Towards Incremental Transformers: An Empirical Analysis of Transformer Models for Incremental NLU

Patrick Kahardipraja　　　**Brielen Madureira**　　　**David Schlangen**
Computational Linguistics, Department of Linguistics
University of Potsdam, Germany
`{kahardipraja,madureiralasota,david.schlangen}@uni-potsdam.de`

Abstract

Incremental processing allows interactive systems to respond based on partial inputs, which is a desirable property *e.g.* in dialogue agents. The currently popular Transformer architecture inherently processes sequences as a whole, abstracting away the notion of time. Recent work attempts to apply Transformers incrementally via *restart-incrementality* by repeatedly feeding, to an unchanged model, increasingly longer input prefixes to produce partial outputs. However, this approach is computationally costly and does not scale efficiently for long sequences. In parallel, we witness efforts to make Transformers more efficient, *e.g.* the Linear Transformer (LT) with a recurrence mechanism. In this work, we examine the feasibility of LT for incremental NLU in English. Our results show that the recurrent LT model has better incremental performance and faster inference speed compared to the standard Transformer and LT with restart-incrementality, at the cost of part of the non-incremental (full sequence) quality. We show that the performance drop can be mitigated by training the model to wait for right context before committing to an output and that training with input prefixes is beneficial for delivering correct partial outputs.

1 Introduction

One fundamental property of human language processing is *incrementality* (Keller, 2010). Humans process language on a word-by-word basis by maintaining a partial representation of the sentence meaning at a fast pace and with great accuracy (Marslen-Wilson, 1973). The garden path effect, for example, shows that language comprehension is approximated incrementally before committing to a careful syntactic analysis (Frazier and Rayner, 1982; Altmann and Steedman, 1988; Trueswell et al., 1994, *inter alia*).

The notion of order along the time axis during computation is a key aspect of incremental processing and thus a desirable property both of cognitively plausible language encoders as well as in applications such as interactive systems (Skantze and Schlangen, 2009). RNNs, for example, are inherently able to process words sequentially while updating a recurrent state representation. However, the Transformer architecture (Vaswani et al., 2017), which has brought significant improvements on several NLP tasks, processes the input sequence as a whole, thus prioritising parallelisation to the detriment of the notion of linear order.

One way to employ non-incremental models in incremental settings is resorting to an incremental interface, like in Beuck et al. (2011), where a complete recomputation of the available partial input happens at each time step to deliver partial output. Madureira and Schlangen (2020) examined the output stability of non-incremental encoders in this *restart-incremental* fashion. While qualitatively feasible, this procedure is computationally costly, especially for long sequences, since it requires as many forward passes as the number of input tokens.

In parallel, there is ongoing research on ways to make Transformers more efficient, *e.g.* the Linear Transformer (LT) introduced by Katharopoulos et al. (2020). Besides being more efficient, LTs can be employed with a recurrence mechanism based on causal masking that turns them into models similar to RNNs. In this work, we examine the suitability of using LTs in incremental processing for sequence tagging and classification in English. We also inspect the use of the delay strategy (Baumann et al., 2011; Oda et al., 2015; Ma et al., 2019) to examine the effect of *right context* availability on the model's incremental performance. Our hypothesis is that recurrence will allow LTs to be better in incremental processing as it captures sequence order. As LTs use an approximation of softmax attention, we also expect a performance drop compared to the standard Transformer while being faster in the incremental setting due to its linear time attention.

2 Related Work

In recent years, neural approaches, including Transformer-based architectures (Vaswani et al., 2017), have become more popular for incremental processing. Given that Transformer models are not inherently incremental, employing them for incremental processing demands adaptation.

In simultaneous translation, for instance, Ma et al. (2019) proposed to use an incremental encoder by limiting each source word to attend to its predecessors and recompute the representation for previous source words when there is new input. Zhang et al. (2021) introduced an average embedding layer to avoid recalculation when using an incremental encoder, while exploiting *right context* through knowledge distillation. An investigation of the use of non-incremental encoders for incremental NLU in interactive systems was conducted by Madureira and Schlangen (2020). The authors employed BERT (Devlin et al., 2019) for sequence tagging and classification using restart-incrementality, a procedure with high computational cost.

The computational cost of a restart-incremental Transformer can be reduced with more efficient models or even avoided if an inherently incremental Transformer architecture existed. Recent works have proposed modifications that could help achieve that. For instance, by approximating the softmax attention with a recurrent state (Katharopoulos et al., 2020; Choromanski et al., 2021; Peng et al., 2021). The Linear Transformer model (Katharopoulos et al., 2020, LT henceforth) can be viewed as an RNN when the attention is causal (see also, very recently, Kasai et al., 2021).

