Multi-Task Self-Supervised Learning for Disfluency Detection

Shaolei Wang,[1] Wanxiang Che,[1]* Qi Liu,[2] Pengda Qin,[3] Ting Liu,[1] William Yang Wang[4]
[1]Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
[2]University of Oxford
[3]Beijing University of Posts and Telecommunications, China
[4]University of California, Santa Barbara, CA, USA
{slwang, car, tliu}@ir.hit.edu.cn, qi.liu@st-hughs.ox.ac.uk, qinpengda@bupt.edu.cn, william@cs.ucsb.edu

Abstract

Most existing approaches to disfluency detection heavily rely on human-annotated data, which is expensive to obtain in practice. To tackle the training data bottleneck, we investigate methods for combining multiple self-supervised tasks- i.e., supervised tasks where data can be collected without manual labeling. First, we construct large-scale pseudo training data by randomly adding or deleting words from unlabeled news data, and propose two self-supervised pre-training tasks: (i) tagging task to detect the added noisy words. (ii) sentence classification to distinguish original sentences from grammatically-incorrect sentences. We then combine these two tasks to jointly train a network. The pre-trained network is then fine-tuned using human-annotated disfluency detection training data. Experimental results on the commonly used English Switchboard test set show that our approach can achieve competitive performance compared to the previous systems (trained using the full dataset) by using less than 1% (1000 sentences) of the training data. Our method trained on the full dataset significantly outperforms previous methods, reducing the error by 21% on English Switchboard.

Introduction

Automatic speech recognition (ASR) outputs often contain various disfluencies, which create barriers to subsequent text processing tasks like parsing, machine translation, and summarization. Disfluency detection (Zayats, Ostendorf, and Hajishirzi 2016; Wang, Che, and Liu 2016; Wu et al. 2015) focuses on recognizing the disfluencies from ASR outputs. As shown in Figure 1, a standard annotation of the disfluency structure indicates the reparandum (words that the speaker intends to discard), the interruption point (denoted as '+', marking the end of the reparandum), an optional interregnum (filled pauses, discourse cue words, etc.) and the associated repair (Shriberg 1994).

Ignoring the interregnum, disfluencies are categorized into three types: restarts, repetitions and corrections. Table 1 gives a few examples. Interregnums are relatively easier to detect as they are often fixed phrases, e.g. "uh", "you know". On the other hand, reparandums are more difficult to detect in that they are in free form. As a result, most previous disfluency detection work focuses on detecting reparandums.

I want a flight [$\underbrace{\textit{to Boston}}_{\text{RM}}$ + $\underbrace{\{\textit{um}\}}_{\text{IM}}$ $\underbrace{\text{to Denver}}_{\text{RP}}$]

Figure 1: A sentence from the English Switchboard corpus with disfluencies annotated. RM=Reparandum, IM=Interregnum, RP=Repair. The preceding RM is corrected by the following RP.

Type	Annotation
repair	[I just + I] enjoy working
repair	[we want + {well} in our area we want] to
repetition	[it's + {uh} it's] almost like
restart	[we would like +] let's go to the

Table 1: Different types of disfluencies.

Most work (Zayats and Ostendorf 2018; Lou and Johnson 2017; Wang et al. 2017; Jamshid Lou, Anderson, and Johnson 2018; Zayats and Ostendorf 2019) on disfluency detection heavily relies on human-annotated data, which is scarce and expensive to obtain in practice. In this paper, we investigate self-supervised learning method (Agrawal, Carreira, and Malik 2015; Fernando et al. 2017) to tackle this training data bottleneck. Self-supervised learning aims to train a network on auxiliary tasks where ground-truth is obtained automatically. The merits of this line work are that they do not require manually annotations but still utilize supervised learning by inferring supervisory signals from the data structure. Neural networks pre-trained with these tasks can be fine-tuned to perform well on standard supervised task with less manually-labeled data than networks which are initialized randomly. In natural language processing domain, self-supervised research mainly focus on word embedding (Mikolov et al. 2013a; 2013b) or language model learning (Bengio et al. 2003; Peters et al. 2018; Radford et al. 2018; Devlin et al. 2019). Motivated by the success of self-supervised learning, we propose two self-supervised tasks for disfluency detection task, as shown in

*Corresponding Author

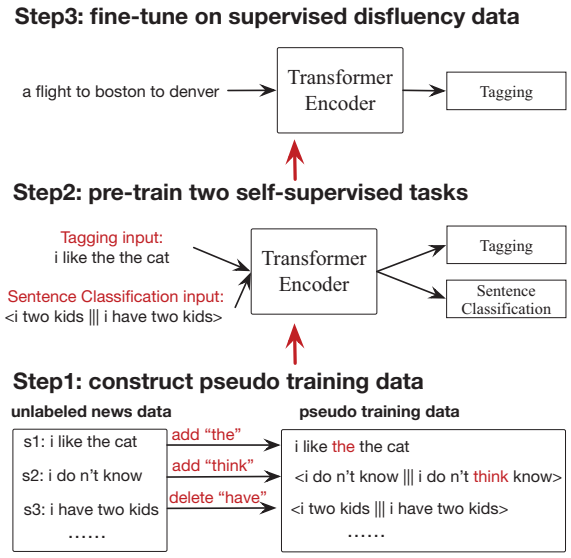

Figure 2: Illustration of our proposed methods.

