Assessing the Performance of an Incremental Natural Language Understanding Model for Noisy Slot Filling

Hannah Regine Fong and Ethel Ong
College of Computer Studies
De La Salle University
Manila, 1004 Philippines
hannah_regine_fong@dlsu.edu.ph, ethel.ong@dlsu.edu.ph

Abstract

Natural language understanding (NLU) systems should mirror the incremental nature of human language processing for a more responsive interaction with users. A recurrent neural network is an ideal option for an incremental NLU system but its performance lags behind bidirectional models and transformers that are not limited to context in a single direction. These models can be applied to an incremental NLU task through a restart-incremental interface where increasing input prefixes are repeatedly passed to the non-incremental models. However, the approach is computationally expensive especially for long input sequences. An alternative is to employ a two-pass model with adaptive revision to avoid unnecessary expensive recomputations. We present our evaluation of the performance of a two-pass incremental NLU model in perturbed scenarios. Results showed that performance degradation occurs when dealing with noisy data. Specifically, fine-grained noises on the character-level (e.g., typos) and word-level (e.g., speech errors) cause more performance losses compared to coarse-grained noises on the sentence-level (e.g., verbosity, simplification, paraphrasing). This underscores the need for the incorporation of robust noise-handling mechanisms in incremental NLU systems.

1 Introduction

Language processing is inherently incremental. Humans produce words one at a time, in both speaking and writing, even without having a fully formed thought in mind. Similarly, they are capable of understanding the meaning of incomplete utterances. Developing incremental natural language understanding (NLU) systems is thus important to mirror the incremental nature of language. This can lead to dialogue and interactive systems with lower latency and faster response time by letting the NLU models process partial utterances from the users.

A recurrent neural network (RNN) is the most ideal neural network architecture for the incremental NLU task because it processes text sequentially, one word at a time (Kahardipraja et al., 2023). It also maintains a recurrent state that stores information from previous time steps which can be used as context to guide the processing of the current input. However, RNN is outperformed by models that leverage bidirectional context such as bidirectional long short-term memory (BiLSTM). The transformer architecture introduced in 2017 uses self-attention mechanisms to capture all relations between tokens simultaneously, and has since achieved SOTA performance in various NLP tasks.

Bidirectional models and transformers are not designed for sequential processing that is needed in incremental NLU systems. To address this, Madureira and Schlangen (2020) deployed a restart-incremental interface where partial prefixes of an utterance are repeatedly fed into these unchanged models. This approach, however, is very computationally expensive due to the recomputations made in every time step. Kahardipraja et al. (2021) experimented with a linear transformer with recurrent computation and found that this achieves better incremental performance at the expense of lower non-incremental performance, which can be mitigated using a delay strategy.

Another key feature that must be present in incremental NLU systems is the ability to revise their previous outputs due to the inherent ambiguity of partial utterances. Restart-incremental systems meet this requirement by recomputing the entire output in every step, which is highly inefficient. Kahardipraja et al. (2023) introduces an adaptive revision policy that only performs recomputations when necessary based on the history of inputs and outputs. Their proposed model, TAPIR, achieved comparable non-incremental performance with a restart-incremental transformer and a better incremental performance and inference speed. However,

TAPIR was only evaluated on clean data, which is not realistic in real dialogue systems that are susceptible to various types of noise including typos, speech errors, verbosity, simplification, and paraphrasing (Dong et al., 2023).

In this paper, we present our experiments in evaluating the impact of different types of input perturbations to the performance and robustness of TAPIR and assessing its effectiveness in real-world scenarios. The dataset from Dong et al. (2023) is utilized for this purpose.

2 Related Works

We briefly present prior approaches to incremental NLU and their performance in perturbed scenarios.

