INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Disfluency Detection with Unlabeled Data and Small BERT Models

Johann C. Rocholl, Vicky Zayats, Daniel D. Walker,
Noah B. Murad, Aaron Schneider, Daniel J. Liebling

Google Research, USA

{jcrocholl, vzayats, danwalkeriv, noahmurad, aaronschneider, dliebling}@google.com

Abstract

Disfluency detection models now approach high accuracy on English text. However, little exploration has been done in improving the size and inference time of the model. At the same time, Automatic Speech Recognition (ASR) models are moving from server-side inference to local, on-device inference. Supporting models in the transcription pipeline (like disfluency detection) must follow suit. In this work we concentrate on the disfluency detection task, focusing on small, fast, on-device models based on the BERT architecture. We demonstrate it is possible to train disfluency detection models as small as 1.3 MiB, while retaining high performance. We build on previous work that showed the benefit of data augmentation approaches such as self-training. Then, we evaluate the effect of domain mismatch between conversational and written text on model performance. We find that domain adaptation and data augmentation strategies have a more pronounced effect on these smaller models, as compared to conventional BERT models.

Index Terms: disfluency detection, on-device models, pretraining, self-training, semi-supervised learning

1. Introduction

On-device AI models have many benefits, including lower inference time, privacy preservation, and the ability to operate without an internet connection. As a result, many apps that transcribe spontaneous speech are moving away from server-side to on-device Automatic Speech Recognition (ASR) pipelines. ASR output may contain noise which affects the readability of such transcripts. Sources of noise include ASR errors, incorrect sentence boundaries, filler words, and disfluencies (filled pauses and self-repairs). These errors can reduce the performance of downstream tasks such as natural language understanding (NLU) or machine translation. Disfluency detection is one way to improve the readability of ASR transcripts, and here we focus on small and fast models based on the BERT architecture [1] that are suitable for use in on-device ASR pipelines.

Disfluencies are irregularities that are an integral part of spontaneous speech and include self-repairs, repetitions, restarts, and filled pauses [2]. Following Shriberg et al. [3], the disfluency annotation includes the *reparandum* (the material that the speaker intends to delete), *interruption point* (+), optional *interregnum* which includes filled pauses and discourse markers (e.g. "uh," "um," "you know,", "I mean"), and an optional *repair* — the material that semantically replaces the reparandum. Below are a few examples of disfluent phrases:

```
[ it's + { uh } it's ] almost...
[ was it, + { I mean, } did you ] put...
[ By + ] it was attached to...
```

Recent developments of smaller on-device BERT-based models (e.g. MobileBERT [4] and Small-vocab BERT [5]) have adapted this architecture for use in resource-constrained contexts including mobile devices. These models are usually pretrained using written text like Wikipedia articles and books, contributing to a domain mismatch with spontaneous speech. Here, we demonstrate that it is possible to train disfluency detection models that are small enough to run in real time on-device, while retaining a high level of accuracy. We build on previous work that showed the benefit of data augmentation approaches such as adding simulated disfluency data and self-training to show how self-training and model pretraining on conversational data can benefit on-device models.

The contributions of this paper are as follows. First, to the best of our knowledge, this is the first work that explores the capabilities of small on-device disfluency detection models, showing that it is possible to achieve only slight degradation in performance on a disfluency detection task with a model as small as 1.3 MiB. This represents a model size reduction by two orders of magnitude compared to a state-of-the-art BERT$_{\text{BASE}}$ model and inference latency reduction by a factor of 8. We also show the importance of pretraining and the effect of domain mismatch between conversational and written text on model performance. In particular, we find that self-training has a more pronounced effect on these smaller models, as compared to conventional BERT models, while pretraining on Reddit improved the performance of a large BERT$_{\text{BASE}}$ model.

2. Related Work

Prior to BERT, LSTM-based models and their variants were a popular choice for disfluency detection task. Those approaches included more traditional usage of LSTMs [6, 7], LSTM variants that explicitly exploited similarity between the reparandum and repair of disfluencies [8, 9, 10], and noisy-channel model [11]. After the development of BERT, larger BERT-based models showed significant improvement over prior work on disfluency detection task [12, 13].