3 Methods

3.1 Overview of the Linear Transformer

In LTs, the similarity score between a query and a key for the i-th position is computed using a kernel function. The causal attention can be written as:

$$\text{Att}_i(Q, K, V) = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i} \quad (1)$$

$$S_i = \sum_{j=1}^{i} \phi(K_j) V_j^T; Z_i = \sum_{j=1}^{i} \phi(K_j) \quad (2)$$

with feature map $\phi(x) = \text{elu}(x) + 1$ where $\text{elu}(\cdot)$ denotes the exponential linear unit (Clevert et al., 2016). Hence, S_i and Z_i can be viewed as a recur-

rent state:

$$S_i = S_{i-1} + \phi(K_i) V_i^T \quad (3)$$
$$Z_i = Z_{i-1} + \phi(K_i) \quad (4)$$

with $S_0 = Z_0 = 0$. As an RNN, the run-time complexity is linear with respect to the sequence length and constant for each added token, which promises faster inference compared to the restart-incremental approach.

3.2 Models

We examine the behaviour of Transformer models used as incremental processors on token level, in five configurations (Table 1):

1. **Baseline**: the standard Transformer encoder incrementalised via restart-incrementality, trained with access to full sequences.

2. **LT**: the LT encoder incrementalised via restart-incrementality, trained with access to full sequences.

3. **LT+R**: the LT encoder trained as in (2) but during test time we use its recurrent state vector to predict the label at each time step, as in an RNN.

4. **LT+R+CM**: the LT encoder trained with causal masking to ensure each token representation can only attend to previous tokens. During inference, we convert the model to an RNN as in (3). Training with input prefixes aims at encouraging the learning of intermediate structures (Köhn and Menzel, 2014) and the anticipation of future output (Ma et al., 2019).

5. **LT+R+CM+D**: similar to (4), but, during training, the output for the input token x_t is obtained at time $t + d$, where $d \in \{1, 2\}$ is the delay, following the approach in Turek et al. (2020). There is evidence that additional *right context* improve the models' incremental performance (Baumann et al., 2011; Ma et al., 2019; Madureira and Schlangen, 2020), which results in a trade-off between providing timely output or waiting for more context to deliver more stable output.

We also delay the output by 1 and 2 time steps for the baseline and LT following Madureira and Schlangen (2020), to provide a fair comparison on incremental metrics. Note that outputs from both (1) and (2) are non-monotonic, as labels can be

reassigned when a new input token is observed. The other models deliver monotonic output for sequence tagging as RNNs. A slight modification is needed for sequence classification as each sequence is mapped to a single label. We average the hidden representation at the last layer and project it linearly, followed by a softmax to obtain the sequence label $\hat{y}_t$ based on the consumed input until time t. For LT+ models, we use incremental averaging to avoid recomputation. By doing this, sequence classification is performed similarly for all models.

Models	Restart Incremental	Recurrence	Causal Masking	Delay
Baseline	✓	-	-	-*
LT	✓	-	-	-*
LT+R	-	✓	-	-
LT+R+CM	-	✓	✓	-
LT+R+CM+D	-	✓	✓	✓

Table 1: Overview of the Transformer models. * means we perform further comparisons with a delayed variant.

4 Experimental Setup

4.1 Datasets

We evaluate our models on 9 datasets in English, which were also used in Madureira and Schlangen (2020). The tasks consist of sequence tagging: slot filling (ATIS, Hemphill et al. (1990); Dahl et al. (1994) and SNIPS, Coucke et al. (2018)), chunking (CoNLL-2000, Tjong Kim Sang and Buchholz (2000)), NER and PoS tagging (OntoNotes 5.0, WSJ section, Weischedel et al. (2013)); and sequence classification: intent detection (ATIS and SNIPS) and sentiment classification (positive/negative, Kotzias et al. (2015) and pros/cons, Ganapathibhotla and Liu (2008)). More details are available in the Appendix.

4.2 Evaluation

The overall performance of the models is measured with accuracy and F1 Score, according to the task. For the incremental evaluation, we report the diachronic metrics proposed by Baumann et al. (2011) and adapted in Madureira and Schlangen (2020): *edit overhead* (EO, the proportion of unnecessary edits over all edits), *correction time score* (CT, the average proportion of time steps necessary to reach a final decision), and *relative correctness* (RC, the proportion of time steps in which the output is a correct prefix of the final, non-incremental output).

To focus on the incremental quality of the models and allow a clear separation between incremen-

tal and non-incremental evaluation, we follow the approach by Baumann et al. (2011) and Madureira and Schlangen (2020), evaluating incremental outputs with respect to the final output produced by the models. While the final output may differ from the gold standard, it serves as the target for the incremental output, as the non-incremental performance is an upper bound for incremental processing (Baumann et al., 2011).

