End-to-End Speech Recognition and Disfluency Removal

Paria Jamshid Lou\(^1\) and Mark Johnson\(^2,1\)

\(^1\)Department of Computing, Macquarie University
\(^2\)Oracle Digital Assistant, Oracle Corporation

\(^1\)paria.jamshid-lou@hdr.mq.edu.au
\(^2\)mark.mj.johnson@oracle.com

Abstract

Disfluency detection is usually an intermediate step between an automatic speech recognition (ASR) system and a downstream task. By contrast, this paper aims to investigate the task of end-to-end speech recognition and disfluency removal. We specifically explore whether it is possible to train an ASR model to directly map disfluent speech into fluent transcripts, without relying on a separate disfluency detection model. We show that end-to-end models do learn to directly generate fluent transcripts; however, their performance is slightly worse than a baseline pipeline approach consisting of an ASR system and a disfluency detection model. We also propose two new metrics that can be used for evaluating integrated ASR and disfluency models. The findings of this paper can serve as a benchmark for further research on the task of end-to-end speech recognition and disfluency removal in the future.

1 Introduction

One characteristic of spontaneous speech which makes it different from written text is the presence of disfluencies. Disfluencies include filled pauses (e.g. *um* and *uh*), repetitions (e.g. *the the*), corrections (e.g. *Show me the flights . . . the early flights*), parenthetical asides (e.g. *you know*), interjections (e.g. *well* and *like*), restarts (e.g. *There’s a . . . let’s go*) and partial words (e.g. *wou- and oper-*) which frequently occur in spontaneous speech\(^1\) and reduce the readability of speech transcripts (Liu et al., 2006). They also pose a major challenge to downstream tasks relying on the output of speech recognition systems, such as parsing and machine translation models (Johnson and Charniak, 2004; Wang et al., 2010; Honnibal and Johnson, 2014). Since these models are usually trained on fluent clean corpora, the mismatch between the training data and the actual use case decreases their performance. To tackle this challenge of spontaneous speech, specialized disfluency detection models are developed and applied as a post-processing step to remove disfluencies from the output of speech recognition systems (Zayats et al., 2016; Wang et al., 2018; Dong et al., 2019). One type of disfluency which is especially problematic for disfluency detection models is speech repair. Shriberg (1994) defines three distinct parts of a speech repair, referred to as *reparandum*, *interregnum* and *repair*. As illustrated in the example below, the reparandum *to Boston* is the part of the utterance that is replaced and is usually followed by an interruption point in the speech signal, the interregnum *uh I mean* is an optional part of a disfluent structure (that consists of a filled pause *uh* and a discourse marker *I mean*) and the repair *to Denver* replaces the reparandum. The fluent version is obtained by deleting reparandum and interregnum words.

\begin{align*}
\text{I want a flight to Boston} & \quad \text{uh I mean to Denver.} \\
\hline
\text{reparandum} & \quad \text{interregnum} & \quad \text{repair}
\end{align*}

Disfluency detection is usually an intermediate step between an ASR model and a downstream task. This pipeline approach is complex to implement and leads to higher inference latency. It also has the potential problem of errors compounding between components, e.g. recognition errors lead to larger disfluency detection errors. End-to-end models, on the other hand, are less prone to such problems. More importantly, end-to-end models can leverage paralinguistic features in speech signal that are not available in pipeline systems. Speech carries extra information beyond the words which might provide useful cues to disfluency detection\(^2\). In this paper,

\(^1\)Shriberg (1994) observed disfluencies once in every 20 words.

\(^2\)Prosodic cues (e.g. *pause*) signal disfluencies by marking the interruption point (Shriberg, 1994; Zayats and Ostendorf, 2019).
we address the task of end-to-end speech recognition and disfluency removal. Specifically, we investigate whether it is possible to train an ASR model end-to-end to directly map disfluent speech into fluent transcripts, without an intermediate disfluency detection step. Some previous work has attempted disfluency detection as part of another task in an end-to-end manner, e.g. joint disfluency detection and constituency parsing (Jamshid Lou et al., 2019) and direct translation from disfluent Spanish speech to fluent English transcripts (Salesky et al., 2019). However, to the best of our knowledge, this is the first work that systematically investigates the task of end-to-end ASR and disfluency removal, serving as a starting point for future research into end-to-end disfluency removal systems. In this paper, we aim to answer the following questions:

- **Can an ASR model directly generate fluent transcripts from disfluent speech?** We might expect an end-to-end ASR model (without an explicit disfluency detection component) not to effectively detect disfluencies. However, we show that end-to-end ASR models do learn to directly generate fluent transcripts and their performance is comparable to a baseline pipeline system (i.e. an ASR model followed by a specialized disfluency detection model).