Prior studies on disfluency detection showed the importance of data augmentation approaches, predominantly due to the relatively small size of Switchboard corpus [14] that is used as the main source of annotated data. Both Wang et al. [15] and Bach et al. [12] showed improvement by simulating and inserting disfluencies in fluent transcripts and using those transcripts as an additional source of data. Similarly, self-training proposed by Jamshid Lou et al. [13] augments Switchboard corpus by using a larger Fisher corpus [16] automatically labeled by an existing disfluency classifier model. While some prior studies incorporate prosody [17, 18] or use the acoustic signal as an input [19, 20], in this work we use manually transcribed text only, leaving the usage of acoustic signals and ASR transcripts for future work.

3. Methods

In this work we explore BERT and some of its more compact variants fine-tuned for the disfluency detection task. Following previous work on disfluency detection with BERT [12, 13] our biggest model that is used for pretraining and self-training is the $\text{BERT}_{\text{BASE}}$ [1] model. Below we provide an overview of some smaller and faster models that are specifically targeted for on-device usage.

3.1. Size Reduction

Recently a number of BERT distillation approaches have been proposed that allow significant size and latency reduction. Most of those models follow the student-teacher paradigm where a large BERT model plays the role of a teacher, while a compact student model (with reduced number of layers, hidden states, number of heads and/or vocabulary size) learns the teacher behaviour using various distillation techniques. In this work we tried the following pretrained distilled BERT models:

DistilBERT [21] In addition to a masked language model (MLM) objective, DistilBERT uses knowledge distillation approach [22], where a student is trained to recover the predictions of a teacher.

MobileBERT [4] In addition to layer-wise knowledge distillation, MobileBERT uses bottleneck structures that allow a model projection from the teacher to the student.

TinyBERT [23] learns a direct mapping for embedding, hidden state, attention, and prediction layers in order to transfer model knowledge from teacher to student.

PD-BERT [24] introduced a set of very small models that are first pretrained using MLM objective, and only then use knowledge distillation similar to DistilBERT.

Small-vocab BERT [5] uses a mixed-vocab training approach to train very small models with a smaller WordPiece vocabulary (5k vs. the usual 30k) because for small models, the token embedding layer accounts for a significant percentage of overall model size.

While some of these approaches propose to fine-tune the teacher model on the task before distillation happens, in our current work for all the above mentioned models we start with an already distilled student model and fine-tune it on disfluency detection task.

3.2. Domain Adaptation

Commonly available pretrained BERT models were trained using text from Wikipedia and Books. While this exposes the model to a large set of topics, the linguistic styles of these corpora do not necessarily match the more conversational and disorganized style of spontaneous speech and dialogue. In this paper we experiment with self-training and pretraining in order to mitigate domain shift caused by mismatch of the styles.

3.2.1. Self-training

We follow Jamshid Lou et al. [13] by using self-training with the Fisher corpus. Specifically, first we train (fine-tune) a full-size $\text{BERT}_{\text{BASE}}$ model on Switchboard corpus and then use that model to add predicted disfluency labels to the Fisher corpus. After that, we use this automatically annotated Fisher corpus, also referred here as "silver" data, as an additional source of training data for further fine-tuning of the $\text{BERT}_{\text{BASE}}$ or smaller student model.

Table 1: *Datasets statistics.*

Dataset	**Sentences**	**Docs**	**Words**	**Disfluent Sentences**	**Disfluent Spans**
Switchboard	77k	1k	1M	25k	34k
Fisher	1.3M	12k	20M	492k	676k
Reddit	314M	135M	3B	N/A	N/A

3.2.2. Pretraining

We hypothesized that, just like self-training on additional conversational text can help the model generalize better, starting with a model that is pretrained on more conversational data will improve the model's ability to create good spontaneous speech representation, leading to better performance in the disfluency classification task. In this work we started with Wiki/Books-pretrained $\text{BERT}_{\text{BASE}}$ and one of the Small-vocab BERT models and continued the pretraining separately on two datasets that were chosen to more closely match casual and spontaneous speech: Reddit comments and transcribed human speech from the Fisher dataset [16] (see Section 4.1).

