Interspeech 2022
18-22 September 2022, Incheon, Korea

Span Classification with Structured Information for Disfluency Detection in Spoken Utterances

Sreyan Ghosh[1,4], *Sonal Kumar*[2], *Yaman Kumar Singla*[3,4,5], *Rajiv Ratn Shah*[4], *S. Umesh*[1]

[1]Speech Lab, Department of Electrical Engineering, IIT Madras,
[2]Cisco Systems, [3]Adobe Media Data Science Research,
[4] IIIT-Delhi, [5] SUNY at Buffalo

gsreyan@gmail.com, skbrahee@gmail.com, ykumar@adobe.com, rajivratn@iiitd.ac.in,
umeshs@ee.iitm.ac.in

Abstract

Existing approaches in disfluency detection focus on solving a token-level classification task for identifying and removing disfluencies in text. Moreover, most works focus on leveraging only contextual information captured by the linear sequences in text, thus ignoring the structured information in the text which is efficiently captured by dependency trees. In this paper, building on the span classification paradigm of entity recognition, we propose a novel architecture for detecting disfluencies in transcripts from spoken utterances, incorporating both contextual information through transformers and long-distance structured information captured by dependency trees, through graph convolutional networks (GCNs). Experimental results show that our proposed model achieves state-of-the-art results on the widely used English Switchboard dataset for disfluency detection and outperforms prior-art by a significant margin. We make all our codes publicly available on GitHub[1].

Index Terms: disfluency detection, computational paralinguistics

1. Introduction

Speech is the natural way to communicate and is still the most preferred medium for communication. Unlike written text, in spoken utterances, humans often fail to pre-mediate what they are going to say, leading to interruption of speech flow. This phenomenon, called disfluency, is a para-linguistic concept that is ubiquitous in human conversations. In the past decade, with Automatic Speech Recognition (ASR) systems achieving near-human performance in transcribing speech-to-text, the use of speech as an input to modern intelligent NLU systems has democratized to an enormous extent. However, these downstream systems, trained on fluent data, can easily get misled due to the presence of disfluencies. Thus disfluency detection and removal can output clean inputs for downstream NLP tasks, like dialogue systems, question answering, and machine translation. Moreover, disfluency detection also finds applications in automatic speech scoring [1, 2].

Figure 1 shows the general structure of a disfluency, whereby it can be divided into 3 main parts, reparandum, an optional interregnum, and repair. Disfluency detection with neural architectures generally focuses on identifying and removing reparandum. Furthermore, reparandum in disfluencies can primarily be categorized into 3 types, repetitions, restarts, and repairs, as shown in Table 1. Repetition occurs when linguistic materials repeat, usually in the form of partial words, words, or short phrases. Substitution occurs when linguistic materials are replaced to clarify a concept or idea. Deletion, also known as false restart, refers to abandoned linguistics materials.

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

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

Table 1: *Different types of disfluencies.*

Type	Example
Repair	[I just + I] enjoy working
Repetition	[it's +{ uh } it's] almost like
Restart	[we would like +] let's go to the
Deletion	[this +{ is } just +] happened yesterday.
Substitution	it's nothing but wood [+ {up here } +] down here.

In the past decade, neural architectures have shown promising results in the task of disfluency detection, where most prior works in this domain primarily report results on SwitchBoard (SWBD) corpus and solve a *sequence-tagging* task. SWBD is a corpus of English telephonic conversations. The task of disfluency detection primarily focuses on detecting reparandums from segmented transcripts of individual SWBD utterances where every word in the utterance transcript is annotated as part of disfluency (or not). Most of these architectures, achieving state-of-the-art performance, solve a token-level classification task, which assigns a label to each token in the input sequence, specifying if the token is part of a disfluency or not. Though token classification is one of the most common choices for Entity Recognition (ER) and has several advantages, including not constraining the output to a single span and incorporating neighboring tag information when implemented with CRFs, we hypothesize that detecting disfluencies, especially reparandums, is different from ER where individual reparandum tokens seldom occur in isolation and make sense only when the entire span of tokens is considered together. Thus, building on the fact that models can make better semantic sense of reparandums out of spans of tokens, we propose to solve a *span classification* task for disfluency detection and devise models that can learn richer *span* representations and not *token* representations.

