Joint, Incremental Disfluency Detection and Utterance Segmentation from Speech

Julian Hough and David Schlangen
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literature
Bielefeld University
firstname.lastname@uni-bielefeld.de

Abstract

We present the joint task of incremental disfluency detection and utterance segmentation and a simple deep learning system which performs it on transcripts and ASR results. We show how the constraints of the two tasks interact. Our joint-task system outperforms the equivalent individual task systems, provides competitive results and is suitable for future use in conversation agents in the psychiatric domain.

1 Introduction

Artificial conversational systems promise to be a valuable addition to the existing set of psychiatric health care delivery solutions. As artificial systems, they can ensure that interview protocols are followed, and, perhaps surprisingly, due to being "just a computer", even seem to increase their interlocutors' willingness to disclose (Lucas et al., 2014). Interactions with such conversational agents have been shown to contain interpretable markers of psychological distress, such as rate of filled pauses, speaking rate, and various temporal, utterance and turn-related interactional features (DeVault et al., 2013). Filled pauses and disfluencies in general have also been shown to predict outcomes to psychiatric treatment (Howes et al., 2012; McCabe et al., 2013).

Currently, these systems are only used to elicit material that is then analysed offline. For offline analysis of transcripts with gold standard utterance segmentation, much work exists on detecting disfluencies (Johnson and Charniak, 2004; Qian and Liu, 2013; Honnibal and Johnson, 2014). To enable more cost-effective analysis, however, and possibly even let the interaction script itself be dependent on an analysis hypothesis, it would be better to be able to work directly off the speech signal, and online (incrementally). This is what we explore in this paper, presenting and evaluating a model that works with online, incremental speech recognition output to detect disfluencies with various degrees of fine-grainedness.

As a second contribution, we combine incremental disfluency detection with another lower-level task that is important for responsive conversational systems, namely the detection of turn-taking opportunities through detection of utterance boundaries. (See for example (Schlangen and Skantze, 2011) for arguments for incremental processing and responsive turn-taking in conversational systems, and (Schlangen, 2006; Atterer et al., 2008; Raux, 2008; Manuvinakurike et al., 2016, *inter alia*) for examples of incremental utterance segmentation). Besides both being relevant for interactive health assessment systems, these tasks also have an immanent connection, as the approach typically used for turn-end detection is simply waiting for a silence of a certain duration, and hence is mislead by intra-turn silent disfluencies. Similarly, without gold standard segmentation, disfluent restarts and repairs may be predicted at fluent utterance boundaries. We hence conjecture that the tasks can profitably be done jointly.

2 Related Work

As a separate task, there has been extensive work on utterance segmentation. Cuendet (2006) reports an NIST-SU utterance segmentation error rate result on the Switchboard corpus at 48.50, using a combination of lexical and acoustic features. Ang et al. (2005) report NIST-SU scores in the region of 34.35–45.92 on the ICSI Meeting Corpus. Martínez-Hinarejos et al. (2015) report state-of-the-art dialogue act segmentation results on Switchboard at 23.0 NIST-SU, however this is not on the level of full dialogues, but on pre-segmented turn stretches. For the equivalent task of sentence boundary detection, Seeker et al. (2016) report an F-score of 0.7665 on Switchboard data, using a joint dependency parsing framework, and Xu et al. (2014) implement a deep learning architecture and report an 0.810 F-score and 35.9 NIST-SU error rate on broadcast news speech using prosodic and lexical features using a DNN for prosodic features, combined with a CRF classifier. However scaling this to spontaneous speech and the challenges of incrementality explained here, is yet to be tested.

Strongly incremental approaches to the task are rare, however (Atterer et al., 2008) achieve a word-by-word F-score of 0.511 on predicting whether the current word is the end of the utterance (dialogue act) on Switchboard, and using ground-truth syntactic information indicating sentence structure information achieve 0.559.

Disfluency detection on pre-segmented utterances in the Switchboard corpus has also had a lot of attention, and has also reached high performance (Johnson and Charniak, 2004; Georgila, 2009; Qian and Liu, 2013; Honnibal and Johnson, 2014). On detection on Switchboard transcripts, Honnibal and Johnson (2014) achieve 0.841 reparandum word accuracy using a joint dependency parsing approach, and Hough and Purver (2014) in a strongly incrementally operating system without look-ahead achieve 0.779, using a pipeline of classifiers and language model features. The potentially live approaches tend to use acoustic information (Moniz et al., 2015) and do not perform on a comparable level to their transcription-based task analogues, nor achieve the same fine-grained analysis of disfluency structure, which is often needed to identify the disfluency type and compute its meaning.

Live incremental approaches to both tasks have not been able to benefit from reliable ASR hypotheses arriving in a timely manner until recently. Now the arrival of improved performance, in terms of low Word Error Rate (WER) and better live performance properties is making this possible (Baumann et al., 2016). In this paper we define a joint task in a live setting. After defining the task we present a simple deep learning system which simultaneously detects disfluencies and predicts up-coming utterance boundaries from incremental word hypotheses and derived information.