4.3 Implementation

We re-implement the Transformer and use the original implementation of the LT.[1] All models are trained to minimise cross-entropy with the AdamW optimiser (Loshchilov and Hutter, 2019). We use 300-D GloVe embeddings (Pennington et al., 2014) which are passed through a linear projection layer with size d_{model}. All experiments were performed on a GPU GeForce GTX 1080 Ti. Details on the implementation, hyperparameters and reproducibility are available in the Appendix. Our implementation is publicly available.[2]

5 Results and Discussion

Tasks	Baseline	LT	LT+R	LT+R +CM	LT+R+ CM+D1	LT+R+ CM+D2
ATIS-Slot	**94.51**	93.67	86.84	93.78	94.38	93.54
SNIPS-Slot	**90.13**	87.98	63.16	81.88	85.72	86.91
Chunk	**91.27**	88.42	67.54	86.63	89.42	89.33
NER	**89.55**	86.13	52.04	69.09	81.39	85.55
PoS Tagging	**96.88**	96.49	89.30	95.11	96.49	96.55
ATIS-Intent	**97.20**	97.09	95.63	95.74	96.53	96.53
SNIPS-Intent	**97.14**	**97.14**	83.71	96.43	**97.14**	96.86
Pos/Neg	**86.00**	85.17	68.00	80.33	81.16	82.67
Pros/Cons	94.42	94.21	90.97	94.37	**94.56**	94.46

Table 2: Non-incremental performance of our models on test sets (first group, F1, second group, accuracy). Here, the baseline performs generally better than the LT variants.

Ultimately, the quality of the output when all input has been seen matters; hence, we first look at the *non-incremental* or full-sequence performance, in Table 2. We can see that LTs do not outperform the baseline here,[3] although they have the advantage of being faster (Table 3). We see two possible reasons for this: First, as the LT variants strictly go left-to-right through the sequence, they have less information for each token to make their decision. This can be alleviated by allowing LT+R+CM to

---

[1]https://linear-transformers.com
[2]https://github.com/pkhdipraja/towards-incremental-transformers
[3]Notice that the baseline differs from Madureira and Schlangen (2020) who used a pretrained BERT model.

wait for 1 or 2 tokens before producing partial output, and indeed we see an overall performance increase of 0.1% - 16.5% for the +D variants. Second, we suspect that the chosen feature map in LTs to approximate the softmax attention is sub-optimal, and a further gating mechanism could yield a better performance (Peng et al., 2021).

Comparing LT+R+CM against LT+R, we observe that training on prefixes yields a better result during test time, as LT+R+CM may learn anticipation as a by-product, in line with the work of Ma et al. (2019). LT+R+CM+D performs competitively with LT, outperforming the latter in 4 out of 9 datasets. This is likely due to the better capability of the delayed network in modelling both non-linear and acausal functions that appear in some of the tasks (Turek et al., 2020).

Figure 1 depicts the incremental metrics of all models. The EO and CT score for sequence tagging is low for all models, which indicates that the models are capable, in general, of producing stable and accurate partial outputs. Notice that the LT+ models are not able to revise the output in sequence tagging. For sequence classification, the models have higher EO and CT score due to the fact that the label is a function of the whole sequence and the model might be unable to reach an early decision without enough right context.

LT+R+CM performs better in incremental metrics compared to the baseline and LT in sequence classification. This is evidence that the notion of order is important for incremental processing, as the recurrent state in the LT allows partial representation updates along the time axis when processing partial input. Here, sequence classification is treated in a similar fashion for all models, by using the average of the hidden representation in the last layer.

All the models have high RC score in general for both sequence tagging and classification. This means that most of the partial outputs are a correct prefix of the final ("non-incremental") output and could fruitfully be used as input to subsequent processors in an incremental pipeline. For RC, LT+R+CM also outperforms both the baseline and LT in all tasks. A delay of 1 or 2 tokens before committing to an output also helps to improve the incremental performance across all models. In terms of incremental inference speed, we see that the recurrent mode is more than 10 times faster compared to using restart-incrementality (Table 3).

