A Neural Attention Model for Disfluency Detection

Shaolei Wang, Wanxiang Che, and **Ting Liu**
Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
{slwang, car, tliu}@ir.hit.edu.cn

Abstract

In this paper, we study the problem of disfluency detection using the encoder-decoder framework. We treat disfluency detection as a sequence-to-sequence problem and propose a neural attention-based model which can efficiently model the long-range dependencies between words and make the resulting sentence more likely to be grammatically correct. Our model firstly encodes the source sentence with a bidirectional Long Short-Term Memory (BI-LSTM) and then uses the neural attention as a pointer to select an ordered subsequence of the input as the output. Experiments show that our model achieves the state-of-the-art f-score of 86.7% on the commonly used English Switchboard test set. We also evaluate the performance of our model on the in-house annotated Chinese data and achieve a significantly higher f-score compared to the baseline of CRF-based approach.

1 Introduction

Disfluency detection is the task of detecting the infelicities in spoken language transcripts. The task is important for natural language understanding, since most downstream NLU systems are built on the fluent utterances. Disfluency of a sentence can be categorized into five classes (Wu et al., 2015): uncompleted words, filled pauses (e.g. "uh", "um"), editing terms (e.g. "you know"), discourse markers (e.g. "i mean") and repairs that are discarded, or corrected by its following words. Typically, as shown in Figure 1, a repair type disfluency consists of a filled pause ("um"), a reparandum ("Boston") and an Interregnum("I mean") followed by its repair ("Denver"). The goal of the repair type disfluency detection is to detect the reparandum. The former four classes of disfluencies are easy to detect as they often consist of fixed phrases (e.g. "uh", "you know"). However, the repair type disfluency (see Table 1 for more examples) is more difficult to detect, because reparandums are in arbitrary form. Most of the previous disfluency detection works focus on detecting the repair type disfluencies.

A flight to $\underbrace{\text{um}}_{\text{FP}}$ $\underbrace{\text{Boston}}_{\text{RM}}$ $\underbrace{\text{I mean}}_{\text{IM}}$ $\underbrace{\text{Denver}}_{\text{RP}}$ Tuesday

Figure 1: A sentence with disfluencies annotated in the style of (Shriberg, 1994) and the Switchboard corpus. FP=Filled Pause, RM=Reparandum, IM=Interregnum, RP=Repair. We follow previous works in evaluating the system on the accuracy with which it identifies speech-repairs, marked *reparandum* above.

Modeling long-range dependencies between repair phrases is one of the core problems for disfluency detection. Previous sequence tagging methods (Ferguson et al., 2015; Georgila, 2009; Qian and Liu, 2013) require carefully designed features to capture the information of long distance, but usually suffer from the sparsity problem. Another line of syntax-based disfluency detection works (Honnibal and Johnson, 2014; Wu et al., 2015) try to model the repair phrases on a syntax tree by compressing the unrelated

* Corresponding author: Wanxiang Che

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 278–287, Osaka, Japan, December 11-17 2016.

they/E had/E they/E used/E to/E have/E well they do have television monitors stationed throughout our buildings.
and the/E other/E one/E is/E her husband is in the navy.
they/E in/E fact/E they/E just/E it was just a big thing recently.

Table 1: Sentences in which the former four classes of disfluencies have been removed. "/E" means that the word belong to a reparandum and should be deleted.

phrases and allowing the repair phrases to interact with each other. However, data that both have syntax trees and disfluency annotations are scarce. The performance of syntax parsing models (about 92% on English) also hinders disfluency detection's performance. There are also works that try to use recurrent neural network (RNN), which can capture dependencies at any length in theory, on disfluency detection problem (Hough and Schlangen, 2015). The RNN method treats sequence tagging as classification on each input token and doesn't model the transition between tags which is important in recognizing the repair phrases of multi-words. Another core problem for disfluency detection is to keep the generated sentences grammatical. However, the sequence tagging methods and RNN method have no power of modeling the linguistic structural integrity. The output of syntax-based methods is a syntax tree and can keep the output sentence grammatical in theory, but limited by the performance of syntax parsing models.

In this paper, we address the above challenges by seeing disfluency detection as a sequence-to-sequence problem and presenting a neural attention-based model to learn the conditional probability of the output which is an ordered subsequence of the input, inspired by the work of (Rush et al., 2015; Vinyals et al., 2015). Our model repurposes the attention mechanism (Bahdanau et al., 2014) to create pointers to the input elements. More specifically, we first encode the input sentence with a bidirectional Long Short-Term Memory (BI-LSTM) and then use the neural attention as a pointer to select a member of the input sequence as the output. In the decoding process, we use the attention mechanism only over the candidate words within a window and select the word with the maximum attention weight in each step. Once a word is selected, all the candidate words before it will be labeled as reparandum and be deleted.