3 The Tasks: Real-time disfluency prediction and utterance segmentation

3.1 Incremental disfluency detection

Disfluencies, in their fullest form as speech repairs, are typically assumed to have a tripartite *reparandum-interregnum-repair* structure (terms originally proposed by Shriberg (1994)), as exhibited by the following example.

$$\texttt{John}\ \underbrace{\texttt{[likes +}}_{\texttt{reparandum}}\ \underbrace{\texttt{\{ uh \}}}_{\texttt{interregnum}}\ \underbrace{\texttt{loves]}}_{\texttt{repair}}\ \texttt{Mary} \qquad (1)$$

If reparandum and repair are absent, the disfluency reduces to an isolated *edit term*. In the example given here, the interregnum is filled by a marked, lexicalised edit term, but more phrasal terms such as *I mean* and *you know* can also occur.

The task of disfluency detection then is to recognise these elements and their structure, and the task of *incremental* disfluency detection adds the challenge of doing this in real-time, from "left-to-right". In that latter setting, detection runs into the same problem as a human processor of such an utterance: Only by the time the interregnum is encountered, or possibly even only when the repair is seen, does it become clear that earlier material now is to be considered as "to be repaired" (reparandum).[1] Hence, the task cannot be set up as a straightforward sequence labelling task where the tags "reparandum", "interregnum" and "repair" are distributed left-to-right over words as indicated in the example above; in this example, it would unfairly require the prediction that "likes" is *going to be* repaired, at a point when no evidence is available for making it.

We follow Hough and Schlangen (2015) and use a tag set that encodes the reparandum start only at a time when it can be guessed, namely at the onset of the actual repair. This is illustrated in Figure 1 in the "disfluency (complex)" row. Here, the word at the repair onset, "to", gets tagged as repair onset (*rpS*) and, at the same time, as repairing material beginning 5 tokens in the past (*-5*, yielding the complex label *rpS-5*). Additionally, we annotate all repair words (as *rpMid*, if the word is neither first nor last word of the repair, and together with the disfluency type, if it is the final word; here, the

[1]Looking at it from a different perspective, this problem has been called the *continuation problem* by Levelt (1983): the repair material can only be integrated with the previous material, if it is identified as replacing the reparandum.

	| A	uh	flight	[to	Boston +	{ uh	I	mean }	to	Denver]	on	Friday |	Thank	you |
Disfluency (simple)	f	e	f	f	f	e	e	e	rpS	f	f	f	f	f
Disfluency (complex)	f	e	f	f	f	e	e	e	$rpS{-}5$	$rpESub$	f	f	f	f
Utterance segmentation	.w-	-w-	-w-	-w-	-w-	-w-	-w-	-w-	-w-	-w-	-w-	-w.	.w-	-w.
Joint task (simple)	.f-	-e-	-f-	-f-	-f-	-e-	-e-	-e-	-rpS-	-f-	-f-	-f.	.f-	-f.
Joint task (complex)	.f-	-e-	-f-	-f-	-f-	-e-	-e-	-e-	-$rpS{-}5$-	-$rpESub$-	-f-	-f.	.f-	-f.

Figure 1: An utterance with the traditional repair disfluency and segmentation annotation in-line (Shriberg, 1994; Meteer et al., 1995) and our incrementally-oriented tag schemes

label is *rpESub* for substitution),[2] editing terms (*e*) and fluent material (*f*) as well. From the complex tag set, we can reconstruct the disfluency structure as in (1) in a strongly incremental fashion. We also define a reduced tag set (shown in Figure 1 as "disfluency (simple)" that only tags fluent words, editing terms, and the repair onset.

3.2 Incremental utterance segmentation

We formulate incremental utterance segmentation as the judgement in real time as to when the current utterance is going to end, and so like (Schlangen, 2006; Atterer et al., 2008), we move from purely *reactive* approach, signalled by silence, to *prediction*. To allow prediction to be possible we use four tags for classifying stretches of acoustic data (which can be the time spans of forced aligned gold standard words, or the word hypotheses timings provided by an ASR), which are equivalent to a BIES (Beginning, Inside, End and Single) scheme for utterances– see Table 1.

The tag set allows evidence from the prior context of the word (the acoustic and linguistic information preceding the word) to be used to predict whether this word continues a current utterance (the `-` prefix) or starts anew (the `.` prefix), and also permits the online prediction of whether the next word (or segment) will continue the current utterance (the `-` suffix) or the current word ends the utterance (the `.` suffix). From these utterance boundary predictions can be derived when `-w.` or `.w.` is predicted (i.e. "will end utterance"). The tag set is summarized in Table 1 and an example is in Fig. 1, row "utterance segmentation".