The primary task of span prediction is to tag a group of one or more contiguous tokens (up to a maximum specified length)

[1] https://github.com/Sreyan88/Disfluency-Detection-with-Span-Classification

10.21437/Interspeech.2022-11242

by aggregating the individual token representations into a single span representation. This kind of learning also has better boundary supervision, and since reparandums generally occur in succession to each other, we posit that this might be an added benefit to using *span classification* for disfluency detection. Also, in recent years there has been a paradigm shift of ER systems for text, from token-level prediction to span prediction, and has seen much success [3].

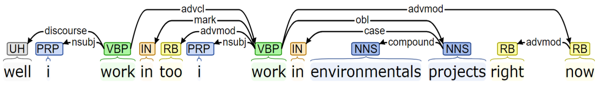


Figure 2: *A sentence annotated with dependency trees*

LSTMs and transformer-based encoders have been two of the most common choices to encode text for the task of disfluency detection [4, 5]. However, these models suffer from certain drawbacks due to their architectural properties. Bi-directional LSTMs fail to capture long-range dependencies [6], and self-attention-based models can only go as far as assigning dynamically learned attention values to each token which has often been found to focus on wrong evidence for task-specific fine-tuning [7]. Thus, integrating structured information such as the interaction between nearby words or long-range word dependencies can aid the model in assigning higher importance to tokens that are syntactically related to the occurrence of another. To do this, we propose to use dependency trees to capture information along multiple dependency arcs between tokens in a sentence. For instance, as we can see in Fig. 2, the span *"i work in"* occurs twice in the sentence however, only the former span should be tagged as disfluent by the system. We hypothesize that the information that the primary nouns of the sentence *"environmentals projects"* only has a dependency path to the latter occurrence of the token *"work"*, might help the system better integrate long-range structured dependencies and thus generate richer representations for disfluent and non-disfluent tokens.

To address the above limitations, in this paper, we propose a novel architecture based on the *span-classification* paradigm, which leverages both contextual information using transformers and structured information obtained through dependency trees using a GCN [8]. More specifically, we concatenate the graph-encoded representation of each token to the transformer output representation of that token via a gating mechanism, post which we calculate the span representation for each span of contiguous words in the input transcript. To the best of our knowledge, this is the first work that leverages structured information in text and solves a span classification task for disfluency detection in spoken utterance transcripts. To sum up, our contributions are as follows:

- We propose a novel architecture for the task of disfluency detection under the *span classification* paradigm of entity recognition and integrate structured information in the text to aid the model in learning better feature representations.
- In addition to achieving state-of-the-art results for disfluency detection on the widely used English SWBD dataset, through thorough quantitative and qualitative analysis we show the importance of each component in our proposed architecture and thereby prove that the proposed learning framework provides better results and mitigates certain drawbacks over solving a *token-classification* task and not using structured information.