While our neural attention-based model is structurally simple, it is very suitable for disfluency detection for that (1) taking into account the language model and a constrained vocabulary (words appear in the input sentence) during generation, which makes the resulting sentence more likely to be grammatically correct, and (2) utilizing the global representation of the input for generating, and selecting the word by joint decision with our neural attention mechanism, which can effectively model the long-range dependencies.

Experiments show that our model achieves the state-of-the-art f-score of 86.7% on the commonly used English Switchboard test set. We also evaluate the performance of our model on Chinese annotated data. As there is no public disfluency data in Chinese, we annotate 200k Chinese sentences manually for training and testing. We achieve a 61.4% f-score with more than 7 points gained compared to the CRF-based baseline, showing that our models are robust.

Our original major contributions in this paper include:

- We firstly study the problem of disfluency detection using the encoder-decoder framework.
- We propose a novel neural attention-based model for disfluency detection, and demonstrate its effectiveness on both English Switchboard corpus and in-house annotated Chinese corpus.

2 Background: Neural Models for Sequence-to-sequence Learning

Sequence-to-sequence learning can be expressed in a probabilistic view as maximizing the likelihood of observing the output (target) sequence given an input (source) sequence.

2.1 RNN Encoder-Decoder

RNN-based Encoder-Decoder is successfully applied to real world sequence to sequence tasks, first by (Sutskever et al., 2014; Cho et al., 2014). In the Encoder-Decoder framework, the source sequence $X = [x_1, ..., x_T]$ is converted into a fixed length vector c by the encoder RNN, i.e.

$$h_t = f(x_t, h_{t-1}); \quad c = q(\{h_1, ..., h_T\}) \tag{1}$$

where $h_t \in R^n$ is a hidden state at time t, and c is a vector generated from the sequence of the hidden states. f and q are some nonlinear functions. (Sutskever et al., 2014) used an LSTM as f and $q(\{h_1, ..., h_T\}) = h_T$, for instance.

The decoder is often trained to predict the next word y_t given the context vector c and all the previously predicted words $\{y_1, ..., y_{t-1}\}$. In other words, the decoder defines a probability over the translation y by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^{T} p(y_t \mid \{y_1, ..., y_{t-1}\}, c), \tag{2}$$

where $y = \{y_1, ..., y_{t-1}\}$. With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c), \tag{3}$$

where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t, and s_t is the hidden state of the RNN. Note that y_t belongs to a fixed output vocabulary dictionary.

2.2 The Attention Mechanism

The attention mechanism was first introduced to sequence to sequence (Bahdanau et al., 2014) to release the burden of summarizing the entire source into a fixed-length vector as context. Instead, the attention uses a dynamically changing context c_t in the decoding process. c_t at each output time t is computed as follows:

$$\begin{aligned} u_{tj} &= v^T tanh(W_1 h_j + W_2 s_{t-1}) \quad j \in (1, ..., n) \\ a_{tj} &= \frac{exp(u_{tj})}{\sum_{k=1}^{T} exp(u_{tk})} \quad j \in (1, ..., n) \\ c_t &= \sum_{j=1}^{n} a_{tj} h_j \end{aligned} \tag{4}$$

where s_{t-1} is the decoder hidden states for time $t-1$. v, W_1, and W_2 are learnable parameters of the model. Lastly, c_t and s_{t-1} are concatenated as the hidden states which will be used for making predictions and fed to the next time step in the decoder RNN. We call u_{tj} a relevance score, or an alignment weight of the j-th input. Note that the vanilla attention mechanism compute the relevance among all the input each step.

3 Model Description

Disfluency detection requires that the output should be an ordered subsequence of the input. The vanilla sequence-to-sequence approach(Sutskever et al., 2014) requires the size of the output dictionary to be fixed a priori, which means that it can only generate one word of the output dictionary at any step. Because of this constraint, we cannot directly apply the vanilla sequence-to-sequence framework to disfluency detection, since it (1) may generate a word out of the input, (2) can't generate the word out of the fixed output dictionary but in the input when applying the model on the test set, and (3) has no power of modeling the order of the generated words. To address the constraint, we propose a new attention-based model as illustrated in Figure 2. Our model is still an encoder-decoder framework in a slightly generalized sense.