3.3 Defining the joint task

Studying the two phenomena in natural dialogue corpora, for example in terms of rich transcription mark-up in the SWBD annotation manual (Meteer et al., 1995), there are several constraints:

	-w-	-w.	.w-	.w.
f	1	1	1	1
e	1	1	1	1
rpS	1	1	0	0

	-w-	-w.	.w-	.w.
f	1	1	1	1
e	1	1	1	1
rpS-[1-8]	1	0	0	0
rpMid	1	0	0	0
rpESub	1	1	0	0
rpEDel	1	1	0	0
rpS-[1-8]ESub	1	1	0	0
rpS-[1-8]EDel	1	1	0	0

Figure 2: The joint tag set for the task. 1= tag in set, simple (top) and complex (bottom).

- *C1* Repair onsets cannot begin an utterance (by definition of first position repairs needing a preceding reparandum).
- *C2* Repairs must be completed within the utterance in which they begin.
- *C3* Utterances can be interrupted or abandoned, but these are different to within-dialogue-act repairs.

Given these constraints, we can generate a joint tag set as a subset of the cross product of both tag schemes. The utterance segmentation tags in Table 1 are combined with the simple strongly incremental disfluency tags described in §3.1. The joint set for both the simple and complex tasks is in Fig. 2, where 1 indicates the tag is in the set and 0 otherwise. In the simple task, there are 10 tags. The joint set for the full task including disfluency structure detection has 53 possible tags (rather than the full cross product, which would be 92). In reality, in the training corpus, only 43 of these possible combinations were found, so this constituted our tag set in practice. See Fig. 1 (bottom 2 rows) for example sequences.

3.4 Research questions

Given the formulation of the joint task, we would like to ask the following questions of scalable, automatic approaches to it:

[2]The other repair type is delete *rpEDel*. Verbatim reparandum-repair repetitions are subsumed by *rpESub*.

`-w-`	a word which continues the current utterance and whose following word will continue it
`-w.`	a word which continues the current utterance and is the last word of it
`.w-`	a word which is the beginning of an utterance and whose following word will continue it
`.w.`	a word constituting an entire utterance

Table 1: The tag set for the continuity of each word within a dialogue act

Q1 Given the interaction between the two tasks, can a system which performs both jointly help improve equivalent systems doing the individual tasks?

Q2 Given the incremental availability of word timings from state-of-the-art ASR, to what extent can word timing data help performance of either task?

Q3 To what extent is it possible to achieve a good online accuracy vs. final accuracy trade-off in a live, incremental, system?

To address these questions we use a combination of a deep learning architecture for sequence labelling and incremental decoding techniques which we will now explain.

4 LSTMs and Incremental Decoding for Live Prediction

Our systems consist of deep learning sequence models which consume incoming words and use word embeddings in addition to other features to predict disfluency and utterance segmentation labels for each word, in a strictly left-to-right, word-by-word fashion. We also use word timings as input to a separate classifier whose output is combined with that of the deep learning architecture in an incremental decoder. See Fig. 3 for the overall architecture. We describe the elements of the system below.

4.1 Input Features

In our systems we use the following input features:

- Words in a backwards window from the most recent word (transcribed or ASR)
- Durations of words in the current window (from transcription or ASR word timings)
- Part-Of-Speech (POS) tags for words in current window (either reference, or from an incremental CRF tagger)

For incremental ASR, we use the free trial version of IBM's Watson Speech-To-Text service.[3] The service provides good quality ASR on noisy

[3] `https://www.ibm.com/watson/developercloud/speech-to-text.html`

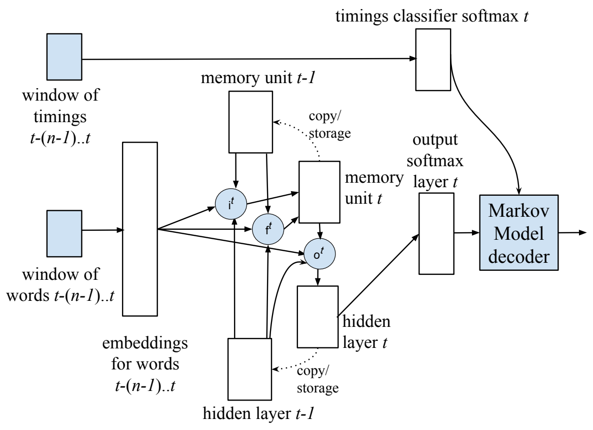

Figure 3: Schematic structure of the system.

data- on our selected heldout data on Switchboard, the average WER is 26.5%. The Watson service, crucially for our task, does not filter out hesitation markers or disfluencies, which is rare for current web-based services (Baumann et al., 2016). The service also outputs results incrementally, so silence-based end-pointing is not used. The service also returns word timings, which upon manual inspection were close enough to the reference timings to use as features in the live version of our system. In this paper, the durations are not features in the principal RNN but in an orthogonal logistic regression classifier– see §4.3.

For POS-tagging, we use the NLTK CRF tagger, which when trained on our training data and tested on our heldout data achieves 0.915 accuracy on all tags, which was sufficiently good for our purposes. Crucially, for the label `UH`, which is important evidence for an edit term, it achieves an F-score of 0.959.

