Empowering Large Language Models for Textual Data Augmentation

Yichuan Li[1*], **Kaize Ding**[2*], **Jianling Wang**[3], **Kyumin Lee**[1]
[1]Worcester Polytechnic Institute, [2]Northwestern University [3]Google DeepMind
{yli29,kmlee}@wpi.edu, kaize.ding@northwestern.edu, jianlingw@google.com

Abstract

With the capabilities of understanding and executing natural language instructions, Large language models (LLMs) can potentially act as a powerful tool for textual data augmentation. However, the quality of augmented data depends heavily on the augmentation instructions provided, and the effectiveness can fluctuate across different downstream tasks. While manually crafting and selecting instructions can offer some improvement, this approach faces scalability and consistency issues in practice due to the diversity of downstream tasks. In this work, we address these limitations by proposing a new solution, which can automatically *generate* a large pool of augmentation instructions and *select* the most suitable task-informed instructions, thereby empowering LLMs to create high-quality augmented data for different downstream tasks. Empirically, the proposed approach consistently generates augmented data with better quality compared to non-LLM and LLM-based data augmentation methods, leading to the best performance on 26 few-shot learning tasks sourced from a wide range of application domains.

1 Introduction

Large language models (LLMs) have recently demonstrated their potential in performing data augmentation on text data (Dai et al., 2023; Chung et al., 2023; Yu et al., 2023; Yoo et al., 2021). Serving as a semantic-preserving transformation function, LLMs transform original texts based on instructions to create diverse and informative data augmentations. With the augmented data, users can further train a spreadable and affordable model (e.g. `OPT` (Zhang et al., 2022)) to perform specific tasks. Unlike traditional heuristic-based methods such as word swapping (Wei and Zou, 2019) and model-based methods like back-translation (Fadaee et al., 2017), LLMs offer great potential to produce more fluent, diverse, and semantically consistent augmentations for text data, owing to their great understanding and generalization capabilities.

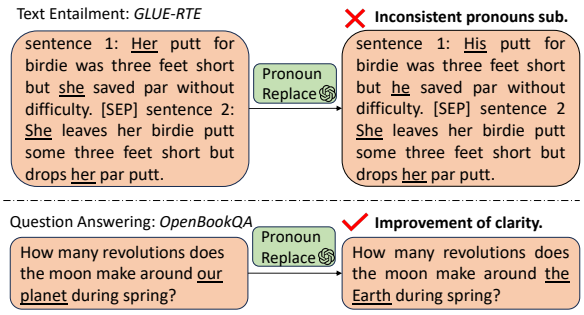


Figure 1: A simple demo of pronouns replacement augmentation instruction on text entailment task: *GLUE-MRPC* (Wang et al., 2019) and question answering task: *OpenBookQA* (Mihaylov et al., 2018).

Despite the early success of LLMs for textual data augmentation, existing methods (Dai et al., 2023) that simply prompt LLMs with human-crafted augmentation instructions (i.e., `Manual-LLMDA` methods) have the following major bottlenecks: (1) Firstly, their efficacy heavily relies on the quality of the augmentation instructions, which are manually engineered by domain experts. This manual process is not only domain knowledge-intensive but also prone to inconsistencies, potentially compromising the quality of augmented data. Subtle variations in how these instructions are formulated can significantly influence the outcomes, as demonstrated by recent studies (Ishibashi et al., 2023; Zhu et al., 2023); (2) Secondly, usually text augmentation instructions are written in a task-agnostic form for a general purpose, however, the lack of context information on downstream tasks could lead to dramatic performance disparity on different downstream tasks, as shown in Fig. 1. Without considering the specific properties of the target tasks, LLM may generate low-quality augmented data (Ribeiro et al., 2020; Wei and Zou, 2019).

[*]The first two authors contributed equally to this work. Kaize Ding is the corresponding author.

Findings of the Association for Computational Linguistics ACL 2024, pages 12734–12751
August 11-16, 2024

To address the aforementioned challenges, in this paper, we introduce a new framework – Self-LLMDA that automates augmentation instruction generation and selection, facilitating LLM to generate task-specific augmented data. The initial phase of Self-LLMDA aims to broaden the span of seed augmentation strategies through the generation of diverse and effective instructions based on LLMs. Following this, Self-LLMDA employs a scoring model to identify and select the most relevant instructions that are likely to bolster the performance of target models. Such a new textual data augmentation approach ensures a balance between the generative breadth of augmentation instructions and targeted precision of task-specific guidance for downstream tasks.

In our study, we conduct extensive experiments across a large collection of few-shot learning tasks used in previous studies (Min et al., 2022; Ye et al., 2021; Khashabi et al., 2020). This collection includes 26 different types of tasks across hate speech detection, question answering, natural language inference, and phrase detection datasets. Our study stands out for its extensive coverage of tasks, setting a new benchmark in the application of LLMs for textual data augmentation when compared to previous work (Dai et al., 2023; Li et al., 2023; Chung et al., 2023). The empirical results demonstrate that the proposed approach Self-LLMDA significantly outperforms various baseline methods in generating high-quality augmented textual data. To summarize, our main contributions are as follows:

- We introduce a framework Self-LLMDA, which automates the generation and selection of task-specific augmentation instructions for LLMs, providing effective data augmentation for text data.
- Through a comprehensive set of experiments, we validate the effectiveness of Self-LLMDA, demonstrating its superior performance in enhancing data quality and model accuracy over existing text data augmentation methods.
- Our in-depth analyses reveal that Self-LLMDA can well generalize across various target models and previously unseen augmentation instructions, demonstrating its versatility and potential for broad applicability.

2 Related Work

2.1 Non-LLM Textual Data Augmentation

Conventional textual data augmentation methods encompass a variety of techniques aimed at enhancing the diversity of textual datasets without relying on large language models (i.e., Non-LLMDA methods). Those methods range from simple heuristic-based methods to generative model-based methods. For heuristic-based approaches, such as synonym replacement (Zhang et al., 2016) and word shuffling, stand out for their computational efficiency and simplicity, making them ideal for large-scale data augmentation with minimal computational demands. Another notable example is the Easy Data Augmentation (EDA) technique introduced by Wei and Zou (2019), which employs token-level perturbations—random insertion, deletion, and swapping—to improve performance across a spectrum of text classification tasks.

For model-based approaches, researchers have employed seq2seq and language models for data augmentation. Back-translation (Fadaee et al., 2017) employs translation models to preserve semantic integrity while generating paraphrases (Fadaee et al., 2017). Conditional masked language models like BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) can also be utilized for data augmentation (Cheng et al., 2022; Wu et al., 2018). By masking words within sentences and subsequently generating replacements, these models introduce linguistic variations. Furthermore, other methods (Kumar et al., 2021; Edwards et al., 2023) leverage the capabilities of generative language models like GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2019) for data augmentation. These approaches perform conditional generation based on class labels. Additionally, some studies have explored augmentation in the feature space. Mixup techniques interpolate within word or sentence embeddings (Guo et al., 2019), while others introduce random multiplicative and additive noise to the feature vectors (Kurata et al., 2016). Despite their utility, these conventional Non-LLMDA methods often come with limitations in readability and contextual consistency.

2.2 LLM-based Textual Data Augmentation

Recent advancements in LLMs have demonstrated their superiority in generating high quality and contextually relevant augmented data (Brown et al., 2020). LLMs are increasingly employed as label-preserving transformation functions, where an original example is transformed or perturbed according to manually crafted instructions (Dai et al., 2023; Yoo et al., 2021; Piedboeuf and Langlais, 2023).

