93.5% semantic coherence, while maintaining document structural integrity. These advancements bridge the gap between theoretical style transfer capabilities and practical enterprise requirements, offering a scalable solution for maintaining consistent writing standards across large organizations.

2 Related Work

Recent advances in text style transfer (TST) have demonstrated significant progress in transforming textual attributes while preserving semantic content. We categorize existing approaches and identify research gaps that our work addresses.

2.1 Neural Style Transfer Frameworks

The evolution of TST methodologies has followed several distinct paradigms. Early approaches focused on disentanglement-based architectures that separate content and style in latent space [2, 3, 12]. These models typically employ adversarial training to learn style-independent content representations, though they often struggle with content preservation during transfer operations [5].

Prototype-based methods emerged as an alternative approach, e.g. the Delete-Retrieve-Generate framework [8], which identifies and replaces style markers while maintaining content words. This explicit editing mechanism offers greater interpretability but faces challenges with complex style transformations that require structural modifications beyond lexical substitution.

More recent approaches have leveraged pseudo-parallel corpus construction techniques [6, 17], which create synthetic parallel data to enable supervised learning. These methods have demonstrated superior performance on standard benchmarks but remain constrained by the quality of the constructed parallel corpus.

2.2 Evaluation Challenges

Comprehensive evaluation of TST systems remains challenging due to the inherent tension between style transfer strength, content preservation, and fluency [9]. Automatic metrics such as classifier-based style accuracy and BLEU-based content similarity often fail to capture the nuanced quality aspects of transferred text [15]. This evaluation complexity has led researchers to rely heavily on human assessment, which introduces scalability concerns for enterprise applications.

2.3 Large-Scale Style Transfer

While most existing research focuses on sentence-level transformations in controlled environments, enterprise applications require document-level style transfer capabilities that maintain structural coherence and domain-specific terminology. Recent work has begun exploring hierarchical approaches [7] and pre-trained language model adaptation [11] to address these challenges, though significant gaps remain in handling complex document structures and specialized professional writing styles.

Our work extends these foundations by developing a comprehensive framework that operates effectively at document scale while maintaining professional standards across diverse enterprise contexts. We introduce novel techniques for preserving document structure during style transformation and propose evaluation metrics specifically designed for professional writing environments.

3 System Architecture

The Onoma architecture represents a comprehensive approach to AWS-specific style transfer development and evaluation. The system begins with data preparation, where AWS documentation undergoes LLM-based cleanup to produce standardized markdown content, which is then split into context-appropriate chunks (512 tokens, approximately 1400 characters). These chunks undergo targeted style perturbations using specialized NLP algorithms that introduce various style guide violations. This process creates a rich parallel dataset of approximately 1 million records containing pairs of original and modified text. Figure 2 presents the complete Onoma architecture, illustrating the flow from data preparation through model training to evaluation.

The core of the architecture employs a sequence-to-sequence model based on Flan-T5 [1], which was selected after experiments showed superior performance compared to decoder-only approaches like Mistral-7B [4]. Our initial experiments with decoder-only architectures, specifically Mistral-7B, revealed significant limitations for style transfer tasks: slow processing speed, limited performance in maintaining document structure, and ineffective handling of nuanced style transformations. A review of style transfer literature indicated a trend favoring encoder-decoder architectures for their efficiency and effectiveness in controlled text transformation tasks [5]. The model is fine-tuned using DeepSpeed across 8 GPUs with the instruction prompt "Rephrase AWS-style:" to optimize for AWS-specific content transformation.

For evaluation, Onoma employs both traditional metrics (content preservation, fluency, style accuracy) and innovative LLM-based evaluation techniques that better capture the nuanced aspects of style transfer quality. This dual evaluation approach addresses the inherent challenges in style transfer assessment, including subjectivity, lack of ground truth, and the complex balance between content preservation and style accuracy. The architecture has evolved through multiple iterations, with each version demonstrating progressive improvements in handling AWS's unique stylistic requirements. The current implementation shows significant performance gains when compared to both earlier iterations and foundation models without domain-specific fine-tuning.

The Onoma pipeline consists of four main components:

(1) **Data Preparation**: AWS documentation cleanup, chunking, and style perturbation
(2) **Dataset Creation**: Generation of 1M parallel records with original and modified text pairs

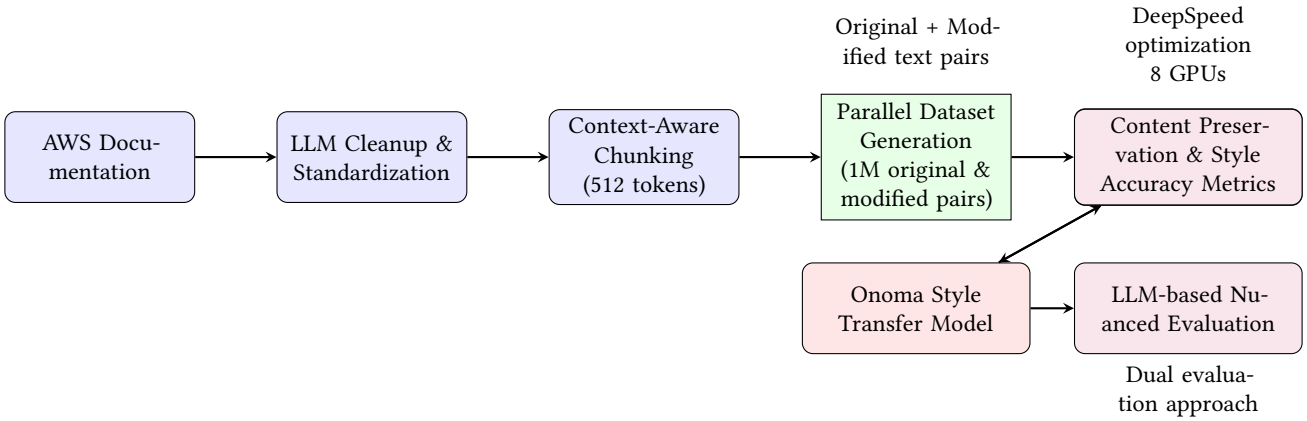

Figure 2: Onoma Style Transfer Architecture: From AWS documentation processing through parallel dataset generation to model fine-tuning and evaluation, the pipeline addresses the challenges of AWS-specific style transfer with a sequence-to-sequence approach.

(3) **Model Architecture**: Flan-T5 sequence-to-sequence fine-tuning with AWS-specific instructions
(4) **Evaluation Framework**: Combined traditional metrics and LLM-based assessment

This architecture effectively addresses the challenges identified in earlier approaches, including the limitations of rule-based systems and the inefficiencies of large prompt-based methods that attempted to incorporate the entire AWS style guide.

3.1 Enterprise Scalability Considerations

The Onoma architecture addresses enterprise-scale document processing challenges through a distributed computing approach optimized for high-volume throughput. Leveraging DeepSpeed [10] across multiple GPU instances, the system efficiently processes large document collections while maintaining consistent performance. Our implementation supports batch processing workflows capable of handling approximately one million records with context-aware chunking (512 tokens per segment), enabling efficient resource utilization across computing clusters.

