Planning and Generating Natural and Diverse Disfluent Texts as Augmentation for Disfluency Detection

Jingfeng Yang, Diyi Yang, Zhaoran Ma
Georgia Institute of Technology
{jyang690, makima, dyang888}@gatech.edu

Abstract

Existing approaches to disfluency detection heavily depend on human-annotated data. Numbers of data augmentation methods have been proposed to alleviate the dependence on labeled data. However, current augmentation approaches such as random insertion or repetition fail to resemble training corpus well and usually resulted in unnatural and limited types of disfluencies. In this work, we propose a simple Planner-Generator based disfluency generation model to generate natural and diverse disfluent texts as augmented data, where the Planner decides on where to insert disfluent segments and the Generator follows the prediction to generate corresponding disfluent segments. We further utilize this augmented data for pretraining and leverage it for the task of disfluency detection. Experiments demonstrated that our two-stage disfluency generation model outperforms existing baselines; those disfluent sentences generated significantly aided the task of disfluency detection and led to state-of-the-art performance on Switchboard corpus. We have publicly released our code at https://github.com/GT-SALT/Disfluency-Generation-and-Detection.

1 Introduction

Disfluency is a para-linguistic concept defining the interruption to the flow of speech (Kowal, 2009). As shown in Figure 1, a standard annotation of the disfluency structure indicates the reparandum (the region to repair), an optional interregnum (filled pauses, discourse cue words, etc.) and the associated repair (corrected linguistic materials) (Nakatani and Hirschberg, 1994). Disfluency detection (Lou et al., 2018; Wang et al., 2019) mainly deals with identifying and removing reparandum[1], since interregnum can be easily detected in that they belong to a closed set of words and phrases, e.g. "*uh*" "*I mean*" "*you know*" etc. The output fluent sentences from disfluency detection can serve as clean inputs for most downstream NLP tasks, like dialogue systems, question answering, and machine translation (Wang et al., 2010).

I want a flight [*to Boston* + {*um*} to Denver]
reparandum interregnum repair

Figure 1: Example of an annotated disfluent sentence.

Type	Example
Repetition	~~they~~ they learn to share.
Deletion	this ~~is~~ just happened yesterday.
Substitution	it's nothing but wood ~~up here~~ down here.

Table 1: Different types of reparandum.

Reparandum in disfluency can be categorized as repetition, deletion and substitution (McDougall and Duckworth, 2017), as shown in Table 1. Repetition occurs when linguistic materials repeat, usually in form of partial words, words or short phrases. Substitution occurs when linguistic materials are replaced in order to clarify a concept or idea. Deletion, also known as false restart, refers to abandoned linguistics materials.

Neural models have achieved reasonable performance in disfluency detection on English Switchboard (SWBD) corpus (Godfrey et al., 1992). Such models involve applying pretrained models to conduct disfluency detection as sequence tagging or seq2seq tasks (Wang et al., 2019, 2018). Recently, data augmentation techniques are also used to generate augmented disfluent sentences for model pretraining. Those models are limited in that the augmented data is generated based on simple heuristics such as random repetition or insertion of ngrams (Wang et al., 2019, 2018). Sentences generated

[1]We use reparandum and disfluent segments interchangeably in this paper.

METHODS	EXAMPLE
Random repetition	(1) ~~that 's~~ that 's really good.
Random insertion	(2) ~~of a~~ that 's really good.
Generation (ours)	(3) ~~it 's~~ that 's really good.
Generation (ours)	(4) that would ~~be more~~ be worth doing.

Table 2: Disfluent text generated from random repetition, insertion, and our Disfluency Generation model

by such methods do not resemble natural disfluent sentences and have different distribution of disfluency patterns from original sentences in SWBD dataset. For example, in Table 2, the random insertion of "*of a*" to the example (2) "*of a that's really good*" results in a quite unnatural disfluent text, not representative of our disfluency corpus nor most commonly used practices. As a result, most sentences generated by random insertion are inefficient as augmented samples, and even lead models to deviate from SWBD dataset. This is also supported by Gontijo-Lopes et al. (2020) that suggested the augmented data should have high affinity to the original dataset. Furthermore, based on our small corpus studies, we observed that current disfluency detection models usually struggle with substitution and deletion based disfluencies, largely due to the under-representation of substitution and deletion in current training corpus. Disfluent sentences generated by random repetition or insertion rarely contain substitutions, and are thus inadequate in introducing diverse forms of disfluency.

To address this gap, we propose a generation based data augmentation method to generate natural and diverse disfluent sentences to further improve the performance of disfluency detection. This method is similar to back-translation (Edunov et al., 2018), which has been shown effective as a way of data augmentation in machine translation. Different from classical neural generation models, our generation model is in a two-stage generation manner, motivated by coarse-to-fine decoding (Dong and Lapata, 2018). Specifically, given a fluent sentence as the input, in the first stage, a *Planner* selects the positions of where to insert reparandum; in the second stage, a *Generator* generates disfluent segments accordingly for the predicted areas. Compared to generic end-to-end generation, our two-stage model separates the generation task into two steps: where to generate and what to generate. Such breakdown enables the model to only generate disfluent segments rather than the whole disfluent sentences, and to carry naturally labeled data as augmentation for disfluency detection. As shown in Table 2, the outputs (3) and (4) from our two-stage Planner-Generator model resemble natural disfluent sentences better than random insertion (2), and also introduces more substitutions (example (3) and (4)). We then utilize this Planner-Generator disfluency generation to create augmented training data for the task of disfluency detection. As an additional benefit, the disfluent texts generated by our generation model can be used as inputs of text-to-speech (TTS) systems to generate more natural speech, thus improving the performance of tasks like automatic film dubbing, robotics, dialogue systems, or speech-to-speech translation, as shown by Betz et al. (2015) and Adell et al. (2006).