Figure 2.

The first task aims to tag corrupted parts from a disfluent sentence, generated by randomly adding words to a fluent sentence. Although there are discrepancies between the distribution of gold disfluency detection data and the generated sentences, this task endows the model to recover the fluent sentences from the disfluent ones, which matches the final goal of disfluency detection.

The second task is sentence classification to distinguish original sentences from corrupted sentences. We align each original sentence from the news dataset and another disfluent sentence generated by randomly deleting or adding words to the original fluent sentence. The goal of the task is to take these sentence pairs as input and predict which sentence is the fluent one. This task enables the model to distinguish grammatically-correct sentences from grammatically-incorrect ones. We hypothesize that this task is helpful for disfluency detection, as one core challenge for disfluency detection is to keep the output sentences grammatically-correct.

The second task can help the first by modeling sentence-level grammatical information. Inspired by the hypothesis, we combine these two tasks to jointly train a network based on the auto-constructed pseudo training data. The pre-trained network is later fine-tuned using human-annotated disfluency detection data.

Our contributions can be summarized as follows:

- We propose two self-supervised tasks for disfluency detection to tackle the training data bottleneck. To our best knowledge, this is the first work to investigate self-supervised representation learning in disfluency detection.
- Based on the two self-supervised tasks, we further investigate multi-task methods for combining the two self-supervised tasks.
- Experimental results on the commonly used English Switchboard test set show that our approach can achieve competitive performance compared to the previous systems (trained using the full dataset) by using less than 1% (1000 sentences) of the training data. Our method trained on the full dataset significantly outperforms previous methods, reducing the error by 21% on English Switchboard.

Related Work

Disfluency Detection

Most work on disfluency detection focus on supervised learning methods, which mainly fall into three main categories: sequence tagging, noisy-channel, and parsing-based approaches. Sequence tagging approaches label words as fluent or disfluent using a variety of different techniques, including conditional random fields (CRF) (Georgila 2009; Ostendorf and Hahn 2013; Zayats, Ostendorf, and Hajishirzi 2014) , Max-Margin Markov Networks (M^3N) (Qian and Liu 2013), Semi- Markov CRF (Ferguson, Durrett, and Klein 2015), and recurrent neural networks (Hough and Schlangen 2015; Zayats, Ostendorf, and Hajishirzi 2016; Wang, Che, and Liu 2016). Noisy channel models (Charniak and Johnson 2001; Johnson and Charniak 2004; Zwarts, Johnson, and Dale 2010; Lou and Johnson 2017) use the similarity between reparandum and repair as an indicator of disfluency. Parsing-based approaches (Rasooli and Tetreault 2013; Honnibal and Johnson 2014; Wu et al. 2015; Yoshikawa, Shindo, and Matsumoto 2016) jointly perform dependency parsing and disfluency detection. The joint models can capture long-range dependency of disfluencies as well as chunk-level information.

There exist a limited effort to tackle the training data bottleneck. Wang et al. (2018) and Dong et al. (2019) use an autoencoder method to help for disfluency detection by jointly training the autoencoder model and disfluency detection model. They construct large-scale pseudo disfluent sentences by using some simple rules and use autoencoder to reconstruct the disfluent sentence. We take inspiration from their method when generating disfluent sentences. They achieve higher performance by introducing pseudo training sentence. However, the performance of their method still heavily relies on annotated data.

Self-Supervised Representation Learning

Self-supervised learning aims to train a network on an auxiliary task where ground-truth is obtained automatically. Over the last few years, many self-supervised tasks have been introduced in image processing domain, which make use of non-visual signals, intrinsically correlated to the image, as a form to supervise visual feature learning (Agrawal, Carreira, and Malik 2015; Wang and Gupta 2015).

In natural language processing domain, self-supervised research mainly focus on word embedding (Mikolov et al. 2013a; 2013b) and language model learning (Bengio et al. 2003; Peters et al. 2018; Radford et al. 2018). For word embedding learning, the idea is to train a model that maps each word to a feature vector, such that it is easy to predict the

words in the context given the vector. This converts an apparently unsupervised problem into a "self-supervised" one: learning a function from a given word to the words surrounding it.