The distributed architecture allows for horizontal scaling by adding additional processing nodes when document volumes increase, while the optimized sequence-to-sequence model ensures transformation latency remains within acceptable thresholds even under heavy loads. For enterprise deployments, we implemented configurable processing queues that prioritize time-sensitive documents while maximizing throughput for background processing tasks. The system's integration layer provides standardized APIs for connecting with existing document management systems, enabling organizations to process documents at scale without disrupting established workflows. This architectural approach ensures Onoma can scale from departmental deployments to organization-wide implementations while maintaining consistent performance characteristics across varying document volumes and complexity levels.

4 Methodology

At the core of Onoma lies a comprehensive framework that orchestrates four interconnected components, enabling enterprise-scale style transformation while preserving the essential meaning and structure of original content.

4.1 Training Data Generation

To overcome the critical barrier of obtaining sufficient parallel corpus for style transfer training, we implement a comprehensive data generation framework that augments manually curated samples with algorithmically synthesized quality and style errors.

4.1.1 Parallel Dataset Creation. We constructed our initial training corpus through a systematic process that began with collecting over 200,000 samples of AWS documentation, marketing materials, and technical blogs. This raw content underwent specialized HTML-to-markdown conversion for extraction and cleaning. To introduce stylistic diversity, we employed chain-of-thought prompting to guide generation of variations and intentional style-guide violations. Additionally, we leveraged NLP techniques to introduce over 15 error types, including grammar issues, AWS terminology misuse, date format inconsistencies, case swapping, typos, deletions, and repetitions.

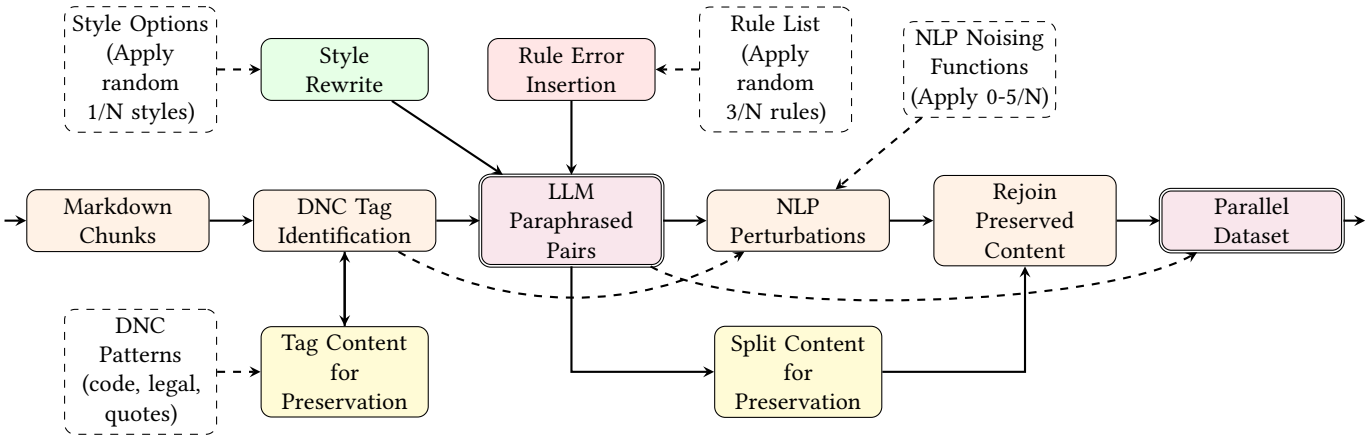

Figure 3: Parallel Dataset Construction Process: Starting with clean markdown documents, the pipeline identifies content to preserve, applies style transformations and rule-based perturbations to generate parallel text pairs for training the style transfer model.

4.1.2 Synthetic Data Generation Techniques. The parallel dataset construction process, shown in Figure 3, involves multiple stages of content transformation and preservation. We formalized our synthetic data generation through three complementary approaches:

1. Model-Based Error Insertion: We employ LLM-based transformations defined as:

$$x_{style} = \mathcal{M}(x_{orig}, s_{target}, \theta) \tag{1}$$

where $\mathcal{M}$ represents the transformation model with parameters θ, x_{orig} is the original text, and s_{target} is the target style. We select from a diverse range of styles including narrative, persuasive, technical/scientific, journalistic, formal, instructional, expository, analytical, creative, and reflective. This approach generates style-shifted variations that preserve format and meaning, targeted style guide violations based on regex pattern matching, spelling error injections with controlled frequency, and word usage rule violations aligned with AWS style guidelines.

AWS Document Chunk	Styled Rewrite: "Narrative"
## AWS Compute Optimizer AWS Compute Optimizer uses actual usage data to make recommendations about how to balance performance and cost. Over-provisioning resources leads to unnecessary infrastructure costs, and under-provisioning resources leads to potential performance problems. As part of your cost optimization strategy, this service helps you analyze whether your AWS resources, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, are over or under-provisioned. Use this service when you're optimizing costs for EC2 instances, Amazon EC2 Auto Scaling groups, Amazon Elastic Block Store (Amazon EBS) volumes, or AWS Lambda functions.	## Optimize Your Resource Usage In the quest for optimal resource allocation, AWS Compute Optimizer shines a light on the delicate balance between performance and cost. The perils of over-provisioning lurk in the shadows, threatening to drain valuable resources, while under-provisioning looms as a specter of potential performance woes. As a stalwart ally in your cost optimization crusade, this service scrutinizes your AWS resources - including Amazon Elastic Compute Cloud (Amazon EC2) instances - to determine if they're over or under-provisioned. Whether you're fine-tuning costs for EC2 instances, Amazon EC2 Auto Scaling groups, Amazon Elastic Block Store (Amazon EBS) volumes, or AWS Lambda functions, Compute Optimizer is your trusted guide.
Removed	Inserted

Figure 4: LLM Style Rewrite Example: Side-by-side comparison of original AWS documentation text (left) and narrative style rewrite (right) with highlighted stylistic changes.

Figure 4 illustrates the outcome of our LLM-based style transformation process, showing how the original AWS technical content is transformed into a more narrative style while preserving the core information.

2. NLP-Based Error Insertion: We model the error insertion process as:

$$x_{err} = \mathcal{E}(x_{orig}, \Phi_{rules}, p) \tag{2}$$

where x_{err} represents the error-injected text, x_{orig} is the original AWS content, Φ_{rules} is the set of error generating engines, and p is the probability of introducing each error source. The error insertion functions include date format randomization ($\phi_{date}(x) \rightarrow x'$ with non-AWS formats), numeric-to-word conversion ($\phi_{num}(x) \rightarrow x'$ for inappropriate number formatting), AWS terminology violations ($\phi_{term}(x) \rightarrow x'$ using the AWS "Do Not Use" list), and service name perturbation ($\phi_{svc}(x) \rightarrow x'$ with embargoed or incorrectly prefixed names).