- **How does the choice of architecture impact disfluency detection and removal in end-to-end speech recognition?** We compare the performance of three neural-based end-to-end ASR and disfluency removal models including a Connectionist Temporal Classification based model, an LSTM-based Sequence-to-Sequence model and a Transformer model and show that a Transformer ASR model has the best performance on disfluency removal.

- **How can we systematically evaluate the performance of an end-to-end ASR and disfluency removal model?** The existing evaluation metrics are designed to measure the performance of a single task, namely speech recognition or disfluency detection, but not both. We introduce two new metrics measuring the disfluency removal and word recognition performance of an end-to-end model.

2 Related Work

Disfluency removal is typically performed by training a specialized disfluency detection model on disfluency labeled data and applying it as a separate component following an ASR model and prior to a downstream task. The specialized disfluency detectors (Zayats et al., 2016; Wang et al., 2016; Jamshid Lou et al., 2018) are usually trained on the Switchboard corpus (Marcus et al., 1999) which is the largest available dataset with gold (i.e. human-annotated) disfluency labels. State-of-the-art disfluency detectors use Transformer models with pretrained contextualised word embeddings (e.g. BERT) (Tran et al., 2019; Jamshid Lou et al., 2019; Dong et al., 2019; Wang et al., 2019a; Jamshid Lou and Johnson, 2020). Multi-task learning has been effective for disfluency detection, for example, a Transformer trained to jointly detect disfluencies and find constituency parse trees would leverage syntactic information and detect disfluencies more accurately (Jamshid Lou et al., 2019). Self-training and ensembling have also shown to provide benefit to disfluency detection (Jamshid Lou and Johnson, 2020). Self-training on disfluent data provides benefits orthogonal to the pre-trained contextualized embeddings and mitigates the scarcity of gold disfluency labeled data. The BERT-based self-attentive parser introduced in Jamshid Lou et al. (2019) is the current state-of-the-art in disfluency detection; thus, we use it as the “off-the-shelf” disfluency detector in our pipeline approach, as explained in Section 5.

The most similar previous work to ours investigates end-to-end speech-to-text translation (Salesky et al., 2019). They train a Sequence-to-Sequence model to directly translate from disfluent Spanish speech to fluent English transcripts without a separate disfluency detection step. They evaluate the performance of their model against fluent transcripts using BLEU and METEOR which are standard metrics for evaluating machine translation systems but are not designed to specifically measure whether a transcript contains disfluencies. These metrics are sensitive to sequence length which makes them undesirable for evaluating end-to-end models incorporating disfluency detection. Fluent transcripts tend to contain fewer tokens per sentence in comparison with disfluent transcripts. Moreover, since they do not compare their end-to-end machine translation and disfluency removal model against a baseline pipeline model (i.e. a ma-
chine translation model followed by a specialized disfluency detection model), the performance of the end-to-end model is unclear in terms of disfluency detection. By contrast, we introduce two new metrics in this paper that systematically measure the fluency of the generated transcripts. We also benchmark our end-to-end model against a state-of-the-art pipeline approach to explicitly evaluate its disfluency detection performance.

