Boosting Disfluency Detection with Large Language Model as Disfluency Generator

Zhenrong Cheng
School of Intelligence Science and Technology
Peking University
Beijing, China
chengzhenrong@stu.pku.edu.cn

Jiayan Guo
School of Intelligence Science and Technology
Peking University
Beijing, China
guojiayan@pku.edu.cn

Hao Sun
School of Intelligence Science and Technology
Peking University
Beijing, China
sunhao@stu.pku.edu.cn

Yan Zhang
School of Intelligence Science and Technology
Peking University
Beijing, China
zhyzhy001@pku.edu.cn

Abstract—Current disfluency detection methods heavily rely on costly and scarce human-annotated data. To tackle this issue, some approaches employ heuristic or statistical features to generate disfluent sentences, partially improving detection performance. However, these sentences often deviate from real-life scenarios, constraining overall model enhancement. In this study, we propose a lightweight data augmentation approach for disfluency detection, utilizing the superior generative and semantic understanding capabilities of large language model (LLM) to generate disfluent sentences as augmentation data. We leverage LLM to generate diverse and more realistic sentences guided by specific prompts, without the need for fine-tuning the LLM. Subsequently, we apply an uncertainty-aware data filtering approach to improve the quality of the generated sentences, utilized in training a small detection model for improved performance. Experiments using enhanced data yielded state-of-the-art results. The results showed that using a small amount of LLM-generated enhanced data can significantly improve performance, thereby further enhancing cost-effectiveness. Our code is available here.

Index Terms—Disfluency Detection, Large Language Model, Data Augmentation

I. INTRODUCTION

Disfluencies, such as repetitions, restarts, repairs, and filled pauses, are common occurrences in human speech. Accurately detecting them is essential for various applications, including Automatic Speech Recognition(ASR) [1], Dialogue Systems [2], and Language Understanding [3]. The goal of disfluency detection is to identify the disfluencies in ASR outputs [4]. As shown in Example 1, the conventional structure of a disfluency includes the following elements: the reparandum *to Boston*, which represents the words intended for removal, the '+' signifying the end of the reparandum, the optional interregnum *um, I mean* with filled pauses or discourse markers, and the repair *to Denver* which substitutes

$$\text{a flight } [\overbrace{\text{to Boston,}}^{\text{reparandum}} + \underbrace{\{\text{um, I mean}\}}_{\text{interregnum}}, \overbrace{\text{to Denver}}^{\text{repair}}] \quad (1)$$

the original reparandum. According to the speech typology proposed by Shriberg et al. [5], disfluencies can be classified into three categories: repetition, substitution, and restart.

Approaches to disfluency detection encompass noise channel, translation-based, parser-based, and sequence labeling models, which have demonstrated reasonable performance. However, these methods are limited by the scarcity of disfluency data. Recent research has highlighted the benefits of data augmentation, such as adding simulated disfluent data and employing self-training techniques [4], [6]–[8]. The previous data augmentation methods typically relied on simple heuristic rules such as random repetition, insertion or deletion of n-gram grammar [9], or extracting statistical features from sentences to generate disfluency data [10]. These methods tend to produce simple types of disfluency sentences (a sentence has only one place or type of disfluency), and their ability to generate disfluency types related to semantics is limited, resulting in sentences lacking authenticity.

Large language models (LLM) are renowned for their robust generative capabilities and semantic understanding [11]–[13]. Recently, Marie et al. [2] propose the use of LLM for disfluency rephrasing. However, this method typically involve the fine-tuning of LLM, incurring significant costs in terms of resources and efforts. Nevertheless, the direct use of LLM in practical applications is challenging due to high memory usage and computational intensity. This has inspired our investigation into the generation of disfluent sentences without the need for fine-tuning LLM. Moreover, there are still challenges associated with using LLMs to generate disfluent sentences: First, how to design effective prompts to instruct LLM in generating disfluent sentences? Secondly, the generated disfluent sentences may exhibit poor quality, deviating from real-world scenarios or being overly simplistic.

