**Abstract**

Recently, dataset-generation-based zero-shot learning has shown promising results by training a task-specific model with a dataset synthesized from large pre-trained language models (PLMs). The final task-specific model often achieves compatible or even better performance than PLMs under the zero-shot setting, with orders of magnitude fewer parameters. However, synthetic datasets have their drawbacks. They have long been suffering from low-quality issues (e.g., low informativeness and redundancy). This explains why the massive synthetic data does not lead to better performance – a scenario we would expect in the human-labeled data. To improve the quality of dataset synthesis, we propose a progressive zero-shot dataset generation framework, PROGEN, which leverages the feedback from the task-specific model to guide the generation of new training data via in-context examples. Extensive experiments on five text classification datasets demonstrate the effectiveness of the proposed approach. We also show PROGEN achieves on-par or superior performance with only 1% synthetic dataset size compared to baseline methods without in-context feedback.

**1 Introduction**

Dataset generation with pre-trained language models (PLMs) has attracted enormous interest recently due to the superior generative capacity of PLMs. Given task-specific supervision, recent work (Anaby-Tavor et al., 2020; Puri et al., 2020; Kumar et al., 2020; Lee et al., 2021, *inter alia*) manages to fine-tune the PLMs to synthesize high-quality datasets for downstream applications. Nevertheless, obtaining task supervision from human experts can be expensive or even unrealistic. Recent attempts (Schick and Schütze, 2021; Wang et al., 2021; Meng et al., 2022, *inter alia*) turn their eyes to the unsupervised dataset generation.

Among them, ZEROGEN (Ye et al., 2022) proposes to first convert the task descriptions into carefully designed prompts (Petroni et al., 2019; Brown et al., 2020), and then use these prompts to steer the PLMs to synthesize the training data for the final task model. This approach allows highly efficient inference as the final task model only has orders of magnitude fewer parameters compared to PLMs, yet achieves compatible or even better performance than PLMs under the zero-shot setting.

The major drawback of synthetic datasets, however, is they often suffer from low-quality issues (e.g., low informativeness, redundancy). Despite we can generate as much data as computational resource allows, the massive generated data does not automatically translate into better performances, unlike in the human-labeling scenario.

To address this problem, we propose a progressive zero-shot dataset generation framework (Figure 1b), called PROGEN. In a nutshell, PROGEN learns a model for a downstream task by performing two phrases alternatively – using

---

*Work done while interning at Shanghai AI Lab.*
PLMs to create labeled examples leveraging the feedback from the current task-specific model, and training a task-specific model given the generated labeled examples. To compute reliable signals as feedback, we employ the influence function (Koh and Liang, 2017; IF) to quantify contribution to the loss for each training point. In the context of zero-shot learning where no human-annotated data is assumed, we integrate a noise-resistant objective in the calculation of IF so that it can tackle the noise in the synthetic dataset. To incorporate feedback into PLMs, we sort the training samples based on their quantified influence score, and formulate those most influential ones as in-context examples (Brown et al., 2020) to steer the generation. Overall, PROGEN has the following advantages: 1) the quality estimation phrase requires no human annotations, thus works in a purely zero-shot learning setting; 2) unlike most controllable generation methods that tune or require the access to PLMs (Keskar et al., 2019; Dathathri et al., 2020; Liu et al., 2021, inter alia), the in-context feedback phrase does not need to modify parameters in the PLM and incurs minimal disturbance to its generation procedure. Our main contributions are three folds:

- We propose a progressive framework for zero-shot dataset generation to generate high-quality dataset (§3);
- We propose noise-resistant influence function to estimate the quality of each sample without any human annotations (§3.1), and a learning-free controllable generation method via in-context feedback (§3.2);
- Across multiple text classification datasets, we show our framework obtains better performance over various prompt-based methods, and achieves on-par zero-shot performance with only 1% synthetic dataset size, when compared to methods without in-context feedback (§4).

Our code can be found at https://github.com/HKUNLP/ProGen.

2 Background

In this section, we briefly review the baseline approaches of zero-shot dataset generation and how the synthesized dataset can be used for zero-shot learning on downstream tasks.

