Strongly Incremental Repair Detection

Julian Hough[1,2]
[1]Dialogue Systems Group
Faculty of Linguistics
and Literature
Bielefeld University
julian.hough@uni-bielefeld.de

Matthew Purver[2]
[2]Cognitive Science Research Group
School of Electronic Engineering
and Computer Science
Queen Mary University of London
m.purver@qmul.ac.uk

Abstract

We present STIR (STrongly Incremental Repair detection), a system that detects speech repairs and edit terms on transcripts incrementally with minimal latency. STIR uses information-theoretic measures from n-gram models as its principal decision features in a pipeline of classifiers detecting the different stages of repairs. Results on the Switchboard disfluency tagged corpus show utterance-final accuracy on a par with state-of-the-art incremental repair detection methods, but with better incremental accuracy, faster time-to-detection and less computational overhead. We evaluate its performance using incremental metrics and propose new repair processing evaluation standards.

1 Introduction

Self-repairs in spontaneous speech are annotated according to a well established three-phase structure from (Shriberg, 1994) onwards, and as described in Meteer et al. (1995)'s Switchboard corpus annotation handbook:

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

From a dialogue systems perspective, detecting repairs and assigning them the appropriate structure is vital for robust natural language understanding (NLU) in interactive systems. Downgrading the commitment of *reparandum* phases and assigning appropriate *interregnum* and *repair* phases permits computation of the user's intended meaning.

Furthermore, the recent focus on *incremental* dialogue systems (see e.g. (Rieser and Schlangen, 2011)) means that repair detection should operate without unnecessary processing overhead, and function efficiently within an incremental framework. However, such left-to-right operability on its own is not sufficient: in line with the principle of strong incremental interpretation (Milward, 1991), a repair detector should give *the best results possible as early as possible*. With one exception (Zwarts et al., 2010), there has been no focus on evaluating or improving the *incremental performance* of repair detection.

In this paper we present STIR (Strongly Incremental Repair detection), a system which addresses the challenges of incremental accuracy, computational complexity and latency in self-repair detection, by making local decisions based on relatively simple measures of fluency and similarity. Section 2 reviews state-of-the-art methods; Section 3 summarizes the challenges and explains our general approach; Section 4 explains STIR in detail; Section 5 explains our experimental set-up and novel evaluation metrics; Section 6 presents and discusses our results and Section 7 concludes.

2 Previous work

Qian and Liu (2013) achieve the state of the art in Switchboard corpus self-repair detection, with an F-score for detecting reparandum words of 0.841 using a three-step weighted Max-Margin Markov network approach. Similarly, Georgila (2009) uses Integer Linear Programming post-processing of a CRF to achieve F-scores over 0.8 for reparandum start and repair start detection. However neither approach can operate incrementally.

Recently, there has been increased interest in left-to-right repair detection: Rasooli and Tetreault (2014) and Honnibal and Johnson (2014) present dependency parsing systems with reparandum detection which perform similarly, the latter equalling Qian and Liu (2013)'s F-score at 0.841. However, while operating left-to-right, these systems are not designed or evaluated for their *incremental* performance. The use of beam search over

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 78–89,
October 25-29, 2014, Doha, Qatar.

different repair hypotheses in (Honnibal and Johnson, 2014) is likely to lead to unstable repair label sequences, and they report repair hypothesis 'jitter'. Both of these systems use a non-monotonic dependency parsing approach that immediately removes the reparandum from the linguistic analysis of the utterance in terms of its dependency structure and repair-reparandum correspondence, which from a downstream NLU module's perspective is undesirable. Heeman and Allen (1999) and Miller and Schuler (2008) present earlier left-to-right operational detectors which are less accurate and again give no indication of the incremental performance of their systems. While Heeman and Allen (1999) rely on repair structure template detection coupled with a multi-knowledge-source language model, the rarity of the tail of repair structures is likely to be the reason for lower performance: Hough and Purver (2013) show that only 39% of repair alignment structures appear at least twice in Switchboard, supported by the 29% reported by Heeman and Allen (1999) on the smaller TRAINS corpus. Miller and Schuler (2008)'s encoding of repairs into a grammar also causes sparsity in training: repair is a general processing strategy not restricted to certain lexical items or POS tag sequences.

The model we consider most suitable for incremental dialogue systems so far is Zwarts et al. (2010)'s incremental version of Johnson and Charniak (2004)'s noisy channel repair detector, as it incrementally applies structural repair analyses (rather than just identifying reparanda) and is evaluated for its incremental properties. Following (Johnson and Charniak, 2004), their system uses an n-gram language model trained on roughly 100K utterances of reparandum-excised ('cleaned') Switchboard data. Its channel model is a statistically-trained S-TAG parser whose grammar has simple reparandum-repair alignment rule categories for its non-terminals (copy, delete, insert, substitute) and words for its terminals. The parser hypothesises all possible repair structures for the string consumed so far in a chart, before pruning the unlikely ones. It performs equally well to the non-incremental model by the end of each utterance (F-score = 0.778), and can make detections early via the addition of a speculative next-word repair completion category to their S-TAG non-terminals. In terms of incremental performance, they report the novel evaluation metric of *time-to-detection* for correctly identified repairs, achieving an average of 7.5 words from the start of the reparandum and 4.6 from the start of the repair phase. They also introduce *delayed accuracy*, a word-by-word evaluation against gold-standard disfluency tags up to the word before the current word being consumed (in their terms, the *prefix boundary*), giving a measure of the stability of the repair hypotheses. They report an F-score of 0.578 at one word back from the current prefix boundary, increasing word-by-word until 6 words back where it reaches 0.770. These results are the point-of-departure for our work.