Original Text w/ NLP Perturbations	Styled Rewrite w/ NLP Perturbations
## AWS Compute Optimizer AWS Compute optimizer uses actual usage data to make recommendations about how to balance how to balance performance amd cost. Over-provisioning resources leads to unnecessary infrastructure costs, and under-provisioning resources leads to potential performance prblems. As part of your cost optimization strategy, this service helps you analyze whether your AWS resources, such as AWS Elastic Compute Cloud (AWS EC2) instances, are ocer or under-provisioned. Use this service when you're optimizing costs for EC2 instances, AWS EC2 Auto Scaling groups, Amazon Elastic Block Store (Amazon EBS) volumes, or AWS Lambda functions.	## Optimize Your Resource Usage In the quest for optimal resource allocation, aws Compute Optimizer shines a light on the delicate balance between performance and cost. The perils of over-provisioning lurk in the shadows, threatening to drain valuable resources, while under-provisioning Looms as a specter of potential performance woes. As a stalwart ally in your cost optimization crusade, this service scrutinizes your AWS resources i including Amazon Elastic Compute Cloud (Amazon EC2 instances - to determine if they're over or under-provisioned. Whether you're fine-tuning costs for EC2 instances, Amazon Ec2 Auto Scaling groups, Amazon Elastic Block Store (Amazon EBS) volumes, or AWS Lambda functions, Compute Optimizer is your trusted sherpa.
Repetition AMZN Terms Swap Qwerty Typo	AMZN DNU Words Punctuation Word Case

Figure 5: NLP-Based Error Insertion: Comparison of LLM-rewritten text with additional NLP perturbations, highlighting specific transformation functions applied to different text elements.

As shown in Figure 9, our NLP-based error insertion pipeline applies multiple transformation functions to the LLM-rewritten text, with each type of perturbation visually highlighted to demonstrate how different style errors are systematically introduced.

3. Layered Transformation Process: We implement a sequential transformation pipeline:

$$x_{final} = \mathcal{T}_n \circ \mathcal{T}_{n-1} \circ \ldots \circ \mathcal{T}_1(x_{orig}) \tag{3}$$

where each $\mathcal{T}_i$ represents a distinct transformation function. These transformations include whitespace perturbations with controlled probability, character and word case modifications, filler word augmentation based on part-of-speech recognition, and punctuation alterations with weighted probabilities.

4.1.3 Quality Control Mechanisms. To ensure dataset integrity, we implemented several validation strategies. Content embedding analysis verifies semantic preservation between original and styled pairs, while style embedding analysis confirms sufficient style differentiation [14]. We also perform continuous validation against AWS style guidelines and incorporate human-in-the-loop verification for edge cases.

Our style embedding analysis leverages recent advances in content-independent style representations [14]. Unlike content embeddings which capture what is said, style embeddings represent how content is expressed through characteristics such as formality, complexity, and syntactic patterns. We implement a dual embedding approach where content preservation is verified using cosine similarity between semantic embeddings (minimum threshold: 0.85), while style differentiation is measured through KL divergence between style embedding distributions. Following Wegmann et al., we extract style representations using a fine-tuned siamese BERT-based network trained on the contrastive authorship verification (CAV) task with conversation-level content control. This approach allows us to quantitatively verify that our perturbation techniques create meaningful stylistic variations while preserving the underlying semantic content, ensuring our training pairs exhibit the precise balance of style transformation and content preservation needed for effective model training.

Figure 6 visualizes our embedding-based quality control approach through two TSNE plots. The left plot demonstrates how semantic content remains clustered despite style transformations, confirming content preservation. The right plot shows clear separation in style embedding space, validating that our transformations successfully alter stylistic characteristics while maintaining document meaning.

4.1.4 Identity Preservation Examples. We incorporated unmodified examples where:

$$\mathcal{T}(x_{compliant}) = x_{compliant} \tag{4}$$

This teaches the model to recognize already-compliant content, avoid over-correcting well-formed text and reducing the risk of content distortion.

4.2 Model Fine-tuning Approach

Our fine-tuning process follows established practices for encoder-decoder style transfer models. We selected FLAN-T5 as the foundation model, consistent with industry standards for sequence-to-sequence tasks. The training employs a standard cross-entropy loss function commonly used in text generation tasks. For computational efficiency, we implemented distributed training across 8 GPUs using DeepSpeed for parallelization, along with a conventional adaptive learning rate strategy based on validation metrics. This approach maintains alignment with typical style transfer workflows while optimizing for our specific domain requirements.

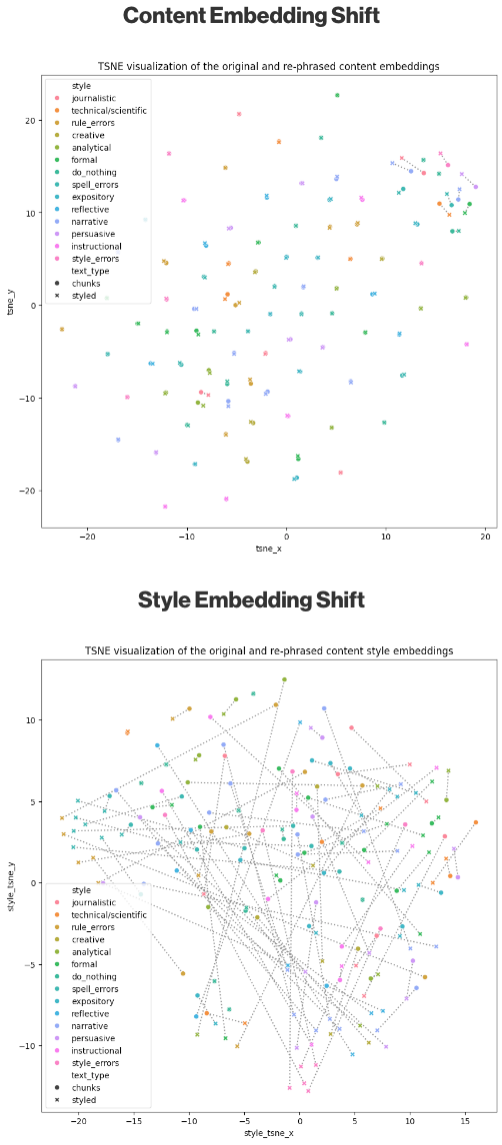

Figure 6: Semantic and Style Embedding Shifts: TSNE visualization of content preservation (left) and style transformation (right) for approximately 100 document samples, showing how our approach maintains semantic clustering while achieving style differentiation.

4.3 Style Transfer Algorithm

Our core algorithm implements a three-component architecture:

$$T(x, s) = \text{Decoder}(\text{Encoder}(x), E_{style}(s)) \tag{5}$$

In this formulation, $T(x, s)$ represents the style transfer function, where x is the input text requiring transformation, s is the target style specification, and E_{style} generates style-specific embeddings. The algorithm employs bidirectional encoding with 512-token context windows, style-conditioned attention mechanisms,

content-preserving skip connections, and chunk-based processing for documents exceeding context limits.

