

Few-shot LLM Synthetic Data with Distribution Matching

Jiyuan Ren*
rjy22@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Qinglin Jia
jiaqinglin2@huawei.com
Huawei Noah's Ark Lab
Beijing, China

Zhaocheng Du*
zhaochengdu@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Sunhao Dai
sunhaodai@ruc.edu.cn
Renmin University of China
Beijing, China

Zhihao Wen
wenzhihao4@huawei.com
Huawei Noah's Ark Lab
Singapore, Singapore

Chuhan Wu
wuchuhan@huawei.com
Huawei Noah's Ark Lab
Beijing, China

Zhenhua Dong†
dongzhenhua@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

CCS Concepts

- Computing methodologies → Natural language generation.

Keywords

Synthetic Data, Large Language Model, Data Augmentation

ACM Reference Format:

Jiyuan Ren, Zhaocheng Du, Zhihao Wen, Qinglin Jia, Sunhao Dai, Chuhan Wu, and Zhenhua Dong. 2025. Few-shot LLM Synthetic Data with Distribution Matching. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3701716.3715245>

1 Introduction

Despite the rapid development of large language models (LLMs), there remains a demand in the industry for smaller online language models tailored to high-latency scenarios [6, 7, 12, 13, 21, 23, 45, 56] like search engines [24]. These models are typically trained on pre-constructed datasets for online services. However, real-world data collection often suffers from various biases like selection bias [1, 8, 43] and long-tail issues [9, 48], and expensive manual cost [5, 10]. Synthetic data generation has the potential to mitigate these biases [30, 40], enhance data diversity [15], and ultimately improve model generalization and accuracy.

To fully harness the potential of synthetic data, it is essential to define how synthetic data can be high quality for a specific task? High quality can be assessed along various dimensions, such as low noise and fairness. However, in profit-driven industrial applications like search engines, we prioritize improving model accuracy as the objective. Based on Murphy's research [32], we propose the following definition of high quality: **the optimal dataset is that which most closely matches the distribution under which the model will be evaluated**. Under this definition, synthetic high-quality data requires matching real data in some key attributes.

Many existing methods adopted distribution matching as the objective function and used all or a subset of the real dataset to train a generative model that approximates the real data distribution, and then synthesizes data based on this model. For instance, RelGAN [33] designs a generative adversarial network architecture for

Abstract

As large language models (LLMs) advance, their ability to perform in-context learning and few-shot language generation has improved significantly. This has spurred using LLMs to produce high-quality synthetic data to enhance the performance of smaller models like online retrievers or weak LLMs. However, LLM-generated synthetic data often differs from the real data in key language attributes (e.g., styles, tones, content proportions, etc.). As a result, mixing these synthetic data directly with real data may distort the original data distribution, potentially hindering performance improvements. To solve this, we introduce **SynAlign**: a synthetic data generation and filtering framework based on key attribute distribution matching. Before generation, SynAlign employs an uncertainty tracker surrogated by the Gaussian Process model to iteratively select data clusters distinct from selected ones as demonstrations for new data synthesis, facilitating the efficient exploration diversity of the real data. Then, a latent attribute reasoning method is employed: the LLM summarizes linguistic attributes of demonstrations and then synthesizes new data based on them. This approach facilitates synthesizing diverse data with linguistic attributes that appear in real data. After generation, the Maximum Mean Discrepancy is used as the objective function to learn the sampling weight of each synthetic data, ensuring distribution matching with the real data. Our experiments on multiple text prediction tasks show significant performance improvements. We also conducted an online A/B test on an online retriever to demonstrate SynAlign's effectiveness. Our code is available [here](#).

*Both authors contributed equally to this research.

†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1331-6/25/04

<https://doi.org/10.1145/3701716.3715245>

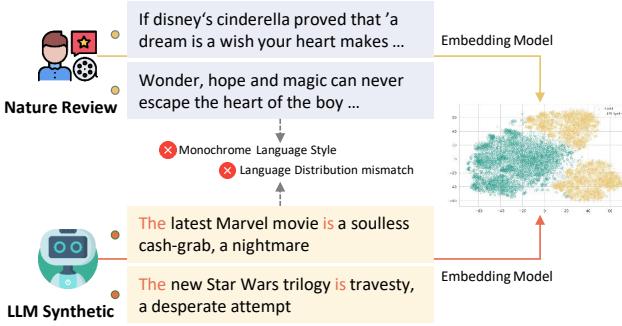


Figure 1: A case on how LLM synthetic samples misalign with human-generated samples.

text data generation tasks, ensuring that the generated data closely aligns with the original data distribution. Similarly, FewGen [31] fine-tunes an autoregressive pre-trained language model on a small sample dataset and employs it as a generator to synthesize a large number of new training samples, thereby augmenting the original training set. However, such methods require training a dedicated generative model for each scenario, incurring high manual costs. Additionally, due to the risk of catastrophic forgetting [35], these generators may lose their in-context learning capability.

Benefiting from massive high-quality training data and carefully designed training processes [34], LLMs have developed strong instruction following and in-context learning abilities that can generate responses for diverse tasks [3]. This has prompted academia and industry to use LLMs for zero-shot data synthesis to train their domain-specific models. For example, GPT-3Mix [51] utilizes large-scale language models to generate mixed samples, enhancing text datasets. Additionally, AttrPrompt [52] generates training data through diverse attribute prompts, improving model performance while reducing data bias.

However, experiments (see section 4.3) show two issues with the data synthesized by LLMs from the distribution matching perspective. First, LLM's zero-shot generation results cannot cover all real data's linguistic attributes (especially those "imperfect" ones like typos or incomplete sentences). Secondly, there is a significant discrepancy in the proportion of linguistic attributes between the LLMs synthetic data and the real data [16, 27]. Both are primarily due to LLM's limitations in understanding long text inputs, which prevents it from incorporating distributions of real data and previously synthetic data. These issues result in a considerable divergence between LLM-synthetic and real data, meaning that directly adding these synthetic data to the real data may not improve model performance.

To address these challenges, we designed **SynAlign** (**S**ynthetic data generation and **A**lignment framework), which incorporates three modules during LLMs' synthetic data generation process: Exploration-aware Sampling, Latent Attribute Reasoning, and Synthetic Distribution Alignment. Exploration-aware sampling is applied to the demonstration sampling process. It utilizes a Gaussian Process model as the samples' uncertainty-tracker. Samples with the highest uncertainty will be selected as demonstrations and their uncertainty will be updated after generation. With this process, data sampling module can efficiently explore real data

distribution. Selected demonstrations will be fed into the Latent Attribute Reasoning module to summarize key attributions covering language contents, styles, etc. Afterward, new data are synthesized based on these generalized latent attributes. After the data synthesis process, the Synthetic Distribution Alignment module employs a post-training approach to assign sampling weights for each synthetic sample by minimizing Maximum Mean Discrepancy between synthetic data and real data. After resampling the synthetic data based on sampling weights, we obtain the final synthetic data that more closely aligns with real data distribution.

Extensive experiments demonstrate that our method synthesizes high-quality data more efficiently, requiring fewer tokens. Compared to existing data generation algorithms, the synthetic data generated by SynAlign consistently achieves superior performance. The main contributions of this paper are as follows:

- We designed an Exploration-aware Sampling module that can efficiently explore all language attributes in real data. Data synthesized with latent attributes extracted from these samples could cover all real data's distribution.
- We designed a Synthetic Distribution Alignment module to post-align synthetic data distribution with real data distribution.
- Extensive experiments are conducted and demonstrate our method can generate high quality synthetic samples efficiently.