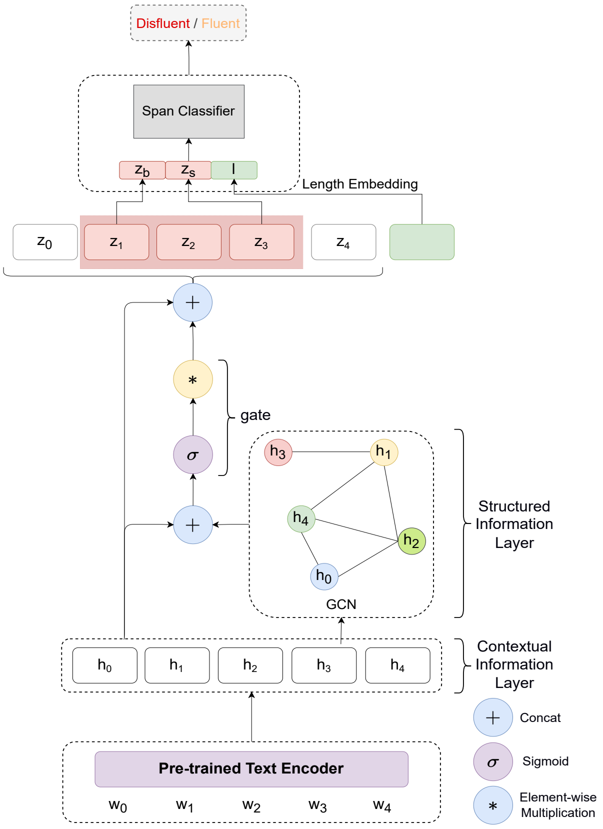


Figure 3: *Proposed architecture*

2. Related Work

Disfluency detection systems can be divided into 4 primary categories. The first one makes use of noisy channel models [9, 10], which require a Tree Adjoining Grammar (TAG) based transducer in the channel model. The second category of models leverages phrase structure, which is often related to transition-based parsing and yet requires annotated syntactic structure [11, 12]. The third is the most common kind and frames the task as a sequence tagging task [13, 14], and the last one employs end-to-end Encoder-Decoder models [5, 15] to detect disfluent segments automatically. In this work, we focus on improving on the third kind, which is the sequence tagging approach. Earlier work on this learning paradigm mostly focused on devising new architecture using Bi-LSTMs, and CRFs [5]. Most recent work on this paradigm tried to alleviate the dependency on human-annotated datasets, where the authors [16, 17] propose self-supervised learning and data augmentation approaches to learning disfluencies and has proven to close the gap with supervised training. Very recently, [18] proposed to solve auxiliary sequence tagging tasks in addition to the original task of tagging disfluent tokens and serves as the current SOTA on the task of disfluency detection.

Though recent years have seen frequent paradigm shifts for the task of ER from *token-level classification* to *span classification*, this framework has been relatively under-studied. [3] conducts a detailed study on the complementary advantages and architectural biases provided by the latter. Additionally, leveraging parse trees to incorporate structured information for improving ER performance has seen growing interest among researchers [19, 20, 21].

3. Proposed Methodology

3.1. Problem Definition

The problem of disfluency detection can be formulated as a sequence tagging task where our primary aim is to identify tokens in spoken utterance transcripts which correspond to a reparandum. We denote the *i*-th sentence with *T* tokens as $s_i = \{w_t | \, t = 1, \cdots \, T\}$, and our complete dataset denoted by $\{s_1,s_2,s_3, \cdots \, s_N\}$, where N is the total number of sentences in our dataset. The corresponding label set is defined by $\{d_1,d_2,d_3, \cdots \, d_N\}$ where $d_{\rm i}$ is the label sequence for each sentence and is denoted by $d_{\rm i} = \{y_t | \, y = 1, \cdots \, T\}$ where $y_t \in \mathcal{Y}$ and $\mathcal{Y} = \{I, O\}$. I here stands for disfluent tokens and O stands for fluent tokens.

3.2. Proposed Architecture

Fig. 3 shows our proposed model architecture. As mentioned earlier, we integrate both structured and contextual information via token representations obtained from what we call Contextual and Structured Information layers, respectively. Post extraction and integration of these features, we pass them to a classification layer and follow a heuristic decoding strategy for tagging our sequences. Our Contextual Information Layer and decoding strategy are in line with the previous implementation in the span classification paradigm [22, 23, 24, 25, 26].

3.2.1. Contextual Information Layer

Given a sentence $\{w_1,w_2,w_3, \cdots \, w_{\rm T}\}$ with T words, the token representation layer outputs an encoding $\mathbf{h}_i$ for each token as follows:

$$\mathbf{h}_1, \cdots, \mathbf{h}_T = \mathrm{EMB}\left(w_1, \cdots, w_T\right), \tag{1}$$

where EMB(.) is a pre-trained contextualized text embedding model from the transformers family and $\mathbf{h}_t \in \mathbb{R}^d$.

