Best of Both Worlds: Making High Accuracy Non-incremental Transformer-based Disfluency Detection Incremental

Morteza Rohanian and **Julian Hough**
Cognitive Science Group
School of Electronic Engineering and Computer Science
Queen Mary University of London
{m.rohanian, j.hough} @qmul.ac.uk

Abstract

While Transformer-based text classifiers pre-trained on large volumes of text have yielded significant improvements on a wide range of computational linguistics tasks, their implementations have been unsuitable for live incremental processing thus far, operating only on the level of complete sentence inputs. We address the challenge of introducing methods for word-by-word left-to-right incremental processing to Transformers such as BERT, models without an intrinsic sense of linear order. We modify the training method and live decoding of non-incremental models to detect speech disfluencies with minimum latency and without pre-segmentation of dialogue acts. We experiment with several decoding methods to predict the rightward context of the word currently being processed using a GPT-2 language model and apply a BERT-based disfluency detector to sequences, including predicted words. We show our method of incrementalising Transformers maintains most of their high non-incremental performance while operating strictly incrementally. We also evaluate our models' incremental performance to establish the trade-off between incremental performance and final performance, using different prediction strategies. We apply our system to incremental speech recognition results as they arrive into a live system and achieve state-of-the-art results in this setting.

1 Introduction

Conversational systems provide a significant addition to the present approaches in mental health care delivery. Interactions with these conversational agents have been shown to contain observable indicators of cognitive states, such as the rate of filled pauses and different temporal and turn-related features (Gratch et al., 2014). Alzheimer's Disease (AD) patients, for example, have trouble performing tasks that leverage semantic information; they have difficulties with verbal fluency and object recognition. AD patients speak more slowly with long pauses and spend extra time looking for the correct word, which leads to speech disfluency (López-de Ipiña et al., 2013; Nasreen et al., 2021). Disfluency markers can be key features for identifying certain cognitive disorders for application in conversational agents (Rohanian et al., 2020).

Such conversational systems are primarily used for content processing, which is then analyzed offline. There is much work on detecting disfluencies for offline analysis of transcripts. However, given that these disfluency detection models do not work for live systems and depend on rich transcription data, including pre-segmentation of dialogue acts, to facilitate more cost-effective analysis of other data, we need systems capable of performing directly and incrementally off the speech signal, or at least from the results of automatic speech recognition (ASR) as they arrive in the system.

As it receives word-by-word data, an incremental model must operate with minimum latency and do so without changing its initial assumptions and delivering its best decisions as early as possible following the principles outlined in (Hough and Purver, 2014). Here we design and evaluating models that work with online, incremental speech recognition output to detect disfluencies with varying levels of granularity.

The best neural language encoders currently used in computational linguistics consider word sequences as a whole, and their implementations have been unsuitable for live incremental processing. Transformers (Vaswani et al., 2017), for instance, operate on representations that do not naturally have an organizing principle of linear word order. We analyze how these models work under incremental frameworks, where it is essential to present partial output relying on partial input pro-

vided up to a certain time step that may occur in interactive healthcare systems. We explore whether we can adjust such models to function incrementally and how useful they are in terms of overall accuracy and incremental metrics.

To further enhance the models' incremental performance, we use two general strategies to adjust the training regime and the real-time procedure: incremental training ('chunk-based' training and add-M training) and incremental decoding (constant latency and prophecies). We employ three prominent decoding methods to predict the rightward context of the word currently being processed: beam search, top-k sampling, and top-p sampling. We also measure our models' incremental performance to set the trade-off between incremental performance and final performance.

2 Related Work

Although considerable work has been done on detecting disfluencies, much of this work uses transcripts as texts rather than live speech inputs, with the goal of 'cleaning' the disfluent content for post-processing purposes. They are almost exclusively conducted on pre-segmented utterances of the Switchboard corpus of telephone conversations (Godfrey et al., 1992). Several disfluency detection efforts involve sentence-based parsing and language models (Johnson and Charniak, 2004; Zwarts et al., 2010). Sequence labeling models with start-inside-outside (BIO) style tags have been used in recent neural sequence approaches to disfluency detection based on bi-directional Long Short Term Memory (BiLSTM) networks and Transformers, in which the sequences are available in full (Zayats et al., 2016; Lou and Johnson, 2020; Wang et al., 2020).

Such offline methods are insufficient if we intend to infer meaning from repairs and edit words for disfluency detection in real-time, which is beneficial in a healthcare domain dialogue system that seeks to get a consistent and clear understanding of user statements and the user's cognitive state.

Methods based on strictly incremental operation have been rare. Hough and Purver (2014) used a line of classifiers and language model features in a strong incremental operating system without looking ahead. Incremental dependency parsing combined with the removal of disfluency was also studied (Rasooli and Tetreault, 2015). Some studies have used recurrent neural networks for live disfluency identification. Using a basic Elman Recurrent Neural Network (RNN), Hough and Schlangen (2015) investigated incremental processing, with an objective coupling detection accuracy with low latency.