2 Related Work

The advent of LLMs has revolutionized synthetic data generation, offering significant advantages over traditional methods. LLMs excel in generating coherent and human-like text, making them effective tools for creating high-quality datasets [29]. Compared to manual data collection, LLMs provide notable benefits in flexibility, efficiency, and scalability. They can tailor datasets to specific needs by adjusting prompts and conditions [14], significantly reduce annotation costs [22], and automate the training pipeline, enabling broader application across multiple domains [17].

Despite these advantages, ensuring the quality and relevance of LLM-generated datasets requires robust generation techniques and curation strategies. Below, we review key methods for data generation, focusing on prompt engineering and multi-step generation, followed by strategies for data curation and distribution alignment.

2.1 LLM-Based Data Generation Methods

Prompt Engineering. Prompt engineering is critical for controlling the quality and diversity of synthetic data. Effective prompts typically define tasks clearly, specify generation conditions (e.g., themes or styles), and include in-context demonstrations, which help guide LLMs toward accurate outputs [18, 28]. For instance, Wang et al. [44] demonstrated how condition-based prompts improve stylistic diversity, while Li et al. [27] highlighted the role of examples in enhancing task alignment.

Multi-Step Generation. Multi-step generation addresses complex data needs by breaking the process into sub-tasks. Sample-wise decomposition divides data into smaller parts for step-by-step generation, improving coherence, as shown by He et al. [20]. Dataset-wise decomposition dynamically adjusts generation conditions to enhance diversity and coverage [44]. Our method builds on these

approaches by introducing systematic condition controls to further improve data quality and diversity.

2.2 Data Curation and Distribution Alignment

While LLMs excel at generating diverse data, synthetic datasets often contain noise or distributional biases that can hinder downstream performance [53]. To address these issues, curation strategies such as sample filtering and label enhancement have been proposed. Sample filtering uses heuristic metrics like confidence scores or influence functions to identify high-quality samples [39, 50]. Label enhancement techniques, such as knowledge distillation [49], refine labels[42] and reduce annotation errors[26].

To align synthetic data distributions with real-world data, methods like Maximum Mean Discrepancy (MMD) have been employed [29]. Our approach integrates systematic sample selection and MMD-based alignment, ensuring the synthetic data closely resembles real-world distributions while maintaining high quality.

3 Method

3.1 Overview

Given a real dataset $D_{ori} = \{(x_i, y_i)\}_{i=1}^N$ and a synthetic dataset $D_{gen} = \{(x'_j, y'_j)\}_{j=1}^M$ generated by a large language model (LLM), our goal is to enhance the quality and diversity of D_{gen} such that it improves the performance of smaller, domain-specific models in downstream tasks. To simplify the modeling of text distributions, both D_{ori} and D_{gen} are mapped into embedding spaces E_{ori} and E_{gen} using Sentence-BERT [36].

Directly generating D_{gen} often leads to a distributional gap between real and synthetic data due to LLM limitations. This gap arises from incomplete coverage of real data diversity or misaligned proportions of data attributes, which can degrade the utility of D_{gen} in downstream tasks.

To address these issues, we propose **SynAlign**, comprising three key modules: 1) **Exploration-aware Sampling**: This module uses a Gaussian Process (GP) uncertainty tracker to actively select diverse and representative real samples as demonstrations for synthetic data generation. 2) **Latent Attribute Reasoning**: Demonstrations are used to reason about key linguistic attributes via a Chain-of-Thought (CoT) [46] paradigm, guiding the LLM to explicitly attend to diverse attributes during generation. 3) **Synthetic Distribution Alignment**: Using Maximum Mean Discrepancy (MMD) [19], this module aligns D_{gen} with D_{ori} by computing sampling weights for synthetic samples, ensuring minimal distributional deviation.

By combining these modules, SynAlign produces high-quality, diverse synthetic data that better aligns with real data distributions, resulting in improved performance in downstream tasks.

3.2 Exploration-aware Sampling

Our method uses a few-shot prompting approach to generate synthetic data using LLM. Selecting representative demonstrations from the real dataset is crucial for guiding the LLM in producing high-quality synthetic data. Random selection may overfit to frequently occurring patterns. We utilize an uncertainty-aware sampling strategy to ensure that the synthetic data process efficiently covers all real data's language attributes.

To achieve this, we use a Gaussian Process model [47] as samples' uncertainty tracker $U(E_{ori})$. When initializing $U(E_{ori})$, we assign each sample's mean value as 0 and uncertainty (variance) as 1 and covariance between samples e_i, e_j as $\kappa(e_i, e_j)$.

$$\begin{bmatrix} U(e_0) \\ \vdots \\ U(e_n) \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \cdots & \kappa(e_1, e_n) \\ \vdots & \cdots & \vdots \\ \kappa(e_n, e_1) & \cdots & 1 \end{bmatrix}\right) \quad (1)$$

Where $\kappa(\cdot, \cdot)$ is chosen as Radial Basis Function Kernel as below to ensure covariance between selected and unselected samples increase as text embedding's similarity decreases.

$$\kappa(e_i, e_j) = \exp\left(-\frac{1}{2\tau} \|e_i - e_j\|\right) \quad (2)$$

Where τ is a hyper-parameter called bandwidth that controls covariance smoothness. Each demonstration selection phase includes the following two steps:

Step 1. Demonstration Selection. Samples with the highest uncertainty (variance) and its k -nearest samples D_{dem} will be selected out as demonstrations according to the newest updated $U(E_{ori})$. These samples are most different to LLM already seen demonstrations regarding linguistic attributes like styles, contexts etc.

$$D_{dem} = k\text{-NN}(D_{ori}, \text{argmax}_i(U(e_i|e_i \in E_{ori})), k) \quad (3)$$

Where $k\text{-NN}$ is the function that returns the k nearest samples of the most uncertain sample indexed by $\text{argmax}_i(U(e_i|e_i \in E_{ori}))$ from real dataset D_{ori} .

Step 2. Uncertainty Update. Once D_{dem} are used as demonstrations for data synthetic, their embedding E_{dem} 's uncertainty is reduced to 0 and constructed as posterior training data $(E_{dem}, 0)$. Those samples together with historical ones $(E_s, 0)$ will be used to update the unselected samples' uncertainty value $U(E_u)$ in the uncertainty tracker U . The updated sample uncertainty is given below:

$$U(E_u)|E_u, E_s, U(E_s) \sim N(\mu^*, \Sigma^*) \quad (4)$$

where μ^* equals to $\mathbf{0}$ because only uncertainty is used in the selection process. And variance(uncertainty) Σ^* is given below:

$$\Sigma^* = K(E_u, E_u) + I - K(E_u, E_s)K(E_s, E_s + I)^{-1}K(E_s, E_u) \quad (5)$$

These two steps are repeated iteratively to select demonstrations for LLM to generate synthetic data until all samples' uncertainties are below a predefined threshold. The overall procedure is listed in Appendix Algorithm 1.

This uncertainty-aware sampling strategy ensures that the selected demonstrations represent the diversity of real data distribution, as measured by the uncertainty in the Gaussian Process model. These selected examples are then used as demonstrations for the few-shot prompting of the LLM in the next stage of the data generation process.

