Improving Text Generation for Product Description by Label Error Correction

Tong Guo

1Affiliation not available

April 01, 2024
Abstract

Text generation is an important method to generate accurate and available product description from product title. Product description generation’s main problem for online E-commerce application is the available rate of generated text. The available rate of online deployment standard needs to reach above 99%. Model-centric method is limited by the quality of the training dataset. To handle the problem, we propose our data-centric method to improve the generation model’s available rate from 88.0% to 99.2%. Our approach helps in building models using LLMs (large language models) annotation results and constructing datasets to obtain better results than LLMs. Also, our method simplifies the human labeling work to 2-class choices to label, which improve the labeling speed. In summary, our method saves about 10x of the labeling time and achieves 99.2% accuracy to be deployed online.

1 Introduction

In e-commerce, product description can attract shoppers and improve sales. But manually writing a successful product description is highly time-consuming. Text generation (Zhang et al., 2022; Prabhumoye et al., 2020) technologies play a crucial role in this range of applications.

In recent years, the development of deep learning (DL) has brought breakthroughs on text generation. The sequence to sequence (Seq2Seq) models use encoder-decoder transformer (Vaswani et al., 2017) for better model flexibility. The most representative models of this type include T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). In this paper, we adopt the T5 (Raffel et al., 2020) model to conduct our data-centric experiments.

Text generation has an input or a source sequence $X$ and an output or a target sequence $Y$ to be generated. In our product description generation tasks, $X$ is the product title and $Y$ is the product description. The examples are shown in Table 1.

In online applications, we mainly face low accuracy or low available rate problem: Model-centric method (Guo, 2021) is limited by the quality of the training dataset. The generation results’ accuracy needs to be above 99%, because too many error descriptions will result in a poor user experience. Our initial available rate is only 88%, after we initially construct the training dataset by querying ChatGPT (Ouyang et al., 2022; OpenAI, 2023). The reason of the low accuracy is: There is a certain amount of error data in the training dataset that is not suitable to display. For example, the product is titled ‘fish-flavored shredded pork’. But the corresponding description in training dataset is ‘this dish is spicy’. So the product description may not suit for not-spicy ‘fish-flavored shredded pork’. More examples are shown in Table 2 and 3.

The speed of manual annotation is also a very critical issue. In our scenario, 80% of the time is spent on labeling data. For example, if it consumes 2 weeks to annotate 100,000 data. So if we use the baseline method which 5x peoples label the same data to ensure data quality, it will consume us 10 weeks to complete the data preparation work. On the other end, it only consumes 1 week if we reduce the labeling time by 2x.

To solve the above problems, we design the Self-Predict and Choose method to correct the error data in ambiguous data, and we design Self-Search and Remove method to solve the problem of in-distribution error data, while overcoming the problem of errors in manual annotations.

The contributions of our paper are:

(1) we propose the data-centric method for improving text generation model’s accuracy or available rate. Our method can achieve 99.2% available rate in our product description generation task to be deployed online.

(2) Our method focuses on using minimal label-
Table 1: Good examples of text generation for product description.

<table>
<thead>
<tr>
<th>Input / Product Title</th>
<th>Output / Product Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birthday cake</td>
<td>Enjoy the taste and feel the beauty of the birthday.</td>
</tr>
<tr>
<td>Spicy chicken chunks</td>
<td>Crispy on the outside and tender on the inside.</td>
</tr>
<tr>
<td>Chicken soup with pig stomach</td>
<td>Nourishing and warming, with a long-lasting aftertaste.</td>
</tr>
<tr>
<td>Cheese hot dog</td>
<td>Full of cheese, can be pulled into strings.</td>
</tr>
<tr>
<td>Original mung bean cake</td>
<td>Made from peeled mung beans, pure and delicate.</td>
</tr>
<tr>
<td>Large cup of milk tea with two flavors</td>
<td>Long-lasting aftertaste, rich and fragrant.</td>
</tr>
</tbody>
</table>

2 Method

The whole pipeline is shown in Figure 1. In this paper, we adopt the T5 (Raffel et al., 2020) model to conduct our experiments. The pipeline contains 6 steps. In this section, we illustrate the detail of each step and how the algorithms work. In the discussion section, we illustrate the motivation why we design these steps.

2.1 Initial Training Dataset Construction

This section corresponds to the Step-1 in Figure 1. We collect our initial training dataset by querying ChatGPT. Each prompt is formed by concatenating a product title. We ask ChatGPT to write descriptions for the products. We tried to add some product attributes as prompt, but most of the ChatGPT’s results do not relate to the product attributes. Table 1, Table 2 and Table 3 show the prompt examples and the ChatGPT’s results. Our T5 (Raffel et al., 2020) model trained on this initial dataset gets 88% available rate, under human evaluation.