3 Speech Recognition and Disfluency Removal

We investigate three different ASR architectures: Connectionist Temporal Classification (CTC), LSTM-based Sequence-to-Sequence and Transformer. Each of these three ASR models is trained twice: (i) in a pipeline approach where the ASR model is trained to transcribe speech, followed by an “off-the-shelf” specialized disfluency detection model, (ii) in an end-to-end approach where the ASR model is trained to jointly transcribe speech and remove disfluencies, which we refer to as an integrated ASR and disfluency model. The ASR models for the two training regimes are identical in terms of architecture and the number of parameters. The only difference is their training data, i.e. the pipeline ASR model is trained on disfluent speech and disfluent transcripts while the end-to-end ASR model is trained on disfluent speech and fluent transcripts. Given the same speech utterance, the same ASR architecture is trained to either produce (i) or (ii):

(i) I want a flight to Boston uh I mean to Denver
(ii) I want a flight to Denver

As input features to the ASR model, we preprocess the speech signal by sampling the raw audio waveform using a sliding window of 25ms with stride 10ms. We then extract 80-dimensional log mel-filterbank coefficients plus three fundamental frequency features from these frames using Kaldi (Povey et al., 2011). We train a CTC-based ASR model, called Jasper (Li et al., 2019), using the OpenSeq2Seq Toolkit\(^3\) (Kuchaiev et al., 2018). Jasper contains 10 blocks of 1D-convolutional layers, each with 5 sub-blocks. A sub-block consists of a 1D-convolutional operation, batch normalization, clipped ReLU activation and dropout. There is a residual connection between each block which is added to the output of the last 1D-convolutional layer in the block before the clipped ReLU activation and dropout. The optimizer used to train the model is stochastic gradient descent with momentum and the loss is CTC (Graves et al., 2006). At decoding time, a candidate list is generated using word-level 4-gram language models and beam search with a width of 2048. For more details, see Li et al. (2019).

We build the encoder-decoder Sequence-to-Sequence model with Bahdanau attention (Bahdanau et al., 2014) using the Espresso Toolkit\(^4\) (Wang et al., 2019b). The Sequence-to-Sequence model uses a 4-layer 2D-convolution, followed by a 3-layer bidirectional LSTM as an encoder and a 3-layer LSTM as a decoder. We train the model using cross-entropy loss and an Adam optimizer. We leverage shallow fusion (Gülçehre et al., 2015) as a language model integration technique. The decoder with shallow fusion computes a weighted sum of two posterior distributions over subword units from the speech recognition model and from the neural language model. For more details, see Wang et al. (2019b).

We also train a Transformer ASR model inspired by Mohamed et al. (2019) using the Fairseq Toolkit\(^5\) (Ott et al., 2019). The Transformer replaces the sinusoidal positional embeddings at the encoder and the decoder with convolutional layers to capture the positional information. The encoder contains two 2D-convolutional blocks with layer norms and ReLU after each convolutional layer. Each convolutional block contains two convolutional layers followed by a 2D max pooling layer with kernel sizes of 3 and 2, respectively. The convolutional layers are used on top of 16 encoder transformer blocks with model hidden dimension 1024 and 16 attention heads. The decoder includes three 1D-convolutional layers, each with a kernel size of 3, and 6 decoder transformer blocks. The Transformer layers learn the global sequential structure of the input while the convolutional layers learn local relationships within a small context. The training criterion is cross-entropy loss and the model is optimized using adadelta. We employ shallow fusion and standard beam search with a beam size of 20 at decoding time. In order to have a fair comparison with other models, we do not pretrain the Transformer. For more details, see Ott

---

\(^3\)https://github.com/NVIDIA/OpenSeq2Seq
\(^4\)https://github.com/freewym/espresso
\(^5\)https://github.com/pytorch/fairseq
et al. (2019). The language models used in the three ASR models are extracted from or trained on the same data used for training the ASR models (i.e. on fluent transcripts for the end-to-end models and on disfluent transcripts for the pipeline models).

4 Evaluating Integrated ASR and Disfluency Models

The performance of ASR models is usually evaluated in terms of word error rate (WER). WER is calculated by finding an alignment between the reference transcript (which is human-transcribed speech) and ASR output so that a minimum number of edits (i.e. substitutions, insertions and deletions) are required for transcribing the ASR output to the reference transcript. Given an alignment, WER is the ratio between the number of incorrectly aligned words and the total number of words in the reference transcript:

\[ \text{WER} = \frac{s + i + d}{n} \]

where \( s \), \( i \) and \( d \) are the number of substitutions, insertions and deletions and \( n \) is the total number of words in the reference transcript. The reference transcript contains both fluent and disfluent words, so a WER of zero on the full transcript means that the system returned all of the disfluent words as well as the fluent words, which is not what an integrated system should do. While WER with respect to the full reference transcript (containing both fluent and disfluent words) is not meaningful for integrated systems intended to produce fluent output, WER with respect to the fluent subsequence is a meaningful measure of overall system, since this is the intended output of an integrated system. However, since disfluencies only comprise around 5% of the total words, the WER score largely reflects how well fluent words are recognized, rather than how well the system handles disfluencies. A system may score poorly on WER even though it is perfect in terms of detecting disfluencies because it fails to correctly recognize the fluent words.