3 Challenges and Approach

In this section we summarize the challenges for incremental repair detection: computational complexity, repair hypothesis stability, latency of detection and repair structure identification. In 3.1 we explain how we address these.

Computational complexity Approaches to detecting repair structures often use chart storage (Zwarts et al., 2010; Johnson and Charniak, 2004; Heeman and Allen, 1999), which poses a computational overhead: if considering all possible boundary points for a repair structure's 3 phases beginning on any word, for prefixes of length n the number of hypotheses can grow in the order $O(n^4)$. Exploring a subset of this space is necessary for assigning entire repair structures as in (1) above, rather than just detecting reparanda: the (Johnson and Charniak, 2004; Zwarts et al., 2010) noisy-channel detector is the only system that applies such structures but the potential runtime complexity in decoding these with their S-TAG repair parser is $O(n^5)$. In their approach, complexity is mitigated by imposing a maximum repair length (12 words), and also by using beam search with re-ranking (Lease et al., 2006; Zwarts and Johnson, 2011). If we wish to include full decoding of the repair's structure (as argued by Hough and Purver (2013) as necessary for full interpretation) whilst taking a strictly incremental and time-critical perspective, reducing this complexity by minimizing the size of this search space is crucial.

Stability of repair hypotheses and latency Using a beam search of n-best hypotheses on a word-by-word basis can cause 'jitter' in the detector's output. While utterance-final accuracy is desired,

for a truly incremental system good intermediate results are equally important. Zwarts et al. (2010)'s time-to-detection results show their system is only certain about a detection after processing the entire repair. This may be due to the string alignment-inspired S-TAG that matches repair and reparanda: a 'rough copy' dependency only becomes likely once the entire repair has been consumed. The latency of 4.6 words to detection and a relatively slow rise to utterance-final accuracy up to 6 words back is undesirable given repairs have a mean reparandum length of ≈1.5 words (Hough and Purver, 2013; Shriberg and Stolcke, 1998).

Structural identification Classifying repairs has been ignored in repair processing, despite the presence of distinct categories (e.g. repeats, substitutions, deletes) with different pragmatic effects (Hough and Purver, 2013).[1] This is perhaps due to lack of clarity in definition: even for human annotators, verbatim repeats withstanding, agreement is often poor (Hough and Purver, 2013; Shriberg, 1994). Assigning and evaluating repair (not just reparandum) structures will allow repair interpretation in future; however, work to date evaluates only reparandum detection.

3.1 Our approach

To address the above, we propose an alternative to (Johnson and Charniak, 2004; Zwarts et al., 2010)'s noisy channel model. While the model elegantly captures intuitions about parallelism in repairs and modelling fluency, it relies on string-matching, motivated in a similar way to automatic spelling correction (Brill and Moore, 2000): it assumes a speaker chooses to utter fluent utterance X according to some prior distribution $P(X)$, but a noisy channel causes them instead to utter a noisy Y according to channel model $P(Y \mid X)$. Estimating $P(Y \mid X)$ directly from observed data is difficult due to sparsity of repair instances, so a transducer is trained on the rough copy alignments between reparandum and repair. This approach succeeds because repetition and simple substitution repairs are very common; but repair as a psychological process is not driven by string alignment, and deletes, restarts and rarer substitution forms are not captured. Furthermore, the noisy channel model assumes an inherently utterance-global process for generating (and therefore finding) an underlying 'clean' string — much as similar spelling correction models are word-global — we instead take a very local perspective here.

In accordance with psycholinguistic evidence (Brennan and Schober, 2001), we assume characteristics of the repair onset allow hearers to detect it very quickly and solve the *continuation problem* (Levelt, 1983) of integrating the repair into their linguistic context immediately, before processing or even hearing the end of the repair phase. While repair onsets may take the form of interregna, this is not a reliable signal, occurring in only ≈15% of repairs (Hough and Purver, 2013; Heeman and Allen, 1999). Our repair onset detection is therefore driven by departures from fluency, via information-theoretic features derived incrementally from a language model in line with recent psycholinguistic accounts of incremental parsing – see (Keller, 2004; Jaeger and Tily, 2011).

Considering the time-linear way a repair is processed and the fact speakers are exponentially less likely to trace one word further back in repair as utterance length increases (Shriberg and Stolcke, 1998), backwards search seems to be the most efficient reparandum extent detection method.[2] Features determining the detection of the reparandum extent in the backwards search can also be information-theoretic: entropy measures of distributional parallelism can characterize not only rough copy dependencies, but distributionally similar or dissimilar correspondences between sequences. Finally, when detecting the repair end and structure, distributional information allows computation of the similarity between reparandum and repair. We argue a local-detection-with-backtracking approach is more cognitively plausible than string-based left-to-right repair labelling, and using this insight should allow an improvement in incremental accuracy, stability and time-to-detection over string-alignment driven approaches in repair detection.

4 STIR: Strongly Incremental Repair detection

Our system, STIR (Strongly Incremental Repair detection), therefore takes a local incremental ap-

[1] Though see (Germesin et al., 2008) for one approach, albeit using idiosyncratic repair categories.

[2] We acknowledge a purely position-based model for reparandum extent detection under-estimates prepositions, which speakers favour as the retrace start and over-estimates verbs, which speakers tend to avoid retracing back to, preferring to begin the utterance again, as (Healey et al., 2011)'s experiments also demonstrate.

