SemAug: Shaping the Future of Semantically-Enriched, Format-Specific Data Augmentation

Anonymous ACL submission

Abstract

In the realm of artificial intelligence, the significance of high-quality data cannot be overstated, especially data that adheres to stringent formatting rules and structures. Addressing this need, our study introduces an advanced data augmentation method specifically designed for format-specific datasets. This method utilizes the capabilities of Large Language Models (LLMs) to generate data that not only meets the rigid formatting criteria but also maintains the integrity of the information. Central to our approach is the integration of specific format requirements into natural language prompts, which guides the LLMs to produce precisely formatted outputs. A salient feature of our approach is its self-evaluative mechanism, which autonomously assesses the semantic quality of the augmented data, distinguishing it from prior methodologies that require manual validation, thereby streamlining the augmentation process. Our research represents a pioneering step forward, enabling more efficient enhancement of datasets that demand exacting format adherence without the extensive resource investment typically associated with such tasks.

1 Introduction

In the era of artificial intelligence (AI), large language models (LLMs) have emerged as transformative tools for natural language understanding and generation tasks. These models have demonstrated remarkable capabilities in diverse applications, including text completion, language translation, and text summarization. However, as these AI systems delve deeper into specialized domains, their adaptability and robustness in handling intricate and rigid data formats have come under scrutiny.

A pivotal challenge hindering the progress of robust AI lies in the intricacies of training data. While deep learning models flourish on vast and varied datasets, obtaining such data, especially in structured or complex formats, remains a daunting endeavor. The process is often hindered by cost, time, and occasional unfeasibility due to issues like privacy or rarity of examples. This has led to the exploration of data augmentation techniques to bolster training datasets. Extensive research has been conducted on data augmentation, and its widespread utilization in natural language processing (NLP) tasks has proven effective in addressing issues such as constrained data availability, uneven sample distribution, and domain-specific challenges like named entity recognition and relation extraction in biomedical text or code generation tasks. Through the incorporation of supplementary samples and introducing diversity, data augmentation contributes to enhancing model robustness and adaptability, simultaneously mitigating overfitting concerns.

Commonly employed data augmentation methods encompass the use of synonyms, antonyms, and variations in word forms, all of which serve to augment the quantity and diversity of training data (Niu and Bansal, 2018). Some traditional techniques include synonym replacement, random deletion and random insertions (Feng et al., 2021). Other methods utilize language models to generate reliable samples for more effective data augmentation, including back translation (Sennrich et al., 2016) and word vector interpolation in the latent space (Jindal et al., 2020). However, existing data augmentation methods often fall short when confronted with the nuances of complex data structures. These methods, while effective in general contexts, tend to struggle with preserving the specificity and intricacies inherent in complex formats (Feng et al., 2021; Shorten et al., 2021; Bayer et al., 2021).

Considering these challenges, the spotlight turns to prompt engineering, which holds promise as a solution. This research posits prompt engineering as a potential tool to harness the power of LLMs for data augmentation, particularly suited for intricate data formats. It seeks to explore how LLMs, when

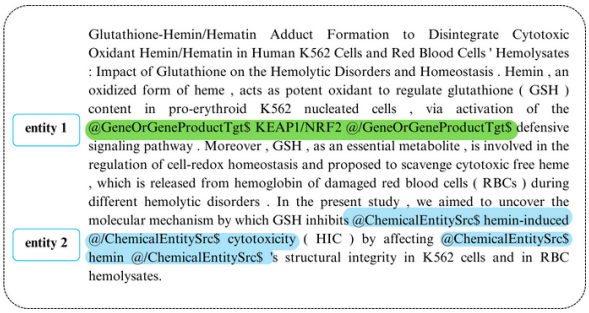

Figure 1: Example of an input in Biomedcai Relation Extraction task. The entity is wrapped in the entity type's tags. Different colors mean different entities in the entity pair. The task is to extract relation between the entity pair.

guided by carefully crafted prompts, can synthesize data that not only maintains the rigidity and context of complex formats but is also virtually indistinguishable from genuine entries. The overarching aim is to bolster AI performance across an array of tasks, including natural language understanding, sentiment analysis, and content generation, ensuring they thrive even in contexts laden with complex data constraints.

In the realm of biomedical text data, the intricacies of rigid formats not only pose substantial challenges for data augmentation but also necessitate meticulous attention to preserve the inherent scientific and terminological precision. As shown in Figure 1, an example of training input to the model is a paragraph that contains an entity pair, with entity tags placing around each entity. The augmentation method should paraphrase the whole paragraph while keeping the entity tags and entity text the same, because biomedical named entities they are lexical sensitivities, which means that they should not be altered without meticulous scrutiny and analysis. This makes it complex for creating augmentation data in biomedical domain, which we should carefully consider the semantic delicacies in the biomedical entities.

