Enhancing Neural Disfluency Detection with Hand-crafted Features

Shaolei Wang, Wanxiang Che, Yijia Liu, and Ting Liu

Research Center for Social Computing and Information Retrieval
School of Computer Science and Technology
Harbin Institute of Technology, China
{slwang,car,yjliu,tliu}@ir.hit.edu.cn

Abstract. In this paper, we apply a bidirectional Long Short-Term Memory with a Conditional Random Field to the task of disfluency detection. Long-range dependencies is one of the core problems for disfluency detection. Our model handles long-range dependencies by both using the Long Short-Term Memory and hand-crafted discrete features. Experiments show that utilizing the hand-crafted discrete features significantly improves the model's performance by achieving the state-of-the-art score of 87.1% on the Switchboard corpus.

Keywords: Disfluency detection, BI-LSTM-CRF, Discrete features, Continuous neural features

1 Introduction

Disfluencies are common in automatic speech recognition (ASR). Detecting disfluencies 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: 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 (see Fig. 1 for a disfluency example with filled pauses, discourse markers and repair words). The former four classes of disfluencies are easy to detect as they often consist of fixed phrases (e.g. "uh", "you know"). While, the repair disfluencies (see Table 1) are more difficult to detect, because they are in arbitrary form [25]. Most of the previous disfluency detection works focus on detecting the repair disfluencies.

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

Fig. 1. A sentence with disfluencies annotated in the style of [23] and the Switchboard corpus. FP=Filled Pause, RM=Reparandum, IM=Interregnum, RP=Repair. We follow previous work 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 detecting the repair disfluencies. Previous sequence tagging methods [5, 6, 21]

Table 1. Repair disfluency sentences. "/E" means that the word belong to a repair and should be deleted.

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.

require carefully designed features to capture information of long distance, but usually suffer sparsity problem. Another line of syntax-based disfluency detection work [9, 25] try to model the repair phrases on a syntax tree by compressing the unrelated phrases and allowing repair phrases to interact with each other. However, data that both have syntax tree and disfluency annotation is scarce. The performance of syntax parsing models (about 92% on English) also hinders the disfluency detection's performance. Recurrent neural network (RNN), which can capture dependencies at any length, has been successfully applied to many NLP tasks, including NER [11] and opinion mining [12]. There is also work that tried to use RNN in disfluency detection problem [10]. However, in term of solving long-range dependencies, [10] overly relies on the RNN itself and doesn't adopt the hand-craft discrete features which has shown effective in previous work [26]. Further more, [10] treats sequence tagging as classification on each input token and doesn't model the transition between tags which is important for recognizing the repair phrases of multi-words.

In this paper, we follow the sequence tagging work and combine CRF and RNN to solve the disfluency detection problem. More specifically, we use a bidirectional Long Short-Term Memory (BI-LSTM) network to encode the words and feed the output to a linear-chained CRF to model the probability of tag sequence, thus result in a BI-LSTM-CRF model. We also study the problem of adopting the hand-crafted discrete features into BI-LSTM-CRF model. We evaluate our model on the English Switchboard corpus. The results show that our model outperforms previous stat-of-the-art systems by achieving a 87.1% F-score.

2 BI-LSTM-CRF Model

RNN has been employed to produce state-of-the-art results on a variety of tasks [11, 12]. It takes a sequence of input $(x_1, x_2, ..., x_n)$ and returns a sequence $(h_1, h_2, ..., h_n)$ in which h_t encodes the information in ranging from x_1 to x_t. In theory, RNN can capture dependencies of the inputs at any length with a hidden layer that memorizes the historical information. Unfortunately, in practice, it fails due to the gradient varnishing/exploding problems [1, 20]. Long Short-Term Memory (LSTM) [8] as a variant of RNN is designed to cope with the gradient varnishing problems. It addresses the varnishing problems with an extra memory "cell" c_t that is constructed as a linear combination of the previous state and input signal. LSTM cells process the inputs with three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. A LSTM memory cell is calculated as:

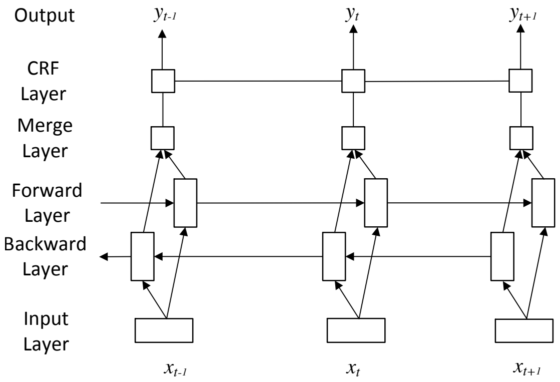

Fig. 2. Main architecture of the BI-LSTM-CRF. Outputs of the hidden layer are given to a CRF Layer after passing through a Merge Layer.

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
c_t &= (1 - i_t) \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product.

For disfluency detection, it is difficult to predict a word as disfluency by only considering its past contexts, because disfluent phrases of this word can occur before or after it. A good model should access to both the past and the future contexts for disfluency detection. In this paper, we encode the past and the future information with BI-LSTM [15]. In the BI-LSTM, the past information is represented with a forward LSTM and the future with a backward LSTM respectively. The hidden states from these two LSTMs are concatenated to form the final output.

h_t can be used to make independent classification on each token to do the disfluency tagging. But such classification scheme is limited when there are strong dependencies between output tags. Disfluency detection is one of the tasks that strong dependencies exist between tags because one repair phase can have multi-words. Instead of modeling tagging decisions independently, we model them jointly using a CRF layer [11, 14, 15, 18]. Fig. 2 illustrates the architecture of BI-LSTM-CRF in detail. For an input sentence $X = (x_1, x_2, ..., x_n)$, we define $P \in R^{n \times k}$ to be the matrix of scores output by the BI-LSTM network, where k is the number of tags and $P_{i,j}$ corresponds to the score of x_i tagged as j. We also define $A \in R^{k \times k}$ to be the matrix of transition scores that $A_{i,j}$ is the score of a transition from tag i to tag j. For a sequence of tags $y = (y_1, y_2, ..., y_n)$, its score is defined as

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}$$

and its probability is defined as

Table 2. Discrete features used in the BI-LSTM and BI-LSTM-CRF 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

duplicate features
$Duplicate(w_i, w_{i+k}), -15 \leq k \leq +15$ 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

$$p(y|X) = \frac{exp\{s(X,y)\}}{\sum_{\widetilde{y} \in Y_X} exp\{s(X,\widetilde{y})\}}$$

where Y_X represents all possible tag sequences of the input sequence X. To learn parameters of the BI-LSTM and the transition matrix A, we minimize the negative log-probability of the correct tag sequence over the input data $\{(X^{(n)}, y^{(n)})\}_{n=1}^{N}$ during training:

$$-\sum_{n=1}^{N} log(p(y^{(n)}|X^{(n)})) = -\sum_{n=1}^{N} \left(s(X^{(n)}, y^{(n)}) - log \left(\sum_{\widetilde{y} \in Y_{X^{(n)}}} exp\{s(X^{(n)}, \widetilde{y})\} \right) \right)$$

While decoding, we predict the highest-scored output sequence using dynamic programming.

3 Utilizing Hand-crafted Features

Previous studies [5, 21] show that hand-crafted features are very effective for achieving good disfluency detection performance, especially those features that capture the duplication between phrases. In this paper, We use two kinds of hand-crafted discrete features as shown in Table 2 and incorporate them into our neural networks by translating them into a 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 in certain distance. The similarity features care whether the surface string of x_t resembles its surrounding words.