To sum up, our contributions are as follows:

- We design a simple two-stage Planner-Generator generation model to generate disfluent texts, and demonstrate its effectiveness over various generation baseline models.
- We utilize our generation model to generate natural and diverse augmented disfluent data for the task of disfluency detection, and obtain state-of-the-art performance. We conduct thorough error analysis and discuss specific challenges faced by current approaches.

2 Related Work

Disfluency Generation Betz et al. (2015) and Adell et al. (2006) used heuristic rules to generate filled pauses, repetitions in disfluent speech generation. Their works demonstrate that disfluency generation enhances the naturalness and intelligibility of speech generated by text-to-speech (TTS) systems. Wang et al. (2018) and Wang et al. (2019) randomly inserted or repeated ngrams to generate augmented disfluent sentences. Disfluent sentences generated in this method have low affinity to disfluent sentences from the benchmark dataset, and they contain few substitutions, causing limited diversity. To achieve better affinity and diversity, we design generation based data augmentation which generates natural disfluent sentences that can then be directly used to train the disfluency detection model. We adapt multi-stage coarse-to-fine neural decoders (Dong and Lapata, 2018) for generation tasks to design our disfluency generation model.

Disfluency Detection Disfluency detection models mainly fall into four categories. The first one utilizes noisy channel models (Zwarts and Johnson, 2011; Lou and Johnson, 2018), which require Tree

Adjoining Grammar (TAG) based transducer in the channel model. The second category leverages phrase structure, which is often related to transition-based parsing yet requires annotated syntactic structure (Rasooli and Tetreault, 2013; Yoshikawa et al., 2016; Wu et al., 2015; Jamshid Lou and Johnson, 2020). The third category frames the task as a sequence tagging task (Ferguson et al., 2015; Hough and Schlangen, 2015; Zayats et al., 2016; Lou and Johnson, 2018; Wang et al., 2019), and the last one employs end-to-end Encoder-Decoder models (Wang et al., 2016, 2018) to detect disfluent segments automatically. Traditional disfluency detection models often required additional features (e.g. POS tags) (Wang et al., 2017), syntactic annotations or external tools (e.g. TAG based transducer) (Lou and Johnson, 2018) for learning. Recent disfluency detection approaches leveraged neural representations and obtained comparable results. For instance, Lou et al. (2018) adopted CNN and introduced the Auto-Correlation Operator which models more accurate word relations and similarities in order to capture "rough copies". However, most of them still heavily depend on human-annotated data. As a result, different kinds of data augmentation approaches and pretraining have been designed to alleviate such dependence. For example, Bach and Huang (2019) incorporated ELMo (Peters et al., 2018) to sequence tagging model and Dong et al. (2019) used a pretrained denoising autoencoder to initialize the encoder-decoder model. Wang et al. (2019) achieved state-of-the-art performance by using data augmentation and BERT (Devlin et al., 2018) in a sequence tagging task, and Wang et al. (2018) obtained similar performance by using the same data augmentation methods in an encoder-decoder fashion. These aforementioned data-augmentation methods created augmented disfluent sentences only by randomly inserting or repeating ngrams. To this end, we introduce generation based data augmentation to first generate disfluencies and then use them for sequence tagging of disfluency detection. Note that there was a similar trend in grammatical error detection. Felice and Yuan (2014); Kasewa et al. (2018) generated sentences with grammatical errors to augment the training data of grammatical error detection.

3 Method

3.1 Disfluency Generation

Our goal is to generate a natural disfluent sentence from a fluent sentence. For this purpose, we introduced a Planner and Generator based model, as shown in Figure 2, which is described as follows.

Let $\boldsymbol{x} = x_1, x_2, \cdots, x_{|x|}$ denote a fluent sentence, $\boldsymbol{y} = y_1, y_2, \ldots, y_{|y|}$ denote the corresponding disfluent sentence. We estimated $p(\boldsymbol{y}|\boldsymbol{x})$ via a two stage generation process:

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{a})p(\boldsymbol{a}|\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{a} = a_1, a_2, \ldots, a_{|a|}$ is a decision sequence with the same length as $\boldsymbol{x}$. a_i is either 1 or 0, which represents whether a disfluent segment (reparandum) should be added after x_i or not. We assumed a_i are independent of each other and further decomposed our objective as follows:

$$p(\boldsymbol{a}|\boldsymbol{x}) = \prod_i p(a_i|\boldsymbol{x}) \tag{2}$$

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{a}) = \prod_j p(y_j|y_{<j}, \boldsymbol{x}, \boldsymbol{a}) \tag{3}$$