Language models have been used as an additional task for the identification of disfluencies, relying on the intuition that disfluencies can be detected by divergences from clean language models, with Johnson and Charniak (2004)'s noisy channel model beginning this effort. Shalyminov et al. (2018) made language modelling an auxiliary task to disfluency detection in a deep multi-task learning (MTL) set-up, gaining accuracy over a vanilla RNN tagger. POS tags have also been used as an input for detecting disfluencies, showing slight increases in disfluency detection over using word values alone (Purver et al., 2018).

While the work above operates only on transcripts pre-segmented into utterances, recent research has been performed on combining disfluency detection with utterance segmentation. This was done in a joint tagset of disfluency, and utterance segmentation tags by (Hough and Schlangen, 2017), showing an improvement over the performance of the individual tasks, and (Rohanian and Hough, 2020) show an improvement in both tasks when framed as a multi-task learning (MTL) set-up with a Long Short-term Memory network (LSTM), also simultaneously doing POS-tagging and language modelling.

The recent live incremental systems fall short of the same accuracies achievable on pre-segmented transcripts, so there is a natural interest in using the best non-incremental sequence models and adapting them for incrementality. Madureira and Schlangen (2020) take up this effort in several other sequence tagging and classification tasks, showing how bidirectional encoders and Transformers can be modified to work incrementally. To reduce the impact of the partiality of the input, the models predict future content and wait for more rightward context. Dalvi et al. (2018) also use truncated inputs during the training phase of live machine translation to address the partial input sentence decoding problem Bidirectional encoders face. Here, we seek to add to this growing effort to investigate the trade-off of incremental performance against the final output quality of deep neural network-based language processing, applied to incremental disfluency detection.

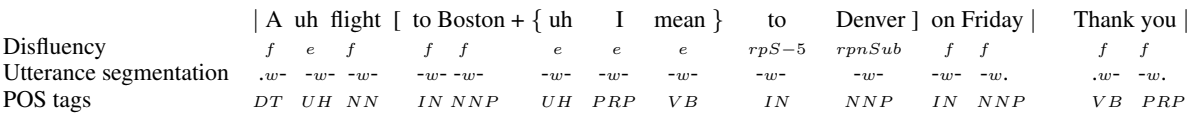


Figure 1: An utterance with the disfluency tags (repair structures and edit terms) and the utterance segmentation tags and POS tags used for preprocessing.

3 Disfluency Detection

Disfluencies are generally assumed to have a reparandum-interregnum-repair structure in their fullest form as speech repairs (Shriberg, 1994; Meteer et al., 1995). A reparandum is a stretch of speech later corrected by the speaker; the corrected expression is a repair, the beginning of which is referred to as *repair onset*. An interregnum word is a filler or a reference expression between the repair and reparandum, usually an interruption and hesitation step when the speaker expresses a repair, giving the structure as in (1).

$$\underset{}{\texttt{John}} \; \underbrace{\texttt{[likes}}_{\text{reparandum}} \; \texttt{+} \; \underbrace{\texttt{\{ uh \}}}_{\text{interregnum}} \; \underbrace{\texttt{loves]}}_{\text{repair}} \; \texttt{Mary} \tag{1}$$

In the absence of reparandum and repair, the disfluency is reduced to an isolated *edit term*. A marked, lexicalised edit term such as a filled pause ("uh" or "um") or more phrasal terms such as "I mean" and "you know" may occur. The identification of these elements and their structure is then the task of disfluency detection.

The task of detecting incremental disfluencies adds to the difficulty of doing this in real-time, word-by-word, from left to right. Disfluency recognition is then treated as the same problem that a human processor faces with a disfluent expression: only when an interregnum is detected, or maybe even when a repair is initiated, does it become clear that the earlier content is now to be regarded as 'to be repaired,' i.e., to be classified as a reparandum. Therefore, the task cannot be defined as a simple sequence labeling task in which the tags for the reparandum, interregnum, and repair phases are assigned left-to-right over words as seen in the above example; in this case, it will require the assumption that "likes" would be repaired, at a time when there is no data to make it available.

We use a tag set that encodes the start of the reparandum only at a time when it can be inferred, primarily when the repair starts – the disfluency detection task is to tag words as in the top line of tags in Fig. 1 as either fluent (f) an edit term (e), a repair onset word ($rpS{-}N$ for the reparandum starting N words back) and a repair end word of the type repeat ($rpnRep$), substitution ($rpnSub$) or delete ($rpnDel$).

4 Model

To incrementalise a Transformer-based model for word-by-word disfluency detection, we devise a model built on top of a pre-trained BERT architecture (Devlin et al., 2019) with a Conditional Random Field (CRF) output architecture to tag sequences with tags such as those in the top line of Fig. 1. We use a BERT-based encoder and try different strategies to incrementalise the system's operation and output, using language models to predict future word sequences as described in Section 5 while maintaining BERT's non-incremental quality.

