The Road to Quality is Paved with Good Revisions: A Detailed Evaluation Methodology for Revision Policies in Incremental Sequence Labelling

Brielen Madureira[1] **Patrick Kahardipraja**[1] **David Schlangen**[1,2]
[1]Computational Linguistics, Department of Linguistics, University of Potsdam, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
`{madureiralasota,kahardipraja,david.schlangen}@uni-potsdam.de`

Abstract

Incremental dialogue model components produce a sequence of output prefixes based on incoming input. Mistakes can occur due to local ambiguities or to wrong hypotheses, making the ability to revise past outputs a desirable property that can be governed by a policy. In this work, we formalise and characterise edits and revisions in incremental sequence labelling and propose metrics to evaluate revision policies. We then apply our methodology to profile the incremental behaviour of three Transformer-based encoders in various tasks, paving the road for better revision policies.

1 Introduction

Since the dawn of Wikipedia, users have made 1.7×10^9 edits to its pages. Its most revised entry contains 56,713 revisions, all documented in the page history.[1] In such an active community, conflicts inevitably occur. Editors can begin competing to override each other's contributions, causing dysfunctional *edit warrings*.[2] To help regulate the environment, an editing policy is in force, aiming at making edits constructive and improving quality.[3]

Edits, revisions and policies are key concepts in incremental processing, where a model must rely on partial input to generate partial output. Incrementality can help optimise reactivity, naturalness, quality and realism in interactive settings (Schlangen and Skantze, 2011). This is particularly relevant in dialogue models whose NLU components need to operate on incoming input, *e.g.* while performing NER, slot filling or disfluency detection, or doing simultaneous translation.

Local ambiguities in the linguistic input and transient mistakes by the model can result in wrong partial hypotheses, so that the ability to *revise*, by *editing* previous outputs, is desirable (Kahardipraja et al., 2023). Beyond monitoring the occurrence of edits, it is also beneficial to have a *policy* regulating when and which revisions should be made, reducing the occurrence of undesirable edits. Existing literature using consolidated incremental evaluation metrics falls short in capturing relevant nuances of the incremental behaviour in terms of revisions.

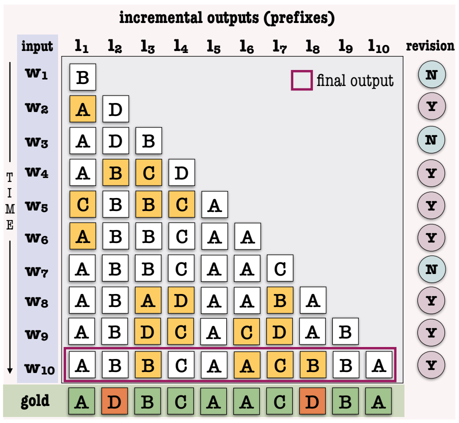

Figure 1: Constructed example of an incremental chart containing output prefixes with marked edits (yellow) and revisions in incremental sequence labelling. Red stands for wrong final predictions wrt. the gold standard.

In this work, we propose an evaluation methodology for revision policies in incremental sequence labelling. A constructed example is shown in Figure 1, with revisions indicated in the right column. Specifically, our contributions to address the identified evaluation gap are: A formalisation of revision policy in incremental sequence labelling, characterising types of edits and of revisions (§4.1-4.2); a proposal of specialised evaluation metrics for revision policies, accompanied by a discussion on the desired behaviour of incremental processors (§4.4-4.5); and a demonstration of our methodology with an analysis of the revision policy in three sequence labelling Transformer-based models (§5).[4]

[1]According to Wikimedia Statistics and wiki Special.
[2]https://en.wikipedia.org/wiki/Wikipedia:Edit_warring
[3]https://en.wikipedia.org/wiki/Wikipedia:Editing_policy

[4]Our implementation is available at `https://github.com/briemadu/inc-eval-revisions` with accompanying documentation on how to run the evaluation for other models.

2 Motivation

Incremental natural language processing[5] has *time* at front line, being pivotal for interactive settings. At each time step, models must operate on partial input to deliver partial output, but sometimes previous decisions have to be revised. For example, at time step 4 in Figure 1, the labels for the input tokens 2 and 3 were edited into new states. With regard to revisions, at least three types of incremental processors exist, as summarised in Table 1:

1. Inherently incremental but monotonic models. They keep an internal state that is updated and used to extend the output at each time step, but cannot revise previous outputs.
2. Non-incremental models used with a *restart-incremental* interface, being forced to perform a full recomputation at each time step. Such models revise the output as a by-product of their recomputations.
3. Incremental models with a dedicated policy to detect the need to perform revisions only when deemed necessary and, more specifically, deciding which parts of the output prefix need to be revised and how.

		non-incremental	**incremental**
revisions	no	n/a	strictly monotonic outputs
	yes	recomputation policy doing revisions as a by-product	revision policy

Table 1: Types of incremental processors.

Monotonicity avoids instability in the output, allowing subprocesses to start immediately, as it is certain that the outputs will not change. However, they never recover from mistakes, which is one of the drawbacks of employing vanilla RNNs and LSTMs (Hochreiter and Schmidhuber, 1997).

Models that depend on the availability of full sentences at once can be "incrementalised" with the *restart-incremental* paradigm (Schlangen and Skantze, 2011), causing revisions to occur via recomputations.[6]

[5]For a review, see Köhn (2018). In other contexts, also referred to as real-time processing (Pozzan and Trueswell, 2015) or streaming (Kaushal et al., 2023).