Additionally, when it comes to structuring datasets from programming languages, the requirement for preserving syntax and semantic accuracy introduces a distinct set of challenges. In programming languages, every character, from a brace to a semicolon, carries significance; a minor alteration can lead to a completely different execution behavior. This sensitivity is especially pronounced in datasets that include numerical values or complex syntactic structures, common in software engineering and computational tasks. For instance, in a code snippet, changing a variable name or altering a loop structure could inadvertently change the algorithm's logic or functionality. This highlights the criticality of maintaining syntactic integrity and the original semantic context in any augmentation process applied to code datasets. The augmentation method must be intricately designed to understand and respect the language's grammar rules, operational semantics, and the contextual role of each code element. Moreover, in the realm of numerical data within code, precision is paramount. Numerical values often represent parameters, dimensions, or constants whose exact values are crucial for the correct functioning of the code. Altering these values without a deep understanding of their context and impact can lead to incorrect results or system failures, particularly in fields like numerical computing or algorithmic processing.

In this paper, we introduce SemAug (Specified-Rule **Aug**mentation and **Sem**antic-Quality), a novel approach combining the SpecRule Augmentor with the SemQ filter to overcome the challenges inherent in complex data domains, particularly in biomedical datasets. Unlike traditional data augmentation techniques that struggle with the intricate formats and rigid constraints of such datasets, SemAug employs prompt engineering to navigate these complexities effectively.

- SpecRule Augmentor under SemAug is designed to generate diverse and realistic data variations, adhering strictly to the scientific accuracy and terminological precision required in biomedical contexts.

- SemQ Filter rigorously evaluates the augmented data, ensuring its adherence to the high-quality standards essential for maintaining scientific validity. Our method contributes significantly to the field by enabling the augmentation of biomedical data in a manner that aligns seamlessly with its specialized requirements.

Moreover, SemAug facilitates the development of machine learning models proficient in handling and extracting insights from these enriched and complexly formatted datasets. This innovative approach marks a pivotal advancement in data augmentation, paving the way for enhanced application of machine learning and artificial intelligence in domains where data complexity and specificity have

traditionally posed significant challenges.

2 Related Works

2.1 Large Language Models

In recent developments within the domain of large language models (LLMs), their proficiency in mimicking human language through extensive training on expansive textual datasets has been well-documented (Brown et al., 2020; Chowdhery et al., 2022; Hoffmann et al., 2022). Crucially, these models exhibit an exceptional ability to follow human-issued instructions, yielding significant zero-shot performance (Wei et al., 2022; Dong et al., 2023). In our research, we harness this particular trait of LLMs, employing them to generate datasets in specific formats by inputting prompts that encompass our defined requirements and rules. This methodology aligns with the broader narrative in LLM research, where the adaptability and responsiveness of these models to guided inputs are being increasingly explored for tailored data generation purposes.

2.2 Data Augmentation

Data augmentation, a common strategy to prevent overfitting in limited data scenarios, typically involves minor, label-preserving modifications to the existing data (Wei and Zou, 2019; Fadaee et al., 2017). Recent studies have explored the use of Large Language Models (LLMs) for this purpose. Specifically, they use few-shot data as demonstrations and prompt LLMs to generate new data (Yoo et al., 2021; Sahu et al., 2022). The capability of LLMs to blend a few-shot dataset and create analogous data has been widely recognized. Lin et al. (2023) introduced the use of Pointwise V-information to sift through and remove non-beneficial data from these generated sets. Most recent approaches involve using models like ChatGPT and GPT-4 for data generation, leading to notable improvements in performance (Dai et al., 2023; Whitehouse et al., 2023). Furthermore, Cheng et al. (2023) demonstrated the efficacy of using GPT-3 generated data to enhance sentence embeddings through contrastive learning. Our methodology diverges by utilizing LLMs as an all-encompassing solution specifically for generating and evaluating augmented data within rigid or specific data formats, effectively obviating the requirement for manual annotation.

2.3 Few-shot and zero-shot learning

A zero-shot prompt poses a query to the LLM on topics it hasn't been explicitly trained on. The term "zero" in "zero-shot" indicates that the LLM lacks specific training on the particular task or question in the prompt. The "shot" denotes providing an example to the LLM. Thus, "zero-shot" implies that the model wasn't trained for the exact task or question and the prompt doesn't offer an illustrative example for reference. An instance of a zero-shot prompt is translation tasks. Even though the model might not have received particular training samples for translations, their vast linguistic training allows them to generalize and provide probable translations without dedicated training for the task. Few-shot prompts resemble zero-shot prompts in that the model hasn't been tailored for the specific question or task. But, unlike zero-shot prompts, few-shot prompts offer an in-context example within the request to guide the model.

3 Methodology

We apply the idea of using examples in few-shot learning; however, the largest difference here is that although examples are used in the prompt, there is no given label for them, instead, the focus on the format and meaning of the dataset is considered to be more important here. This means that the concept used here is zero-shot learning. Figure 2 shows the overflow of our method. There are two main steps here: creating data augmentation following requirements (SpecRule Augmentor) and self-evaluating the quality of generated data (SemQ Filter)

