Adapting Translation Models for Transcript Disfluency Detection

Qianqian Dong,[1,2] **Feng Wang,**[1] **Zhen Yang,**[1,2] **Wei Chen,**[1] **Shuang Xu,**[1] **Bo Xu**[1,2*]
[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{dongqianqian2016, feng.wang, yangzhen2014, wei.chen.media, shuang.xu, xubo}@ia.ac.cn

Abstract

Transcript disfluency detection (TDD) is an important component of the real-time speech translation system, which arouses more and more interests in recent years. This paper presents our study on adapting neural machine translation (NMT) models for TDD. We propose a general training framework for adapting NMT models to TDD task rapidly. In this framework, the main structure of the model is implemented similar to the NMT model. Additionally, several extended modules and training techniques which are independent of the NMT model are proposed to improve the performance, such as the constrained decoding, denoising autoencoder initialization and a TDD-specific training object. With the proposed training framework, we achieve significant improvement. However, it is too slow in decoding to be practical. To build a feasible and production-ready solution for TDD, we propose a fast non-autoregressive TDD model following the non-autoregressive NMT model emerged recently. Even we do not assume the specific architecture of the NMT model, we build our TDD model on the basis of Transformer, which is the state-of-the-art NMT model. We conduct extensive experiments on the publicly available set, Switchboard, and in-house Chinese set. Experimental results show that the proposed model significantly outperforms previous state-of-the-art models.

Introduction

Disfluency is a characteristic of spontaneous speech which makes it different from written text. Transcript disfluency detection is a task of recognizing non-fluent word sequences in spoken language transcripts or automatic speech recognition (ASR) results (Lou and Johnson 2017; Wang et al. 2017). Disfluencies are informally defined as interruptions in the normal flow of speech that occur in different forms, including false starts, repairs, repetitions and filled pauses. Frequent disfluencies introduce obstacles to many natural language processing (NLP) tasks, such as dialogue systems, machine translation, natural language understanding and so on. Therefore, it's necessary to pre-process disfluencies in source corpus before passed to downstream NLP tasks. As shown in Figure 1, standard annotation of disfluency structure (Shriberg 1994) indicates the reparandum

$$\text{A flight [} \underbrace{\text{to Boston}}_{RM} + \{ \overbrace{\text{uh}}^{IM} \} \underbrace{\text{to Denver}}_{RP} \text{]}$$

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

(words that are discarded, or corrected by the following words), where the interruption point (+) marking the end of the reparandum, and the begin of the associated repair, or an optional interregnum (filled pauses, discourse cue words, etc). Some disfluencies are complicated and may contain diverse reparandums with arbitrary forms, varying in length, location, and structure. In our experiments, we find that the longest reparandum in the publicly available set, Switchboard, has as long as 15 words. And there are large similarities in syntax between the reparandum chunk and the following repair chunk. However, the mainstream method in this area is to view detecting disfluencies as a sequence labeling task which is usually good at capturing local context information and weak in modeling long-range dependencies (Lou and Johnson 2017; Wang et al. 2017).

In this paper, we investigate the long-range dependencies in TDD by referring to NMT model. In contrast to previous neural approaches, our approach treats TDD as a translation task from a source sentence (the disfluent sentence) into a target sentence (a fluent version in the same language as the source). We propose a general training framework for adapting the NMT model to TDD task. In the proposed framework, the main structure of our model is implemented similar to the NMT model. Note that we do not assume the specific architecture of the NMT model. Additionally, several extended modules and training techniques are proposed to improve the performance, including the constrained decoding, denoising autoencoder initialization and a TDD-specific training object. While we achieve substantial improvements by adapting NMT model to TDD naively, the model is too slow in decoding to be practical. To tackle this problem, we extend the TDD model with non-autogressive decoding based on the idea of non-autogressive NMT model. With

*Corresponding author

this extension, the decoding speed of the proposed model is accelerated by 6 times, which is ready for production. We conduct extensive experiments on the publicly available set, Switchboard, and in-house Chinese set. Experimental results show that our final model achieves a new state-of-the-art result.

To summarize, this paper makes the following contributions:

- To the best of our knowledge, we are among the first endeavors to employ the NMT model to the task of TDD. We provide a new point of view for investigating TDD.
- We propose a general training framework for adapting NMT models to TDD task, which can be applied to any NMT model. Apart from the significant improvement on performance, our model achieves fast decoding speed which enables it ready for production.
- We build our model based on the newly emerged state-of-the-art NMT model and conduct extensive experiments on the publicly available test set Switchboard and in-house Chinese test set. Experimental results show that our model achieves the state-of-the-art performance on both test sets. Additionally, we give deep analysis to investigate the strength of our approach over previous methods.