Planner At the first *Planning* stage, we used an encoder e_1 to obtain representations of $\boldsymbol{x}$:

$$\boldsymbol{h} = h_1, h_2, \cdots, h_{|x|} = f_{e_1}(x_1, x_2, \cdots, x_{|x|}) \tag{4}$$

Then we used $\boldsymbol{h}$ to get the decision probability a_i:

$$p(a_i|\boldsymbol{x}) = \text{softmax}(W_1 h_i + b_1) \tag{5}$$

Generator: Encoder We used another encoder e_2 to get the representations of $\boldsymbol{x}$ as the conditional state of the second stage:

$$\hat{\boldsymbol{h}} = \hat{h}_1, \hat{h}_2, \cdots, \hat{h}_{|x|} = f_{e_2}(x_1, x_2, \cdots, x_{|x|}) \tag{6}$$

Encoder e_1 and e_2 can be Bidirectional LSTM or Transformer (Vaswani et al., 2017).

Generator: Decoder $p(y_j|y_{<j}, \boldsymbol{x}, \boldsymbol{a})$ was computed based on the output ($\bar{h}_j$) of the corresponding step of decoder. As shown in Figure 2, in our **PG** model, the input z_j of j-th step of decoder is determined by the value of corresponding a_i (E is the embedding layer of decoder):

$$z_j = \begin{cases} h_i & \text{if } a_i = 0 \text{ or } (a_i = 1 \text{ and} \\ & y_j \text{ is the first word of reparandum}) \\ E(y_{j-1}) & \text{otherwise,} \end{cases} \tag{7}$$

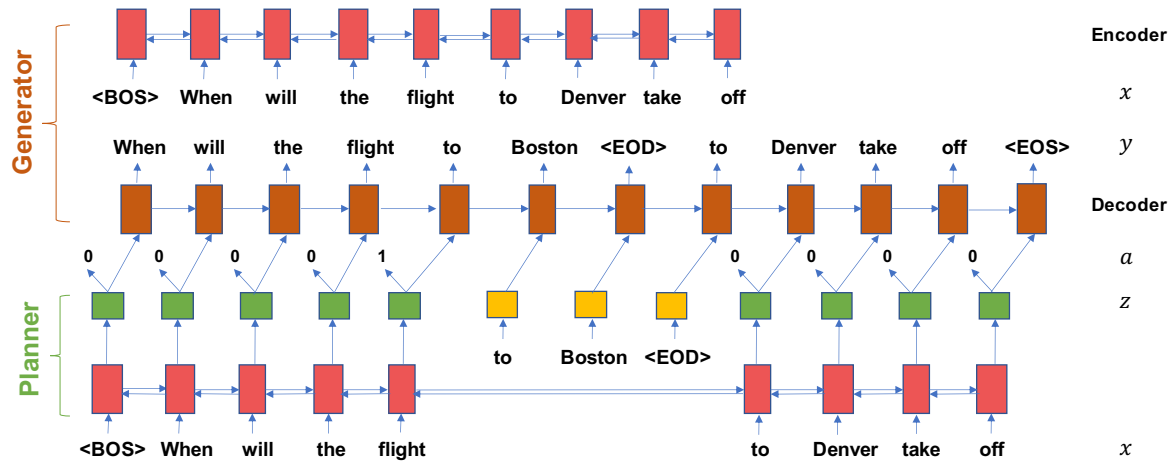

Figure 2: Our two-stage disfluency generation model with Planner and Generator (PG model).

Alternatively, to make the model focus less on local contexts for less copied words, we can use a decoder with less connection with Planner **(PG-LC)**:

$$z_j = \begin{cases} E(x_i) & \text{if } a_i = 0 \\ h_i & \text{if } a_i = 1 \text{ and} \\ & y_j \text{ is the first word of reparandum} \\ E(y_{j-1}) & \text{otherwise,} \end{cases} \tag{8}$$

We can also use a decoder with no connection with Planner **(PG-NC)**, where we separate Generator from Planner and only use the decision sequence to guide generation. This modification is the basis of the models with higher generation diversity:

$$z_j = \begin{cases} E(x_i) & \text{if } a_i = 0 \text{ or } (a_i = 1 \text{ and} \\ & y_j \text{ is the first word of reparandum}) \\ E(y_{j-1}) & \text{otherwise,} \end{cases} \tag{9}$$

We used LSTM as the decoder, and the decoder's hidden vector at the j-th time step is computed by

$$\bar{h}_j = f_{\text{LSTM}}(\bar{h}_{j-1}, z_j) \tag{10}$$

where $\bar{h}_0 = \hat{h}_{|x|}$ if we use the last hidden state of encoder to initialize the first state of decoder; $\bar{h}_0 = \mathbf{0}$ if we do not use such initialization **(ID)**, decreasing the decoder's dependence on the encoder for high diversity of generated disfluent segments.

Based on encoder's hidden vectors $\hat{\boldsymbol{h}}$ and decoder's hidden vectors $\bar{\boldsymbol{h}}$, we used attention and copying mechanism to compute $p(y_j|y_{<j}, \boldsymbol{x}, \boldsymbol{a})$, similarly to See et al. (2017). Alternatively, we also computed it without attention **(AD)** or copying mechanism **(CD)** for high diversity of the generated reparandum. The decoder can also be replaced with Transformer or GPT2 (Radford et al., 2019).