2.2 Out-of-distribution Ambiguous Data

Out-of-distribution ambiguous data is observed in our training dataset. The examples are shown in Table 2. These data has similar inputs, the outputs are very different. Some of them are error data.

2.3 In-distribution Error Data

In-distribution error data is observed in our training dataset. The examples are shown in Table 3. These data and its similar data are error data. These error data cannot be found by using Algorithm 1.

2.4 Self-Predict and Choose

This section corresponds to the Step-2 in Figure 1. The algorithm detail is shown in Algorithm 1.

Because we have observed that there are many ambiguous data in the training dataset. So we design this algorithm is to correct error data in the
ambiguous data by predicting itself and human re-labeling. We train the seq2seq model until the dev loss no longer decreases.

In Algorithm 1, we have the model_v0 trained on the dataset of last step. Then we use model_v0 to predict outputs for the inputs of training dataset. If the model output is significantly different from the output of the same input in the training dataset, then we manually choose a better output for the input. Then we get the corrected dataset_v1. In this paper, if the model output and training data’s output do not have common token, we identify they are significantly different and not similar.

2.5 Self-Search and Remove

This section corresponding to the Step-5 in Figure 1. The algorithm detail is shown in Algorithm 2.

Because we have observed that there are many extremely error data in the training dataset. These extremely error data are in-distribution with the training dataset. So these data cannot be fixed by Self-Predict and Choose method of Algorithm 1. Algorithm 1 is mainly to find the out-of-distribution data.

So we design this algorithm is to fix these data in training dataset by manually annotating the seed dataset. The seed dataset is randomly sampled from the training dataset. We try to retrieve the error data by the smallest number of seed data.

In Algorithm 2, we have the model_v1 trained on the dataset of last step. Then we use model_v1 to predict results for the seed dataset and manually find the error data and right data in seed dataset. Then we use this human feedback: We search in the training dataset by querying each error data of seed dataset. Then we find all the most similar data to the error data in training dataset and remove

<table>
<thead>
<tr>
<th>Input / Product Title</th>
<th>Output / Product Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latte coffee</td>
<td>Can’t sleep? Have a cup of latte coffee.</td>
</tr>
<tr>
<td>Latte coffee with fresh milk</td>
<td>The milk foam is dense and delicate, and the coffee is rich and smooth.</td>
</tr>
<tr>
<td>Classic latte coffee</td>
<td>A classic cup, everything is perfect.</td>
</tr>
</tbody>
</table>

Table 2: Ambiguous data of text generation for product description. For similar inputs, the outputs are ambiguous.

<table>
<thead>
<tr>
<th>Input / Product Title</th>
<th>Output / Product Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cappuccino</td>
<td>Can’t sleep? Have a cup of cappuccino.</td>
</tr>
<tr>
<td>Stir-fried rice noodles with chicken</td>
<td>Let our cuisine take you to heaven.</td>
</tr>
<tr>
<td>Roasted lamb skewers</td>
<td>Take care of your stomach.</td>
</tr>
</tbody>
</table>

Table 3: In-distribution error data of text generation for product description. These data are not suitable for online.

```
Algorithm 2 Self-Search and Remove

Input: Training dataset dataset_v1.
Output: Training dataset dataset_v2.

seed_dataset = []
model_v1 = train_on(dataset_v1)
for data ← dataset_v1 do
    model_output = model_v1(data.input)
    if exist_common_token(data.output, model_output) then
        // The in-distribution data
        seed_dataset.append(data)
    end if
end for
seed_dataset = sample(seed_dataset)
dataset_v2 = dataset_v1
for seed_data ← seed_dataset do
    model_output = model_v1(seed_data.input)
    if human_eval(model_output) == 'wrong'
        for data ← dataset_v1 do
            // Similarity search in training dataset
            if exist_most_common_tokens(seed_data, data) then
                dataset_v2.remove(data)
            end if
        end for
    end if
end for
return dataset_v2
```
them from the training dataset. Then we get the cleaned dataset_v2. In this paper, if the error data and training data have the most amount of common tokens, we identify they are the most similar.

We also tested embedding-based method for the similar search. We extract embedding from the seq2seq model’s encoder for the similarity calculation. We did not observe any significant improvement for embedding-based search.