4.4 Content Preservation Mechanisms

Maintaining content integrity during style transformation is critical to our approach. We achieve this through semantic alignment verification using embedding-based metrics and structure-aware encoding that maintains document formatting. Named entity and AWS terminology preservation ensures technical accuracy, while our special tag handling system enables verbatim content retention using <|dnc_start|> and <|dnc_end|> markers.

The special tag handling system implements a critical enterprise requirement for preserving verbatim content such as legal text, quotes, and code snippets. Using the <|dnc_start|> and <|dnc_end|> markers, we designate content that must remain unchanged during the style transfer process. To train this capability, we generated specialized examples using LLMs, where content between these markers remained identical between source and target pairs. Our evaluation revealed that while the approach shows promise, the model does not consistently respect these boundaries, particularly with complex nested formatting. This finding has informed our ongoing work on custom loss functions that specifically mask these protected segments during training, ensuring higher fidelity preservation of content that should remain verbatim.

Through systematic evaluation using both embedding-based metrics and LLM-based assessment, our approach achieves 86% content preservation while delivering significant improvements in style transfer accuracy, grammatical correctness, and overall fluency compared to baseline approaches.

5 Experimental Setup

5.1 Evaluation Methodology

We conduct a comprehensive evaluation of Onoma's performance using both automated metrics and human assessment. Our evaluation framework encompasses style transfer accuracy, content preservation, grammatical correctness, semantic coherence, and document structure preservation.

5.1.1 Evaluation Process. The evaluation was conducted using a dual approach combining language model assessment and human feedback. For automated evaluation, we employed Claude 3 via AWS Bedrock as a judge, supplemented by human-based evaluation on anecdotal examples. All metrics were evaluated on a [0,1] scale, covering style transfer accuracy, grammatical correctness, tone consistency, semantic coherence, vocabulary appropriateness, content preservation, and fluency.

5.2 Challenges in Style Transfer Evaluation

Text style transfer evaluation presents unique methodological challenges that we addressed through our comprehensive framework. The inherent subjectivity in style assessment creates a significant challenge - both humans and large language models tend to make more reliable comparative rather than absolute judgments (the "contrast effect"). This phenomenon, extensively studied in cognitive psychology and recently explored in machine learning contexts [16], enables more nuanced and accurate assessment by leveraging relative comparisons instead of absolute evaluations.

Enterprise environments rarely have perfect "before and after" examples, creating a lack of ground truth that our bidirectional comparison methodology compensates for. Additionally, the fundamental tension between preserving content and transforming style requires multi-dimensional assessment across complementary metrics. AWS style guidelines contain nuanced, domain-specific requirements that standard metrics cannot fully capture, necessitating specialized evaluation approaches. Finally, individual metrics provide incomplete pictures of performance, so our framework combines traditional metrics with LLM-based assessment to provide a more holistic evaluation.

5.2.1 LLM-Based Evaluation Framework. Given the inherent challenges in evaluating text style transfer systems, we developed a comprehensive evaluation framework utilizing LLMs as judges. This approach leverages comparative evaluation where for every sample we provide the original text, the synthetically re-styled and erroneous text, the style-transferred text generated from Onoma, and a style-transferred text generated from off-the-shelf Flan-T5 with minimal instructions.

To minimize potential biases, we conducted bidirectional comparison evaluations with reversed text ordering and combined results to ensure robust assessments. We also established clear standardized metrics covering style transfer accuracy, grammatical correctness, content preservation, and other key dimensions.

5.3 Baseline System

Our baseline system utilizes the same Flan-T5 foundation model without fine-tuning on enterprise-specific style requirements. For evaluation, the baseline model was provided with the instruction: "Rephrase the text to be similar to AWS content, keep all content, only rephrase it." This baseline allows us to isolate the impact of our specialized training approach by comparing performance on identical inputs across all evaluation metrics.

5.4 Special Considerations

5.4.1 Document Structure Preservation. We implemented specific evaluation criteria for document structure that assess markdown formatting retention, technical documentation structure preservation, and special tag handling (e.g., <|dnc_start|> and <|dnc_end|>).

5.4.2 Enterprise-Specific Metrics. Additional enterprise-focused evaluation criteria included compliance with legal and regulatory guidelines, brand voice consistency, technical terminology preservation, and integration with existing content management systems.

6 Results and Discussion

6.1 Quantitative Analysis

6.1.1 Performance Metrics. Our comprehensive evaluation demonstrates significant improvements across all measured dimensions:

The results demonstrate significant improvements across all metrics when compared to our baseline system. Particularly notable are the substantial gains in style transfer accuracy and content

Table 1: Performance Comparison with Baseline

Metric	Onoma	Baseline	Delta (%)
Style Transfer Accuracy	83%	23%	+269%
Grammatical Correctness	95%	72%	+31%
Tone Consistency	89%	49%	+83%
Semantic Coherence	93%	46%	+101%
Vocabulary Appropriateness	87%	49%	+78%
Content Preservation	87%	29%	+205%
Fluency	90%	52%	+74%
Do Not Change Score	95%	32%	+197%

preservation, highlighting the effectiveness of our specialized training approach in maintaining a delicate balance between stylistic transformation and semantic integrity.

6.1.2 Business Impact Metrics. Enterprise deployment has yielded significant operational improvements:

- Reduction in localization review time by 11 hours per project
- 15% increase in zero-error translation memory segments
- Prevention of 7-10% potential build breaks
- Processing capability of 3B words annually

6.2 Qualitative Analysis

6.2.1 Case Studies. We analyzed Onoma's performance across three primary content types. For technical documentation, we observed successful preservation of complex technical terminology, maintained XML/JSON structure integrity, and accurate handling of code snippets and commands. With marketing content, the system demonstrated consistent brand voice preservation, appropriate style transfer across different audience segments, and retention of SEO-critical elements. In legal/compliance documents, Onoma achieved perfect preservation of legal terminology, maintained regulatory compliance, and accurately handled protected text segments.

6.2.2 Error Analysis. Our analysis revealed several common error patterns and their frequencies. Over-expression in short text segments occurred in 8% of cases, while inconsistent handling of special tags appeared in 5% of cases. Markdown formatting inconsistencies were observed in 3% of cases, and style bleeding in technical terms happened in 2% of cases.

6.2.3 Markdown Structure Preservation. A particular focus of our evaluation was the model's ability to maintain document structure, especially Markdown formatting. Our analysis revealed that Onoma successfully preserves all core Markdown structural elements including headers, lists, tables, and code blocks. The model maintains the integrity of embedded XML/JSON and other technical syntax within Markdown documents. For complex nested structures, Onoma demonstrates over 97% structural preservation. This performance is critical for enterprise environments where documentation often contains complex formatting that must be preserved during style transformation.