Related Works

Transcript Disfluency Detection

Transcript disfluency detection has gained much attention within NLP and ASR community recently, as the demand for spoken cascaded systems combining ASR and NMT models increases and develops. Existing methods are mainly divided into four types: sequence labeling, parsing-based, noisy channel model, encoder-decoder model. The sequence labeling method labels each word as fluent or not using different model structures, including conditional random fields (Ostendorf and Hahn 2013; Zayats, Ostendorf, and Hajishirzi 2014), hidden Markov models (Liu et al. 2006; Ferguson, Durrett, and Klein 2015), Recurrent Neural Network (Hough and Schlangen 2015; Zayats, Ostendorf, and Hajishirzi 2016) or others (Georgila 2009; Qian and Liu 2013). The parsing-based approaches joint the task of detecting disfluencies and identifying the syntactic structure of the sentence (Rasooli and Tetreault 2013; Honnibal and Johnson 2014; Yoshikawa, Shindo, and Matsumoto 2016). Noisy channel models use the similarity between reparandum and repair as an indicator and language models as a reranker to detect disfluency (Johnson and Charniak 2004; Zwarts and Johnson 2011; Lou and Johnson 2017). The encoder-decoder models regard TDD as a sequence-to-sequence problem (Wang, Che, and Liu 2016). Recently, researchers have used translation models to perform TDD. Neubig et al. (Neubig et al. 2012) presented a monotonic statistical machine translation approach for transforming faithful transcripts to clean transcripts on Japanese. Cho et al. (Cho et al. 2016) investigated a multilingual approach for TDD on English and German multilingual speech transcripts. Wang et al. (Wang et al. 2018) proposed a semi-supervised approach to utilize large amounts of WMT2014 unannotated data. The state of the art of TDD on Switchboard without external data is achieved by sequence labeling method (Wang et al. 2017). These models have comparable performance, but have no power to solve the long-range dependencies problem, which is essential to detect complicated disfluencies with longer spans or distances. Works in Wang et al. (Wang, Che, and Liu 2016) can capture long-range dependencies by using an attention-based Recurrent Neural Network (RNN), but can't utilize chunk-level information well at the cost of time complexity and computational complexity.

Neural Machine Translation

In the field of machine translation, the state of the art neural sequence-to-sequence methods have led to a paradigm shift away from phrase-based statistical machine translation (SMT) to NMT. The most popular model for neural machine translation is the attention-based encoder-decoder model (Bahdanau, Cho, and Bengio 2014). The encoder takes the source sentence as input and calculates a representation of each word in the source sentence. The decoder outputs a translation relying on the representations of the source sentence. The encoder and decoder can be parametrized as RNN, Convolutional Neural Networks (CNN) or hierarchical self-attention models (Vaswani et al. 2017). In this paper, we experiment with Transformer, which achieved state-of-the-art performance on WMT 2014 English-to-French translation task with markedly less training cost. Its fundamental module is self-attention, a mechanism relating all the position-pairs of a sequence to extract a more expressive sequence representation. The self-attention can draw the dependencies between different positions through the position-pair computation. Much of the recent state of the art models in NMT are auto-regressive. When generating a sequence of translation from left to right, predicting the arbitrary symbol first requires generating all symbols before the current symbol. During training, the ground truth is known, so conditioning on previous symbols can be parallelized. But when actually generating sequential output during inferring, their autoregressive nature prevents these models from utilizing parallel computation. Some recent works have attempted to speed up decoding by training a non-autoregressive translation (NAT) model (Gu et al. 2017). The autoregressive connection from an existing encoder-decoder model is directly removed and a great speed increase has been achieved in NAT.