Zero-shot Dataset Generation

Take text classification task as an example, vanilla zero-shot dataset generation methods (Meng et al., 2022; Ye et al., 2022) aims to generate a synthetic dataset \( \mathcal{D} = \{(x, y)\} \) with the help of a PLM \( \mathcal{P} \). They first sample a class label \( y \) from a uniform distribution:

\[
y \sim U(y_1, y_2, \ldots, y_k),
\]

where \( k \) is the number of classes. They then wrap \( y \) up into a label-descriptive prompt \( T(y) \) to steer the generation of \( x \):

\[
x \sim \mathcal{P}(\cdot|T(y)).
\]

Since the parameters of \( \mathcal{P} \) is frozen and the generation \( x \) for each \( y \) is deterministic, different sampling algorithms (e.g., Top-k sampling (Fan et al., 2018) and nucleus sampling (Holtzman et al., 2020)) can be adopted to increase the diversity of generated dataset. A synthetic dataset is constructed after pairing the generated \( x \) with \( y \).

Dataset-generation-based Zero-shot Learning

The vast linguistic (Jawahar et al., 2019; Goldberg, 2019; Tenney et al., 2019) and factual (Petroni et al., 2019; Jiang et al., 2020b) knowledge encoded in PLMs’ parameters is the key towards the success of conventional prompt-based zero-shot learning (PROMPTING) (Brown et al., 2020). However, PROMPTING fails to fully exert the capacity of PLMs and heavily relies on gigantic PLMs during inference. This motivates another line of work (Meng et al., 2022; Ye et al., 2022) to explore a more flexible and efficient way of conducting zero-shot learning based on dataset generation. Given the synthetic dataset generated as above, a task-specific model is trained, allowing any task-specific inductive bias and with an order-of-magnitude smaller number of parameters compared to PLMs. The performance of the final task-specific model is mostly dominated by the quality of the synthetic dataset. A low-quality dataset degrades the final zero-shot performance.

3 PROGEN

We now describe our framework for progressive zero-shot dataset generation via in-context feedback (PROGEN), as shown in Figure 2. We follow ZEROGEN (Ye et al., 2022) to build the backbone of our framework. Concretely, we first train a task-specific model (TAM) with partially generated dataset. Then, assuming no access to human...
Formally, influence function measures the change in the model’s loss on the test data-point $z_{\text{test}} = (x, y)$ if we up-weight the loss of a training data-point $z$ by $\epsilon$:

$$I_{\text{up,loss}}(z, z_{\text{test}}) \triangleq \frac{dL(\hat{\theta}_t, \hat{\theta})}{d\epsilon} \bigg|_{\epsilon=0} = \nabla_\theta L(z_{\text{test}}, \hat{\theta}) + \epsilon L(z, \hat{\theta})$$

$$I_{\text{up,loss}}(z, D_{\text{val}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta}) + \epsilon L(z, \hat{\theta}),$$

where $
abla_\theta$ denotes the gradient with respect to the parameter $\theta$. Our noise-robust validation-set level influence function is defined as:

$$I_{\text{up,loss}}(z, D_{\text{val}}) = -\nabla_\theta L'(D_{\text{val}}, \hat{\theta}) = -H^{-1}_\theta \nabla_\theta L(z, \hat{\theta}),$$

where $L'$ is a noise-tolerant loss. In this work, we adopt Reverse Cross-Entropy (RCE) loss (Wang et al., 2019), which has the following form:

$$L'(D_{\text{val}}, \hat{\theta}) = \sum_{i=1}^{|D_{\text{val}}|} \ell'_{\hat{\theta}}(\hat{y}_i, y_i) = -\frac{1}{|D_{\text{val}}|} \sum_{i=1}^{C} \hat{y}_i^c \log(y_i^c),$$

where $\hat{y}_i$ is the predicted class for sample $i$, $C$ is the number of classes. A smaller negative value of $\ell_{\hat{\theta}}(0)$ is approximated to a constant $A$ in the case of $y_i^c = 0$. 