2.1 Incremental NLU

An RNN is the most straightforward neural network architecture to use for incremental NLU due to its ability to process sequences per word and to produce an output at each time step. Liu and Lane (2016) utilized a conditional RNN for incremental joint intent detection (ID), slot filling (SF), and language modeling (LM). Their results indicate that jointly modeling the intent and slot label history as new input words arrive leads to better ID and LM performance with minor degradation in SF.

Despite its strong sequence modeling ability, an RNN is still unable to achieve a strong non-incremental performance due to its strict left-to-right processing (Kahardipraja et al., 2023). Madureira and Schlangen (2020) adapted the BiRNNs, BiLSTMs, and the transformer architectures for incrementality by using a restart-incremental interface, where increasing input prefixes are repeatedly fed into an unchanged non-incremental model. Results showed that the transformer-based model achieved the best non-incremental performance in various sequence tagging and sentence classification tasks. However, it demonstrated worse incremental performance compared to the bidirectional models in terms of edit overhead (EO), correction time (CT), and relative correctness (RC), especially in sequence tagging tasks. This degradation in incremental performance can be mitigated through strategies such as truncated training, delayed output, and prophecies.

A restart-incremental transformer is computationally expensive especially when dealing with long sequences. Instead of processing a sequence of n tokens once, the restart-incremental approach requires processing n sequences, each with $\sum_{k=1}^{n} k$ tokens. To reduce the computational cost, Kahardipraja et al. (2021) applied the linear transformer model introduced by Katharopoulos et al. (2020), which replaces the traditional softmax attention with a feature map-based dot product attention, achieving an improved time and memory complexity. Results showed that the linear transformer using recurrent computation performed worse compared to the restart-incremental transformer and linear transformer models across all the sequence tagging and classification tasks investigated in the paper. This may be attributed to the strict left-to-right processing and sub-optimal approximation of the softmax attention. However, it is significantly more efficient by not performing recomputations at each time step. The performance of the recurrent linear transformer can be improved through the combination of training with causal masking, input prefixes, and delay. This variation also achieves the best incremental performance.

Kahardipraja et al. (2023) combined the advantages of RNNs and transformers for incremental NLU by developing the Two-pass model for Adaptive Revision (TAPIR). TAPIR uses an RNN as the incremental processor (i.e., first pass) and a transformer as the reviser (i.e., second pass). Revisions are necessary in incremental NLU due to the inherent ambiguity in partial utterances or the model's poor approximation. TAPIR uses an adaptive policy which avoids making unnecessary revisions. It performs policy learning as a supervised problem through the incorporation of supervision signals, in the form of action sequences, into the training process. The action sequences consist of WRITE or REVISE actions that indicate whether the partial outputs at a particular time step must be edited or not. These are generated using a linear transformer, which combines the recurrence mechanism of RNNs and the backward update ability of transformers. Results showed that TAPIR achieved comparable non-incremental performance with better incremental performance compared to the baseline restart-incremental transformer.

2.2 Noisy NLU

Real dialogue systems encounter a lot of input perturbations and errors such as typos, ASR speech errors, simplification, verbosity, and paraphrasing (Dong et al., 2023). Constantin et al. (2019) maintained that partial utterances in incremental systems are noisier due to the short available context. How-

ever, most existing state-of-the-art NLU models are usually trained on perturbation-free datasets, which leads to poor performance in real scenarios. Liu and Lane (2016) evaluated their incremental joint ID and SF model, trained on clean data, using noisy ASR speech input. They obtained a worse performance on the noisy data with a higher intent error by 2.87 and a lower slot F1-score by 7.77%. Constantin et al. (2019) incorporated human, ASR, and artificial noises into the training data. The artificial noises were generated using random substitution, insertion, and deletion of words in the original clean utterances. Results showed that the model trained on noisy data achieved a better performance than those trained on clean data.

3 Task Description

We provide a formal definition of the slot filling task and describe the DemoNSF dataset used in the experiment. Additionally, the architecture of the TAPIR model by Kahardipraja et al. (2023) is outlined, which serves as the reference incremental model used in the study.