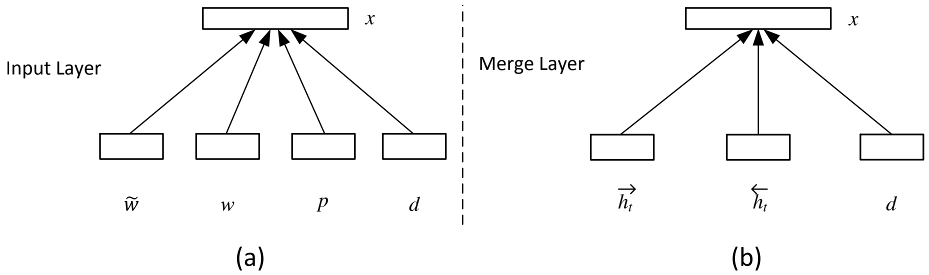

Fig. 3. Two methods of combining discrete features and continuous neural features. $\widetilde{w}$ is a pretrained embedding of current word. w is a learned embedding of current word; p is a learned embedding of current POS tag of the word; d is a continuous embedding of discrete features. $\overrightarrow{h_t}$ is the output of the forward LSTM. $\overleftarrow{h_t}$ is the output of the backward LSTM

In this paper, we try two different methods to incorporate the hand-crafted discrete features d. In the first method, we treat d as an input to the LSTM along with the fintune word embedding w, fixed word embedding $\widetilde{w}$ and POS-tag embedding p. These four parts are concatenated together, transformed by a matrix V and fed to a rectified layer to learn feature combination, as

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

where $[\widetilde{w}; w; p; d]$ means the concatenation. This method is shown in Fig. 3(a). By feeding d into LSTM, we allow to encode a long-range of hand-crafted features.

In the second method, we treat d as input to the CRF-layer, along with the outputs of the forward and the backward LSTMs. Formally, it can be calculated as

$$x = max\{0, V[\overrightarrow{h_t}; \overleftarrow{h_t}; d] + b\}$$

where $\overrightarrow{h_t}$ is the output of the forward LSTM, $\overleftarrow{h_t}$ is the output of the backward LSTM. This method is shown schematically as in Fig. 3(b).

4 Network Training

4.1 Parameters

Pretrained word embedding. There are lots of methods for creating word embeddings. As [4] does, we use a variant of the skip n-gram model introduced by [17], 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 [19]. 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 corpora.

Hyper-Parameters. Our BI-LSTM and BI-LSTM-CRF models use two hidden layers for the forward and backward LSTMs whose 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 dimensions of labels in BI-LSTM-CRF are set to 16.

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 [7] 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 [24] to regularize our model. We apply dropout not only on input and output vectors of BI-LSTM, but also between different hidden layers of BI-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. We conduct our experiments on the English Switchboard corpus. Following the experiment settings in [2, 9, 25], 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 [9], 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’ and ‘i mean’ into single token. Automatic POS tags generated from pocket_crf [21] are used as POS-tag in our experiments.

Metric and Tagging Schemes. Following previous works [5, 25], token-based precision (P), recall (R), and F-score (F1) are used as the evaluation metrics. We use BIESO tagging scheme in our experiments.

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

Table 3. Experimental results on the development and test data for investigating the effect of CRF-layer.

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	93.8%	77.7%	85.0%	92.0%	74.5%	82.3%
BI-LSTM	94.0%	71.0%	80.9%	93.2%	70.3%	80.1%
BI-LSTM-CRF	94.8%	71.1%	81.3%	94.0%	71.3%	81.1%

5.2 Effect of CRF-layer

To investigate the effect of combining BI-LSTM and CRF, we build the following systems:

- CRF: a baseline system and the hand-crafted discrete features are the same with those in [5]. Note that the hand-crafted features in Table 2 are contained in the CRF model
- BI-LSTM: model that encodes word and POS-tag with BI-LSTM and use the output of the merge layer in Fig. 2 as features for a logistic classifier.
- BI-LSTM-CRF: model that takes the same input as BI-LSTM but compute the probability of tag sequence with a CRF layer.