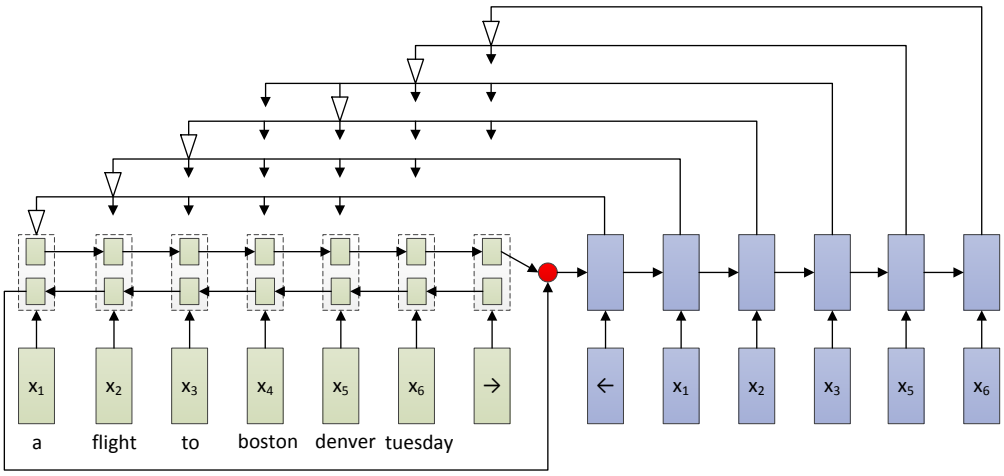

Figure 2: our attention-based network: An encoding RNN converts the input sequence to a vecter that is fed to the generating network. At each step, the generating network produces a vector that modulates a content-based attention mechanism over the words in a window. The word with maximum attention value is selected.

3.1 Input representation

To represent each input token, we use four vectors: a learned word embedding w; a fixed word embedding $\widetilde{w}$; a learned POS-tag embedding p; a hand-crafted feature representation d. The four vectors are concatenated together, transformed by a matrix V and fed to a rectified layer to learn feature combination, as

$$x = max\{0, V[\widetilde{w}; w; p; d] + b\} \tag{5}$$

where $[\widetilde{w}; w; p; d]$ means the concatenation.

We extract two kinds of hand-crafted discrete features (as shown in Table 2) for each token in a sentence and incorporate them into our neural networks by translating them into the 0-1 vector d. The dimension of d is 78, which equals to the number of discrete features. For a token x_t, d_i fires if x_t matches the i-th pattern of the feature templates. The duplicate features care whether x_t has a duplicated word/POS-tag in certain distance. The similarity features care whether the surface string of x_t resembles its surrounding words.

duplicate features
$Duplicate(i, w_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if w_i equals w_{i+k}, the value is 1, others 0
$Duplicate(p_i, p_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if p_i equals p_{i+k}, the value is 1, others 0
$Duplicate(w_i w_{i+1}, w_{i+k} w_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $w_i w_{i+1}$ equals $w_{i+k} w_{i+k+1}$, the value is 1, others 0
$Duplicate(p_i p_{i+1}, p_{i+k} p_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $p_i p_{i+1}$ equals $p_{i+k} p_{i+k+1}$, the value is 1, others 0
similarity features
$fuzzyMatch(w_i, w_{i+k}), k \in \{-1, +1\}$: $similarity = num\_same\_letters/(len(w_i) + len(w_{i+k}))$. if $similarity > 0.8$, the value is 1, others 0

Table 2: Discrete features used in our neural attention-based networks. p is the POS tag. w is the word. Duplicate indicates if the two units are same. fuzzyMatch indicates the similarity of two words.

3.2 Encoder

We use bidirectional LSTM-based RNN which consists of forward and backward RNNs to transform the source sequence into a series of hidden states, with each hidden state h_t corresponding to word x_t. The

Algorithm 1 : Learning algorithm of our neural attention-based model for disfluency detection