3.1 Data Augmentation with User's Requirements

Given a original dataset containing n samples $D_o = \{d_1, d_2, ..., d_n\}$, the data augmentation D_a can be obtained through:

$$D_a = \{\text{SpecRuleAugmentor}(d_i) \mid d_i \in D_o\} \quad (1)$$

where SpecRuleAugmentor is a specified prompt used in the LLM in Figure 3

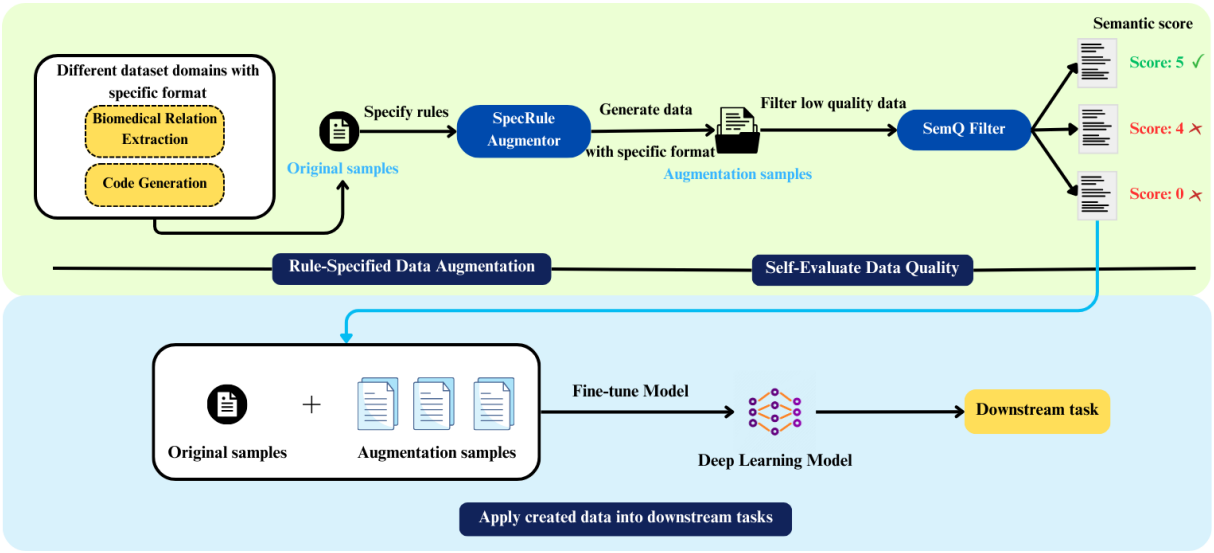

Figure 2: Overall Flow of The Framework

You are creating data augmentation through <REQUEST>. The data created must be in the format: <OUTPUT'S FORMAT RULE>. For example 1...n, given <INPUT EXAMPLES + FORMAT'S RULE>. Here is your task, given input: <INPUT>, please follow the rules above and <REQUEST>

Figure 3: Requirement-Specified Prompt for Generating Rigid Data Augmentation (SpecRule Augmentor)

Instead of focusing on letting the model know how to create a pair of training data (input + label), the prompt concentrates on emphasizing the model to create data following requirements of correct format. An example of SpecRule Augmentor applied in Biomedical Relation Extraction and Code Generation task is shown in Figure 6 and Figure 7 in Appendix correspondingly.

3.2 Self-Evaluative Data Augmentation's Quality

Following augmentation, we apply the SemQ Filter to each document $d_{i'}$ in D_a to assess its quality. The SemQ Filter functions as a prompt-based scoring mechanism, as illustrated in Figure 4, and assigns a quality score to each document. This process is represented as:

$$\text{Score}(d_{i'}) = \text{SemQFilter}(d_{i'}) \text{ for each } d_{i'} \in D_a \quad (2)$$

To ensure the augmented dataset's quality, We apply a threshold-based filtering, retaining only those documents from D_a with a SemQ Filter score above 4. The augmented dataset, D_{filtered}, consists of documents from D_a that meet the quality criterion below:

$$D_{\text{filtered}} = \{d_{i'} \in D_a \mid \text{Score}(d_{i'}) > 4\} \quad (3)$$

The final dataset D_{final} is formed by combining the original dataset D_o with the high-quality, filtered augmented data D_{filtered}:

$$D_{\text{final}} = D_o \cup D_{\text{filtered}} \quad (4)$$

Your task is to conduct a semantic alignment evaluation between an 'Original' paragraph and its 'Paraphrased' counterpart. Carefully analyze how well the paraphrased version maintains the meaning, intent, and key information of the original. After your assessment, provide a score exclusively in the range of 0 to 5. Here, 0 represents a complete loss of the original meaning (indicating no semantic alignment), and 5 signifies an excellent preservation of the original's meaning (indicating perfect semantic alignment). Here is your task, given 'Original' version: <ORIGINAL>; 'Paraphrased' version: <PARAPHRASED>, please evaluate their semantic meaning and provide only a single number between 0 and 5 that reflects the level of semantic consistency between the original and paraphrased texts.

Figure 4: Semantic Checking Prompt for Quality Assurance (SemQ Filter)

The SemQ Filter prompt tells the model to do a self-evaluation on both format and semantic checking. Moreover, instead of using "True", "False" evaluation in the prompt, we see that telling the model to give a score from 0-5 can help identify

real high-quality data in the generated samples. Example of SemQ scores given for different generated samples can be seen in Table 5 in Appendix.

4 Experiments

4.1 Dataset

BioRED Dataset (Luo et al., 2022): The task is to extract biomedical relation across multiple entity types with multiple relation types. There are multiple entity types (gene, chemical, disease, ...) and relation pairs (disease-gene, chemical-chemical) at the document level. There are a total of 600 PubMed abstracts, with 400 documents for training, 100 for dev and 100 for validation.