The Approach

The Model Overview

In this paper, the task of TDD is formulated as a translation task from the disfluent utterance to fluent utterance. That is, the source sentences are transcripts said by a lecturer not fully-prepared to give a speech and potentially contain disfluencies, whereas the target sentences are the corrected fluent sentences. As shown in Figure 2, the model is based on the encoder-decoder architecture: the encoder transforms a disfluent sequence $X = (x_1, ..., x_T)$ to a hidden representation $h = (h_1, ..., h_L)$. Given h, the decoder then generates

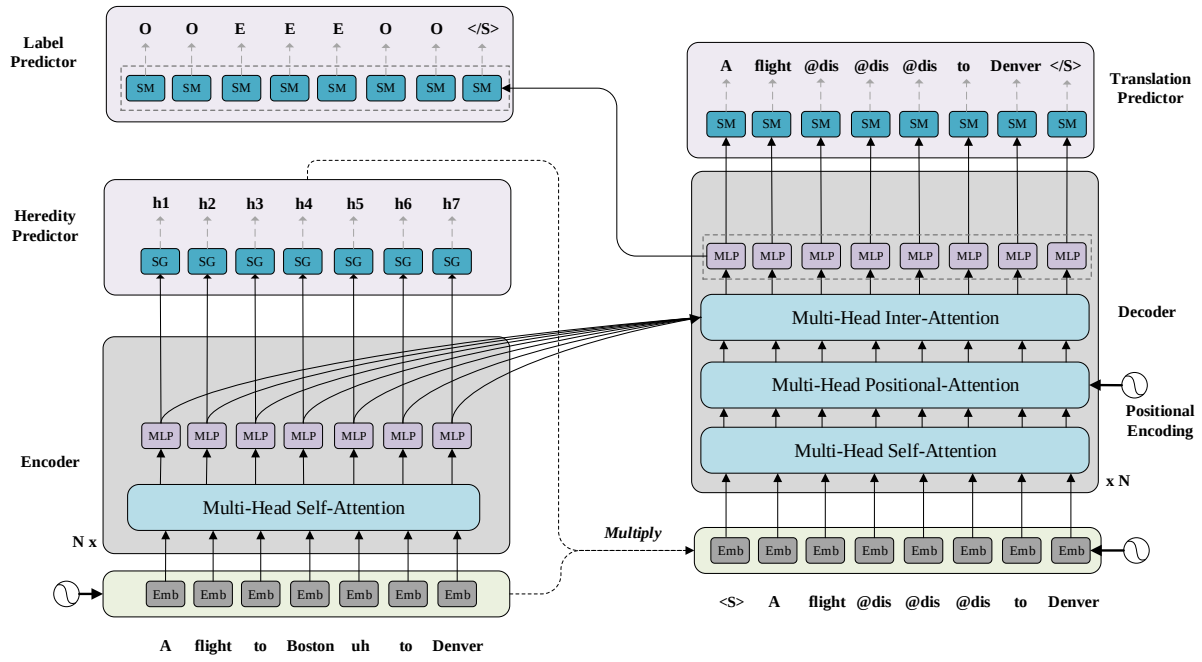

Figure 2: The architecture of our base model. Each sublayer inside the encoder and decoder also includes layer normalization and a residual connection. "SM"=Softmax function and "SG"=Sigmoid function. "Multi-Head Self/Inter/Positional-Attention" is the core structures of Transformer.

an output fluent sequence $Y = (y_1, ..., y_T)$. The distribution over possible output sentence Y is:

$$p_{\mathcal{AR}}(Y|X;\theta) = \prod_{t=1}^{T} p(y_t|y_{0:t-1}, x_{1:T};\theta) \quad (1)$$

We adopt Transformer (Vaswani et al. 2017) as our base model structure, consisting of 6 complex attention/self-attention blocks in the encoder and decoder.

The TDD task is different from machine translation. The goal of the TDD task is to accurately label the disfluencies in the source sentence. The goal of machine translation is to reasonably interpret the target sentence corresponding to the source sentence. In order to adapt the translation model to the TDD task, we propose several extended modules and training techniques, which will be discussed in the sections below. These adaptations are implemented incrementally.

Constrained Decoding

We introduce constrained decoding to make the decoding process effective for the TDD task (marked as "T-TDD"). In order to ensure the consistency in the word order and token coverage of the output result and the source sentence, we have adopted the idea of the copy neural network (Gu et al. 2016). We control the word ordering problem during decoding by utilizing two softmax layers, named "Label Predictor" and "Translation Predictor" respectively, as can be seen in Figure 2. When predicting the token y_t at the current step, we additionally predict a label z_t using a softmax layer to decide whether to predict a "@dis". "@dis" is a special symbol that means the word is disfluent. When z_t is equal to "E", the model predicts a "@dis", otherwise the model copies the current token from the source sentence. We introduce a weighting factor α to balance the two softmax outputs. Then the loss function of the model is as follows:

$$\begin{aligned} & L(x, y, z|\theta) \\ =& (1-\alpha)L(x, y|\theta) + \alpha L(x, z|\theta) \\ =& -\sum_{t=1}^{T}(1-\alpha) log P(y_t|x, y_{<t}, \theta) \\ & \qquad -\sum_{t=1}^{T}\alpha log P(z_t|x, y_{<t}, \theta) \end{aligned} \quad (2)$$