Training and Inference The training objective is to maximize the log likelihood of the disfluent sentence given the fluent sentence:

$$\max \sum_{(\boldsymbol{x},\boldsymbol{a},\boldsymbol{y})\in\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{a}) + \log p(\boldsymbol{a}|\boldsymbol{x}), \tag{11}$$

here $\mathcal{D}$ represents all training pairs.

During inference stage, Planner chose 0 or 1 with higher probability in each step to generate the decision sequence $\boldsymbol{a}$. Alternatively, the Planner can also be an oracle Planner, whose predictions are gold decision sequences for the purpose of higher accuracy, or a heuristic Planner, whose predictions are selected according to simple heuristics for higher diversity in data augmentation. When generating the final disfluent sentence $\boldsymbol{y}$, assume y_j is generated based on a_i. If $a_i = 0$, we directly copy x_{i+1} as y_j; if $a_i = 1$, the Generator generates a sequence of words as reparandum before copying x_{i+1}.

3.2 Disfluency Detection

We regarded the disfluency detection task as a sequence tagging task. We denoted i-th sentence with T words as $s_i = \{w_t|t = 1, \ldots, T\}$, the input of our model is $\{s_1, s_2, \ldots, s_N\}$, where N is the number of sentences in the dataset. The corresponding output is $\{q_1, q_2, \ldots, q_N\}$, where q_i is the label sequence of i-th sentence, $q_i = \{l_t|t = 1, \ldots, T\}$. $l_t \in \{I, O\}$, where I (O) represents that the word is in (or outside) the region of reperandum.

3.2.1 Heuristic based Data Augmentation

Pretraining the model on augmented data has been proved as effective (Wang et al., 2019, 2018) before training the model on the SWBD dataset. Note that, compared to multi-task learning and using Sentence Pair Classification Task as an auxiliary task by Wang et al. (2019), our study mainly focuses on sequence tagging pretraining. To generate an augmented disfluent sentence for any fluent sentence, we followed the augmentation method in Wang et al. (2019). First we used the heuristic of randomly choosing one to three positions in a fluent sentence. Then, for each position k:

- *Insertion* : we randomly picked a m-gram (m is randomly selected from one to six) from the news corpus and inserted it to the position k.
- *Repetition* : m (the length of repeated words, randomly selected from one to six) words starting from the position k were repeated.

3.2.2 Generation based Data Augmentation

These augmented sentences generated from *Insertion* are often not natural, since those inserted m-grams are randomly picked from the whole corpus which may be irrelevant to the current sentence. This creates large discrepancies between the distribution of augmented sentences and the original corpus, and further hinders the effectiveness of augmented data. To introduce more natural and diverse generated disfluencies, we introduced this generation based data augmentation mode:

- *Generation*: we used our PG-based model to generate reparandum starting from position k.

3.2.3 Sequence Tagging

For the sequence tagging model, instead of using Transformer or the combination of trainable Transformer and frozen BERT as Wang et al. (2019) did, we directly adopted trainable BERT for both pretraining and fine tuning. First we got the probability of labels of each word:

$$\begin{aligned}\{h_1, h_2, \ldots, h_T\} &= \text{BERT}(\{w_1, w_2, \ldots, w_T\}) \\ p_t &= \text{softmax}(W h_t + b)\end{aligned} \tag{12}$$

Eventually, the goal of the model is to minimize the objective, the cross-entropy (CE) loss:

$$L = \mathbb{E}_{(s,l)} \sum_{t=1}^{T} \text{CE}(l_t, p_t) \tag{13}$$

4 Experiments and Results

4.1 Dataset

For disfluency detection, we used English Switchboard Dataset. Similar to Charniak and Johnson (2001), we split the dataset to training set sw23[⋆].dps, development set sw4[5-9][⋆].dps, and test set sw4[0-1][⋆].dps. Following Hough and Schlangen (2015), we lower-cased the text and removed all punctuation and partial words. For disfluency generation, all sentences with reparandum were treated as disfluent sentences. Specifically, our training set contains 29k disfluent sentences out of 173k sentences. In development set, 2k sentences in a total of 10k sentences are disfluent sentences. In test set, 1.6k sentences out of 7.9k sentences are disfluent sentences.

4.2 Evaluation

To measure whether generated disfluent sentences are natural, we compared them with reference disfluent sentences based on two generation related metrics: BLEU (Papineni et al., 2002) and Sentence Accuracy, i.e. the percentage of the generated sentences that exactly match the ground-truth disfluent sentences.

Furthermore, we evaluated the naturalness of model outputs according to human judgment. Due to budget issue, we only selected the model (PG-NC-AD-ID) and the baseline (*Insertion* & *Repetition*) with the highest diversity based on automatic measures. For those two models, we randomly selected 100 generated disfluent sentences and they were assessed on Amazon Mechanical Turk. We elicited 3 responses per HIT. For each sentence, *Natural* was marked with a score of one, *Unnatural* sentences with 0.5, and zero for *Incomprehensible* ones. Average Human-evaluated Naturalness (HN) score thus ranged from 0 (worst) to 1 (best).