To understand the models' behaviour better, especially pertaining to their potential for real-time applications, we also examine their incremental inference speed for different sequence lengths as shown in Figure 2. As expected, LT+R+CM scales linearly and outperforms the baseline and LT considerably as the sequence becomes longer. The run-time performance of LT is slightly better than the baseline because of its linear time attention, however it is still slower compared to LT+R+CM as it is restart-incremental.
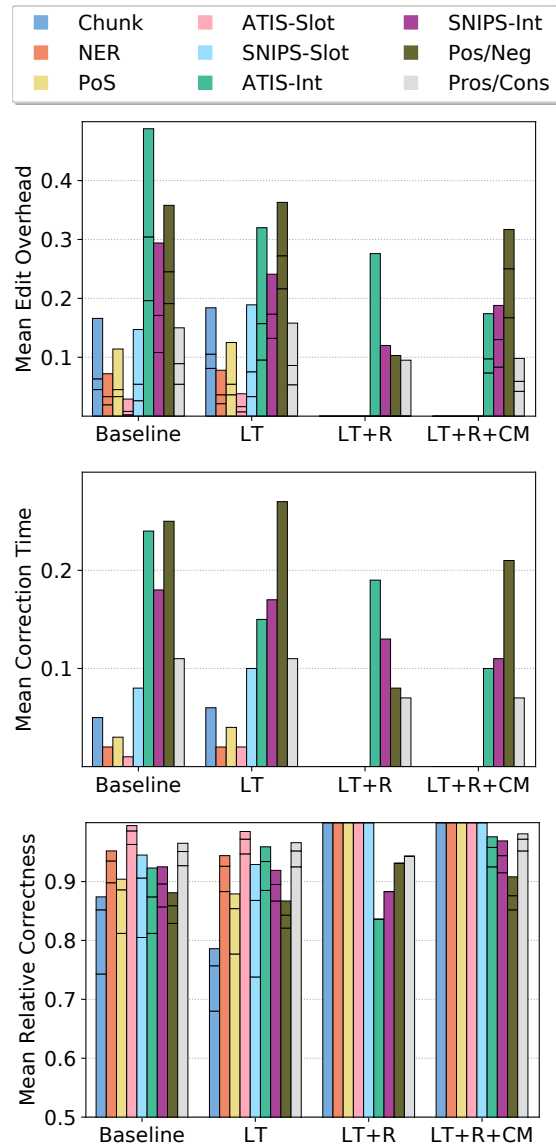

Figure 1: Incremental evaluation on the test sets. EO, CT and RC ∈ [0, 1], y-axes are clipped to improve readability. Lower is better for EO and CT, higher for RC. For EO, the lines on the bars refer to original, delay=1 and delay=2, from top to bottom, and vice versa for RC, showing that delay improves the results. LT+R+CM performs better compared to the baseline and LT.

Tasks	Baseline	LT	LT+R+CM
ATIS-Slot	0.983	1.025	13.780
SNIPS-Slot	1.021	1.137	14.957
Chunk	0.393	0.448	6.023
NER	0.382	0.436	5.745
PoS Tagging	0.383	0.435	5.831
ATIS-Intent	0.883	1.005	13.310
SNIPS-Intent	0.995	1.129	14.907
Pos/Neg	0.725	0.767	9.962
Pros/Cons	1.073	1.228	14.979
Average	0.76 (1×)	0.85 (1.12×)	**11.06 (14.55×)**

Table 3: Comparison of incremental inference speed on test sets, measured in sequences/sec. All the models have similar size with 4 layers, feed-forward dimension of 2048 and self-attention dimension of 512.
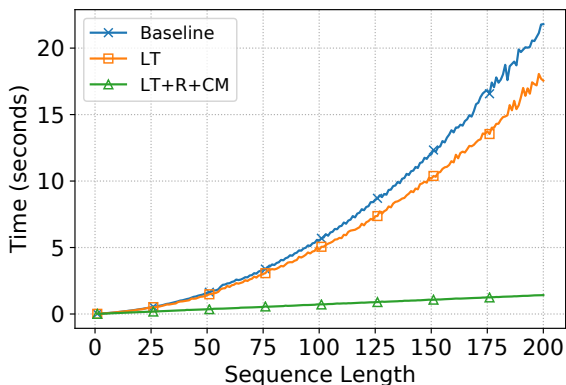

Figure 2: Incremental inference speed of models from Table 3 with increasing sequence length. LT+R+CM scales linearly with sequence length unlike the baseline and LT. Note that the incremental inference speed of LT+R+CM is similar to LT+R.

5.1 Ablations

We examine the importance of word and positional embeddings on the baseline and LT+R+CM for non-incremental metrics (Table 4). We find that using pre-trained GloVe (Pennington et al., 2014) embeddings is beneficial for the models' performance. On average, it contributes 2.74 accuracy and 5.16 F1 for the baseline, while improving LT+R+CM by 1.6 and 2.55 points for accuracy and F1. On the other hand, we observe that positional embeddings play a less significant role in LT+R+CM compared to the baseline. Without them the performance, on average, for LT+R+CM improves in accuracy by 0.15 and the F1 score degrades by 1.35. The baseline, however, experiences degradation in performance by 1.79 and 18.46 points for accuracy and F1, on average. The recurrence mechanism may be a reason for the effect of positional embeddings being less pronounced in LT+R+CM.