3.1 Annotation-free Quality Estimation

There are many factors in measuring the quality of a dataset, e.g., diversity, annotation correctness, spurious biases (Mishra et al., 2020; Wiegreffe and Marasovic, 2021). However, it is often very subjective, making it unrealistic to calculate them all automatically. Our solution to this is to infer the quality of the individual samples in synthetic datasets using the performance of the final task-specific model trained on the dataset as the surrogate. Concretely, we propose to apply influence function (Koh and Liang, 2017) on the task-specific model to give sample-level influence scores with regard to the loss of validation set. However, a clean validation set, which is crucial for producing reliable influence scores, is inaccessible in the zero-shot learning setting. Thus, we use a synthetic validation set and harness the influence function with a noise-robust objective to handle the potential noise in the synthetic validation set.
\( I_{\text{up,loss}} \) indicates that upweighting the corresponding training sample will decrease the validation loss more, thus the sample is more valuable. After sorting training samples with \( I_{\text{up,loss}} \) ascendingly, we select top-\( M \) (e.g., 50) most valuable samples to form a tiny dataset, which we denote as \( \mathcal{D}_{\text{helpful}} = \{ (x, y) \} \).

In practice, we randomly sample a subset of \( \mathcal{D}_{\text{train}} \) and adopt the stochastic estimation method described in Koh and Liang (2017) to efficiently compute \( I_{\text{up,loss}} \).

### 3.2 Feedback via In-context Learning

After identifying influence scores for each previously generated sample, we hypothesize that including additional samples similar to those most helpful ones can boost downstream performance. Instead of purely paraphrasing those helpful samples individually, which may hinder the diversity of synthetic dataset, we expect the model to learn from the overall distribution of those helpful samples and generate new samples of similar quality.

Motivated by the striking capability of the in-context Learning (Brown et al., 2020) for PLMs, we propose to use the identified important samples as in-context examples, so that they can shift the generation distribution of PLMs to the ones that are more beneficial to the training of the final task-specific model. Formally, each \( x \) is now generated as follows:

\[
\mathbf{x} \sim \mathcal{P}(|T(y_1, \mathbf{x}_1), \ldots, T(y_k, \mathbf{x}_k), T(y))
\]

where all the in-context examples \( \{(x_i, y_i)\}_{i=1}^{k} \) are randomly selected from \( \mathcal{D}_{\text{helpful}} \). Compared with controllable text generation methods that directly modify the parameters in the PLM, we argue that the in-context learning methods incur minimal disturbance to the model’s generation procedure.

The overall framework of PROGen is elaborated in Algorithm 1.

### 4 Experiments

#### 4.1 Setup

**Datasets** We evaluate our method on five natural language text classification datasets, including IMDB (Maas et al., 2011), SST-2 (Socher et al., 2013), Rotten Tomatoes (Pang and Lee, 2005), Elec (McAuley and Leskovec, 2013), and Yelp (Zhang et al., 2015). Among these datasets, IMDB, SST-2, and Rotten Tomatoes are binary classification benchmarks for movie reviews, Elec and Yelp are binary classification tasks for electronic product reviews and restaurant reviews, respectively. The sizes of the training and test set are 25k/25k, 6.9k/0.8k, 8.5k/1k, 25k/25k, and 560k/38k for IMDb, SST-2, Rotten Tomato, Elec, and Yelp, respectively.

**Evaluation Strategy** Following Ye et al. (2022), we evaluate the quality of the synthetic dataset by first training a task-specific model (TAM) with the dataset, and then testing it on a human-annotated dataset (i.e., test set). We also explore other evaluation metrics in § 4.4.

**Baselines** The TAM trained with the synthetic dataset can perform zero-shot inference, hence, we compare PROGen with other zero-shot learning baselines:

- **Prompting.** The prompt-based zero-shot classification method based via PLMs (Brown et al., 2020).
- **Prompting”.** The calibrated prompting method that reweighs each option according to its priori likelihood (Holtzman et al., 2021).
- **ZeroGen.** A recent zero-shot learning work via dataset generation (Ye et al., 2022). They first generate a dataset with a carefully designed instruction, and then train a tiny task-specific model (TAM) to conduct zero-shot inference.

We also provide a non-zero-shot learning baseline **SUPERVISED** where the same TAM is used but trained on the human-annotated training set.