Deletion-Penalty MLE Objective

Because TDD is essentially a two-category annotation task, it is to determine whether the current word is fluent or not. Fluent sentences are obtained by deleting disfluencies. In light of the actual needs of this task, the precision is more important than the recall. The missing deletions will only affect the semantic expression, but the wrong deletions will impair the integrity of the semantics. Therefore, we modify the training objective of the original NMT model by adding a penalty factor for the deletion operations (marked as "+DP-MLE"). Adopting the marking symbols of the translation system, each token y_i in target sentence is generated by the token x_i in the source sentence. The maximum likelihood

estimation (MLE) loss is scaled by a factor λ if y_i is equal to the special token "@dis". Therefore, the final loss of $L(x, y)$ in Equation 2 is calculated as follows:

$$L(x, y) = -\sum_{t=1}^{T} f(y_t) log P(y_t|x, y_{<t}, \theta),$$

where $f(y_t)$ is determined by:

$$f(y_t) = \begin{cases} \lambda & \text{if } y_t = \text{"@dis"} \\ 1 & otherwise \end{cases} \quad (3)$$

DAE Initialization

Denoising autoencoder (DAE) is generally employed to effectively pre-train a deep neural network in low-resource NMT. DAEs take a pair of noisy-clean data, and are forced to learn a robust representation of the features. Thus they can correctly recover the correct examples from their corrupted counterparts. TDD can be treated as a denoising task where disfluencies are corruptions that have to be deleted. This is similar to the idea of DAE. So we use this method to pre-train our model, called "DAE Initialization" (marked as "+DAE"). Figure 3 shows the process of DAE initialization.

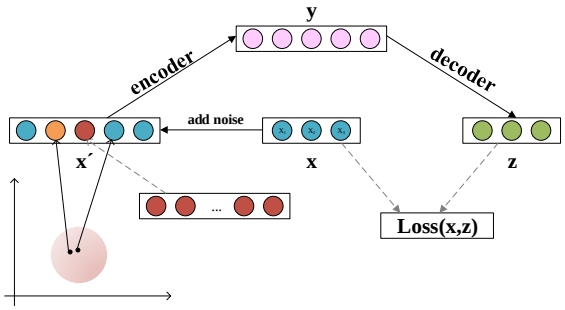

Figure 3: The architecture of the DAE initialization. x is the fluent sentence, and x' is the damaged sentence.

We achieve our goal by damaging the fluent target sentences in the original training data and then reconstructing them. According to the section of Introduction, there are high semantic and structural similarities between reparandums and repairs in the disfluencies. Therefore, we add the noise by randomly selecting several source words in the fluent sentence x and then selecting the words in the pre-trained word embeddings that are closest to the source words as noise to be inserted at the preceding positions of the source words to simulate reparandums. The number of source words being selected is decided by the amount of artificial data we need. The similarity of word embeddings is measured using cosine similarity and noise words with a similarity less than 0.8 will be discarded. The words in the high-frequency set of interregnums are further randomly inserted between the two chunks above to simulate interregnums. The pre-trained word embeddings are trained on the training source data using *word2vec*[1]. We pre-train our models with the artificial data until the loss function converges, then we use the true training corpus to continue training.

Accelerating Decoding Speed

Although our model has achieved improved performance, the general NMT model is limited by the slow decoding speed, which seriously affects the practicality of our TTD model. We adopt the idea of NAT to speed up the decoding process of TDD models (marked as "+NAT"). NATs introduce a sequence of discrete latent variables, which is designed to model the alignment between the source and target or capture the dependencies among target variables. The latent variables sequence will be generated autoregressively, and then each position in the target sequence from the latent variables will be reconstructed in parallel. In NMT, NATs may cause performance degradation due to lack of ground truth input during decoding and destruction of the sequential essence.