3.2.2. Structured Information Layer

To incorporate the long-range dependencies and structured information between the tokens in the input sentence, we propose the use of an additional graph-encoded representation $g_{\rm i}$ for each token using dependency parse trees [27, 28]. We follow a simple approach whereby we use a Graph Convolution Network (GCN) to capture information along multiple dependency arcs between words in a sentence. Thus, given the context embeddings from the Bi-LSTM, the graph-encoded representation for each token t is obtained as follows:

$$\mathbf{g}_1, \cdots, \mathbf{g}_T = \mathrm{GCN}\left(\mathbf{h}_1, \cdots, \mathbf{h}_T\right), \tag{2}$$

Post this step, we implement a gate for dynamically adjusting the contribution of features contributed by structured information in tokens, obtained from the GCN. Following the practice in previous work [29], we combine the representations from both Contextual and Structured information as follows:

$$\mathbf{gate} = \sigma\left(\mathbf{W}_g^{\top}[\mathbf{h}_t; \mathbf{g}_t] + \mathbf{B}_g\right) \tag{3}$$

where $\mathbf{W}_g \in \mathbb{R}^{2d \times d}$ is a weight matrix and σ is the element-wise sigmoid function. The final graph-encoded representation for each token w_t is then obtained by $\mathbf{g}_t = \mathbf{gate}.\mathbf{g}_t$. Finally, for each token $w_{\mathbf{i}}$, the final representation fed to the span representation layer is $\mathbf{z_t}$ = [$\mathbf{h}_t$; $\mathbf{g}_t$].

3.2.3. Span Representation Layer

After encoding the tokens in a sentence, we enumerate through all the possible m spans $J = \{j_1, \cdots, j_{\rm i}, \cdots, j_m\}$ upto a maximum specified length (in terms of the number of tokens) for sentence $s = \{w_1, \cdots, w_T\}$ and then re-assign a label $y_i \in \{I, O\}$ for each span j_i. For example, for the sentence "NLP is um important", all possible spans (or pairs of start and end indices) are {(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 3), (2,4), (1, 3), (1,4)}, and all these spans are labelled O except (3, 3) which is labelled I. We denote b_i and s_i as the start and end indices of span j_i respectively. We then formulate the vectorial representation of each span as the concatenation of the representations of the starting token $\mathbf{z}_{b_i} \in \mathbb{R}^{2d}$, the ending token $\mathbf{z}_{s_i} \in \mathbb{R}^{2d}$, and a length embedding $\ell_i \in \mathbb{R}^{len}$.The length embedding is implemented as a look-up table and learnt while training the model. The final vector representation for each span fed into the span prediction layer is now $\mathbf{j}_i = [\mathbf{z}_{b_i} \, ; \mathbf{z}_{s_i} \, ; \ell_i]$.

3.2.4. Span Prediction Layer

The final span representations $\mathbf{j}_i$ is then passed through a linear transformation followed by a softmax operation as follows:

$$\mathbf{P}\left(\hat{y}_{i_k} \mid \mathbf{j}_i\right) = \frac{\mathrm{score}\left(\mathbf{j}_i, \mathbf{y}\right)}{\sum_{y' \in \mathcal{Y}} \mathrm{score}\left(\mathbf{j}_i, \mathbf{y}'\right)}, \tag{4}$$

$$\mathrm{score}\left(\mathbf{j}_i, \mathbf{y}_k\right) = \exp\left(\mathbf{j}_i^T \mathbf{y}_k\right), \tag{5}$$

where $\hat{y}_{i_k}$ is the probability that the span j_i belongs to class k, and $\mathbf{y}_k$ is a learnable representation of the class k.

3.2.5. Decoding

Since our task of disfluency detection does not have overlapping spans, we follow the heuristic decoding method from literature for non-nested entities to avoid the prediction of overlapped spans. Specifically, for overlapped spans, we keep the span with the highest prediction probability and drop the others.

4. Experiments

4.1. Dataset

We evaluate our models on the human-annotated transcriptions [30] from the English Switchboard Dataset (SWBD) [31]. SWBD is one of the most widely used datasets for evaluating disfluency detection models and frameworks. Following [32], we split the entire dataset into training set sw23[$\star$].dps, development set sw4[5-9][$\star$].dps, and test set sw4[0-1][$\star$].dps. Next, for pre-processing the transcripts before feeding them into our model, we follow the pre-processing steps mentioned by [14] and convert all the text to lower-case and remove all punctuation and partial words.