4.2 Architectures

We use two well-studied deep learning architectures for our sequence labelling task– the Elman Recurrent Neural Network (RNN) and the Long Short-Term Memory (LSTM) RNN. Architecturally the RNNs here reproduce approximately the identical set-up as described in (Mesnil et al., 2013; Hough and Schlangen, 2015).

Input and word embeddings Following (Mesnil et al., 2013), we use 1-of-N, or 'one-hot', vectors as our raw input to the network, which provide unique indices to dense vectors in a word embedding matrix. The initial word embeddings were obtained from Switchboard data using the python implementation of `word2vec` in gensim,[4] using a skip-gram context model. The training data for the initial embeddings was cleaned of disfluencies, effecting a 'clean' language model (Johnson and Charniak, 2004). These embeddings were then further updated as part of the objective function during the task-specific training itself. Instead of single word/POS inputs we use context windows which, like n-gram language models, are *backwards* from the current word. The internal representation of context windows of length n in the network is created through the ordered concatenation of the n corresponding word embedding vectors of size 50, resulting in an input to the network of dimension $\mathbb{R}^{50n}$. We use n =2 in our experiments here.

RNN architecture and activation functions In addition to the embedding layer, we use a (recurrent) hidden layer of 50 nodes and an output layer the size of our training tag sets (43 nodes for the complex task and 10 nodes for the simple task). The standard Elman RNN dynamics in the recurrent hidden layer at time t is as in (3), where the hidden layer $h(t)$ is calculated as the Sigmoid function (2) of the addition of the weight matrix U' applied via dot product to the current input vector $x(t)$ and the weight matrix V' applied via dot product to the stored previous value of the hidden layer at time $t{-}1$, i.e. $h(t{-}1)$.

$$s(x) = \frac{1}{1+e^{-x}} \tag{2}$$

$$h(t) = s(U'x(t) + V'h(t{-}1)) \tag{3}$$

We use the standard softmax function for the node activation function of the output layer.

At decoding time, the compression of the context into the hidden layer allows us to save the current state of the decode live compactly from ASR results as they become available to the network. In order to integrate the new incoming words and POS tags with the history, it is only necessary to store the current hidden layer activation $h(t)$ (and the output softmax layer too, if that is being used by another process), and wait for new information to the input layer.

[4]`http://radimrehurek.com/gensim/`

LSTM unit In our LSTM, we include recurrent LSTM units that uses the input $x(t)$, the hidden state activation $h(t{-}1)$, and memory cell activation $c(t{-}1)$ to compute the hidden state activation $h(t)$ at time t. It uses a combination of a memory cell c and three types of gates: input gate i, forget gate f, and output gate o to decide if the input needs to be remembered (using the input gate), when the previous memory needs to be retained (forget gate), and when the memory content needs to be output (using the output gate). For each time step t the cell activations $c(t)$ and $h(t)$ are computed by the below steps, whereby the $\odot$ is element-wise multiplication.

$$\begin{aligned} i(t) &= s(W_i'x(t) + U_i'h(t{-}1) + V_i'c(t{-}1)) \qquad (4) \\ f(t) &= s(W_f'x(t) + U_f'h(t{-}1) + V_f'c(t{-}1)) \\ c(t) &= f(t) \odot c(t{-}1) + i(t) \odot tanh(W_c'x(t) + U_c'h(t{-}1)) \\ o(t) &= s(W_o'x(t) + U_o'h(t{-}1) + V_o'c(t)) \\ h(t) &= o(t) \odot tanh(c(t)) \end{aligned}$$

While many more weight matrices need to be learned (all the W', U' and V' subscripted matrices), as with the standard RNN, at decoding time it is efficient to store the current decoding state in a compact way, as it is only neccessary to save the activation of the memory cell $c(t)$ and the hidden layer $h(t)$ to save the current state of the network. See Fig. 3 for the schematic overall disfluency detection architecture for the LSTM.

Learning: error function and parameter update As is common for RNNs (De Mulder et al., 2015) we use negative log likelihood loss (NLL) as a cost function and use stochastic gradient descent over the parameters, including the embedding vectors, to minimize it. We use a batch size of 9 words, consistent with our repair tag scheme. Both networks use a learning rate of 0.005 and L2 regularisation on the parameters to be learned with a weight of 0.0001.

4.3 Incremental decoding and timing driven classifier

Markov model For decoding optimization we use Viterbi decoding on the sequence of softmax output distributions from the network in the spirit of (Guo et al., 2014). We use a Markov model which is hand-crafted to ensure legal tag sequences are outputted for the given tag set. In our joint task, this permits 'late' detection of an utterance boundary if the probability for a `-w.` and following `.w-` or `.w.` tag on their own are not the arg max, but their combined probability permits the best sequence. Similarly, in the complex task, repairs where evidence of a repair end tag is strong, but the repair onset tag was not the arg max can be detected at the repair end. From an incremental perspective, in Viterbi decoding there is the danger of output 'jitter'. We investigate how different output representations have different effects on output prediction stability in our evaluation.