Utterance segmentation Our models are designed to work not only with pre-segmented data but also on raw transcripts and ASR results, where utterance segmentation is required to leverage the use of sentence-based linguistic knowledge in BERT. Utterance segmentation has a clear interdependence with and influence on the detection of disfluency as disfluent restarts and repairs may be incorrectly predicted at fluent utterance boundaries without segmentation. In this paper, rather than performing utterance segmentation in tandem with disfluency detection, we perform it on words as they arrive in the system as a live segmentation task before sending the current prefix of the utterance to the disfluency detection system. We use the word-by-word segmentation system from (Rohanian and Hough, 2020) where four output tags define ranges of transcribed words or word hypotheses using a BIES tag scheme (Beginning, Inside, End, and Single) to allow for the prediction of an utterance ending. The tagset allows information to be captured from the context of the word to decide whether this word continues a current utterance (the – prefix) or starts anew (the . prefix), and also allows live prediction of whether the next word will continue the current utterance (the – suffix) or

whether the current word finishes the utterance (the . suffix). An example of the scheme is shown in the second line of Fig. 1.

CRF We use a CRF output architecture to predict a tag for every token. Although this model generates predictions for the whole sequence, the labels are outputted individually. There are important dependencies between adjacent labels in disfluency detection, and explicit modeling of these relationships can help. The addition of the CRF enables the model to test for the most optimal path across all available label sequences.

4.1 Input Features

In addition to the word values, we also experiment with two other inputs:

Part-of-speech tags POS tags may enhance the identification of disfluencies on various settings. POS tagging helps detect disfluency structure as the parallelism between the reparandum and repair in substitutions, as shown in the repeated $IN\ NNP$ sequences in Fig. 1.

Word timings We also experiment with the duration from the ending of the previous word to the ending of the current word as it enters the system, either from ground truth word transcriptions or from ASR results.

5 Strategies for Incrementalising BERT

Here we describe the different strategies we used to modify the training and live decoding methods of non-incremental models to detect speech disfluencies word-by-word incrementally. The general principle is to leverage high accuracy full sequence classification using BERT but deploying it on sequences, including future predictions for words up to the hypothesised end of the current utterance.

5.1 Modifying the Training Procedure

Training is performed on full sentences/utterances, but the decoder produces outputs based on partial input data at the test time. This disparity between training and decoding can potentially affect our models' performance. Based on (Dalvi et al., 2018), we present two methods to address this issue: chunk-based training and add-M training.

Chunk-based training In chunk-based training, we change the training scheme by removing the ends of each sentence in the training set and simply break each training sentence into chunks of N tokens. Here we use 2 and 3 for N.

Add-M training We begin with the first N words in training sentences in add-M training. The next training instances are then generated by $N+M, N+2M, N+3M...$ words before the end of the sentence is reached. In our experiments, we found setting N=1 and M=1 worked best.

5.2 Modifying the Decoding Procedure

Constant latency The technique of constant latency requires allowing certain 'future' words to be seen before a label to previous words is given. It is a form of look-ahead based on Baumann et al. (2011), in which before making the first decision with respect to previous time steps, the processor is required to wait for some correct context. We explore the one- or two-word contexts of our input. This suggests that the model generates the first label for word t after the word $t+1$ is seen or the model observes words $t+1$ and $t+2$ before tagging word t. This has an inherent limit on the latency achievable, and we use this as a baseline incremental decoding system.

Prophecy-based decoding For our other decoding strategies, we use a 'prophecy'-based approach to predicting future word sequences, following the task of open-ended language generation, which, given an input text passage as context, is to produce text that constitutes a cohesive continuation (Holtzman et al., 2019). Inspired by (Madureira and Schlangen, 2020), using the GPT-2 language model (Radford et al., 2019), we first give each word as a left context and create a continuation until the end of an utterance to create a hypothetical complete context that satisfies the requirements of the models' non-incremental structure.

Formally, with m tokens $x_1...x_m$ as our context, the task is to create the next n continuation tokens to achieve the completed sequence $x_1...x_{m+n}$. It is assumed that the models compute $P(x_{1:m+n})$ using a standard left-to-right decomposition of the text probability as in (2). This process is used to build the utterance continuation token-by-token using a specific decoding technique.

$$P(x_{1:m+n}) = \prod_{i=1}^{m+n} P(x_i|x_1...x_{i-1}) \qquad (2)$$

Three of the most common decoding methods are used in this paper: Beam search, Top-k sampling, and Top-p sampling. Example word sequence prophecies from these decoding methods