Language model pre-training (Bengio et al. 2003; Peters et al. 2018; Radford et al. 2018; Devlin et al. 2019) is another line of self-supervised learning task. A trained language model learns a function to predict the likelihood of occurrence of a word based on the surrounding sequence of words used in the text. There are mainly two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based approach, such as ELMo (Peters et al. 2018), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al. 2018) and BERT (Devlin et al. 2019), introduces minimal task-specific parameters and is trained on the downstream tasks by simply fine-tuning the pre-trained parameters.

Liu et al. (2016) propose a method to automatically generate large-scale pseudo training data for zero pronoun resolution, and propose a two-step training mechanism to overcome the gap between the pseudo training data and the real one.

Motivated by the success of self-supervised learning, we propose a token-level tagging task and a sentence-level classification task especially powerful for disfluency detection task.

Multi-Task Learning

MTL (Multi-Task Learning) has been used for a variety of NLP tasks including named entity recognition and semantic labeling (Martínez Alonso and Plank 2017), super-tagging and chunking (Bingel and Søgaard 2017) and semantic dependency parsing (Peng, Thomson, and Smith 2017). The benefits of MTL largely depend on the properties of the tasks at hand, such as the skewness of the data distribution (Martínez Alonso and Plank 2017), the learning pattern of the auxiliary and main tasks where "target tasks that quickly plateau" benefit most from "non-plateauing auxiliary tasks" (Bingel and Søgaard 2017) and the "structural similarity" between the tasks (Peng, Thomson, and Smith 2017). In our work, we use the sentence classification task to help the tagging task by integrating sentence-level grammatical information.

Proposed Approach

Self-Supervised Learning Task

Let $S = \{w_1, w_2, ..., w_n\}$ be an ordered sequence of n tokens, which is taken from raw unlabeled news data, assumed to be fluent. We then propose two self-supervised tasks.

Tagging Task The input of the tagging task is a disfluent sentence S_{disf}, generated by randomly adding words to a fluent sentence. S_{disf} is fed into a transformer encoder network to learn the representation of each word, $\{h_1, h_2, ..., h_n\}$. The goal is to detect the added noisy words by associating a label for each word, where the labels D and O means that the word is an added word and a fluent word, respectively. Although the distribution of the tagging task data is different from the distribution of the gold disfluency detection data, the training goal is to keep the generated sentences fluent by deleting disfluent words, which matches the goal of disfluency detection. We argue that the tagging model can capture more sentence structural information which is helpful for disfluency detection.

We start from a fluent sequence S and introduce random perturbations to generate a disfluent sentence S_{disf}. More specifically, we propose two types of perturbations:

- *Repetition*(k) : the m (randomly selected from one to six) words starting from the position k are repeated.
- *Inserting*(k) : we randomly pick a m-gram (m is randomly selected from one to six) from the news corpus and insert it to the position k.

For the input fluent sentence, we randomly choose one to $three$ positions, and then randomly take one of the two perturbations for each selected position to generate the disfluent sentence S_{disf}. It is important to note that it is possible that in some cases S_{disf} will itself form a fluent sentence and hence violate the definition of the disfluent sentence. We do not address this issue and assume that such cases will be relatively few and will not harm the training goal when the training data is large.

Sentence Classification Task The input of sentence classification task is a sentence pair $< S_1, S_2 >$, where one is a fluent sentence and the other one is disfluent, generated by randomly adding or deleting some words from the corresponding fluent sentence. The sentence pair is fed into a transformer encoder network to obtain a sentence pair representation h_s. The goal of the task is to discriminate between fluent sentence and corresponding disfluent one. We define a label set, $\{add_0, add_1, del_0, del_1\}$, where add_0 and del_0 mean that the first input sentence S_1 is generated by randomly adding and deleting some words from the second sentence S_2, respectively. We hypothesize that this task can capture sentence-level grammatical information, which is helpful for disfluency detection whose training goal is to keep the generated sentence fluent by deleting the disfluent words.

We construct two kinds of disfluent sentences for this task. We use the same method described in the tagging task to construct the disfluent sentence S_{add} with added noisy words. For the disfluent sentence S_{del} with deleted words, we consider a new type of perturbations:

- *Delete*(k) : for selected position k, m (randomly selected from one to six) words starting from this position are deleted.

For the input fluent sentence, we randomly choose one to three positions, and then take the *Delete*(k) perturbation to generate S_{del}. Note that one sentence can only be used to generate one kind of disfluent sentence to prevent the model from learning some statistical rules (e.g. the sentence with intermediate length is a fluent sentence) beyond our goals.