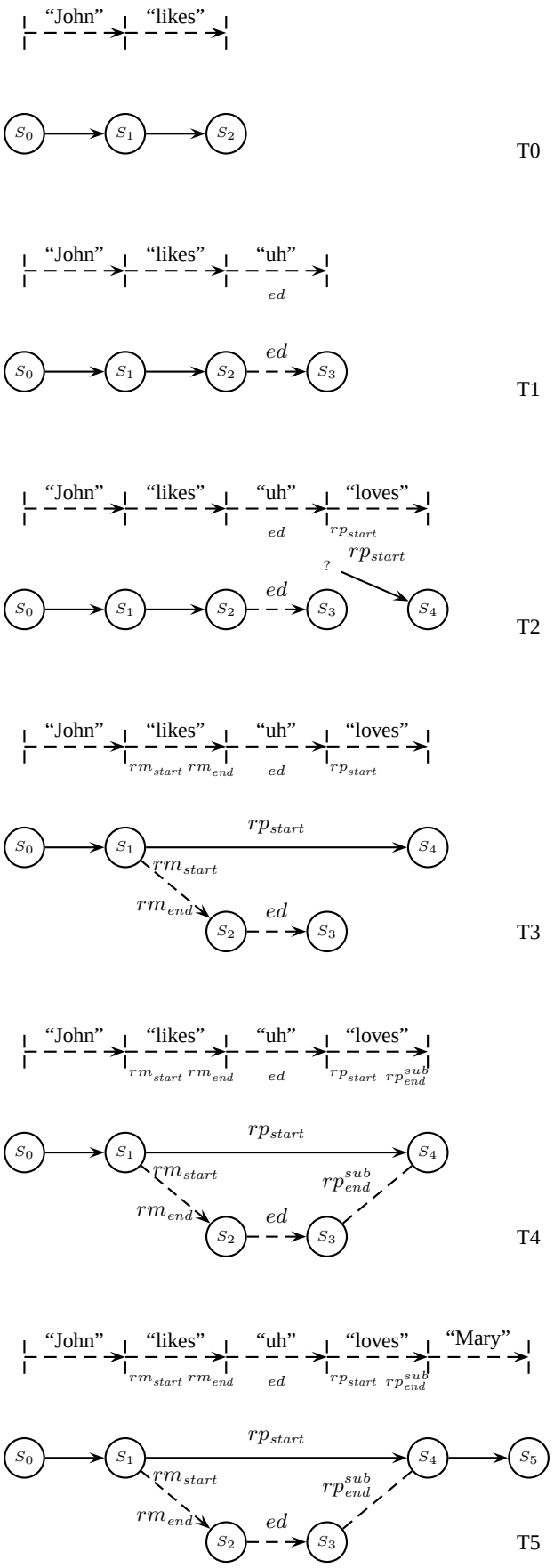


Figure 1: Strongly Incremental Repair Detection

proach to detecting repairs and isolated edit terms, assigning words the structures in (2). We include interregnum recognition in the process, due to the inclusion of interregnum vocabulary within edit term vocabulary (Ginzburg, 2012; Hough and Purver, 2013), a useful feature for repair detection (Lease et al., 2006; Qian and Liu, 2013).

$$\begin{cases} ...[rm_{start}...rm_{end} + \{ed\}rp_{start}...rp_{end}]... \\ ...\{ed\}... \end{cases} \quad (2)$$

Rather than detecting the repair structure in its left-to-right string order as above, STIR functions as in Figure 1: first detecting edit terms (possibly interregna) at step T1; then detecting repair onsets rp_{start} at T2; if one is found, backwards searching to find rm_{start} at T3; then finally finding the repair end rp_{end} at T4. Step T1 relies mainly on lexical probabilities from an edit term language model; T2 exploits features of divergence from a fluent language model; T3 uses fluency of hypothesised repairs; and T4 the similarity between distributions after reparandum and repair. However, each stage integrates these basic insights via multiple related features in a statistical classifier.

4.1 Enriched incremental language models

We derive the basic information-theoretic features required using n-gram language models, as they have a long history of information theoretic analysis (Shannon, 1948) and provide reproducible results without forcing commitment to one particular grammar formalism. Following recent work on modelling grammaticality judgements (Clark et al., 2013), we implement several modifications to standard language models to develop our basic measures of fluency and uncertainty.

For our main fluent language models we train a trigram model with Kneser-Ney smoothing (Kneser and Ney, 1995) on the words and POS tags of the standard Switchboard training data (all files with conversation numbers beginning sw2*,sw3* in the Penn Treebank III release), consisting of ≈100K utterances, ≈600K words. We follow (Johnson and Charniak, 2004) by cleaning the data of disfluencies (i.e. edit terms and reparanda), to approximate a 'fluent' language model. We call these probabilities p^{lex}_{kn}, p^{pos}_{kn} below.[3]

[3]We suppress the ^{pos} and ^{lex} superscripts below where we refer to measures from either model.

We then derive *surprisal* as our principal default lexical uncertainty measurement s (equation 3) in both models; and, following (Clark et al., 2013), the (unigram) Weighted Mean Log trigram probability (WML, eq. 4)– the trigram logprob of the sequence divided by the inverse summed logprob of the component unigrams (apart from the first two words in the sequence, which serve as the first trigram history). As here we use a local approach we restrict the WML measures to single trigrams (weighted by the inverse logprob of the final word). While use of standard n-gram probability conflates syntactic with lexical probability, WML gives us an approximation to *incremental syntactic probability* by factoring out lexical frequency.

$$s(w_{i-2} \dots w_i) = -\log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1}) \quad (3)$$

$$WML(w_0 \dots w_n) = \frac{\sum_{i=2}^{i=n} \log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1})}{-\sum_{j=2}^{n} \log_2 p_{kn}(w_j)} \quad (4)$$

Distributional measures To approximate uncertainty, we also derive the entropy $H(w \,|\, c)$ of the possible word continuations w given a context c, from $p(w_i \,|\, c)$ for all words w_i in the vocabulary – see (5). Calculating distributions over the entire lexicon incrementally is costly, so we approximate this by constraining the calculation to words which are observed at least once in context c in training, $w_c = \{w | count(c, w) \geq 1\}$, assuming a uniform distribution over the unseen suffixes by using the appropriate smoothing constant, and subtracting the latter from the former – see eq. (6).