3.1 Slot Filling Task

Slot filling is a sequence tagging task that assigns a semantic label to every token in a given utterance. Given an input word sequence with N tokens $x = (x_1, ..., x_N)$, SF tags each token with a slot label $y = (y_1, ..., y_N)$ from a predefined list of slot labels. In this paper, we follow the IOB tagging format where the "B" prefix indicates the beginning or first token of a slot, "I" indicates a token inside or at the end of a slot, and "O" indicates a word that does not belong to the predefined list of slot labels in the dataset. Table 1 shows a sample slot annotation where "*stansted airport*" is tagged as a "*depart*" slot, denoting the place of departure, and "11:45" is tagged as a "*leave*" slot, denoting the time of leaving. The other tokens are labeled as "O", indicating that they do not belong to any slot.

3.2 Dataset

We adopted the dataset from Dong et al. (2023) and refer to it as **DemoNSF**, after the multi-task demonstration-based generative framework they proposed for noisy slot filling. DemoNSF is a noise-robustness evaluation dataset that classifies noises into sentence-level (verbosity, paraphrasing, simplification), character-level (typos), and word-level (speech). An example of a clean utterance and its perturbed versions is presented in Table 2.

3.3 Two-pass Model for Adaptive Revision (TAPIR)

The TAPIR architecture, depicted in Figure 1, has four (4) components: incremental processor, reviser, memory, and controller. The incremental processor (i.e., first-pass model) is a recurrent LSTM network that outputs a slot label for each new input token per time step. The reviser (i.e., second-pass model) is a transformer that is used to recompute the slot labels of the entire partial input at a specific time step. The memory stores the history of inputs and outputs in caches for fast access. The controller is a modified LSTMN that parametrizes the revision policy which models the effect of the new input token on past outputs.

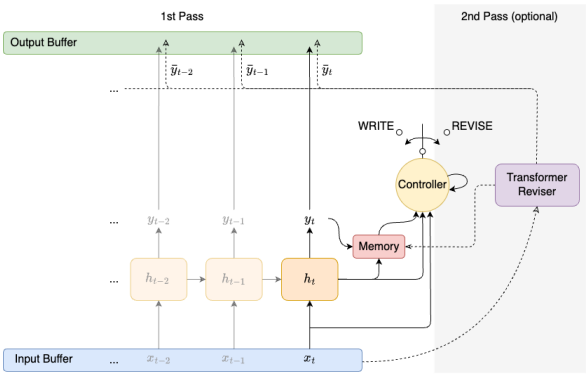

Figure 1: TAPIR Architecture Diagram

When a new input token x_t is fed into the incremental processor, it produces an output label y_t. Then, the controller takes x_t, the hidden state of the incremental processor h_t, and the input-output representation in the memory Γ^p to compute action a_t using the revision policy. The action to be selected also depends on the threshold τ such that the REVISE action is chosen if the policy value is greater than or equal to τ. Otherwise, the WRITE action is chosen. If a REVISE action is selected, the input buffer containing the partial input thus far will be passed to the reviser to recompute the output labels $\overline{y}_1, ..., \overline{y}_{t-1}, \overline{y}_t$. Simultaneously, the projected output vector z and input-output representation φ in the caches will also be recomputed. Otherwise, if the WRITE action is selected, y_t is added to the output buffer. The caches Γ^z, Γ^p, and Γ^h are updated for the current time step, and the algorithm proceeds to process the next token.