To solve these problems, in this work, we introduce the use of LLM's powerful generation capabilities and natural

language understanding to generate disfluent sentences as augmentation data for training a small disfluency detection model. This approach distills the knowledge from LLM into small detector models, resulting in efficient and lightweight models. Unlike previous efforts in disfluency generation, our approach eliminates the necessity for fine-tuning LLM. This makes it cost-effective and highly efficient, leading to excellent results even with a small amount of generated data.

To summarize, our contributions as follows:

- We propose a framework that addresses data sparsity issues by generating disfluent data using LLM as augmentation data. This approach allows us to distill knowledge into a small model, achieving a lightweight and efficient disfluency detection model.
- We design specific prompts instead of fine-tuning LLM to generate annotated disfluent sentences. Compared to other methods, our generation approach is cost-effective, and the generated disfluent sentences are more realistic and diverse.
- To achieve better data augmentation effects, we introduce a data filtering step and conduct comprehensive experiments using the filtered data to evaluate the effectiveness of our approach.

Notably, our method achieves state-of-the-art results using a limited amount of disfluent sentences, which ensures greater efficiency and cost-effectiveness.

II. RELATED WORK

A. Disfluency Detection

The landscape of disfluency detection models falls into five main categories: sequence tagging, noisy channel, translation-based, parsing-based, and data augmentation methods. Sequence tagging treats disfluency detection as a sequence labeling task, with Lin et al. [14] employing an attention-based structure in the multi-task learning framework, and Zayats et al. [15] utilizing text-based distributional prediction of acoustic cues for vector z-score features. Rocholl et al. [8] proposed on-device lightweight BERT architectures. Lou et al. [16] introduced a new auto-correlational kernel, and Wang et al. [17] employed RNN. The noisy channel model, introduced by Lou et al. [18], treats disfluencies as errors during message encoding and decoding, then utilizes a long short-term memory neural network language model to rescore candidate disfluency analyses. Translation-based approaches formulate disfluency detection as encoder-decoder systems. Dong et al. [19] adapted the neural machine translation model, and Wang et al. [20] extended the traditional encoder-decoder model by leveraging two independent encoders. In the parsing-based approach, as explored by Sen et al. [21], semantic parsing is employed to identify disfluency. Tran et al. [22] utilized acoustic-prosodic features extracted from the text. To address data scarcity and reduce reliance on gold-standard annotated data, researchers have proposed data augmentation techniques. Rocholl et al. [8] emploted Wikipedia+Books for model pretraining.
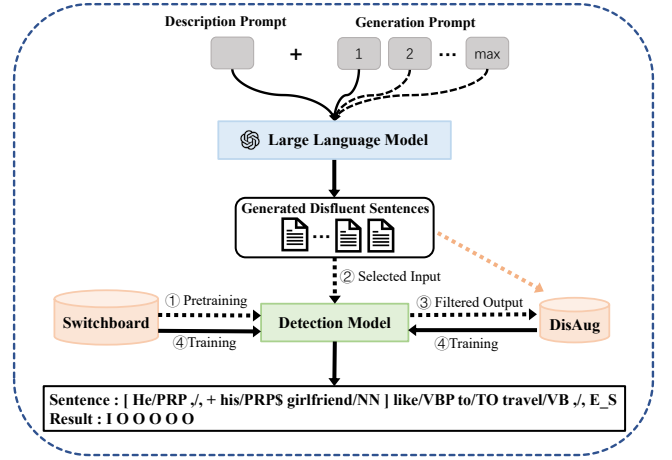

Fig. 1: An Overview of Our Approach: For disfluency generation, we begin by providing a description prompt to the LLM and guide its generation of disfluent sentences through a series of generation prompts. Subsequently, we use the trainable detection model to filter out high-quality sentences after pretraining on Switchboard, namely DisAug. For disfluency detection, DisAug is incorporated with Switchboard as training data to train the original detection model.