4. Experiments

4.1. Data and Tokenization

For training, we use transcripts from Switchboard corpus of English conversational dialogue [14]. This corpus is widely used in disfluency detection research because it has gold labels for disfluent spans of text. We use the established train/dev/test splits [26]. We removed commas and filled pauses ("uh", "huh", "uh-huh", "um") because they are not found in most of the ASR output that we want to annotate with our models.

In most of our experiments, unless mentioned otherwise, we report metrics on the task of labeling both the reparandum and interregnum (e.g. "you know", "well", "I mean") of disfluencies. This is in contrast to the majority of prior studies that typically focus on the recognition of the reparandum only. Our motivation here is to identify spans that should be elided in order to improve the overall readability of a transcript. For the purpose of comparison to prior work, we also include results for our best-performing model configurations retrained using the standard reparandum-only task (see Section 4.6).

In this work we experiment with using the Fisher dataset [16] for both pretraining and self-training. In order to reduce fragmentation, we preprocess Fisher by combining utterances across interruptions and combining short contiguous utterances from the same speaker. For the next sentence prediction task used during model pretraining, we sorted the processed utterances by the timestamp of the first component utterance.

In addition to the Fisher corpus, we experiment with using Reddit comments for pretraining our models. The main benefit of using Reddit is that it is a large corpus of casual/conversational discussions. Since it is text-based, it doesn't contain disfluencies like the spontaneous speech recorded in Fisher, but it is still much more conversational than most text from Wikipedia. We used the 2019 portion of the Pushshift dataset [27]. For the next sentence prediction pretraining task, each comment was treated as a separate document and segmented into sentences using NLTK's[1] `sent_tokenize`.

[1]`https://www.nltk.org`

Table 2: *Model size vs performance on Switchboard dev dataset. Asterisk indicates models using Wiki/books checkpoints available via Huggingface Transformers [25]. For the rest of the models we used compatible initial checkpoints pretrained on the same Wiki/books dataset that have not yet been released. Bold font highlights the smallest models with good performance.*

Model	Layers	Hidden Size	Heads	Params	Size (MiB)	Latency (ms)	Precision	Recall	F1	Precision	Recall	F1
BERT$_{BASE}$ [1]	12	768	12	108.9M	416	72	92.9	90.2	91.5	93.2	90.5	91.8
DistilBERT* [21]	6	768	12	66.4M	253	42	93.2	87.7	90.4	93.8	88.9	91.3
TinyBERT* [23]	6	768	12	66.4M	253	42	93.6	87.0	90.2	93.5	88.8	91.1
	4	312	12	14.3M	54	13	90.6	84.1	87.2	91.7	87.5	89.5
MobileBERT* [4]	24	128	4	24.6M	95	41	92.4	89.7	91.0	92.7	89.9	91.3
Small-vocab BERT [5]	24	192	3	11.7M	45	32	90.9	87.8	89.3	91.5	89.0	90.2
	6	256	8	4.6M	18	12	91.9	85.3	88.5	92.8	87.5	90.1
	12	**128**	4	3.1M	**12**	15	91.4	86.4	88.8	**92.5**	**88.2**	**90.3**
	6	**96**	4	1.2M	**4.7**	9	90.3	79.1	84.3	**92.5**	**85.5**	**88.9**
PD-BERT* [24]	4	128	2	4.8M	18	7	91.9	77.9	84.3	92.4	85.0	88.5
	2	128	2	4.4M	17	5	89.6	72.1	79.9	92.8	79.9	85.9
								baseline			**self-trained**	

Table 1 shows summary stats on the Switchboard training, Fisher, and Reddit sets, including the numbers of total sentences, "documents" (comments/conversations), words, sentences with a tagged disfluency, and total spans tagged as disfluent in each. For the last two we use labels from BERT$_{BASE}$ model for Fisher, and human-annotations for Switchboard.

4.2. Metrics

Our main metric for model performance is token-level F1 score (the harmonic mean of token-level precision and recall). We report model sizes in number of parameters and/or mebibytes (1 MiB = 1024^2 bytes). Latency is measured in milliseconds for the median time that it takes to perform batch inference on 8 example sentences on GPU.