Manual inspection showed this approximation to be very close, and the trie structure of our n-gram models allows efficient calculation. We also make use of the Zipfian distribution of n-grams in corpora by storing entropy values for the 20% most common trigram contexts observed in training, leaving entropy values of rare or unseen contexts to be computed at decoding time with little search cost due to their small or empty w_c sets.

$$H(w \mid c) = -\sum_{w \in Vocab} p_{kn}(w \mid c) \log_2 p_{kn}(w \mid c) \quad (5)$$

$$\begin{aligned} H(w \mid c) \approx & \left[-\sum_{w \in w_c} p_{kn}(w \mid c) \log_2 p_{kn}(w \mid c) \right] \\ & - [n \times \lambda \log_2 \lambda] \\ & \text{where } n = |Vocab| - |w_c| \\ & \text{and } \lambda = \frac{1 - \sum_{w \in w_c} p_{kn}(w \mid c)}{n} \end{aligned} \quad (6)$$

Given entropy estimates, we can also similarly approximate the Kullback-Leibler (KL) divergence (relative entropy) between distributions in two different contexts c_1 and c_2, i.e. $\theta(w|c_1)$ and $\theta(w|c_2)$, by pair-wise computing $p(w|c_1) \log_2(\frac{p(w|c_1)}{p(w|c_2)})$ only for words $w \in w_{c_1} \cap w_{c_2}$, then approximating unseen values by assuming uniform distributions. Using p_{kn} smoothed estimates rather than raw maximum likelihood estimations avoids infinite KL divergence values. Again, we found this approximation sufficiently close to the real values for our purposes. All such probability and distribution values are stored in incrementally constructed directed acyclic graph (DAG) structures (see Figure 1), exploiting the Markov assumption of n-gram models to allow efficient calculation by avoiding re-computation.

4.2 Individual classifiers

This section details the features used by the 4 individual classifiers. To investigate the utility of the features used in each classifier we obtain values on the standard Switchboard heldout data (PTB III files sw4[5-9]*: 6.4K utterances, 49K words).

4.2.1 Edit term detection

In the first component, we utilise the well-known observation that edit terms have a distinctive vocabulary (Ginzburg, 2012), training a bigram model on a corpus of all edit words annotated in Switchboard's training data. The classifier simply uses the surprisal s^{lex} from this edit word model, and the trigram surprisal s^{lex} from the standard fluent model of Section 4.1. At the current position w_n, one, both or none of words w_n and w_{n-1} are classified as edits. We found this simple approach effective and stable, although some delayed decisions occur in cases where s^{lex} and WML^{lex} are high in both models before the end of the edit, e.g. "I like" $\rightarrow$ "I $\{like\}$ want...". Words classified as ed are removed from the incremental processing graph (indicated by the dotted line transition in Figure 1) and the stack updated if repair hypotheses are cancelled due to a delayed edit hypothesis of w_{n-1}.

4.2.2 Repair start detection

Repair onset detection is arguably the most crucial component: the greater its accuracy, the better the input for downstream components and the lesser the overhead of filtering false positives required.

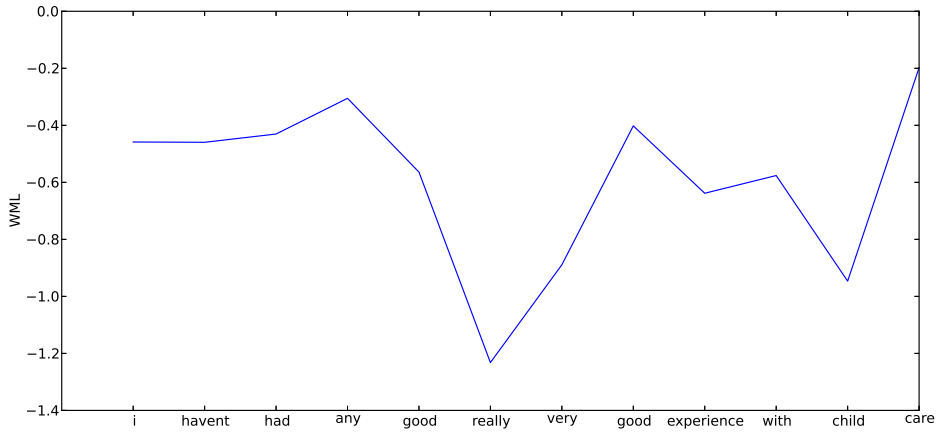


Figure 2: WML^{lex} values for trigrams for a repaired utterance exhibiting the drop at the repair onset

We use Section 4.1's information-theoretic features s, WML, H for words and POS, and introduce 5 additional information-theoretic features: ΔWML is the difference between the WML values at w_{n-1} and w_n; ΔH is the difference in entropy between w_{n-1} and w_n; $InformationGain$ is the difference between expected entropy at w_{n-1} and observed s at w_n, a measure that factors out the effect of naturally high entropy contexts; $BestEntropyReduce$ is the best reduction in entropy possible by an early rough hypothesis of reparandum onsets within 3 words; and $BestWMLBoost$ similarly speculates on the best improvement of WML possible by positing rm_{start} positions up to 3 words back. We also include simple alignment features: binary features which indicate if the word w_{i-x} is identical to the current word w_i for $x \in \{1, 2, 3\}$. With 6 alignment features, 16 N-gram features and a single logical feature $edit$ which indicates the presence of an edit word at position w_{i-1}, rp_{start} detection uses 23 features– see Table 1.