B. Disfluency Generation

Passali et al. [9] used heuristic rules to generate disfluenct sentences. Gupta et al. [23] built DISFL-QA, a benchmark dataset of contextual disfluency in QA through manual annotation. Wang et al. [7] and Wang et al. [4] generated disfluenct sentences by randomly repeatting, injecting and deleting n-grams from unlabeled news data as additional training data. However, their approaches are admittedly generating less natural disfluent sentences than with a neural model. Yang et al. [10] introduced the PGmodel, a state-of-the-art data augmentation model, to generate disfluency, in which a Planner determines the appropriate placement of disfluent segments, and a Generator creates the corresponding disfluent segments. Marie et al. [2] proposed a disfluency paraphraser, utilizing LLM to rephrase fluent sentences into disfluent ones. While their approaches can generate natural disfluent sentences, but it requires training an additional generation model, leading to higher costs for fine-tuning LLM.

III. METHODOLOGY

In this section, we first define the problem of disfluency detection and introduce the detection model. Next, we describe the prompt designs guiding the Language Model (LLM) to generate disfluent sentences as augmented data, as well as the data filters used to select high-quality generated sentences. The overall framework is illustrated in Fig. 1.

A. Task Definition

We consider the disfluency detection task as a sequence labeling task. The model takes as input a dataset containing

N sentences $\{S_1, S_2, ..., S_N\}$, where the i-th sentence is represented as $s_i = \{w_1, w_2, ..., w_T\}$, with T being the number of words in that sentence. Our objective is to output a label sequence for each sentence, denoted as $Y_i = \{y_1, y_2, ..., y_T\}$, where $I(O)$ indicates whether the word is inside (or outside) the reparandum region.

B. Disfluency Generation

1) Data Generation: The LLM demonstrates a profound comprehension of context and semantics, highlighting its remarkable ability to generate coherent and fluent sentences. However, when it comes to generating sentences with disfluencies, an area where LLM faces challenges, it cannot be directly instructed as in the case of generating fluent sentences tasks. Our goal is to enable the LLM to generate disfluent sentences using a simple and efficient approach. Therefore, we propose a Description-then-Generation disfluency generation method. First, we employ a data description prompt for the LLM. This prompt conveys data descriptions as instructions, enabling the LLM to understand the intended data for generation, along with its concepts and features. Next, we input generation prompts into the LLM. This further enhances the LLM's understanding of the desired generated data, facilitating the creation of diverse disfluent sentences. Given the LLM's extensive contextual understanding, we set the maximum number of generation round as *max round*, requiring the reintroduction of the Description Prompt after every *max round* rounds of generation.

> ▍ ***Description Prompt.***
>
> The following is the description of disfluent sentence:
> 1. Labeling Format
> Filled pause: {F ...}; Explicit editing term: {E ...};
> Discourse marker: {D ...}; Coordinating conjunction:
> {C ...}; Aside: {A ...}.
> 2. Part-of-speech Tags Format
> POS tagging assigns grammatical labels to words in a sentence, such as NN for nouns, VB for verbs, JJ for adjectives, RB for adverbs, PRP for pronouns, IN for prepositions, CC for conjunctions, DT for determiners, UH for interjections, and CD for numbers.
> 3. Disfluency Types
> Repetition: certain words or phrases are repeated; Substitution: replacement of the preceding word or phrase; Deletion: removal of specific words; Chaining: continuity where something from each part carries over into the next; Nesting: a restart within a restart.
> 4. These disfluency types can combine to form more complex sentences. The '[ ]' denotes the disfluent part, and the '+' symbol indicates the end of reparandum.