4.3. Hyper-parameter Tuning

We optimize the learning rate (around 2e-4), training batch size (from 32 to 512), number of training epochs (up to 55 for small models), and (for self-training) the percentage of "silver" data included in each training batch (around 70%). We used a black-box hyper-parameter optimization system [28]. For most experiments we ran 64 trials total, with 8 evaluations in parallel. Each individual trial (one set of hyper-parameters) ran on a single NVIDIA P100 GPU for about 30 minutes to 2 hours. All results shown are parameterizations that yielded the best F1 score on the Switchboard dev set.

4.4. Results

First, we compare the model performance for BERT and its smaller variants mentioned in Section 3.1 of different sizes that were pretrained on Wikipedia and books corpus, results are presented in Table 2. As expected, model performance generally declines with the size of the model. While comparing different models, we found that among the tiniest of the models that are capable to fit on-device, Small-vocab BERT has the best performance. In addition to evaluating various models using Switchboard corpus, we also experiment with incorporating self-training approach using "silver" Fisher data. The results for self-training experiments are also presented in Table 2.

While performing self-training for models of different sizes we have noticed an interesting trend — the amount to which self-training improves model performance appears to increase as the model size decreases, making self-training especially important for smaller models. This trend can be seen more easily in Figure 1 on BERT$_{BASE}$, MobileBERT, and Small-vocab BERT models, and similar trends were observed with PD-BERT models. By looking at the predictions on the dev set we did not notice much difference in prediction quality between BERT$_{BASE}$ and Small-vocab BERT besides the slightly elevated number of errors associated with false-positive repetition detection in a Small-vocab BERT.

4.5. Effect of Pretraining

In order to measure the impact of domain mismatch between the pretraining data and the target data (spontaneous speech) on final model performance, we experiment with BERT pretraining using the Fisher and Reddit corpora for a large BERT$_{BASE}$ configuration and Small-vocab BERT 12×128. Following Devlin

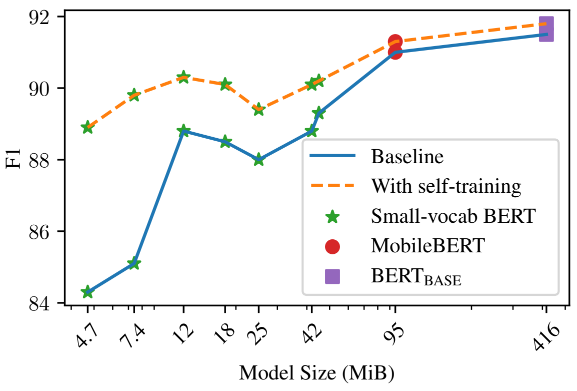

Figure 1: *F1 scores for model configurations with and without self-training, shows performance correlated with model size, self-training helps more for small models. Not all models shown, though similar patterns exist for other model types.*

et al. [1] we restrict pretraining to 1M steps with an initial learning rate of 0.0001 which decreased in equal increments at each step to zero. In order to identify the best stopping criteria, we evaluate model performance on disfluency detection task every 200K steps. For Fisher the optimal checkpoint was found after pretraining for 200K-400K steps, while for Reddit the optimal stopping point was at 600K-1M steps, which can be explained by the large size disparity between Fisher and Reddit.

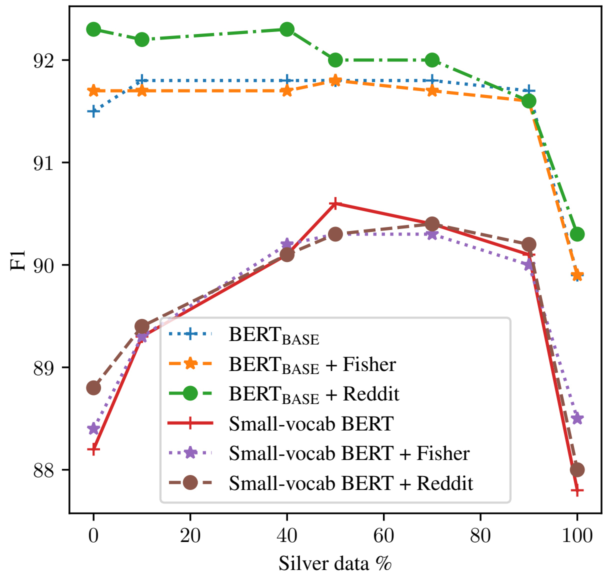

Figure 2: *Self-training silver data percentage v.s. F1 for the BERT$_{BASE}$ and Small-vocab BERT 12×128 configurations, with and without pretraining on Reddit or Fisher.*