Timing driven classifier As an edition to the decoding step, we experimented with an independent timing driven classifier which consumes the durations of the last three words and outputs a probability that this is a fluent continuation or the beginning of a new utterance. We train a logistic regression classifier on our training data. Combining this two-class probability with the probability of the relevant utterance segmentation tags in decoding boosted performance considerably.

5 Evaluation Criteria

Accuracy On transcripts, we calculate repair onset detection accuracy $\mathbf{F_{rpS}}$, where applicable reparandum word accuracy $\mathbf{F_{rm}}$, and F1 accuracy for edit term words $\mathbf{F_{e}}$, which includes interregna. For utterance segementation we also use word-level F1 scores for utterance boundaries (end-of-utterance words) $\mathbf{F_{uttSeg}}$. Carrying out the task live, on speech recognition hypotheses which very well may not be identical to the annotated gold-standard transcription, requires the use of time-based metrics of local accuracy in a time window (i.e. *within this time window, has a disfluency/utterance boundary been detected, even if not on the identical words?*)– we therefore calculate the F1 score over 10 second windows of each speaker's channel. While this window-ing can give higher scores on certain phenomena, it tends to follow the word-level F-score so is a good time-based indicator of accuracy.

For utterance segmentation, for comparison to previous work we also use **NIST-SU** error rate (Ang et al., 2005). NIST-SU is the ratio of the number of incorrect utterance boundary hypotheses (missed boundaries and false positives) made by a system to the number of reference boundaries.

For a more coarse-grained metric which includes both tasks, which is useful in our target domain of interactions in a clinical context (Howes et al., 2014), we look at the **rpS : UttSeg ratio per speaker correlation** (Pearson's R). This gives us the best approximation as to how good the system is at estimating repair rate per utterance.

Timeliness and diachronic metrics Crucial for the live nature of the system, we measure latency (i.e. *how close to the actual time a disfluency or boundary event occurred has one been predicted?*) and also stability of output over time (i.e. *how much does the output change?*). For latency we use Zwarts et al. (2010)'s *time-to-detection* metric: the average distance (in numbers of words) consumed before first detection of gold standard repairs from the repair onset word, $\mathbf{TD_{rpS}}$.[5] We generalize this measure to the other tags of interest to give $\mathbf{TD_{e}}$ and $\mathbf{TD_{uttSeg}}$ and also, particularly crucially for the ASR results, report the metrics in terms of time in seconds.[6]

For stability, incorporating insights from the evaluation of incremental processors by Baumann et al. (2011), we measure the **edit overhead (EO)** of the output labels– this is the percentage of unnecessary edits (insertions and deletions) required to get to the final labels outputted by the system.

6 Experimental Set-up

We experiment with the 2 joint output representations in Fig. 1 and implement an RNN and LSTM using Theano (Bergstra et al., 2010) as an extension to the code in Mesnil et al. (2013). We also run the 3 individual versions of the tasks with the tag sets shown in Fig. 1 for comparison. We also train a word timings driven classifier which adds information to the decoding step as explained above to try to answer Q2.[7]

Data We train on transcripts and test on both transcripts and ASR hypotheses. We use the standard Switchboard training data for disfluency detection (all conversation numbers beginning sw2*,sw3* in the Penn Treebank III release: 100k utterances, 650K words) and use the standard heldout 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 punctuation removed from all files.[8] For

[5]Our measure is in fact one word earlier by default than Zwarts et al. (2010) as we take detection after the end of the repair onset word as the earliest possible detection point.

[6]These measures only apply to repairs and utterance boundaries detected correctly.

[7]All experiments are reproducible. The code can be downloaded at `https://github.com/dsg-bielefeld/deep_disfluency`

[8]We include partial words as these may in theory become available from the ASR in the live setting.

Eval. Method	System	F_{rm} (per word)	F_{rps} (per word)	F_{rps} (per 10s window)	F_e (per word)	F_e (per 10s window)	F_{uttSeg} (per word)	F_{uttSeg} (per 10s window)	NIST SU (word)	rps / $uttSeg$ / speaker correl.
Transcript	LSTM +timing	-	0.719	0.764	**0.918**	0.889	**0.748**	0.707	**43.64**	0.91
	LSTM	-	**0.720**	**0.766**	0.915	**0.890**	0.688	0.666	51.89	**0.92**
	LSTM(complex) +timing	**0.601**	0.693	0.730	0.91	0.888	0.707	0.685	50.07	0.82
	LSTM(complex)	0.599	0.686	0.727	0.907	0.889	0.638	0.638	58.91	0.84
	RNN +timing	-	0.683	0.730	0.909	0.886	0.704	**0.710**	52.42	0.86
	RNN	-	0.685	0.728	0.908	0.884	0.647	0.635	57.75	0.87
	RNN(complex) +timing	0.572	0.663	0.715	0.908	0.882	0.699	0.669	50.89	0.83
	RNN(complex)	0.568	0.659	0.713	0.905	0.882	0.621	0.613	60.74	0.81
ASR	LSTM +timing	-	-	0.551	-	**0.727**	-	**0.685**	-	0.72
	LSTM	-	-	0.548	-	0.726	-	0.630	-	0.79
	LSTM(complex) +timing	-	-	0.555	-	0.721	-	0.665	-	0.68
	LSTM(complex)	-	-	**0.557**	-	0.721	-	0.601	-	0.67
	RNN +timing	-	-	0.542	-	0.718	-	0.681	-	0.69
	RNN	-	-	0.540	-	0.718	-	0.627	-	0.68
	RNN(complex) +timing	-	-	0.543	-	0.718	-	0.663	-	0.72
	RNN(complex)	-	-	0.540	-	0.718	-	0.577	-	**0.81**

Table 2: Non-incremental (dialogue-final) results on transcripts and ASR results.