4.2. Experimental Setup

All our models are implemented using the PyTorch [33] deep learning framework. We use Flair [34] to implement all our *token-level classification* baselines. For both our *token-level classification* and *span classification* models we use either of BERT$_{\rm BASE}$ or ELECTRA$_{\rm BASE}$ as our pre-trained contextualized token embedding model, and adopt the pre-trained checkpoints and implementation from the huggingface library [35].

We fine-tune our sequence tagging models with a batch size of 32 using adam optimizer with an initial learning rate of 5 ×

10^{-5}. The dimension of our length embedding len in $\mathbb{R}^{len}$ is 300 and our *Structured Information Layer* has 2 layers of GCN.

4.3. Baselines and Compared Methods

For baselines, we resort to *token-level classification* baselines with or without Conditional Random Field (CRF) decoders [36] under the *IO* tagging scheme. CRF as a decoder has been a common choice for sequence labeling tasks due to its ability to incorporate neighboring label information by considering the state transition probability of neighboring labels in *token-level classification* models. Following prior-art, we choose BERT and ELECTRA from the transformers family as our contextualized text encoder for both the *token-level classification* and *span classification* setups.

4.4. Experimental Results

Table 2 shows the evaluation results on the SWBD test compared to prior-art on disfluency detection. Consistent with prior-art, we show the *F₁* scores for all our approaches. As we clearly see, our best-proposed model (Span Classification w GCN) outperforms state-of-the-art (SOTA) [18] by 1.1% and our baselines by a significant margin. Though our *span classification* models alone report significant gains, integrating structured information with GCNs improves performance over it. Here, we want to reiterate the fact that including structured information for disfluency detection is a logical step and might steer further research in this direction. Additionally, in the next section, we also make an effort to analyze the benefits of adding structured information to our *span classifier* model.

Table 2: *Evaluation results of our proposed model compared to the baselines and prior-art on the Switchboard test set. The best scores are denoted in bold.*

Model	P	R	F_1
Prior-art			
Semi-CRF [13]	90.0	81.2	85.4
Bi-LSTM [4]	91.6	80.3	85.9
Attention-based [5]	91.6	82.3	86.7
Transition-based [37]	91.1	84.1	87.5
Self-supervised [17]	93.4	87.3	90.2
Self-trained [11]	87.5	93.8	90.6
EGBC [38]	95.7	88.3	91.8
BERT fine-tune [38]	94.7	89.8	92.2
BERT-CRF-Aux [18]	94.6	91.2	92.9
ELECTRA-CRF-Aux [18]	94.8	91.6	93.1
Our Experiments			
Our Baselines			
Token Classification BERT	91.8	84.9	88.2
Token Classification ELECTRA	90.8	88.3	89.5
Token Classification BERT-CRF	93.4	82.6	87.7
Token Classification ELECTRA-CRF	92.5	87.2	89.8
Span Classification BERT	95.1	93.0	94.1
Span Classification ELECTRA	90.1	**94.0**	92.3
Span Classification BERT-GCN	**95.2**	93.2	**94.2**
Span Classification ELECTRA-GCN	91.7	94.0	92.9

4.5. Qualitative Results Analysis

In this section, we first analyze with some example instances where *span classification* does better than *token-level classification*. Next, we also study the benefit of integrating graph-encoded structured information in our *span classification* model.

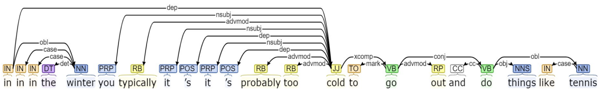
Span Classification vs. Token-level Classification-Highlighted words represent the prediction made by our *span classifier* and ground truth annotations, while the underlined words represent the predictions by our *token-level classifier* model.