EvolInstruct-80k-v1 dataset[1] : EvolInstruct-80k-v1 dataset (license cc-by-nc-sa-4.0) is an open-source dataset generated using Evol-Teacher (Luo et al., 2023). The dataset contains about 78000 instructions and their code solution. We use this dataset to fine-tune model for Code Generation task.

HumanEval (Chen et al., 2021): HumanEval contains 164 handwritten Python programming problems with a function signature, docstring, body, and several unit tests. It focuses on testing the execution ability of the code using the evaluation metric pass@k.

MultiPL-E dataset (Cassano et al., 2022): MultiPL-E is a benchmark for evaluating large language models for code generation that supports 18 programming languages. It takes the OpenAI "HumanEval" Python benchmark and uses little compilers to translate them to other languages. In this paper, we do experiments on C++, Java, JavaScript (JS), R, Bash, Rust.

Grade School Math 8K (GSM8K) (Cobbe et al., 2021): GSM8k is a dataset of 8.5K high-quality linguistically diverse grade school math word problems. The dataset was created to support the task of question answering on basic mathematical problems that require multi-step reasoning.

4.2 Experimental Design

Data augmentation: We use open-source LLM OpenChat-3.5 (Wang et al., 2023) as the model for SpecRule Augmentor and SemQ Filter.

Biomedical Relation Extraction: BioRED dataset is used for training and evaluating. We fine-tune PubMedBERT (Gu et al., 2020) in 50 epochs with learning rate 1e-5, batch size 10.

Code Generation: For fine-tuning our model, we utilized the EvolInstruct-80k dataset, selecting Deepseek-Coder-1.3b-Instruct (DeepSeek, 2023) as the foundational model. The model is fine-tuned in 800 steps, with learning rate 2e-5. For evaluation purposes, we employed datasets including HumanEval, MultiPL-E, GSM8K. Since there is a difference between the performance stated in DeepSeek (2023) and those obtained through our use of the bigcode-evaluate-harness (Ben Allal et al., 2022), we opted to rely on the performance evaluations derived from bigcode-evaluate-harness for all models to ensure consistency in comparative analysis. Temperature is set to 0.2 for all models, and the primary metric used for evaluating model performance in this task was pass@1. Our model's performance was benchmarked against several others, including CodeGeeX2 6B (Zheng et al., 2023), StarCoder 15B (Li et al., 2023), CodeLlama 7B, CodeLlama-Instruct 7B, CodeLlama 13B (Roziere et al., 2023).

Mathematics Reasoning: We additionally assess the influence on the performance of a model that has been fine-tuned for the code generation task when applied to a different task. This involves comparing a model already fine-tuned for code generation with various other Large Language Models, including Llama 7B, Llama 13B (Touvron et al., 2023), Minerva 8B (Lewkowycz et al., 2022), CodeGeeX2 7B, StarCoder 15B, and Deepseek-Coder-1.3b-Instruct.

Table 1: Data augmentation in different dataset domains. Ori: Original, Aug: After Augmentation, Sem: After semantic filtering.

Dataset	Ori	Aug	Sem	Success rate
BioRED	28233	87519	60188 (**+31955**)	53.9%
Evol Instruct	78381	156762	133568 (**+55187**)	70.4%

4.3 Results

Generated data: In this study, we embarked on an extensive data augmentation initiative spanning two distinct and challenging domains: biomedical relation extraction (BioRED) and code generation (Evol Instruct). Table 1 presents a comprehensive overview of the data augmentation output and sub-

[1] https://huggingface.co/datasets/nickrosh/Evol-Instruct-Code-80k-v1.

sequent filtration efficacy across these domains. Notably, the success rates post-semantic validation—courtesy of our SemQ filter—demonstrate task-specific variability, which we attribute to the inherent complexity and structural rigidity of each dataset. An example result of SemQ can be seen in Table 5. For the biomedical relation extraction task, the data augmentation process involves intricate manipulation of text interspersed with rigorously formatted tags such as @GeneOrGeneProduct and @ChemicalEntity. These tags, which require precise placement and pairing of opening (@GeneOrGeneProduct) and closing (@/GeneOrGeneProduct) markers (detailed example can be seen in Table 6 in Appendix), significantly elevate the complexity of the augmentation task. Consequently, a substantial proportion of the augmented dataset failed to meet the stringent criteria set forth by the SemQ Filter, as it rigorously scrutinizes the congruence of these tags within the augmented narratives. This meticulous semantic checking process is pivotal in ensuring the scientific accuracy of the augmented data, a non-negotiable aspect in biomedical research where precision is paramount.