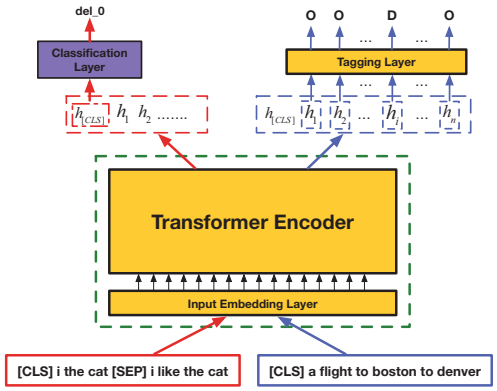

Figure 3: Model structure. The parameters of input embedding layer I, encoder layer E, and tagging layer T (yellow box) are shared among pre-training and fine-tuning

Network Structure

As shown in Figure 3, the model consists of four parts: an input embedding layer I, an encoder layer E, a tagging layer T for the tagging task, and a classification layer C for the classification task.

For I, given a token, its input representation is constructed by summing the corresponding token, position, and segment embeddings. For E, we use the multi-layer bidirectional transformer encoder described in Vaswani et al. (2017).

For the tagging task, E takes a sequence of input $([CLS], x_1, x_2, ..., x_n)$ and returns a representation sequence $(h_{[CLS]}, h_1, h_2, ..., h_n)$. Then the representation sequence $(h_1, h_2, ..., h_n)$ is sent to T to get a sequence of labels $(y_1, y_2, ..., y_n)$, where $y_i \in \{O, D\}$.

For the sentence classification task, E takes two sequences of input $([CLS], x_1^1, x_2^1, ..., x_m^1, [SEP], x_1^2, x_2^2, ..., x_n^2)$ and returns a representation sequence $(h_{[CLS]}, h_1^1, h_2^1, ..., h_m^1, h_{[SEP]}, h_1^2, h_2^2, ..., h_n^2)$. Then we send the representation $h_{[CLS]}$ to C to get the classification label y, where $y \in \{add_0, add_1, del_0, del_1\}$.

Multi-Task Pre-training Procedure

Multi-task learning helps in sharing information between different tasks and across domains. Our primary aim is to use the sentence classification task to help the tagging task by integrating sentence-level grammatical information.

Under the multi-task learning framework, the parameters of I and E are shared. We denote T_{ind} and C_{ind} as the representations of the tagging layer and the classification layer, respectively. The total loss of the multi-task neural network is calculated as:

$$Loss = Loss_{tag} + Loss_{cl},$$

where $Loss_{tag}$ means the loss of tagging task, and $Loss_{cl}$ means the loss of sentence classification task.

In practice, we construct mini-batches of training examples, where 30% of the data are single sentences used for the tagging task, and another 70% are sentence pairs for the sentence classification task. Since parts of the encoder are shared among both tasks, we optimize both loss terms concurrently.

Disfluency Detection Fine-tuning

We directly fine-tune the pre-trained tagging model (including input embedding layer I, encoder layer E, and tagging layer T) on gold human-annotated disfluency detection data. Given a pre-trained tagging model, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target disfluency detection data, and it allows us to train a robust disfluency detection model even for small datasets. For fine-tuning, most model hyperparameters are the same as in pre-training, with the exception of the batch size, and number of training epochs.

Experiment

Settings

Dataset. English Switchboard (SWBD) (Godfrey, Holliman, and McDaniel 1992) is the standard and largest (1.73×10^5 sentences for training) corpus used for disfluency detection. We use English Switchboard as main data. Following the experiment settings in Charniak and Johnson (2001), we split the Switchboard corpus into train, dev and test set as follows: train data consists of all sw[23]$*$.dff files, dev data consists of all sw4[5-9]$*$.dff files and test data consists of all sw4[0-1]$*$.dff files. Following Honnibal and Johnson (2014), we lower-case the text and remove all punctuations and partial words.[1] We also discard the 'um' and 'uh' tokens and merge 'you know' and 'i mean' into single tokens.

Unlabeled sentences are randomly extracted from WMT2017 monolingual language model training data (News Crawl: articles from 2016), consisting of English news[2]. Then we use the methods described previously to construct the pre-training dataset. The training set of the tagging task contains 3 million sentences, in which half of them are pseudo disfluent sentences S_{disf} and others are fluent sentences directly extracted from the news corpus. We use 9 million sentence pairs for the sentence classification task.

Metric. Following previous works (Ferguson, Durrett, and Klein 2015), token-based precision (P), recall (R), and (F1) are used as the evaluation metrics.