---

### Algorithm 1 Progressive Zero-shot Dataset Generation

**Require:** a PLM, a TAM, feedback interval \( I \), iterations \( T \).

1. \( \mathcal{D}_{\text{train}} \leftarrow \emptyset \)
2. \( \mathcal{D}_{\text{helpful}} \leftarrow \emptyset \)
3. \( \mathcal{D}_{\text{val}} \leftarrow \) Generate a validation set with PLM.
4. for feedback iteration \( t = 1, 2 \ldots T \) do
5. \( \mathcal{D}_{\text{new}} \leftarrow \) Generate a dataset of size \( I \) with PLM and \( \mathcal{D}_{\text{helpful}} \) via Eqn. 6.
6. \( \mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{new}} \)
7. TAM \leftarrow Training with \( \mathcal{D}_{\text{train}} \) and \( \mathcal{D}_{\text{val}} \).
8. \( \mathcal{D}_{\text{helpful}} \leftarrow \) Select most helpful subset from \( \mathcal{D}_{\text{train}} \) with TAM and \( \mathcal{D}_{\text{val}} \) via Eqn. 4.
9. end for

**Output:** \( \mathcal{D}_{\text{train}} \).
We provide an ablation study over various types of feedback, and summarize the results in Table 2. We find that it incorporates the previously generated dataset as feedback to steer generation. We provide an ablation study over various types of feedback, and summarize the results in Table 2. We find that

### Implementation Details
Following Ye et al. (2022), we use GPT2-XL (Radford et al., 2018) and Nucleus Sampling (Holtzman et al., 2020) with $p = 0.9$ for dataset generation. Regarding prompt selection, we adopt a series of prompts for each task. The details of prompt selection are provided in Appendix A. By default, the feedback interval $I$ is set to 1k, and iteration $T$ is set to 100, which ends up with a dataset of size 100k in total. Calculating IF score for all training points is computationally expensive, thus we only sample 10k samples in each iteration. In practice, we find generate data using feedback all the time hinders diversity, thus we only apply feedback half of the time.

We implement an LSTM-based model and a DistilBERT model as TAM to measure the quality of the synthetic dataset. For the LSTM-based model, we use Adam optimizer (Kingma and Ba, 2015), a learning rate of 1e-3, an embedding dim of 100, a hidden size of 300, and a layer number of 1. For DistilBERT, we fine-tune on each dataset with Adam optimizer, with a learning rate of 2e-5, a weight decay of 0.01, and other default hyper-parameters as suggested by HuggingFace Transformers library (Wolf et al., 2019). While using stochastic estimation in the influence function, we randomly sample 10k samples from the whole synthetic dataset, and calculate influence score for those samples over the whole validation set. This operation roughly costs 7 minutes. For all the experiments, we run on a single NVIDIA A100 GPU, and generating 100k examples cost 28h on average for PROGEN.

### 4.2 Main Results
We evaluate the generated datasets by training two different task-specific models and testing their performance on multiple downstream tasks. The results are shown in Table 1. We find that PROMPTING suffers from high variance over various prompts in zero-shot learning, and PROMPTING* substantially improves average accuracy and reduces variance across different choices of the prompt through calibration, which is also observed by previous work (Zhao et al., 2021; Holtzman et al., 2021; Min et al., 2022a). Compared with PROMPTING and PROMPTING*, ZEROGEN achieves superior performance by distilling task-related knowledge through dataset generation and handling the downstream tasks with a discriminator rather than a generator. Despite ZEROGEN’s success, PROGEN further boosts both average accuracy and variance by improving the quality of the synthesized dataset via a generate-then-feedback framework.

### 4.3 Ablations
One of the main contributions of the proposed progressive zero-shot dataset generation framework is that it incorporates the previously generated dataset as feedback to steer generation. We provide an ablation study over various types of feedback, and summarize the results in Table 2. We find that
Figure 3: Comparison of various ways to select important examples and the corresponding effects on test set accuracy and loss when removing them. Helpful-CE: helpful examples identified by vanilla influence function with cross-entropy loss. Helpful-RCE: helpful examples identified by robust influence function with reverse cross-entropy loss.

providing random synthetic examples as feedback consistently outperforms the baseline method. Our hypothesis is that task-related in-context examples may demonstrate further task-related information than the prompt, and thus benefit the generation process. Selecting examples with vanilla influence function (i.e., CE), only achieves on-par performance with random selection, due to the fact that the signal comes from the noisy validation set is not reliable. In contrast, applying a noise-tolerant objective (i.e., RCE) on the validation set achieves superior performance, which is resistant to the noise in validation set and is able to find more accurate important examples. This proves better task-related signals can further improve the generation quality. Moreover, we find selecting in-context examples from test set examples obtains the best results, which indicates the model does learn from better in-context examples.

4.4 Analysis

Noise-tolerant influence function provides better estimation in a noisy-validation set scenario.