Unlike general translation tasks, source and target sentences belong to two different languages. In TDD task, a fluent transcription and a disfluent transcription are actually in the same language and there is a small editing distance between them. In the settings of our model, the input and output have strict one-to-one alignment and high similarity, which can compensate for the alignment and dependencies mentioned above. We introduce a continuous latent variable, called "heredity", to learn the inheritance relationship between the source and target. Therefore, we can automatically learn the generated ground truth during training and inferring. As can be seen in Figure 2, we directly generate the input embeddings for the decoder in the translation process, which is obtained by multiplying the heredity and embedding input of the encoder element by element. We first initialize *heredities* with an all-ones matrix and then generate a translation non-autoregressively conditioned on *heredities*. *Heredities* are random variables between 0 and 1 for each word in the source sentence to represent the conversion relation of embeddings from the encoder input to the decoder input. The heredity predictor module models the heredity $p(h_t|x_{1:T})$ at each position independently using a one-layer neural network with a sigmoid activation function on top of the last encoder layer. The distribution over possible output sentence $Y = \{y_1, ..., y_T\}$ becomes:

$$\begin{aligned} & p_{\mathcal{NAR}}(Y|X;\theta) \\ & = \arg\max_{H} p(H|x_{1:T};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:T};\theta,H) \\ & where\ H = h_{1:T}; h_{1:T} \in [0,1] \end{aligned} \quad (4)$$

The heredity values are a property of each input word but depend on information and context from the entire sentence, so the module has the ability to exploit chunk-level features and capture complicated disfluencies with longer spans or distances.

[1]https://code.google.com/archive/p/word2vec

Experimental Setup

Dataset

To directly compare with previous state-of-the-art results in the filed of TDD, we limit our training data strictly to public resources. Our training data includes the Switchboard disfluency-annotated corpus (Switchboard portion) of the English Penn Treebank and an in-house Chinese dataset. Firstly, we conduct our experiments on the English Switchboard corpus. Following the experiment settings in (Charniak and Johnson 2001; Honnibal and Johnson 2014; Wu et al. 2015), we use directory 2 and 3 in PARSED/MRG/SWBD as our training set and split directory 4 into test set, development set, and others. The development data consists of all sw4[5-9]*.dps files, and test data consists of all sw4[0-1]*.dps files. Based on whether the repair is empty, the same as the reparandum or different from the reparandum, respectively, the disfluencies can further be divided into three types: restarts without repairs ("restart"), repetitions ("repetition"), and restarts with repairs ("repair"). The statistics on the training set, development set, and test set in Switchboard can be seen in Table 1.

Dataset	Total	Restart	Repair	Repetition
Train	169494	2726	15662	19519
Dev	9835	261	1354	1375
Test	7677	138	912	1165

Table 1: The statistics on the training set, development set, and test set in Switchboard. Note that a sentence may contain multiple kinds of disfluencies.

We generate the fluent target corpus according to the disfluency annotations in the Switchboard corpus by substituting the reparandum, interregnum, filled pauses, discourse markers and so on with the disfluency tag "@dis", not making a distinction among different kinds of disfluencies. By this way, we can get a one-to-one parallel corpus of equal length, which can partially solve the over-translation and under-translation. Following previous works (Johnson and Charniak 2004; Honnibal and Johnson 2014), we remove all punctuations and partial words[2]. Instead of lowercasing all training and test data, we may also want to keep most words in their natural case, and escape special characters using scripts included in Moses [3].

Since no public Chinese corpus is available now, for our Chinese experiments, we collect about $30k$ spoken sentences from personal statement transcriptions and annotate them with only disfluency annotations according to the guideline similar to Switchboard (Meteer et al. 1995). We respectively select about $1k$ sentences for development and testing. The rest are used for training. The details for our Chinese TDD dataset and our annotation rules is available online [4].

[2]words are recognized as partial words if they are tagged as "XX" or end with "-"

[3]https://github.com/moses-smt/mosesdecoder

[4]https://github.com/dqqcasia/Translation_Disfluency_Detection/tree/master/data/chinese_disfluency

Following previous works (Wang et al. 2017; Lou and Johnson 2017), token-based precision (P), recall (R), and F-score (F) are used as the evaluation metrics.

Training Details

Our models are implemented with TensorFlow[5]. We use the hyperparameter settings of the base Transformer model described in Vaswani et al. (2017) for encoder stack and decoder stack. We share encoder and decoder word embeddings during training and inference. For Switchboard dataset, all the sentences are tokenized using Moses and we use a shared word-level vocabulary of 20000. For Chinese corpus, we use a shared character-level vocabulary of 3000 for modeling without word segmentation. We also try experiments using a word-level vocabulary, but the result is far inferior to the character-level model. The main reason may be that the word segmentation for Chinese spoken text is very poor. Sentence pairs are batched together by approximate sequence length. Each batch contains a set of sentence pairs with approximately 7000 source tokens and target tokens. Our models are trained for a max 200000 steps on 2 NVIDIA Titan-X GPUs.