[6]Also called *incremental interface* (Beuck et al., 2011a) or *beat-driven approach* (Baumann et al., 2011).

Cutting-edge NLP models currently rely on Transformers (Vaswani et al., 2017), which are non-incremental. Using them in a *restart-incremental* fashion requires recomputing from scratch at every time step, which we hereby name the *naive recomputation policy*. It is a very expensive policy because, for a sequence of n tokens, the complexity is $\sum_{i=1}^{n} i^2$ (*i.e.* the n-th square pyramidal number). Besides, this naive approach wastes computational budget, because not all recomputations cause revisions. The results reported by Kahardipraja et al. (2023), for example, show that only around 25% of the recomputations actually changed the output prefix. The disadvantages of the naive policy can be alleviated by a smarter policy that cuts down the number of time steps with recomputations.

Still, beyond deciding when to *recompute*, a revision policy par excellence should directly guide the more specific decision of when (and what) to actually *revise*, and must be evaluated accordingly.

3 Related Literature

Revisability is in the nature of incremental processing: Hypothesis revision is a necessary operation to correct mistakes and build up a high-quality final output (Schlangen and Skantze, 2011). Still, there is a trade-off between requiring that later modules handle a processor's revisions and buying stability by reducing some of its incrementality, which makes the concept of *hypothesis stability* very relevant (Baumann et al., 2009). Beuck et al. (2011a) argue that performing revisions should not take as long as the initial processing, so as to retain the advantages of incremental processing. They propose two strategies: Allowing revisions only within a fixed window or limiting their types. Empirically determining how often a model changes the output is an aspect of their analysis we also rely on.

The restart-incremental paradigm was investigated for Transformer-based sequence labelling by Madureira and Schlangen (2020) and Kahardipraja et al. (2021); recently, adaptive policies were proposed to reduce the computational load (Kaushal et al., 2023; Kahardipraja et al., 2023). Rohanian and Hough (2021) and Chen et al. (2022) explored adaptation strategies to use Transformers for incremental disfluency detection. In simultaneous translation, where policies are a central concept (Zheng et al., 2020a; Zhang et al., 2020), the restart-incremental approach is in use and revisions are studied (Arivazhagan et al., 2020; Sen et al., 2023).

latency, quality, stability	simultaneous translation	Arivazhagan et al. (2020) Ma et al. (2020)
quality, responsiveness, robustness, stability	speech recognition and diarization	Baumann et al. (2009) Addlesee et al. (2020)
similarity, timing, diachronic	general	Baumann et al. (2011)
fluency, latency, quality, recovery capabilities, timing	simultaneous interpreting (MT and speech synthesis)	Baumann et al. (2014)
decisiveness, monotonicity, stability, timeliness	POS tagging	Beuck et al. (2011a)
amount of predicted information, connectedness, delay, inclusiveness, monotonicity, quality	parsing	Beuck et al. (2011b, 2013) Köhn and Menzel (2014)
cognitive aspects, efficiency	neural coreference resolution	Grenander et al. (2022)
jumpiness, position	reference resolution	Schlangen et al. (2009)
accuracy, integration, representational similarity	sequence-to-sequence	Ulmer et al. (2019)
consistency, diminishing returns, interruptibility, monotonicity, preemptability, (recognisable) quality	anytime algorithms	Zilberstein (1996)

Table 2: Overview of relevant properties for incremental evaluation in various tasks.

Sequence labelling is a staple of various incremental linguistic tasks possibly used in dialogue systems, like SRL (Konstas et al., 2014), POS-tagging (Beuck et al., 2011a), dialogue act segmentation (Manuvinakurike et al., 2016), disfluency detection (Hough and Schlangen, 2015) and dependency parsing (Honnibal and Johnson, 2014).

Revision Categorisation and Prediction Approaches to categorise the properties of revisions or edits exist in various areas. Faigley and Witte (1981) examine the effects and causes of revisions in writing, providing a taxonomy on whether revisions change meaning and bring new information. Afrin and Litman (2018) classify revision quality by whether they improve student essays. Anthonio et al. (2020) categorise revisions and edits in WikiHow in terms of what they cause to the text. Wikipedia's edits have also been classified according to factuality and fluency (Bronner and Monz, 2012) and intents (Rajagopal et al., 2022). Other typologies and taxonomies have been proposed for translation revisions (Fujita et al., 2017) and multilingual NLG revision operations (Callaway, 2003).

Vaughan and McDonald (1986) outline three phases of the revision process in NLG: Recognition, editing and re-generation. Revision rules have been applied for incremental summarisation by Robin (1996). Non-incremental revision learning models also exist, relying on revision rules for dependency parsing (Attardi and Ciaramita, 2007) or classification in POS-tagging (Nakagawa et al., 2002). Predicting stability and accuracy of hypotheses is a relevant task (Selfridge et al., 2011), which allows to distinguish hypotheses that will survive and are thus more reliable (Baumann et al., 2009).

Incremental Evaluation Table 2 presents an overview of relevant properties for incremental evaluation. In their seminal work, Baumann et al. (2011) define three general categories of metrics for incremental processing: *similarity*, *timing* and *diachronic*, which can be employed in incremental sequence labelling. They are suitable for capturing *e.g.* instability (edit overhead), quality of prefixes (correctness) and lag (correction time). Kaushal et al. (2023) propose streaming exact match, comparing prefixes with the final gold standard. While these metrics capture instability and correctness of output prefixes, we lack a standard way to evaluate the quality of the performed revisions. We thus complement their work by proposing fine-grained metrics focusing on revisions and recomputations.