> ▍ ***Generation Prompt.***
>
> These example sentences contain part-of-speech tags and disfluency annotation. Please generate more

sentences which are similar with example sentences. The generated sentences also have part-of-speech tags and disfluency annotation. Just give me answers, no more tips words and don't copy the example sentences. Note: generated sentences need to belong various disfluency types such as restart, delete, substitution. Two or Three Example sentences: (Providing examples of disfluent sentence)

Algorithm 1 Disfluent Sentence Generation Algorithm

Input: Prompt 1: Description prompt; Prompt 2: Generation prompt;
Output: Generated disfluent sentences S_{dis};
1: set $i = 0$, Maximum number of generation round max_round ;
2: Input Prompt 1 to the LLM;
3: **while** $i \leq max\_round$ **do**
4:      Input Prompt 2 to the LLM;
5:      Get generated sentences-> S_{dis}
6:      $i + +$;
7: **end while**
8: Return S_{dis}

It is worth noting that the example sentences in the generation prompt not only contain disfluency annotations but also include part-of-speech tags. This inclusion enables the LLM to better discern the various types of disfluent words and disfluent grammatical structures. We can specify the types of disfluencies present in the example sentences to control the disfluency types generated in the resulting sentences. Besides, the inclusion of example sentences contributes to making the sentences generated by the LLM more human-like.

2) Uncertainty-aware Data Filtering: Considering that LLM tends to bias towards generating fluent sentences, leading to a decline in the quality of disfluent sentences, and aiming to achieve higher performance gains with minimal augmented data, inspired by graph collaborative filtering [24], we introduce an uncertainty-aware data filtering. This process identifies high-confidence disfluent sentences and excludes those potentially detrimental to model performance.

The uncertainty measure is computed by averaging the predicted probabilities of words labeled as disfluent. The higher the predicted probability for disfluent labels, the lower the uncertainty of the disfluent sentence. Therefore, a value closer to 1 indicates higher sentence quality. The computation formula is presented in Eq. 2, where p_j represents the predicted probability of the j-th word being disfluent, and y_j is the label associated with the j-th word, indicating whether it is disfluent (labeled as 'I'). We set the hyperparameter λ as the minimum threshold. If the computed average predicted probability falls below this predefined threshold, it indicates that the sentence is considered to have high uncertainty in its disfluent portions. Therefore, such sentences will be filtered out or excluded from further consideration.

$$\frac{1}{K} \sum_{j=1}^{K} p_j, \forall p_j \vDash y_j \in \{I\} \tag{2}$$

3) Model Agnostic Disfluency Detection: There are many models available for sequence labeling tasks of disfluency

TABLE I: Generated Disfluent Sentences. The data annotation includes part-of-speech tagging and disfluency labeling. The underlined part represents the disfluent portion in the original sentence, and the boldface represents the disfluent portion in the generated sentence.

| Original Sentence	Generated Disfluent Sentence
Heuristic Method	
Sometimes/RB ,/, {F uh/UH ,/, } it/PRP 's/BES a/DT bit/NN of/IN [a/DT ,/, + a/DT] problem/NN ,/, {D you/PRP know/VBP ,/, } because/IN I/PRP guess/VBP I/PRP do/VBP n't/RB really/RB manage/VB my/PRP$ money/NN the/DT way/NN I/PRP should/MD ./. E_S	At/IN times/NNS ,/, {F uh/UH ,/, } [**it/PRP ,/, + it/PRP**] can/MD be/VB [**a/DT little/JJ ,/, + a/DT little/JJ**]problematic/JJ ,/, {D you/PRP know/VBP ,/, } because/IN I/PRP suppose/VBP I/PRP do/VBP n't/RB quite/RB manage/VB my/PRP$ money/NN as/IN effectively/RBas/IN I/PRP should/MD ./. E_S
{C and/CC } they/PRP are/VBP beginning/VBG to/TO be/VB a/DT budget/NN problem/NN but/CC ,/, {F uh/UH ,/, } [have/VBP not/RB been/VBN really/RB up/IN until/IN this/DT ,/, + up/IN to/IN this/DT] point/NN ./. E_S	{C and/CC } they/PRP are/VBP becoming/VBG a/DT budget/NN issue/NN but/CC ,/, {F uh/UH ,/, } have/VBP n't/RB really/RB [**caught/VBN up/IN until/IN this/DT ,/, + caught/VBN up/IN to/IN this/DT**] point/NN ./. E_S
Now/UH I/PRP know/VBP my/PRP$ boss/NN has/VBZ bought/VBN the/DT software/NN ,/, {F um/UH ,/, } [that/WDT he/PRP can/MD ,/, + that/WDT his/PRP$] checkbook/NN is/VBZ [on/IN ,/, + on/IN] + on/IN] a/DT disk/NN ,/, E_S	Now/UH [**I/PRP realize/VBP ,/, + {F um/UH ,/, } I/PRP understand/VBP**] that/DT my/PRP$ boss/NN has/VBZ procured/VBN the/DT software/NN ,/, [**indicating/VBG that/WDT he/PRP can/MD ,/, + allowing/VBG him/PRP$ to/TO**]store/VB his/PRP$ checkbook/NN details/NNS on/IN a/DT disk/NN ,/, E_S
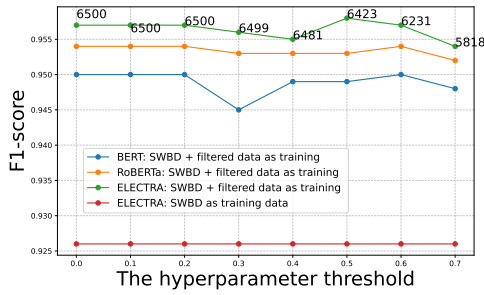
Fig. 2: F1-scorce under different confidence thresholds. The values on the line represent the quantity of generated sentences filtered at that threshold.