	Tasks	Score	– GloVe	– Pos	– Glove & Pos
	Baseline				
F1	ATIS-Slot	94.51	-3.02	-17.99	-20.44
	SNIPS-Slot	90.13	-6.74	-16.39	-21.47
	Chunk	91.27	-5.28	-18.19	-19.36
	NER	89.55	-5.61	-21.27	-26.07
Accuracy	PoS Tagging	96.88	-0.79	-3.31	-4.15
	ATIS-Intent	97.20	-2.35	-2.69	-3.81
	SNIPS-Intent	97.14	-0.57	+0.72	-0.43
	Pos/Neg	86.00	-8.83	-3.83	-12.00
	Pros/Cons	94.42	-1.18	+0.15	-1.64
	LT+R+CM				
F1	ATIS-Slot	93.78	-0.42	-0.25	-1.37
	SNIPS-Slot	81.88	-2.44	-0.55	-2.79
	Chunk	86.63	-3.82	-3.35	-7.30
	NER	69.09	-3.52	-1.24	-6.69
Accuracy	PoS Tagging	95.11	-0.74	-0.79	-1.58
	ATIS-Intent	95.74	-0.89	-0.11	-0.67
	SNIPS-Intent	96.43	+0.14	0.00	+0.28
	Pos/Neg	80.33	-6.00	+1.17	-9.00
	Pros/Cons	94.37	-0.53	+0.46	-0.83

Table 4: Ablation of GloVe and positional embeddings on the baseline and LT+R+CM for non-incremental metrics.

6 Conclusion

We studied the use of Transformer encoders for incremental processing and concluded that it is possible to deploy them as incremental processors with certain trade-offs. With recurrent computation, the Linear Transformer (LT) has inferior non-incremental performance compared to the regular Transformer and the LT with restart-incrementality. However, it has the great advantage of being much more efficient for incremental processing, since recomputation at each time step is avoided. The output of the recurrent LT is generally more stable for sequence classification and monotonic for tagging. Its non-incremental performance drop can be mitigated by introducing delay, which also improves the incremental metrics. It is also beneficial to train such model with input prefixes, allowing it to learn more robust predictions.

Acknowledgements

References

Gerry Altmann and Mark Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238.

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

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011. 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).

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Vienna, Austria, May 3-7, 2021*.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471, Florence, Italy. Association for Computational Linguistics.

Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14(2):178–210.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 241–248, Manchester, UK. Coling 2008 Organizing Committee.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. Finetuning pretrained transformers into RNNs. *arXiv preprint*, arXiv:2103.13076.

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*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Frank Keller. 2010. Cognitively plausible models of human language processing. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 60–67, Uppsala, Sweden. 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.

Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 597–606, New York, NY, USA. Association for Computing Machinery.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. 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.

William Marslen-Wilson. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–3.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 198–207, Beijing, China. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Vienna, Austria, May 3-7, 2021*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 745–753, Athens, Greece. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

John C. Trueswell, Michael K. Tanenhaus, and Susan M. Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language*, 33(3):285–318.

Javier Turek, Shailee Jain, Vy Vo, Mihai Capotă, Alexander Huth, and Theodore Willke. 2020. Approximating stacked and bidirectional recurrent architectures with the delayed recurrent neural network. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9648–9658. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0 ldc2013t19.

Shaolei Zhang, Yang Feng, and Liangyou Li. 2021. Future-guided incremental transformer for simultaneous translation. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Lukáš Žilka and Filip Jurčíček. 2015. Lectrack: Incremental dialog state tracking with long short-term memory networks. In *Text, Speech, and Dialogue*, pages 174–182, Cham. Springer International Publishing.

A Reproducibility

We describe in more detail the hyperparameters and implementation of our models.

Data

We mostly follow Madureira and Schlangen (2020). We use only the WSJ section of OntoNotes with splits following Pradhan et al. (2013). For Pos/Neg and Pros/Cons datasets, we split them randomly with a proportion of 70% train, 10% validation, and 20% test set due to the unavailability of an official splitting scheme. We removed sentences longer than 200 words as they were infeasible to compute. We use the preprocessed data and splits for SNIPS and ATIS made available by E et al. (2019).

Training details

Our models are trained for 50 epochs, using early stopping with patience of 10 and dropout of 0.1. For AdamW (Loshchilov and Hutter, 2019), we use $\beta_1 = 0.9$ and $\beta_2 = 0.98$. The learning rate is increased linearly for the first 5 epochs. After 30, 40, and 45 epochs, we decay the learning rate by 0.5. Xavier initialisation (Glorot and Bengio, 2010) is applied to all parameters. The number of attention heads is set to 8, where the dimension of each head is self-attention dimension $d/8$.

We also apply label smoothing (Szegedy et al., 2016) with $\epsilon = 0.1$ for sequence classification to make the model more robust for incremental processing. For OOV words, we randomly replace tokens by "UNK" token with $p = 0.02$ during training and use it for testing (Žilka and Jurčíček, 2015). We perform hyperparameter search using Comet's Bayesian search algorithm [4], maximising F1 score for sequence tagging and accuracy for sequence classification on the validation set. The hyperparameter search trials are limited to 20 for all of our experiments. The hyperparameters for LT were also used for LT+R. We use similar hyperparameters for LT+R+CM and LT+R+CM+D. We set the seed to 42119392 for all of our experiments.