4 Evaluation Methodology

In this section, we present our evaluation methodology for incremental sequence labelling with a focus on revisions. After formalising the task, we characterise revisions and edits, define policies and revision-oriented metrics and discuss the ideal behaviour of incremental sequence labelling models.

4.1 Formalisation

We begin by formalising incremental sequence labelling tasks, extending the similar definition of streaming sequence tagging by Kaushal et al.

(2023) with *edits* and *revisions*. Like them, we assume an idealised format where incremental units are well-defined, fixed and complete input tokens, and a model that produces a label for every new input token, so that the output is necessarily extended at every time step. Note, however, that incremental processors may have to operate at sub-token level or with transitional input, which requires the capability of retracting decisions and adjusting to varying length in real-time. In some models, outputs may not have an immediate one-to-one correspondence to the input (*e.g.* due to a delay strategy (Baumann et al., 2011), or to techniques like opportunistic decoding (Zheng et al., 2020b)) and parallel hypotheses can be kept in memory. See Schlangen and Skantze (2011) for details.

Let $L = \{L_1, \dots, L_M\}$ be a set of labels. In standard sequence labelling, the task is to map an input sequence of n tokens $(w_i)_{i=1}^n$ to an output sequence of n labels $(l_i)_{i=1}^n$, $l_i \in L$. Each output label l_i classifies its corresponding token w_i. The task is more complex than plain token-level classification because the sequential nature of the input and the output need to be taken into account when predicting labels. If available, a gold-standard sequence $(g_i)_{i=1}^n$, with $g_i \in L$, is used to evaluate the correctness of the predicted output sequence.

In an incremental setting, the input is provided in a piecemeal fashion, one token at a time. At each time step $t = 1, 2, \dots, n$, an increasing input prefix $(w_i)_{i=1}^t$ is available to the model and an output prefix $(l_i)_{i=1}^t$ is predicted. Therefore, an input sequence with n tokens will result in n output prefixes $p_1, p_2, \dots, p_n$, which we consider to be partial hypotheses for the final output. Each p_i is a sequence of i labels, containing one additional label at the right in relation to p_{i-1}. The last hypothesis p_n is the final decision of the model, having observed the full input. The complete sequence of prefixes can be represented as a lower triangular matrix, whose cells c_i^j contain the label assigned to w_i at time j and each row i contains p_i. We can represent the incremental input and output in an *incremental chart* (IC) as follows:

$$\begin{array}{r|r|ccccc}
\hline
w_1 & p_1 = & l_1^1 & & & & \\
w_1, w_2 & p_2 = & l_1^2 & l_2^2 & & & \\
w_1, w_2, w_3 & p_3 = & l_1^3 & l_2^3 & l_3^3 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\
w_1, w_2, \dots, w_n & p_n = & l_1^n & l_2^n & l_3^n & \cdots & l_n^n \\
\cline{2-7}
 & \text{gold} = & g_1 & g_2 & g_3 & \cdots & g_n \\
\hline
\end{array}$$

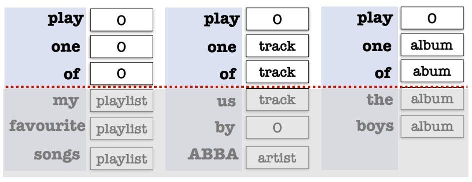

Figure 2: Illustrative example of multiple locally valid hypotheses for the prefix *play one of*. Only after more input is processed definite labels can be assigned.

At each time step t, the observation of the new input token w_t causes the model to i) extend the output sequence with one label for w_t (an addition) and ii) optionally also change its current hypotheses $l_1, \dots, l_{t-1}$ for previous tokens (substitutions).

An *edit* occurs at time t for label i if $l_i^t \neq l_i^{t-1}$, meaning that the model's prediction for w_i's label changed. A *revision* occurs when, apart from the compulsory addition, a prefix changes at time t in relation to the previous prefix, *i.e.* when at least one label is edited.[7] In Figure 1, revisions occur at time steps $2, 4, 5, 6, 8, 9$ and 10. Highlighted labels in the prefixes are edits.

Gold Standard Evaluation can be done with respect to incremental or non-incremental gold standards (Baumann et al., 2011). Often, only the non-incremental version is available, *i.e.* the labels on the complete sequence, assigned having all left and right context taken into account. A genuinely incremental gold standard contains step-by-step gold prefixes encoding interpretations that are *locally valid* until right context renders it invalid, as illustrated in Figure 2.[8] Since it is usually not available, we can instead "incrementalise" the final gold standard by deriving all its prefixes as hard labels. But this approach somewhat unfairly expects that, even at steps with multiple locally valid interpretations, the model commits to the final decision without observing the input that actually induces that interpretation as correct and the others as wrong. Moreover, using an independent gold standard conflates the external overall performance of the model with the quality of its internal incrementality; an alternative is to consider the final output of the model as a silver standard (Baumann et al., 2011). The correctness of labels and prefixes is then measured with a metric M with respect to the defined target.

[7]The addition is not taken into account here, as it has no precedent label to be compared to at this point. The first time step is by definition not a revision, since there is no prefix yet.

[8]For existing examples, see Hrycyk et al. (2021), Rawat and Barres (2022) and Beuck et al. (2011b).