In order to understand the impact of pretraining, we ran experiments combining pretraining and self-training in various configurations. For all of our 6 settings, we experiment with including different percentages of silver data and plot F1 scores for BERT$_{BASE}$ and Small-vocab BERT 12×128 in Figure 2. We used a derivative free optimization package to optimize the training parameters (batch size, learning rate, number of iterations, etc.) and chose the "best" result for each model configuration at each silver data percentage which is likely why this figure differs somewhat from that in [8]. From those trends we notice that pretraining on Reddit benefits the larger BERT$_{BASE}$ model, achieving F1 score 92.3, while not necessarily helping smaller model (F1 score 90.4).

Most of the self-training results presented here use the same Fisher corpus, with disfluency labels predicted by a BERT$_{BASE}$ model fine-tuned on Switchboard only. As an additional experiment, we used our best BERT$_{BASE}$ model from Table 4 (pretrained on Reddit, self-trained on Fisher, and hyper-parameter tuned for best performance on Switchboard dev set, F1 score 90.9 on test set) to predict better labels on Fisher corpus. When using this improved dataset for self-training the Small-vocab BERT 6×96 model, we found no significant performance improvement on Switchboard test set, presumably because this tiny model was already pretty close to optimal performance for its size. But for the 12×128 model, this iterative approach improved precision, recall and F1 score by about 0.4 points, which is similar to the 0.5 point improvement for BERT$_{BASE}$ + Reddit + self-training.

Table 3: *Results of 8-bit quantization using TensorFlow Lite. Model size (MiB) about 28% of floating point model, latency (ms) for single-sentence inference on Android Pixel 3a, performance degradation on Switchboard dev set ≤ 0.2 points.*

Model		**Size**	**Lat**	**Prec**	**Rec**	**F1**
MobileBERT		25	162	92.6	89.9	91.2
Small-vocab	12×128	3.3	32	92.3	88.0	90.1
	6×96	1.3	11	92.4	85.4	88.8

4.6. Comparison to Other Work

Finally, we compare our best performing models against previously published results using Switchboard test set in Table 4. To make our findings comparable, for this set of experiments we retrained and evaluated the models with labels associated with reparandum only, without explicitly marking the interregnum. Unfortunately we were not able to find published size or latency numbers for all of these results.

Table 4: *Model comparison on Switchboard test set without including interregnum in the label.*

Arch	**Model**	**Prec**	**Rec**	**F1**
CRF	Ferguson et al. [17]	90.1	80.0	84.8
LSTM	Zayats et al. (2016) [6]	91.8	80.6	85.9
	Jamshid Lou et al. [8]	89.5	80.0	84.5
	Zayats et al. (2018) [9]	-	-	86.7
BERT	Bach et al. (2019) [12]	94.7	89.8	92.2
	Jamshid Lou et al. [13]	86.7	91.9	89.2
	+ ensemble of 4 models	87.5	93.8	90.6
	BERT$_{BASE}$ (416 MiB)	92.6	88.4	90.4
	+ Reddit + self-trained	93.1	88.9	90.9
Small-vocab BERT	12×128 self-trained	92.0	87.9	89.9
	+ quantized (3.3 MiB)	91.9	87.8	89.8
	6×96 self-trained	91.3	85.9	88.5
	+ quantized (1.3 MiB)	91.1	85.8	88.4

5. Conclusions and Future Work

Self-training can be used to transfer knowledge from a large teacher model to a much smaller student. We present results for several experiments with different combinations of the number of layers, hidden size, attention heads, vocabulary size, and distillation methods. Additional pre-training on conversational speech corpora helps improve disfluency detection performance compared to pretraining only on Wikipedia+Books. Quantization can reduce model size even more. By combining these techniques, we reduced model size by 99% and inference latency by 80% while maintaining competitive performance.