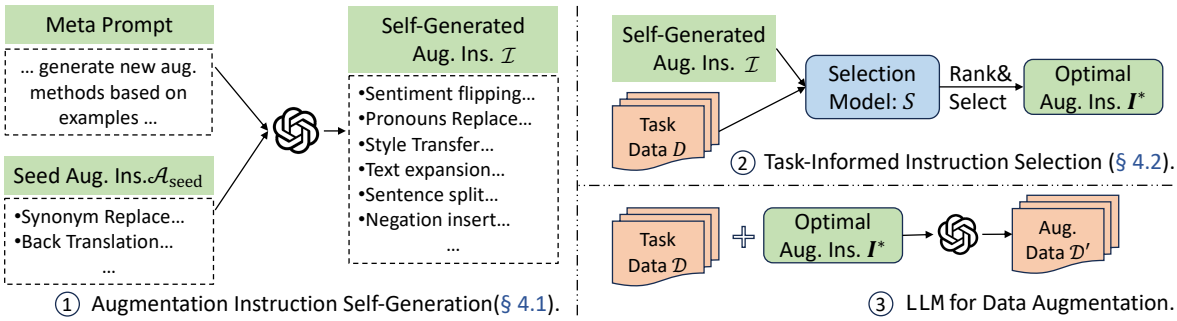


Figure 2: The pipeline of `Self-LLMDA`. We first prompt the LLM to generate a diverse set of candidate augmentation instructions (§ 4.1). Then we select the instruction (§ 4.2) and apply it with the task data to LLM to get augmentations.

Concurrently, several studies (Chung et al., 2023; Yu et al., 2023; Li et al., 2023; Ubani et al., 2023; Meng et al., 2022) have explored the generation of conceptually similar yet semantically distinct synthetic examples. These methods, however, mostly rely on manual instruction design. In contrast, our work automatically generates label-preserving augmentation instructions by prompting LLMs, thus reducing dependency on manually crafted instructions. Furthermore, we introduce an instruction selection model that chooses appropriate instructions for arbitrary downstream tasks.

3 Preliminary

Problem Definition. Textual data augmentation involves applying a label-preserving transformation function $T(\cdot)$ to a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{k}$, where each example consists of an input text $\mathbf{x}_i$ (a sequence of tokens) and a corresponding label $\mathbf{y}_i$ (also a sequence of tokens). The augmented dataset $\mathcal{D}'$ is generated as follows, ensuring that the output label $\mathbf{y}_i'$ remains unchanged:

$$\mathbf{x}_i' = T(\mathbf{x}_i), \mathbf{y}_i' = \mathbf{y}_i. \tag{1}$$

A target model F is then trained on the union of the original and augmented datasets, $\mathcal{D} \cup \mathcal{D}'$, with the training objective defined as:

$$\mathcal{L}_{(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_i) \in \mathcal{D} \cup \mathcal{D}'}(F_\theta(\hat{\mathbf{x}}_i), \hat{\mathbf{y}}_i). \tag{2}$$

Therefore, designing an effective transformation function $T(\cdot)$ that produces high-quality augmented data $\mathcal{D}'$ is crucial for improving the downstream performance of model F_θ.

`Manual-LLMDA`. For `Manual-LLMDA` methods, the transformation function $T(\cdot)$ is realized through a combination of an LLM and a manual-crafted instruction $\mathbf{I}_{\text{man}}$ (e.g., paraphrasing). The LLM is prompted to generate semantic-preserving transformations of the input text $\mathbf{x}_i$ for the augmented dataset $\mathcal{D}'$:

$$\mathbf{x}_i' = \text{LLM}(\mathbf{I}_{\text{man}}, \mathbf{x}_i), \mathbf{y}_i' = \mathbf{y}_i \tag{3}$$

4 Proposed Approach – `Self-LLMDA`

To reduce the human efforts in designing augmentation instructions and selecting a task-specific instruction for a given task, we propose `Self-LLMDA` depicted in Fig. 2. The process begins with the LLM generating a diverse set of potential instructions $\mathcal{I} = \{\mathbf{I}_j\}_{j=0}^{n}$ from a given set of seed instructions $\mathcal{I}_{\text{seed}} = \{\mathbf{I}_{\text{man}}\}$:

$$\mathcal{I} = \text{LLM}(\mathcal{I}_{\text{seed}}). \tag{4}$$

A selection model S then scores these generated instructions against the dataset $\mathcal{D}$ to identify the most suitable instruction $\mathbf{I}^*$:

$$\mathbf{I}^* = S(\mathcal{I}, \mathcal{D}). \tag{5}$$

Based on the selected instruction $\mathbf{I}^*$, the LLM performs data augmentation on $\mathcal{D}$, producing an enhanced augmented dataset $\mathcal{D}'$ for training the target model more effectively.

4.1 Augmentation Instruction Self-Generation

Inspired by the self-instruct methodology (Wang et al., 2022), this phase generates augmentation instructions from a seed set of 13 human-crafted instructions. These seed instructions act as exemplars, guiding the LLMs toward the creation of novel and diverse instructions that maintain the semantic integrity of the input text. To generate a broad and diverse set of augmentation instructions without the bias introduced by a few task examples, we exclude the task-specific data from the instruction generation. This will leverage the zero-shot learning capabilities of LLMs to produce a wide array of potential augmentation instructions. We use the following prompt to encourage LLMs to explore various augmentation techniques:

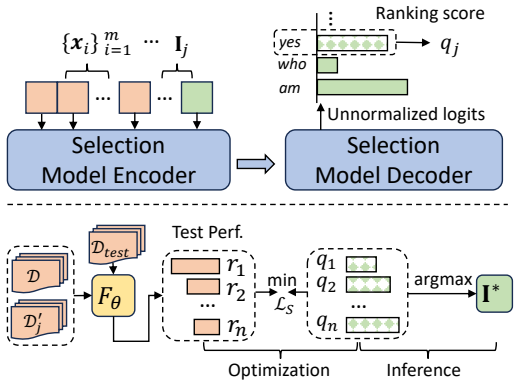


Figure 3: Illustration of the Instruction selection scoring model. F_θ is the target model.

> "*Come up with a series of textual data augmentation methods and you need to generate more **diverse** data augmentation method that can **keep the semantic meaning** of the input sentence.* $\{\mathbf{I}_{seed}\}$ "

Through iterative cycles of generation and refinement, we filter out instructions that are too similar to existing ones based on ROUGE-L (Lin, 2004). The unique generated instructions from each iteration are then incorporated back into the seed instruction pool, enriching the seed instructions for subsequent generation rounds. This process is repeated until we reach a collection of 100 augmentation instructions. To ensure diversity and eliminate redundancy, we further refine this set by removing duplicates based on their method names. This filtration results in a final set of 51 unique augmentation instructions.

4.2 Task-Informed Instruction Selection

Recognizing that augmentation instructions may not be universally applicable across different tasks, we implement a selection mechanism, tailored to the specific requirements of each task and its corresponding target model. This process involves a scoring model S to evaluate the suitability of each instruction for the task at hand. The scoring model S, as shown in Fig. 3, outputs a ranking score q_j indicating the instruction's effectiveness based on the pair of instruction and task dataset. Based on the notable instruction-following capabilities of FLAN-T5 (Chung et al., 2022; Raffel et al., 2023), we choose FLAN-T5-Large (Chung et al., 2022; Raffel et al., 2023) as the backbone of our scoring model. The input for scoring model S is:

> "*Given the dataset for **task** $\mathcal{T}$ and the instruction data, determine if this is a suitable instruction to address the task for **model** F. **Task Dataset:*** $\{\mathbf{x}_i\}_{i=0}^m$ ***Instruction:*** $\mathbf{I}_j$. *Is this instruction appropriate?*"

where $\mathcal{T}$ is the task name (e.g. GLUE-RTE), F is the target model name (e.g. OPT-125m). Since most of the tasks did not have a task description and manually designing the task description is time consuming, we utilize the few-shot examples $\{\mathbf{x}_i\}_{i=0}^m$ from the dataset as the task description. Here, we calculate q_j by assessing the logit value of the "yes" token of the last position of input from FLAN-T5-Large, as shown in Fig. 3. Next, we will introduce the optimization and inference procedure of the scoring model S respectively.

Model Optimization. The instruction selection model is trained to prioritize generated augmentation instructions based on their impact on downstream task performance. Its goal is to assign the highest scores to instructions that lead to the most effective data augmentation. To enhance scalability and computational efficiency, our model optimizes the selection process for a given task $\mathcal{D}$ by sampling a subset of augmentation instructions $\{\mathbf{I}_j\}_{j=0}^n$ (where $n > 1$) from the pool of candidates. The model then computes scores $\{q_j\}_{j=0}^n$, representing the relative effectiveness of each instruction. The optimization objective is formulated as a cross-entropy loss, designed to accurately distinguish between the effectiveness of these instructions $\{\mathbf{I}_j\}_{j=0}^n$. The loss function is given by:

$$\mathcal{L}_S = -\sum_{j=0}^{n} \text{is\_max}(r_j) \log \sigma(q_j) \tag{6}$$

Here, is_max serves as a binary indicator function that identifies the instruction yielding the maximum effectiveness, and σ is `softmax` that normalizes the q_j (a probability generating "yes" token interpreted as a ranking score associated with jth instruction) over the sampled augmentation instructions, r_j is the downstream task performance of a trained target model on augmented data $\mathcal{D}'_j \cup \mathcal{D}$.