1. i work part time at night and he works and my husband works full time days
2. so it was an age where it was we thought it would be good for them to have the discipline that goes with with having a pet

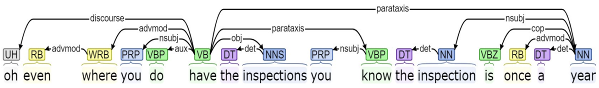
As we clearly see in both the examples, both spans are not obvious disfluencies other than the repetition in eg. 2 which was predicted right by the *token-level classifier*. We hypothesize that longer *restarts* and *repairs* like the ones in the examples are better captured by our *span classifier*, where the group of words with proper boundary supervision helps our *span classifier* better capture semantics than our *token-level classifier* which classifies tokens in isolation.

GCN vs w/o GCN-Highlighted words represent the prediction made by our *span classifier* with the GCN module and ground truth annotations, while the underlined words represent the predictions by our *span classifier* trained without it.

1. in in in the winter you typically it's it's probably too cold to go out and do things like tennis


dep
nsubj
advmod
nsubj
dep
nsubj
dep
advmod
advmod
obl
case
case
det
xcomp
mark
advmod
conj
cc
obj
obl
case
IN IN IN DT NN PRP RB PRP POS PRP POS RB RB JJ TO VB RP CC VB NNS IN NN
in in in the winter you typically it 's it 's probably too cold to go out and do things like tennis

2. oh even where you do have the inspections you know the inspection is once a year


parataxis
discourse
advmod
nsubj
aux
advmod
parataxis
obj
det
nsubj
det
nsubj
cop
advmod
det
UH RB WRB PRP VBP VB DT NNS PRP VBP DT NN VBZ RB DT NN
oh even where you do have the inspections you know the inspection is once a year

In the first example above, the word "typically" is a difficult fluent word to capture since it occurs between 2 disfluent words. However, the GCN guides our *span classifier* through the nominal subject dependency arc, which, when combined with contextual representation, helps our model understand that "typically" is semantically related to "cold". Through the second example, we also hypothesize that our *span classifier w/o GCN* model might be biased to finding repeated nearby words and classify spans wrongly with any former span consisting of the repeated word. In this case, structured dependency of the former "inspections" with other nearby words helps the *span classifier w GCN* model classify it as fluent.

5. Conclusions

In this paper, we propose a novel model architecture for the task of disfluency detection in spoken utterance transcripts. Our proposed model achieves SOTA on the widely used English SWBD for disfluency detection. As part of future work, we would like to investigate how to better integrate structured and contextual information for better disfluency detection.

6. References

[1] P. Bamdev, M. S. Grover, Y. K. Singla, P. Vafaee, M. Hama, and R. R. Shah, "Automated speech scoring system under the lens," *International Journal of Artificial Intelligence in Education*, pp. 1–36, 2022.

[2] Y. Kumar, S. Aggarwal, D. Mahata, R. R. Shah, P. Kumaraguru, and R. Zimmermann, "Get it scored using autosas—an automated system for scoring short answers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 9662–9669.

[3] J. Fu, X. Huang, and P. Liu, "SpanNER: Named entity re-/recognition as span prediction," in *ACL-IJCNLP 2021*, pp. 7183–7195.

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

[5] S. Wang, W. Che, and T. Liu, "A neural attention model for disfluency detection," in *COLING 2016*, pp. 278–287.

[6] Y. Bengio, *Learning deep architectures for AI.* Now Publishers Inc, 2009.

[7] V. Gupta, R. A. Bhat, A. Ghosal, M. Srivastava, M. Singh, and V. Srikumar, "Is my model using the right evidence? systematic probes for examining evidence-based tabular reasoning," *arXiv preprint arXiv:2108.00578*, 2021.

[8] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," *arXiv preprint arXiv:1809.10185*, 2018.

[9] S. Zwarts and M. Johnson, "The impact of language models and loss functions on repair disfluency detection," in *ACL:HLT 2011*, pp. 703–711.

[10] P. Jamshid Lou, P. Anderson, and M. Johnson, "Disfluency detection using auto-correlational neural networks," in *EMNLP 2018*, pp. 4610–4619.

[11] P. J. Lou and M. Johnson, "Improving disfluency detection by self-training a self-attentive model," *arXiv preprint arXiv:2004.05323*, 2020.

[12] M. S. Rasooli and J. Tetreault, "Joint parsing and disfluency detection in linear time," in *EMNLP 2013*, pp. 124–129.