To see whether using a robust loss function on the validation set contributes to a more accurate estimate, we use a fixed synthetic dataset, remove the estimated important examples, and show how the accuracy and loss of a task-specific model trained with the remained dataset changes. We study three estimation methods, various remove ratios, and evaluate on SST-2 gold test set, as shown in Figure 3. We find filtering random synthetic examples almost does not hurt accuracy and achieves similar accuracy with only 10% examples. Removing helpful examples identified by influence function with cross-entropy increases loss to some extent, but the degree of change is less than using reverse cross-entropy. This shows reverse cross-entropy augmented influence function could offer a more accurate estimate. We also compare results on the test set and artificial noisy test set as validation set in Appendix C, and demonstrate that the two losses are similar when the validation set is clean, but reverse cross-entropy is more effective when the validation set is noisy.

Format of in-context examples is important.

Given the identified important examples, it’s also unknown how to organize these examples as in-context examples. Previous work suggests the order of these examples plays a key role in model performance (Kumar and Talukdar, 2021; Lu et al., 2022), and the performance improves as the number of in-context examples increase (Brown et al., 2020). However, these effects are still under-explored in zero-shot dataset generation.

Suppose we have identified a bunch of positive and negative important examples, and are going to generate with a positive sentiment prompt (e.g., “The movie review in positive sentiment is: ‘”), we study the following formats of in-context examples:

- **Base**: no in-context examples (ZEROGEN).
- **F-1**: positive and negative examples are randomly placed.
- **F-2**: positive examples are placed before negative.
- **F-3**: positive examples are placed after negative.
- **F-4**: only positive examples are placed.

\(^2\)It assumes the validation set objective is the same as the training set.

\(^3\)To prevent trivial solutions (i.e., directly copying in-context examples), we remove the generated texts that are highly overlapped with any given in-context examples.
• **F-5**: only positive examples are placed, but the label information is not expressed (e.g., using prompt "The movie review is: "<X>"" for each in-context example, where <X> is the text placeholder to fill in for each in-context example.).

Besides, we also study the number of in-context examples, and the results are shown in Figure 4. We have the following observations. First, a modest number of in-context examples (e.g., no more than 8) consistently improves the performance, however, more in-context examples do not always turn into better performance. We leave the study on larger PLMs (e.g., GPT-3) and PLMs supporting more in-context examples as future work. Second, the performance of different orders of positive and negative examples varies in different tasks, while masking the label information (i.e., F-5) consistently improves performance.

**Prompt selection is less important for PROGEN while scaling dataset is still valuable.** In this part, we visualize the performance change over two representative prompts (i.e., a good prompt $P_2$ and a relatively bad prompt $P_1$) and various scales of dataset, and the comparison is shown in Figure 5.

Overall, we find that PROGEN is more effective for the bad prompt than the good one, and the bad prompt can achieve comparable results with the good one with PROGEN on all the datasets. This indicates the quality of dataset can be iteratively improved with previous lower-quality dataset slice, which shares a similar spirit with Self-Training (Lee et al., 2013) that also learns from its own predictions. Besides, we find PROGEN can achieve similar or superior performance to ZEROGEN with only 1% (100k vs. 1k) size of synthetic dataset. This becomes more meaningful when we only have restricted access to PLMs in real-world applications.

**In-depth analysis of the synthetic dataset.** In previous sections, we measure the quality of the synthetic dataset by training a TAM with that dataset and evaluating on downstream human-annotated data. In this section, we provide other measurements for a more comprehensive understanding of the synthetic dataset. We measure $D$ from two perspectives, i.e., texts and labels. Regarding texts, we use Self-BELU (Zhu et al., 2018) to measure its own diversity and MAUVE (Pillutla et al., 2021) to measure the similarity between prediction distribution and ground-truth distribution. We measure the correctness of labels with an oracle model trained on human-annotated data as done in Ye et al. (2022). The comparison of different metrics on the Elec dataset is reported in Table 3.
Table 3: Quality of various generated datasets measured by metrics in different granularity. PROGEN (Gold) refers to selecting in-context examples from gold test set. \( \mathcal{X} \) and \( \hat{\mathcal{X}} \) represent synthetic set and test set, respectively.