Baseline. We build two baseline systems including:(1) **Transition-based** (Wang et al. 2017) is a neural transition-based model and achieves the current state-of-the-art result by integrating complicated hand-crafted features. We directly use the code released by Wang et al. (2017).[3] (2) **Transformer-based** is a multi-layer bidirectional transformer encoder with random initialization to directly train on human-annotated disfleuncy detection data. The network structure is similar to our model except for the classification layer C in Figure 3. We add it to show that the improvements do not come from the multi-layer bidirectional transformer encoder, yet from the pre-training process.

[1] words are recognized as partial words if they are tagged as 'XX' or end with '-'.

[2] http://www.statmt.org/wmt17/translation-task.html

[3] https://github.com/hitwsl/transition_disfluency

Method	Full						1000 sents					
	Dev			Test			Dev			Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Transition-based	92.2	84.7	88.3	92.1	84.1	87.9	82.2	57.4	67.6	81.2	56.7	66.8
Transformer-based	86.5	70.4	77.6	86.1	71.5	78.1	78.2	51.3	62.0	79.1	51.1	62.1
Our self-supervised	92.9	88.1	**90.4**	93.4	87.3	**90.2**	90.0	82.8	**86.3**	88.6	83.7	**86.1**

Table 2: Experiment results on English Switchboard data, where "Full" means the results using 100% human-annotated data, and "1000 sents" means the results using less than 1% (1000 sentences) human-annotated data.

Method	**P**	**R**	**F1**
UBT (Wu et al. 2015)	90.3	80.5	85.1
Semi-CRF (Ferguson et al., 2015)	90.0	81.2	85.4
Bi-LSTM (Zayats et al., 2016)	91.8	80.6	85.9
LSTM-NCM (Lou and Johnson 2017)	-	-	86.8
Transition-based (Wang et al. 2017)	91.1	84.1	87.5
Our self-supervised (1000 sents)	88.6	83.7	86.1
Our self-supervised (Full)	**93.4**	**87.3**	**90.2**

Table 3: Comparison with previous state-of-the-art methods on the test set of English Switchboard. "Full" means using 100% human-annotated data for fine-tuning, and "1000 sents" means using less than 1% (1000 sentences) human-annotated data for fine-tuning.

Training Details

In all experiments including the transformer-based baseline and our self-supervised method, we use a transformer architecture with 512 hidden units, 8 heads, 6 hidden layers, GELU activations (Hendrycks and Gimpel 2016), and a dropout of 0.1. We train our models with the Adam optimizer.

For the joint tagging and sentence classification objectives, we use streams of 128 tokens and a mini-batches of size 256. We use learning rate of 1e-4 and epoch of 30. When fine-tuning on gold disfluency detection data, most model hyperparameters are the same as in pre-training, with the exception of the batch size, learning rate, and number of training epochs. We use batch size of 32, learning rate of 1e-5, and epoch of 20.

Performance On English Switchboard

Table 2 shows the overall performances of our model on both development and test sets. We can see that our self-supervised method outperforms the baseline methods in all the settings. Surprisingly, our self-supervised method achieves almost 20 point improvements over transition-based method when using less than 1% (1000 sentences) human-annotated disfluency detection data.

We compare our self-supervised model to five top performing systems, which rely on large-scale human-annotated data and complicated hand-crafted features. Our model outperforms the state-of-the-art, achieving a 90.2% F1-score as shown in Table 3. We attribute the success to the strong ability to learn global sentence-level structural information. Surprisingly, with less than 1% (1000 sentences) human-annotated training data, our model achieves comparable F1-score as the previous top performing systems using 100% human-annotated training data, which shows that our self-supervised method can substantially reduce the need for human-annotated training data. Note that we do not compare our work with the work of Wang et al. (2018) using semi-supervised method for disfluency detection. Wang et al. (2018) treat interregnum and reparandum types equally when training and evaluating their model, while others (including ours and all the baselines in Table 3) only focus on reparandums which are more difficult to detect.

Method	Full			1000 sents		
	P	R	F1	P	R	F1
Random-Initial	86.1	71.5	78.1	79.1	51.1	62.1
Tagging	91.8	84.0	87.7	85.1	79.6	82.3
Classification	91.2	83.1	87.0	83.2	78.3	80.7
Multi-Task	**93.4**	**87.3**	**90.2**	**88.6**	**83.7**	**86.1**

Table 4: Ablation over the two self-supervised tasks. "Random-Initial" means training transformer network on gold disfluency detection data with random initialization.

Hyperparams			Dev Set Performance	
#L	#H	#A	F1 (1000 sents)	F1 (Full)
2	256	4	78.1	85.7
2	256	6	79.7	86.8
4	256	6	81.8	88.2
4	256	8	82.9	89.1
6	256	8	84.5	89.6
6	512	8	**86.3**	**90.3**

Table 5: Ablation over model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "Full" means fine-tuning on 100% gold training data, and "1000 sents" means fine-tuning on less than 1% (1000 sentences) gold training data.