Eval. Method	System	F_{rps} (per word)	F_{rps} (per 10s window)	F_e (per word)	F_e (per 10s window)	F_{uttSeg} (per word)	F_{uttSeg} (per 10s window)	NIST SU (word)
Transcript	LSTM (uttSeg only)	-	-	-	-	0.727	0.679	46.17
	LSTM (disf only)	0.711	0.760	0.912	0.886	-	-	-
	LSTM (joint task)	**0.719**	**0.764**	**0.918**	**0.889**	**0.748**	**0.707**	**43.64**
ASR	LSTM (uttSeg only)	-	-	-	-	-	0.657	-
	LSTM (disf only)	-	0.531	-	0.721	-	-	-
	LSTM (joint task)	-	**0.551**	-	**0.727**	-	**0.685**	-

Table 3: Comparison of the joint vs. individual task performances

the ASR results evaluation, we only select a subset of the heldout and test data whereby both channels achieved below 40% WER to ensure good separation– this left us with 18 dialogues in the validation data and 17 dialogues for testing.

We train all RNNs for a maximum of 50 epochs else halt training if there is no improvement on the best F_{rm} score on the transcript validation set after 10 epochs.

7 Results and Discussion

Our dialogue-final accuracy results are in Table 2. On transcripts, our best per-word F_{rpS} reaches 0.720 and best F_e reaches 0.918. For utterance segmentation, perword accuracy reaches 0.748 and the lowest NIST-SU error rate is 43.64. This is competitive with (Seeker et al., 2016)'s 0.767 F-score and out-performs (Cuendet, 2006) on the Switchboard data. The best rpS : $uttSeg$ correlation per speaker reaches 0.92 ($p<0.0001$).

In comparison to incremental approaches, we outperform (Atterer et al., 2008)'s 0.511 accuracy on end-of-utterance. Their work allows no prediction lag in a strictly incremental setting, so is at a disadvantage, however our result of 0.748 on transcripts is reported alongside the average time to detection of 0.399 words, which suggests on average the uttSeg when predicted correctly, is done so with no latency.

With the exception of one metric, the LSTM outperforms the RNN on transcripts. The systems using the timing model in general outperform those with lexical information only on the utterance segmentation metrics, whilst not having an impact on disfluency detection.

According to the window-based accuracies, on ASR results there is significant degradation in accuracy for repair onsets (best F_{rpS}=0.557) however utterance segmentation did not suffer the same loss, with the best system achieving 0.685 accuracy. The rpS : $uttSeg$ Pearson's R correlation per speaker reaches 0.81 ($p<0.0001$) in a system with otherwise poor performance– the second

Eval. method	System	TTD_{rps} (word)	TTD_{rps} (time in s)	TTD_e (word)	TTD_e (time in s)	TTD_{uttSeg} (word)	TTD_{uttSeg} (time in s)	EO
Transcript	LSTM +timing	0.004	0.253	0.573	0.614	0.399	1.837	11.44
	LSTM	**0.003**	**0.248**	0.591	0.605	0.327	1.114	11.05
	LSTM(complex)	0.093	0.281	**0.114**	**0.348**	**0.283**	**1.107**	**7.63**
	LSTM(complex) +timing	0.090	0.293	0.135	0.483	0.369	1.960	8.51
ASR	LSTM +timing	-	0.202	-	0.734	-	3.247	20.71
	LSTM	-	**0.199**	-	0.649	-	**1.645**	**20.44**
	LSTM(complex)	-	0.236	-	**0.341**	-	2.303	20.70
	LSTM(complex) +timing	-	0.239	-	0.594	-	4.099	21.46

Table 4: Incremental results on transcripts and ASR results.

best achieved was 0.79 ($p<0.0001$).

For disfluency detection, standard approaches use pre-segmented utterances to evaluate performance, so this result is difficult to compare. However in the simple task, the accuracy of 0.720 repair onset prediction is respectable (comparable to (Georgila, 2009)), and is useful enough to allow realistic relative repair rates, in line with our motivation. The complex tagging system performs poorly on repairs compared to the literature, however the lack of segementation makes this a considerably harder task, in the same way as dialogue act tagging results are lower on unsegmented transcripts (Martínez-Hinarejos et al., 2015). Edit term detection performs very well at 0.918, approaching the state-of-the-art on Switchboard reported at 0.938 (Hough and Purver, 2014).