We also designed metrics to measure the diversity of disfluent segments, similarly to Li et al. (2015). Specifically, we calculated the number of new unigrams and bigrams in the generated disfluent segments. The value was scaled by the total number of generated tokens in the disfluent segments (shown as Diverse-1 and Diverse-2 in Table 3). To evaluate disfluency detection, we used standard metrics: Precision, Recall and F-score.

4.3 Training Details

For disfluency detection, we used BERT-base-uncased (Wolf et al., 2019). In both pretraining

	Without Oracle Decision					With Oracle Decision			
	BLEU	Sent Acc	Diverse-1	Diverse-2	HN	BLEU	Sent Acc	Diverse-1	Diverse-2
Simple Copy	0.8023	0	-	-	-	-	-	-	-
Seq2Seq (Luong et al., 2015)	0.7244	0.1036	-	-	-	-	-	-	-
CopyNet (Gu et al., 2016)	0.8037	0.1390	-	-	-	-	-	-	-
BART (Lewis et al., 2019)	0.8050	0.0644	-	-	-	-	-	-	-
Insertion & Repetition	0.5792	0.0006	47.34%	43.04%	0.6517	-	-	-	-
PG-Transformer (ours)	0.8176	0.1308	2.63%	0.79%	-	0.8662	0.3519	0.82%	4.02%
PG (ours)	0.8173	**0.1428**	1.34%	2.16%	-	**0.8727**	**0.3765**	0.85%	3.37%
PG-LC (ours)	**0.8177**	0.1421	1.70%	0.51%	-	0.8684	0.3588	1.74%	0.48%
PG-NC (ours)	0.8155	0.0992	4.15%	1.71%	-	0.7144	0.1642	3.34%	7.81%
PG-CD (ours)	**0.8297**	0.1346	1.50%	1.82%	-	0.8716	0.3481	1.40%	3.57%
PG-NC-CD (ours)	0.8179	0.1030	3.47%	13.35%	-	0.7579	0.1598	7.19%	36.73%
PG-NC-AD (ours)	0.8178	0.1061	4.38%	8.64%	-	0.7738	0.1819	9.63%	34.26%
PG-NC-AD-ID (ours)	0.7925	0.0310	**61.04%**	**52.06%**	**0.7642**	0.747	0.0499	**64.85%**	**69.75%**

Table 3: Disfluency generation results in terms of BLEU, Sentence Accuracy (Sent Acc), Human-evaluated Naturalness (HN), Diverse-1 and Diverse-2. We show the performances of all variants of our PG-based models.

	Insertion	*Repetition*	*Generation*	Total
BERT-RI3	1.5M	1.5M	0M	3M
BERT-G3	0M	0M	3M	3M
BERT-GR3	0M	1.5M	1.5M	3M
BERT-GRI3	1M	1M	1M	3M
BERT-GRI20	6.6M	6.7M	6.7M	20M

Table 4: Composition of augmented data for each model.

and fine-tuning stages, we used Adam optimizer with learning rate 1e-5 and batch size 32. For disfluency generation, we trained LSTM with learning rate 1e-2 and Transformer with learning rate 1e-4.

4.4 Models and Baselines

For pretraining, we followed Wang et al. (2019) to use WMT2017 monolingual language model training data as unlabeled data. The data augmentation methods in Section 3.2 were used to generate augmented disfluent sentences. Wang et al. (2019) used 3 million sentences in the sequence tagging task and 9 million sentence pairs in the sentence classification task. We used 3 million sentences for fair comparison and also experimented with 20 million sentences to examine the effect of data size. In Table 4, we show the composition of augmented sentences in all of our models. Note that Wang et al. (2018) and Bach and Huang (2019) treated interregnum and reparandum types equally as disfluent segments when training and evaluating their models, while others in Table 5 only focused on reparandum which is more difficult to detect. Bach and Huang (2019) used a different way of splitting training and development set, whose training set had more data. Given the different setups, we did not compare with Wang et al. (2018) and Bach and Huang (2019).

For disfluency generation, we applied various combinations of our model settings described in 3.1. In **PG-Transformer**, encoders and decoders are all Transformer, while all the other models use LSTM. Planner-Generator (**PG**), PG with less Planner-Generator connection (**PG-LC**), and PG with no Planner-Generator connection (**PG-NC**) are models that generate relatively natural disfluent sentences (high BLEU and Sent Acc). For higher diversity, **PG-CD** is PG without copying mechanism. Likewise, **PG-NC-CD** is PG-NC without copying mechanism, while **PG-NC-AD** is PG-NC without attention mechanism. In the extreme case, **PG-NC-AD-ID** is PG-NC without attention mechanism and encoder-Initialized decoder for high diversity. We used Simple Copy (directly copy input as output), Random *Insertion & Repetition* of ngrams, LSTM and Attention based Seq2Seq model, CopyNet and pretrained BART as baselines.

Since our models enable the control of generating reparandum based on any given decision sequences, we examined their performances with and without oracle decision sequences, i.e. the positions of the reparandum in generated sentences are the same as the references.