Ablation Studies

Effect of the two Self-Supervised Tasks

We explicitly compare the impact of the tagging task and the sentence classification task. As shown in Table 4, both of our two self-supervised tasks achieve higher performance compared with the baseline system with random initialization.

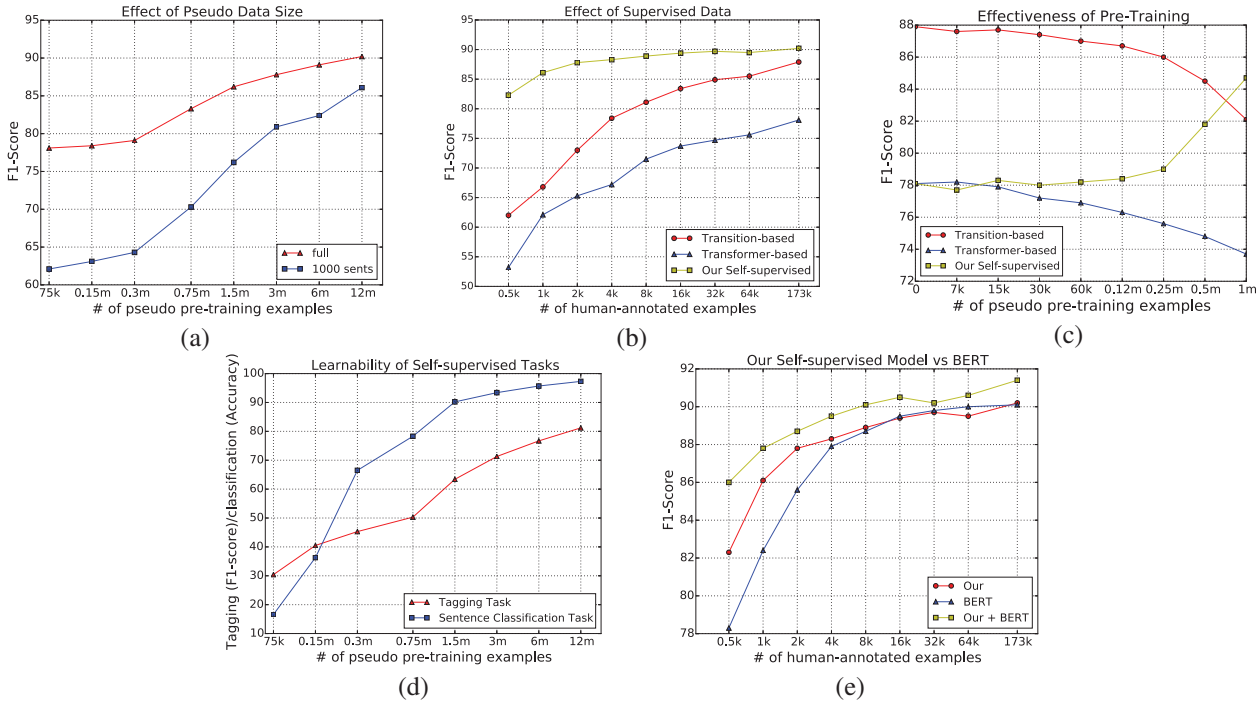

Figure 4: (**a**) Plot showing the impact of pseudo training data size to disfluency detection. (**b**) Plot showing the impact of human-annotated data size when fine-tuning. (**c**) Plot showing the effectiveness of pre-training compared with the baseline methods of directly merging gold training data with the pseudo training data. (**d**) Plot showing learnability of self-supervised tasks. (**e**) Plot showing the effectiveness of multi-task learning.

Higher performance is achieved by combining the two self-supervised tasks, which demonstrates that the two tasks can coexist harmoniously, and share useful information between each other.

Effect of Model Size

We explore the effect of model size on fine-tuning disfluency detection task. We mainly tune the number of layers, hidden units, and attention heads, while otherwise using the same hyperparameters as described previously. In order to avoid the impact of randomness, we run 3 random restarts of fine-tuning and report the average F1-Score on the Dev set. Results are shown in Table 5. We can see that larger models always lead to a strict F1-score improvement on the Dev set. It is also surprising that larger models always reduce the performance gap between fine-tuning using full gold training data and fine-tuning using less than 1% (1000 sentences) training data. We believe that much more performance will be gained if we keep increasing the model size.

Analysis

Varying Amounts of Pseudo Data

We observed the impact of pseudo training data size to disfluency detection task. Figure 4 (a) reports the results of adding varying amounts of pseudo training data to the self-supervised pre-training model. We observe that F1-score keeps growing when the amount of automatically-generated data increases. We conjecture that our two self-supervised tasks and disfluency detection task can coexist harmoniously, and more automatically-generated training data will bring more structural information. Another surprising observation is that the performance on the small supervised dataset (1000 sentences) grows faster, which shows that our method has huge potential to tackle the training data bottleneck.