We design the removing operation in this algorithm. Because it makes annotation errors tolerable. We observe that, removing some correct data due to annotation errors does not have a significant impact on the final result.

3 Experiment

In this section, we illustrate the dataset size, model parameters and experimental results.

3.1 Evaluation

Generation Accuracy In this paper, we do not use BLEU to evaluate the generation results. In our scenario, our goal is to determine whether the generated text is available. We use human annotation to compute:

\[ \text{Acc} = \frac{N_{\text{good}}}{N_{\text{total}}} \]

where \( N_{\text{good}} \) is the available generated text number and \( N_{\text{total}} \) is the total texts that are human annotated.

Online Dataset The online dataset is all the data in the application database. It contains 500,000,000 data. Other datasets are all sampled from this dataset. It is the dataset for the final model inference.

Evaluation Dataset We sample 5,000 data from all the 500,000,000 data online for manual evaluation for each step. We evaluate the model performance for the models of Step-2, Step-4, Step-6.

Training Dataset The initial training dataset is prepared by querying ChatGPT. Considering the resource cost, we have prepared 300,000 data.

Seed Dataset Seed dataset is sampled from the training dataset for Algorithm 2. Based on how many error data we want to retrieve, the approximate size of seed dataset can be calculated as:

\[ N_{\text{seed}} = \frac{(1 - \text{Acc}_{\text{training}}) \times N_{\text{training}}}{K_{\text{search}} (1 - \text{Acc}_{\text{training}})} \]

Then we get:

\[ N_{\text{seed}} = \frac{N_{\text{training}}}{K_{\text{search}}} \]

where \( N_{\text{seed}} \) is the seed dataset size. \( \text{Acc}_{\text{training}} \) is the generation accuracy. \( K_{\text{search}} \) is the average searched texts amount by each data of seed dataset.

Dev Dataset We split 1:20 from the training dataset as the dev dataset. The dev dataset is used to select the optimal model.

3.2 Experimental Setup

Both the T5 (Raffel et al., 2020) encoder and decoder have 8 transformer layers. The hidden size is 768 and the attention head number is 12. We compared T5 and GPT-2 (Radford et al., 2019) on the same dataset and ultimately chose T5.
### 3.3 Experimental Results

The experiment results is shown in Table 5. Our goal is to achieve the standard for online deployment, so we manually evaluate the available rate of test dataset as the evaluation metric. The T5 (Raffel et al., 2020) model trained on the initial training dataset by querying ChatGPT gets 88.0% available rate. After we use our Self-Predict and Choose method, the accuracy is improved to 95.1%. After we use our Self-Search and Remove method. The accuracy is ultimately improved to 99.2%. The Self-Predict and Choose method step improve the accuracy to 95.1%, which means we have a good foundation to perform error data removing in the next steps. Then the Self-Search and Remove method can consume fewer annotation resources.

### 4 Discussion

In this section, we discuss the motivation why we design our method and the advantage of our method.

#### 4.1 Motivation

In this sub-section we illustrate why we design the algorithms.

We design Algorithm 1 because we found a certain amount of ambiguous data in dataset_v0. There are multiple significantly different outputs for similar inputs. Therefore, using the method of Self-Predict can find these ambiguous data efficiently. We design Algorithm 2 because we found data with error labels in dataset_v1. Data with error labels have similar inputs. Also, error data have similar common patterns. Therefore, using the method of Self-Search can find the error data in dataset_v1 efficiently.

#### 4.2 Baseline Solutions

In this sub-section, we discuss other possible solutions to this problem. In summary, the essence of each method is the comparison of labeling efficiency.

**Essay Question or Choices Question** Essay question means annotators write the text answer. Writing the text answer by human without references is hard and time consuming. So in each manual annotation step, we give the annotators reference annotation results to choose from, rather than answering.

**Choose from Multiple Outputs** If we query ChatGPT and get multiple results for each input, we can manually choose the best output. The disadvantage is that it consumes multiples of the labelling time.

**Labeling Each Data By Multiple Times** To ensure the quality of the dataset, we can label each data by multiple times and get all the correct data. The disadvantage is that it also consumes multiples of the labelling time.

### 4.3 The Advantage Of Our Method

In this section we illustrate the advantage of our method, compared to the baseline methods above.

First, we want to fix the ambiguous data in initial training dataset. If we label all the training dataset to remove the ambiguous data. The labeling amount is about 10x to our method. Our Self-Predict and Choose method avoids to search all the data in the training dataset and narrow the scope to be labeled.