The utility of a joint task As can be seen in Table 3, the overall best performing systems on the individual tasks do not reach the results in any relevant metric of the best performing combined system. The disfluency-only systems were run ignoring all utterance boundary information, which puts this setting at a disadvantage to previous approaches, however it is clear that on unsegmented data our posing of the task jointly is useful.

Incrementality Incrementally the differences between the architectures was neglible– results for the LSTM are in Table 4. The latency for repair onset detection is very low, being detected as little as 0.196 seconds after the onset word is finished (or on transcripts largely directly after the word has been consumed as TTD_{rps} (word) = 0.003). Utterance boundaries were detected just over a second after the end of the last word of the previous utterance. However, the fact that TTD_{uttSeg} on the word level reaches 0.283 suggests the time-based average is being weighed down by occasional long silences, which could be thresholded in future work. The EO measure of stability is severely affected by jittering ASR hypotheses, but given its worst result is 21.46% this is still a fairly stable incremental system.

Error Analysis To explore the errors being made by the systems, and how the RNN and LSTM may differ in ability, we performed an error analysis on the simple versions with the timing models– see Fig. 4. One can observe a boost in recall for various repair types in the LSTM, where it is performing better on repairs with longer reparanda. Characterizing repetitions as verbatim repeats, substitutions as the other repairs marked with a repair phase, and deletes as those without one, we see the LSTM outperforming the RNN on the rarer types. Whilst the problem is attenuated by the memory facility of the LSTM, our best system still suffers the vanishing gradient problem for predicting longer repairs with reparanda over 3 words long. Also we show in $uttSeg$ detection all systems falter on long distance projections with coordinating conjunctions, which would potentially be dealt with more easily in a parsing framework, or a hierarchical deep learning framework.

We also investigated the $uttSeg$ detection errors and see that the networks are generally not confusing disfluencies with boundaries. However, our best system incorrectly labelled 3.6% of the reference $uttSeg$s as rpS (hence also affecting the precision of the rpS prediction)– upon inspection these were largely abandoned utterances, which according to the constraint C3 we posited above are not marked as disfluencies in the same way intra-utterance repairs are in the reference. Due to the original annotation instructions of (Meteer et al., 1995), these are segmented and not included in the traditional disfluency detection task. However,

(a)

Reparandum length	(support)	RNN recall %	LSTM recall %
1	(1487)	72.2	78.3
2	(477)	59.3	64.8
3	(155)	47.7	57.4
4	(73)	47.9	49.3
5	(31)	41.9	45.2
6	(15)	40.0	60.0
7	(9)	33.3	33.3
8	(4)	25.0	25.0

(b)

Repair Type	(support)	RNN recall %	LSTM recall %
repeat	(1043)	79.1	83.4
substitution	(1076)	59.2	66.4
delete	(132)	19.7	30.3

(c)

$uttSeg$ Error Type	RNN % error	LSTM % error
FN, predicted rpS	2.9	3.6
FN, predicted e	4.9	4.4
FN, predicted CC	16.3	15.0
FN, predicted subj	7.0	6.5
FN, predicted proper	1.2	1.1
FN, predicted it	1.1	1.2
FN, predicted grounding	1.0	0.8
FN, predicted other	8.8	8.6
FN all	43.1	41.1
FP, predicted $uttSeg$ for rpS	0.9	0.5
FP, predicted $uttSeg$ for e	3.4	2.7
FP, predicted $uttSeg$ for CC	5.1	3.6
FP, predicted $uttSeg$ for subj	2.0	1.6
FP, predicted $uttSeg$ for proper	0.9	0.6
FP, predicted $uttSeg$ for it	0.5	0.4
FP, predicted $uttSeg$ for grounding	0.7	0.4
FP, predicted $uttSeg$ for other	6.9	2.7
FP all	20.4	12.5

Figure 4: Error analysis: (a) recall rates for rpS onsets of repairs with different reparandum lengths and (b) types, and (c) the source of errors in $uttSeg$ detection.

intuitively these can be construed as a disfluency type, and in future we will treat them as a special type of uttSeg/disfluency hybrid.

As can be seen in Fig. 4 (c) other main sources of error are on coordinating conjunctions (CC) such as ‘and’ and ‘or’, nouns with nominative subject marking case like ‘I’ and ‘we’ (subj), other proper nouns, variants of ‘it’ and grounding utterances like ‘yeah’ and ‘okay’. $uttSeg$ detection in both systems achieved high precision but relatively low recall.

8 Conclusion

We have presented the joint task of incremental utterance segmentation and disfluency detection and show a simple deep learning system which performs it on transcripts and ASR results. As regards the research questions posed in §3.4, in answer to Q1, we showed that, all else being equal, a deep learning system can perform both tasks jointly improves over equivalent systems doing the individual tasks. In answer to Q2, we showed that word timing information, both from transcripts and ASR results, helps the utterance segmentation and the joint task across all settings whilst not aiding disfluency detection on its own, and in response to Q3, we achieve a good online accuracy vs. final accuracy trade-off in a live, incremental, system, however still experience some time delays for utterance segmentation in our most accurate system.