In order to use our model to generate diverse disfluent sentences, we experimented with different variants of PG and found PG-NC-AD-ID produced better performances. Thus we finally chose PG-NC-AD-ID and the heuristic Planner applying the position choosing heuristic described in Section 3.2.1, since the model-based Planner always chose certain most probable positions, and generated less diverse disfluent sentences, which did not work well as augmented data GPT2 was used to replace the LSTM decoder and trained on a partial pretraining dataset to alleviate the domain gap.

	Setting	Precision	Recall	F-score
LSTM (Zayats et al., 2016)	seq-tagging & ad-hoc features	0.878	0.711	0.786
semi-CRF (Ferguson et al., 2015)	seq-tagging & ad-hoc features	0.900	0.812	0.854
Bi-LSTM (Zayats et al., 2016)	seq-tagging & ad-hoc features	0.918	0.806	0.859
LSTM-NCM (Lou and Johnson, 2018)	seq-tagging	-	-	0.868
Transition-based (Wang et al., 2017)	parsing & ad-hoc features	0.911	0.841	0.875
NMT (Dong et al., 2019)	sequence to sequence & denoising	0.945	0.841	0.890
Transformer (Wang et al., 2019)	seq-tagging & multitask & 3M pretraining	0.934	0.873	0.902
Transformer & BERT (Wang et al., 2019)	seq-tagging & multitask & 3M pretraining	-	-	0.914
BERT (ours)	seq-tagging	0.949	0.867	0.906
BERT-G3 (ours)	seq-tagging & 3M pretraining	0.946	0.878	0.911
BERT-RI3 (ours)	seq-tagging & 3M pretraining	0.951	0.881	0.915
BERT-GR3 (ours)	seq-tagging & 3M pretraining	0.946	0.890	0.917
BERT-GRI3 (ours)	seq-tagging & 3M pretraining	**0.951**	**0.894**	**0.922**†
BERT-GRI20 (ours)	seq-tagging & 20M pretraining	0.945	**0.902**	**0.923**

Table 5: Results of disfluency detection. F-score is the major metric. "ours" represents our implementations. The mark † denotes that the results are significant with the significance level $p < 0.05$. Specifically, p-value is 0.0003 comparing BERT-GRI3 and BERT. p-value is 0.0259 comparing and BERT-GRI3 and BERT-RI3.

4.5 Disfluency Generation Result

Table 3 shows the disfluency generation results. Despite its relatively high diversity, *Insertion & Repetition* baseline had a low BLEU score and an almost zero Sent Acc, which indicates that disfluent sentences generated in such manners are neither natural nor similar to real disfluency distributions in SWBD dataset. Simple Copy baseline maintained high BLEU yet failed to generate any disfluent sentences with zero Sent Acc. Other neural baselines were able to achieve reasonable BLEU and Sent Acc. However, their results could not serve as augmented data to pretrain sequence tagging models, since there was no indication where were the disfluent segments in output sentences.

All of our proposed PG-based models outperformed *Insertion & Repetition* in terms of BLEU and Sent Acc, which shows that our generated results were closer to natural disfluent sentences than random *Insertion & Repetition* of ngrams were. Among our models, PG-Transformer, PG and PG-LC generated the most natural disfluent sentences, leading to the highest BLEU and Sent Acc. Our LSTM-based models PG and PG-LC outperformed all of the baselines in terms of Sent Acc and BLEU, despite that PG-Transformer was slightly overshadowed by CopyNet in terms of Sent Acc. The performance boost of our PG based models mainly came from our two-stage Planner-Generator process, since the hidden states of the first stage were used as initial input to guide the generation of reparandum in the second stage.

We found that without copying mechanism , PG-CD model harmed Sent Acc but would not drastically decrease BLEU compared with PG. Without the state passing between Planner and Generator Decoder, PG-NC severely harmed Sent Acc as well as BLEU, while it improved the generation diversity. Without copying mechanism and state passing (PG-NC-CD), the diversity boosted significantly. This demonstrates that copying mechanism and state passing between Planner and Generator Decoder together forced the model to generate repetitions. The deletion of those two mechanisms were responsible for increased substitutions and deletions and decreased repetitions in results, leading to a higher diversity of disfluent sentences.

Without the attention between Generator Encoder and Generator Decoder, PG-NC-AD had little improvement in diversity compared to PG-NC-CD. However, when deleting the mechanism of using the last state of Generator Encoder as the initial state of Generator Decoder (PG-NC-AD-ID), diversity increased substantially. This made the Generator Decoder an unconditional language model trained on the dataset. Although the PG-NC-AD-ID model decreased BLEU and Sent Acc, it still generated more natural disfluent sentences than *Insertion & Repetition*, as demonstrated by higher automatic evaluation metrics (BLEU, Sent Acc) and human evaluation metric (HN). Considering that PG-NC-AD-ID outperformed *Insertion & Repetition* in all metrics, we used this model to generate diverse and natural augmented disfluent sentences for disfluency detection. As we expected, with oracle decision sequences, nearly all models achieved significantly better BLEU and Sent Acc.