3.3 Latent Attribute Reasoning

Demonstrations selected from the previous module are used to feed and assist LLM in understanding the diverse linguistic attributes in real data so that the synthetic data produced by LLM won't be monochrome. To explicitly ensure the synthetic data captures all linguistic attributes in the real data while maintaining content diversity, a two-stage process was designed by first reasoning out

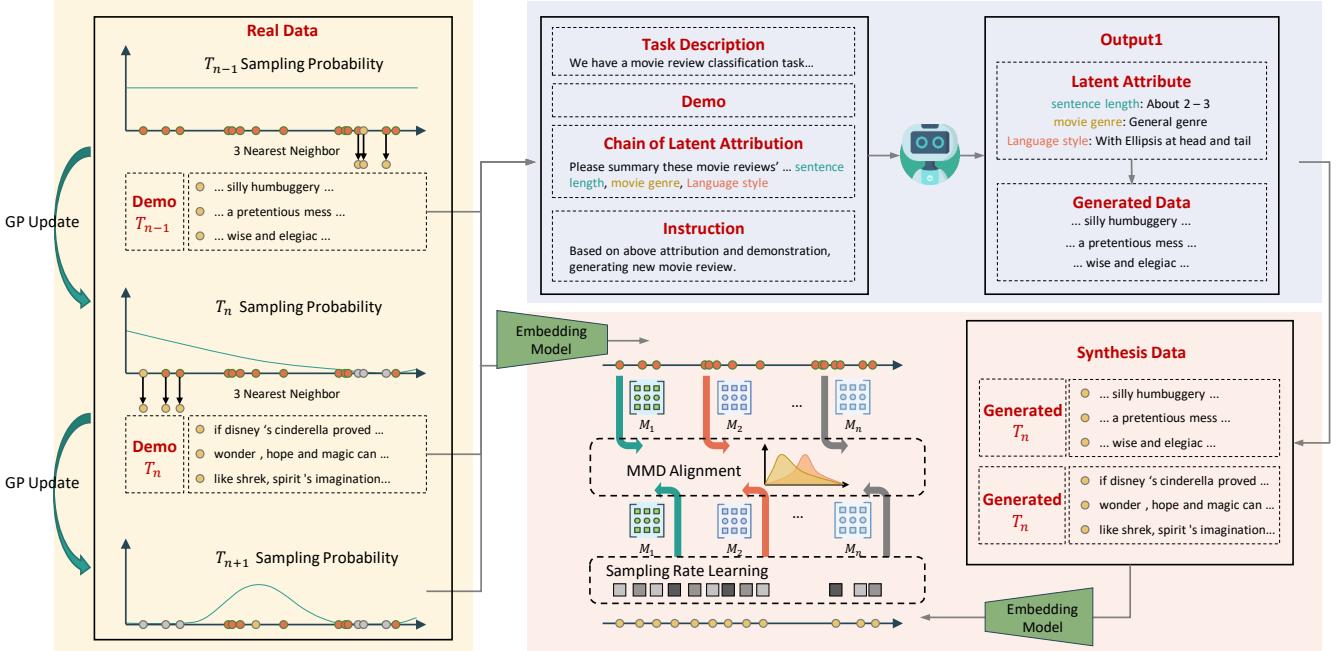


Figure 2: SynAlign comprises three modules for aligning the distribution of synthetic samples. First, the Exploration-aware Sampling Module selects real samples based on uncertainty to provide diverse inputs for the LLM. Next, the Latent Attribute Reasoning Module uses these samples as demonstrations to identify and generalize key language attributes for synthetic data generation. Finally, the Synthetic Distribution Alignment Module assigns sampling weights to synthetic data, which are then resampled accordingly to ensure alignment with the real data distribution.

linguistic attributes and then generating diverse content based on them.

Stage 1. Key Attribute Reasoning stage. The goal of this stage is to identify and summarize the key linguistic attributes of the selected demonstrations, which serve as a blueprint for synthetic data generation. To ensure the synthetic data D_{gen} reflects the structure and diversity of the real data D_{ori} , we let D_{gen} mimick key attributes $A = \{a_1, \dots, a_n\}$ extracted from D_{dem} . Referring to prior work AttrPrompt [52], we use LLM to identify crucial attributes. For instance, attributes in an Amazon product review dataset might include *Product Info*, *Usage Experience*, and *Writing Style*. These attributes provide a framework for summarizing the sampled examples.

Once the key attributes are identified, we construct reasoning prompts P_1 instructing the LLM to analyze D_{dem} and extract these key attributes. This results in a JSON format *Attribute Summary Set S*:

$$S = LLM(D_{dem}, A, P_1) = \{(a_1, v_1), (a_2, v_2), \dots, (a_n, v_n)\} \quad (6)$$

where each tuple (a_i, v_i) represents summarized attributes of selected demonstrations.

Stage 2. Attribute-Based Data Generation stage. We leverage the attribute summarized in Stage 1 to guide the LLM in generating synthetic data. By incorporating attribute summaries S into generation prompts P_2 , LLM can synthesize samples reflecting these key attributes, such as product information and writing style. For each attribute tuple in S , the LLM generates a synthetic sample

that adheres to the specified attributes, ensuring the generated data is both diverse and representative of the original dataset. Finally, by generating new data for each attribute summary, we construct the synthetic dataset D_{gen} .

$$D_{gen} = LLM(S, P_2) = \{(x_0, y_0), (x_1, y_1), \dots, (x_M, y_M)\} \quad (7)$$

The size of D_{gen} depends on the number of summaries in S and the samples generated per summary. The generated data covers a wide range of latent attributes, maintaining alignment with the original data distribution while introducing new variations in style and content.

3.4 Synthetic Distribution Alignment

Due to the limited input length, LLMs cannot fully account for the distribution of D_{gen} and D_{ori} . This limitation often results in discrepancies between the linguistic attribute distributions of the synthesized and real data, potentially causing a "seesaw effect" that degrades the model's accuracy in practical applications. To enhance distribution matching, we want to learn a post-transformational function $F_\omega(\cdot)$ to minimize the distance between these two distributions:

$$\operatorname{argmin}_{F_\omega(\cdot)} \operatorname{Dist}(D_{ori} || F_\omega(D_{gen})) \quad (8)$$

However, Due to the discrete and high-dimensional nature of linguistic data, measuring their distribution exactly is intractable. To address this, we adopted the Maximum Mean Discrepancy (MMD) method to approximate this matching objective. The basic idea of

MMD in our application is matching the mean embedding of E_{gen} and E_{ori} projected in Reproducing Kernel Hilbert Space (RKHS), which is equivalent to matching these two distributions [55].

$$\text{argmin}_{F_\omega}(\cdot) \sup_{\|\phi_\theta\|_H \leq 1} (E[\phi_\theta(D_{ori})] - E[\phi_\theta(F_\omega(D_{gen}))]) \quad (9)$$

Where the ϕ_θ is a family of functions parameterized by θ and H represent the RKHS, considering the ground truth distribution is intangible, we adopt its empirical approximation by making the following changes: (1) map the text into an embedding space to obtain continuous representations; (2) simplify the transformation function as a data sampling weight ω and (3) assuming $\phi_\theta(\cdot)$ as a $R^n \rightarrow R^1$ random linear projection matrix family Θ . Finally, we can derive the following objective function.

$$\text{argmin}_{\omega} E_{\theta \sim \Theta} \left\| \frac{1}{N} \sum_{i=1}^N \theta \cdot E_{ori} - \frac{1}{M} \sum_{i=1}^M \theta \cdot (\omega \cdot E_{gen}) \right\|^2 \quad (10)$$

To ensure diversity in the projections matrix set Θ , we use Gram-Schmidt orthogonalization[25] to initialize these random matrices. The parameters of each projection matrix θ_i are orthogonalized regarding the previously initialized matrix $\theta_{1:i-1}$, ensuring each network captures different aspects of the data.

$$\Theta = \{\theta_i | <\theta_i, \theta_j> = \delta_{ij} \forall i, j, \delta_{ij} = 1 \text{ if } i == j, \text{ otherwise } 0\} \quad (11)$$

The final sample weight of each synthetic sample ω is calculated by solving the linear equation set described in formula 10. In our implementation, we use gradient descent to solve ω iteratively.