Second, the manually labeling error may cause wrong removing from the training dataset. In Step-5, we want to remove the error data while also overcoming manual annotation errors. So we design this algorithm that will not affect the model performance even if some right data is removed by mistake. Because we observe that some of the right data may be removed, but the error data has a greater negative impact.

### 4.4 The Pipeline Steps Order

The order of several steps in our pipeline is crucial for the result. We illustrate the reasons in this section.

We put the Step-3 before the Step-5. The reason is that we first need to correct the ambiguous data of out-of-distribution. Then we use Step-5 to remove in-distribution error data. If there are more ambiguous data in Step-3, there will be more error data in Step-5. It will consume more time to achieve the standard.
<table>
<thead>
<tr>
<th>Model Version</th>
<th>Human Evaluation Accuracy</th>
<th>Training Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChatGPT API</td>
<td>89.0%</td>
<td>-</td>
</tr>
<tr>
<td>Step-2 Model</td>
<td>88.0%</td>
<td>300,000</td>
</tr>
<tr>
<td>Step-4 Model</td>
<td>95.1%</td>
<td>300,000</td>
</tr>
<tr>
<td>Final Model</td>
<td>99.2%</td>
<td>285,000</td>
</tr>
</tbody>
</table>

Table 5: The experiment result corresponding to the Figure 1’s pipeline steps. We adopt T5 (Raffel et al., 2020) to conduct our experiments. The results contains 3 milestones: The Step-2 model is trained on the initial training dataset. Step-4 Model is trained on the dataset after the Self-Predict and Choose method, which dataset is based on the dataset of Step-2. The final model is trained on the dataset after the Self-Search and Remove method, which dataset is based on the dataset of Step-4.

5 Related Work

5.1 Transformer-based Models

The pre-trained model based on Transformer (Vaswani et al., 2017) has greatly improved the performance in various NLP tasks. The learning objectives include masked language modeling (MLM) and causal language modeling (CLM). MLM-based Language Models include BERT (Devlin et al., 2018), ROBERTA (Liu et al., 2019). CLM-based Language Models include the GPT series works (Radford et al., 2018, 2019; Brown et al., 2020) and other decoder-only transformer models (Keskar et al., 2019).

5.2 Seq2Seq Models

The sequence to sequence models (Sutskever et al., 2014) use encoder-decoder transformer (Vaswani et al., 2017) for better model flexibility. The seq2seq model is widely used in the field of text generation (Luong et al., 2014; Bahdanau et al., 2014). We adopt Seq2Seq models implement our text generation tasks. The most representative models of this type include T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). In this paper, we adopt the T5 model to conduct our experiments. We compared T5 and GPT-2 (Radford et al., 2019) on the same dataset and ultimately chose T5.

5.3 Product Description Generation

There are many works in this area. (Chen et al., 2019) adds personalized features to solve the personalized product description task. (Zhang et al., 2019) focuses on designing the pattern controlled decoder to ensure the quality of the description. (Wang et al., 2017) propose a system framework for product description generation. (Chan et al., 2019) focuses on the model-centric method to solve this problem. In our paper, we focus more on achieving the standard for online deployment by efficient human participation.

5.4 Data-Centric Method

Data-centric (Zha et al., 2023; OpenAI, 2023; Ouyang et al., 2022; Batini et al., 2009; Ratner et al., 2016) focuses a greater emphasis on enhancing the quality and quantity of the data with the model relatively fixed. Data-centric representative tasks includes data collection, data labeling, data augmentation. Data-centric AI methods are categorized into automation and collaboration depending on whether human participation is needed. Our method need human participation and focuses on the label-again way to improve the quality and quantity of dataset.

5.5 Label Error Detection

Label error detection (Wang and Mueller, 2022; Yu et al., 2023; Hendrycks and Gimpel, 2016; Yu et al., 2022; Yue and Jha, 2022; Song et al., 2022; Nataraajan et al., 2013) and confident learning (Northcutt et al., 2021; Kuan and Mueller, 2022) is the core of our method. Based on the idea of noisy data detection, we design algorithms to make the most efficient use of annotation manpower and achieve sufficient accuracy.

6 Conclusion

Product description generation’s problems for online E-commerce application is the available rate of generated text and the time consuming to annotate for training dataset. Model-centric method is limited by the quality of the training dataset. We propose our data-centric method to improve the accuracy to 99.2% to achieve the standard for online deployment. Our method also saves about 10x the annotating time.
References


Chenglin Yu, Xinsong Ma, and Weiwei Liu. 2023. Delving into noisy label detection with clean data.