On the other hand, the Code Generation task, while still subject to strict syntactic and semantic constraints, exhibited a higher pass rate post-augmentation. In this domain, the augmentation process must ensure that the syntactical structure and numerical values within the code remain unaltered to preserve the original logic and functionality. Despite these constraints, the augmentation method proved to be more adaptable to the programming context, resulting in a greater volume of data successfully passing through the SemQ Filter. This higher success rate suggests that our augmentation approach is effectively attuned to the nuances of programming languages, capable of generating syntactically sound and semantically coherent code snippets that align with the original intention of the code's author. The disparity in the success rates between the two tasks underscores the challenges that specialized domains pose to data augmentation techniques. In domains like biomedical research, where the data is densely annotated and laden with domain-specific terminologies, the augmentation process must navigate a complex landscape of linguistic and semantic rules. Similarly, in programming language datasets, the augmentation must contend with the rigid syntax and operational semantics that are fundamental to the code's execution. These findings highlight the need for augmentation strategies that are not only context-aware but also capable of discerning and adhering to the nuanced rules that govern data in specialized fields.

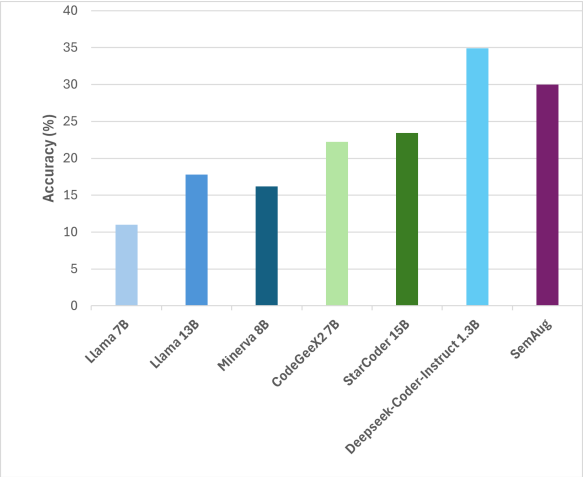

Figure 5: Performance on Math Reasoning Task.

Biomedical Relation Extraction Task: Table 2 shows that our methodology demonstrates substantial improvement across all three evaluation schemas. When augmented data is introduced, SemAug not only excels in extracting entity pairs within a relation but also in discerning the type of relation and further characterizing the novelty of the entity pairs. This is evident from the marked 8.8% enhancement in the F1-score, showcasing the robustness of our approach in handling the intricacies of biomedical data and contributing to a more nuanced understanding and extraction of relationships within it.

Code Generation Task: It is clear from Table 3 that SemAug demonstrates a significant enhancement over the base model, Deepseek-Coder-Instruct, across various programming languages. Particularly notable are the improvements in languages like Java and C++, where SemAug shows an increase of 5.0% and 7.0% respectively, suggesting that our method has effectively leveraged the underlying LLM to produce superior data augmentation outcomes. The consistent increments across multiple languages affirm the robustness and versatility of SemAug, validating its potential as a powerful tool for tasks requiring nuanced language comprehension and generation.

Mathematics Reasoning Task: SemAug was fine-tuned exclusively on a code instruction dataset

Table 2: Evaluation results on RE task comparison when added augmented data: extracting the entity pairs within a relation, second schema: extracting the entity pairs and the relation type and the third schema: further labeling the novelty for the extracted pairs. All numbers are F-scores. The <G,D> is the concept pair of the gene (G) and the disease (D). The columns of those entity pairs present the RE performance in F-scores. Bold denotes best model

Eval Schema	Methods	All	<G,D>	<G,G>	<G,C>	<D,V>	<C,D>	<C,V>	<C,C>
Entity pair	BERT-GT	72.1	63.8	78.5	77.7	69.8	76.2	58.5	74.9
	PubMedBERT	72.9	67.2	78.1	78.3	**67.9**	76.5	**58.1**	78.0
	SemAug	**78.3**	**76.1**	**86.2**	**85.7**	66.0	**78.2**	38.1	**85.9**
		↑ 5.4	↑ 8.9	↑ 7.1	↑ 7.4	↓ 1.9	↑ 1.7	↓ 20	↑ 7.9
+Relation type	BERT-GT	56.5	54.8	63.5	60.2	42.5	67.0	11.8	52.9
	PubMedBERT	58.9	56.6	66.4	59.9	**50.8**	65.8	25.8	54.4
	SemAug	**64.4**	**64.3**	**76.1**	**64.3**	47.0	**72.2**	9.5	**59.7**
		↑ 5.5	↑ 7.7	↑ 9.7	↑ 4.4	↓ 3.8	↑ 6.4	↓ 16.3	↑ 5.3
+Novelty	BERT-GT	44.5	37.5	47.3	55.0	36.9	51.9	11.8	48.5
	PubMedBERT	47.7	40.6	54.7	54.8	42.8	51.6	**12.9**	50.3
	SemAug	**56.5**	**51.1**	**69.1**	**57.9**	**44.9**	**78.2**	9.5	**56.5**
		↑ 8.8	↑ 10.5	↑ 14.4	↑ 3.1	↑ 2.1	↑ 26.6	↓ 3.4	↑ 6.2

Table 3: Evaluation results on Completion Task. The evaluation metric is pass@1. Bold denotes best model. A demarcation line is placed between the final two models to emphasize our primary objective is to compare the enhancements our method brings relative to the base model.