To address the limitations of the existing metric, we introduce two new evaluation metrics which assess the output of an integrated model in terms of fluency and word recognition accuracy in Section 4.1. We then demonstrate the problems associated with the standard ASR alignment algorithm and how it can lead to undesirable alignments for evaluating integrated ASR and disfluency models. As a solution, we modify standard alignment weights to correctly align reference transcripts (which may contain disfluencies) with integrated model outputs in Section 4.2.

4.1 Fluent and Disfluent Error Rate Scores

To overcome the limitations of WER, we use the standard WER evaluation to evaluate fluent and disfluent words separately. In this way, the quality of integrated model outputs is evaluated in terms of both fluency and word recognition. We calculate the word error rate on fluent words (which we call the fluent error rate or FER) as the number of substitutions \( s_f \), deletions \( d_f \) and insertions \( i_f \) among fluent words divided by the total number of fluent words in the reference transcript \( n_f \):

\[ \text{FER} = \frac{s_f + i_f + d_f}{n_f} \]

We define the word error rate on disfluent words (which we call the disfluent error rate or DER) as anything other than a deletion (i.e. substitutions \( s_d \), insertions \( i_d \) and copies \( c_d \)) among disfluent words divided by the total number of disfluent words in the reference transcript \( n_d \) as follows:

\[ \text{DER} = \frac{s_d + i_d + c_d}{n_d} \]

For instance, FER and DER are respectively equal to 0.5 and 0.4 in Figure 1. For calculating FER and DER, we need to align the reference transcripts (i.e. human-transcribed speech with gold disfluency labels) to the integrated model outputs, which are expected to be fluent. The aligner used for this purpose is explained in the following section.

**Figure 1**: Ref is the reference transcript which is human-transcribed speech with gold disfluency labels, shown in red. E2E represents the output of an integrated ASR and disfluency removal model.

4.2 Aligning Integrated Model Output to Reference Transcripts

In this section, we first describe the standard ASR alignment algorithm and explain why it sometimes finds misleading alignments of the output from integrated ASR and disfluency systems. We then

---

*https://github.com/pariajm/e2e-asr-and-disfluency-removal-evaluation*
4.2.1 Problems with Standard ASR Alignment

To illustrate the problems with standard ASR alignment algorithms, consider Figure 2, where the outputs from an integrated model have been aligned with the reference transcripts using two different alignment weights. The first alignment, indicated as Align 1, is generated by the Sclite Toolkit. Sclite\(^7\) is a standard toolkit for evaluating ASR outputs which finds an alignment using dynamic programming algorithms such that a copy, deletion, insertion and substitution cost 0, 3, 3 and 4, respectively. Align 2, on the other hand, is what we expect an aligner to produce in order to have meaningful FER and DER evaluations. As shown in Align 1, the fluent words in the outputs of the integrated system are aligned with the disfluent words in the reference transcripts rather than the fluent words. Since we expect the reference transcript to contain both fluent and disfluent words and the output of an integrated system to discard the disfluencies, the standard alignment weights fail to properly align the integrated model output to the reference transcript. Align 1 and Align 2 have the same alignment cost with the standard weights, so an aligner using the standard weights has no reason to prefer one over the other. The problem that arises here is that since many disfluent words are copies of fluent words, if the same cost is used to align fluent and disfluent words, the alignment will be ambiguous (i.e. there will be multiple alignments with the same cost). Thus, to force the aligner to prefer aligning null (i.e. deletions) for disfluent words and copy for fluent words, we modify the alignment weights so the intuitively correct alignment scores better, and so will be chosen by the alignment algorithm.