Function: Training instance $(x_i, y_i)_{i=1}^{N}$ {y_i is the positions of the words selected from x_i}

for $e = 1, T$ **do**
  for $i = 1, N$ **do**
    $MAX\_DISF\_LEN \leftarrow 10$, $len \leftarrow length(x_i)$
    $(h_1, ..., h_{len}) \leftarrow encode(x_i)$
    $Decode\text{-}LSTM \leftarrow initial\_lstm(h_1, h_{len})$
    $log\_probs \leftarrow \{\}$
    $p \leftarrow 0, t \leftarrow 0$
    while $p < len$ **do**
      $Attention\_Start \leftarrow p, Attention\_End \leftarrow min\{p + MAX\_DISF\_LEN, len\}$
      $attention \leftarrow []$
      for k in $(Attention\_Start, Attention\_End)$ **do**
        compute the attention weight u_{tk} using Equation 6
        APPEND(attention, u_{tk})
      end for
      $position\_of\_correct = y_i^t$, $position\_of\_predict = p + attention.index(max\{attention\}))$
      APPEND(log_probs, $neg\_log\_softmax(attention, position\_of\_correct - p)$)
      $t \leftarrow t + 1, p \leftarrow position\_of\_correct + 1$
      $Decode\text{-}LSTM$.input($x_i^{position\_of\_correct}$)
    end while
    Stochastic gradient descent update on the loss $\lambda = sum(log\_probs)$
  end for
end for

forward RNN reads the input sequence $x = (x_1, ..., x_T)$ in a forward direction, resulting in a sequence of hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_T})$. The backward RNN reads x in an opposite direction and outputs $(\overleftarrow{h_1}, ..., \overleftarrow{h_T})$. We concatenate a pair of the hidden states at each time step to build a sequence of annotation vectors $(h_1, ..., h_T)$ where $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. Each annotation vector h_j encodes the information about the j-th word with respect to all the other surrounding words in the sentence.

3.3 Attention-based Decoder

We address the constraint of the fixed output dictionary in vanilla sequence-to-sequence approaches by modifying the decoder framework. The mainly changes are (as shown in Algorithm 1) as bellows:

- We select a word from the candidate words $(x_i, ...x_j)$ rather than on a fixed output dictionary in each step. This mechanism keeps all the generated words come from the input sentence and has the ability to generate a word that occurs in the test sentence but not in the train sentence.
- Once a word x_k is selected, all the words $(x_i, ..., x_{k-1})$ will be deleted and the candidate words in the next step will be $(x_{k+1}, ...x_l)$. This mechanism keeps the selected words in the order that they appear in the input sentence.

Now the key question is how to choose the correct word from the candidate words $(x_i, ...x_j)$ in each step t. We achieve it by selecting the word x_k with the highest relevance weight u_{tk}, which is computed by repurposing the attention mechanism of Section 2.2. The attention mechanism of Section 2.2 computes the relevance weight vector u_t on all the input sentence $(x_1, ..., x_T)$ in each step t. This mechanism is unsuited to our task since we only need to consider the relevance weights of the subsequence $(x_i, ...x_j)$ in each step t. Hence, we make a modification of the attention mechanism in Equation 4 and model $p(y_t \mid \{y_1, ..., y_{t-1}\}, c)$ in Equation 3 as follows:

$$\begin{aligned} u_{tk} &= v^T tanh(W_1 h_k + W_2 s_{t-1} + W_3 d_{k-i}) \quad k \in (i, ..., j) \\ p(y_t \mid \{y_1, ..., y_{t-1}\}, c) &= softmax(u_t) \end{aligned} \tag{6}$$

where d_{k-i} is the embedding of the distance between previous selected word x_{i-1} and the word x_k. The distance information is very important for our model, since the latter word is more likely to be selected when two words have similar context embeddings in disfluency detection. Once x_k is selected, we will use it to update the state of the decoder RNN and select another word from the words ranging from

$(x_{k+1}, ..., x_l)$ (as shown in Figure 2). To learn the parameters of our neural attention-based model, we minimize the negative log-probability of the output sequence over the input data $\{(x_i, y_i)\}_{n=1}^{N}$ during training:

$$-\sum_{i=1}^{N} log(p(y_i|x_i)) = -\sum_{i=1}^{N} log(\prod_{t=1}^{T} softmax(u_t)) \quad (7)$$

The detailed learning algorithm is shown in Algorithm 1.

4 Network training

4.1 Parameters

Pretrained word embedding. There are lots of methods for creating word embeddings. As (Dyer et al., 2015) does, we use a variant of the skip n-gram model introduced by (Ling et al., 2015), named "structured skip n-gram", where a different set of parameters are used to predict each context word depending on its position relative to the target word. The hyperparameters of the model are the same as in the skip n-gram model defined in word2vec (Mikolov et al., 2013). We set the window size to 5, and use a negative sampling rate to 10. The AFP portion of English Gigaword corpus (version 5) is used as the training corpus.