TAPIR is trained in a two-step process where the reviser is first trained independently using cross entropy loss. Subsequently, the incremental processor and the controller are trained together with a

Table 1: Sample Utterance from the DemoNSF Dataset with Slot Annotation in IOB Format

Utterance	i	would	like	to	leave	from	stansted	airport	after	11:45	.
Slot Label	O	O	O	O	O	O	B-depart	I-depart	O	B-leave	O

Table 2: 5 Types of Noise in the DemoNSF Dataset

Noise Type	**Utterance**
Clean	i would like to leave from stansted airport after 11:45 .
Verbose	looking to leave from stansted airport after quarter of twelve because i have a presentation at work that morning
Paraphrase	departing from stansted airport after 11:45
Simplification	stansted airport after 11:45
Typos	leave stansted air aftr 11:45
Speech	so i would like to leave from stanstead airport after eleven forty five

combined loss. The controller requires the training set to have supervision signals in the form of WRITE/REVISE action sequences for policy learning. The action sequences are generated using a linear transformer trained with causal masking to simulate a recurrence mechanism. The trained linear transformer is then deployed on the same dataset without masks in a restart-incremental setting to collect the outputs for partial prefixes. These will be used to derive the action sequences by comparing the partial outputs at time step $t-1$ with that of the current time step t. If there are differences with the partial outputs, excluding y_t, a REVISE action is appended to the sequence. Otherwise, a WRITE action is added.

4 Method

TAPIR was evaluated on the noisy DemoNSF dataset to determine its non-incremental performance, incremental performance, and incremental inference speed. The application of a delay strategy was also explored. The results are compared to those of a reference restart-incremental transformer and the non-incremental results of DemoNSF as reported by Dong et al. (2023).

4.1 Dataset

The training set of DemoNSF, based on MultiWOZ (Budzianowski et al., 2018), consists of 56,117 utterances across 4 domains (i.e., attraction, hotel, restaurant, and train). The validation set has 5,000 utterances. The clean and perturbed test sets were taken from RADDLE (Peng et al., 2021), and annotated for the SF task using the IOB tagging format. RADDLE is a crowd-source noise robustness evaluation benchmark for dialogue systems. Table 3 shows the number of utterances per type of test set. There are 27 slot labels, some examples of which are *area*, *type*, *name*, *price*, and *day*.

Table 3: Number of Utterances in the DemoNSF Dataset

Dataset		**Number of utterances**
Training		56,117
Validation		5,000
Test	Clean	306
	Verbose	306
	Paraphrase	298
	Simplification	307
	Typos	301
	Speech	298

4.2 Experiments

There are 5 major steps in the implementation of the TAPIR model: (1) Train the action generator – linear transformer with causal masking – on the train and validation sets; (2) Generate the action sequences for the train and validation sets using the trained action generator; (3) Train the transformer reviser; (4) Train the two-pass configuration (i.e., the combination of the incremental processor, controller, and the transformer reviser); and (5) Evaluate the model on the clean and perturbed test sets.

Hyperparameter tuning was performed for the transformer reviser and the two-pass configuration model using Optuna. Due to resource constraints, the number of search trials was limited to 10 and 5 for the transformer reviser and the two-pass con-

figuration model, respectively. The obtained hyperparameters are shown in Tables 4 and 5. The hyperparameters for the transformer reviser was also applied for the reference restart-incremental model. The search space and other training parameters were adopted from the reference work by Kahardipraja et al. (2023).

Table 4: Transformer Reviser and Reference Model Hyperparameters

Hyperparameter	**Value**
Layers	3
Gradient Clipping	-1
Learning Rate	7e-05
Batch size	16
Feed-forward dimension	1024
Self-attention dimension	512

Table 5: Two-pass Configuration Hyperparameters

Hyperparameter	**Value**
LSTM Layers	4
Controller Layers	2
Gradient Clipping	1.0
Learning Rate	7e-05
Batch Size	16
LSTM Hidden Dimension	512
Controller Hidden Dimension	512
Memory Size	5

A delay with a look-ahead window of size 1 and 2 was applied in the training and inference of TAPIR to investigate whether the availability of the right context can lead to better performance. This means that the slot label for input x_t is outputted at time step $t + d$, where d denotes the delay. For the reference restart-incremental transformer, the delay was only incorporated in the inference.