4.2.2 Alignment Weights for Integrated ASR and Disfluency Models

We use two sets of weights for finding an alignment between the reference and the integrated model output. We use the standard alignment weights described in Section 4.2.1 for aligning fluent words, and slightly modify the weights to discourage aligning disfluent words in the reference transcript with words in the integrated model output. For the fluent region, a correct alignment operation is a *copy* while for the disfluent region, a correct alignment is a *deletion*. As shown in Table 1, the alignment cost is slightly higher for inserting, copying and substituting a disfluent word and slightly lower for deleting a disfluent word. Having a higher alignment cost for disfluent words results in a preference to align the words in integrated model outputs with fluent words as illustrated in Figure 3. Ambiguities can still arise even if disfluent words have a higher alignment cost than fluent words. However, these ambiguities do not affect the disfluency evaluation scores as our disfluency evaluation scores only depend on whether a word is disfluent or not.

In summary, although WER is a standard metric for evaluating ASR models, it is insufficient for evaluating integrated ASR and disfluency systems as it measures the overall word recognition accuracy, and does not specifically focus on how well the end-to-end system handles disfluencies. Alternatively, we propose a modified alignment strategy with different weights for fluent and disfluent word alignments. Thus, it is possible to calculate word

---

\[^7\]https://github.com/usnistgov/SCTK
Figure 3: Ref is the reference transcript which is human-transcribed speech with gold disfluency labels, shown in red. Align refers to the alignment between the integrated ASR and disfluency model output and the reference transcript generated by the modified alignment weights where different costs are allocated for aligning fluent and disfluent words.

| Operation | Fluent | Disfluent |
|-----------|--------|-----------|
| Copy (c)  | 0      | 0 + 10⁻⁷  |
| Insertion (i) | 3      | 3 + 10⁻⁷  |
| Deletion (d) | 3      | 3 - 10⁻⁷  |
| Substitution (s) | 4      | 4 + 10⁻⁷  |

Table 1: The two sets of weights used to align disfluent and fluent words separately.

error rate on fluent and disfluent regions separately. Our new evaluation metrics and alignment weights are useful for aligning and evaluating any system trained to remove disfluency in its output.

5 Experiments

We train our ASR models on two corpora of English conversational telephone speech: (i) Switchboard-1 Release 2 (SWBD) (Godfrey and Holliman, 1993) and (ii) Fisher Part 1 (Cieri et al., 2004) and Part 2 (Cieri et al., 2005). Switchboard-1 Release 2 is a collection of about 2400 telephone conversations (260 hours of speech), of which 1126 conversations were hand-annotated with disfluencies as part of the Penn Treebank Release 3 dataset (Marcus et al., 1999), which we refer to as gold data. The original release of Switchboard does not contain time-alignment annotations which are required for preparing the ASR training data. Mississippi State University researchers ran a clean-up project on Switchboard-1 Release 2 and produced accurate time alignments which we use for speech segmentation.

Fisher Part 1 and 2 are a collection of about 11700 telephone conversations (total 2000 hours of speech), which contain time-aligned transcripts, but no disfluency annotations. To identify the disfluencies in the Fisher data and the portion of the SWBD data with no gold disfluency labels, we use an “off-the-shelf” state-of-the-art disfluency detection model (Jamshid Lou et al., 2019). We call the automatically annotated data silver data. The disfluency detection model used to obtain silver data is a BERT-based self-attentive parser that jointly finds a constituency parse tree and detects disfluencies in speech transcripts. Different versions of the parser are available; we use the parser trained on the Penn Treebank Release 3 Switchboard corpus with partial words kept in the data for which they reported an f-score of 94.4 on the SWBD dev set. We remove all disfluent words (tagged as “EDITED” and “INTJ”), as well as partial words (words tagged “XX” and words ending in “.”) and punctuation from the SWBD and Fisher data. We use the standard data splits for training our models as well as the language models (Charniak and Johnson, 2001): training data consists of the sw[23].text files and fe_03_*_.txt, dev data consists of the sw4[5-9].text files and test data consists of the sw4[0-1].text files.