detection. We aim to use data augmentation to enable simple models to perform better. Therefore, we directly chose trainable language models as detectors for pretraining and fine-tuning. Specifically, we obtain the hidden state h_i for each word w_i, and then calculate the label probabilities p_i for each word, as shown in Eq. 3.

$$H = h_1, h_2, ..., h_T = \text{Detector}(w_1, w_2, ..., w_T)$$
$$p_i = \text{softmax}(Wh_i + b) \quad (3)$$

The objective of this model is to minimize the cross-entropy loss:

$$L = \mathbb{E}_{s,l} \sum_{i=1}^{T} \text{CrossEntropy}(l_i, p_i) \quad (4)$$

where l_i and p_t represent the true label predicted probability distribution of the i-th word respectively. $\mathbb{E}_{s,l}$ represents the expected value of all possible labeling sequences.

IV. EXPERIMENTS

A. Experiment Setup

Dataset. We choose Switchboard (SWBD) [25], the standard and largest disfluency detection benchmark dataset for evaluating the proposed method. Following the experiment setup outlined in Charniak and Johnson [26], we partitioned the Switchboard corpus into training, development, and test

TABLE II: Percentage of generated disfluency sentences.

Generation Model	Repetition	Substitution	Deletion
PG [10]	85.68%	13.08%	1.24%
PG-NC-AD-ID [10]	19.80%	25.27%	54.93%
DisAug	34.57%	44.10%	21.32%

sets. The training set comprises 173k sentences, the development set includes 10k sentences, and the test set encompasses 7.9k sentences.

Baselines. Transformers [7] and Transformers & BERT [7] employed heuristic methods to generate disfluent data for pretraining, while BERT-CRF [27] and ELECTRA-CRF [27] adopted a multitask approach. BERT-GCN [28] leveraged structural information for span-level detection, which is currently the SOTA method for the disfluency detection task. And we compare the PGmodel [10] (i.e., BERT-GRI3M model and BERT-GRI20M model) and other models that performed disfluency detection using different data augmentation techniques. Marie et al. [2] fine-tuned T5 to rewrite fluent sentences as disfluent ones, generating data for training a disfluency classifier. As the generated data has not been disclosed, we employ the disfluency classifier as the detector for experimental comparisons. In addition, we employ three disfluency detectors, namely bert-base-uncased [29], roberta-base [30], and google/electra-base-discriminator [31], using their fine-tuning outcomes on SWBD as baselines for comparison. In order to comprehensively examine the effect of the data enhancement we generated, we also use DISFL-QA [23] as the augmented data for experimental comparison.