After the importance weight of each sample ω has been learned in the distribution alignment process, we can perform re-sampling on the original synthetic dataset D_{gen} based on ω with replacement to construct a new synthetic dataset D'_{gen} which will have similar linguistic attribute distribution with the data D_{ori} . Mixing up the D'_{gen} with the original data D_{ori} can bring higher model performance than using D_{gen} . The pseudocode of our algorithm is given in 1

4 Experiments

4.1 Experimental Setup

We evaluate our method on three widely-used text classification datasets: SST-2 [41], AGNEWS [54], and Amazon [2], covering diverse tasks such as sentiment analysis, topic classification, and product review classification. These datasets are chosen for their varying challenges, including class imbalance and linguistic diversity, making them ideal for testing synthetic data generation methods. Table 6 summarizes their key characteristics.

To assess model performance, we adopt two standard metrics: Accuracy (Acc), which measures the proportion of correctly classified samples, and the F1 Score, calculated as the macro-average across all classes. The latter is particularly useful for datasets with imbalanced class distributions.

We compare it against several baselines, as summarized below.

- **Gold:** Models are trained solely on the original dataset without any synthetic data.
- **SimPrompt** [4]: Augments the original dataset with ζ synthetic samples generated using simple class-conditional prompts.
- **AttrPrompt** [52]: Uses attribute-rich prompts to generate diverse synthetic data.

- **SynAlign(all):** Trains models on the full synthetic dataset D_{gen} without distribution alignment.
- **SynAlign(random):** Trains models on the original dataset combined with ζ randomly selected samples from D_{gen} .
- **SynAlign(mmd):** Our proposed method, which selects ζ samples from D_{gen} using the MMD-based sampling approach described in Section 3.

We fine-tune pre-trained models, including DistilBERT [37] and BERT-base-uncased [11], to evaluate the generalization of our method across different model scales. The synthetic data D_{gen} is generated following the pipeline described in Section 3. Details on implementation, including training hyperparameters and optimization, are provided in the A.2.

4.2 Main Experimental Results

Table 1 reports results in terms of **Accuracy (Acc)** and **F1 score** for both BERT-base-uncased[11] and DistilBERT[38]. The results show that augmenting the original dataset with synthetic data consistently improves performance over the **Gold** baseline. This trend is consistent across datasets and models, demonstrating the benefit of synthetic data augmentation. Importantly, **DistilBERT**, despite its smaller size, achieves competitive results compared to BERT-base-uncased, highlighting the scalability of our method to lightweight models.

Among baseline methods, **AttrPrompt** generally outperforms **SimPrompt**, likely due to its ability to generate more diverse and representative synthetic samples. However, both are consistently surpassed by our proposed **SynAlign** approach, which uses exploration-aware sampling and distribution alignment to select high-quality synthetic data. Within SynAlign, the MMD-based sampling strategy (**SynAlign(mmd)**) delivers the best results, outperforming **SynAlign(all)** (which uses all generated samples) and **SynAlign(random)** (which selects samples randomly). These results highlight the importance of informed sample selection, as not all synthetic data contributes equally to performance.

SynAlign(mmd) consistently achieves the highest Accuracy and F1 scores across datasets and models. For example, on SST-2, SynAlign(mmd) with BERT-base-uncased achieves 93.30% Accuracy, outperforming the Gold baseline (92.48%) and all other augmentation methods. Similarly, on AGNEWS and Amazon, SynAlign(mmd) delivers superior results, demonstrating robustness across different tasks and domains.

Overall, the experimental results confirm the effectiveness of **SynAlign(mmd)** in leveraging synthetic data for model improvement. By selectively augmenting datasets with well-aligned samples, our method achieves consistent performance gains across datasets, domains, and model architectures, while maintaining scalability to lightweight models like DistilBERT.

4.3 Generated Data Analysis

Distributional Alignment: We measure the alignment between the original and generated data using **Wasserstein distance**, which quantifies the cost of transforming one distribution into another. Lower values indicate better alignment. Sentence embeddings are extracted using a pre-trained BERT model and visualized with t-SNE for qualitative analysis.

Table 1: Main results across three datasets. We report Accuracy (Acc) and F1 score for each method and model. The best results are highlighted in bold.

| Method | Model | SST-2 | | AGNews | | Amazon | |
|-------------------------|-------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | | Acc | F1 | Acc | F1 | Acc | F1 |
| LLM zero-shot | | 0.9351 | 0.9335 | 0.8232 | 0.8264 | 0.7556 | 0.732 |
| Gold | BERT-base-uncased | 0.9248 | 0.9246 | 0.9286 | 0.9398 | 0.8204 | 0.7961 |
| | DistilBERT | 0.9044 | 0.9045 | 0.9260 | 0.9248 | 0.8053 | 0.8053 |
| SimPrompt | BERT-base-uncased | 0.9264 | 0.9259 | 0.9403 | 0.9402 | 0.8274 | 0.8113 |
| | DistilBERT | 0.9148 | 0.9164 | 0.9346 | 0.9339 | 0.8165 | 0.8165 |
| AttrPrompt | BERT-base-uncased | 0.9260 | 0.9264 | 0.9441 | 0.9417 | 0.8305 | 0.8142 |
| | DistilBERT | 0.9231 | 0.9215 | 0.9395 | 0.9395 | 0.8262 | 0.8262 |
| SynAlign(all) | BERT-base-uncased | 0.9292 | 0.9299 | 0.9460 | 0.9441 | 0.8374 | 0.8244 |
| | DistilBERT | 0.9252 | 0.9212 | 0.9456 | 0.9451 | 0.8351 | 0.8351 |
| SynAlign(random) | BERT-base-uncased | 0.9286 | 0.9322 | 0.9429 | 0.9455 | 0.8204 | 0.8031 |
| | DistilBERT | 0.9203 | 0.9207 | 0.9441 | 0.9442 | 0.8296 | 0.8296 |
| SynAlign(mmd) | BERT-base-uncased | 0.9330 | 0.9352 | 0.9475 | 0.9470 | 0.8381 | 0.8266 |
| | DistilBERT | 0.9282 | 0.9211 | 0.9464 | 0.9433 | 0.8312 | 0.8312 |

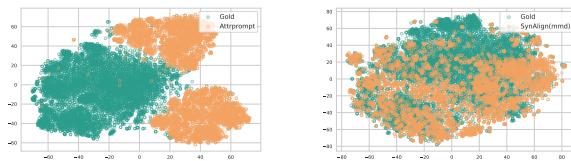


Figure 3: t-SNE visualization of sentence embeddings from SST-2. (a) Comparison between Gold and AttrPrompt; (b) Comparison between Gold and SynAlign (MMD). SynAlign (MMD) achieves better alignment.