4.6 Disfluency Detection Result

We used the above generation based augmented data to further improve disfluency detection. The

	Repetition	Substitution	Deletion
Insertion & Repetition	46.64%	13.14%	40.22%
PG	85.68%	13.08%	1.24%
PG-NC-AD-ID	19.80%	25.27%	54.93%

Table 6: Different types of generated disfluecies.

	Repetition	Substitution	Deletion
BERT	0.64%	8.51%	2.74%
BERT-RI3	0.35%	8.83%	2.68%
BERT-GR3	0.40%	8.10%	2.95%
BERT-GRI3	0.40%	7.60%	2.60%

Table 7: The percentage of false negative errors in reference test set of disfluency detection (lower is better).

results are shown in Table 5. We found that BERT without pretraining already achieved competitive results. BERT-G3 performed better than BERT, showing the effectiveness of our generation based data augmentation. Our BERT-RI3 performed similarly to Wang et al. (2019), although we did not use Sentence Pair Classification Task as an auxiliary task during pretraining. The reason might be that we fine-tuned the BERT model during both pretraining and SWBD training, while Wang et al. (2019) trained a Transformer during these stages and combined it with a fixed BERT when training on SWBD. Overall, when using *Repetition* & *Insertion* to do data augmentation, BERT-RI3 performed better than BERT.

After replacing *Insertion* with *Generation*, BERT-GR3 outperformed BERT-RI3 and Wang et al. (2019). When adding *Generation* upon *Repetition* and *Insertion*, BERT-GRI3 achieved even better performance, a new state-of-the-art performance. We also did significance test with Bootstrap (Berg-Kirkpatrick et al., 2012), BERT-GRI3 significantly outperformed BERT-RI3 and BERT with significance level p=0.0259 and p=0.0003 respectively. This not only demonstrated the effectiveness of our disfluency generation based data augmentation, but also showed that disfluencies generated by our generation model are orthogonal to those generated by *Insertion* & *Repetition*. A comparison between the precision and recall of BERT-GRI3 and BERT revealed that the improvements of pretraining mainly come from its higher recall, indicating that pretraining can help the model to detect more disfluencies while obtaining similar accuracies. When pretraining data size was increased, BERT-GRI20 did not significantly outperform BERT-GRI3.

Impact of Augmented Disfluency Types: We summarized different types of generated disfluencies in Table 6 to show how our model contributed to disfluency detection. *Insertion* & *Repetition* generated limited substitutions, which caused a lack of natural and diverse disfluencies. Although our PG model achieved state-of-the-art performance in terms of BLEU and Sent Acc, it mainly generated repetitions, leading to low diversity. This was potentially caused by two factors. First, the disfluent segments in the training dataset were dominated by 65.39% repetitions, in comparison to 18.99% substitutions and 15.62% deletions. Second, copying words and phrases during generation for neural models proved to be the most convenient and consistent approach, even without copying mechanism. Our PG-NC-AD-ID model generated more substitutions and deletions compared with PG, leading to the highest diversity. Compared to random *Insertion* & *Repetition*, it also generated substantially more substitutions, leading to a more effective data augmentation. The decreased number of repetitions can be fixed by combining it with random repetition, like BERT-GR3. Table 7 presents the proportion that was not identified by our disfluency detection models among all repetitions, substitutions and deletions in reference test set. Comparing our generation based augmentation (BERT-GRI3 and BERT-GR3) with other methods (BERT-RI3 and BERT), we found that pretraining on our generated data can reduce substitution errors and improve the final metric recall in Table 5, contributed by increased natural substitutions generated by our disfluency generation model.

Type	Count	Percentage
Noisy Annotation	112	24.83%
Substitution (False Negative)	103	22.84%
Deletion (False Negative)	90	19.96%
Repetition (False Negative)	45	9.98%
Ambiguous	38	8.43%
False Positive	21	4.66%
Other	42	9.31%

Table 8: Challenge types in disfluency detection. Ambiguous are cases where annotations and predictions are both correct.

4.7 Error Analysis and Challenges

We manually annotated the errors made by our disfluency detection model case by case, and presented a thorough error analysis in terms of different types of errors in Table 8. Note that nearly one fourth "wrong" predictions were in fact correct. These mismatches were caused by improper annotation. For example, the sentence "*the thing*

is is that's not enough" was annotated as a fluent sentence, while the first "is" should be reparandum. Similar noisy annotation issues in the SWBD dataset were a major hindrance to achieving higher performance. With respect to other errors, we saw much more false negatives than false positives. Among false negatives, errors were dominated by substitutions and deletions, although the proportion of repetitions (65.39%) was much more than substitutions (18.99%) and deletions (15.62%) in the original SWBD dataset. This showed that current models do relatively well in identifying repetitions, while detecting substitutions and deletions is still challenging for the model.

5 Conclusion

This work presents a simple two-stage disfluency generation model to generate natural and diverse disfluent texts. We further used them as augmented data for pretraining and aiding the task of disfluency detection. Experiments demonstrate that our proposed disfluency generation model outperformed existing baselines; those disfluent sentences generated significantly aided the task of disfluency detection and led to state-of-the-art performances.

Acknowledgments

We thank the members of Georgia Tech SALT group for their feedback on this work. We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPU used for this research. Diyi Yang is supported in part by a grant from Google. We also thank Nihal Singh and Jinge Yao for their discussion.