Hyperparameters	
Layers	1, 2, 3, 4
Gradient clipping	no clip, 0.5, 1
Learning rate	$5e^{-5}, 7e^{-5}, 1e^{-4}$
Batch size	32, 64, 128
Feed-forward dimension	1024, 2048
Self-attention dimension	256, 512

Table 5: Hyperparameter search space. We use the same search space for all of our models.

Tasks	Average sequence length
ATIS-Slot	10.26
SNIPS-Slot	9.08
Chunk	23.55
NER	24.14
PoS Tagging	24.14
ATIS-Intent	10.26
SNIPS-Intent	9.08
Pos/Neg	13.95
Pros/Cons	8.99

Table 6: Average sequence length on test sets for each task.

---

[4] https://www.comet.ml/docs/python-sdk/introduction-optimizer/

Tasks	Layers	Gradient clip	Learning rate	Batch size	Feed-forward	Self-attention
Baseline						
ATIS-Slot	2	no clip	$1e^{-4}$	32	1024	256
SNIPS-Slot	3	0.5	$1e^{-4}$	64	1024	512
Chunk	4	1	$7e^{-5}$	32	2048	512
NER	4	no clip	$7e^{-5}$	32	2048	512
PoS Tagging	4	0.5	$1e^{-4}$	32	1024	512
ATIS-Intent	4	no clip	$1e^{-4}$	32	1024	256
SNIPS-Intent	3	0.5	$1e^{-4}$	64	1024	512
Pos/Neg	2	0.5	$5e^{-5}$	64	2048	256
Pros/Cons	3	1	$7e^{-5}$	64	2048	512
LT						
ATIS-Slot	3	1	$7e^{-5}$	32	2048	512
SNIPS-Slot	3	no clip	$1e^{-4}$	32	1024	512
Chunk	3	0.5	$1e^{-4}$	64	1024	512
NER	2	1	$1e^{-4}$	32	1024	512
PoS Tagging	2	0.5	$7e^{-5}$	32	2048	512
ATIS-Intent	2	1	$7e^{-5}$	32	2048	512
SNIPS-Intent	3	0.5	$1e^{-4}$	32	1024	256
Pos/Neg	4	1	$1e^{-4}$	64	2048	512
Pros/Cons	3	no clip	$7e^{-5}$	32	2048	256
LT+R+CM						
ATIS-Slot	3	no clip	$1e^{-4}$	64	2048	512
SNIPS-Slot	3	no clip	$7e^{-5}$	32	1024	512
Chunk	3	1	$7e^{-5}$	32	1024	512
NER	4	1	$7e^{-5}$	64	1024	512
PoS Tagging	2	no clip	$1e^{-4}$	64	1024	512
ATIS-Intent	3	1	$5e^{-5}$	32	2048	512
SNIPS-Intent	3	no clip	$1e^{-4}$	32	2048	512
Pos/Neg	4	1	$1e^{-4}$	32	2048	256
Pros/Cons	2	0.5	$5e^{-5}$	32	1024	512

Table 7: Hyperparameters used for our experiments. The best configuration for LT was also used for LT+R, while the best configuration for LT+R+CM was also used for LT+R+CM+D1 and LT+R+CM+D2.

Tasks	Baseline	LT	LT+R	LT+R +CM	LT+R+ CM+D1	LT+R+ CM+D2
ATIS-Slot	2.0M	9.9M	9.9M	9.9M	9.9M	9.9M
SNIPS-Slot	10.0M	10.0M	10.0M	10.0M	10.0M	10.0M
Chunk	18.5M	12.2M	12.2M	12.2M	12.2M	12.2M
NER	24.4M	16.0M	16.0M	20.2M	20.2M	20.2M
PoS Tagging	20.2M	18.1M	18.1M	16.0M	16.0M	16.0M
ATIS-Intent	3.5M	6.7M	6.7M	9.9M	9.9M	9.9M
SNIPS-Intent	10.0M	6.0M	6.0M	13.1M	13.1M	13.1M
Pos/Neg	4.3M	14.4M	14.4M	6.9M	6.9M	6.9M
Pros/Cons	14.1M	8.5M	8.5M	8.8M	8.8M	8.8M

Table 8: Number of parameter for each model.

Tasks	Datasets	Labels	Train	Valid	Test
Slot filling	ATIS (Hemphill et al., 1990; Dahl et al., 1994)	127	4,478	500	893
Slot filling	SNIPS (Coucke et al., 2018)	72	13,084	700	700
Chunking	CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000)	23	7,922	1,014	2,012
Named entity recognition	OntoNotes 5.0, WSJ section (Weischedel et al., 2013)	37	30,060	5,315	1,640
Parts-of-Speech tagging	OntoNotes 5.0, WSJ section (Weischedel et al., 2013)	48	30,060	5,315	1,640
Intent detection	ATIS (Hemphill et al., 1990; Dahl et al., 1994)	26	4,478	500	893
Intent detection	SNIPS (Coucke et al., 2018)	7	13,084	700	700
Sentiment classification	Positive/Negative (Kotzias et al., 2015)	2	2,100	300	600
Sentiment classification	Pros/Cons (Ganapathibhotla and Liu, 2008)	2	32,088	4,602	9,175

Table 9: Tasks, datasets and their size.