Table 3 shows that BI-LSTM-CRF achieves significant improvements over BI-LSTM by adding CRF layer for joint decoding. This result demonstrates the necessary of handling the label bias problem in disfluency detection. We also note that the CRF with rich hand-crafted discrete features outperforms the BI-LSTM-CRF with only continuous neural features. A natural question that arises from this contrast is whether hand-crafted discrete features and continuous neural features can be integrated for better accuracies. We will study it in next section.

5.3 Effect of Hand-crafted Features

Based on the BI-LSTM-CRF, we compare two methods of combining hand-crafted discrete features and continuous neural features. In Table 4, "-HIDDEN" means combining hand-crafted discrete features with hidden outputs of BI-LSTM. "-INPUT" means combining discrete feature embeddings with POS-tag and word embedding to the input layer. From the result of Table 4, both the BI-LSTM-CRF-HIDDEN and the BI-LSTM-CRF-INPUT outperform the BI-LSTM-CRF by a large margin. The result confirms that hand-crafted discrete features and continuous neural features can be integrated to achieve better performance. By comparing two methods of combining hand-crafted discrete features, BI-LSTM-CRF-INPUT achieves better performance than BI-LSTM-CRF-HIDDEN. We attribute it to the fact that BI-LSTM-CRF-INPUT allow the dense and hand-crafted discrete features to interact with each other by non-linear transformation and encode the discrete features of long range with LSTM at the same time. We also test the effect of hand-crafted features on BI-LSTM model and get similar results with BI-LSTM-CRF.

Table 4. Experimental results on the development and test data for investigating the effect of integrating hand-crafted features.

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	93.8%	77.7%	85.0%	92.0%	74.5%	82.3%
BI-LSTM	94.0%	71.0%	80.9%	93.2%	70.3%	80.1%
BI-LSTM-HIDDEN	93.1%	78.6%	85.2%	92.9%	76.7%	84.0%
BI-LSTM-INPUT	93.2%	82.6%	87.6%	92.7%	80.6%	86.2%
BI-LSTM-CRF	94.8%	71.1%	81.3%	94.0%	71.3%	81.1%
BI-LSTM-CRF-HIDDEN	93.4%	78.6%	85.4%	91.7%	78.6%	84.6%
BI-LSTM-CRF-INPUT	92.8%	84.3%	88.3%	91.0%	83.6%	**87.1**%

Table 5. Comparison of our BI-LSTM-CRF with the previous state-of-the-art methods on the test set.

Method	**P**	**R**	**F1**
BI-LSTM-CRF-INPUT	91.0%	83.6%	**87.1**%
M^{3}N [21]	-	-	84.1%
Joint Parser [9]	-	-	84.1%
semi-CRF [5]	90.1%	80.0%	84.8%
UBT [25]	90.3%	80.5%	85.1%

5.4 Final Result and Comparsion with Previous Work

We compare our best model (BI-LSTM-CRF-INPUT) to four previous top performance systems. Our method outperforms state-of-the-art work and achieves a 87.1% F-score as shown in Table 5. Our model achieves 2 point improvements over UBT [25], which is the best syntax-based method for disfluency detection. The best performance by linear statistical sequence labeling methods is the semi-CRF method [5], achieving a 84.8% F1 score without leveraging prosodic features. Our model beats the semi-CRF model, obtaining 2.3 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.

5.5 Ablation Test

To test the individual effectiveness of duplicate features and similarity features, we conduct feature ablation experiments for the BI-LSTM-CRF-INPUT model. Table 6 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 BI-LSTM-CRF. This indicates that duplicate features and similarity features are both important to the BI-LSTM-CRF and they provide different kinds of information for disfluency detection.

Table 6. Results of feature ablation experiments on BI-LSTM-CRF-INPUT models.