We conclude that our joint-task system for disfluency detection and utterance segmentation shows a new benchmark for the joint task on Switchboard data and due its incremental functioning on unsegmented data, including ASR result streams, it is suitable for live systems, such as conversation agents in the psychiatric domain. In future work we intend to optimize the inputs to our networks after this exploration, including using raw acoustic features, and combining the task with language modelling and dialogue act tagging.

Acknowledgments

We thank the EACL reviewers for their helpful comments. This work was supported by the Cluster of Excellence Cognitive Interaction Technology ‘CITEC’ (EXC 277) at Bielefeld University, funded by the German Research Foundation (DFG), and the DFG-funded DUEL project (grant SCHL 845/5-1).

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP (1)*, pages 1061–1064.

Michaela Atterer, Timo Baumann, and David Schlangen. 2008. Towards incremental end-of-utterance detection in dialogue systems. In *COLING (Posters)*, pages 11–14.

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

Timo Baumann, Casey Kennington, Julian Hough, and David Schlangen. 2016. Recognising conversational speech: What an incremental asr should do for a dialogue system and how to get there. In *International Workshop on Dialogue Systems Technology (IWSDS) 2016*. Universität Hamburg.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

Sébastien Cuendet. 2006. Model adaptation for sentence unit segmentation from speech. Technical report, IDIAP.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98.

David DeVault, Kallirroi Georgila, and Ron Artstein. 2013. Verbal indicators of psychological distress in interactive dialogue with a virtual human. In *Proceedings of SigDial 2013*, pages 193–202.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 554–559. IEEE.

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

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, Doha, Qatar, October. Association for Computational Linguistics.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Proceedings of Interspeech 2015*, pages 849–853.

Christine Howes, Matt Purver, Rose McCabe, Patrick GT Healey, and Mary Lavelle. 2012. Helping the medicine go down: Repair and adherence in patient-clinician dialogues. In *Proceedings of SemDial 2012 (SeineDial): The 16th Workshop on the Semantics and Pragmatics of Dialogue*, page 155.

Christine Howes, Julian Hough, Matthew Purver, and Rose McCabe. 2014. Helping, i mean assessing psychiatric communication: An application of incremental self-repair detection. In *Proceedings of the 18th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialWatt)*, pages 80–89, Edinburgh, September.

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

Willem J. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(4):41–104.

Gale M. Lucas, Jonathan Gratch, Aisha King, and Louis Philippe Morency. 2014. It's only a computer: Virtual humans increase willingness to disclose. *Computers in Human Behavior*, 37:94–100.

Ramesh Manuvinakurike, Maike Paetzel, Cheng Qu, David Schlangen, and David DeVault. 2016. Toward Incremental Dialogue Act Segmentation in Fast-Paced Interactive Dialogue Systems. In *Proceedings of the 17th Annual SIGdial Meeting on Discourse and Dialogue*. Forthcoming.

Carlos-D Martínez-Hinarejos, José-Miguel Benedí, and Vicent Tamarit. 2015. Unsegmented dialogue act annotation and decoding with n-gram transducers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):198–211.

Rosemarie McCabe, Patrick GT Healey, Stefan Priebe, Mary Lavelle, David Dodwell, Richard Laugharne, Amelia Snell, and Stephen Bremner. 2013. Shared understanding in psychiatrist–patient communication: Association with treatment adherence in schizophrenia. *Patient education and counseling*, 93(1):73–79.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775.

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.

Helena Moniz, Jaime Ferreira, Fernando Batista, and Isabel Trancoso. 2015. Disfluency detection across domains. In *The 6th Workshop on Disfluency in Spontaneous Speech (DiSS)*.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of NAACL-HLT*, pages 820–825.

Antoine Raux. 2008. *Flexible turn-taking for spoken dialog systems*. Ph.D. thesis, US National Science Foundation.

David Schlangen and Gabriel Skantze. 2011. A General, Abstract Model of Incremental Dialogue Processing. *Dialoge & Discourse*, 2(1):83–111.

David Schlangen. 2006. From reaction to prediction: Experiments with computational models of turn-taking. In *Proceedings of Interspeech 2006, Panel on Prosody of Dialogue Acts and Turn-Taking*.

Anders Seeker, Agnieszka Björkelund, Wolfgang Falenska, and Jonas Kuhn. 2016. How to train dependency parsers with inexact search for joint sentence boundary detection and parsing of entire documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1923–1934, Berlin. ACL.

Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tür, and Gökhan Tür. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech communication*, 32(1):127–154.

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

Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Engsiong Chng, and Haizhou Li. 2014. A deep neural network approach for sentence boundary detection in broadcast news. In *Proceedings of INTERSPEECH*, pages 2887–2891.

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*, pages 1371–1378, Stroudsburg, PA, USA. Association for Computational Linguistics.