Model Inference. When encountering a new task, the selection model S evaluates all potential instructions to determine the most suitable one $\mathbf{I}^*$, denoted by the highest score:

$$\mathbf{I}^* = \mathbf{I}_{\text{argmax}(\{q_j\}_{j=0}^{|\mathcal{I}|})} \tag{7}$$

This optimal instruction, $\mathbf{I}^*$, is then employed to prompt the LLMs to generate augmented data. This selection mechanism ensure the use of the most effective instruction for enhancing data utility across diverse NLP tasks.

5 Experiment

5.1 Experimental Setup

Evaluation Datasets. In this study, we select 26 few-shot learning tasks spanning a wide range of NLP challenges, sourced from CrossFit (Ye et al., 2021), UnifiedQA (Khashabi et al., 2020), and MetaICL (Min et al., 2022). These datasets were chosen for their diversity, encompassing both classification tasks (Class)—such as natural language inference, paraphrase detection, and hate speech identification—and non-classification (Non-Class) tasks, notably question answering, to ensure a broad evaluation spectrum. The selection of tasks is significantly larger and more diverse than that in other relevant works (Dai et al., 2023; Chung et al., 2023).

To investigate the generalization ability of `Self-LLMDA`, we split the 26 tasks into training and test tasks as for form "train→ test". We train the augmentation instruction selection methods on training tasks and evaluated it on test tasks. The task split involves four settings: Class → Class, Class → Non-Class, Non-Class→ Class, and Random → Random, where "Random" represents a mixture of randomly selected tasks [*]. This design allows us to investigate the performance of selection models when applied across similar and disparate task types, providing insights into their generalizability and effectiveness.

Evaluation Metrics. To handle all types of tasks simultaneously, we unify all downstream tasks, including classification and non-classification tasks, using a text-to-text approach (Raffel et al., 2023). For each task, we feed the input text to the target model F_θ and train it to generate the corresponding target text. We choose `OPT` (Zhang et al., 2022) from three different sizes (e.g. 125m, 350m and 1.3b[*]) as our target models F_θ. During training[*], F_θ takes the training example $\mathbf{x}_i$ as the input, and is optimised to generate $\mathbf{y}_i$ using the negative likelihood objective function:

$$\mathcal{L}_{F_\theta}(\mathbf{y}_i) = -\sum_{t=1}^{|\mathbf{y}_i|} \log P_{F_\theta}(y_i^t | \mathbf{x}_i, \mathbf{y}_i^{<t}) \quad (8)$$

During inference time, given the test input $\mathbf{x}_{test}$ as well as a set of candidates $\mathcal{C}$, which is either a set of labels (in classification tasks) or answer options (in non-classification tasks), the F_θ computes the conditional probability of each label $\mathbf{c} \in \mathcal{C}$, where $\mathbf{c}$ is a sequence of tokens. The label with the maximum conditional probability is returned as a prediction:

$$\text{argmax}_{\mathbf{c}\in\mathcal{C}} \left(\sum_{t=1}^{|\mathbf{c}|} \log P_{F_\theta}(c^t | \mathbf{x}_{\text{text}}, \mathbf{c}^{<t}) \right) \quad (9)$$

Specifically, we use *macro-F1* for classification tasks, and *accuracy* for non-classification tasks in our experiment. The overall performance is then quantified by computing the macro-average of these scores across all tasks, encapsulating both *accuracy* and *macro-F1* metrics. To ensure robustness and reduce sampling bias, each experiment under each splitting setting is replicated with five different random seeds. For each few-shot task, we adopt a uniform approach by randomly selecting $k = 16$ training examples. Following (Min et al., 2022), we did not make perfect label balance between k training examples.

Baseline Methods. In this study, we compare our novel augmentation pipeline, `Self-LLMDA`, with two different categories of data augmentation methods as baselines: `Non-LLMDA` and `Manual-LLMDA` For both `Manual-LLMDA` and `Self-LLMDA`, we employ GPT-3.5 Turbo as the backbone LLM. For detailed descriptions of these baseline methods, please see Appendix E. Specifically:

- **`Non-LLMDA` methods.** This category includes 13 traditional augmentation techniques: Character-Level: Operations such as random swaps, OCR Errors simulation, deletions, insertions, and substitutions. Word-Level: Transformations, including word swaps, deletions, spelling errors, and embedding-based insertions. Contextual-Level: Utilization of language models for word insertions (e.g., using `GPT2` (Brown et al., 2020)) and substitutions (e.g., with `BERT` (Devlin et al., 2018)), and back-translation (Fadaee et al., 2017).

[*]Details of training and testing tasks split is in Tab. 9.

[*]Due to GPU memory constraints, the training mini batch size for the 1.3B model is set to 2, while the batch sizes for the 125M and 350M models are set to 8. This difference in batch sizes may cause the 1.3B model to achieve worse performance compared to the 125M and 350M models.

[*]Detailed hyparameter setting is in Appendix A.

	TextulDA	Class → Class			Class → Non-Class			Non-Class → Class			Random → Random		
		125m	350m	1.3b	125m	350m	1.3b	125m	350m	1.3b	125m	350m	1.3b
	Original	44.80	41.94	42.82	38.49	42.04	42.42	46.49	44.13	44.52	42.73	42.92	44.10
Non-LLMDA	Char. Swap	44.94	42.22	42.28	39.46	40.56	42.76	47.08	44.69	44.06	43.89	42.45	42.90
	Char. OCR	43.72	43.70	43.66	39.31	41.02	43.73	45.91	46.02	45.11	42.95	43.02	43.83
	Char. Delete	43.98	43.33	42.35	38.99	40.50	43.08	46.01	45.04	44.02	42.53	42.77	43.52
	Char. Insert	45.03	43.14	41.19	39.22	40.61	42.77	46.69	44.61	42.44	43.29	42.86	42.37
	Char. Subs.	43.87	43.07	40.29	39.39	40.22	42.46	46.11	45.22	43.25	43.41	42.65	43.08
	Word Swap	43.83	44.56	42.75	39.15	40.84	43.51	46.25	46.21	44.93	43.24	43.54	44.30
	Word Delete	44.24	42.38	43.90	39.54	40.34	43.12	45.70	44.49	45.56	43.41	42.02	44.28
	Spell Error	44.82	42.66	44.38	38.68	41.03	42.85	46.34	44.85	44.97	42.71	43.08	43.71
	GPT2 Insert	43.37	43.36	44.13	39.79	40.92	43.69	45.07	45.93	46.11	42.65	43.07	44.44
	Word2vec Insert	46.47	41.80	41.86	39.67	40.80	42.66	48.36	45.13	43.78	44.18	42.95	42.86
	BERT Subs.	44.88	44.01	44.45	39.57	40.23	42.86	46.87	45.32	46.35	43.17	42.92	43.79
	Word2vec Insert	44.24	43.89	42.68	38.90	40.21	42.80	46.17	45.58	44.34	43.07	42.76	44.06
	Back Translation	44.19	44.00	46.23	39.78	40.89	42.96	45.97	45.31	47.39	43.24	43.60	43.95
	Average	44.42	43.23	43.08	39.34	40.62	43.01	46.34	45.26	44.79	43.21	42.89	43.62
	Best	46.47	44.56	46.23	39.79	41.03	43.73	48.36	46.21	47.39	44.18	43.47	44.30
Manual-LLMDA	Char. Perturb	44.72	43.56	43.30	39.39	40.61	45.54	47.02	45.32	46.25	43.68	43.27	44.06
	Word Insert/Delete	44.80	44.27	44.82	38.81	40.97	**44.49**	46.58	45.93	46.59	42.95	43.68	44.46
	Word Swap	45.45	43.88	45.90	39.01	40.66	43.28	47.04	45.20	46.80	43.63	42.63	45.79
	Word Replace	45.29	45.94	46.53	39.14	41.05	43.10	47.36	47.98	47.62	43.52	44.57	45.89
	Grammar Transform	45.39	45.62	44.98	39.29	41.34	43.00	46.81	46.76	46.41	43.24	43.75	45.53
	Data Mix	44.63	44.01	43.73	38.50	41.29	43.67	46.02	45.68	43.93	42.76	43.26	43.01
	Paragraph Shuffle	45.78	42.03	45.80	39.29	40.70	43.28	47.70	45.02	48.02	44.43	43.25	45.41
	Mask Predict	42.83	43.15	45.87	39.58	40.59	44.05	46.02	44.60	45.70	42.59	43.16	44.64
	Sentence Reorder	44.18	43.84	**48.98**	39.37	40.76	43.20	46.30	46.35	48.46	43.35	43.70	45.64
	Contextual Replace	43.52	46.26	43.26	38.73	40.99	43.49	45.38	47.62	43.92	42.58	45.26	44.76
	Paraphrase	45.56	45.30	42.53	39.44	40.94	41.09	48.02	47.31	43.99	43.97	44.10	42.80
	POS Augment	44.97	44.48	47.51	39.12	41.14	43.57	47.06	46.07	47.25	43.49	43.21	45.75
	Back Translation	45.79	43.38	45.30	39.18	41.12	43.80	47.02	45.58	45.10	43.36	43.20	44.41
	Average	44.83	44.28	45.27	39.14	40.93	43.50	46.79	46.10	46.15	43.35	43.61	44.78
	Best	45.79	46.26	48.98	39.58	41.34	44.49	48.02	47.98	48.02	44.43	45.26	45.89
	Self-LLMDA	**51.72**	**54.98**	48.80	**40.02**	**42.80**	43.80	**50.00**	**52.75**	**49.48**	**46.64**	**48.83**	**48.95**