We hypothesised repair onsets rp^{start} would have significantly lower p^{lex} (lower lexical-syntactic probability) and WML^{lex} (lower syntactic probability) than other fluent trigrams. This was the case in the Switchboard heldout data for both measures, with the biggest difference obtained for WML^{lex} (non-repair-onsets: -0.736 (sd=0.359); repair onsets: -1.457 (sd=0.359)). In the POS model, entropy of continuation H^{pos} was the strongest feature (non-repair-onsets: 3.141 (sd=0.769); repair onsets: 3.444 (sd=0.899)). The trigram WML^{lex} measure for the repaired utterance "I haven't had any [good + really very good] experience with child care" can be seen in Figure 2. The steep drop at the repair onset shows the usefulness of WML features for fluency measures.

To compare n-gram measures against other local features, we ranked the features by Information Gain using 10-fold cross validation over the Switchboard heldout data– see Table 1. The language model features are far more discriminative than the alignment features, showing the potential of a general information-theoretic approach.

4.2.3 Reparandum start detection

In detecting rm_{start} positions given a hypothesised rp_{start} (stage T3 in Figure 1), we use the noisy channel intuition that removing the reparandum (from rm_{start} to rp_{start}) increases fluency of the utterance, expressed here as $WMLboost$ as described above. When using gold standard input we found this was the case on the heldout data, with a mean $WMLboost$ of 0.223 (sd=0.267) for reparandum onsets and -0.058 (sd=0.224) for other words in the 6-word history- the negative boost for non-reparandum words captures the intuition that backtracking from those points would make the utterance less grammatical, and conversely the boost afforded by the correct rm_{start} detection helps solve the continuation problem for the listener (and our detector).

Parallelism in the onsets of rp_{start} and rm_{start} can also help solve the continuation problem, and in fact the KL divergence between $\theta^{pos}(w \mid rm_{start}, rm_{start-1})$ and $\theta^{pos}(w \mid rp_{start}, rp_{start-1})$ is the second most useful feature with average merit 0.429 (+- 0.010) in cross-

validation. The highest ranked feature is ΔWML (0.437 (+- 0.003)) which here encodes the drop in the $WMLboost$ from one backtracked position to the next. In ranking the 32 features we use, again information-theoretic ones are higher ranked than the logical features.

average merit	average rank	attribute
0.139 (+- 0.002)	1 (+- 0.00)	H^{pos}
0.131 (+- 0.001)	2 (+- 0.00)	WML^{pos}
0.126 (+- 0.001)	3.4 (+- 0.66)	WML^{lex}
0.125 (+- 0.003)	4 (+- 1.10)	s^{pos}
0.122 (+- 0.001)	5.9 (+- 0.94)	$w_{i-1} = w_i$
0.122 (+- 0.001)	5.9 (+- 0.70)	BestWMLBoost^{lex}
0.122 (+- 0.002)	5.9 (+- 1.22)	InformationGain^{pos}
0.119 (+- 0.001)	7.9 (+- 0.30)	BestWMLBoost^{pos}
0.098 (+- 0.002)	9 (+- 0.00)	H^{lex}
0.08 (+- 0.001)	10.4 (+- 0.49)	ΔWML^{pos}
0.08 (+- 0.003)	10.6 (+- 0.49)	ΔH^{pos}
0.072 (+- 0.001)	12 (+- 0.00)	$POS_{i-1} = POS_i$
0.066 (+- 0.003)	13.1 (+- 0.30)	s^{lex}
0.059 (+- 0.000)	14.2 (+- 0.40)	ΔWML^{lex}
0.058 (+- 0.005)	14.7 (+- 0.64)	BestEntropyReduce^{pos}
0.049 (+- 0.001)	16.3 (+- 0.46)	InformationGain^{lex}
0.047 (+- 0.004)	16.7 (+- 0.46)	BestEntropyReduce^{lex}
0.035 (+- 0.004)	18 (+- 0.00)	ΔH^{lex}
0.024 (+- 0.000)	19 (+- 0.00)	$w_{i-2} = w_i$
0.013 (+- 0.000)	20 (+- 0.00)	$POS_{i-2} = POS_i$
0.01 (+- 0.000)	21 (+- 0.00)	$w_{i-3} = w_i$
0.009 (+- 0.000)	22 (+- 0.00)	edit
0.006 (+- 0.000)	23 (+- 0.00)	$POS_{i-3} = POS_i$

Table 1: Feature ranker (Information Gain) for rp_{start} detection- 10-fold x-validation on Switchboard heldout data.

4.2.4 Repair end detection and structure classification

For rp_{end} detection, using the notion of parallelism, we hypothesise an effect of divergence between θ^{lex} at the reparandum-final word rm_{end} and the repair-final word rp_{end}: for repetition repairs, KL divergence will trivially be 0; for substitutions, it will be higher; for deletes, even higher. Upon inspection of our feature ranking this KL measure ranked 5th out of 23 features (merit= 0.258 (+- 0.002)).

We introduce another feature encoding parallelism $ReparandumRepairDifference$: the difference in probability between an utterance cleaned of the reparandum and the utterance with its repair phase substituting its reparandum. In both the POS (merit=0.366 (+- 0.003)) and word (merit=0.352 (+- 0.002)) LMs, this was the most discriminative feature.