Varying Amounts of Supervised Data

We explore how fine-tuning scales with human-annotated data size, by varying the amount of human-annotated training data the model has access to. We plot F1-score with respect to the amounts of human-annotated disfluency detection data for fine-tuning in Figure 4 (b). Compared with the baseline systems, fine-tuning based on our self-supervised models improves performance considerably when limited gold human-annotated training data is available, but those gains diminish with more high-quality human-annotated data. Using only 2% of the labeled data, our approach already performs as well or better than the previous state-of-the-art transition-based method using 100% of the human-annotated training data, demonstrating that our self-supervised tasks are particularly useful on small datasets.

Effectiveness of Pre-Training

We explore the contribution of pre-training to the final experimental results. As the pseudo-training data (S_{disf} con-

structed by adding words) is similar to the gold disfluency detection data in format, a natural idea is to directly train previous methods with the mixed pseudo S_{disf} data and gold training data. So we re-train the baseline transition-based method and transformer-based method by merging the gold training data with the pseudo S_{disf} data. As shown in Figure 4 (c), F1-scores of the two baseline methods keep decreasing when the amount of the pseudo training data increases, while F1-score of our self-supervised method keeps increasing. The results show that our self-supervised method is much more effective compared with the methods of directly merging the gold training data with the pseudo data. We attribute the F1-score decrease of baseline methods to the discrepancies between the distribution of gold disfluency detection sentence and the pseudo disfluent sentence S_{disf}.

Learnability of Self-Supervised Tasks

Our self-supervised tasks are very similar to supervised tasks, excepted that the training data is collected without manual labeling. To prove the learnability of our self-supervised tasks, we explore the performance of our self-supervised models on the pseudo testing data when pre-training. We plot the performance with respect to the amounts of pseudo training data for pre-training in Figure 4 (d). The performance keeps growing when the amount of automatically-generated data increases, achieving about 80% F1-score on tagging task and 97% accuracy on sentence classification task, respectively. The results show that our self-supervised tasks are reasonable, which can really capture more sentence structural information.

Repetitions vs Non-repetitions

Repetition disfluencies are easier to detect and even some simple hand-crafted features can handle them well. Other types of reparandums such as repair are more complex (Zayats, Ostendorf, and Hajishirzi 2016; Ostendorf and Hahn 2013). In order to better understand model performances, we evaluate our model's ability to detect repetition vs. non-repetition (other) reparandum. The results are shown in Table 6. All three models achieve high scores on repetition reparandum. Our self-supervised model is much better in predicting non-repetitions compared to the two baseline methods. We conjecture that our self-supervised tasks can capture more sentence-level structural information.

Comparison with BERT

We would like to see the performance comparison between our pre-trained model and BERT. The large version of pre-trained BERT model (24-layer transformer blocks, 1024 hidden-size, and 16 self-attention heads, totally 340M parameters) is used for the comparison. In our experiment, we follow the hyper-parameters (e.g. batch size of 32) of Devlin et al. (2019) when fine-tuning BERT. Additionally, for the large version of pre-trained BERT model, fine-tuning was sometimes unstable on small datasets, so we run 3 random restarts and select the best model on the development set.

Compared with BERT, we use much smaller training corpus and model parameters (6-layer transformer blocks, 512 hidden-size, and 8 self-attention heads) limited by devices. Results are shown in Table 7. Both our method and BERT outperforms the baseline model with random initialization, which proves the strong ability of pre-training model. Although our pre-training corpus and model parameters are much smaller than BERT, we achieve a similar result with BERT when fine-tuning on full gold training data. Surprisingly, our method achieves 3.7 point improvements over BERT when fine-tuning on 1%(1000 sentences) of the gold training data. We also try to combine our pre-train model and BERT by concatenating their hidden representation. The result shows that much higher performance is achieved. We also plot the performance with respect to the length of human-annotated disfluency detection data in Figure 4 (e). The performance is always much higher by combining our pre-train model and BERT. This proves that our model and BERT can coexist harmoniously, and capture different aspects of information helpful for disfluency detection.

Method	Repet	Non-repet	Either
Transition-based	93.8	68.3	87.9
Transformer-based	93.6	58.9	78.1
Our self-supervised	93.7	70.8	**90.2**

Table 6: F1-score of different types of reparandums on English Switchboard test data.