Hyper-Parameters. The LSTMs of both encoder and decoder has two hidden layers and their dimensions are set to 100. Pretrained word embeddings have 100 dimensions and the learned word embeddings have also 100 dimensions. Pos-tag embeddings have 12 dimensions. The dimension of distance d in Equation 6 is set to 8.

Parameter initialization. The learned parameters in the neural networks are randomly initialized with uniform samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of rows and columns in the parameter structure.

4.2 Optimization Algorithm

Parameter optimization. Parameter optimization is performed with stochastic gradient descent (SGD) with an initial learning rate of $\eta_0 = 0.1$ and a gradient clipping of 5.0. The learning rate is updated on each epoch of training as $\eta_t = \eta_0/(1 + \rho t)$, in which t is the number of epoch completed and the decay rate $\rho = 0.05$.

Early Stopping. We use early stopping (Giles, 2001) based on performance on dev sets. The best parameters appear at around 12 epochs, according to our experiments.

Dropout Training. To reduce overfitting, we apply the dropout method (Srivastava et al., 2014) to regularize our model. We apply dropout not only on input and output vectors of LSTM, but also between different hidden layers of LSTM. We observe a significant improvement on model performances after using dropout.

Unknown Word Handling. As described in section 3, the input layer contains a learned vector representation for the word w and a corresponding fixed pretrained vector representation $\widetilde{w}$. We randomly replace the singleton word in the training data with the UNK token during training, but keep corresponding $\widetilde{w}$ unchanged. This technique can deal with out-of-vocabulary words.

5 Experiments

5.1 Settings

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. We extract the repair disfluencies according to the EDITED label in the Switchboard corpus. Following (Honnibal and Johnson, 2014), we lower-case the text and remove all punctuations and partial words[1]. We also discard all the 'um' and 'uh' tokens and merge 'you know'

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

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	93.8%	77.7%	85.0%	92.0%	74.5%	82.3%
Attention-based method	93.0%	81.6%	86.9%	91.6%	82.3%	86.7%

Table 3: Experiment results on the development and test data of English Switchboard data.

Method	**P**	**R**	**F1**
Attention-based	91.6%	82.3%	**86.7**%
M^3N (Qian and Liu, 2013)	-	-	84.1%
Joint Parser (Honnibal and Johnson, 2014)	-	-	84.1%
semi-CRF (Ferguson et al., 2015)	90.1%	80.0%	84.8%
UBT (Wu et al., 2015)	90.3%	80.5%	85.1%

Table 4: Comparison of our neural attention-based model with the previous state-of-the-art methods on the test set of English Switchboard data.

and 'i mean' into single token. Automatic POS-tags generated from pocket_crf (Qian and Liu, 2013) are used as POS-tag in our experiments.

No public Chinese corpus is available now. For our Chinese experiments, we collect about 200k spoken sentences from minutes of meetings and annotate them with only disfluency annotations according to the guideline proposed by (Meteer et al., 1995). We respectively select about 20k sentences for development and testing. The rest are used for training. We use the word segmentation and POS-tag tools provided by the Language Technology Platform (Che et al., 2010) for preprocessing the original data in our experiments.

Metric. Following previous works (Ferguson et al., 2015; Wu et al., 2015), token-based precision (P), recall (R), and f-score (F1) are used as the evaluation metrics.

5.2 Performance of disfluency detection on English Swtichboard corpus

We build a baseline system using the Conditional Random Field (CRF) model. The hand-crafted discrete features of our CRF refer to those in (Ferguson et al., 2015). Table 3 shows the result of our model on both the development and test set.

We compare our neural attention-based model to four previous top performance systems. Our model outperforms state-of-the-art work and achieves a 86.7% f-score as shown in Table 4. Our model achieves 1.6 point improvements over UBT (Wu et al., 2015), which is the best syntax-based method for disfluency detection. The best performance by linear statistical sequence labeling methods is the semi-CRF method (Ferguson et al., 2015), achieving a 84.8% F1 score without leveraging prosodic features. Our model beats the semi-CRF model, obtaining 1.9 point improvements. Note that our method performs better than previous methods not only on precision , but also on recall. The comparison shows that our model is a good solution to disfluency detection.

5.3 Ablation Test

As described in section 3.1, we extract two kinds of hand-crafted discrete features for each token in a sentence. The duplicate features are extracted, because the reparandum(RM) and the following repair(RP) often share some identical words/POS-tags. In some cases, the word in a reparandum is the singular or plural of the word in the following repair, hence we extract the similarity features.