<table>
<thead>
<tr>
<th>Granularity</th>
<th>Self-BLEU ( f(\mathcal{X}) )</th>
<th>MAUVE ( f(\mathcal{X}, \hat{\mathcal{X}}) )</th>
<th>Correctness ( f(\mathcal{X}, \hat{\mathcal{X}}, \mathcal{Y}) )</th>
<th>TAM ( f(\mathcal{X}, \hat{\mathcal{X}}, \hat{\mathcal{Y}}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt P1</td>
<td>ZEROGEN</td>
<td>19.46</td>
<td>79.13</td>
<td>62.44</td>
</tr>
<tr>
<td></td>
<td>PROGEN</td>
<td>20.70</td>
<td>81.69</td>
<td>74.48</td>
</tr>
<tr>
<td></td>
<td>PROGEN (Gold)</td>
<td>27.37</td>
<td>92.45</td>
<td>89.01</td>
</tr>
<tr>
<td>Prompt P2</td>
<td>ZEROGEN</td>
<td>15.98</td>
<td>68.99</td>
<td>79.64</td>
</tr>
<tr>
<td></td>
<td>PROGEN</td>
<td>17.66</td>
<td>74.64</td>
<td>80.96</td>
</tr>
<tr>
<td></td>
<td>PROGEN (Gold)</td>
<td>28.12</td>
<td>94.34</td>
<td>87.61</td>
</tr>
<tr>
<td>Prompt P3</td>
<td>ZEROGEN</td>
<td>16.08</td>
<td>77.61</td>
<td>82.99</td>
</tr>
<tr>
<td></td>
<td>PROGEN</td>
<td>19.80</td>
<td>80.55</td>
<td>84.30</td>
</tr>
<tr>
<td></td>
<td>PROGEN (Gold)</td>
<td>26.16</td>
<td>93.75</td>
<td>89.59</td>
</tr>
</tbody>
</table>

Firstly, we find ZEROGEN achieves the highest diversity, and PROGEN degrades diversity. This indicates PLMs can generate texts similar to in-context examples, and the high Self-BLEU score for PROGEN (Gold) is mainly due to the limited number of in-context examples (i.e., 38k for Elec test set vs. 100k for synthetic dataset). Secondly, PROGEN well shifts the generation distribution towards ground-truth distribution, and PROGEN (Gold) achieves significantly higher MAUVE scores. Finally, both PROGEN and PROGEN (Gold) increase label correctness, which is also highly reflected in TAM. Overall, providing feedback improves synthetic datasets in both text distribution and label correctness, but also slightly decreases diversity.

5 Related Work

5.1 Dataset Generation with PLMs

The accuracy of neural models highly depends on the availability of large-scale human-annotated training data, which, however, can be prohibitively expensive to obtain at scale. Recent advances in generative language models (Radford et al., 2019; Brown et al., 2020) arouse great interests on generating synthetic dataset with PLMs. Some works generate data with a generative model fine-tuned on the public human-annotated dataset (Anaby-Tavor et al., 2020; Puri et al., 2020; Kumar et al., 2020; Lee et al., 2021). Regarding the low quality generations, sample selection (Yang et al., 2020; Liu et al., 2022a) have also been used as postprocessing, which is complementary to our method that improves the dataset quality during generation.

In the context of zero-shot dataset generation, previous approaches adopt prompt-based methods (Jiang et al., 2020a; Shin et al., 2020; Mishra et al., 2021) to generate data without any human-annotations (Schick and Schütze, 2021; Meng et al., 2022; Ye et al., 2022; Gao et al., 2022). The synthetic dataset can be used to train a task-specific model and perform zero-shot inference on downstream tasks. In contrast to our work, all the previous works generate the whole dataset at once, while we consider the quality of previously generated instances and improve the dataset quality during generation.

5.2 In-context Learning

Brown et al. (2020) suggest that large PLMs can learn a task by conditioning on a few input-output demonstration pairs as prompt. This paradigm, known as In-context learning, is especially attractive as it eliminates the need for updating parameters of the large language model. Subsequent works include better ways of choosing in-context examples (Liu et al., 2022b; Lu et al., 2022; Rubin et al., 2021), learning with an in-context learning objective (Min et al., 2022a; Chen et al., 2022), empirical analysis of why in-context learning works (Min et al., 2022b), theoretical analysis that in-context learning can be formalized as Bayesian inference (Xie et al., 2021), and explorations on other tasks (e.g., semantic parsing (Pasupat et al., 2021), dialogue state tracking (Hu et al., 2022; Xie et al., 2022)). To the best of our knowledge, all previous works study in-context learning in a few-shot learning setting. In contrast, this work focuses on a zero-shot learning setting for dataset generation task and the PLMs’ ability to learn from in-context synthetic important examples to produce better dataset.