	Quality	Edits (labels)	Example	Revisions (prefixes)	Example
Convenience	convenient	change incorrect label	(5,1)	change incorrect prefix	5
	inconvenient	change correct label	(4,2)	change correct prefix	4
Effectiveness	effective	incorrect label $\rightarrow$ correct	(5,4)	improve prefix correctness	6
	ineffective	incorrect label $\rightarrow$ incorrect	(9,3)	do not change prefix correctness	9
	defective	correct label $\rightarrow$ incorrect	(4,3)	worsen prefix correctness	4
Novelty	innovative	label $\rightarrow$ new state	(9,6)	n/a	n/a
	repetitive	label $\rightarrow$ previous state	(6,1)	n/a	n/a
(Local) Recurrence	recurrent	subsequence with $>$ 1 edit	(9,3)	subsequence with $>$ 1 revision	8
	steady	subsequence with 1 edit	(4,2)	subsequence with 1 revision	2
Oscillation	oscillating	label with $>$ 1 edit	(6,1)	$>$ 1 revision	all
	stable	label with 1 edit	(4,2)	single revision	-
Company	accompanied	prefix with $>$ 1 edit	(9,6)	prefix with $>$ 1 edit	5
	isolated	prefix with 1 edit	(6,1)	prefix with 1 edit	6
Connectedness	connected	other neighbouring edit	(9,4)	only connected edits	9
	disconnected	no neighbouring edits	(5,1)	only disconnected edits	2
	both	n/a	n/a	both types of edits	5
Distance	short range	near current time step	(5,4)	only short range edits	2
	long range	far from current time step	(9,3)	only long range edits	6
	both	n/a	n/a	both types of edits	5
Definiteness	definite	label $\rightarrow$ final state	(4,2)	prefix $\rightarrow$ final state	10
	temporary	label $\rightarrow$ temporary state	(5,3)	prefix $\rightarrow$ temporary state	8
Time	intermediate	input still partial	(5,4)	input is still partial	4
	final	at final time step	(10,3)	at the final time step	10

Table 3: Characterisation of edits and revisions. The examples refer to Figure 1, pointing to the (time step, label index) positions for edits and time steps for revisions. Here the gold standard is used to judge prefix correctness.

4.2 Characterisation of Revisions and Edits

In this section, we propose a detailed characterisation for the types of edits and revisions based on ten dimensions, summarised in Table 3, as a means to evaluate revision policies. In the next paragraphs, we assume that either a genuine or a constructed incremental sequence of target prefixes has been selected according to the current needs. We will use Figure 1 and its gold standard as examples.[9]

To characterise edits, we consider the state of an output label in the current prefix in relation to its state in the previous prefix, which are different by definition. They relate to a label's development in time (vertically in their IC column) or to the prefix they belong to (horizontally in their IC row). The dimensions to characterise edits serve the purpose of defining the qualities of the revisions, which operate on prefixes.

4.2.1 Edits

The main aspect to account for is whether labels need to be edited in the first place and, if yes, whether they are edited into the desired state. Edits on correct labels are *inconvenient*, and also *defective*, since the label will fatally change into a wrong label. This happens, for instance, at l_2 in step 4, as the correct label D is edited into a wrong B. Edits on incorrect labels are *convenient* and can be *effective* (if it enters into a correct state, like l_4 at $t = 5$, which changes from an incorrect D to a correct C) or *ineffective* (if it enters into another incorrect state, *e.g.* l_3 at $t = 9$, which changed from an incorrect A to a still incorrect D).

Other dimensions can be used to analyse the behaviour of the processor. *Innovative* edits cause the label to change into a new state. For instance, l_6 becomes a C for the first time at $t = 9$. In the next step, it is edited back into its previous state A, and we consider it to be a *repetitive* edit.

Local *recurrence* refers to whether the edit occurs in isolation in neighbouring time steps (edit subsequences in an IC's column). *Oscillation* refers to how many edits occur in its complete column, just one (*stable*) or more (*oscillating*). For instance, l_3 has two groups of recurrent edits along the time axis, whereas l_2 has one *steady* and stable edit.

Company characterises whether the edit occurs with other edits in a prefix (same IC's row). In Figure 1, l_6 is edited together with other labels at $t = 9$, whereas l_1 is edited in isolation at $t = 6$. *Ac-*

[9]More examples are available in the code repository.

companied edits can be either *connected* (*i.e.* with directly neighbouring edited labels, as in $t = 4$) or *disconnected* to the other edits in its prefix.

Short or *long range* refers to how far the edited label is from the current time step, defined by a distance parameter d. If we set $d = 2$, the edit that changes l_4 into a C at $t = 5$ is short range because it is less than 2 time steps away from the current token being processed. On the other hand, l_3 is edited at $t = 9$, very distant from the right frontier.

Edits can also be *definite* or *temporary*. Definite edits make the label enter into its final state, like l_2 at $t = 4$. Temporary edits are those like the B for l_3 at $t = 5$: It still gets edited further before a final decision is reached (here, also a B). Besides, edits can occur in *intermediate* steps during processing, when the input sequence is incomplete, or at the *final* time step, when the full sequence is available.

4.2.2 Revisions

Similar to edits, revisions are *inconvenient* if they occur on correct prefixes (that should not change), and thus also *defective*, because correctness necessarily decreases. The prefix at $t = 3$ is correct, so the revision at $t = 4$ causes the labels to become wrong. *Convenient* revisions are *effective* if they improve correctness, like at $t = 6$ where the number of correct labels in the prefix increases from 3 to 4, otherwise they can be *ineffective* (edits occur but correctness remains the same, like at $t = 9$) or again *defective*.

Revisions are *locally recurrent* when other revisions occur in neighbouring time steps. We see that from $t = 4$ to $t = 6$. The revision at $t = 2$ is *steady*, as no other revisions occur immediately before or after it. If only one revision occur while a sequence is processed, it is *stable*, otherwise it is *oscillating*. In our example, all revisions are therefore oscillating.