4.3 Classifier pipeline

STIR effects a pipeline of classifiers as in Figure 3, where the ed classifier only permits non ed words to be passed on to rp_{start} classification and for rp_{end} classification of the active repair hypotheses, maintained in a stack. The rp_{start} classifier passes positive repair hypotheses to the rm_{start} classifier, which backwards searches up to 7 words back in the utterance. If a rm_{start} is classified, the output is passed on for rp_{end} classification at the end of the pipeline, and if not rejected this is pushed onto the repair stack. Repair hypotheses are are popped off when the string is 7 words beyond its rp_{start} position. Putting limits on the stack's storage space is a way of controlling for processing overhead and complexity. Embedded repairs whose rm_{start} coincide with another's rp_{start} are easily dealt with as they are added to the stack as separate hypotheses.[4]

Classifiers Classifiers are implemented using Random Forests (Breiman, 2001) and we use different error functions for each stage using MetaCost (Domingos, 1999). The flexibility afforded by implementing adjustable error functions in a pipelined incremental processor allows control of the trade-off of immediate accuracy against runtime and stability of the sequence classification.

Processing complexity This pipeline avoids an exhaustive search all repair hypotheses. If we limit the search to within the $\langle rm_{start}, rp_{start} \rangle$ possibilities, this number of repairs grows approximately in the triangular number series– i.e. $\frac{n(n+1)}{2}$, a nested loop over previous words as n gets incremented – which in terms of a complexity class is a quadratic $O(n^2)$. If we allow more than one $\langle rm_{start}, rp_{start} \rangle$ hypothesis per word, the complexity goes up to $O(n^3)$, however in the tests that we describe below, we are able to achieve good detection results without permitting this extra search space. Under our assumption that reparandum onset detection is only triggered after repair onset detection, and repair extent detection is dependent on positive reparandum onset detection, a pipeline with accurate components will allow us to limit processing to a small subset of this search space.

[4]We constrain the problem not to include embedded deletes which may share their rp_{start} word with another repair – these are in practice very rare.

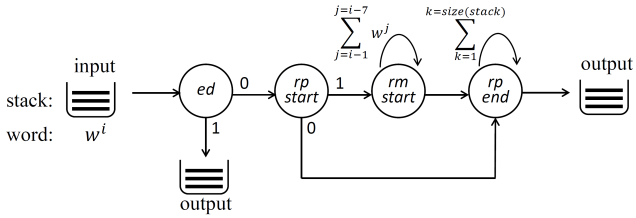


Figure 3: Classifier pipeline

5 Experimental set-up

We train STIR on the Switchboard data described above, and test it on the standard Switchboard test data (PTB III files 4[0-1]*). In order to avoid overfitting of classifiers to the basic language models, we use a cross-fold training approach: we divide the corpus into 10 folds and use language models trained on 9 folds to obtain feature values for the 10th fold, repeating for all 10. Classifiers are then trained as standard on the resulting feature-annotated corpus. This resulted in better feature utility for n-grams and better F-score results for detection in all components in the order of 5-6%.[5]

Training the classifiers Each Random Forest classifier was limited to 20 trees of maximum depth 4 nodes, putting a ceiling on decoding time. In making the classifiers cost-sensitive, MetaCost resamples the data in accordance with the cost functions: we found using 10 iterations over a resample of 25% of the training data gave the most effective trade-off between training time and accuracy.[6] We use 8 different cost functions in rp_{start} with differing costs for false negatives and positives of the form below, where R is a repair element word and F is a fluent onset:

$$\begin{array}{c} \\ R^{gold} \\ F^{gold} \end{array} \begin{array}{c} \begin{array}{cc} R^{hyp} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \end{array}$$

We adopt a similar technique in rm_{start} using 5 different cost functions and in rp_{end} using 8 different settings, which when combined gives a total of 320 different cost function configurations. We hypothesise that higher recall permitted in the pipeline's first components would result in better overall accuracy as these hypotheses become refined, though at the cost of the stability of the hypotheses of the sequence and extra downstream processing in pruning false positives.

We also experiment with the number of repair hypotheses permitted per word, using limits of 1-best and 2-best hypotheses. We expect that allowing 2 hypotheses to be explored per rp_{start} should allow greater final accuracy, but with the trade-off of greater decoding and training complexity, and possible incremental instability.

As we wish to explore the incrementality versus final accuracy trade-off that STIR can achieve we now describe the evaluation metrics we employ.

5.1 Incremental evaluation metrics

Following (Baumann et al., 2011) we divide our evaluation metrics into *similarity metrics* (measures of equality with or similarity to a gold standard), *timing metrics* (measures of the timing of relevant phenomena detected from the gold standard) and *diachronic metrics* (evolution of incremental hypotheses over time).

Similarity metrics For direct comparison to previous approaches we use the standard measure of overall accuracy, the F-score over reparandum words, which we abbreviate $\mathbf{F_{rm}}$ (see 7):

$$\begin{aligned} \text{precision} &= \frac{rm^{correct}}{rm^{hyp}} \\ \text{recall} &= \frac{rm^{correct}}{rm^{gold}} \\ \text{F}_{rm} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{7}$$

We are also interested in repair structural classification, we also measure F-score over *all* repair components (rm words, ed words as interregna and rp words), a metric we abbreviate $\mathbf{F_s}$. This is not measured in standard repair detection on Switchboard. To investigate incremental accuracy we evaluate the *delayed accuracy* (**DA**) introduced by (Zwarts et al., 2010), as described in section 2 against the utterance-final gold standard disfluency annotations, and use the mean of the 6 word F-scores.

[5] Zwarts and Johnson (2011) take a similar approach on Switchboard data to train a re-ranker of repair analyses.