4.3 Evaluation

The non-incremental and incremental performance of TAPIR were compared with a reference model, which is a Transformer encoder applied in a restart-incremental interface. This performs revisions at every time step due to recomputations. Additionally, the non-incremental performance is compared to the DemoNSF model, a generative framework that performs multilevel data augmentation to create a noisy candidate pool (Dong et al., 2023). This is then used in the three noisy auxiliary pre-training tasks (noise recovery, random mask, and hybrid discrimination) to learn the semantic structural information of noises in different levels. It also incorporates demonstrations in the generative model. The demonstrations are retrieved from the top k most similar utterances to the input from the noisy candidate pool.

The non-incremental performance of the models for the SF task, measured using the F1 score, reveals their ability to arrive at a correct final output. The incremental performance demonstrates the ability of the models to generate correct and stable partial outputs and to recover from wrong outputs timely. This is measured based on three metrics: edit overhead (EO), correction time score (CT), and relative correctness (RC) whose values range from 0 to 1 (Madureira and Schlangen, 2020). EO is the proportion of unnecessary edits, where a value closer to 0 indicates fewer edits made. CT is the average proportion of time steps needed before a final decision is reached, where a value closer to 0 denotes sooner final decisions. RC is the proportion of output prefixes that match the final output, where a value closer to 1 indicates the ability of the system to generate correct prefixes of the final output. It must be noted that RC is evaluated based on the final non-incremental output, instead of the gold standard to focus the evaluation on the incremental performance of the models. The incremental inference speed was also measured to determine if the models are computationally efficient at inference.

5 Results

Aside from presenting the non-incremental performance, incremental performance, and incremental inference speed obtained by TAPIR on the DemoNSF, a qualitative analysis of how TAPIR performs incremental slot filling is provided.

5.1 Non-incremental Performance

The non-incremental performance results of the models across the different test sets are shown in Table 6.

5.1.1 DemoNSF vs. Incremental SF Models

The performance of the models on the clean test set are relatively similar. However, the performance gap between DemoNSF and the incremen-

Table 6: Non-incremental Performance of DemoNSF, Reference, and TAPIR models

Test Set		DemoNSF	Reference	TAPIR		
				Delay 0	Delay 1	Delay 2
Clean		95.72	95.24	95.34	94.81	94.89
	Verbose	82.37	79.55	77.94	75.39	76.16
Sentence-level	Paraphrase	89.98	88.6	86.09	85.2	88.05
	Simplification	89.49	81.77	82.45	84.32	82.39
Character-level	Typos	76.63	66.2	60.43	62.69	62.42
Word-level	Speech	87.55	74.73	71.72	72.07	70.84

tal models (i.e., reference and TAPIR) becomes more pronounced on the noisy test sets. This is expected because the incremental models were not exposed to input perturbations during training unlike DemoNSF which is a noisy SF framework. This highlights the need to adapt the training of incremental systems to improve their robustness against noisy inputs, which are prevalent in real dialogue systems.

5.1.2 Reference Model vs. TAPIR

TAPIR achieves comparable performance with the restart-incremental transformer even with fewer recomputations, demonstrating the effectiveness of the revision policy on the clean test set. TAPIR also achieved a higher F1 score on the simplification test set. This may be attributed to the shorter available context, which can make transformers less effective. The LSTM component (i.e., first-pass model) of TAPIR is well-suited for handling simplified utterances because it can effectively use the available left context for prediction without relying on long-range dependencies.

The reference model outperformed TAPIR, with differences ranging from 1.61% to 5.77%, on the noisy test sets excluding simplification. This may be because the reference model is deployed in a restart-incremental fashion, which enables it to perform revisions as new input token arrives. Hence, this reveals the weakness of the current revision policy employed in TAPIR in noisy scenarios.