Table 1: The performance of different data augmentation methods. Char. and Subs. are the abbreviations of character and substitute, respectively. Underlined indicates best performance under each augmentation method group while **Bold** indicates the best result of the whole table. In each group, the last two rows represent the aggergated result of the whole group of augmentation methods (e.g. average and best).

- **Manual-LLMDA methods.** This set comprises 13 manually designed augmentation instructions for LLM, including: Character-Level: Perturbations similar to those in Non-LLMDA. Word-Level: Swaps, replacements, and part-of-speech (POS) enhancements. Sentence-Level: Reordering and data mixing strategies. Contextual-Level: Predictive masking, contextual substitutions, and back-translation.

We also report the average and best performance of Non-LLMDA and Manual-LLMDA for better comparison. An extensive ablation study of our task-informed selection model, presented in § 5.3.

5.2 Main Results

The analysis of experimental results presented in Tab. 1 reveals several findings: **Firstly**, there is performance inconsistency among the different instructions from Manual-LLMDA. The impact of augmentation instructions varies across different downstream tasks and models. This highlights the difficulty in creating universally effective data augmentation instruction. **Secondly**, Manual-LLMDA is not always better than Non-LLMDA. In controlled comparisons focusing on specific augmentation topics, Manual-LLMDA's advantages over Non-LLMDA were not clearly evident. For example, in the contexts of "Back Translation" and "Word Swap", Non-LLMDA outperformed Manual-LLMDA in 5 out of 12 and 7 out of 12 cases, respectively. **Lastly**, the experimental results show the superiority of Self-LLMDA. Our proposed model consistently outperformed these baseline methods, highlighting the effectiveness of integrating automatic instruction generation with targeted task-specific instruction selection. This approach not only optimizes performance but also reduces the manual efforts typically required to design effective augmentation strategies, showcasing the potential of our model in enhancing data augmentation practices.

5.3 Ablation Study

We add an ablation study to understand the impact of two key components in our framework: aug-

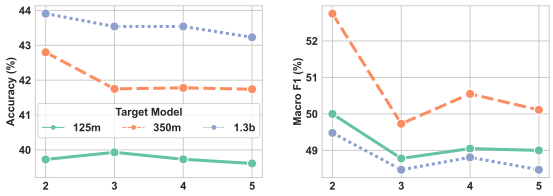
Select. Method	Class→Non-Class			Non-Class→Class		
	125m	350m	1.3b	125m	350m	1.3b
`Manual-LLMDA`$^{+}$	39.62	41.74	**44.38**	48.02	48.51	48.07
`Random-Select`	39.34	40.31	42.68	46.15	44.34	43.98
`Empirical-Select`	39.17	41.18	43.14	47.19	47.30	44.41
`LLM-Select`	38.77	41.06	43.07	46.81	48.02	46.42
`Self-LLMDA`	**40.02**	**42.80**	43.80	**50.00**	**52.75**	**49.48**

Table 2: Ablation study of `Self-LLMDA`.

mentation instruction self-generation and the task-informed instruction selection. **Firstly**, we train a task-informed instruction selection model S on manually-crafted instructions from `Manual-LLMDA` and named it `Manual-LLMDA`$^{+}$ to understand the contribution of the contribution of LLM self-generated augmentation instructions. **Secondly**, we test the efficacy of our selection model by comparing three alternative selection strategies: (1) Random-select, which randomly select instruction from the pool of augmentation methods for each task; (2) Empirical-select, which selects the prompt that yielded the highest average performance across training tasks, under the assumption that successful prompts on training tasks will generalize well to test tasks; and (3) LLM-Select, which prompts the LLM to chooses the most suitable instruction from candidates based on its internal decision-making processes. The Results in Tab. 2 show that `Self-LLMDA` consistently outperforms these alternative methods, indicating the benefits of instruction self-generation and task-informed selection in enhancing model performance.

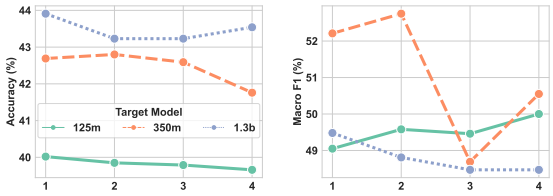
5.4 Hyperparameter Analysis

Here, we closely examined the impact of two critical hyperparameters on the training of our task-informed instruction model: n and m. The hyperparameter n specifies the number of augmentation instructions to be sampled for optimizing Eq. 6. It should be noticed that, we only vary n at training time, while at inference, we will calculate the score for all the generated instructions and choose the one with the largest score. On the other hand, m determines the number of examples from the task dataset that are used to represent the task, influencing the model's performance during both the optimization and inference phases. Our analysis, depicted in Fig. 5, highlights several key findings: *(1) Optimal Number of Instructions:* We found that setting $n = 2$ leads to the best performance, outperforming other configurations. This suggests that a pairwise comparison, as formulated in Eq. 6, is most effective for our model's learning process. *(2)*



(a) Class→Non-Class (b) Non-Class→Class

Figure 4: Hyperparameter analysis of n, which dictates the number of augmentation instructions sampled during the training of the selection model.



(a) Class→Non-Class (b) Non-Class→Class

Figure 5: Analysis of the hyperparameter m, which determines the number of examples randomly sampled to represent a task.

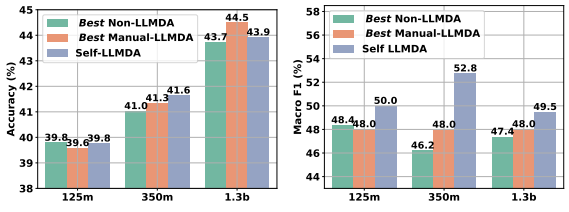
Representative Examples: Interestingly, a smaller number of examples (m) appear to better capture the essence of the tasks. This observation indicates that a larger set of examples could introduce noise, potentially detracting from the model's ability to accurately represent tasks for instruction selection.

5.5 In-Depth Analysis of the Task-Informed Instruction Selection Model

In this section, we provide a detailed analysis of the performance and generalization capabilities of our instruction selection model S, focusing on its generalizability to unknown augmentation instructions, unknown target models, and the specific case studies of the augmentation instructions it selects.