[13] J. Ferguson, G. Durrett, and D. Klein, "Disfluency detection with a semi-markov model and prosodic features," in *ACL:HLT 2015*, pp. 257–262.

[14] J. Hough and D. Schlangen, "Recurrent neural networks for incremental disfluency detection," in *Interspeech 2015*, pp. 849–853.

[15] F. Wang, W. Chen, Z. Yang, Q. Dong, S. Xu, and B. Xu, "Semi-supervised disfluency detection," in *ICCL 2018*, pp. 3529–3538.

[16] J. Yang, D. Yang, and Z. Ma, "Planning and generating natural and diverse disfluent texts as augmentation for disfluency detection," in *EMNLP 2020*, pp. 1450–1460.

[17] S. Wang, Z. Wang, W. Che, and T. Liu, "Combining self-training and self-supervised learning for unsupervised disfluency detection," in *EMNLP 2020*, pp. 1813–1822.

[18] D. Lee, B. Ko, M. C. Shin, T. Whang, D. Lee, E. Kim, E. Kim, and J. Jo, "Auxiliary Sequence Labeling Tasks for Disfluency Detection," in *Interspeech 2021*, 2021, pp. 4229–4233.

[19] Z. Jie and W. Lu, "Dependency-guided lstm-crf for named entity recognition," *arXiv preprint arXiv:1909.10148*, 2019.

[20] P.-H. Li, R.-P. Dong, Y.-S. Wang, J.-C. Chou, and W.-Y. Ma, "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks," in *EMNLP 2017*.

[21] R. Wang, X. Xin, W. Chang, K. Ming, B. Li, and X. Fan, "Chinese ner with height-limited constituent parsing," in *AAAI 2019*.

[22] Z. Jiang, W. Xu, J. Araki, and G. Neubig, "Generalizing natural language analysis through span-relation representations," in *ACL 2020*, pp. 2120–2133.

[23] H. Ouchi, J. Suzuki, S. Kobayashi, S. Yokoi, T. Kuribayashi, R. Konno, and K. Inui, "Instance-based learning of span representations: A case study through named entity recognition," in *ACL 2020*, 2020, pp. 6452–6459.

[24] J. Yu, B. Bohnet, and M. Poesio, "Named entity recognition as dependency parsing," in *ACL 2020*, pp. 6470–6476.

[25] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, "A unified MRC framework for named entity recognition," in *ACL 2020*, 2020, pp. 5849–5859.

[26] X. Mengge, B. Yu, Z. Zhang, T. Liu, Y. Zhang, and B. Wang, "Coarse-to-Fine Pre-training for Named Entity Recognition," in *EMNLP 2020*, pp. 6345–6354.

[27] N. Chomsky, "Three models for the description of language," *IRE Transactions on information theory*, vol. 2, no. 3, pp. 113–124, 1956.

[28] ——, *Aspects of the Theory of Syntax.* MIT press, 2014, vol. 11.

[29] J. Yu, J. Jiang, L. Yang, and R. Xia, "Improving multimodal named entity recognition via entity span detection with unified multimodal transformer." Association for Computational Linguistics, 2020.

[30] V. Zayats, T. Tran, R. Wright, C. Mansfield, and M. Ostendorf, "Disfluencies and human speech transcription errors," *Interspeech 2019*.

[31] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *IEEE ICASSP 1992*, vol. 1, pp. 517–520.

[32] E. Charniak and M. Johnson, "Edit detection and parsing for transcribed speech," in *NAACL 2001*.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[34] N. . (Demonstrations), "Flair: An easy-to-use framework for state-of-the-art nlp," pp. 54–59.

[35] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *EMNLP 2020: System Demonstrations*, pp. 38–45.

[36] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *ICML 2001*.

[37] S. Wang, W. Che, Y. Zhang, M. Zhang, and T. Liu, "Transition-based disfluency detection using lstms," in *EMNLP 2017*, pp. 2785–2794.

[38] N. Bach and F. Huang, "Noisy bilstm-based models for disfluency detection." in *Interspeech 2019*, pp. 4230–4234.