Results and Analysis

We conduct experiments on both the commonly used English Switchboard test set and a set of in-house annotated Chinese data. The following is the results:

Effects of α

We conduct experiments to find the optimal values of α. Experimental results on Switchboard are shown in Table 2. In following experiments, the value of α is set to 0.2 on Switchboard, and set to 0.3 on Chinese data.

Value of α	Dev			Test		
	P	R	F	P	R	F
0.05	84.0	87.3	85.6	84.2	86.3	85.3
0.10	89.7	85.2	87.4	89.2	84.2	86.6
0.15	90.8	85.2	87.9	90.2	84.2	87.1
0.20	93.3	84.4	88.7	92.5	83.5	**87.8**
0.25	91.2	84.9	87.9	91.4	84.2	87.6
0.30	90.8	85.3	87.9	90.4	84.7	87.5

Table 2: Performance on Switchboard data with different values of α.

Effects of λ

λ affects the precision/recall trade-off and requires tuning for specific tasks. Experimental results on Switchboard can be seen in Table 3. In our experiments, the optimal value of λ is 1.5 on Switchboard and 2 on Chinese data respectively.

Effects of artificial data size

Table 4 shows the relationship between the performance of the method and the size of the artificial data[6]. An appropri-

[5]https://github.com/dqqcasia/Translation_Disfluency_Detection

[6]We try to do experiments with more artificial data, but the performance of the model no longer increases.

Value of λ	Dev			Test		
	P	R	F	P	R	F
0.5	90.7	86.3	88.4	90.3	84.9	87.5
1	93.3	84.4	88.7	92.5	83.5	87.8
1.5	93.8	84.3	88.8	93.2	83.4	**88.0**
2	91.5	86.1	88.7	90.7	84.6	87.5

Table 3: Performance on Switchboard data with different values of λ.

ate amount of artificial data can increase the recall rate and F-score of our model, but it will cause a certain degree of accuracy reduction due to the unreality of the artificial data. We also try to mix the artificial training samples directly into the real training samples, but there's a performance degradation for F-score, which may be due to the artificial data destroying the distribution of real data. Relatively speaking, the DAE initialization method is more robust. The optimal size of artificial data on Chinese data is 50k.

Size of artificial data	Dev			Test		
	P	R	F	P	R	F
0k	93.8	84.3	88.8	93.2	83.4	88.0
50k	88.8	87.9	88.3	89.2	87.1	88.1
150k	91.2	87.2	89.1	90.7	86.4	88.5
300k	89.4	88.5	89.0	89.3	88.1	**88.7**

Table 4: Performance on Switchboard data with different sizes of artificial data.

Effects of the "heredity"

We design two sets of non-autoregressive experiments. To prove the validity of the "heredity", We also do experiments with models without the "heredity", where the embedding input of the decoder is set to be 0 (marked as "w/o Heredity"). It can be seen from the experimental results in Table 5 that models with the "heredity" (marked as "w/ Heredity") are superior to models without the "heredity", with a performance improvement of 0.9 point.

+NAT model	Dev			Test		
	P	R	F	P	R	F
w/o Heredity	93.8	83.8	88.5	93.4	82.6	87.7
w/ Heredity	92.0	86.4	89.1	94.5	84.1	**89.0**

Table 5: Performance on Switchboard data of models with or without the "heredity".

Because there was no publicly available analysis on speed in prior works related to TDD, no comparison is given against prior work. We compare the parameters and speed between "T-TDD" and "+NAT" model, as shown in Table 6. The "T-TDD" model has 87.5M parameters. The "+NAT" model using *heredity* introduces additional 0.26M parameters, which is quite small compared to the number of parameters in the base model. Introducing the non-autoregressive property slows down the training speed, but acceptably and not significantly. When running on two Titan-X GPU devices, the speed of the "T-TDD" model is 0.0012s per token, and the speed of the "+NAT" model is 0.0002s per token, which is almost six times faster. Speed is measured using the average decoding speed of a batch. "+NAT" model can better satisfy the real-time requirements in speech translation system.

Method	Parameters	Train Time	Speed
T-TDD	87.5M	12h	0.0012s/token
+NAT	87.7M	20h	**0.0002s/token**

Table 6: Comparison on parameters and speed on Switchboard data.