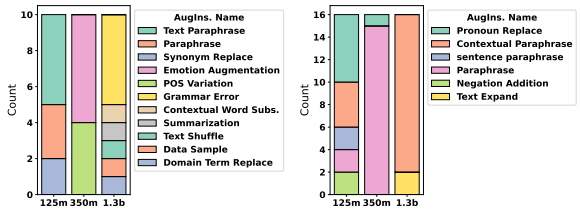
Generalization to Unknown Augmentation Instructions. In this analysis, we delve into the selection model's adaptability to unknown augmentation instructions by simulating a dynamic environment where new instructions are generated asynchronously by the LLMs. This scenario mirrors practical applications where the augmentation instruction set can expand without necessitating retraining of the selection model. To test this, we constrained the training phase of the selection model to a limited subset of self-generated augmentation instructions (30% of all generated by the LLMs), utilizing the whole generated instructions for evaluation at inference time.

As the results shown in Fig. 6, we can ob-



(a) Class → Non-Class. (b) Non-Class → Class.

Figure 6: Result of generalization to unknown augmentation instruction selection.



(a) Class→Non-Class. (b) Non-Class→Class.

Figure 7: Selected augmentation instructions from task-informed augmentation selection model.

serve a performance improvement of our selection model over the best performance of `Non-LLMDA` and `Manual-LLMDA`. This indicates the robustness of our selection model in adapting to incremental augmentation instructions, effectively selecting suitable instructions even when faced with previously unknown instructions. These observations highlight the efficacy of our selection model in a dynamic augmentation scenario.

Generalization to Unknown Target Models. Our study extended to evaluate the adaptability of our task-informed selection model across diverse target models. By applying the selection model, initially trained on the task performance of a specific target model, to different models. The results of these experiments are presented in Tab. 3. Our findings show that the augmentation instructions selected by our model remain effective even when applied to different target models. Notably, in most scenarios, our model `Self-LLMDA`, when transferred to alternate target models, outperformed the best results obtained using `Non-LLMDA` and `Manual-LLMDA`. This indicates that the underlying pattern determining instruction effectiveness via our instruction selection model is transferable.

Train.	Class → Non-Class			Non-Class → Class		
	125m	350m	1.3b	125m	350m	1.3b
Best `Non-LLMDA`	39.79	41.03	43.73	48.36	46.21	47.39
Best `Manual-LLMDA`	39.58	41.34	**44.49**	48.02	47.98	48.02
125m	**40.02**	41.97	43.56	**50.00**	**54.12**	**49.83**
350m	39.96	**42.80**	43.66	49.96	52.75	48.82
1.3b	39.85	42.42	43.80	49.04	51.22	49.48

Table 3: Transferability of the Task-Informed Selection Model. Our selection model, initially trained on a specific target model (indicated by each row in the second group), when applied to alternate target models (represented in each column).

Analysis of Selected Instructions. We conducted a detailed analysis of the augmentation instructions chosen by our selection model, and the findings visualized in Fig. 7. The key insights from this analysis are as follows: *(1) Diversity of Selected Instructions:* The distribution of selected instructions showcases a wide variety in the types of augmentations chosen by the model, with 3, 2, and 6 unique data augmentation instructions identified for the 125m, 350m, and 1.3b models under Class→Non-Class, respectively. This demonstrates the model's ability to adapt and select from a broad spectrum of augmentation strategies to meet the specific requirements of different tasks. *(2) Variability across Models:* The selection patterns exhibit notable differences when the model is applied to various target models. This variability indicates preference differences across different target models. *(3) Preference for Paraphrase-Based Instructions:* A significant portion of the selected instructions fall into the category of paraphrase-based augmentations, such as "Text Paraphrase", "Paraphrase", "Contextual Paraphrase", and "Sentence Paraphrase". This preference not only highlights the effectiveness and general applicability of paraphrase-based augmentations but also illustrates our task-informed selection model's nuanced capability to discern and recommend the most suitable paraphrase variation for a given task.

6 Conclusion

In this work, we introduced `Self-LLMDA`, a novel framework that leverages the capabilities of LLMs for textual data augmentation. Our approach addresses the challenges associated with traditional data augmentation methods and the limitations of manual instruction generation in LLM-based augmentation. `Self-LLMDA` automates the generation and selection of augmentation instructions, thereby significantly enhancing the quality and applicability of augmented data across diverse downstream tasks. Tested across 26 diverse few-shot learning tasks, `Self-LLMDA` consistently outperforms both `Non-LLMDA` and `Manual-LLMDA` methods, showcasing its effectiveness and applicability.

7 Limitations

This study acknowledges several constraints that delineate the scope of our current work and outline directions for future research:

- **Evaluation on a Limited Range of LLMs:** Our experiments were conducted primarily with GPT 3.5 Turbo due to the high costs associated with using OpenAI models. While promising results in Tab. 1 suggest that our proposed `Self-LLMDA` method could potentially perform even better on more advanced models like GPT 4 Turbo, comprehensive testing was not feasible. Similarly, the computational demands of evaluating open-source LLMs such as LLAMA-70b-chat (Touvron et al., 2023), coupled with the extensive number of tasks in our study, exceeded our resources. Despite these limitations, we are optimistic that `Self-LLMDA` would exhibit enhanced performance across a broader spectrum of LLMs.

- **Meta-Prompting Exploration:** Within the `Self-LLMDA` framework, we employed one meta-prompt to guide the LLM in generating diverse and relevant augmentation instructions. However, our exploration of meta-prompting techniques was limited. We acknowledge that more sophisticated prompt engineering could further refine the quality and effectiveness of generated instructions. Investigating more advanced meta-prompting strategies remains an area for future exploration.

- **Analysis of Ensemble Augmentation Methods:** Our research did not investigate the potential benefits of combining multiple sets of augmented data (e.g., $\mathcal{D} \cup \mathcal{D}'_1 \cup \mathcal{D}'_2$). Such ensemble approaches introduce additional complexities, such as determining the optimal number of augmentation instructions to include. While we hypothesize that ensemble augmentation could improve model performance, this aspect falls outside the current study's scope and is earmarked for subsequent investigation.

References

Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2022. A survey on data augmentation for text classification. *ACM Computing Surveys*, 55(7):1–39.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation.

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.

Qiao Cheng, Jin Huang, and Yitao Duan. 2022. Semantically consistent data augmentation for neural machine translation via conditional masked language model.

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.

John Joon Young Chung, Ece Kamar, and Saleema Amershi. 2023. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. *arXiv preprint arXiv:2306.04140*.

Claude Coulombe. 2018. Text data augmentation made simple by leveraging nlp cloud apis.

Haixing Dai, Zheng Liu, Wenxiong Liao, Xiaoke Huang, Zihao Wu, Lin Zhao, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Chataug: Leveraging chatgpt for text data augmentation. *ArXiv*, abs/2302.13007.

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

Aleksandra Edwards, Asahi Ushio, Jose Camacho-Collados, Hélène de Ribaupierre, and Alun Preece. 2023. Guiding generative language models for data augmentation in few-shot text classification.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. *arXiv preprint arXiv:1705.00440*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study.

Yoichi Ishibashi, Danushka Bollegala, Katsuhito Sudoh, and Satoshi Nakamura. 2023. Evaluating the robustness of discrete prompts. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2373–2384, Dubrovnik, Croatia. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. AEDA: An easier data augmentation technique for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2748–2754, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2021. Data augmentation using pre-trained transformer models.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Labeled Data Generation with Encoder-Decoder LSTM for Semantic Slot Filling. In *Proc. Interspeech 2016*, pages 725–729.

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

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023. Synthetic data generation with large language models for text classification: Potential and limitations. *arXiv preprint arXiv:2310.07849*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Frédéric Piedboeuf and Philippe Langlais. 2023. Is ChatGPT the ultimate data augmentation algorithm? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15606–15615, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

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.

Solomon Ubani, Suleyman Olcay Polat, and Rodney Nielsen. 2023. Zeroshotdataaug: Generating and augmenting training data with chatgpt.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2018. Conditional bert contextual augmentation.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. pages 7163–7189.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. Gpt3mix: Leveraging large-scale language models for text augmentation.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts.

A Detailed Experiment Settings

Generation Configuration. We utilize gpt-3.5-turbo as our backbone LLM for augmentation instruction generation and data augmentation. We set the temperature for both of them as 0.7. For the instruction generation, we follow the generation hyper-parameter setting from Wang et al. (2022). For data augmentation, we utilize the default generation hyper-parameter from Chat Completion. The whole experiment including generating augmentation instructions and generating augmentation data costs us $82 USD in total, according to OpenAI's pricing (Input $0.0005 / 1K tokens and output $0.0015 / 1K tokens). However, the total experiment cost around $200 USD for debugging and exploration.