Company, *connectedness* and *distance* refer to what types of edits the revision causes. At the second time step, the prefix contains only a *disconnected* and *short range* edit, whereas at the fifth time step we observe *accompanied* edits, one *connected* and one *disconnected* group and one short and two long range edits.

Definite revisions create prefixes that will not be further edited. In our example, this only happens in the last time step; all others are *temporary*. *Intermediate* revisions happen when the input is not yet completed, otherwise they are *final*.

4.2.3 Recomputations

In models that detach recomputations from revisions, the recomputations should also be evaluated. Recomputations are *active* if they actually result in a revision, otherwise they are *inactive*. The quality of the resulting revisions can then be evaluated with the characteristics above.

4.3 Policies

To perform good revisions, a model must decide *when* to recompute or revise. For that decision, both a *revision policy* and a *recomputation policy* can be generally defined as:

$$\pi : \text{IC} \rightarrow [0,1] \qquad \pi(\text{IC}_t) = \Pr(r|\text{IC}_t) \quad (1)$$

It gives the probability of performing a revision or recomputation r, respectively, given the state of the incremental chart at time t.[10] When $\Pr(r|\text{IC}_t) > \tau$, where τ is a threshold hyperparameter, a revision/recomputation is performed. If the revisions are not a mere consequence of full recomputations, the model must then also decide *what* and *how* to edit.

4.4 Metrics

Traditional sequence labelling evaluation metrics like accuracy or F1 can be computed on label, sequence or dataset level. The incremental dimension requires its own metrics, some of which we discussed in §3. Here, we propose specific metrics to evaluate revision and/or recomputation policies. For each time step t in a sequence, either a revision (R) occurred, which is sometimes effective (R_e), or only an addition (A). Assuming we have established a metric for prefix correctness,[11] we know whether the prefix at $t-1$ was correct (C) or incorrect (I). That results in a distribution of N actions in $\{R, A\} \times \{C, I\}$. From these counts, we derive the metrics in Table 4, computed either per sequence or over the whole dataset. Models that have the option to *recompute* (R') can also be evaluated in $\{R', \neg R'\} \times \{C, I\}$ with two additional metrics.

Since only *effective* revisions are actually desired, the R in the numerators can be replaced by R_e for a more focused evaluation. Revisions can

[10]It is also possible to make the policy dependent only in a portion of the IC, as done *e.g.* by Kahardipraja et al. (2023).

[11]A binary variable or a continuous variable, like accuracy, with a defined threshold for tolerated incorrectness.

		The fraction of...
Rate of Revision	R/N	time steps in which the model revises
Rate of Recomputation	R'/N	time steps in which the model recomputes
Rate of Active Recomputation	$(R' \cap R)/R'$	recomputations that actually causes a revision
R-Pertinence	$(R \cap I)/R$	revisions that edit incorrect prefixes (adapted precision)
R-Appropriateness	$(R \cap I)/I$	incorrect prefixes that are revised (adapted recall)
A-Pertinence	$(A \cap C)/A$	additions upon correct prefixes (adapted precision)
A-Appropriateness	$(A \cap C)/C$	correct prefixes that are not revised (adapted recall)
R$_e$-Pertinence	$(R_e \cap I)/R$	revisions that effectively edit incorrect prefixes
R$_e$-Appropriateness	$(R_e \cap I)/I$	incorrect prefixes that are revised effectively

Table 4: Proposed metrics for evaluating recomputation and revision policies. N is the total number of time steps.

be further weighted by how often and how far in the sentence processing they happen. Similarly, edits can be assessed by their correction time and survival time (Baumann, 2013).

4.5 Ideal Processor

Let us now delineate the ideal behaviour of a revision policy for an incremental sequence labelling model. A utopian model would always output the correct label and thus never need to produce edits or revisions (Kahardipraja et al., 2023).[12] But due to the incremental nature of language processing, models should not be penalised for building hypotheses that are *locally valid*, as long as a revision is timely triggered. That is, however, complex to know in raw textual input where local ambiguities are not identified. Instead, we can characterise an outlook according to desirable principles and available resources. In scenarios with an infinite time budget, we can simply wait for the input to be complete. If computation budget can be afforded, restart-incrementality is a good fit. But the constraints are not always so loose.

An ideal revision policy should thus revise as rarely as possible for stability. If a prefix/label is correct, the policy should avoid revising it, whereas an incorrect prefix/label should be revised (maybe not immediately, but eventually). It should always trigger effective, convenient, and definite revisions, preferably in earlier time steps.[13] Recurrent or oscillating revisions cause more instability and should be avoided. Innovative edits are preferable (as long as they are effective), and short range is better to be combined with delay strategies. Connectedness is a relevant dimension for BIO labelling schemes: If, for instance, the beginning label is edited, ideally the middle labels should change simultaneously. Finally, accompanied edits can be further evaluated in their relation to each other and the linguistic input. A good recomputation policy should, additionally, always result in active revisions.

In terms of metrics, R-Pertinence and A-Appropriateness should be exactly 1, *i.e.* all revisions should occur upon incorrect prefixes and all correct prefixes should not be revised. A-Pertinence and R-Appropriateness should be as high as possible, but cannot be expected to be exactly 1 because it may take some time steps until the input that actually resolves the ambiguity or mistake is observed.