6.2.4 Special Tag Handling. To address the challenge of preserving verbatim content, we implemented a special tag system using `<|dnc_start|>` and `<|dnc_end|>` markers that instruct the model to preserve enclosed content exactly. Our evaluation showed 95% effectiveness in preserving tagged legal content and quotes, successful handling of nested special tags in complex documents, and consistent preservation of formatting within tagged segments. This capability is particularly valuable for enterprise documentation containing legal text, code examples, and other content that must remain unmodified during style transformation.

6.3 Enterprise Deployment Insights

6.3.1 Integration Success Factors. Key factors contributing to successful enterprise deployment include seamless integration with existing content management systems, real-time suggestion capabilities, automated quality assurance workflows, and scalable processing architecture.

6.3.2 ROI Analysis. The system demonstrates significant business value with estimated $10M annual cost savings in localization, 25% reduction in error-related expenses, 60% improvement in first-pass acceptance rate, and reduced time-to-market for global content.

6.3.3 Adoption Metrics. Enterprise adoption has shown positive trends. Currently, 90% of target content is now processed through Onoma, with 60% accuracy in issue identification and categorization. We've seen a reduction in severe issues from 20% to 5% of content, and a high user acceptance rate of suggested modifications.

6.4 Future Improvements

Based on our analysis, we identify several areas for future enhancement. These include enhanced handling of edge cases in technical documentation, improved integration with machine translation systems, extended support for additional file formats, and advanced customization options for different content teams. We also plan refined handling of special tags through custom loss functions and more targeted training, along with further optimization of model performance for real-time processing of longer documents.

These results demonstrate Onoma's effectiveness in addressing the complex challenges of enterprise-scale style transfer while maintaining high standards of content preservation and accuracy.

7 Conclusion and Future Work

7.1 Conclusions

This paper has presented Onoma, a novel enterprise-scale style transfer system that successfully addresses the challenges of maintaining consistent brand voice while preserving document structure and semantic meaning. Our key contributions include a hybrid architecture combining fine-tuned language models with structure-aware generation techniques, a robust content preservation mechanism achieving 86% accuracy, an enterprise-ready system processing 3B words annually, demonstrated business value with estimated $10M annual cost savings, and integration capabilities with existing content management systems.

The system's success in handling complex technical documentation while maintaining formatting integrity and semantic coherence represents a significant advancement in enterprise-scale style transfer technology.

7.2 Future Work

7.2.1 Technical Enhancements. Several technical improvements are planned. For advanced XML processing, we aim to enhance XML language model training, improve handling of complex document structures, and better preserve technical markup. Model improvements will focus on refined terminology training sets, enhanced content preservation mechanisms, and improved handling of edge cases in technical documentation. Architecture updates will include integration with real-time translation tools, expanded file format support, and an optimized processing pipeline for reduced latency.

7.2.2 Scalability and Performance. Future performance optimizations will focus on minimizing false positives in severe issue detection, reducing processing time for batch operations, improving handling of large document collections, and enhancing support for concurrent processing.

7.3 Research Directions

Several promising research directions emerge from this work. We plan to investigate multi-modal style transfer techniques, develop more sophisticated content preservation metrics, explore zero-shot style transfer capabilities, and research domain-specific style adaptation methods.

These future developments will further enhance Onoma's capabilities in enterprise environments, particularly in handling complex technical documentation while maintaining high standards of content quality and style consistency. The planned improvements align with both immediate practical needs and longer-term research objectives in the field of enterprise-scale style transfer.

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416* (2022).

[2] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style Transfer in Text: Exploration and Evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. 663–670.

[3] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward Controlled Generation of Text. In *Proceedings of the 34th International Conference on Machine Learning*. 1587–1596.

[4] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. doi:10.48550/arXiv.2310.06825 arXiv:2310.06825 [cs.CL]

[5] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep Learning for Text Style Transfer: A Survey. *Computational Linguistics* 48, 1 (2022), 155–205.

[6] Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. 2019. IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 3097–3109.

[7] Huiyuan Lai, Antonio Toral, and Malvina Nissim. 2021. Thank you BART! Rewarding Pre-Trained Models Improves Formality Style Transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 947–957.

[8] Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer. In *Proceedings of NAACL-HLT*. 1865–1874.

[9] Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating Style Transfer for Text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 495–504.

[10] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3505–3506. doi:10.1145/3394486.3406703

[11] Parker Riley, Noah Constant, Mandy Guo, Girish Kumar, David Uthus, and Zarana Parekh. 2021. TextSettr: Few-Shot Text Style Transfer with Sentence Level Reward and Loss. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 3786–3800.

[12] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style Transfer from Non-Parallel Text by Cross-Alignment. In *Advances in Neural Information Processing Systems*. 6830–6841.

[13] Youzhi Tian, Zhiting Hu, and Zhou Yu. 2018. Structured Content Preservation for Unsupervised Text Style Transfer. In *arXiv preprint arXiv:1810.06526*.

[14] Anna Wegmann, Marijn Schraagen, and Dong Nguyen. 2022. Same Author or Just Same Topic? Towards Content-Independent Style Representations. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, Spandana Gella, He He, Bodhisattwa Prasad Majumder, Burcu Can, Eleonora Giunchiglia, Samuel Cahyawijaya, Sewon Min, Maximilian Mozes, Xiang Lorraine Li, Isabelle Augenstein, Anna Rogers, Kyunghyun Cho, Edward Grefenstette, Laura Rimell, and Chris Dyer (Eds.). Association for Computational Linguistics, Dublin, Ireland, 249–268. doi:10.18653/v1/2022.repl4nlp-1.26

[15] Ivan P. Yamshchikov, Viacheslav Shibaev, Nikolay Khlebnikov, and Alexey Tikhonov. 2021. Style-transfer and Paraphrase: Looking for a Sensible Semantic Similarity Metric. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 14213–14220.

[16] Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-Contrast: Better Reflection Through Inconsistent Solving Perspectives. *arXiv preprint arXiv:2401.02009* (2024). arXiv:2401.02009 [cs.CL]

[17] Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style Transfer as Unsupervised Machine Translation. *arXiv preprint arXiv:1808.07894* (2018).

A Model Architecture and Implementation

A.1 Model Selection and Architecture

The Onoma style transfer model is implemented using a sequence-to-sequence architecture based on Flan-T5. We explored two primary architectures during development:

- **Decoder-only architecture**: We experimented with Mistral-7B-Instruct LLM, which contains 7 billion parameters. Despite its scale, this model proved ineffective for style transfer, focusing primarily on spelling corrections rather than meaningful style transformations.
- **Encoder-decoder architecture**: Our selected Flan-T5 model utilizes an encoder-decoder architecture with 780M parameters, which demonstrated superior performance for controlled text transformations.