Repetitions vs non-repetitions

In order to better understand our model's performance, we evaluate our model for detecting repetition and non-repetition reparandum, as shown in Table 7. And we obtain the same conclusion: it is much easier to detect repetitions than non-repetitions. Our model is better at detecting non-repetitions than the previous systems, and achieves a 78.3% F-score. It proves that our model can not only capture the features of the local chunk level, but also capture the global features and long-distance dependencies, which is necessary to correctly detect the repairs and restarts. Detecting non-repetitions still needs more powerful models.

Method	Repetitions	Non-repetitions	Either
CRF	93.8	61.4	82.6
Bi-LSTM	93.1	65.3	85.8
Transition-based	93.3	68.7	87.5
+NAT	93.6	**78.3**	**89.0**

Table 7: F-score of different types of reparandums on the test set of English Switchboard. Results of non-translation models come from Wang et al. (2017).

Main results

Switchboard corpus Table 8 shows some examples of the predictions of our model for Switchboard. We compare our best model to seven previous top performing systems as shown in Table 9. Our model outperforms the state-of-the-art work by 1.5 points, and achieves an 89.0% F-score. Our model achieves 3.9 points improvements over UBT (Wu et al. 2015), which is the best syntax-based method for TDD. Our model obtains a 3.6 points improvement compared to the best performance by linear statistical sequence labeling methods leveraging prosodic features (Ferguson, Durrett, and Klein 2015). Our model also achieves 2.3 points improvement over the attention-based model (Wang, Che, and Liu 2016), which also regards TDD as a sequence-to-sequence problem using RNN. Our model surpasses the best noisy channel models (Lou and Johnson 2017) by 2.2 points. We attribute the success to our model's strong ability to learn global chunk-level features and the effective state representation.

Chinese corpus Table 10 shows TDD results of Chinese. Our best model obtains a 1.7 points improvement compared

Source:	Huh well um you know I guess it 's pretty deep feelings uh
Ref:	@dis @dis @dis @dis @dis I guess it 's pretty deep feelings @dis
Model:	@dis @dis @dis @dis @dis I guess it 's pretty deep feelings @dis
Source:	I mean you know in retrospect um was it was it Mondale or Dukakis that said you know I 'll I 'll tell you
Ref:	@dis @dis @dis @dis in retrospect @dis was it @dis @dis Mondale or Dukakis that said @dis @dis I 'll @dis @dis tell you
Model:	@dis @dis @dis @dis in retrospect @dis was it @dis @dis Mondale or Dukakis that said @dis @dis I 'll @dis @dis tell you
Source:	Um do you think I mean do you think our the investment in lives and money was worth it
Ref:	@dis do you think @dis @dis @dis @dis @dis @dis the investment in lives and money was worth it
Model:	@dis do you think @dis @dis @dis @dis @dis **our @dis** investment in lives and money was worth it
Source:	And one of the things we discussed was you know where our where the budget how the budget situation just got out of hand
Ref:	@dis one of the things we discussed was @dis @dis @dis @dis @dis @dis @dis how the budget situation just got out of hand
Model:	@dis one of the things we discussed was @dis @dis **where our** @dis @dis **budget** how the budget situation just got out of hand

Table 8: Examples of the results of our model for Switchboard test data. Underlined words indicate disfluencies in the source sentence, and words in bold indicate where our model predicts incorrectly.

Method	P	R	F
M^3N (Qian and Liu, 2013)	-	-	84.1
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
Attention-based (Wang et al., 2016)	91.6	82.3	86.7
LSTM-NCM (Lou and Johnson, 2017)	-	-	86.8
Transition-based (Wang et al., 2017)	91.1	84.1	87.5
OURS	94.5	84.1	**89.0**

Table 9: Comparison with previous state-of-the-art methods on the test set of English Switchboard.

with the baseline "T-TDD" model with a best 52.8% F-score. The performance on Chinese is much lower than that on English Switchboard. Due to the high cost of data annotation, the Chinese training set is of small magnitude. Hence the training of the model is not sufficient. Meanwhile, consistent with previous researches (Wang et al. 2017), we find Chinese disfluencies have more complicated structure, and the proportion of repair type disfluency and restart type disfluency is much higher, which increases the difficulty of detection. In addition, the performance of Chinese spoken language segmentation is poor. However, using the character modeling unit loses the very strong chunk-level features inside words. The fitting ability of the model declines with a relatively small vocabulary. So the gap between the performance on Chinese and English Switchboard does make sense.