Figure 3 illustrates the t-SNE visualizations for SST-2, comparing the original dataset (**Gold**) with data generated by different methods. In Figure 3(a), AttrPrompt’s generated data forms tight clusters far from the original data, suggesting that it captures only a limited subset of the original linguistic patterns. In contrast, Figure 3(b) shows that **SynAlign (MMD sampling)** generates data that closely aligns with the original distribution, covering a broader and more representative area of the original data space.

Quantitatively, Table 2 reports the Wasserstein distances for all datasets. **SynAlign (random)** achieves lower distances compared to **SimPrompt** and **AttrPrompt**, while **SynAlign (MMD)** consistently achieves the best alignment, demonstrating its effectiveness in selecting synthetic samples that better reflect the original data distribution.

Vocabulary Diversity: We measure vocabulary diversity by calculating the vocabulary size, defined as the number of unique words in each dataset. As shown in Table 3, **SynAlign (MMD)** generates datasets with higher vocabulary diversity than **SimPrompt** and comparable diversity to **AttrPrompt**. For example, on SST-2, SynAlign (MMD) achieves a vocabulary size of 7.4k, significantly larger than SimPrompt (1k) and close to AttrPrompt (7.2k).

Table 2: Wasserstein Distance between original and generated datasets. Lower values indicate better alignment.

| Data to compare | SST-2 | AGNEWS | Amazon |
|------------------------------|---------------|---------------|---------------|
| Gold vs SimPrompt | 0.1924 | 0.0468 | 0.0933 |
| Gold vs AttrPrompt | 0.0366 | 0.0913 | 0.1640 |
| Gold vs SynAlign(random) | 0.0082 | 0.0647 | 0.1209 |
| Gold vs SynAlign(mmd) | 0.0077 | 0.0516 | 0.1068 |

Table 3: Vocabulary sizes of original datasets and synthetic datasets generated by different methods.

| Data to compare | SST-2 | AGNEWS | Amazon |
|----------------------|-------|--------|--------|
| Gold | 13k | 158k | 90k |
| SimPrompt | 1k | 27k | 28k |
| AttrPrompt | 7.2k | 30k | 22k |
| SynAlign(all) | 7.9k | 32k | 35k |
| SynAlign(mmd) | 7.4k | 31k | 29k |

Although the generated datasets have smaller vocabulary sizes than the original datasets, this is expected due to the limited input prompts provided to the LLM. However, combining synthetic and original data compensates for this limitation, as evidenced by the performance improvements in Section 3. These results show that SynAlign not only enhances distributional alignment but also generates more linguistically diverse text, contributing to its superior performance.

4.4 Ablation Studies

4.4.1 Exploration-aware Sampling Coverage Speed Efficiently covering the original data distribution is critical for few-shot prompting. We compare our **exploration-aware sampling** with random sampling by evaluating their respective coverage rates of the original

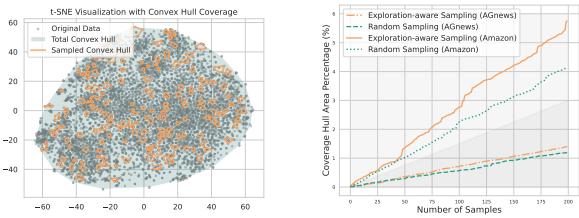


Figure 4: (a) Convex hull coverage example for SST-2. (b) Coverage rate comparison between Gaussian Process Active Sampling and random sampling on AGNEWS and Amazon. Gaussian Process achieves faster and broader coverage.

Table 4: Performance comparison between single-stage and two-stage generation on AGNEWS and Amazon datasets.

| Method | AGNEWS | Amazon |
|-------------------------|---------------|---------------|
| Gold | 0.8204 | 0.9353 |
| AttrPrompt | 0.8274 | 0.9441 |
| SynAlign(all) | 0.8319 | 0.9430 |
| One Stage | 0.8177 | 0.9437 |
| SynAlign(random) | 0.8319 | 0.9439 |
| SynAlign(mmd) | 0.8381 | 0.9475 |
| Two Stage | 0.8372 | 0.9460 |
| SynAlign(all) | 0.8212 | 0.9439 |
| SynAlign(mmd) | 0.8381 | 0.9475 |

data distribution. The coverage area is measured as the convex hull formed by selected samples in the t-SNE-reduced embedding space, with coverage rates computed iteratively for 200 sampling steps.

Figure 4(a) illustrates an example of convex hull coverage for SST-2, while Figure 4(b) compares coverage rates across AGNEWS and Amazon. Exploration-aware sampling consistently achieves faster and broader coverage of the data distribution compared to random sampling. Specifically, it avoids redundant sampling in densely populated regions and captures diverse, representative samples more effectively. This result highlights the efficiency of Gaussian Process Active Sampling in improving few-shot prompting performance.

4.4.2 Benefits of Latent Attribute Reasoning Our method employs a **two-stage generation process** that separates attribute generation (e.g., sentiment or topic) from synthetic data generation. This design encourages diversity compared to single-stage generation, which directly relies on prompt examples.

Table 4 compares the performance of models trained on data generated by single-stage and two-stage approaches. The results show that the two-stage generation consistently outperforms the single-stage approach across AGNEWS and Amazon datasets. For example, on AGNEWS, SynAlign(mmd) achieves an accuracy of 83.81% with two-stage generation, compared to 83.19% with single-stage generation. These improvements highlight the importance of reasoning about latent attributes to improve the quality and diversity of synthetic data.

4.4.3 Impact of MMD Distribution Alignment The MMD-based sampling strategy selects synthetic samples that are closely aligned with the original data distribution. As shown in Table 1, SynAlign (MMD) achieves the highest performance across all datasets, outperforming both random sampling and other baselines. For example, on AGNEWS, SynAlign (MMD) achieves an accuracy of 0.9475, compared to 0.9429 for random sampling.

The Wasserstein distance results in Table 2 further validate the effectiveness of MMD sampling. SynAlign (MMD) consistently achieves smaller distances compared to random sampling, indicating better alignment with the original data distribution. This improved alignment explains the observed gains in downstream performance.

4.5 Hyperparameter Analysis

4.5.1 RBF Kernel Length Scale τ in Exploration-aware Sampling The RBF kernel's length scale τ is a crucial parameter in the Gaussian Process model for tracking sample uncertainty. It controls the smoothness of the covariance function and determines the range of influence of selected samples. Smaller τ values lead to highly localized effects, while larger values smooth the uncertainty estimates over broader regions.

We evaluate the impact of τ on the convex hull coverage rate across SST-2, AGNEWS, and Amazon datasets. As shown in Figure 5 (a), the coverage rate exhibits consistent patterns across datasets. When τ is too small (e.g., $\tau < 0.3$), the coverage rate is low due to excessive focus on densely populated regions, leading to redundant sampling. As τ increases, the coverage rate improves, reaching its peak at dataset-specific optimal values (e.g., $\tau = 0.9$ for SST-2). However, when τ becomes too large ($\tau > 1.5$), the coverage rate declines as the sampling behavior becomes overly smooth, resembling random sampling.

To better understand the differences between datasets, we analyzed the distribution of pairwise Euclidean distances between sentence embeddings. These analyses, presented in the Appendix, indicate that AGNEWS has a more spread-out embedding space compared to SST-2 and Amazon, which explains why it benefits from a larger τ for optimal coverage.