5.1.3 Delay Strategy

A delay of 1 is the most effective in improving the performance of TAPIR on the typos, speech, and simplification test sets. These datasets are characterized by syntax errors and lack of context. Hence, the left token is effective in disambiguating the noisy inputs. TAPIR achieved higher performance on the paraphrase test set using a delay of 2. This indicates that the availability of a longer context aids in disambiguating the rich and varied syntax in paraphrased text data. The delay strategy was not effective in improving the performance of TAPIR on the clean and verbose test sets, indicating that the addition of a delay may impair the model's ability to learn the relationship between the input and the delayed output. These results show that the effectiveness of the delay strategy and the look-ahead window size depends on the type of data to be processed. A longer delay does not necessarily lead to better performance.

The performance of the incremental models on the different noisy test sets are ranked, from best to worst, as follows: (1) clean, (2) paraphrase, (3) simplification, (4) verbose, (5) speech, and (6) typos. This shows that incremental models are more sensitive to fine-grained noises, with character-level noise (i.e., typos) negatively affecting its performance the most, followed by word-level noise (i.e., speech). This is because incremental models process sequences one token at a time, thus fine-grained noises have a significant impact due to the lack of access to the full context.

5.2 Incremental Performance

The incremental performance of the reference model and TAPIR are shown in Figure 2. For no delay, TAPIR evidently outperformed the reference model across the three incremental metrics (i.e., CT, EO, and RC). This implies that it is better at producing stable outputs that are correct prefixes of the final non-incremental output. Applying the delay strategy generally reduces the EO and CT with minimal improvement on RC for both models. However, it can be observed that their incremental performances are worse on the noisy test sets

Table 7: Comparison of Incremental Inference Speed (in sequences/sec.) Between the Reference Model and TAPIR

Test Set		Reference	TAPIR
Clean		3.05	11.87 (3.89x)
	Verbose	2.18	8.76 (4.02x)
Sentence-level	Paraphrase	3.32	13.58 (4.09x)
	Simplification	5.32	19.43 (3.65x)
Character-level	Typos	3.51	12.84 (3.66x)
Word-level	Speech	3.11	11.97 (3.85x)
Average		3.42	**13.07 (3.82x)**

compared to those on the clean test set. The graph clearly shows that the incremental performance degrades most significantly on the typos test set, illustrating that incremental systems are highly affected by character-level input perturbations.

5.3 Incremental Inference Speed

Table 7 compares the incremental inference speed between the reference model and TAPIR. TAPIR has significantly faster inference speed, being able to process 3.82x sequences per second compared to the reference model. This confirms that using transformers in a restart-incremental manner for incremental NLU is computationally costly due to the unnecessary recomputations of the entire partial output at every time step.

5.4 Qualitative Analysis

Figure 3 illustrates two examples of how TAPIR performs incremental slot filling—one where it performed a correct revision and another where it made a mistake due to an input perturbation. In the example on top with the input sequence "*stansted airport after 11:45*", the model mistakenly revised the slot label of the token "*airport*" from "*I-depart*" to "*I-dest*" when it encountered the new input token "*after*" at $t = 3$. At $t = 4$, the controller was able to identify the output inconsistency where the "*I-dest*" is preceded by "*B-depart*" when it should be "*B-dest*". Hence, it emitted a REVISE action to correct the slot label of "*airport*" back to "*I-depart*".

In the second example, the model was able to generate the correct input prefixes up to $t = 4$, which was when a typo "*aftr*" arrived in the input sequence. However, upon the arrival of the final token, the controller mistakenly revised the correct prefixes "*B-depart*" and "*I-depart*" into "*O*", which may be attributed to the inclusion of the typo "*aftr*" in the history of inputs used for the computation of the next action.