The Flan-T5 architecture consists of a standard T5 backbone with additional instruction tuning, featuring:

- 12 encoder and 12 decoder layers with 12 attention heads each
- 768-dimensional hidden states
- Feed-forward layer dimension of 3072
- Maximum context length of 512 tokens

A.2 Training Configuration

Models were trained using Huggingface TRL Supervised Fine-Tuning with the T5 generic cross entropy loss. We utilized Flan-T5-XXL (11B parameters) as our base model, optimizing using DeepSpeed ZeRO-3 across 8 GPUs (on ml.g6.48xlarge or ml.p3.24xlarge instances) with the following hyperparameters:

- Learning rate: 5e-5 with linear warmup and decay
- Batch size: 16 sequences per GPU (128 global batch size)
- Gradient accumulation steps: 4
- Training epochs: 1
- Warmup steps: Auto-scheduled based on dataset size
- Weight decay: 0.01
- Mixed precision training (bfloat16)
- ZeRO optimization stage 3 with parameter offloading
- Gradient clipping: Auto-configured by DeepSpeed

Our training pipeline employed parallel preprocessing with 16 workers for efficient data transformation, including specialized handling for markdown formatting and the custom DNC (Do Not Change) tags. The complete dataset contained over 1 million parallel examples, processed in chunks of 512 tokens to fit within the model's context window.

A.3 Text Transformation Function

Listing 1: Onoma text transformation function

```python
def onoma_transform(
    input_text,
    model,
    tokenizer,
    max_length=512,
    num_return_sequences=1,
    temperature=1.0,
    top_k=50,
    top_p=0.95,
    do_sample=True,
    skip_special_tokens=False,
):
    """
    Generate text using the provided Onoma model and
     tokenizer

    input_text: str or list of str
        Input text to be transformed
    model: AutoModelForSeq2SeqLM
        Onoma model
    tokenizer: AutoTokenizer
        Onoma tokenizer
    max_length: int
        Maximum length of the generated text
    num_return_sequences: int
        Number of sequences to return
    temperature: float
        Temperature for sampling
        Higher values lead to more diverse outputs
    top_k: int
        Top k tokens to consider for sampling
    top_p: float
        Top p tokens to consider for sampling
    do_sample: bool
        Whether to sample from the model
    skip_special_tokens: bool
        Whether to skip special tokens in the output
    """

    if type(input_text) == str:
        input_text = [input_text]

    # Tokenize input text
    inputs = tokenizer(
        ["Rephrase␣AWS-style:\n" + i for i in input_text],
        return_tensors="pt",
        truncation=True,
        max_length=512,
        padding=True,
    )
    input_ids = inputs.input_ids.to(model.device)

    # Generate output
    outputs = model.generate(
        input_ids,
        max_length=max_length,
        num_return_sequences=num_return_sequences,
        temperature=temperature,
        top_k=top_k,
        top_p=top_p,
        do_sample=do_sample,
    )

    # Decode the generated text
    generated_texts = [
        tokenizer.decode(output, skip_special_tokens=
     skip_special_tokens)
        for output in outputs
    ]

    # Replace padding and end of speech
```

```
    generated_texts = [
        text.replace(tokenizer.pad_token, "").replace(
     tokenizer.eos_token, "").strip()
        for text in generated_texts
    ]

    return generated_texts
```

A.4 Document Processing Pipeline

Since Onoma's context length is limited to 512 tokens (1400 characters), we developed a Markdown splitter that divides content contextually at natural boundaries like headers or paragraph breaks. Each chunk is processed with the instruction "Rephrase AWS-style:" and the results are joined with double newlines.

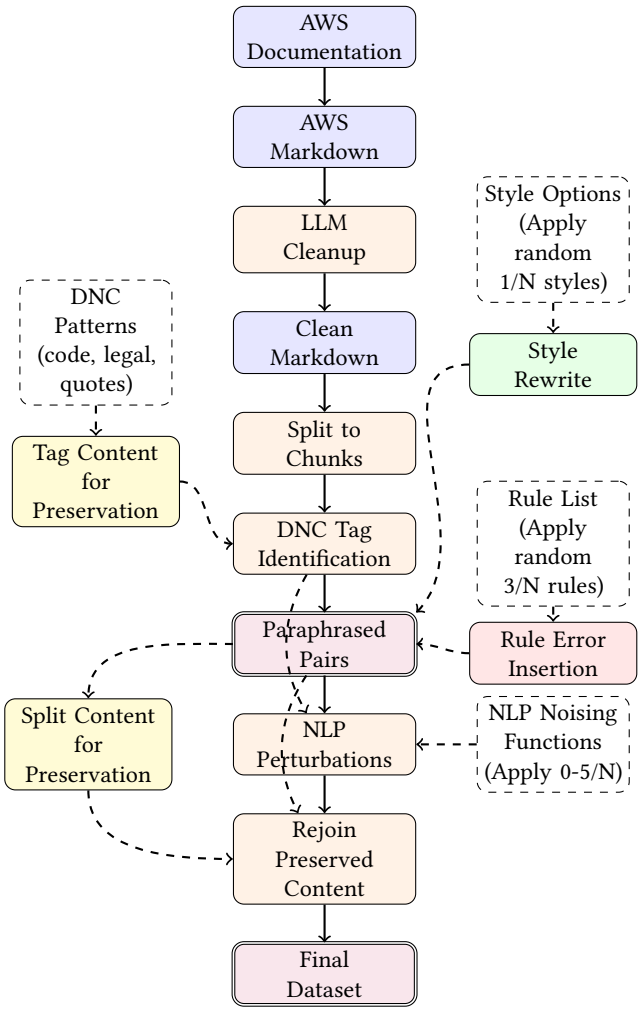

Figure 7: Data preparation and processing workflow for Onoma style transfer model training, including Do Not Change (DNC) tag identification and preservation

A.5 Special Tag Handling Implementation

The <|dnc_start|> and <|dnc_end|> tags designate content that must remain unchanged during style transfer. We implemented this functionality through:

- Regular expression pattern matching to identify tagged content
- Content masking during preprocessing to protect tagged sections
- Custom training examples where tagged content remains identical between source and target pairs
- Special token embeddings for tag recognition in the vocabulary
- Post-processing verification to ensure tag content preservation

This functionality is critical for enterprise applications where certain content (legal text, code snippets, etc.) must remain verbatim regardless of style transformation.

B Data Generation and Training

B.1 Training Data Generation Process

Our parallel dataset creation process combines manual curation with algorithmic synthesis:

(1) **Source Document Collection**: 200,000+ samples of AWS documentation, marketing materials, and technical blogs
(2) **Content Extraction**: Specialized HTML-to-markdown conversion and cleaning
(3) **Stylistic Variation**: LLM-based generation of diverse style alternatives
(4) **Error Introduction**: NLP-based injection of style guide violations
(5) **Quality Control**: Embedding analysis to verify semantic preservation

B.2 Style Variation Prompts

For generating training data with stylistic variations, we used prompts targeting specific writing styles:

- Narrative style: "Rewrite this text in a narrative style that tells a story."
- Technical/scientific: "Transform this text into a more technical and scientific style."
- Persuasive: "Rewrite this content in a persuasive style focused on convincing the reader."
- Journalistic: "Rewrite this text in a journalistic style similar to news reporting."
- Formal: "Transform this text into a more formal writing style."