Model	Size	HumanEval	MultiPL-E					
			C++	Java	R	Rust	Bash	JS
CodeGeeX2	6B	33.5	29.2	23.5	6.8	17.9	6.3	24.8
StarCoder	16B	30.4	28.6	28.5	13.6	21.2	9.3	31.7
CodeLlama	7B	30.0	25.5	29.2	16.8	26.3	8.1	31.8
CodeLlama-Istruct	7B	45.7	8.7	28.8	14.9	26.3	8.7	33.1
CodeLlama	13B	35.1	34.1	32.2	**22.4**	**28.2**	**11.8**	38.3
Deepseek-Coder-Instruct	1.3B	51.8	29.2	31.0	12.4	23.6	7.5	39.1
SemAug	1.3B	**53.7**	**34.2**	**38.0**	13.7	24.4	11.2	**39.1**
		↑ 1.9	↑ 5.0	↑ 7.0	↑ 1.3	↑ 0.8	↑ 3.7	↑ 0.0

that did not contain data explicitly related to mathematics. As shown in Figure 5, despite the dataset's focus being misaligned with the task, SemAug exhibits commendable resilience, with only a marginal decrease in performance post-fine-tuning. This slight dip, however, underscores the importance of task-aligned data for fine-tuning, to fully harness the model's capabilities in domain-specific tasks. The fine-tuning process, although not directly contributing to performance in this task, demonstrates the model's flexibility and provides insights into the transferability of learned representations across different contexts.

5 Conclusion and Future Work

This paper introduces a groundbreaking approach to format-specific data augmentation, uniquely coupled with a self-evaluative mechanism. These datasets are characterized by their strict structural requirements, which might include specific tagging systems in biomedical data or syntactic rules in programming languages. Unlike general, non-format-specific datasets, which can accommodate a broader range of augmentation techniques, format-specific datasets demand precise adherence to their unique formatting rules to retain data integrity and utility.

Our method distinguishes itself by seamlessly adapting to these stringent requirements across diverse domains without the need for extensive mod-

ifications to the existing system architecture. The key to this flexibility is the strategic alteration of natural language prompts, which guides the augmentation process to produce outputs that conform to the particular format constraints. This capability illustrates the versatility and efficacy of SemAug in addressing the challenges posed by structured datasets. Additionally, a key advancement of SemAug over previous methods is its autonomous capability to evaluate and ensure the semantic quality of augmented data. Traditional approaches often rely on human oversight, which can be time-consuming and prone to error. SemAug circumvents this dependency, offering a more efficient and reliable means of data augmentation that minimizes the need for manual intervention. Moreover, the accessibility of our methodology to non-experts is another significant aspect. By reducing reliance on extensive programming knowledge, it opens up opportunities for a broader range of users to engage in advanced data processing and augmentation. This democratization of technology can lead to innovative applications and solutions, previously unattainable due to technical barriers.

In conclusion, this research not only provides a novel technical solution but also fosters a more inclusive approach to complex data processing. As we continue to explore and refine this method, we anticipate it will unlock new possibilities and applications, significantly contributing to the advancement of various fields that rely on specialized data formats.

Limitations

The effectiveness of our method is closely tied to the performance of Large Language Models (LLMs). This implies that the quality of our data augmentation output, as well as the efficacy of the self-evaluation process, is contingent on the accuracy and reliability of the LLMs' results. Hence, suboptimal performance from these models could potentially impact the overall quality and reliability of the augmented data generated by our method. In this study, we have utilized a single LLM for data augmentation purposes. Future investigations should explore the application of various LLMs to determine their influence on the ultimate effectiveness of the model.

Ethics Statement

We comply with the ACL Code of Ethics.

References

Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2021. A survey on data augmentation for text classification. *CoRR*, abs/2107.03158.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. 2022. Multipl-e: A scalable and extensible approach to benchmarking neural code generation.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Qinyuan Cheng, Xiaogui Yang, Tianxiang Sun, Linyang Li, and Xipeng Qiu. 2023. Improving contrastive learning of sentence embeddings from ai feedback.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben

Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

DeepSeek. 2023. Deepseek coder: Let the code write itself. `https://github.com/deepseek-ai/DeepSeek-Coder`.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Amit Jindal, Arijit Ghosh Chowdhury, Aniket Didolkar, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020. Augmenting NLP models using latent feature interpolations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6931–6936, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Yen-Ting Lin, Alexandros Papangelis, Seokhwan Kim, Sungjin Lee, Devamanyu Hazarika, Mahdi Namazifar, Di Jin, Yang Liu, and Dilek Hakkani-Tur. 2023. Selective in-context data augmentation for intent detection using pointwise V-information. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1463–1476, Dubrovnik, Croatia. Association for Computational Linguistics.

Ling Luo, Po-Ting Lai, Chih-Hsuan Wei, Cecilia N Arighi, and Zhiyong Lu. 2022. BioRED: a rich biomedical relation extraction dataset. *Briefings in Bioinformatics*, 23(5):bbac282.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

Tong Niu and Mohit Bansal. 2018. Adversarial over-sensitivity and over-stability strategies for dialogue models.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. 2022. Data augmentation for intent classification with off-the-shelf large language models. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. 2021. Text data augmentation for deep learning. *Journal of Big Data*, 8.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Chenxi Whitehouse, Monojit Choudhury, and Alham Fikri Aji. 2023. Llm-powered data augmentation for enhanced cross-lingual performance.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x.