We consider a pipeline approach as our baseline and apply the “off-the-shelf” disfluency detection model to the output of the baseline ASR models. As our evaluation metrics, we report WER, FER and DER for the end-to-end and the pipeline models. Since the goal of an integrated system is to find only the fluent words, we evaluate WER only on fluent words. For calculating FER and DER, we align the output of the integrated models and the output of the pipeline ASR and disfluency detector to the reference transcripts with gold disfluency labels. We report DER results for detecting edited disfluencies, interjections and partial words. In order to have a fair comparison, we report all the results of the paper on the subset of the Switchboard dev and test sets with gold disfluency labels.

Available at http://www.openslr.org/5/
6 Results

We compare the performance of our integrated ASR and disfluency models (trained on fluent transcripts) to the baseline pipeline models consisting of the ASR models (trained on disfluent transcripts) combined with the “off-the-shelf” disfluency detection model. As shown in Table 3, the WER of end-to-end models is higher than that of the pipeline models, indicating that word recognition is generally more difficult when the ASR model is trained on disfluent speech and fluent transcripts.

The baseline ASR models (without disfluency detection) have the lowest error rate on fluent areas (i.e. FER). However, when we apply the “off-the-shelf” disfluency detection model on the output of the baseline ASR models, FER significantly increases, indicating that errors made by the “off-the-self” disfluency detection model harm the detection of fluent words. The fluent error rate of the end-to-end models is lower than that of the pipeline models. Comparing the disfluent error rate of the end-to-end and baseline models, we realize that simply training an ASR model on disfluent speech and fluent transcripts significantly decreases the number of disfluencies in the output. However, this is not sufficient for outperforming the baseline pipeline models on detecting and removing disfluencies, indicating that more complex architectures or mechanisms are required for effective end-to-end ASR and disfluency detection. The pipeline models have access to more information (i.e. the annotated disfluencies) than the end-to-end models; however, it is not clear if or how it would improve system performance. Of the three end-to-end models, the Transformer has the best performance on disfluency removal which we speculate is due to the self-attention mechanism which has been previously shown effective in detecting disfluencies in speech transcripts (Tran et al., 2019; Jamshid Lou et al., 2019; Dong et al., 2019; Wang et al., 2019a). We also compare the end-to-end ASR and disfluency removal models with the pipeline ASR and disfluency detection on the Switchboard test set, as demonstrated in Table 4.

Table 2: Some examples from the SWBD dev set and corresponding transcripts. Ref is the reference transcript which is human-transcribed speech with gold disfluency labels, shown in red. E2E represents the output of the end-to-end Transformer ASR and disfluency removal model. Pipe refers to the output of the pipelineTransformer ASR and “off-the-shelf” disfluency detection model.

| Ref | 1. the rights of that individual have been you know impugned . . . |
|-----|---------------------------------------------------------------|
| 1   | . . . the rights of that individual or have been you know immune . . . |
| E2E | . . . the rights of that individual have been you know impugned . . . |

| Ref | 2. I actually my dad’s almost ninety . . . |
|-----|------------------------------------------|
| 2   | . . . I yeah cause my dad’s almost ninety . . . |
| E2E | . . . actually my dad’s almost ninety . . . |

| Ref | 3. I’ve been to a couple of games before |
|-----|----------------------------------------|
| 3   | I’ve been to a couple of games before |
| E2E | I’ve been to a couple of games before |

| Ref | 4. So from that standpoint it’s pretty small it’s pretty small |
|-----|-------------------------------------------------------------|
| 4   | So from that standpoint it’s pretty small it’s pretty small |
| E2E | So from that standpoint it’s pretty small it’s pretty small |

| Ref | 5. It’s I’m sure there’s a lot of differences in the way it’s done now and then |
|-----|---------------------------------------------------------------------------------|
| 5   | I’m sure there’s a lot of differences in the way it’s done now and then |
| E2E | I’m sure there’s a lot of differences in the way it’s done now and then |