5 Architecture Profiling

We now apply our methodology to profile the revision policy behaviour of three models: The reference restart-incremental Transformer and the two TAPIR variations, which have a recomputation policy, proposed by Kahardipraja et al. (2023). We evaluate them on three sequence labelling tasks: Slot filling (Coucke et al., 2018), POS tagging (Silveira et al., 2014) and NER (Tjong Kim Sang and De Meulder, 2003), using the final output as gold standard.[14] Note that the same profiling can be applied to any model with the ability of performing revisions on any sequence labelling task.

Quantitative Assessment Table 5 shows that the recomputation policy implemented in TAPIR reduces the number of restarts to between 10% and 25% in comparison to the restart incremental ap-

[12] That is indeed the case for strictly monotonic models if we use their final output as gold standard.

[13] In the beginning, the absence of both right and left context makes prediction harder. Towards the end, the availability of more left context should lead to less, and better, revisions.

[14] Here we use only the buffer outputs to evaluate the resulting revisions on prefixes that would have been passed on to downstream processors. We do not consider the temporary outputs of the LSTM that the original model had access to when deciding to perform a recomputation. Please refer to the original paper for the details on non-incremental and incremental performance on these tasks.

	% recomputation			% active recomputation			% revision		
	NER	POS	Slot	NER	POS	Slot	NER	POS	Slot
Rest.Incremental-Transformer	100.00	100.00	100.00	7.77	19.29	21.23	7.77	19.29	21.23
TAPIR-LTReviser	13.77	24.52	20.34	20.23	39.55	39.44	2.78	9.69	8.02
TAPIR-TrfReviser	10.36	20.23	21.41	25.36	34.09	33.65	2.62	6.89	7.20

Table 5: Rate of (active) recomputations and of revisions for each model and task.

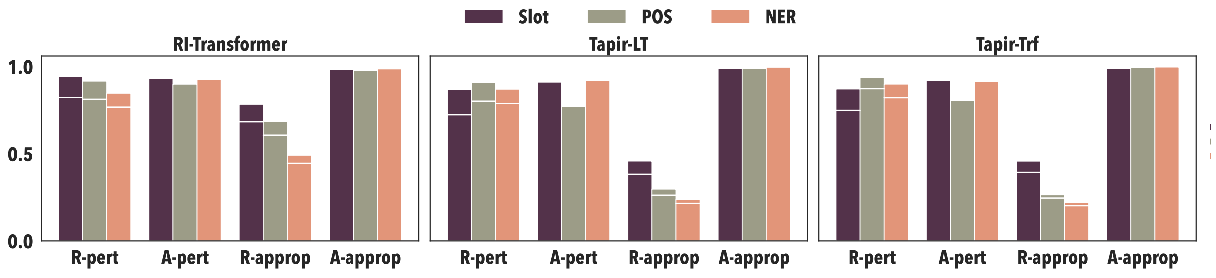

Figure 3: Revision metrics for all models and tasks. The white lines represent only the effective revisions.

proach, considerably alleviating the computation load; the number of revisions is also 2 to 3 times lower. Still, only up to 40% of the remaining recomputations are active, which means that the use of computational budget is still suboptimal. Furthermore, in Figure 3 we see that A-Appropriateness is very close to 1, as it should be. R-Pertinence is slightly below the ideal 1, but still greater than 0.8 in all cases, although it is around 0.1 lower when only effective revisions are considered. A-Pertinence is at similar values, with a lower result for POS-tagging. R-Appropriateness and R_e-appropriateness, however, are low in the restart-incremental Transformer and becomes even lower in the TAPIR models.

This may be evidence that the TAPIR models are waiting for more input before deciding to recompute an incorrect prefix, which is in line with the shifts in the distributions we observe in Figure 4. TAPIR tends to have more revisions towards the end of the sentence than the restart-incremental Transformer. This strategy can indeed help revisions be more effective, given that more left context is available, but it also results in having to wait longer for final decisions, which is not ideal.

The cumulative distributions of the fraction of time steps with revisions per sentence, shown in Figure 5, illustrate that the policy reduces the number of revisions per sentence: 50% or less of the sentences have no revisions in the naive policy, which makes all recomputation effort be used to perform only an addition, while TAPIR's policy caused more sentences to not trigger revisions.

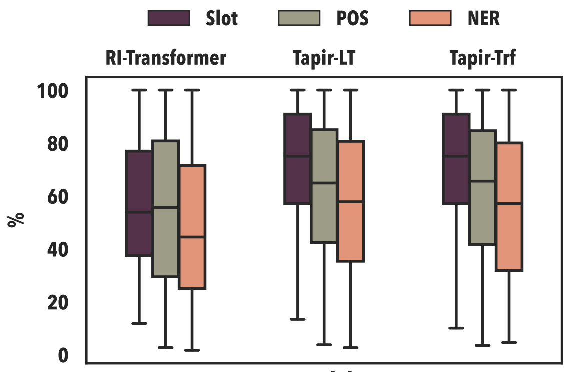

Figure 4: How far in the sentence processing (% of time steps or tokens) revisions occur.

Qualitative Assessment Figures 6 and 7 show the percentages of edits and revisions types to characterise TAPIR-TrfReviser's policy. In terms of edits, most are effective, convenient, innovative and steady. Only around 50% are short range, which means that delay strategies would have limited improvements in reducing edit overhead. For slot filling, around 20% of the edits occur in the last time

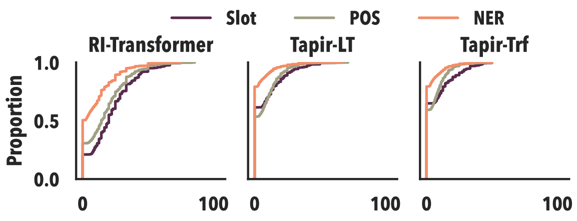

Figure 5: Proportion of time steps with revisions per sentence (cumulative).