Our	Dev			Test		
Methods	P	R	F	P	R	F
T-TDD	82.4	42.4	56.0	78.2	37.9	51.1
+DP-MLE	86.2	41.9	56.4	86.2	37.3	52.0
+DAE	73.5	47.6	57.8	72.0	40.8	52.1
+NAT	76.6	46.9	58.2	77.2	40.1	**52.8**

Table 10: Performance on Chinese annotated data.

Our model achieves consistent performance in both Chinese and English experiments. The results of the base "T-TDD" model have significantly exceeded the current best system. This proves the superiority of the translation mechanism in the sequence modeling task. We add penalties to the loss of the deletion operation, making the model a bias towards precision, hence an increase in precision and a drop in recall. Thus we can flexibly adjust the precision, recall, and f-value according to the actual scene. DAE initialization can improve the model's performance notably on English Switchboard. This shows that the damaged sentences we build can better simulate the phenomenon of transcript disfluencies. The introduction of the "heredity" leads to non-autoregressive models, and the experimental results show that the performance of the "+NAT" model has significantly improved compared to the base model. We conjecture it benefits from the consistency of training and inferring processes in the non-autoregressive model, which is inconsistent in autoregressive model.

Conclusion

We propose a general training framework for adapting the NMT model to TDD task. Several extended modules and training techniques are proposed to improve the performance, including the constrained decoding, denoising autoencoder initialization and a TDD-specific training object. And we extend the TDD model with non-autogressive decoding based on the idea of non-autogressive NMT model. Our models achieve the state-of-the-art F-scores on both the commonly used English Switchboard test set and an in-house annotated Chinese data set. Experiments show that translation models have an advantage over all published methods for TDD. We plan to test different NMT models in the proposed training framework, and investigate how the ability of the NMT model correlates with final TDD performance. Besides, we will investigate the effects of other NMT techniques on TDD for further research. We believe our models are applicable to other similar sequence correction tasks, such as grammar error correction, word reordering, and so on.

Acknowledgements

This work is supported by the Beijing Brain Science Project under Grant No. Z181100001518006. We would like to thank Shuang Xu for her preparing data used in this work. Additionally, we also want to thank Linhao Dong and Cheng Yi for their invaluable discussions on this work.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Charniak, E., and Johnson, M. 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*, 1–9. Association for Computational Linguistics.

Cho, E.; Niehues, J.; Ha, T.-L.; and Waibel, A. 2016. Multilingual disfluency removal using nmt. *iwslt2016*.

Ferguson, J.; Durrett, G.; and Klein, D. 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*, 257–262.

Georgila, K. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 109–112. Association for Computational Linguistics.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Gu, J.; Bradbury, J.; Xiong, C.; Li, V. O.; and Socher, R. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Honnibal, M., and Johnson, M. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics* 2(1):131–142.

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

Johnson, M., and Charniak, E. 2004. A tag-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Liu, Y.; Shriberg, E.; Stolcke, A.; Hillard, D.; Ostendorf, M.; and Harper, M. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on audio, speech, and language processing* 14(5):1526–1540.

Lou, P. J., and Johnson, M. 2017. 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)*, volume 2, 547–553.

Meteer, M. W.; Taylor, A. A.; MacIntyre, R.; and Iyer, R. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Neubig, G.; Akita, Y.; Mori, S.; and Kawahara, T. 2012. A monotonic statistical machine translation approach to speaking style transformation. *Computer Speech & Language* 26(5):349–370.

Ostendorf, M., and Hahn, S. 2013. A sequential repetition model for improved disfluency detection. In *INTERSPEECH*, 2624–2628.

Qian, X., and Liu, Y. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 820–825.

Rasooli, M. S., and Tetreault, J. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 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*, 6000–6010.

Wang, S.; Che, W.; Zhang, Y.; Zhang, M.; and Liu, T. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2785–2794.

Wang, F.; Chen, W.; Yang, Z.; Dong, Q.; Xu, S.; and Xu, B. 2018. Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3529–3538.

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

Wu, S.; Zhang, D.; Zhou, M.; and Zhao, T. 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)*, volume 1, 495–503.

Yoshikawa, M.; Shindo, H.; and Matsumoto, Y. 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*, 1036–1041.

Zayats, V.; Ostendorf, M.; and Hajishirzi, H. 2014. Multi-domain disfluency and repair detection. In *Fifteenth Annual Conference of the International Speech Communication Association*.

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

Zwarts, S., and Johnson, M. 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*, 703–711. Association for Computational Linguistics.