6 Conclusions

This work proposes PROGEN for zero-shot dataset generation, which progressively improves the dataset quality by leveraging feedback from a task-specific model trained on the current dataset. By evaluating zero-shot performance with the trained model, we show PROGEN can generate a much smaller (e.g., 1%) synthetic high-quality dataset that achieves comparable or superior performance to baseline method. We also provide a variety of analyses, including formats of in-context examples, other measurements on synthetic datasets, and the influence of prompt selection.
Limitations

Our work depends on the PLMs’ following abilities: (1) learning from in-context examples; (2) generating relatively high-quality data when using only the manual prompt. This means that if the task can not be well described by a prompt or the PLM is not exposed to enough task-related data in the pre-training stage, the progressive dataset generation process may fail due to the extremely low-quality initial dataset slice and validation set. It can also affect the task-specific model’s ability to identify important examples: a noisy validation data point can fool the classifier into trusting mislabeled examples that fall close to it, further degrading the generation quality. In addition, calculating influence function in practice suffers from low-efficiency issues. In practice, we sample a subset from the entire synthesized dataset in each iteration to reduce the computation, which can be sub-optimal in quality estimation accuracy.

Acknowledgement

We thank the anonymous reviewers whose suggestions helped clarify this work. This work is partially supported by the Shanghai Committee of Science and Technology (Grant No. 21DZ1100100), and the joint research scheme of the National Natural Science Foundation of China (NSFC) and the Research Grants Council (RGC) under grant number N_HKU714/21.

References


Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding. CoRR, abs/2202.04538.


A Prompt Design

Table 4 summarizes the text prompts used in this work. We choose prompts with the most representative "Control Code" and "Natural Language" styles as discovered by Ye et al. (2022). We also include a two-stage prompt $P_3$ that uses an additional generated text by PLM as condition, e.g., the movie name is generated by prompt "Movie: ".

B Detailed Results on Each Prompt

The detailed results on each prompt are reported in Table 5 and Table 6.

C Robust Influence Function on Artificial Noisy Data

To investigate the ability of robust influence function in a noisy-validation set scenario, we create an artificial noisy dataset based on the human-annotated dataset. Specifically, we reverse a portion (e.g., 40%) of ground-truth labels in the training and validation set, and compare the results of using gold validation set and mislabeled validation set when calculating influence function. We show the result comparison in Figure 6, where we remove examples based on the calculated score, retrain the model with the rest of dataset, and evaluate it on a held-out human-annotated set. We find when the validation set is well labeled, both cross entropy-based and reverse cross entropy-based influence function achieve on-par performance. However, a noisy validation set with cross-entropy has a great impact on the quality estimation – removing identified helpful examples only slightly degrades accuracy. In contrast, a noise-resistant objective still provides reliable estimates.

Figure 6: Comparison on artificial noisy dataset.
### Table 4: Multiple text prompts for each setting. "<X>" refers to the test input for PROMPTING setting. "<TASK>" represents "movie", "electronic product" and "restaurant" for IMDb/SST-2/Rotten Tomatoes, Elec, Yelp datasets, respectively. "<C>" is a generated text formulated as additional condition to steer generation.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Id Prompt Details</th>
<th>Label word &lt;Y&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROMPTING/PROMPTING*</td>
<td>&lt;Y&gt; &lt;TASK&gt; Review: &quot;&lt;X&gt;&quot; The &lt;TASK&gt; review in &lt;Y&gt; sentiment is: &quot;&lt;X&gt;&quot;</td>
<td>Positive/Negative</td>
</tr>
<tr>
<td>ZEROGEN/PROGEN</td>
<td>&lt;Y&gt; &lt;TASK&gt; Review: &quot;&quot; The &lt;TASK&gt; review in &lt;Y&gt; sentiment is: &quot;&quot; The &lt;TASK&gt; review in &lt;Y&gt; sentiment for &lt;TASK&gt; &quot;&lt;C&gt;&quot; is: &quot;&quot;</td>
<td>Positive/Negative</td>
</tr>
</tbody>
</table>