Meta Prompts for Data Augmentation As shown in step (3) in Fig. 2, we also need a meta prompt to encourage `Self-LLMDA` to augment high quality data. The main reason for this meta-prompt setting is because in some augmentation instructions they will discuss some external tools like word-embedding, other language models, if we did not provide the meta-prompt, the LLM will reject the generation of augmented data. The design of meta prompt is as follows:

> *Please do the following data augmentation steps to the text delimited by triple backticks. If you need any external resources or data, you can just simulate the environment by yourself and finish that step based on your own knowledge since you are the best language model in word.* ***Augmentation Instructions:*** $\mathbf{I}_j$, ***Input Data:*** $\mathbf{x}_i$ $\{\mathbf{I}_{seed}\}$

Task-informed Instruction Selection. The instruction ranking model is initialized with FLAN-T5-Large(Radford et al., 2019) and is trained using Adafactor (Shazeer and Stern, 2018) with learning rate 5e-5 and dropout 0.1. We train the selection model for 100 epochs and set the early stop with patience 20 epochs. We employ the validation set from training tasks to select the best checkpoint. The search space for different hyperparameter analysis are as follows:

Target Model Finetuning. We use `OPT` (Zhang et al., 2022) from 125m, 350m, 1.3b different sizes. For all of them we use `AdamW`(Loshchilov and Hutter, 2019) as our optimizer with learning rate 5e-5 with 10 training epochs. Due to the constraint of GPU memory, for 125m and 350m we set the batch size as 8, while 1.3b we set the batch size as 2. All these experiments is tested on one NVIDIA A100 A100-40G GPU cards.

Symbol	Description	Search Space
n	Number of sampled augmentation instruction	{2, 3, 4, 5}
m	Number of sampled examples from task dataset	{1, 2, 3, 4}
-	batch size	{4, 8, 16}
-	epochs	100

Table 4: The search space of augmentation selection.

B Analysis of Self-Generated Instructions

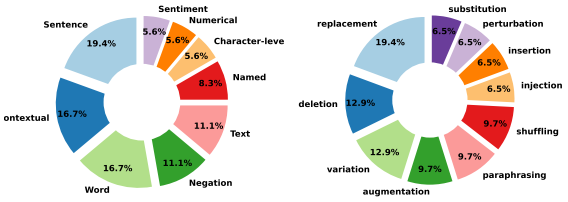
In our analysis, we delve into the characteristics and diversity of the self-generated augmentation prompts created by `Manual-LLMDA`.

Statistical Information. To facilitate a structured examination, we categorize these prompts based on the textual data augmentation taxonomy outlined by Bayer et al. (2022). The distribution and basic statistics of these various augmentation methods are detailed in Tab. 5.

Augmentation Type	Count	AVG Length
`Manual-LLMDA`	13	28.15
- character level	1	32.00
- word level	3	20.67
- phrase level	4	31.75
- document level	5	29.00
`Manual-LLMDA`	51	25.58
- misc.	1	22.00
- character level	2	24.50
- word level	10	22.80
- phrase level	18	26.72
- document level	20	26.25

Table 5: Statistics of augmentation prompts.

Naming Conventions. A notable aspect of our analysis involves examining the naming conventions of the augmentation methods. Recognizing that the method names often provide a high-level summary of the augmentation approach (e.g., <method name>), we further explore the linguistic patterns within these names. Specifically, we conduct an analysis focusing on the first and last words of each method name. This approach allows us to gain insights into the thematic and functional aspects of the augmentation methods. The distribution of these first and last words in method names is visually represented in Fig. 8. This visual representation aids in understanding the range and focus of the augmentation techniques generated by `Manual-LLMDA`.



(a) First word. (b) Last word.

Figure 8: The first words and last words from the Chat-Self. We filter these words by appearing more than once.

By analyzing these key linguistic elements, we aim to shed light on the creative breadth and thematic focus of the self-generated augmentation instructions.

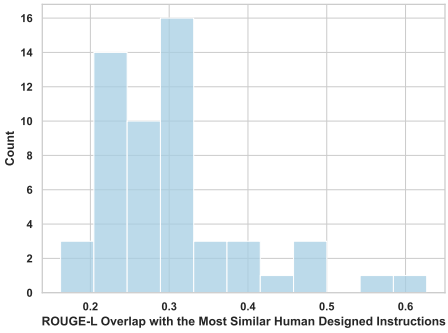


Figure 9: Distribution of the ROUGE-L scores between generated instructions and their most similar human-designed instructions.

C Analysis of Generated Data Across Different Augmentation Methods

In this analysis, we aim to discern the differences among the original dataset, non-LLM augmented data, data augmented via human-designed instructions, and data augmented using `Manual-LLMDA` generated instructions. Our focus is on the surface-level characteristics of the augmented content, and we consolidate data across all tasks for a comprehensive view. Key observations from the analysis, as detailed in Tab. 6, include the following:

Length of Content. Data augmented by LLM-based methods, on average, exhibits longer content compared to both the original and traditionally augmented datasets. This increase in length could offer a broader spectrum of training examples, potentially aiding in better generalization of target models. However, it also introduces a challenge of dataset inconsistency and the risk of adding unwanted variations.

Perplexity Scores. Interestingly, LLM-augmented content achieves lower perplexity scores (as measured on GPT2-small) compared to traditional augmentation methods. This suggests that the target model like GPT2-small has a better grasp of content augmented by LLMs. A possible explanation for the higher perplexity scores observed in non-LLM text augmentations is that the character and word-level changes might introduce new, irrelevant tokens into the text, thereby increasing complexity.

Closeness to Original Examples. Compared to non-LLM augmentation methods, LLM-based augmentations tend to produce content that is more closely related or less diverse relative to the original examples. This observation points to a potential trade-off between relevance and diversity in the augmented content generated by LLMs.

Metrics	Method	Mean	Std.	25%	50%	75%
Sentiment	Ori.	0.05	0.24	0.00	0.00	0.14
	Traditional	0.05	0.23	0.00	0.00	0.12
	Human	0.06	0.23	0.00	0.00	0.15
	ChatGPT	0.06	0.23	0.00	0.00	0.16
Grammar Error	Ori.	1.50	2.34	0.00	1.00	2.00
	Traditional	4.92	5.02	2.00	4.00	7.00
	Human	7.78	47.76	0.00	1.00	4.00
	ChatGPT	4.19	27.51	0.00	1.00	3.00
Words	Ori.	29.06	38.84	11.00	20.00	34.00
	`Non-LLMDA`	30.01	39.01	11.00	21.00	36.00
	`Manual-LLMDA`	95.34	285.95	14.00	30.00	71.00
	`Self-LLMDA`	66.04	182.91	14.00	27.00	57.00
Perplexity On GPT2-small	Ori.	530.13	5867.93	46.42	93.78	253.85
	`Non-LLMDA`	1354.91	5912.87	231.22	595.03	1305.86
	`Manual-LLMDA`	614.80	12284.76	17.72	58.94	184.42
	`Self-LLMDA`	491.83	9715.57	21.28	59.85	173.28
Distance to Original	`Non-LLMDA`	0.26	0.15	0.12	0.25	0.39
	`Manual-LLMDA`	0.17	0.04	0.15	0.17	0.18
	`Self-LLMDA`	0.15	0.05	0.12	0.15	0.18

Table 6: Characteristics of augmented data.

Aug. Type	small	medium	large
Trad.	42.37/44.18	46.92/49.72	44.81/47.33
Human.	**42.43**/44.52	47.25/49.52	**48.44/53.32**
LLM.	42.03/**48.47**	**47.80/51.09**	45.88/52.17

Table 7: Main results, using target models from `GPT2` family. Two numbers indicate the single best augmentation method across tasks and the task specific best augmentation method. Bold indicates the best average result except results.