6 Discussion

Results revealed that TAPIR outperforms the more naive restart-incremental model in terms of non-incremental performance, incremental performance, and incremental inference speed. However, TAPIR experiences performance degradation when dealing with noisy input data. From our findings, three key considerations emerge for the development of incremental NLU systems:

Robustness to Noise. Despite its sophisticated architecture, TAPIR experiences notable performance degradation when processing noisy input data. It was observed that fine-grained noises at the character-level (e.g. typos) and word-level (e.g. speech errors) cause more significant performance losses. This sensitivity arises because incremental systems process input per token, leading to a higher impact of noise due to the absence of the full context. These findings emphasize the need to incorporate robust noise-handling mechanisms in incremental NLU systems to achieve reliable performance in real-world scenarios where noise is unavoidable.

Revision Policy. The ability to revise is crucial in incremental NLU tasks to resolve misinterpretations that occur due to the inherent ambiguity of partial utterances. The adaptive revision policy of TAPIR is key to its significantly faster inference speed compared to a restart-incremental model by avoiding unnecessary recomputations. It was also proven to be effective on clean and simplified test sets. However, TAPIR falls behind the restart-incremental model on the noisy test sets, revealing its weakness under more challenging scenarios. This underscores the need for further refinement of the adaptive revision policy without incurring significant computational cost.

Delay strategy. Delaying the output generally results in better performance by providing the incremental model with additional context. The results showed that a delay of 1 is effective in improving the non-incremental performance on test sets characterized by syntax errors and limited context, such as typos and speech errors. A delay of 2 improves performance on paraphrased inputs by providing a longer context that can help disambiguate syntactically varied sequences. However, it is worth noting that the delay strategy was not effective on clean