4.5.2 Number of Nearest Neighbors k in Exploration-aware Sampling The parameter k , which determines the number of nearest neighbors selected during each sampling iteration, controls the trade-off between sample diversity and efficiency in Exploration-aware Sampling. Larger k values enable broader coverage of the embedding space, while smaller values focus on fewer, more representative samples.

We evaluate the impact of k on the convex hull coverage rate under the optimal RBF kernel length scale identified earlier. Figure 5 (b)-(d) shows that the coverage rate increases with k across all datasets, but the rate of improvement diminishes as k becomes large. However, larger k values incur higher computational costs, particularly for datasets with large embedding spaces like AGNEWS. Interestingly, when k is small, the performance of Exploration-aware Sampling is comparable to random sampling. This demonstrates that our method is more effective at utilizing larger sampling budgets to achieve higher coverage.

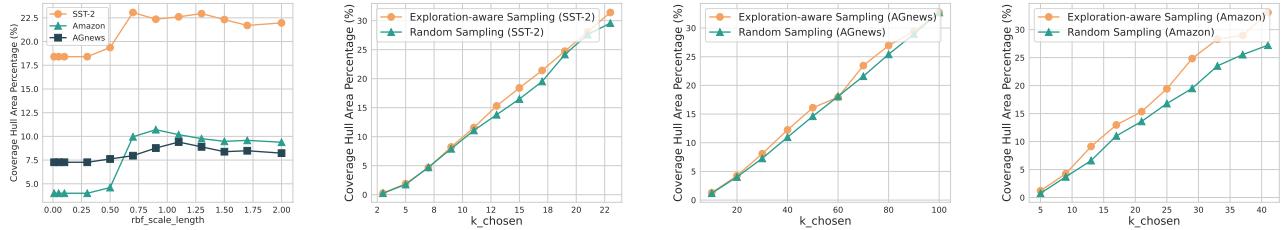


Figure 5: Coverage rate as a function of RBF kernel length scale τ for Exploration-aware Sampling across three datasets: SST-2, AGNEWS, and Amazon. Each curve represents the convex hull coverage rate at fixed sampling iterations.

4.5.3 Number of Projection Matrices $|\Theta|$ in Synthetic Distribution Alignment In the Synthetic Distribution Alignment stage, the number of projection matrices $|\Theta|$ plays a crucial role in aligning the original data distribution D_{ori} with the generated data distribution D_{gen} through MMD. This parameter determines the expressiveness of the alignment process: too few projection matrices may inadequately capture distributional differences, while too many may lead to computational overhead or overfitting.

To evaluate the effect of $|\Theta|$, we vary its value across $\{10, 50, 100, 500, 1000\}$ and measure the Wasserstein distance between D_{ori} and D_{gen} . Table 5 reports the results for SST-2, AGNEWS, and Amazon. For SST-2 and Amazon, the Wasserstein distance is minimized at $|\Theta| = 50$, indicating that moderate numbers of projection matrices are sufficient for effective alignment in datasets with simpler embedding spaces. In contrast, for AGNEWS, which has a more complex embedding space due to its higher diversity, the Wasserstein distance continues to decrease as $|\Theta|$ increases, reaching its lowest value at $|\Theta| = 100$ (0.00715). Further increases in $|\Theta|$ yield diminishing returns while increasing computational costs.

Table 5: Wasserstein distance between D_{ori} and D_{gen} as a function of the number of projection matrices $|\Theta|$.

| Dataset | 10 | 50 | 100 | 500 | 1000 |
|---------|---------|----------------|----------------|---------|---------|
| SST-2 | 0.11069 | 0.10516 | 0.10735 | 0.10931 | 0.10801 |
| AGNEWS | 0.00773 | 0.00748 | 0.00715 | 0.00705 | 0.00702 |
| Amazon | 0.06055 | 0.06357 | 0.06419 | 0.06375 | 0.06310 |

4.6 Online A/B Test

We deployed the SynAlign framework in the pre-ranking module of AppGallery's search advertising system to address two key challenges: the long-tail nature of app ads, which results in limited search data, and the significant query-app discrepancies caused by abstract app names (e.g., 'Presidential Election' vs. 'TikTok'). Training relevance models on exposure-click data using contrastive learning struggles with generalization due to these issues. While LLMs can augment user queries, zero-shot LLM synthesis often generates queries that deviate from real-world user preferences, leading to distribution mismatches and potential model convergence issues.

To address the distribution mismatch between synthetic and real queries, we utilized the SynAlign framework. During the SynAlign

synthesis process, we applied uncertainty sampling to efficiently analyze all query-item pairs and identify common user query patterns (e.g., app names, substrings, typos). These patterns were then combined with app names and input into Qwen2.5 to generate hundreds of thousands of synthetic queries. Both synthetic and real queries were mapped into the embedding space, where SynAlign's MMD-based method was used to assign a weight to each synthetic query. A new dataset was created by sampling queries based on these weights. Offline testing showed that models enhanced with distribution-aligned synthetic queries achieved a 0.26% improvement in AUC compared to unenhanced models.

For online deployment, synthetic queries were mixed with daily updated real queries for the experimental group, while the control group used only real queries. Over a week, the experimental group achieved a 2.86% increase in RPM and a 2.31% increase in CPM. This SynAlign-based data synthesis approach is now fully deployed to serve all users.

5 Conclusion

In conclusion, we observed that the data generated by LLMs often struggles to align perfectly with domain-specific linguistic styles. Directly mixing LLM-generated data with original data can disrupt the original data distribution, leading to degraded model performance. To address this issue, we proposed the SynAlign framework. This framework begins with an Exploration-aware Sampling module, which allows the LLM to efficiently perceive the full distribution of real-world data. Next, the Latent-Attribute Reasoning module summarizes and generalizes the linguistic attributes to guide the LLM in generating standardized synthetic data. Finally, the Synthetic Distribution Alignment module uses an MMD-based approach to align the distributions of synthetic and original data, effectively enhancing the performance of domain-specific tasks. Extensive experiments were conducted and proved the effectiveness of our method.

References

- [1] Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. *arXiv preprint arXiv:2204.01171* (2022).
- [2] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*. 440–447.

- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [4] Derek Chen, Celine Lee, Yunan Lu, Domenic Rosati, and Zhou Yu. 2023. Mixture of Soft Prompts for Controllable Data Generation. arXiv:2303.01580 [cs.CL] <https://arxiv.org/abs/2303.01580>
- [5] Sunhao Dai, Weihao Liu, Yiqi Zhou, Liang Pang, Rongju Ruan, Gang Wang, Zhenhua Dong, Jun Xu, and Ji-Rong Wen. 2024. Cocktail: A Comprehensive Information Retrieval Benchmark with LLM-Generated Documents Integration. *Findings of the Association for Computational Linguistics: ACL 2024* (2024).
- [6] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1126–1132.
- [7] Sunhao Dai, Ninglu Shao, Jieming Zhu, Xiao Zhang, Zhenhua Dong, Jun Xu, Quanyu Dai, and Ji-Rong Wen. 2024. Modeling user attention in music recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 761–774.
- [8] Sunhao Dai, Yiqi Zhou, Jun Xu, and Ji-Rong Wen. 2023. Dually Enhanced Delayed Feedback Modeling for Streaming Conversion Rate Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 390–399.
- [9] Yi Dai, Hao Lang, Yinhe Zheng, Fei Huang, and Yongbin Li. 2023. Long-tailed question answering in an open world. arXiv preprint arXiv:2305.06557 (2023).
- [10] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. 2018. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–40.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] <https://arxiv.org/abs/1810.04805>
- [12] Zhaocheng Du, Junhao Chen, Qinglin Jia, Chuhuan Wu, Jieming Zhu, Zhenhua Dong, and Ruiming Tang. 2024. LightCS: Selecting Quadratic Feature Crosses in Linear Complexity. In *Companion Proceedings of the ACM on Web Conference 2024*. 38–46.
- [13] Zhaocheng Du, Chuhuan Wu, Qinglin Jia, Jieming Zhu, and Xu Chen. 2024. A Tutorial on Feature Interpretation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 1281–1282.
- [14] Ronen Eldan and Yuanzhi Li. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759 [cs.CL] <https://arxiv.org/abs/2305.07759>
- [15] Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data augmentation for finetuning text generators. arXiv preprint arXiv:2010.01794 (2020).
- [16] Jingtong Gao, Zhaocheng Du, Xiaopeng Li, Xiangyu Zhao, Yichao Wang, Xiangyang Li, Huijing Guo, and Ruiming Tang. 2025. SampleLLM: Optimizing Tabular Data Synthesis in Recommendations. arXiv preprint arXiv:2501.16125 (2025).
- [17] Jiahui Gao, Renjie Pi, Yong Lin, Hang Xu, Jiacheng Ye, Zhiyong Wu, Weizhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2023. Self-Guided Noise-Free Data Generation for Efficient Zero-Shot Learning. arXiv:2205.12679 [cs.CL] <https://arxiv.org/abs/2205.12679>
- [18] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences of the United States of America* 120 (2023). <https://api.semanticscholar.org/CorpusID:257766307>
- [19] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.
- [20] Xingwei He, Zhenghao Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. 2023. Annollm: Making large language models to be better crowdsourced annotators. arXiv preprint arXiv:2303.16854 (2023).
- [21] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. arXiv preprint arXiv:2305.02301 (2023).
- [22] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuxin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large Language Models Can Self-improve. <https://openreview.net/forum?id=NiEtU7blzN>
- [23] Pengyu Jia, Yeqing Wang, Zhaocheng Du, Xiangyu Zhao, Yichao Wang, Bo Chen, Wanyu Wang, Huirong Guo, and Ruiming Tang. 2024. Erase: Benchmarking feature selection methods for deep recommender systems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5194–5205.
- [24] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.
- [25] Steven J Leon, Åke Björck, and Walter Gander. 2013. Gram-Schmidt orthogonalization: 100 years and more. *Numerical Linear Algebra with Applications* 20, 3 (2013), 492–532.
- [26] Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy F Chen, Zhengyu Liu, and Dify Yang. 2023. Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation. arXiv preprint arXiv:2310.15638 (2023).
- [27] 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 (2023).
- [28] Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jimmeng Rao, Steven Zheng, Daiyi Peng, Dify Yang, Denny Zhou, et al. 2024. Best practices and lessons learned on synthetic data. In *First Conference on Language Modeling*.
- [29] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. arXiv:2406.15126 [cs.CL] <https://arxiv.org/abs/2406.15126>
- [30] Yan Lyu, Sunhao Dai, Peng Wu, Quanyu Dai, Yuhao Deng, Wenjie Hu, Zhenhua Dong, Jun Xu, Shengyu Zhu, and Xiao-Hua Zhou. 2022. A Semi-Synthetic Dataset Generation Framework for Causal Inference in Recommender Systems. arXiv preprint arXiv:2202.11351 (2022).
- [31] Yi Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek F. Abdelzaher, and Jiawei Han. 2022. Tuning Language Models as Training Data Generators for Augmentation-Enhanced Few-Shot Learning. In *International Conference on Machine Learning*. <https://api.semanticscholar.org/CorpusID:253384628>
- [32] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.
- [33] Weili Nie, Nina Narodytska, and Ankit B. Patel. 2019. RelGAN: Relational Generative Adversarial Networks for Text Generation. In *International Conference on Learning Representations*. <https://api.semanticscholar.org/CorpusID:68160504>
- [34] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.
- [35] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. 2021. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.
- [36] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084 (2019).
- [37] V Sanh. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019).
- [38] V Sanh. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019).
- [39] Nabeel Seadat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. 2023. Curated Ilm: Synergy of Ilms and data curation for tabular augmentation in ultra low-data regimes. arXiv preprint arXiv:2312.12112 (2023).
- [40] Mohamed Ashik Shahul Hameed, Asifa Mehmood Qureshi, and Abhishek Kaushik. 2024. Bias Mitigation via Synthetic Data Generation: A Review. *Electronics* (2027-9292) 13, 19 (2024).
- [41] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.
- [42] Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W White, Longqi Yang, et al. 2024. Tnt-ilm: Text mining at scale with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5836–5847.
- [43] Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. arXiv preprint arXiv:2005.03642 (2020).
- [44] Ruida Wang, Wangchunshu Zhou, and Mrinmayra Sachan. 2023. Let's Synthesize Step by Step: Iterative Dataset Synthesis with Large Language Models by Extrapolating Errors from Small Models. arXiv preprint arXiv:2310.13671 (2023).
- [45] Yeqing Wang, Zhaocheng Du, Xiangyu Zhao, Bo Chen, Huirong Guo, Ruiming Tang, and Zhenhua Dong. 2023. Single-shot feature selection for multi-task recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 341–351.
- [46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.
- [47] Christopher Williams and Carl Rasmussen. 1995. Gaussian processes for regression. *Advances in neural information processing systems* 8 (1995).
- [48] Fei Wu, Raphael Hoffmann, and Daniel S Weld. 2008. Information extraction from Wikipedia: Moving down the long tail. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 731–739.
- [49] Ruixuan Xiao, Yiwen Dong, Junbo Zhao, Runze Wu, Minmin Lin, Gang Chen, and Haobo Wang. 2023. Freeal: Towards human-free active learning in the era of

- large language models. *arXiv preprint arXiv:2311.15614* (2023).
- [50] Jiacheng Ye, Jiahui Gao, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Progen: Progressive zero-shot dataset generation via in-context feedback. *arXiv preprint arXiv:2210.12329* (2022).
- [51] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomeyeong Park. 2021. GPT3Mix: Leveraging Large-scale Language Models for Text Augmentation. In *Conference on Empirical Methods in Natural Language Processing*. <https://api.semanticscholar.org/CorpusID:233296100>
- [52] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J. Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias. *ArXiv* abs/2306.15895 (2023). <https://api.semanticscholar.org/CorpusID:259275123>
- [53] Yue Yu, Yuchen Zhuang, Rongzhi Zhang, Yu Meng, Jiaming Shen, and Chao Zhang. 2023. Regen: Zero-shot text classification via training data generation with progressive dense retrieval. *arXiv preprint arXiv:2305.10703* (2023).
- [54] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *NeurIPS* 28 (2015).
- [55] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.
- [56] Yuang Zhao, Zhaocheng Du, Qinglin Jia, Linxuan Zhang, Zhenhua Dong, and Ruiming Tang. 2024. Retrievable Domain-Sensitive Feature Memory for Multi-Domain Recommendation. *arXiv preprint arXiv:2405.12892* (2024).