Method	Dev			Test		
	P	R	F1	P	R	F1
BI-LSTM-CRF	94.8%	71.1%	81.3%	94.0%	71.3%	81.1%
+ Duplicate	94.5%	80.5%	86.9%	93.2%	79.7%	86.0%
+ Similarity	94.9%	73.9 %	83.1%	94.1%	73.6 %	82.6%
+ Duplicate + Similarity	92.8%	84.3%	88.3%	91.0%	83.6 %	**87.1**%

6 Related Work

Most related works on disfluency detection are aimed at detecting repair type of disfluencies. [13] proposed a TAG-based noisy channel model for disfluency detection. The TAG model was used to find rough copies. Following the work of [13], [27] extended the TAG model using minimal expected f-loss oriented n-best reranking with additional corpus for language model training. [21] 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. [5] 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. [16] involved disfluency detection in a PCFG parser to parse the input along with detecting disfluencies. [22] designed a joint model for both disfluency detection and dependency parsing. [9] 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. [25] 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. [10] 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. [3] used word embeddings learned by an RNN as features in a CRF classifiers.

7 Conclusion and Future Work

In this paper, we have explored an application of BI-LSTM-CRF networks to the task of disfluency detection. Our method explores the combination of hand-crafted discrete features and continuous neural features. Experimential result shows that our method achieves the best reported performance on the English Switchboard corpus.

In the future, we will try to model the task of disfluency detection as a sequence to sequence learning problem and 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

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 61133012 and 61370164.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on 5(2), 157–166 (1994)
2. Charniak, E., Johnson, M.: 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. pp. 1–9. Association for Computational Linguistics (2001)
3. Cho, E., Ha, T.L., Waibel, A.: Crf-based disfluency detection using semantic features for german to english spoken language translation. IWSLT, Heidelberg, Germany (2013)
4. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: 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). pp. 334–343. Association for Computational Linguistics (July 2015)
5. Ferguson, J., Durrett, G., Klein, D.: 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. pp. 257–262. Association for Computational Linguistics (2015)
6. Georgila, K.: 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. pp. 109–112. Association for Computational Linguistics (2009)
7. Giles, R.C.S.L.L.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference. vol. 13, p. 402. MIT Press (2001)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
9. Honnibal, M., Johnson, M.: Joint incremental disfluency detection and dependency parsing. Transactions of the Association for Computational Linguistics 2, 131–142 (2014)
10. Hough, J., Schlangen, D.: Recurrent neural networks for incremental disfluency detection. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
11. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. Computer Science (2015)
12. Irsoy, O., Cardie, C.: Opinion mining with deep recurrent neural networks. In: EMNLP. pp. 720–728 (2014)
13. Johnson, M., Charniak, E.: A tag-based noisy channel model of speech repairs. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. p. 33. Association for Computational Linguistics (2004)
14. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)

16. Lease, M., Johnson, M.: Early deletion of fillers in processing conversational speech. In: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers. pp. 73–76. Association for Computational Linguistics (2006)
17. Ling, W., Dyer, C., Black, A., Trancoso, I.: 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. pp. 1299–1304 (2015)
18. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354 (2016)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
20. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. arXiv preprint arXiv:1211.5063 (2012)
21. Qian, X., Liu, Y.: Disfluency detection using multi-step stacked learning. In: HLT-NAACL. pp. 820–825 (2013)
22. Rasooli, M.S., Tetreault, J.R.: Joint parsing and disfluency detection in linear time. In: EMNLP. pp. 124–129 (2013)
23. Shriberg, E.E.: Preliminaries to a theory of speech disfluencies. Ph.D. thesis, Citeseer (1994)
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)
25. Wu, S., Zhang, D., Zhou, M., Zhao, T.: 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). pp. 495–503. Association for Computational Linguistics (2015)
26. Zhang, M., Zhang, Y., Vo, D.T.: Neural networks for open domain targeted sentiment. In: Conference on Empirical Methods in Natural Language Processing (EMNLP 2015) (2015)
27. Zwarts, S., Johnson, M.: 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. pp. 703–711. Association for Computational Linguistics (2011)