A Appendix

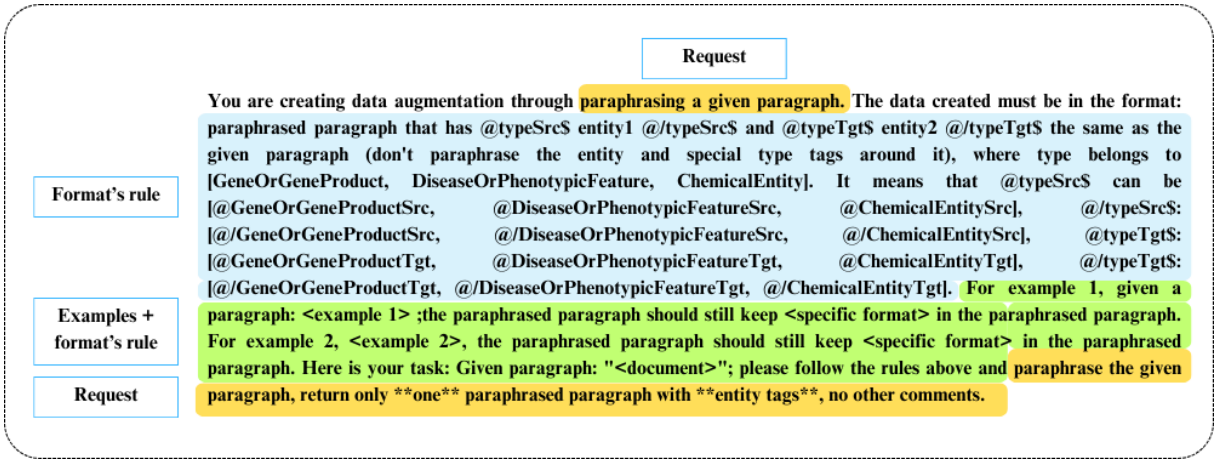

Figure 6: SpecRule Augmentor in Biomedical Relation Extraction task.

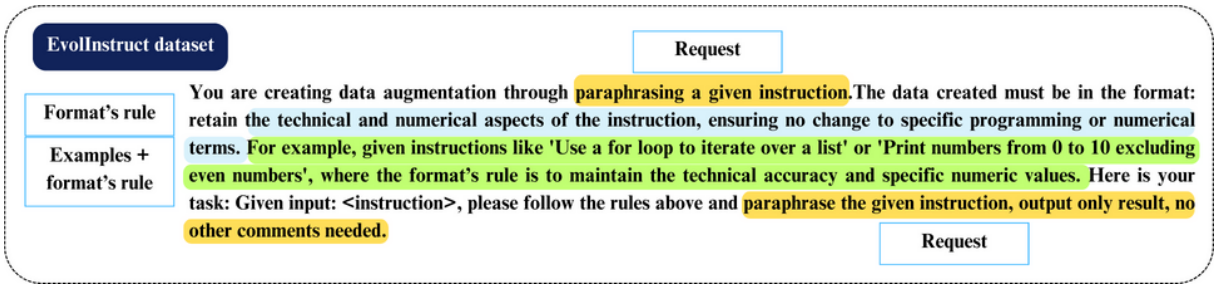

Figure 7: SpecRule Augmentor in Code Instruction task.

Table 4: Performance on Math Reasoning Dataset.

Model	Size	Accuracy
LLama	7B	11.0
LLama	13B	17.8
Minerva	8B	16.2
CodeGeeX2	7B	22.2
StarCoder	16B	23.4
Deepseek-Coder-Instruct	1.3B	**34.9**
SemAug	1.3B	30.0

Table 5: Generated samples on EvolInstruct-80k-v1 dataset. Highlights mean specific syntax or number that are kept unchanged after augmentation.

Original	Altered text	Score
Edit the existing HTML code to create an **ordered** list with the following requirements: - The ordered list should be numbered in **descending** order. - Each fruit name in the list should be displayed in **uppercase** letters. - Add a CSS style to the list items to have a background color of **yellow**. - The fruit names should be sorted in alphabetical order. - Only include fruits that start with a **vowel** in the list. - Each fruit name should be followed by a **comma**. - Add a CSS style to the list to have a border of **1px solid black**. - The **ordered** list should be displayed **horizontally** instead of **vertically**. - Add a CSS style to the list items to have a font size of **20px**. - The list should have a **maximum** of 5 items.	Modify the existing HTML code to generate an **unordered** list with the following constraints: - The list items should be numbered in **ascending** order. - Each fruit name in the list should be displayed in **lowercase** letters. - Include a CSS style to the list items to have a background color of **red**. - The fruit names should be sorted in **reverse** alphabetical order. - Only include fruits that start with a **consonant** in the list. - Each fruit name should be followed by a **semicolon**. - Add a CSS style to the list to have a border of **2px dotted blue**. - The **unordered** list should be displayed vertically instead of **horizontally**. - Add a CSS style to the list items to have a font size of **18px**. - The list should have a **minimum** of 5 items.	0
Create an object with key-value pairs using JavaScript.	**Generate** an object with key-value pairs in JavaScript **using a for loop**.	4
Write a function to determine the largest prime number **in** an **unordered** array. array = [6, 8, 1, 3, 4]	**Identify** the largest prime number **within** an **unstructured** array. array = [6, 8, 1, 3, 4]	5
Write a code to generate a random password using the random library of Python. The password should satisfy the following requirements: - The **length of the password should be** 8 characters. - The password should **contain** at least one uppercase letter, one lowercase letter, one digit, and one special character (!#$% ^&*). - The password should not **start** with a digit. - The password should not contain any consecutive repeating characters. **Note: You are not allowed to use any external libraries or built-in functions for generating passwords.**	Compose a random password using Python's random library, ensuring the following criteria: - The **password length must be** 8 characters. - The password should **include** at least one uppercase letter, one lowercase letter, one digit, and one special character (!#$% ^&*). - The password should not **commence** with a digit. - The password must not contain any consecutive repeating characters. **Without utilizing any external libraries or built-in functions for password generation, follow the aforementioned rules and rephrase the given instruction.**	5