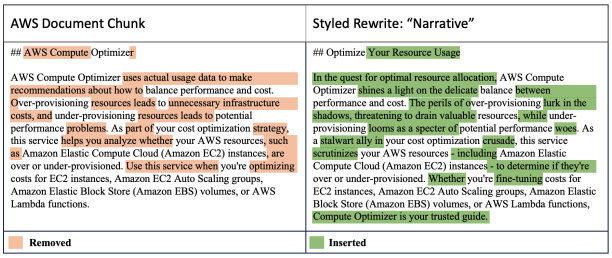

Figure 8: Example of LLM-generated style variation using narrative style prompt

B.3 NLP-Based Style Perturbations

Our approach included systematic perturbation of AWS content using specialized NLP algorithms that introduced realistic style guide violations while maintaining document structure and semantic meaning:

(1) **Date Format Randomization**: Converted AWS standard date formats to non-compliant alternatives (e.g., from "January 15, 2025" to "15/01/25")
(2) **Numeric-to-Word Conversion**: Transformed numeric representations to word forms contrary to AWS style guidelines (e.g., changing "5 instances" to "five instances")
(3) **Terminology Violations**: Substituted AWS-approved terms with alternatives from the "Do Not Use" list (e.g., changing "primary-replica" to "master-slave")
(4) **Service Name Perturbation**: Altered AWS service names by removing AWS prefixes or using deprecated/embargoed terminology
(5) **Word/Phrase Repetition**: Duplicated words or phrases within text to simulate common writing errors
(6) **Homophone Swap**: Replaced words with phonetically similar alternatives (e.g., "there" for "their")
(7) **Character Case Modification**: Changed capitalization patterns of words, particularly in service names and technical terms
(8) **Filler Word Augmentation**: Inserted unnecessary qualifier phrases like "it seems that" or "basically" based on part-of-speech recognition
(9) **Word Deletion**: Randomly removed words to create grammatically incorrect or incomplete sentences
(10) **Punctuation Perturbation**: Modified, added, or removed punctuation with weighted probabilities

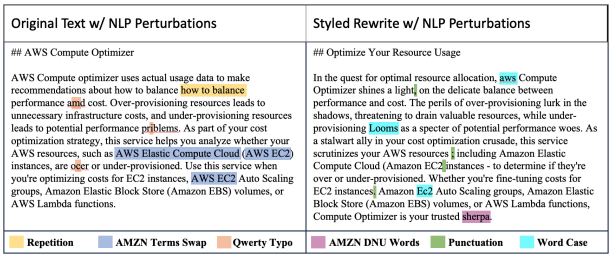

Figure 9: Example of NLP-based error insertion showing multiple style guide violations

B.4 Markdown Cleanup Prompt

For preprocessing source documents before training, we utilized the following LLM prompt:

```
You are an expert markdown cleanup engine that's tasked with scrubbing website text for any obvious html to markdown format conversion errors. You are given markdown text as input, with which you then fix the clear text conversion errors (formatting, characters decoding errors, spacing, etc) without altering or adding to the underlying content. Once the text is cleaned, output the cleaned markdown without any prefixing.

REMOVE/CORRECT any instances of:

* Obvious markdown formatting issues (spacing, indents, etc.)
* Encoding errors (e.g., commas replace with non-ascii or similar)
* Repeating sections obviously caused by a website scraping issue
* Malformed table format (e.g. any content that appears to be scraped and is not properly aligned)
* Repeating newlines
* Newlines in lists (there should be no newline between list items)
* Obvious textual errors such as spelling, grammar, punctuation errors, repeated words/sentences

Output Directions:

* Make sure you keep ALL of the input content, do not delete any content unless you have to.
* Preserve the original markdown content, formatting, including headers, lists, and links.

YOU MUST WRAP THE GENERATED CLEAN MARKDOWN CONTENT IN TAGS LIKE SO:

<clean_markdown_content>
...
</clean_markdown_content>
```

B.5 Data Quality Control

To ensure dataset integrity, we implemented multiple verification steps:

- **Content Embedding Analysis**: Cosine similarity between original and styled embeddings to verify semantic preservation (minimum threshold: 0.85)
- **Style Embedding Analysis**: KL divergence between style distributions to confirm sufficient style differentiation
- **Style Guide Compliance Verification**: Automated checks against AWS style guidelines for both original and perturbed content
- **Human Verification**: Manual review of a stratified sample (1% of dataset) to confirm quality and catch edge cases

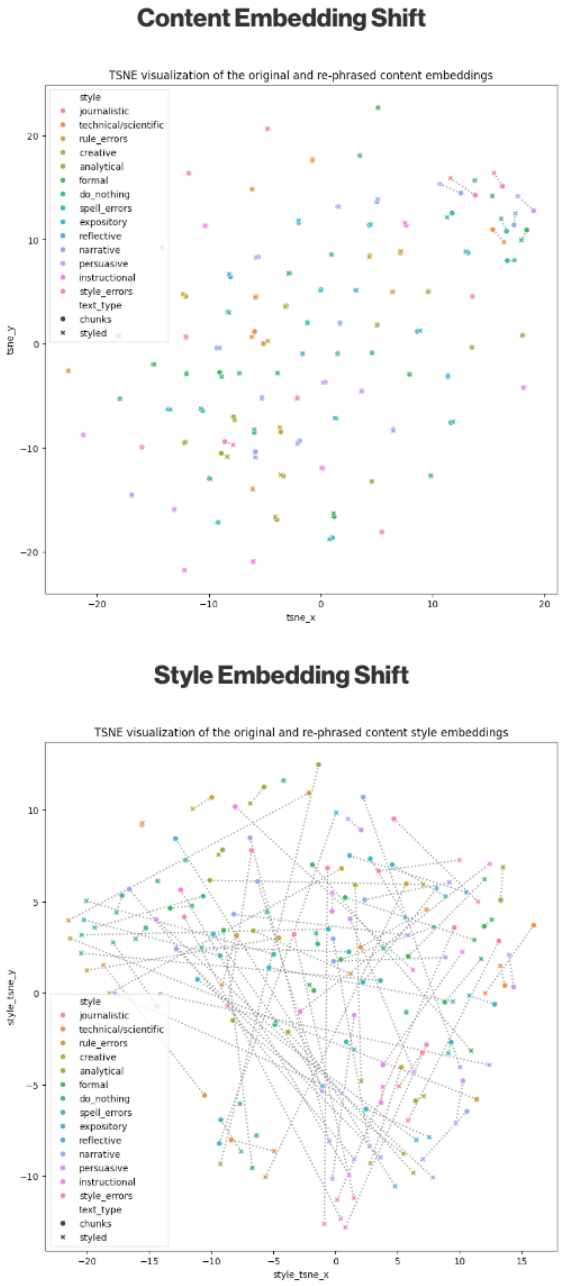

Figure 10: Analysis of semantic preservation vs. style divergence in the training dataset