Implementation Details. We use OpenAI's text-davinci-003 language model as the disfluency generation model. For disfluency detection, we utilize the Adam optimizer with a learning rate of 1e-5 and batch size 32 during the pretraining and fine-tuning stages. And, we uniformly employ the ELECTRA model fine-tuned on SWBD for data filtering, with the confidence threshold hyperparameter λ set to 0.5.

B. Experimental Results

1) Disfluency Generation Results: We generated 6.5K original disfluent sentences, and then pretrained the ELECTRA model using SWBD as a data filter, resulting in 6423 selected sentences, which we named 'DisAug'. Table I illustrates the comparison between DisAug and sentences generated by other methods. DisAug serves as our augmented data for disfluency detection. In Table II, we showcase the percentage distribution of three types of disfluencies observed in the generated dataset: repetition, substitution, and deletion. PG [10] and PG-NC-AD-ID [10] refer to two different generation methods employed in the PG model. It's worth noting that many sentences in DisAug exhibit intricate patterns. In comparison to disfluent sentences generated through heuristic-based and PG model methods, sentences in DisAug often manifest more than one type of disfluency and tend to feature complex patterns of substitution.

Figure 2 shows the number of disfluent sentences filtered by different label confidence thresholds and the results as

TABLE III: Results of disfluency detection. The †mark denotes that the results are significant with the significance level $p < 0.05$. F1-score is the major metric.

Baselines				
Model	**Setting**	**Precision**	**Recall**	**F1-score**
Transformers [7]	multitask & 3M pretraining	0.934	0.873	0.902
Transformers & BERT [7]	multitask & 3M pretraining	-	-	0.914
BERT-CRF [27]	multitask	0.946	0.912	0.929
ELECTRA-CRF [27]	multitask	0.948	0.916	0.931
BERT-GCN [28]	Span Classification	<u>0.952</u>	<u>0.932</u>	<u>0.942</u>
BERT-GRI3M(PGmodel) [10]	3M pretraining	0.951	0.894	0.922
BERT-GRI20M(PGmodel) [10]	20M pretraining	0.945	0.902	0.923
Disfluency Classifier		0.944	0.842	0.890
BERT	SWBD as training data (173K)	0.949	0.867	0.906
RoBERTa		0.946	0.883	0.913
ELECTRA		0.939	0.914	0.926
ELECTRA	SWBD + DISFL-QA (7.1K) as training data	0.940	0.906	0.923

Ours				
Model	**Setting**	**Precision**	**Recall**	**F1-score**
Disfluency Classifier	DisAug + FluData as training data (8K)	0.944	0.880	0.742
	DisAug pretraining	0.934	0.856	0.893
	SWBD + Original Data as training data	**0.956**	0.925	0.940
	SWBD + DisAug (6.4K) as training data	0.943	**0.944**	**0.944**
BERT	DisAug + FluData as training data (8K)	0.903	0.920	0.912
	DisAug pretraining	0.954	0.895	0.923
	SWBD + Original Data as training data	0.954	**0.947**	**0.950**
	SWBD + DisAug (6.4K) as training data	0.956	0.942	0.949
RoBERTa	DisAug + FluData as training data (8K)	0.947	0.882	0.914
	DisAug pretraining	0.949	0.894	0.920
	SWBD + Original Data as training data	**0.957**	0.949	0.953
	SWBD + DisAug (6.4K) as training data	0.947	**0.961**	**0.954**
ELECTRA	DisAug + FluData as training data (8K)	0.918	0.945	0.931
	DisAug pretraining	0.947	0.927	0.937
	SWBD + Original Data as training data	0.955	0.959	0.957
	SWBD + DisAug (6.4K) as training data	**0.955**	**0.960**	**0.958†**

augmented data to enhance disfluency detection. We can observe that the enhancement effect on disfluency detection shows little fluctuation between different filtered generated sentences. This suggests that the sentences generated using our method exhibit stability and efficiency, thereby improving the model's performance in disfluency detection while reducing the workload for data generation.