Augmentation Instruction Pitfalls Across Tasks The effectiveness of augmentation instructions can vary depending on the specific characteristics of the tasks at hand (Ribeiro et al., 2020; Wei and Zou, 2019). To illustrate this, we present a case study focusing on the augmentation instruction *Pronoun replacement: replace pronouns in the text with their corresponding nouns or vice versa, maintaining the*

semantic meaning of the sentence. For the sake of brevity, we will use the abbreviation PR to refer to *pronoun replacement.* We consider two categories of tasks: text entailment (TE) and question answering (QA). As shown in Tab. 8, the results indicate that PR yields suboptimal performance on TE tasks, while it achieves good performance on QA tasks. This discrepancy can be attributed to the inherent characteristics of these tasks. TE tasks heavily rely on capturing the overall semantic meaning and logical relationships within the text, which may not always be preserved when applying pronoun replacement (Gao et al., 2022). In contrast, QA tasks aim to locate and provide specific information relevant to the given question (Joshi et al., 2020). By replacing pronouns with their corresponding nouns, the model can more easily identify the relevant entities and establish a clearer connection between the question and the answer, ultimately benefiting the QA task performance.

Type	Task	Rank (/51)
TE	glue-mrpc	48
	glue-rte	44
	glue-wnli	33
	medical_questions_pairs	48
QA	qasc	1
	openbookqa	27
	commonsense_qa	4
	quartz-no_knowledge	34
	quartz-with_knowledge	9

Table 8: Performance comparison of the pronoun replacement (PR) augmentation instruction on text entailment (TE) and question answering (QA) tasks.

D Dataset Collection

In Tab. 9, we list all 26 tasks and how we splitting them into training and testing for evaluating the model generalization to unknown downstream tasks. Each task will have 16 training and validation examples but with full test examples. We utilize the code from `CrossFit`(Ye et al., 2021) to extract and split the training, validation and testing for each task.

E Details of Baseline Methods

E.1 Augmentation Methods of `Non-LLMDA`

All of the implementation of `Non-LLMDA` are from Ma (2019). Here is an elaboration on each of the mentioned `Non-LLMDA` augmentation methods:

Character-Level Augmentations Random Swap (Belinkov and Bisk, 2018): This involves swapping adjacent characters within words to simulate typos that might occur during typing. For example, "example" might become "exmaple". OCR Replace: Simulating errors commonly introduced by Optical Character Recognition (OCR) software when digitizing text. Characters that look similar, like 'o' and '0' or 'l' and '1', might be substituted for one another. Delete: Randomly removing characters from words to mimic typographical errors or omissions. Insert: Adding extra characters into words at random positions, simulating common typos or spelling errors. Substitute: Replacing characters in words with other characters, not necessarily similar in appearance, to create variations in the text.

Word-Level Augmentations. Swap (Wei and Zou, 2019): Changing the positions of two adjacent words in a sentence to add syntactic variability while largely preserving the sentence's meaning. Delete: Removing words from sentences randomly to simulate information loss and encourage the model to learn from incomplete data. Spell Error (Coulombe, 2018): Introducing common spelling mistakes into words to mimic human error and increase the model's exposure to varied spellings. Word2Vector Insert (Morris et al., 2020): Identifying suitable locations in a sentence to insert synonyms or related words based on word embeddings (like word2vec representations), enhancing semantic diversity.

Contextual-Level Augmentations Insert Word using `GPT2` (Kumar et al., 2021): Leveraging a pretrained model like `GPT2` to generate contextually relevant words to insert into sentences, increasing the complexity and variability of the sentence structures. Substitute Word using `BERT` (Kumar et al., 2021): Using a model like `BERT` to identify and replace words with contextually appropriate synonyms or related terms, maintaining the sentence's overall meaning while altering its surface form. Back-Translation (Fadaee et al., 2017): Translating a sentence into another language and then back into the original language. This process often introduces syntactic and lexical variations, providing a paraphrased version of the original sentence that retains its semantic content.

Category	Train/Test	Task Names			
Class → Class	Train	financial_phrasebank	ethos-religion	glue-wnli	glue-mrpc
	Test	tweet_eval-stance_feminist ethos-race superglue-cb	climate_fever tweet_eval-stance_atheism ethos-national_origin	poem_sentiment sick medical_questions_pairs	tweet_eval-hate glue-rte hate_speech18
Class→Non-Class	Train	tweet_eval-stance_atheism tweet_eval-stance_feminist glue-rte sick	superglue-cb climate_fever ethos-national_origin poem_sentiment	financial_phrasebank glue-mrpc glue-wnli hate_speech18	ethos-religion tweet_eval-hate medical_questions_pairs ethos-race
	Test	quarel superglue-copa quartz-with_knowledge	codah qasc commonsense_qa	ai2_arc quartz-no_knowledge	openbookqa dream
Non-Class → Class	Train	quarel superglue-copa quartz-with_knowledge	codah qasc commonsense_qa	ai2_arc quartz-no_knowledge	openbookqa dream
	Test	tweet_eval-stance_atheism tweet_eval-stance_feminist glue-rte sick	superglue-cb climate_fever ethos-national_origin poem_sentiment	financial_phrasebank glue-mrpc glue-wnli hate_speech18	ethos-religion tweet_eval-hate medical_questions_pairs ethos-race
Random → Random	Train	quartz-with_knowledge financial_phrasebank	tweet_eval-stance_feminist ai2_arc	codah superglue-cb	ethos-race
	Test	quartz-no_knowledge glue-rte glue-mrpc climate_fever tweet_eval-stance_atheism	hate_speech18 commonsense_qa poem_sentiment tweet_eval-hate openbookqa	medical_questions_pairs superglue-copa quarel qasc sick	ethos-religion ethos-national_origin dream glue-wnli

Table 9: All tasks used in this paper. We split them into training and testing sets under different experiment setting.

F Additional Experiment

We also compare our method with other data augmentation techniques from `Non-LLMDA` and `Manual-LLMDA`. The `Non-LLMDA` includes EDA (Wei and Zou, 2019) and AEDA (Karimi et al., 2021), while `Manual-LLMDA` includes GPT3Mix (Yoo et al., 2021) and ZeroGen (Ye et al., 2022). As shown in Tab. 10, our proposed method `Self-LLMDA` significantly outperforms these baseline methods in the Class→Class and Random→Random settings. However, in the Non-Class→Class setting, `Self-LLMDA` falls behind GPT3Mix. This may indicate suboptimal transferability of `Self-LLMDA` in this specific scenario. It is worth noting that GPT3Mix is designed specifically for classification tasks, whereas `Self-LLMDA` can be applied to a wide range of text-related tasks, demonstrating its versatility and broader applicability.

F.1 Augmentation Instructions of `Manual-LLMDA`

In Tab. 11, we will represent the manually crafted augmentation instructions. The format of these augmentation instructions is "<method name>: <method instruction>".

G Self Instructions Generation

The instructions automatically generated by LLM is shown in Tab. 12 and Tab. 13.

TextualDA	Class→ Class		Class→ Non-Class		Non-Class→ Class		Random→Random	
	125m	350m	125m	350m	125m	350m	125m	350m
EDA	45.28	44.55	39.34	41.27	47.60	46.97	43.89	44.16
AEDA	43.85	42.92	39.56	41.53	42.61	43.50	42.61	43.50
ZeroGen-LLM	45.66	45.81	**40.43**	40.62	47.69	48.58	44.14	44.74
GPT3Mix	51.62	51.36	-	-	**55.81**	**54.04**	-	-
`Self-LLMDA`	**51.72**	**54.98**	40.02	**42.80**	50.00	52.75	**46.64**	**48.83**

Table 10: Performance comparison with other non-LLM-based and LLM-based textual data augmentations.