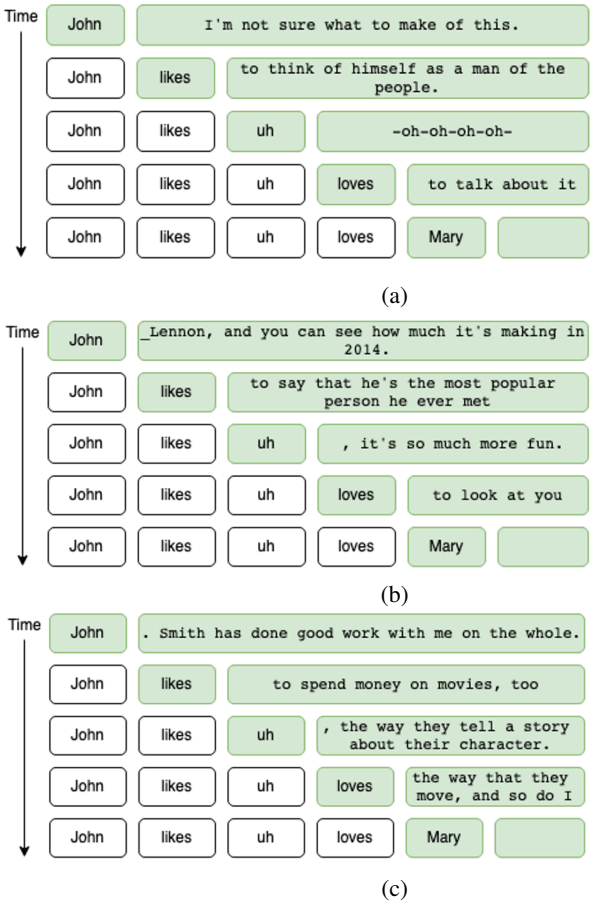


Figure 2: Using a 'prophecy'-based approach to predict future word sequences, following the task of open-ended language generation with three different decoding methods. (a) Beam search. (b) Top-k sampling. (c) Top-p sampling.

are shown in Fig. 2. The right-most block shows the prediction of the continuation of the word sequences as each new word in the sequence "John likes uh loves Mary" is fed into the language model.

Beam search Assuming that the model gives a greater likelihood to better quality text, we are looking for a sequence with the highest probability. During the search, a group of stacks is used to hold hypotheses. Beam size N is used to manage the search space by expanding the top N hypotheses in the existing stack. We used beam size 10 for all the models.

Top-k sampling We define sampling as randomly choosing the next word based on its conditional probability distribution as in (3).

$$x_i \sim P(x|x_{1:i-1}) \tag{3}$$

In the Top-k sampling, the most probable next k words are extracted and the probability mass is redistributed between only the following k words (Fan et al., 2018). Given a distribution $P(x|x_{1:i-1})$, we extract its top-k vocabulary $V^{(k)} \subset V$ as the set of size k which maximizes $\sum_{x \in V^{(k)}} P(x|x_{1:i-1})$. After an initial investigation, we set k to 50 in all experiments.

Top-p sampling Rather than selecting only the most probable K words, in Top-p sampling, we select the smallest possible range of words with their total likelihood exceeds the probability p (Holtzman et al., 2019). The probability mass is then redistributed between this set of words. With this method, the size of the word set will dynamically adjust based on the probability distribution of the next word. With the distribution $P(x|x_{1:i-1})$, we consider its top-p sequence, with vocabulary $V^{(p)} \subset V$ as the smallest set with $P(x|x_{1:i-1}) \geq p$. We set $p = 0.95$.

6 Experimental Set-up

We train on transcripts and test on both transcripts and ASR hypotheses. All models in testing have strictly word-by-word left to right input. In addition to using the latest word hypothesis as input, we train and evaluate the presented models with two kinds of additional inputs: time elapsed from the end of the previous word (hypothesis) to the current one and the POS tag of the current word. Results on the development set were used to find the best model to be evaluated on the test set.

We used the data from (Hough and Schlangen, 2017) for ASR hypotheses – this was generated by a free trial version of IBM's Watson Speech-To-Text service for incremental ASR. The service offers good quality ASR on noisy data-on our selected held-out data on Switchboard, and the average WER is 26.5%. The Watson service, crucially for our task, does not filter out hesitation markers or disfluencies (Baumann et al., 2017). The service delivers results incrementally, so silence-based end-pointing is not used. It also outputs word timings, which are close enough to the source timings to use as features in the live version of our system.

The word embedding for LSTM was initialised with 50-dimensional embedding trained on Google News (Mikolov et al., 2013). The model has been implemented using Tensorflow 2.1. We train all models for a maximum of 50 epochs; otherwise, stop training if there is no improvement on the best score on the validation set after 7 epochs.