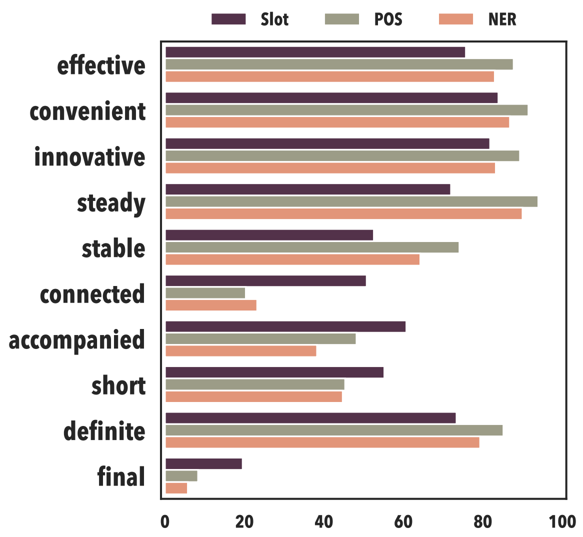

Figure 6: Edits by TAPIR-TrfReviser's policy.

	Slot	POS	NER
effective	74.9	87.3	82.1
defective	16.6	7.8	12.5
ineffective	8.5	4.9	5.3
convenient	87.1	93.8	89.9
inconvenient	12.9	6.2	10.1
steady	59.0	85.0	83.0
recurrent	41.0	15.0	17.0
oscillating	72.9	76.5	63.3
stable	27.1	23.5	36.7
isolated edit	61.8	72.2	78.3
accompanied edits	38.2	27.8	21.7
connected edits	28.8	8.9	11.6
disconnected edits	67.9	87.1	85.9
dis and connected edits	3.3	4.0	2.5
short range	50.7	45.0	43.4
long range	31.9	41.9	48.4
short and long range	17.5	13.0	8.3
temporary	46.3	51.9	41.6
definite	53.7	48.1	58.4
intermediate	80.1	91.5	94.3
final	19.9	8.5	5.7

Figure 7: Revisions by TAPIR-TrfReviser's policy.

step, which is undesired, because it means that the intermediate predictions for these labels are wrong until the model processes the full sentence.

Regarding revisions, TAPIR's policy works best for POS-tagging in terms of effectiveness, convenience, oscillation and recurrence, and worse for slot filling. Most of the edits are isolated, which means that recomputations have been performed for the full partial input to only result in one edit. The proportion of short vs. long range and temporary vs. definite revisions was, in general, balanced. We also see that proportionally fewer revisions occurred in the final step. Although the high percentage of intermediate revisions is high, Figure 4 shows that they are happening towards the end, which prevents incremental subprocessors to reliably count on the intermediate outputs. Slot filling is, here, an example of the occurrence of final revisions being less than ideal.

Based on these results, we conclude that TAPIR's policy is very successful in reducing the number of recomputations and also in revising less, but there is room for improving the quality of the resulting revisions, both in terms of metrics and of characteristics. This speaks for a more dedicated revision policy that could avoid full recomputations and use the state of the incremental chart and internal representations of the model for a more fine-grained prediction of which labels should change.

6 Conclusion

In this work, we have argued that the importance of a solid evaluation framework for revision policies in incremental sequence labelling cannot be overstated. Despite being very useful to capture some incremental aspects like instability or timeliness, existing evaluation metrics set aside other major strands of revisions. To fill that void, we have introduced metrics, characteristics and rationale to support the analysis of revision policies. This methodology serves as a tool to ascertain their quality, to determine their appropriateness in different contexts and to compare different policies.

We identify a few more roads to quality: The creation of incremental gold standards containing locally valid hypothesis, the development of fine-grained revision policies predicting what to revise and a more systematic integration of linguistic aspects of the input into the evaluation procedure. For those willing to drive those routes, we hope our methodology has paved the road well.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions. We also thank Kaushal et al. (2023) for a conversation on this topic at EACL, in particular about the locally valid hypotheses.

References

Angus Addlesee, Yanchao Yu, and Arash Eshghi. 2020. A comprehensive evaluation of incremental speech recognition and diarization for conversational AI. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3492–3503, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tazin Afrin and Diane Litman. 2018. Annotation and classification of sentence-level revision improvement. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 240–246, New Orleans, Louisiana. Association for Computational Linguistics.

Talita Anthonio, Irshad Bhat, and Michael Roth. 2020. wikiHowToImprove: A resource and analyses on edits in instructional texts. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5721–5729, Marseille, France. European Language Resources Association.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. Association for Computational Linguistics.

Giuseppe Attardi and Massimiliano Ciaramita. 2007. Tree revision learning for dependency parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 388–395, Rochester, New York. Association for Computational Linguistics.

Timo Baumann. 2013. *Incremental Spoken Dialogue Processing: Architecture and Lower-level Components*. Ph.D. thesis, Universität Bielefeld, Germany.

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 380–388, Boulder, Colorado. Association for Computational Linguistics.

Timo Baumann, Srinivas Bangalore, and Julia Hirschberg. 2014. Towards simultaneous interpreting: the timing of incremental machine translation and speech synthesis. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Papers*, pages 163–168, Lake Tahoe, California.