Name	Augmentation Instruction
Word Replace	**Synonym Replacement:** Replace certain words in the text with their synonyms while keeping the sentence structure intact. This can be done using pre-built synonym databases or word embeddings.
Back Translation	**Back Translation:** Translate the text into another language using machine translation and then translate it back to the original language. This process introduces variations in the sentence structure and wording.
Paraphrase	**Paraphrase:** Render the same text in different words without losing the meaning of the text itself. More often than not, a paraphrased text can convey its meaning better than the original words.
Word Insert/Delete	**Random Insertion/Deletion:** Randomly insert or delete words in the text to create new variations of the original sentences.
Word Swap	**Random Swapping:** Randomly swap the positions of words in the text to create new sentence arrangements.
Mask Predict	**Masking/Prediction:** Mask certain words in the text and train the model to predict those masked words. This is similar to the concept of masked language modeling used in models like BERT.
Char. Perturb	**Character-level Perturbation:** Instead of operating at the word level, perform data augmentation at the character level. This can involve randomly replacing, inserting, or deleting characters within the text, leading to novel variations.
Sentence Reorder	**Sentence Reordering:** Randomly reorder the sentences in the text while maintaining the coherence of the overall passage. This can help the model become more robust in understanding different sentence arrangements.
Contextual Replace	**Contextual Synonym Replacement:** Instead of blindly replacing words with synonyms, consider the context of the sentence to choose appropriate synonyms. This can be achieved by using contextual word embeddings or language models like ELMo or BERT.
POS Augment	**Part-of-Speech Augmentation:** Identify the part-of-speech tags of words in the text and replace words with synonyms that have the same part-of-speech. This ensures that the grammatical structure of the sentence remains intact.
Grammar Transform	**Grammar Transformation:** Apply various grammar rules to the text, such as changing active voice to passive voice, transforming affirmative sentences to negative, or converting declarative sentences into questions.
Data Mix	**Data Mixing:** Combine two or more texts from different sources to create a new mixed-text data point. This can introduce diversity in the content and writing style.
Paragraph Shuffle	**Paragraph Shuffling:** Shuffle the order of paragraphs in longer texts to create new document structures. This can be particularly useful for tasks that involve document-level understanding.

Table 11: Manually crafted augmentation instructions.

Augmentation Instruction
Contextual word replacement: replace certain words in the text with contextually similar words. this can be done by using word embeddings to find words that have similar meanings or are used in similar contexts.
Sentiment flipping: change the sentiment of the text by flipping positive sentiments to negative and vice versa. this can help generate diverse data for sentiment analysis tasks.
Style transfer: transform the writing style of the text while maintaining its original content. this can involve converting formal language to informal, changing the tone, or adapting the text to a specific genre.
Contextual paraphrasing: paraphrase sentences in the text while considering the surrounding context. this ensures that the paraphrased sentences maintain the same meaning within the given context.
Domain adaptation: modify the text to make it more suitable for a different domain or topic. this can involve replacing domain-specific terms, adjusting terminology, or incorporating relevant keywords.
Text summarization: generate a summary of the text by condensing its main points into a shorter version. this can help create diverse data for summarization tasks and provide alternative perspectives on the original text.
Contextual word insertion: insert new words into the text that are contextually relevant and maintain the semantic meaning of the sentence. this can be done by using word embeddings to find words that are commonly used in similar contexts.
Text paraphrasing: paraphrase the text by rephrasing sentences while preserving the original meaning. this can be done using techniques such as sentence splitting, word substitution, and sentence rearrangement.
Contextual sentence deletion: delete certain sentences from the text while maintaining the coherence and semantic meaning of the remaining sentences. this can help create diverse data for document summarization tasks.
Text expansion: expand the text by adding additional details, examples, or explanations to enhance its content. this can be done by incorporating information from external sources or generating new sentences based on the existing text.
Emotion augmentation: add emotional expressions or sentiments to the text to convey different emotions. this can help create diverse data for emotion classification or sentiment analysis tasks.
Named entity substitution: replace named entities (such as names of people, organizations, or locations) in the text with other similar entities. this can introduce variations while maintaining the context of the sentence.
Sentence reordering: rearrange the order of sentences within the text while keeping the semantic coherence intact. this can create new narrative structures and perspectives.
Grammatical error injection: introduce grammatical errors into the text by randomly modifying verb tenses, subject-verb agreement, or punctuation. this can simulate noisy real-world data and improve model robustness.
Word embedding replacement: replace certain words in the text with their word embeddings. this can introduce variations while preserving the semantic meaning of the sentence.
Synonym replacement: replace words in the text with their synonyms. this can create alternative phrasing while maintaining the overall meaning.
Contextual word substitution: substitute words in the text with other words that have similar contextual meanings. this ensures that the replacement maintains the intended semantics.
Sentence splitting: split longer sentences into shorter ones, or vice versa, to create new sentence structures. this can help the model understand different sentence lengths and improve its generalization ability.
Sentence combination: combine multiple shorter sentences into a single longer sentence or vice versa. this can create diverse sentence structures and test the modelś comprehension of complex sentences.
Negation insertion: introduce negations into the text by adding words like "not" or "no" to change the polarity of certain statements. this can help the model understand negative contexts better.
Word masking: randomly mask certain words in the text by replacing them with a special token. this forces the model to rely on the surrounding context to understand the meaning of the masked word.
Character-level augmentation: modify individual characters within words, such as changing vowels or consonants, to generate diverse textual variations.
Sentence shuffling: shuffle the order of sentences within a paragraph or document to create new arrangements and test the modelś ability to understand different contexts.
Paraphrasing: rewrite the text using different phrasing or sentence structures while maintaining the original meaning.
Numerical value perturbation: add or subtract small random values to numerical values in the text to create slight variations.
Pos tagging variation: randomly change the part-of-speech tags of words in the text while ensuring grammatical correctness. this can introduce different syntactic patterns and word usages.
Sentence deletion: remove one or more sentences from the text to create a shorter version while still maintaining coherence and semantic meaning.
Sentiment modification: change the sentiment or emotion expressed in the text while keeping the content intact. this can involve altering positive statements to negative ones or vice versa.
Negation addition: add negation words (e.g., "not," "never") to the text to create negative versions of the original sentences.
Word order variation: randomly change the order of words within phrases or clauses in the text to generate new sentence arrangements.
Word sense disambiguation: replace ambiguous words in the text with their different senses to create diverse interpretations of the sentence.

Table 12: Automatic generated augmentation instructions.

	Augmentation Instruction
	Negation transformation: transform positive statements into negative ones by adding negation words or phrases, or vice versa.
	Contradiction generation: introduce contradictions within the text by modifying certain statements to be opposite to what they originally conveyed.
	Named entity replacement: identify named entities in the text (e.g., names of people, organizations) and replace them with similar entities to generate new variations.
	Word order shuffling: randomly shuffle the order of words within a sentence to create new sentence structures.
	Grammar error injection: introduce grammatical errors such as incorrect verb conjugation, subject-verb agreement, or punctuation mistakes to simulate natural language variation.
	Text shuffling: shuffle the order of sentences within a paragraph or paragraphs within a document to create new arrangements. this helps diversify the structure and flow of the text.
	Contextual deletion: remove certain words or phrases from the text while ensuring that the remaining content still conveys the same overall meaning. this tests the modelś ability to understand and fill in missing information.
	Sentence paraphrasing: generate paraphrases of the original sentences while preserving their meaning. this can be achieved through techniques like back-translation or paraphrase generation models.
	Named entity variation: replace named entities (such as names, locations, organizations) in the text with different variations to create diverse instances of the same sentence.
	Numerical variation: modify numerical values in the text by adding/subtracting a small random value or replacing them with synonyms/alternative representations.
	Negation augmentation: introduce negations in the text by adding "not" or other negation words to certain phrases or sentences. this helps the model understand negative contexts better.
	Data sampling: randomly sample subsets of the data to create smaller training sets for faster experimentation and exploration of different data distributions.
	Backtranslation: translate the text into another language and then translate it back to the original language. this can introduce variations in sentence structure and word choice while preserving the overall meaning.
	Sentence splitting/merging: split long sentences into shorter ones or merge short sentences into longer ones to create new sentence structures.
	Pronoun replacement: replace pronouns in the text with their corresponding nouns or vice versa, maintaining the semantic meaning of the sentence.
	Word deletion: randomly delete certain words from the text to create shorter or more concise sentences. this can simulate scenarios where some information is missing or incomplete.
	Character-level perturbation: introduce noise at the character level by randomly changing, deleting, or inserting characters in the text. this can help models become more robust to noisy inputs and improve generalization.
	Domain-specific term replacement: identify domain-specific terms in the text and replace them with synonyms or related terms specific to another domain. this can help models generalize better across different domains.
	Negation/positive conversion: convert negative statements to positive ones or vice versa to generate different perspectives or sentiments.
	Part-of-speech tagging: modify the part-of-speech tags of words in the text to create new grammatical arrangements and sentence structures.

Table 13: Automatic generated augmentation instructions.