To further investigate the disfluency removal performance of the three end-to-end models, we randomly select 100 sentences from the Switchboard dev set containing disfluencies. We categorize disfluencies into repetition, correction and restart according to the Shriberg (1994) typology of speech repairs. Repetitions are repairs where the reparandum and repair portions of the disfluency are identical, while corrections are where the reparandum and repairs differ (which are much harder to detect). Restarts are where the speaker abandons a sentence.
Table 3: Word error rate (WER) with respect to the fluent transcript, fluent error rate (FER) and disfluent error rate (DER) on the SWBD dev set. “Gold Transcripts + DF” = the gold transcripts followed by the “off-the-shelf” disfluency detection model (DF), “base” = the baseline ASR (trained on disfluent transcripts), “pipe” = the baseline ASR + DF, “E2E” = end-to-end ASR and disfluency removal (trained on fluent transcripts).

| Model          | WER  | FER  | DER  |
|---------------|------|------|------|
| CTC (base)    | 12.4 | 10.2 | 93.5 |
| CTC (pipe)    | 12.4 | 13.5 | 20.2 |
| CTC (E2E)     | 13.6 | 11.1 | 22.6 |
| Seq2Seq (base)| 8.7  | 7.7  | 95.0 |
| Seq2Seq (pipe)| 8.7  | 9.1  | 18.8 |
| Seq2Seq (E2E) | 10.5 | 8.9  | 21.8 |
| Transformer (base)| 9.5 | 8.5  | 94.6 |
| Transformer (pipe)| 9.5 | 10.2 | 18.6 |
| Transformer (E2E)| 11.2| 9.4  | 20.2 |
| Gold Transcripts + DF | - | 2.2  | 16.8 |

Table 4: Word error rate (WER) with respect to the fluent transcripts, fluent error rate (FER) and disfluent error rate (DER) on the SWBD test set. “Gold Transcripts + DF” = the gold transcripts followed by the “off-the-shelf” disfluency detection model (DF), “base” = the baseline ASR (trained on disfluent transcripts), “pipe” = the baseline ASR + DF, “E2E” = end-to-end ASR and disfluency removal (trained on fluent transcripts).

| Model          | WER  | FER  | DER  |
|---------------|------|------|------|
| CTC (base)    | 12.5 | 11.8 | 94.7 |
| CTC (pipe)    | 12.5 | 13.1 | 23.3 |
| CTC (E2E)     | 14.3 | 12.4 | 26.2 |
| Seq2Seq (base)| 11.2 | 10.4 | 22.6 |
| Seq2Seq (pipe)| 11.2 | 11.6 | 22.6 |
| Seq2Seq (E2E) | 12.2 | 10.1 | 25.6 |
| Transformer (base)| 11.2| 10.5 | 95.2 |
| Transformer (pipe)| 11.2| 12.1 | 22.2 |
| Transformer (E2E)| 13.8| 11.6 | 24.0 |
| Gold Transcripts + DF | - | 2.7  | 17.7 |

Table 5: Disfluent error rate (DER) of three end-to-end ASR and disfluency removal models for different types of disfluency on a subset of the SWBD dev set containing 145 disfluent structures — including 76 repetitions (Rep.), 58 corrections (Cor.) and 11 restarts (Res.).

| Model          | Rep. | Cor. | Res. | All  |
|---------------|------|------|------|------|
| CTC           | 23.6 | 33.5 | 36.0 | 28.9 |
| Seq2Seq       | 22.5 | 29.5 | 35.1 | 27.1 |
| Transformer   | 22.1 | 25.8 | 35.1 | 25.0 |

6.1 Qualitative Analysis

We conduct a qualitative analysis on the Switchboard dev set to characterize the disfluencies that the pipeline model cannot detect but the end-to-end model can and vice versa. We provide representative examples in Table 2. ASR errors usually lead to disfluency detection errors in the pipeline model (see #1-3). On the other hand, the end-to-end model sometimes fails at detecting repetitions which are the most common type of disfluency. While the specialized disfluency detector is good at detecting repetitions in speech transcripts, it seems that identifying repetitions in speech signal is non-trivial for the end-to-end model (see #4 and #5).