	Tasks	Baseline	LT	LT+R	LT+R +CM	LT+R+ CM+D1	LT+R+ CM+D2
F1	ATIS-Slot	96.93	95.88	89.67	96.07	96.72	94.60
	SNIPS-Slot	91.36	90.02	58.91	83.41	86.44	87.81
	Chunk	92.14	89.66	69.49	87.58	90.46	90.26
	NER	86.43	81.87	51.09	67.50	78.54	82.47
Accuracy	PoS Tagging	96.34	95.72	90.02	94.86	95.84	95.88
	ATIS-Intent	98.40	98.40	93.20	98.60	98.00	98.00
	SNIPS-Intent	99.43	98.86	89.43	99.00	99.00	99.00
	Pos/Neg	87.67	84.67	68.67	83.67	83.33	82.67
	Pros/Cons	95.13	94.98	90.59	94.85	95.11	94.98

Table 10: Non-incremental performance of our models on validation sets for reproducibility purpose.

Tasks/Models	EO	CT	RC	EO$\Delta1$	EO$\Delta2$	RC$\Delta1$	RC$\Delta2$
ATIS-Slot							
Baseline	0.029	0.012	0.963	0.008	0.002	0.986	0.995
LT	0.038	0.017	0.947	0.016	0.007	0.972	0.985
LT+R	0.000	0.000	1.000	-	-	-	-
LT+R+CM	0.000	0.000	1.000	-	-	-	-
LT+R+CM+D1	-	0.000	-	0.000	-	1.000	-
LT+R+CM+D2	-	0.000	-	-	0.000	-	1.000
SNIPS-Slot							
Baseline	0.147	0.078	0.805	0.054	0.026	0.906	0.945
LT	0.189	0.103	0.738	0.075	0.033	0.868	0.929
LT+R	0.000	0.000	1.000	-	-	-	-
LT+R+CM	0.000	0.000	1.000	-	-	-	-
LT+R+CM+D1	-	0.000	-	0.000	-	1.000	-
LT+R+CM+D2	-	0.000	-	-	0.000	-	1.000
Chunk							
Baseline	0.166	0.046	0.743	0.063	0.045	0.852	0.874
LT	0.184	0.056	0.680	0.105	0.081	0.757	0.786
LT+R	0.000	0.000	1.000	-	-	-	-
LT+R+CM	0.000	0.000	1.000	-	-	-	-
LT+R+CM+D1	-	0.000	-	0.000	-	1.000	-
LT+R+CM+D2	-	0.000	-	-	0.000	-	1.000
NER							
Baseline	0.072	0.019	0.898	0.033	0.019	0.935	0.952
LT	0.078	0.022	0.883	0.036	0.021	0.926	0.944
LT+R	0.000	0.000	1.000	-	-	-	-
LT+R+CM	0.000	0.000	1.000	-	-	-	-
LT+R+CM+D1	-	0.000	-	0.000	-	1.000	-
LT+R+CM+D2	-	0.000	-	-	0.000	-	1.000
PoS Tagging							
Baseline	0.114	0.032	0.812	0.045	0.033	0.886	0.904
LT	0.125	0.036	0.777	0.054	0.036	0.854	0.879
LT+R	0.000	0.000	1.000	-	-	-	-
LT+R+CM	0.000	0.000	1.000	-	-	-	-
LT+R+CM+D1	-	0.000	-	0.000	-	1.000	-
LT+R+CM+D2	-	0.000	-	-	0.000	-	1.000

Table 11: Mean value of Edit Overhead, Correction Time Score and Relative Correctness on test sets for sequence tagging. Δt denotes delay for t time steps.