A large version of the pre-trained BERT is used with 340M parameters (24-layer blocks, 16 self-

Input	Model	Pre-segmented transcripts (per word)			Transcripts (per word)			ASR (per 10 second window)		
		F_{rm}	F_{rpS}	F_e	F_{rm}	F_{rpS}	F_e	F_{rm}	F_{rpS}	F_e
Words	STIR (HS'15/ PHH'18)	0.741 / 0.749	-/0.827	0.880/-	-	-	-	-	-	-
	RNN (HS'15)	0.689	-	0.873	-	-	-	-	-	-
	LSTM	0.686	0.771	0.928	0.59	0.678	0.904	-	0.548	0.726
	LSTM-MTL (RH'20)	0.737	0.799	0.938	0.629	0.743	0.917	-	0.573	0.757
	BERT	0.758	0.851	**0.960**	0.659	0.782	0.947	**0.524**	0.603	**0.812**
Word + Timings	LSTM	0.681	0.777	0.921	0.623	0.718	0.908	-	0.555	0.721
	LSTM-MTL (RH'20)	0.741	0.812	0.929	0.629	0.741	0.922	-	0.559	0.751
	BERT	0.752	0.842	0.958	**0.678**	0.791	0.939	0.502	0.594	0.793
Word + POS	STIR (HP'14 / PHH'18)	**0.779 / 0.768**	-/0.833	0.937/-	-	-	-	-	-	-
	RNN (HS'15 / PHH'18)	0.711 / 0.668	-/0.790	0.902/-	-	-	-	-	-	-
	LSTM joint tagset (HS'17)	-	-	-	0.599	0.686	0.907	-	0.557	0.726
	LSTM-MTL (SEL'18)	0.753	0.816	0.919	-	-	-	-	0.548	-
Words + Timings + POS	LSTM joint tagset (HS'17)	-	-	-	0.601	0.719	0.918	-	0.555	0.727
	LSTM	0.692	0.778	0.931	0.601	0.720	0.910	-	0.557	0.727
	LSTM-MTL (RH'20)	0.743	0.811	0.932	0.633	0.743	0.931	-	0.571	0.757
	BERT	0.757	**0.853**	0.958	0.676	**0.802**	**0.944**	0.522	**0.605**	0.809

Table 1: Final disfluency detection accuracy results on Switchboard data

attention heads, and 1024 hidden-size) for the model. In our analysis, when fine-tuning BERT, we followed the hyper-parameters of (Devlin et al., 2019). Since the datasets we use are tokenized, and each token has a matching tag, we adopt the directions provided by (Devlin et al., 2019) to deal with the sub-tokenization of BERT: to determine its label, the scores of the first sub-token are used, and further sub-token scores are discarded.

Data We use standard Switchboard training data (all conversation numbers starting sw2*,sw3 * in the Penn Treebank III release: 100k utterances, 650k words) and use standard held-out data (PTB III files sw4[5-9] *: 6.4k utterances, 49k words) as our validation set. We test on the standard test data (PTB III files 4[0-1] *) with partial words and punctuation stripped away from all files. We only choose a subset of the held-out and test data for the ASR results in assessment, whereby both channels achieve below 40 percent WER to ensure good separation- this left us with 18 dialogues in validation data and 17 dialogues for test data.

6.1 Evaluation Criteria

We calculate F1 accuracy for repair onset detection F_{rpS} and for edit term words F_e, which includes interregna and F_{rm} for reparandum detection. Performing the task live, on hypotheses of speech recognition that may not be quite equivalent to the annotated gold-standard transcription involves the use of time-based local accuracy metrics in a time window (i.e., within this time frame, has a disfluency been detected, even if not on the identical words?)-we, therefore, measure the F1 score over 10-second windows of each speaker's channel.

For incremental performance, we measure latency and output stability over time. We use the first time to detection (FTD) metric of (Zwarts et al., 2010) for latency: the average latency (in number of words) before the first detection of a gold standard repair onset or edit term word. For stability, we evaluate the edit overhead (EO) of output labels (Baumann et al., 2011), the proportion of the unnecessary edits (insertions and deletions) required to achieve the final labels produced by the model, with perfect performance being 0%.

6.2 Competitor Baselines

We compare our incrementalised BERT model against a number of existing baselines, largely from existing incremental disfluency detection systems trained and tested on the same data:

STIR (HP'14/HS'15/PHH'18): Hough and Purver (2014)'s STrongly Incremental Repair detection (STIR) non-deep model using n-gram language model features in a pipeline of Random Forest classifiers. The reparandum is detected by a backward search, showing robustness for longer lengths of repair compared to deep sequence tagging models (Purver et al., 2018). A state-of-the-art incremental model on pre-segmented transcripts.

RNN (HS'15): (Hough and Schlangen, 2015)'s RNN-based model, the first deep learning-based

Training Scheme	Model	Final output F1			Incrementality	
		F_{rm}	F_{rpS}	F_e	EO	FTD
Chunk	LSTM	.591	.674	.901	0.21	0.06
	MTL	.631	.739	.911	0.41	0.07
	BERT	.647	.780	.938	0.61	0.32
Add-M	LSTM	.598	.683	.909	**0.20**	**0.03**
	MTL	.628	.751	.921	0.38	0.10
	BERT	**.664**	**.788**	**.949**	0.60	0.31

Table 2: Final accuracy vs. incremental performance trade-off in the different models on un-segmented transcripts.