C Evaluation Methodology

C.1 Comprehensive Evaluation Framework

Our evaluation methodology addresses the inherent challenges in style transfer assessment:

- **Subjectivity**: Utilizing comparative rather than absolute judgments to improve reliability
- **Lack of ground truth**: Implementing bidirectional comparison to compensate for imperfect reference data
- **Content-style balance**: Measuring multiple dimensions to capture the content-style tradeoff
- **Domain specificity**: Developing AWS-specific evaluation criteria
- **Metric integration**: Combining traditional metrics with LLM-based assessment

C.2 LLM-as-Judge Evaluation Prompt

Our evaluation leverages large language models as judges using the following prompt:

```
"""Compare <output_A> and <output_B> against the target <target> text given the <input> text was the input to two style transfer models intended to generate text that is as close to the <target> as possible. Score each metric from 0.0 to 1.0:
   1. Style Transfer Accuracy
   - 0.0 = completely different style from target
   - 1.0 = perfectly matches target style
   2. Grammatical Correctness
   - 0.0 = severe grammatical errors compared to target
   - 1.0 = perfect grammar
   3. Tone Consistency
   - 0.0 = inconsistent/inappropriate tone compared to target
   - 1.0 = tone is perfectly aligned with the target
   4. Semantic Coherence
   - 0.0 = completely different topic/meaning from target
   - 1.0 = perfectly matches target topic/meaning
   5. Vocabulary Appropriateness
   - 0.0 = inappropriate/different vocabulary than target
   - 1.0 = perfectly matches target vocabulary
   6. Content Preservation
   - 0.0 = completely different content/format from the target, include a lot of additional information
   - 1.0 = perfectly preserves the target content and format
   7. Fluency
   - 0.0 = unnatural/incoherent text
   - 1.0 = perfectly natural and coherent

   8. Do Not Change Score
   - If <input> contains <|dnc_start|>[... content ...]<|dnc_end|>:
   * 1.0 if content appears exactly as is in output
   * 0.0 if changed or missing
   - 0.5 if <|dnc_start|> and <|dnc_end|> tags does not appear in the <input> text
   Compare <output_A> to <output_B>:
   Relative Quality: -1.0 (<output_A> is much worse than <output_B>) to +1.0 (<output_A> is much better than <output_B>)
   <input>
   {restyled}
   </input>
   <output_A>
   {generated_a}
```

```
        </output_A>
        <output_B>
        {generated_b}
        </output_B>
        <target>
        {original}
        </target>
        Return only this Python dictionary:
        {{
            "output_A": {{
                "style_transfer_accuracy": float,
                "grammatical_correctness": float,
                "tone_consistency": float,
                "semantic_coherence": float,
                "vocabulary_appropriateness": float,
                "content_preservation": float,
                "fluency": float,
                "do_not_change": float
            }},
            "output_B": {{
                "style_transfer_accuracy": float,
                "grammatical_correctness": float,
                "tone_consistency": float,
                "semantic_coherence": float,
                "vocabulary_appropriateness": float,
                "content_preservation": float,
                "fluency": float,
                "do_not_change": float
            }},
            "overall": {{
                "relative_quality": float,
                "comments": "brief_evaluation"
            }}
        }}
        All scores must be floats. Regular metrics: 0.0 to
    1.0. Relative quality: -1.0 to 1.0. Return only the
    dictionary, no explanations.
    """
```

C.3 Bidirectional Evaluation Methodology

To reduce potential biases in evaluation, we conducted each assessment twice with reversed text ordering (A<->B) and combined the results. This methodology leverages the cognitive principle that both humans and AI systems make more reliable comparative judgments than absolute ones (the "contrast effect").

D Additional Experimental Results

D.1 Detailed Performance Comparison

Table 2: Comprehensive Performance Comparison with Baseline

Metric	Onoma	Baseline	Delta (%)
Style Transfer Accuracy	83%	23%	+269%
Grammatical Correctness	95%	72%	+31%
Tone Consistency	89%	49%	+83%
Semantic Coherence	93%	46%	+101%
Vocabulary Appropriateness	87%	49%	+78%
Content Preservation	87%	29%	+205%
Fluency	90%	52%	+74%
Do Not Change Score	95%	32%	+197%

E Example System Outputs

E.1 Baseline vs. Onoma Comparison

The following examples demonstrate Onoma's superior performance compared to the baseline Flan-T5 model with minimal instructions.

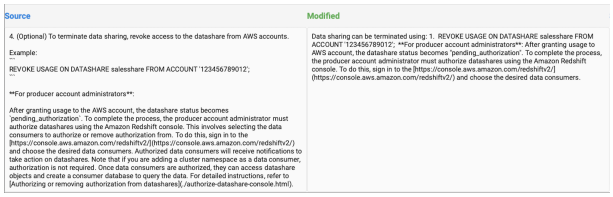

Figure 11: Base Flan-T5 Rewrite

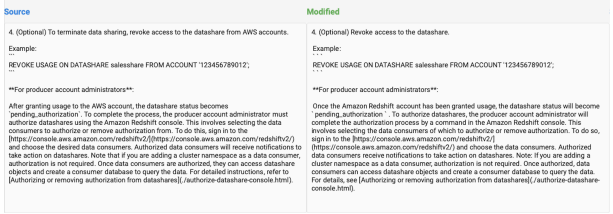

Figure 12: Fine-Tuned Flan-T5 Rewrite (Onoma)

E.2 Special Tag Handling Examples

Onoma's ability to preserve verbatim content within special tags is demonstrated below:

Input with Tags:
To get started with Amazon S3, <|dnc_start|>use the following command: `aws s3 ls`<|dnc_end|>. This will list all your S3 buckets.

Onoma Output:
To get started with Amazon Simple Storage Service (Amazon S3), <|dnc_start|>use the following command: `aws s3 ls`<|dnc_end|>. This command lists all your Amazon S3 buckets.

Figure 13: Example of content preservation using special tags

E.3 Markdown Formatting Preservation

Onoma maintains document structure integrity while applying style transformations:

Original Markdown:
Getting Started with AWS
You can use these services:
* EC2 - For virtual servers
* S3 - For storage
* DynamoDB - For NoSQL database

Onoma Output:
Getting Started with AWS
You can use these services:
* Amazon Elastic Compute Cloud (Amazon EC2) - For virtual servers
* Amazon Simple Storage Service (Amazon S3) - For storage
* Amazon DynamoDB - For NoSQL database

Figure 14: Example of Markdown formatting preservation with service name expansion

E.4 Complex Formatting Examples

Onoma effectively handles complex document structures, including tables:

Complex Input:
Using AWS Services Together
Service	Purpose	Common Uses
EC2	Compute	Web servers
S3	Storage	Static websites
RDS	Database	Data storage

Onoma Output:
Using AWS Services Together
Service	Purpose	Common Uses
Amazon EC2	Compute	Web servers
Amazon S3	Storage	Static websites
Amazon RDS	Database	Data storage

Figure 15: Example of complex formatting preservation with AWS style application