While the focus of this paper is on investigating simple semi-supervised learning techniques for task-agnostic distilled models, we leave it to future work to understand if we can further improve model performance by using semi-supervised techniques before or during model distillation.

6. Acknowledgements

The authors would like to thank everyone who provided feedback on drafts of this paper, especially Shyam Upadhyay, Raghav Gupta, and Dirk Padfield. Thanks also to Max Gubin for supplying the processing scripts for the Reddit data.

7. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[2] E. A. Schegloff, G. Jefferson, and H. Sacks, "The preference for self-correction in the organization of repair in conversation," *Language*, vol. 53, no. 2, pp. 361–382, 1977. [Online]. Available: https://doi.org/10.2307/413107

[3] E. Shriberg, R. Bates, and A. Stolcke, "A prosody only decision-tree model for disfluency detection," in *Fifth European Conference on Speech Communication and Technology*, 1997.

[4] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: a compact task-agnostic BERT for resource-limited devices," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 2158–2170. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.195

[5] S. Zhao, R. Gupta, Y. Song, and D. Zhou, "Extremely small BERT models from mixed-vocabulary training," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 2753–2759. [Online]. Available: https://www.aclweb.org/anthology/2021.eacl-main.238

[6] V. Zayats, M. Ostendorf, and H. Hajishirzi, "Disfluency detection using a bidirectional LSTM," *CoRR*, vol. abs/1604.03209, 2016. [Online]. Available: http://arxiv.org/abs/1604.03209

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

[8] P. Jamshid Lou, P. Anderson, and M. Johnson, "Disfluency detection using auto-correlational neural networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4610–4619. [Online]. Available: https://www.aclweb.org/anthology/D18-1490

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

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

[11] P. Jamshid Lou and M. Johnson, "Disfluency detection using a noisy channel model and a deep neural language model," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 547–553. [Online]. Available: https://www.aclweb.org/anthology/P17-2087

[12] N. Bach and F. Huang, "Noisy BiLSTM-based models for disfluency detection," in *Proceedings of Interspeech 2019*, 2019, pp. 4230–4234. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1336

[13] P. Jamshid Lou and M. Johnson, "Improving disfluency detection by self-training a self-attentive model," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 3754–3763. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.346

[14] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proceedings of ICASSP*, vol. I, 1992.

[15] F. Wang, W. Chen, Z. Yang, Q. Dong, S. Xu, and B. Xu, "Semi-supervised disfluency detection," in *Proceedings of the 27th International Conference on Computational Linguistics*, Aug. 2018, pp. 3529–3538. [Online]. Available: https://www.aclweb.org/anthology/C18-1299

[16] C. Cieri, D. Miller, and K. Walker, "The Fisher Corpus: a resource for the next generations of speech-to-text," in *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Lisbon, Portugal: European Language Resources Association (ELRA), May 2004. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2004/pdf/767.pdf

[17] J. Ferguson, G. Durrett, and D. Klein, "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*, 2015, pp. 257–262.

[18] V. Zayats, T. Tran, R. Wright, C. Mansfield, and M. Ostendorf, "Disfluencies and Human Speech Transcription Errors," in *Proceedings of Interspeech 2019*, 2019, pp. 3088–3092. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-3134

[19] T. Kourkounakis, A. Hajavi, and A. Etemad, "Fluentnet: End-to-end detection of speech disfluency with deep learning," *arXiv preprint arXiv:2009.11394*, 2020.

[20] P. J. Lou and M. Johnson, "End-to-end speech recognition and disfluency removal," *arXiv preprint arXiv:2009.10298*, 2020.

[21] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[23] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Nov. 2020, pp. 4163–4174. [Online]. Available: https://www.aclweb.org/anthology/2020.findings-emnlp.372

[24] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[25] 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[26] E. Charniak and M. Johnson, "Edit detection and parsing for transcribed speech," in *Second Meeting of the North American Chapter of the Association for Computational Linguistics*, 2001.

[27] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The Pushshift Reddit dataset," 2020.

[28] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17, New York, NY, USA, 2017, pp. 1487–1495. [Online]. Available: https://doi.org/10.1145/3097983.3098043