References

Jordi Adell, Antonio Bonafonte, David Escudero, and D Informatics. 2006. Disfluent speech analysis and synthesis: a preliminary approach. In *in Proc. of 3th International Conference on Speech Prosody*. Citeseer.

Nguyen Bach and Fei Huang. 2019. Noisy bilstm-based models for disfluency detection. *Proc. Interspeech 2019*, pages 4230–4234.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Simon Betz, Petra Wagner, and David Schlangen. 2015. Micro-structure of disfluencies: Basics for conversational speech synthesis. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

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

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu. 2019. Adapting translation models for transcript disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6351–6358.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520.

Raphael Gontijo-Lopes, Sylvia J Smullin, Ekin D Cubuk, and Ethan Dyer. 2020. Affinity and diversity: Quantifying mechanisms of data augmentation. *arXiv preprint arXiv:2002.08973*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. *Interspeech 2015*.

Paria Jamshid Lou and Mark Johnson. 2020. Improving disfluency detection by self-training a self-attentive model. *arXiv*, pages arXiv–2004.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. *arXiv preprint arXiv:1810.00668*.

Sabine Kowal. 2009. *Communicating with one another: Toward a psychology of spontaneous spoken discourse*. Springer Science & Business Media.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency detection using auto-correlational neural networks. *arXiv preprint arXiv:1808.09092*.

Paria Jamshid Lou and Mark Johnson. 2018. Disfluency detection using a noisy channel model and a deep neural language model. *arXiv preprint arXiv:1808.09091*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Kirsty McDougall and Martin Duckworth. 2017. Profiling fluency: An analysis of individual variation in disfluencies in adult males. *Speech Communication*, 95:16–27.

Christine H Nakatani and Julia Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. *The Journal of the Acoustical Society of America*, 95(3):1603–1616.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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

Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 124–129.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. 2018. Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3529–3538.

Shaolei Wang, Wanxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2019. Multi-task self-supervised learning for disfluency detection. *arXiv preprint arXiv:1908.05378*.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2785–2794.

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. Automatic disfluency removal for improving spoken language translation. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5214–5217. IEEE.

Thomas Wolf, L Debut, V Sanh, J Chaumond, C Delangue, A Moi, P Cistac, T Rault, R Louf, M Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771*.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 495–503.

Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1036–1041.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 703–711. Association for Computational Linguistics.

A Model Parameters

A.1 Disfluency Detection

In disfluency detection, we used BERT-base-uncased (Wolf et al., 2019) for sequence tagging. In both pretraining and fine-tuning stages, we used Adam optimizer with learning rate 1e-5 (searched from [1e-4, 1e-5]) and batch size 32. Hyper-parameters were searched manually according to F-score. We ran 20 epochs for pretraining and ran 20 epochs for training on SWBD dataset. After each epoch, we decayed the learning rate by 0.985.

A.2 Disfluency Generation

For disfluency generation, we used one-layer LSTM with hidden size 300 and a dropout of 0.5 (searched from [0.1, 0.3, 0.5]), and then trained it with learning rate 1e-2 (searched from [1e-2, 1e-3]). Alternatively, we used Transformer with 512 hidden units in attention layer, 2048 hidden units in feed-forward layer, 8 heads, 6 hidden layers, GELU activation (Hendrycks and Gimpel, 2016), and a dropout of 0.1 (searched from [0.1, 0.3, 0.5]), and then trained it with learning rate 1e-4 (searched from [1e-3, 1e-4, 1e-5]). Hyper-parameters were searched manually according to Sent Acc. As we expected, the performance of Transformer is more sensitive to learning rate than LSTM. We used Adam optimizer and batch size 64 for both of them. After each epoch, we decayed the learning rate by 0.985. We trained them for 30 epochs. When we used GPT2 (Wolf et al., 2019) as decoder, it was trained on another 3M WMT2017 mono-lingual language model training data for 10 epochs.

B Computational Requirements

We ran our models on GeForce RTX 2080 GPU. Each disfluency generation model required 1 hour to finish training (GPT2 and Transformer required 4 hours). Each disfluency detection model required 2 hours to finish training on SWBD data. Pre-training disfluency detection models on 3M data required 5 days on 1 GPU. Pretraining models on 20M data required 7 days on 4 GPUs.

C Evaluation Metrics

As for metrics, we used NLTK to compute BLEU. Other metrics are computed by our scripts written according to descriptions in the paper. Metrics on validation sets were close to those reported on test sets for all experiments.

As for Human-evaluated Naturalness on AMT, we provided the description and example sentences of three levels of disfluent sentences (incomprehensible, unnatural and natural). For example, "Natural disfluent sentence" is "Perfectly natural speech. Similar to the talk you could probably have with someone in life." To improve annotation quality, annotators should have >5000 HITs approved, >98% HIT Approval Rate and located in the US. We also require annotators to pass a qualification test consisting of samples with expected answers before they work on the annotation, to make sure that they have a good understanding of our task. Annotators are paid $0.08 for annotating each sentence, and each sentence was rated by three workers.

D Dataset

SWBD dataset is a part of PDTB and WMT2017 mono-lingual language model training data can be downloaded from News Crawl: articles from 2016.