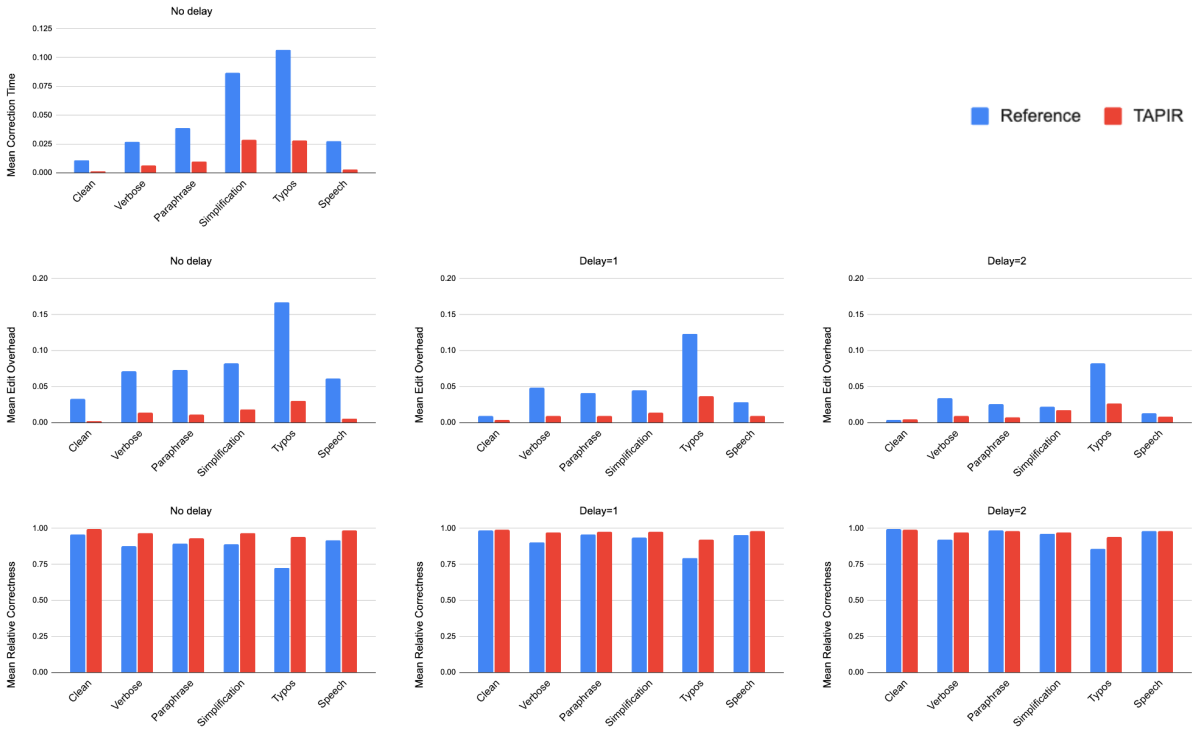

Figure 2: Incremental Performance of the Reference Model and TAPIR

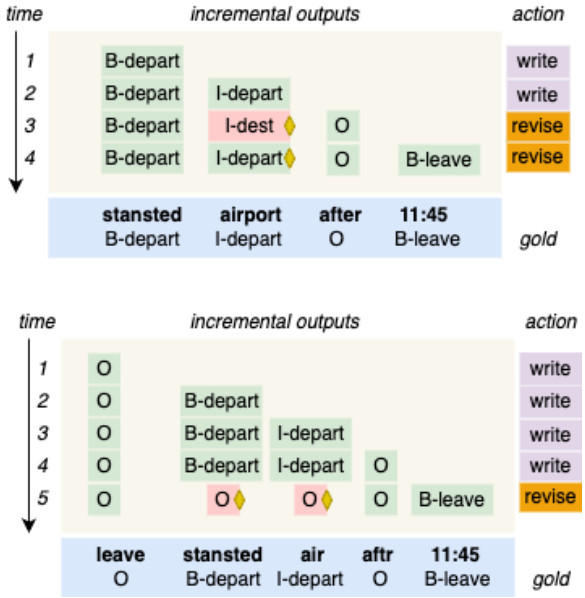

Figure 3: Examples of Incremental Inference Using TAPIR

and verbose test sets, suggesting that inappropriate delay settings can impair the model's ability to learn correct input-output relationships. These findings demonstrate that longer delays do not necessarily lead to better performance. Therefore, the delay strategy must be tailored to the specific nature of the input data to achieve siginificant performance improvement.

7 Conclusion

We evaluated the robustness of TAPIR in noisy slot filling task and assessed the impact of input perturbations to the performance of incremental NLU systems. Results showed that TAPIR lags behind the reference restart-incremental transformer on noisy test sets, which reveal the weakness of its adaptive revision policy on more challenging scenarios. It was also observed that character-level and word-level noises cause larger performance losses, demonstrating the sensitivity of incremental NLU models to fine-grained noise due to the absence of a global context. Employing a delay strategy can improve non-incremental and incremental performance. However, the optimal size of the look-ahead window depends on the nature of the input data. Future work can focus on developing robust noise-handling mechanisms for incremental NLU systems. Further research must also be conducted on improving the revision policies that can effectively balance the frequency and accuracy of revisions. Furthermore, researchers can look into optimizing delay strategies to improve the performance of incremental NLU systems.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Stefan Constantin, Jan Niehues, and Alex Waibel. 2019. Incremental processing of noisy user utterances in the spoken language understanding task. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 265–274, Hong Kong, China. Association for Computational Linguistics.

Guanting Dong, Tingfeng Hui, Zhuoma GongQue, Jinxu Zhao, Daichi Guo, Gang Zhao, Keqing He, and Weiran Xu. 2023. DemoNSF: A multi-task demonstration-based generative framework for noisy slot filling task. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10506–10518, Singapore. 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.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Bing Liu and Ian Lane. 2016. Joint online spoken language understanding and language modeling with recurrent neural networks. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 22–30, Los Angeles. 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.

Baolin Peng, Chunyuan Li, Zhu Zhang, Chenguang Zhu, Jinchao Li, and Jianfeng Gao. 2021. RADDLE: An evaluation benchmark and analysis platform for robust task-oriented dialog systems. 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 4418–4429, Online. Association for Computational Linguistics.