### Table 5: Results on each prompt with DistilBERT as task-specific model.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Prompt</th>
<th>IMDB</th>
<th>SST-2</th>
<th>Rotten Tomato</th>
<th>Elec</th>
<th>Yelp</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPERVISED</td>
<td>-</td>
<td>87.24</td>
<td>89.68</td>
<td>83.67</td>
<td>92.63</td>
<td>95.42</td>
<td>89.73</td>
</tr>
<tr>
<td>PROMPTING</td>
<td>P1</td>
<td>60.36</td>
<td>52.64</td>
<td>52.91</td>
<td>68.08</td>
<td>68.33</td>
<td>60.46</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>80.64</td>
<td>89.45</td>
<td>84.24</td>
<td>77.44</td>
<td>82.70</td>
<td>82.89</td>
</tr>
<tr>
<td>PROMPTING*</td>
<td>P1</td>
<td>75.73</td>
<td>76.72</td>
<td>73.55</td>
<td>76.41</td>
<td>75.57</td>
<td>75.60</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>78.89</td>
<td>88.53</td>
<td>83.77</td>
<td>79.65</td>
<td>85.03</td>
<td>83.17</td>
</tr>
<tr>
<td>ZEROGEN</td>
<td>P1</td>
<td>74.20</td>
<td>75.57</td>
<td>69.50</td>
<td>81.85</td>
<td>85.08</td>
<td>77.24</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>83.27</td>
<td>86.12</td>
<td>82.46</td>
<td>86.62</td>
<td>88.67</td>
<td>85.43</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>83.76</td>
<td>86.61</td>
<td>83.11</td>
<td>87.58</td>
<td>89.76</td>
<td>86.16</td>
</tr>
<tr>
<td>PROGEN</td>
<td>P1</td>
<td>84.22</td>
<td>87.16</td>
<td>83.02</td>
<td>88.00</td>
<td>89.06</td>
<td>86.29</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>84.31</td>
<td>86.01</td>
<td>81.52</td>
<td>90.27</td>
<td>89.48</td>
<td>86.32</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>83.82</td>
<td>88.42</td>
<td>84.05</td>
<td>88.72</td>
<td>89.63</td>
<td>86.93</td>
</tr>
</tbody>
</table>

### Table 6: Results on each prompt with LSTM as task-specific model.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Prompt</th>
<th>IMDB</th>
<th>SST-2</th>
<th>Rotten Tomato</th>
<th>Elec</th>
<th>Yelp</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPERVISED</td>
<td>-</td>
<td>84.60</td>
<td>76.30</td>
<td>77.49</td>
<td>86.36</td>
<td>91.30</td>
<td>83.21</td>
</tr>
<tr>
<td>PROMPTING</td>
<td>P1</td>
<td>60.36</td>
<td>52.64</td>
<td>52.91</td>
<td>68.08</td>
<td>68.33</td>
<td>60.46</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>80.64</td>
<td>89.45</td>
<td>84.24</td>
<td>77.44</td>
<td>82.70</td>
<td>82.89</td>
</tr>
<tr>
<td>PROMPTING*</td>
<td>P1</td>
<td>75.73</td>
<td>76.72</td>
<td>73.55</td>
<td>76.41</td>
<td>75.57</td>
<td>75.60</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>78.89</td>
<td>88.53</td>
<td>83.77</td>
<td>79.65</td>
<td>85.03</td>
<td>83.17</td>
</tr>
<tr>
<td>ZEROGEN</td>
<td>P1</td>
<td>60.33</td>
<td>63.88</td>
<td>61.73</td>
<td>71.16</td>
<td>81.29</td>
<td>67.68</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>74.80</td>
<td>80.50</td>
<td>76.92</td>
<td>74.12</td>
<td>84.58</td>
<td>78.18</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>75.40</td>
<td>82.22</td>
<td>78.80</td>
<td>82.24</td>
<td>85.39</td>
<td>80.81</td>
</tr>
<tr>
<td>PROGEN</td>
<td>P1</td>
<td>77.43</td>
<td>81.08</td>
<td>76.64</td>
<td>79.21</td>
<td>84.44</td>
<td>79.76</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>77.30</td>
<td>79.13</td>
<td>76.17</td>
<td>84.38</td>
<td>85.97</td>
<td>80.59</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>78.81</td>
<td>82.68</td>
<td>78.99</td>
<td>84.96</td>
<td>87.68</td>
<td>82.62</td>
</tr>
</tbody>
</table>