Timo Baumann, Okko Buß, and David Schlangen. 2011. Evaluation and Optimisation of Incremental Processors. *Dialogue and Discourse*, 2(1):113–141.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011a. Decision strategies for incremental POS tagging. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 26–33, Riga, Latvia. Northern European Association for Language Technology (NEALT).

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2013. Predictive incremental parsing and its evaluation. In *Computational Dependency Theory*, pages 186–206. IOS Press.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011b. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*, pages 290–299.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 356–366, Avignon, France. Association for Computational Linguistics.

Charles Callaway. 2003. Multilingual revision. In *Proceedings of the 9th European Workshop on Natural Language Generation (ENLG-2003) at EACL 2003*, Budapest, Hungary. Association for Computational Linguistics.

Angelica Chen, Vicky Zayats, Daniel Walker, and Dirk Padfield. 2022. Teaching BERT to wait: Balancing accuracy and latency for streaming disfluency detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 827–838, Seattle, United States. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint*, arXiv:1805.10190.

Lester Faigley and Stephen Witte. 1981. Analyzing revision. *College composition and communication*, 32(4):400–414.

Atsushi Fujita, Kikuko Tanabe, Chiho Toyoshima, Mayuka Yamamoto, Kyo Kageura, and Anthony Hartley. 2017. Consistent classification of translation revisions: A case study of English-Japanese student translations. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 57–66, Valencia, Spain. Association for Computational Linguistics.

Matt Grenander, Shay B. Cohen, and Mark Steedman. 2022. Sentence-incremental neural coreference resolution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 427–443, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

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

Julian Hough and David Schlangen. 2015. Recurrent Neural Networks for Incremental Disfluency Detection. In *Interspeech 2015*, pages 849–853.

Lianna Hrycyk, Alessandra Zarcone, and Luzian Hahn. 2021. Not so fast, classifier – accuracy and entropy reduction in incremental intent classification. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 52–67, Online. Association for Computational Linguistics.

Patrick Kahardipraja, Brielen Madureira, and David Schlangen. 2021. Towards incremental transformers: An empirical analysis of transformer models for incremental NLU. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1178–1189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Patrick Kahardipraja, Brielen Madureira, and David Schlangen. 2023. TAPIR: Learning adaptive revision for incremental natural language understanding with a two-pass model. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4173–4197, Toronto, Canada. Association for Computational Linguistics.

Ayush Kaushal, Aditya Gupta, Shyam Upadhyay, and Manaal Faruqui. 2023. Efficient encoders for streaming sequence tagging. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 418–429, Dubrovnik, Croatia. Association for Computational Linguistics.

Arne Köhn. 2018. Incremental natural language processing: Challenges, strategies, and evaluation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2990–3003, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Arne Köhn and Wolfgang Menzel. 2014. Incremental predictive parsing with TurboParser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 803–808, Baltimore, Maryland. Association for Computational Linguistics.

Ioannis Konstas, Frank Keller, Vera Demberg, and Mirella Lapata. 2014. Incremental semantic role labeling with Tree Adjoining Grammar. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 301–312, Doha, Qatar. Association for Computational Linguistics.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

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

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 Meeting of the Special Interest Group on Discourse and Dialogue*, pages 252–262, Los Angeles. Association for Computational Linguistics.

Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. 2002. Revision learning and its application to part-of-speech tagging. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 497–504, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Lucia Pozzan and John C Trueswell. 2015. Revise and resubmit: How real-time parsing limitations influence grammar acquisition. *Cognitive Psychology*, 80:73–108.

Dheeraj Rajagopal, Xuchao Zhang, Michael Gamon, Sujay Kumar Jauhar, Diyi Yang, and Eduard Hovy. 2022. One document, many revisions: A dataset for classification and description of edit intents. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5517–5524, Marseille, France. European Language Resources Association.

Mrinal Rawat and Victor Barres. 2022. Real-time caller intent detection in human-human customer support spoken conversations. In *Communication in Human-AI Interaction Workshop*.

Jacques Robin. 1996. Evaluating the portability of revision rules for incremental summary generation. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 205–214, Santa Cruz, California, USA. Association for Computational Linguistics.

Morteza Rohanian and Julian Hough. 2021. Best of both worlds: Making high accuracy non-incremental transformer-based disfluency detection incremental. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3693–3703, Online. Association for Computational Linguistics.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies. In *Proceedings of the SIGDIAL 2009 Conference*, pages 30–37, London, UK. Association for Computational Linguistics.

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

Ethan Selfridge, Iker Arizmendi, Peter Heeman, and Jason Williams. 2011. Stability and accuracy in incremental speech recognition. In *Proceedings of the SIGDIAL 2011 Conference*, pages 110–119, Portland, Oregon. Association for Computational Linguistics.

Sukanta Sen, Rico Sennrich, Biao Zhang, and Barry Haddow. 2023. Self-training reduces flicker in retranslation-based simultaneous translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3734–3744, Dubrovnik, Croatia. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Dennis Ulmer, Dieuwke Hupkes, and Elia Bruni. 2019. Assessing incrementality in sequence-to-sequence models. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 209–217, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Marie M. Vaughan and David D. McDonald. 1986. A model of revision in natural language generation. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 90–96, New York, New York, USA. Association for Computational Linguistics.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. Learning adaptive segmentation policy for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2280–2289, Online. Association for Computational Linguistics.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020a. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

Renjie Zheng, Mingbo Ma, Baigong Zheng, Kaibo Liu, and Liang Huang. 2020b. Opportunistic decoding with timely correction for simultaneous translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 437–442, Online. Association for Computational Linguistics.

Shlomo Zilberstein. 1996. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73.