Model	F1		
	Repeats	Substitution	Deletes
With Standard Training			
LSTM	0.94	0.70	0.48
MTL	**0.96**	0.72	0.46
BERT	**0.96**	0.77	**0.54**
With Add-M Training			
LSTM	0.95	0.71	0.48
MTL	**0.96**	0.73	0.47
BERT	**0.96**	**0.79**	**0.54**

Table 3: Performance on different types of repair.

incremental disfluency detection model using the same tagset as in our model. Results from Purver et al. (2018) are used, which reproduced the model with some degradation in the results.

LSTM: An LSTM version of Hough and Schlangen (2015) on pre-segmented transcripts

LSTM joint tagset (HS'17) Hough and Schlangen (2017)'s model, which simultaneously predicts utterance segmentation using a joint tag set of utterance segmentation tags and disfluency tags, the latter of which is the same as our own. This is the only other work to use word timing information and to be testable on ASR results.

LSTM-MTL (SEL'18) Shalyminov et al. (2018)'s multi-task learning model, which tags according to our tag set but simultaneously does language modelling by predicting the probability of the current word given the history. Also adds ground-truth POS tags to input.

LSTM-MTL (RH'20): Rohanian and Hough (2020)'s multi-task learning model, which simultaneously predicts utterance segmentation, POS tags and language model probabilities, exhibiting state-of-the-art results for a strictly incremental deep model. The model is used as described by the authors and also here with the addition of timing information and gold standard POS information (as opposed to simultaneously predicted POS tags). It is also applied to ASR results as it is a suitable model to do so. This same model provides the automatic live utterance segmentation in our own model.

7 Results

The results in terms of the final output of our best performing incremental BERT system in the three testing regimes versus its competitors is shown in Table 1.[1] We found our best model was the add-M trained model, and the best decoding strategy was using top-p sampling for predicting future words.

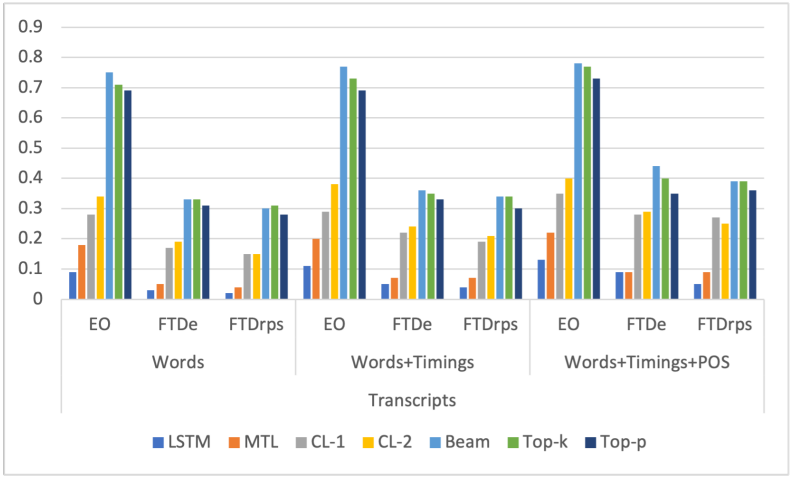
Disfluency detection on transcripts For repair detection, our system's best F_{rpS} score for detecting repair onsets on pre-segmented transcripts at 0.853 beats state-of-the-art incremental systems. This performance degrades using automatic segmentation to 0.802, a state-of-the-art result for this setting. Its F_{rm} accuracy of 0.757 on reparandum words on pre-segmented transcripts is only beaten by HP'14/PHH'18 model using word and POS input, making it a state-of-the-art strictly incremental deep model. This performance degrades to 0.678 on raw transcripts but is a state-of-the-art result for this setting. In terms of edit term detection, state-of-the-art detection results of 0.960 and 0.944 are achieved on the pre-segmented and unsegmented settings, improving over the existing benchmarks of HP'14 and RH'20. These results suggest we have achieved the aim of a strictly incremental model achieving high final accuracies.

Disfluency detection on ASR results Using the ASR results from HS'17 for comparison, a significant improvement can be seen over the previously reported results on F_{rpS} and F_e per 10-second window, improving from 0.557 to 0.605 and from 0.727 to 0.809 respectively. Given the previously reported best system gave strong correlations in terms of real repair rates, this is encouraging that our system could be very useful in a live setting.