To test the individual effectiveness of duplicate features and similarity features, we conduct feature ablation experiments for our neural attention-based model. Table 5 shows the result. We can see that both the two kinds of features contribute to the performance improvements of disfluency detection and we can achieve a higher performance by integrating all of them into our model. This indicates that duplicate features and similarity features are both important to the neural model and they provide different kinds of information for disfluency detection.

Method	Dev			Test		
	P	R	F1	P	R	F1
Attention-based	93.0%	81.6%	86.9%	91.6%	82.3%	**86.7**%
- Duplicate	93.6%	78.0%	85.1%	90.8%	76.3%	82.9%
- Similarity	93.3%	82.0 %	87.3%	92.3%	80.9%	86.2%
- Duplicate - Similarity	93.2%	76.8%	84.2%	89.4%	76.2%	82.3%

Table 5: Results of feature ablation experiments on English Switchboard data.

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	76.5%	42.0%	54.2%	75.9%	41.6%	53.8%
Attention-based method	83.7%	50.6%	63.1%	82.4%	48.9%	61.4%

Table 6: Disfluency detection performance on Chinese annotated data.

5.4 Performance of disfluency detection on Chinese annotated corpus

In addition to English experiments, we also apply our method on Chinese annotated data. As there is no standard Chinese corpus, no Chinese experimental results are reported in (Honnibal and Johnson, 2014) and (Qian and Liu, 2013). We only use the CRF-based labeling model as our baselines. Table 6 shows the results of Chinese disfluency detection. Our models outperform the CRF model by more than 7 points on f-score which shows that our method is more effective.

6 Related Work

Most related works on disfluency detection are aimed at detecting repair type of disfluencies. (Johnson and Charniak, 2004) proposed a TAG-based noisy channel model for disfluency detection. The TAG model was used to find rough copies. Following the work of (Johnson and Charniak, 2004), (Zwarts and Johnson, 2011) extended the TAG model using minimal expected f-loss oriented n-best reranking with additional corpus for language model training. (Qian and Liu, 2013) proposed a muiti-step learning method using weighted max-margin markov network (M^3N). They showed that M^3N model outperformed many other labeling models such as CRF model. (Ferguson et al., 2015) used the Semi-Markov CRF model for disfluency detection and achieved high f-score by integrating prosodic features.

Many syntax-based approaches have been proposed which jointly perform dependency parsing and disfluency detection. (Lease and Johnson, 2006) involved disfluency detection in a PCFG parser to parse the input along with detecting disfluencies. (Rasooli and Tetreault, 2013) designed a joint model for both disfluency detection and dependency parsing. (Honnibal and Johnson, 2014) presented a new joint model by extending the original transition actions with a new "Edit" transition. This model achieved good performance on both disfluency detection and parsing. (Wu et al., 2015) proposed a right-to-left transition-based joint method and achieved the state-of-the-art performance compared with previous syntax-based approaches.

RNN had been used to disfluency detection. (Hough and Schlangen, 2015) explored incremental detection, with an objective that combines detection performance with minimal latency. This approach achieved worse performance compared with other works for the latency constraints. (Cho et al., 2013) used word embeddings learned by an RNN as features in a CRF classifiers.

7 Conclusion and Future Work

In this paper, we firstly regard the disfluency detection as a sequence-to-sequence problem and propose a neural attention-based model. Our model break the constraint of vanilla sequence-to-sequence method that the size of the output dictionary should be fixed a priori by modifying the structure of decoder. Experimential results show that our method achieves the best reported performance on the commonly used English Switchboard corpus and a better performance than CRF model on in-house Chinese annotated data.

In the future, we will try to incorporate character-based representations into the encoder model. We would also like to jointly model disfluency detection and automatic punctuation using some neural network.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61370164 and 61632011.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *In ICLR 2015, arXiv preprint arXiv:1409.0473*.

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.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Coling 2010: Demonstrations*, pages 13–16, Beijing, China, August. Coling 2010 Organizing Committee.

Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2013. Crf-based disfluency detection using semantic features for german to english spoken language translation. *IWSLT, Heidelberg, Germany*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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 334–343. Association for Computational Linguistics, July.

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. Association for Computational Linguistics.

Kallirroi Georgila. 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*, pages 109–112. Association for Computational Linguistics.

Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 33. Association for Computational Linguistics.

Matthew Lease and Mark Johnson. 2006. Early deletion of fillers in processing conversational speech. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 73–76. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *EMNLP*, pages 124–129.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September. Association for Computational Linguistics.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

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. Association for Computational Linguistics.

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.