Method	F1 (Full)	F1 (1000 sents)
Random-Initial	78.1	62.1
BERT-fine-tune	90.1	82.4
Our self-supervised	90.2	86.1
Combine	**91.4**	**87.8**

Table 7: Comparison with BERT. "random-initial" means training transformer network on gold disfluency detection data with random initialization. "combine" means concatenating hidden representations of BERT and our self-supervised models for fine-tuning.

Conclusion

In this work, we propose two self-supervised tasks to tackle the training data bottleneck. Experimental results on the commonly used English Switchboard test set show that our approach can achieve competitive performance compared to the previous systems (trained using the full dataset) by using less than 1% (1000 sentences) of the training data. Our method trained on the full dataset significantly outperforms previous methods, reducing the error by 21% on English Switchboard.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. Shaolei Wang was supported by China Scholarship Council (CSC), and the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153. Wanxiang Che is the corresponding author.

References

Agrawal, P.; Carreira, J.; and Malik, J. 2015. Learning to see by moving. In *Proceedings of ICCV*, 37–45.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Bingel, J., and Søgaard, A. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of EACL*, 164–169.

Charniak, E., and Johnson, M. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL*.

Dong, Q.; Wang, F.; Yang, Z.; Chen, W.; Xu, S.; and Xu, B. 2019. Adapting translation models for transcript disfluency detection. In *Proceedings of AAAI*.

Ferguson, J.; Durrett, G.; and Klein, D. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of NAACL*, 257–262.

Fernando, B.; Bilen, H.; Gavves, E.; and Gould, S. 2017. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of CVPR*, 3636–3645.

Georgila, K. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of NAACL*.

Godfrey, J. J.; Holliman, E. C.; and McDaniel, J. 1992. Switchboard: Telephone speech corpus for research and development. In *icassp*, 517–520. IEEE.

Hendrycks, D., and Gimpel, K. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Honnibal, M., and Johnson, M. 2014. Joint incremental disfluency detection and dependency parsing. *TACL* 2.

Hough, J., and Schlangen, D. 2015. Recurrent neural networks for incremental disfluency detection. In *INTERSPEECH*.

Jamshid Lou, P.; Anderson, P.; and Johnson, M. 2018. Disfluency detection using auto-correlational neural networks. In *Proceedings of EMNLP*, 4610–4619.

Johnson, M., and Charniak, E. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of ACL*.

Liu, T.; Cui, Y.; Yin, Q.; Zhang, W.; Wang, S.; and Hu, G. 2016. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. *arXiv preprint arXiv:1606.01603*.

Lou, P. J., and Johnson, M. 2017. Disfluency detection using a noisy channel model and a deep neural language model. *Proceedings of ACL*.

Martínez Alonso, H., and Plank, B. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of EACL*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *CoRR*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Ostendorf, M., and Hahn, S. 2013. A sequential repetition model for improved disfluency detection. In *Interspeech*.

Peng, H.; Thomson, S.; and Smith, N. A. 2017. Deep multitask learning for semantic dependency parsing. In *ACL*.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Qian, X., and Liu, Y. 2013. Disfluency detection using multi-step stacked learning. In *HLT-NAACL*, 820–825.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Rasooli, M. S., and Tetreault, J. R. 2013. Joint parsing and disfluency detection in linear time. In *EMNLP*, 124–129.

Shriberg, E. E. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. Dissertation, Citeseer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, X., and Gupta, A. 2015. Unsupervised learning of visual representations using videos. In *Proceedings of ICCV*.

Wang, S.; Che, W.; Zhang, Y.; Zhang, M.; and Liu, T. 2017. Transition-based disfluency detection using lstms. In *Proceedings of EMNLP*, 2785–2794.

Wang, F.; Chen, W.; Yang, Z.; Dong, Q.; Xu, S.; and Xu, B. 2018. Semi-supervised disfluency detection. In *COLING*.

Wang, S.; Che, W.; and Liu, T. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING*.

Wu, S.; Zhang, D.; Zhou, M.; and Zhao, T. 2015. Efficient disfluency detection with transition-based parsing. In *ACL*.

Yoshikawa, M.; Shindo, H.; and Matsumoto, Y. 2016. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *EMNLP*.

Zayats, V., and Ostendorf, M. 2018. Robust cross-domain disfluency detection with pattern match networks. *arXiv preprint arXiv:1811.07236*.

Zayats, V., and Ostendorf, M. 2019. Giving attention to the unexpected: Using prosody innovations in disfluency detection. *arXiv preprint arXiv:1904.04388*.

Zayats, V.; Ostendorf, M.; and Hajishirzi, H. 2014. Multi-domain disfluency and repair detection. In *Proceedings of Interspeech*.

Zayats, V.; Ostendorf, M.; and Hajishirzi, H. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.

Zwarts, S.; Johnson, M.; and Dale, R. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of COLING*, 1371–1378.