Table 6: Generated samples on BioRED. Highlighted words denote the entities with special tags around them. Blue words stand for those modifications while keeping the same semantic meaning.

Original	Altered text
Curcumin **prevents** @ChemicalEntitySrc$ maleate-induced @/ChemicalEntitySrc$ nephrotoxicity: relation to hemodynamic alterations , oxidative stress , mitochondrial oxygen consumption and activity of respiratory complex I. The **potential** protective effect of the dietary antioxidant curcumin (120 mg/Kg/day for 6 days) **against the renal injury induced by** @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ was **evaluated**. Tubular proteinuria and oxidative stress were **induced by** a single injection of @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ (400 mg/kg) in rats. @ChemicalEntitySrc$ Maleate-induced @/ChemicalEntitySrc$ renal injury included **increase in** renal vascular resistance and in the urinary excretion of total protein, glucose, sodium, neutrophil gelatinase-associated lipocalin (NGAL) and N-acetyl b-D-glucosaminidase (NAG), upregulation of kidney injury molecule (KIM)-1, **decrease in** renal blood flow and @GeneOrGeneProductTgt$ claudin-2 @/GeneOrGeneProductTgt$ expression **besides of** necrosis and apoptosis of tubular cells on 24 h. Oxidative stress was determined by measuring the oxidation of lipids and proteins and **diminution** in renal Nrf2 levels. Studies were also **conducted** in renal epithelial LLC-PK1 cells and in mitochondria isolated from kidneys of all the experimental groups. @ChemicalEntitySrc$ Maleate @/ChemicalEntitySrc$ **induced** cell damage and reactive oxygen species (ROS) production in LLC-PK1 cells in culture. **In addition**, @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ treatment **reduced** oxygen consumption in ADP-stimulated mitochondria and diminished respiratory control index when using malate/glutamate as substrate. The activities of both complex I and aconitase were also **diminished**. All the above-described **alterations** were prevented by curcumin. It is concluded that curcumin is able to **attenuate** in vivo @ChemicalEntitySrc$ maleate-induced @/ChemicalEntitySrc$ nephropathy and in vitro cell damage. The in vivo protection was **associated to** the prevention of oxidative stress and preservation of mitochondrial oxygen **consumption** and activity of respiratory complex I, and the in vitro protection was **associated to** the prevention of ROS production .	Curcumin **protects against renal toxicity induced by** @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$: relation to **changes in** hemodynamics, oxidative stress, mitochondrial oxygen **use** and activity of respiratory complex I. The **possible** protective effect of the dietary antioxidant curcumin (120 mg/Kg/day for 6 days) **against kidney injury caused by** @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ was **looked at**. Tubular proteinuria and oxidative stress were **brought on** by a single injection of @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ (400 mg/kg) in rats. @ChemicalEntitySrc$ Maleate-induced @/ChemicalEntitySrc$ kidney injury included **increased** renal vascular resistance and urinary excretion of total protein, glucose, sodium, neutrophil gelatinase-associated lipocalin (NGAL) and N-acetyl b-D-glucosaminidase (NAG), upregulation of kidney injury molecule (KIM)-1, **decreased** renal blood flow and @GeneOrGeneProductTgt$ claudin-2 @/GeneOrGeneProductTgt$ expression **in addition to** necrosis and apoptosis of tubular cells at 24 hrs. Oxidative stress was determined by measuring the oxidation of lipids and proteins and **reduction** in renal Nrf2 levels. Studies were also **done** in renal epithelial LLC-PK1 cells and in mitochondria isolated from kidneys of all the experimental groups. @ChemicalEntitySrc$ Maleate @/ChemicalEntitySrc$ **caused** cell damage and reactive oxygen species (ROS) production in LLC-PK1 cells in culture. **Additionally**, @ChemicalEntitySrc$ maleate @/ChemicalEntitySrc$ treatment **lowered** oxygen consumption in ADP-stimulated mitochondria and **reduced** respiratory control index when using malate/glutamate as substrate. The activities of both complex I and aconitase were also **decreased**. All the above described changes were prevented by curcumin. It is concluded that curcumin is able to **reduce** in vivo @ChemicalEntitySrc$ maleate-induced @/ChemicalEntitySrc$ nephropathy and in vitro cell damage. The in vivo protection was **linked to** prevention of oxidative stress and preservation of mitochondrial oxygen use and activity of respiratory complex I, and the in vitro protection was **linked to** the prevention of ROS production.