2) Disfluency Detection Results: To further validate the effectiveness of the generated disfluent data and improve the performance of disfluency detection, we designed four setups of experiments: (1) Combing DisAug with FluData (fluent sentences generated by LLM) as the training set for fine-tuning the model; (2) Pretraining the language model using DisAug and then fine-tuning it on the SWBD training set; (3) Adding original generated data to the SWBD training set as training data for fine-tuning the model; (4) Adding DisAug to the SWBD training set as training data for fine-tuning the model. In these four experimental setups, the dev
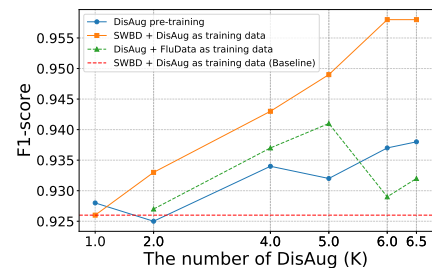

Fig. 3: Results of Disfluency Detection with ELECTRA as a Detector under Different Experimental Setups.

set and test set remained consistent and unchanged. The main experimental results are presented in Table III. Figure 3 illustrates the variation in the F1-score of the ELECTRA detector for disfluency detection when different quantities of generated data are employed across various experiments.

From the experimental results, we observed a significant improvement in the model's performance with the generated augmented data. Incorporating DisAug into SWBD for training the ELECTRA model achieved SOTA results. Compared to the manually annotated DISFL-QA dataset, our approach has achieved superior results, demonstrating the efficiency and cost-effectiveness of our method. We exceeded the majority of baselines by using 8K generated sentences as training data, effectively reducing training costs. Additionally, by adjusting the proportion of generated disfluent and fluent sentences, we can further enhance the disfluency detection performance. The original data and DisAug yielded competitive results, which indirectly verified the effectiveness of our data generation method.

V. CONCLUSION

In this study, we propose a method that guides LLM with specific prompts to generate more diverse and natural disfluent sentences, thereby enhancing disfluency detection. We also implement a data filtering process to improve data quality. Experimental results consistently affirm the efficiency and cost-effectiveness of our approach, demonstrating its effectiveness with limited data resources. In the future, we plan to explore the integration of speaker characteristics to deploy our method in online environments. We will consider extending its application to other tasks, such as disfluent paraphrasing.

REFERENCES

[1] Hayato Futami, Emiru Tsunoo, Kentaro Shibata, Yosuke Kashiwagi, Takao Okuda, Siddhant Arora, and Shinji Watanabe, "Streaming joint speech recognition and disfluency detection," in *ICASSP 2023-2023*. IEEE, 2023, pp. 1–5.

[2] Benjamin Marie, "Disfluency generation for more robust dialogue systems," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 11479–11488.

[3] Tanvi Dinkar, Chloé Clavel, and Ioana Vasilescu, "Fillers in spoken language understanding: Computational and psycholinguistic perspectives," *arXiv preprint arXiv:2301.10761*, 2023.

[4] Shaolei Wang, Zhongyuan Wang, Wanxiang Che, and Ting Liu, "Combining self-training and self-supervised learning for unsupervised disfluency detection," in *EMNLP*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, Eds., Online, Nov. 2020, pp. 1813–1822, Association for Computational Linguistics.

[5] Elizabeth Shriberg, Rebecca A. Bates, and Andreas Stolcke, "A prosody only decision-tree model for disfluency detection," in *EUROSPEECH*, 1997.

[6] Paria Jamshid Lou and Mark Johnson, "Improving disfluency detection by self-training a self-attentive model," in *ACL*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, Eds., Online, July 2020, pp. 3754–3763, Association for Computational Linguistics.

[7] Shaolei Wang, Wangxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang, "Multi-task self-supervised learning for disfluency detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 9193–9200.

[8] Johann C Rocholl, Vicky Zayats, Daniel D Walker, Noah B Murad, Aaron Schneider, and Daniel J Liebling, "Disfluency detection with unlabeled data and small bert models," *arXiv preprint arXiv:2104.10769*, 2021.