[6] As (Domingos, 1999) demonstrated, there are only relatively small accuracy gains when using more than this, with training time increasing in the order of the re-sample size.

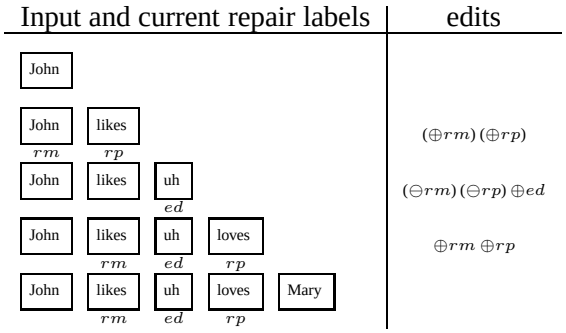
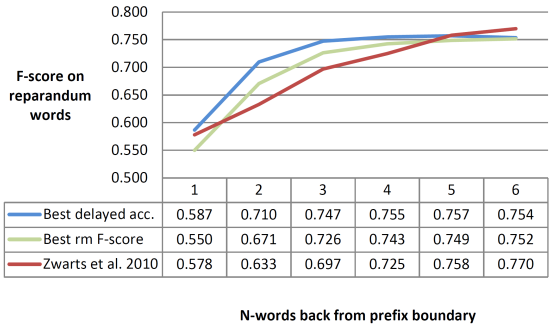


Figure 4: Edit Overhead- 4 unnecessary edits



	1	2	3	4	5	6
Best delayed acc.	0.587	0.710	0.747	0.755	0.757	0.754
Best rm F-score	0.550	0.671	0.726	0.743	0.749	0.752
Zwarts et al. 2010	0.578	0.633	0.697	0.725	0.758	0.770

Figure 6: Delayed Accuracy Curves

Timing and resource metrics Again for comparative purposes we use Zwarts et al's *time-to-detection* metrics, that is the two average distances (in numbers of words) consumed before first detection of gold standard repairs, one from rm_{start}, $\mathbf{TD_{rm}}$ and one from rp_{start}, $\mathbf{TD_{rp}}$. In our 1-best detection system, before evaluation we know a priori TD_{rp} will be 1 token, and TD_{rm} will be 1 more than the average length of $rm_{start} - rp_{start}$ repair spans correctly detected. However when we introduce a beam where multiple rm_{start}s are possible per rp_{start} with the most likely hypothesis committed as the current output, the latency may begin to increase: the initially most probable hypothesis may not be the correct one. In addition to output timing metrics, we account for intrinsic processing complexity with the metric *processing overhead* (**PO**), which is the number of classifications made by all components per word of input.

Diachronic metrics To measure stability of repair hypotheses over time we use (Baumann et al., 2011)'s *edit overhead* (**EO**) metric. EO measures the proportion of edits (add, revoke, substitute) applied to a processor's output structure that are unnecessary. STIR's output is the repair label sequence shown in Figure 1, however rather than evaluating its EO against the current gold standard labels, we use a new mark-up we term the *incremental repair gold standard*: this does not penalise lack of detection of a reparandum word rm as a bad edit until the corresponding rp_{start} of that rm has been consumed. While F_{rm}, F_s and DA evaluate against what Baumann et al. (2011) call the *current gold standard*, the incremental gold standard reflects the repair processing approach we set out in 3. An example of a repaired utterance with an EO of 44% ($\frac{4}{9}$) can be seen in Figure 4: of the 9 edits (7 repair annotations and 2 correct fluent words), 4 are unnecessary (bracketed). Note the final $\oplus rm$ is not counted as a bad edit for the reasons just given.

6 Results and Discussion

We evaluate on the Switchboard test data; Table 2 shows results of the best performing settings for each of the metrics described above, together with the setting achieving the highest total score (**TS**)– the average % achieved of the best performing system's result in each metric.[7] The settings found to achieve the highest $\text{F}_{\mathbf{rm}}$ (the metric standardly used in disfluency detection), and that found to achieve the highest TS for each stage in the pipeline are shown in Figure 5.

Our experiments showed that different system settings perform better in different metrics, and no individual setting achieved the best result in all of them. Our best utterance-final F_{rm} reaches 0.779, marginally though not significantly exceeding (Zwarts et al., 2010)'s measure and STIR achieves 0.736 on the previously unevaluated F_s. The setting with the best DA improves on (Zwarts et al., 2010)'s result significantly in terms of mean values (0.718 vs. 0.694), and also in terms of the steepness of the curves (Figure 6). The fastest average time to detection is 1 word for TD_{rp} and 2.6 words for TD_{rm} (Table 3), improving dramatically on the noisy channel model's 4.6 and 7.5 words.

Incrementality versus accuracy trade-off We aimed to investigate how well a system could do in terms of achieving both good final accuracy and incremental performance, and while the best F_{rm} setting had a large PO and relatively slow DA increase, we find STIR can find a good trade-off set-

[7] We do not include time-to-detection scores in TS as it did not vary enough between settings to be significant, however there was a difference in this measure between the 1-best stack condition and the 2-best stack condition – see below.