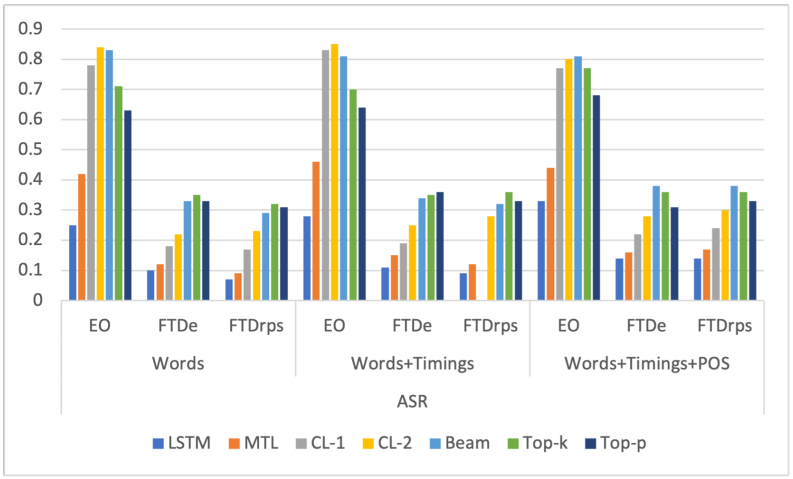
7.1 Incremental Performance

The purpose of this paper was to adapt a high-performing, non-incremental model for incremental operation. As can be seen in Table 2 and in Fig. 3, while our BERT model with top-p sample utterance prediction outperforms the multi-task

[1] Experiments are reproducible from https://github.com/mortezaro/tr-disfluency



(a)



(b)

Figure 3: Incremental results of first time to detection (FTD) metric for rpS and e and edit overhead (EO) for disfluency detection labels.(a) On unsegmented transcripts. (b) On ASR results.

model and vanilla LSTM model in terms of final output accuracy, its incremental output stability is slightly below its competitors, with the best edit overhead of 63% unnecessary edits versus 25% (LSTM joint tagset (HS'17)) and 42% (LSTM-MTL (RH'20)) on ASR results, meaning the output is slightly, though not severely, more jittery.

Of the prophecy-based approaches, we found the top-p sampling method gave the most stable results (EO=61% with chunk training, EO=60% with add-M training) and beam search gave the least stable. As shown in Fig. 3, while the constant latency approaches offer large advantages in EO over prophecy-based models on transcripts, that advantage disappears on ASR results, where the prophecy models generally outperform them. As can be seen in Table 2, there is a slight improvement in stability across all systems using the add-M training regime for final output and incremental performance.

In terms of latency, results are even more encouraging, with the best FTD for rpS of 0.31 words (versus 0.03 and 0.07) on transcripts, which shows a relatively short latency of detecting the repair for the first time– this suggests a responsive, sensitive system.

7.2 Error Analysis

We conduct an error analysis in terms of performance on different repair types and in terms of repairs with different lengths. Table 3 shows the performance in terms of F_{rpS} score on detecting repairs of the three different types: verbatim repeats, substitutions, and deletes (restarts). Our BERT model performs best, either jointly or uniquely, across all three types, with a gain of 0.06 over its nearest competitors for substitutions and deletes. Through large-scale training, the enhanced linguistic knowledge equips it to recognize the syntactic

Model	Reparandum length						Reparandum length of nested disfluencies					
	1	2	3	4	5	6	1	2	3	4	5	6
With Standard Training												
LSTM	.843	.675	.405	.311	.134	.131	.747	.586	.382	**.320**	.110	.104
MTL	.856	.683	.431	.335	.134	.131	.763	.586	**.405**	.291	.110	.104
BERT	**.892**	.716	.469	**.379**	**.310**	**.187**	.818	.623	**.405**	**.320**	**.130**	**.140**
With Add-M Training												
LSTM	.843	.675	.434	.334	.134	.131	.741	.586	.382	**.320**	.110	.104
MTL	.851	.709	.468	.335	.134	.131	.779	.586	**.405**	.291	**.130**	.104
BERT	**.892**	**.719**	**.472**	**.379**	**.310**	**.187**	**.833**	**.645**	**.405**	**.320**	**.130**	**.140**

Table 4: F1 of models on repairs with reparanda of different length

and lexical parallelism in more complex repairs while retaining high accuracy on repeats. Table 4 shows the degradation in performance in detecting repairs of different lengths. With Add-M training, the BERT model degrades less and performs (joint) best on all lengths and nested disfluencies. While the performance on length five repairs is considerably better than the other deep models, the 0.187 accuracy on length six repairs is what gives it a slight disadvantage compared to the HP'14 explicit backtracking system (reported as high as 0.500 in PHH'18), which likely accounts for the lower F_{rm} score despite the superior F_{rpS} score of our system.

8 Discussion and Conclusion

Our incremental GPT-2 and BERT-driven system performs well at detecting repair disfluencies on pre-segmented and unsegmented transcripts, achieving state-of-the-art results for a strictly incremental repair onset detection. Our system is competitive at reparadnum word detection and achieves state-of-the-art results in edit term detection. The results on ASR transcripts are also state-of-the-art.