[9] Tatiana Passali, Thanassis Mavropoulos, Grigorios Tsoumakas, Georgios Meditskos, and Stefanos Vrochidis, "LARD: Large-scale artificial disfluency generation," in *LREC*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis, Eds., Marseille, France, June 2022, pp. 2327–2336, European Language Resources Association.

[10] Jingfeng Yang, Diyi Yang, and Zhaoran Ma, "Planning and generating natural and diverse disfluent texts as augmentation for disfluency detection," in *EMNLP*, 2020, pp. 1450–1460.

[11] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al., "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[12] Hao Sun, Hengyi Cai, Bo Wang, Yingyan Hou, Xiaochi Wei, Shuaiqiang Wang, Yan Zhang, and Dawei Yin, "Towards verifiable text generation with evolving memory and self-reflection," *arXiv preprint arXiv:2312.09075*, 2023.

[13] Hao Sun, Xiao Liu, Yeyun Gong, Yan Zhang, and Nan Duan, "Allies: Prompting large language model with beam search," in *Findings of EMNLP 2023*, 2023.

[14] Binghuai Lin and Liyuan Wang, "Joint prediction of punctuation and disfluency in speech transcripts.," in *Interspeech*, 2020, pp. 716–720.

[15] Vicky Zayats and Mari Ostendorf, "Giving attention to the unexpected: Using prosody innovations in disfluency detection," *arXiv preprint arXiv:1904.04388*, 2019.

[16] Paria Jamshid Lou, Peter Anderson, and Mark Johnson, "Disfluency detection using auto-correlational neural networks," in *EMNLP*. 2018, pp. 4610–4619, Association for Computational Linguistics.

[17] Shaolei Wang, Wanxiang Che, and Ting Liu, "A neural attention model for disfluency detection," in *COLING*, 2016, pp. 278–287.

[18] Paria Jamshid Lou and Mark Johnson, "Disfluency detection using a noisy channel model and a deep neural language model," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada, July 2017, pp. 547–553, Association for Computational Linguistics.

[19] Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu, "Adapting translation models for transcript disfluency detection," in *AAAI*, 2019, vol. 33, pp. 6351–6358.

[20] Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu, "Semi-supervised disfluency detection," in *COLING*, 2018, pp. 3529–3538.

[21] Priyanka Sen and Isabel Groves, "Semantic parsing of disfluent speech," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, Eds., Online, Apr. 2021, pp. 1748–1753, Association for Computational Linguistics.

[22] Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf, "Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information," in *NAACL-HLT 2018*, pp. 69–81.

[23] Aditya Gupta, Jiacheng Xu, Shyam Upadhyay, Diyi Yang, and Manaal Faruqui, "Disfl-qa: A benchmark dataset for understanding disfluencies in question answering," *arXiv preprint arXiv:2106.04016*, 2021.

[24] Jiayan Guo, Lun Du, Xu Chen, Xiaojun Ma, Qiang Fu, Shi Han, Dongmei Zhang, and Yan Zhang, "On manipulating signals of user-item graph: A jacobi polynomial-based graph collaborative filtering," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 602–613.

[25] John J Godfrey, Edward C Holliman, and Jane McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, speech, and signal processing, ieee international conference on*. IEEE Computer Society, 1992, vol. 1, pp. 517–520.

[26] Eugene Charniak and Mark Johnson, "Edit detection and parsing for transcribed speech," in *NAACL*, 2001.

[27] Dongyub Lee, Byeongil Ko, Myeong Cheol Shin, Taesun Whang, Daniel Lee, Eun Hwa Kim, EungGyun Kim, and Jaechoon Jo, "Auxiliary sequence labeling tasks for disfluency detection," in *ICASSP 2020*.

[28] Sreyan Ghosh, Sonal Kumar, Yaman Kumar Singla, Rajiv Ratn Shah, and Srinivasan Umesh, "Span classification with structured information for disfluency detection in spoken utterances," *arXiv*, 2022.

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

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[31] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.