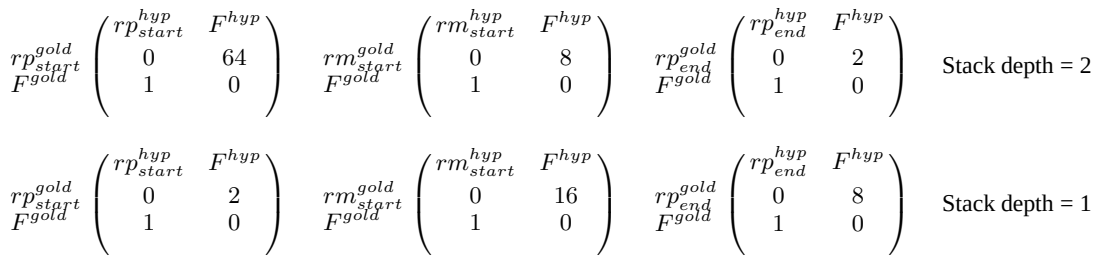


Figure 5: The cost function settings for the MetaCost classifiers for each component, for the best F_{rm} setting (top row) and best total score (TS) setting (bottom row)

	F_{rm}	F_s	DA	EO	PO
Best Final rm F-score (F_{rm})	**0.779**	0.735	0.698	3.946	1.733
Best Final repair structure F-score (F_s)	0.772	**0.736**	0.707	4.477	1.659
Best Delayed Accuracy of rm (DA)	0.767	0.721	**0.718**	1.483	1.689
Best (lowest) Edit Overhead (EO)	0.718	0.674	0.675	**0.864**	1.230
Best (lowest) Processing Overhead (PO)	0.716	0.671	0.673	0.875	**1.229**
Best Total Score (mean % of best scores) (TS)	*0.754*	*0.708*	*0.711*	*0.931*	*1.255*

Table 2: Comparison of the best performing system settings using different measures

	F_{rm}	F_s	DA	EO	PO	TD_{rp}	TD_{rm}
1-best rm_{start}	0.745	0.707	0.699	3.780	1.650	1.0	2.6
2-best rm_{start}	0.758	0.721	0.701	4.319	1.665	1.1	2.7

Table 3: Comparison of performance of systems with different stack capacities

ting: the highest TS scoring setting achieves an F_{rm} of 0.754 whilst also exhibiting a very good DA (0.711) – over 98% of the best recorded score – and low PO and EO rates – over 96% of the best recorded scores. See the bottom row of Table 2. As can be seen in Figure 5, the cost functions for these winning settings are different in nature. The best non-incremental F_{rm} measure setting requires high recall for the rest of the pipeline to work on, using the highest cost, 64, for false negative rp_{start} words and the highest stack depth of 2 (similar to a wider beam); but the best overall TS scoring system uses a less permissive setting to increase incremental performance.

We make a preliminary investigation into the effect of increasing the stack capacity by comparing stacks with 1-best rm_{start} hypotheses per rp_{start} and 2-best stacks. The average differences between the two conditions is shown in Table 3. Moving to the 2-stack condition results in gain in overall accuracy in F_{rm} and F_s, but at the cost of EO and also time-to-detection scores TD_{rm} and TD_{rp}. The extent to which the stack can be increased without increasing jitter, latency and complexity will be investigated in future work.

7 Conclusion

We have presented STIR, an incremental repair detector that can be used to experiment with incremental performance and accuracy trade-offs. In future work we plan to include probabilistic and distributional features from a top-down incremental parser e.g. Roark et al. (2009), and use STIR's distributional features to classify repair type.

Acknowledgements

We thank the three anonymous EMNLP reviewers for their helpful comments. Hough is supported by the DUEL project, financially supported by the Agence Nationale de la Research (grant number ANR-13-FRAL-0001) and the Deutsche Forschungsgemainschaft. Much of the work was carried out with support from an EPSRC DTA scholarship at Queen Mary University of London. Purver is partly supported by ConCreTe: the project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

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

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

S.E. Brennan and M.F. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech. *Journal of Memory and Language*, 44(2):274–296.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Alexander Clark, Gianluca Giorgolo, and Shalom Lappin. 2013. Statistical representation of grammaticality judgements: the limits of n-gram models. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 28–36, Sofia, Bulgaria, August. Association for Computational Linguistics.

Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM.

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.

Sebastian Germesin, Tilman Becker, and Peter Poller. 2008. Hybrid multi-step disfluency detection. In *Machine Learning for Multimodal Interaction*, pages 185–195. Springer.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

P. G. T. Healey, Arash Eshghi, Christine Howes, and Matthew Purver. 2011. Making a contribution: Processing clarification requests in dialogue. In *Proceedings of the 21st Annual Meeting of the Society for Text and Discourse*, Poitiers, July.

Peter Heeman and James Allen. 1999. Speech repairs, intonational phrases, and discourse markers: modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–571.

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. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*, pages 92–101, Amsterdam, December.

T Florian Jaeger and Harry Tily. 2011. On language utility: Processing complexity and communicative efficiency. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):323–335.

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*, pages 33–39, Barcelona. Association for Computational Linguistics.

Frank Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In *EMNLP*, pages 317–324.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Matthew Lease, Mark Johnson, and Eugene Charniak. 2006. Recognizing disfluencies in conversational speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1566–1573.

W.J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

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.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 569–576. Association for Computational Linguistics.

David Milward. 1991. *Axiomatic Grammar, Non-Constituent Coordination and Incremental Interpretation*. Ph.D. thesis, University of Cambridge.

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

Mohammad Sadegh Rasooli and Joel Tetreault. 2014. Non-monotonic parsing of fluent umm I mean disfluent sentences. *EACL 2014*, pages 48–53.

Hannes Rieser and David Schlangen. 2011. Introduction to the special issue on incremental processing in dialogue. *Dialogue & Discourse*, 2(1):1–10.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 324–333. Association for Computational Linguistics.

Claude E. Shannon. 1948. A mathematical theory of communication. technical journal. *AT & T Bell Labs*.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

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

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 703–711, Stroudsburg, PA, USA. Association for Computational Linguistics.

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