The high sequence-final performance comes at the expense of marginally increased jitter in the word-by-word output, but with sensitive and fast repair detection, on average first detecting the repair under a third of a second after the end of the repair onset word. These results suggest it is beginning to enjoy the best of both worlds in leveraging the right-ward context which BERT uses for its high performance, while the continuation predictions from the GPT-2 model are good enough to allow good incremental performance before the true right-ward context is available.

The linguistic knowledge in the BERT model allows it to recognize parallelism in reparandum and repair phases and the absence thereof to increase performance on detecting substitution and delete repairs. This improvement to existing deep disfluency detection models, and, with appropriate use of open-ended language generation techniques with a GPT-2 language model, its good incremental performance, is consistent with a growing body of work (Heeman and Allen, 1999; Johnson and Charniak, 2004; Zwarts et al., 2010; Hough and Purver, 2014; Shalyminov et al., 2018; Rohanian and Hough, 2020), showing good language modelling can lead to good disfluency detection, as they are inherently part of the same process.

Our system still fails to detect longer repairs compared to an explicit backtracking mechanism like (Hough and Purver, 2014). While the vanishing gradient problem is partly overcome here, the strictly left-to-right constraint on decoding puts memory limitations on any repair detection system. In future, we will explore efficient ways to navigate this space whilst not filtering out rarer repair forms.

The results on ASR results show our disfluency detection system is ready for use in a live setting with a good degree of accuracy, and work is currently underway to use it to help detect a variety of different cognitive conditions, including Alzheimer's Disease, in a live diagnostic system.

Acknowledgments

We thank the anonymous ACL-IJCNLP reviewers for their helpful comments and Matthew Purver for his continuous support and supervision on the wider project.

References

Timo Baumann, Okko Buß, and David Schlangen. 2011. Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, 2(1):113–141.

Timo Baumann, Casey Kennington, Julian Hough, and David Schlangen. 2017. Recognising conversational speech: What an incremental asr should do for a dialogue system and how to get there. In *Dialogues with social robots*, pages 421–432. Springer.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 517–520. IEEE Computer Society.

Jonathan Gratch, Ron Artstein, Gale M Lucas, Giota Stratou, Stefan Scherer, Angela Nazarian, Rachel Wood, Jill Boberg, David DeVault, Stacy Marsella, et al. 2014. The distress analysis interview corpus of human and computer interviews. In *LREC*, pages 3123–3128.

Peter A Heeman and James Allen. 1999. Speech repains, intonational phrases, and discourse markers: modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–572.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Julian Hough and Matthew Purver. 2014. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89.

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

Julian Hough and David Schlangen. 2017. Joint, incremental disfluency detection and utterance segmentation from speech. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 326–336.

Karmele López-de Ipiña, Jesus-Bernardino Alonso, Carlos Manuel Travieso, Jordi Solé-Casals, Harkaitz Egiraun, Marcos Faundez-Zanuy, Aitzol Ezeiza, Nora Barroso, Miriam Ecay-Torres, Pablo Martinez-Lage, et al. 2013. On the selection of non-invasive methods based on speech analysis oriented to automatic alzheimer disease diagnosis. *Sensors*, 13(5):6730–6745.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *ACL*, pages 33–39.

Paria Jamshid Lou and Mark Johnson. 2020. Improving disfluency detection by self-training a self-attentive model. *arXiv preprint arXiv:2004.05323*.

Brielen Madureira and David Schlangen. 2020. Incremental processing in the age of non-incremental encoders: An empirical assessment of bidirectional models for incremental NLU. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 357–374, Online. Association for Computational Linguistics.

M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Disfluency annotation stylebook for the switchboard corpus. ms. Technical report, Department of Computer and Information Science, University of Pennsylvania.

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

Shamila Nasreen, Morteza Rohanian, Matthew Purver, and Julian Hough. 2021. Alzheimer's dementia recognition from spontaneous speech using disfluency and interactional features. *Frontiers in Computer Science*, 3:49.

Matthew Purver, Julian Hough, and Christine Howes. 2018. Computational models of miscommunication phenomena. *Topics in cognitive science*, 10(2):425–451.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Morteza Rohanian and Julian Hough. 2020. Reframing incremental deep language models for dialogue processing with multi-task learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 497–507, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Morteza Rohanian, Julian Hough, and Matthew Purver. 2020. Multi-modal fusion with gating using audio,

lexical and disfluency features for alzheimer's dementia recognition from spontaneous speech. In *Proc. Interspeech*, pages 2187–2191.

Igor Shalyminov, Arash Eshghi, and Oliver Lemon. 2018. Multi-task learning for domain-general spoken disfluency detection in dialogue systems. In *Proceedings of the 22nd SemDial Workshop on the Semantics and Pragmatics of Dialogue (AixDial)*, Aix-en-Provence.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California, Berkeley.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Shaolei Wang, Wangxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2020. Multi-task self-supervised learning for disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9193–9200.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1371–1378.