Tasks/Models	EO	CT	RC	EOΔ1	EOΔ2	RCΔ1	RCΔ2
ATIS-Intent							
Baseline	0.488	0.238	0.812	0.304	0.196	0.874	0.923
LT	0.320	0.152	0.885	0.157	0.095	0.934	0.959
LT+R	0.276	0.191	0.836	-	-	-	-
LT+R+CM	0.174	0.097	0.925	-	-	-	-
LT+R+CM+D1	-	0.056	-	0.097	-	0.958	-
LT+R+CM+D2	-	0.032	-	-	0.073	-	0.976
SNIPS-Intent							
Baseline	0.294	0.176	0.857	0.171	0.108	0.896	0.925
LT	0.241	0.172	0.867	0.173	0.132	0.895	0.919
LT+R	0.120	0.131	0.883	-	-	-	-
LT+R+CM	0.188	0.112	0.915	-	-	-	-
LT+R+CM+D1	-	0.073	-	0.130	-	0.944	-
LT+R+CM+D2	-	0.044	-	-	0.083	-	0.969
Pos/Neg							
Baseline	0.358	0.249	0.829	0.245	0.191	0.859	0.881
LT	0.363	0.272	0.821	0.272	0.216	0.843	0.867
LT+R	0.103	0.081	0.931	-	-	-	-
LT+R+CM	0.317	0.205	0.852	-	-	-	-
LT+R+CM+D1	-	0.162	-	0.250	-	0.876	-
LT+R+CM+D2	-	0.117	-	-	0.167	-	0.908
Pros/Cons							
Baseline	0.150	0.112	0.927	0.089	0.054	0.951	0.965
LT	0.158	0.113	0.925	0.086	0.053	0.952	0.966
LT+R	0.095	0.071	0.943	-	-	-	-
LT+R+CM	0.098	0.069	0.952	-	-	-	-
LT+R+CM+D1	-	0.040	-	0.059	-	0.972	-
LT+R+CM+D2	-	0.027	-	-	0.042	-	0.981

Table 12: Mean value of Edit Overhead, Correction Time Score and Relative Correctness on test sets for sequence classification. Δt denotes delay for t time steps.

Tasks/Models	EO	CT	RC
ATIS-Slot			
Baseline			
– GloVe	0.030	0.016	0.955
– Pos	0.109	0.058	0.836
– GloVe & Pos	0.108	0.059	0.834
LT+R+CM			
– GloVe	0.000	0.000	1.000
– Pos	0.000	0.000	1.000
– GloVe & Pos	0.000	0.000	1.000
SNIPS-Slot			
Baseline			
– GloVe	0.154	0.086	0.778
– Pos	0.264	0.172	0.597
– GloVe & Pos	0.236	0.149	0.645
LT+R+CM			
– GloVe	0.000	0.000	1.000
– Pos	0.000	0.000	1.000
– GloVe & Pos	0.000	0.000	1.000
Chunk			
Baseline			
– GloVe	0.177	0.059	0.675
– Pos	0.325	0.190	0.296
– GloVe & Pos	0.360	0.200	0.276
LT+R+CM			
– GloVe	0.000	0.000	1.000
– Pos	0.000	0.000	1.000
– GloVe & Pos	0.000	0.000	1.000
NER			
Baseline			
– GloVe	0.079	0.021	0.875
– Pos	0.096	0.040	0.792
– GloVe & Pos	0.114	0.046	0.759
LT+R+CM			
– GloVe	0.000	0.000	1.000
– Pos	0.000	0.000	1.000
– GloVe & Pos	0.000	0.000	1.000
PoS Tagging			
Baseline			
– GloVe	0.124	0.035	0.790
– Pos	0.184	0.079	0.567
– GloVe & Pos	0.195	0.087	0.545
LT+R+CM			
– GloVe	0.000	0.000	1.000
– Pos	0.000	0.000	1.000
– GloVe & Pos	0.000	0.000	1.000

Table 13: Ablation of GloVe and positional embeddings on the baseline and LT+R+CM for incremental metrics in sequence tagging.

Tasks/Models	EO	CT	RC
ATIS-Intent			
Baseline			
– GloVe	0.505	0.248	0.798
– Pos	0.472	0.235	0.807
– GloVe & Pos	0.497	0.239	0.802
LT+R+CM			
– GloVe	0.179	0.101	0.922
– Pos	0.166	0.091	0.927
– GloVe & Pos	0.190	0.115	0.916
SNIPS-Intent			
Baseline			
– GloVe	0.247	0.172	0.869
– Pos	0.263	0.165	0.872
– GloVe & Pos	0.195	0.139	0.892
LT+R+CM			
– GloVe	0.162	0.093	0.925
– Pos	0.179	0.105	0.921
– GloVe & Pos	0.176	0.102	0.919
Pos/Neg			
Baseline			
– GloVe	0.452	0.368	0.769
– Pos	0.356	0.256	0.826
– GloVe & Pos	0.477	0.375	0.774
LT+R+CM			
– GloVe	0.324	0.197	0.859
– Pos	0.284	0.192	0.856
– GloVe & Pos	0.306	0.180	0.865
Pros/Cons			
Baseline			
– GloVe	0.162	0.116	0.924
– Pos	0.167	0.120	0.924
– GloVe & Pos	0.188	0.142	0.910
LT+R+CM			
– GloVe	0.096	0.065	0.955
– Pos	0.095	0.064	0.955
– GloVe & Pos	0.094	0.066	0.954

Table 14: Ablation of GloVe and positional embeddings on the baseline and LT+R+CM for incremental metrics in sequence classification.