7 Conclusion

We showed WER is insufficient for evaluating end-to-end ASR and disfluency removal systems and alternatively introduced two metrics reflecting how well end-to-end systems handle disfluencies. We also showed the disfluency removal performance of end-to-end models is comparable to that of pipeline ASR and specialized high performance disfluency models. The best end-to-end system uses a Transformer, that’s what the best “off-the-shelf” disfluency detection system does, too. In the future, we aim to retrain the “off-the-shelf” disfluency detector on ASR outputs using cross-validation. It is interesting to investigate how modifying the training loss would affect disfluency detection in end-to-end models. We also intend to augment the end-to-end Transformer model with special mechanisms which have been previously shown effective for disfluency detection in speech transcripts.
References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv e-prints, abs/1409.0473.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies (NAACL), pages 118–126, Stroudsburg, USA.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2004. Fisher English Training Speech Part 1 Transcripts LDC2004T19. Published by: Linguistic Data Consortium, Philadelphia, USA.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2005. Fisher English Training Speech Part 2 Transcripts LDC2005T19. Published by: Linguistic Data Consortium, Philadelphia, USA.

Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu. 2019. Adapting translation models for transcript disfluency detection. In Proceedings of the 2019 AAAI Conference on Artificial Intelligence, pages 6351–6358, Honolulu, USA.

John Godfrey and Edward Holliman. 1993. Switchboard-1 Release 2 LDC97S62. Published by: Linguistic Data Consortium, Philadelphia, USA.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the International Conference on Machine Learning (ICML), pages 369–376, Pittsburgh, USA.

Çalar Gülcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv e-prints, abs/1503.03535.

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

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency detection using auto-correlational neural networks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4610–4619, Brussels, Belgium.

Paria Jamshid Lou and Mark Johnson. 2020. Improving disfluency detection by self-training a self-attentive model. arXiv e-prints, abs/2004.05323.

Paria Jamshid Lou, Yufei Wang, and Mark Johnson. 2019. Neural constituency parsing of speech transcripts. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 2756–2765, Minneapolis, USA.

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 (ACL), pages 33–39, Barcelona, Spain.

Oleksii Kuchaiiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Jason Li, Huyen Nguyen, Carl Case, and Paulius Micikevicius. 2018. Mixed-precision training for NLP and speech recognition with OpenSeq2Seq. arXiv e-prints, abs/1805.10387.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. 2019. Jasper: An end-to-end convolutional neural acoustic model. In Proceedings of the 19th Annual Conference of the International Speech Communication Association (Interspeech), pages 71–75, Graz, Austria.

Yang Liu, Elisabeth Shriberg, Andreas Stolckeand, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. IEEE Transactions on Audio, Speech, and Language Processing, 14(5):1526–1540.

Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 LDC99T42. Published by: Linguistic Data Consortium, Philadelphia, USA.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. 2019. Transformers with convolutional context for ASR. arXiv e-prints, abs/1904.11660.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) (Demonstrations), pages 48–53, Minneapolis, USA.

Daniel Povey, Arnab Ghoshal, et al. 2011. The Kaldi Speech Recognition Toolkit. In Proceedings of IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU), Hawaii, USA.

Elizabeth Salesky, Matthias Sperber, and Alexander Waibel. 2019. Fluent translations from disfluent speech in end-to-end speech translation. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 2786–2792, Minneapolis, USA.
Elizabeth Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Department of Psychology, University of California, Berkeley, USA.

Trang Tran, Jiahong Yuan, Yang Liu, and Mari Ostendorf. 2019. *On the role of style in parsing speech with neural models*. In *Proceedings the 19th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 4190–4194, Graz, Austria.

Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. 2018. *Semi-supervised disfluency detection*. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 3529–3538, Santa Fe, USA.

Shaolei Wang, Wanxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2019a. *Multi-task self-supervised learning for disfluency detection*. arXiv e-prints, abs/1908.05378.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. *A neural attention model for disfluency detection*. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 278–287, Osaka, Japan.

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. *Automatic disfluency removal for improving spoken language translation*. In *Proceedings of 2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5214–5217, Dallas, USA.

Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur. 2019b. *Espresso: A fast end-to-end neural speech recognition toolkit*. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 136–143, Singapore, Singapore.

Vicky Zayats and Mari Ostendorf. 2019. *Giving attention to the unexpected: Using prosody innovations in disfluency detection*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 86–95, Minneapolis, USA.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. *Disfluency detection using a bidirectional LSTM*. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2523–2527, San Francisco, USA.