A Appendix

A.1 Dataset Information

To evaluate the effectiveness of SynAlign, we conduct experiments on three widely used datasets: SST-2, AGNEWS, and Amazon. Table 6 summarizes the key attributes of these datasets.

For each dataset, we report the number of training samples, test samples, and classes. Additionally, we include the amount of synthetic data generated by the LLM and the final number of samples selected through SynAlign’s sampling strategy. These datasets span diverse domains (e.g., reviews, news, and web content), ensuring a comprehensive evaluation of the proposed framework.

A.2 Implementation Details

A.2.1 Hardware Configuration All experiments were conducted on a server equipped with an NVIDIA A6000 GPU (48GB memory) and an Intel(R) Xeon(R) Gold 6242R CPU @ 3.10GHz.

A.2.2 Classifier Training Parameters For the final classification task, we fine-tune a pre-trained language model using the parameters listed in Table 7. These include the learning rate (lr), batch size, training epochs, weight decay, and warmup ratio. The warmup ratio specifies the proportion of training steps used for learning rate warmup to stabilize training.

Algorithm 1 The Algorithm of the Proposed SynAlign

```

Input: Real dataset  $D_{ori}$ , Sample uncertainty tracker  $U(\cdot)$ , Key Attribute Set  $A$ , Embedding model  $F$ 
Output: Original Dataset  $D_{gen}$ 
Mapping  $D_{ori}$  to  $E_{ori}$  with  $F$ 
Initialize  $U(E_{ori})$  with standard GP model with RBF kernel
Initialize generated text pool  $D_{gen} = \emptyset$ 
while  $\max(U(E_{ori})) > \sigma$  do
    Select demonstrations  $D_{dem}$  with formula 3
    Set  $U(E_{dem})$  as 0 and update  $U(E_{ori})$  with formula 4, 5
    Extract key attribute set  $S$  from  $D_{dem}$  with formula 6
    Generate  $D'_{gen}$  based on  $S$  with formula 7
     $D_{gen} = D'_{gen} \cup D_{gen}$ 
end while
Mapping  $D_{gen}$  to  $E_{gen}$  with  $F$ 
Initialize Random Matrix set  $\Theta$  with Gram-Schmidt algorithm
Initialize sampling weight  $\omega$  for each embedding in  $E_{gen}$ 
Train  $\omega$  by minimizing MMD loss between  $E_{ori}$  and  $E_{gen}$ 
Resample  $D_{gen}$  based on  $\omega$  as the final synthetic data  $D_{gen}$ 
Return  $D_{gen}$ 
```

Table 6: Summary of datasets used in the experiments.

| Attribute | SST-2 | AGNEWS | Amazon |
|-------------------|--------|--------|--------|
| Domain | Review | News | Web |
| #Train | 67k | 120k | 13.8k |
| #Test | 1.8k | 7.6k | 1.2k |
| #Class | 2 | 4 | 23 |
| #Generated | 9k | 16k | 18.4k |
| #Sampled | 6k | 6k | 13.8k |

A.2.3 Convex Hull Coverage To calculate the coverage rate in 4.4.1, we first reduce the dimensionality of the sentence embeddings for both the original and generated datasets using t-SNE. A k-d tree is then constructed using the 2D t-SNE embeddings of the original dataset to enable efficient neighbor searching. The convex hull of the t-SNE embeddings of the original dataset is taken as the target distribution’s total coverage area. We initialize an empty buffer set \mathcal{B} to store convex hulls formed during sampling. At each sampling iteration, a single example is selected, and its embedding is combined with the k -nearest neighbors (identified using the k-d tree) to create a new convex hull. If this convex hull overlaps with any existing convex hulls in \mathcal{B} , they are merged to form a larger convex hull. The total area of all convex hulls in \mathcal{B} is then calculated and compared to the total coverage area of the original dataset to compute the coverage rate. This process is repeated iteratively for 200 sampling steps.

A.3 Prompt Design

To guide the synthetic data generation process, we design two stages of prompts tailored to capture and generalize the linguistic attributes of each dataset. The prompts are presented as examples in Figures 6 and 7.

A.3.1 Stage 1: Attribute Summarization In the first stage, the LLM analyzes a set of example data from the target dataset and generates a structured summary of its key linguistic and semantic attributes. The prompt instructs the model to extract attributes such as topics, language habits, and writing styles, depending on the dataset. For instance, for the SST-2 dataset, the attributes include *movie genres*, *topics*, *language habits*, and *review length*. This summarization step ensures that the generated synthetic data adheres to the original dataset’s characteristics.

A.3.2 Stage 2: Synthetic Data Generation In the second stage, the LLM generates synthetic data based on the summarized attributes from the first stage. For example, in the Amazon dataset, the model is instructed to write multiple unique product reviews in the same category while adhering to the provided attribute summary. The prompt ensures diversity by specifying that each review should focus on distinct subtopics and employ varied language styles. Similar prompts are designed for the SST-2 and AGNEWS datasets, with dataset-specific attributes and generation instructions.

A.3.3 Dataset-specific Attributes The attributes extracted and used for generation vary across datasets, as summarized below:

- **SST-2:** Movie genres, topics, language habits, and review length.

Table 7: Parameters for Classifier Training.

| Parameter | SST-2 | AGNEWS | Amazon |
|------------------------|-------|--------|--------|
| lr | 5e-5 | 5e-5 | 5e-5 |
| Batch size | 32 | 32 | 32 |
| Training epochs | 6 | 6 | 6 |
| Weight decay | 1e-4 | 1e-4 | 1e-4 |
| Warmup ratio | 6% | 6% | 6% |

Prompt structure of stage 1

[Task]
You are provided with an example from <Dataset descriptions>. Analyze the example and summarize its key characteristics, which will guide the generation of new data.

[Sample Data]
Here are examples in JSON format from the dataset:
<Example data str>

[Instructions]
1. Summarize the following characteristics:
<List of attributions>
2. Provide the summary in the following JSON format.

Now, begin summarizing:

Prompt structure of stage 2

[Task]
Suppose you are an Amazon review writer. Please write <#gen> unique reviews for different but similar products in the same category as the product described below. Each review should focus on a different product in the same category, highlighting similar core issues but with distinct language and perspectives.

[Demostription summary]
Here is the summary of the example provided:
<Sampled data attribution summary>

[Instructions]
1. Each review should describe a product in the same category as the one in the Product Info, but for a different product with similar characteristics.
2. Follow the attributes described in the review_info (e.g., usage experience, writing style, subtopics).
3. Select only 1-2 subtopics from the provided list for each review.
4. Ensure each review is unique, concise, and distinct in language.

Now, begin summarizing:

Figure 6: Prompt used in Stage 1 (Attribute Summarization).
The LLM is instructed to analyze a dataset example and extract key linguistic and semantic attributes, such as topics, language habits, and writing styles, to guide the synthetic data generation process.

- **AGNEWS:** News topics, writing style, news length, subtopics, and location.
- **Amazon:** Product information, usage experience, writing style, review length, language habits, and subtopics.

These prompts are designed to adapt to the characteristics of each dataset, ensuring that the synthetic data captures the essential features of the original data while maintaining diversity and linguistic consistency.

Figure 7: Prompt used in Stage 2 (Synthetic Data Generation).
The LLM generates multiple unique samples based on the attributes summarized in Stage 1. For example, in the Amazon dataset, it generates reviews for similar products while ensuring diversity in subtopics and language styles.