Exploring Looping Effects in RNN-based Architectures

Andrei Shcherbakov[μ] **Saliha Muradoğlu**[ΩΦ] **Ekaterina Vylomova**[μ]
[Ω]The Australian National University (ANU) [μ]The University of Melbourne
[Φ]ARC Centre of Excellence for the Dynamics of Language (CoEDL)
sandreas@unimelb.edu.au, saliha.muradgolu@anu.edu.au
ekaterina.vylomova@unimelb.edu.au

Abstract

The paper investigates *repetitive loops*, a common problem in contemporary text generation (such as machine translation, language modelling, morphological inflection) systems. We hypothesized that a model's failure to distinguish respective latent states for different positions in an output sequence may be the primary cause of the looping. Therefore, we propose adding a position-aware discriminating factor to the model in attempt to reduce that effect. We conduct a study on neural models with recurrent units by explicitly altering their decoder internal state. We use a task of morphological reinflection as a proxy to study the effects of the changes. Our results show that the probability of the occurrence of repetitive loops is significantly reduced by introduction of an extra neural decoder output. The output should be specifically trained to produce gradually increasing value upon generation of each character of a given sequence. We also explored variations of the technique and found that feeding the extra output back to the decoder amplifies the positive effects.

1 Introduction

Over the last few years we witnessed a significant progress in the field of Natural Language Processing (NLP). Many state-of-the-art models are based on neural architectures with recurrent units. For instance, Sutskever et al. (2014) proposed one of the first neural machine translation models that achieved results comparable with statistical models. Similarly, Plank et al. (2016) introduced a neural POS tagging model as a new state-of-the-art on the task. Recently, neural architectures almost superseded non-neural (finite-state or rule-based) approaches in morphology modelling tasks such as morphological reinflection (Cotterell et al., 2016, 2017) with average accuracy being over 90% on high-resource languages. Error analysis conducted by Gorman et al. (2019) demonstrated that among general misprediction errors such as syncretism, the models also produce certain "silly" errors that human learners do not make. One case of such errors, a looping error, is particularly notable. This type of error is not specific to the task and several other papers reported a similar problem (Holtzman et al., 2019; Vakilipourtakalou and Mou, 2020). Still, the causes and the nature of the error remains under-studied. Here we provide some insights on the causes of the issues and possible remedy to it. We consider morphological reinflection task for our experiments since it has low time and space requirements and, therefore, allows us to reproduce cases of looping in sufficient quantities and analyse them relatively easy.

2 Morphological reinflection task

Morphological inflection is the task of generating a target word form (e.g., "runs") from its lemma ("to run") and a set of target morphosyntactic features (tags, "Verb;Present Tense;Singular;3rd Person"). The task is called morphological *reinflection* when the lemma form is replaced with any other form and, optionally, its morphosyntactic features. This is a type of string-to-string transduction problem that in many cases pre-supposes nearly monotonic alignment between the strings. Traditionally, researchers either hand-engineered (Koskenniemi, 1983; Kaplan and Kay, 1994) or used trainable (Mohri, 1997; Eisner, 2002) finite state transducers to solve the task. Most recently, neural models were shown to outperform most non-neural systems, especially in the case of high-resource languages (Cotterell et al., 2016; Vylomova et al., 2020).

3 Data

In terms of the study we focus on two typologically diverse languages, Nen (Evans and Miller, 2016; Evans, 2017, 2019) and Russian. Nen is a Papuan language of the Morehead-Maro (or Yam) family, spoken in the Western province of Papua New Guinea by approximately 400 people. The language is highly under-resourced, and Muradoglu et al. (2020) is the only computational work on it we are aware of, and in current study we use the data derived from their corpus.

Russian, a Slavic language from Indo-European family, on the other hand, is considered as high-resource. We use the splits from the SIGMORPHON–CoNLL 2017 shared task on morphological reinflection (Cotterell et al., 2017).

We used medium sized training sets which occurred to yield highest rates of looped sequences in predicted word forms. The number of samples in the datasets are presented in Table 1.

	Nen	Russian
Training samples	1589	1000
Development samples	227	1000

Table 1: Dataset sizes

4 Experiments

We reused the hard attention model specifically designed for the morphological reinflection task (Aharoni and Goldberg, 2017) for our explorations. The model uses an external aligner (Sudoh et al., 2013) to extract input-to-output character sequence transformation steps for a given morphological sample. Instances of a special character (STEP) are inserted into transformed words to represent alignment step advances. The resulting seq2seq model is trained to perform transformation from a given lemma into a target inflected form which contains STEP characters. The model consists of two modules; (1) an array of LSTM (Hochreiter and Schmidhuber, 1997) encoders and (2) an LSTM decoder. When a STEP character occurs in a target sequence (either learnt or predicted), the encoder array index advances to the next position. It corresponds to advancing current pointer in the lemma by one character. In such a way, a hard monotonic attention schema is implemented.

In our experiments we computed counts of looped sequences in generated word forms during model evaluation rounds that were carried out upon each epoch of model training. We distinguished a generated character sequence as looped if it satisfies both of the following conditions: (1) the sequence contains at least 3 repeated instances of some character subsequence at its very end, and (2) the total length of those repeated subsequences reaches at least 8 characters. While applying such a criterion, we considered predicted sequences in their alphabetical form, with all STEP characters stripped off.[1]

We hypothesized that the looping is primarily caused by merging of decoder states relevant to different word positions. Therefore, introduction of variables that are guaranteed to be different at distinct stages of output word form production should reduce looped prediction rate. Presence of such a variable would facilitate distinguishing states that correspond to different parts of generated word, if even closely surrounding character sequences are similar. To implement this idea, we introduced an extra decoder output that is trained to always be increasing while new output characters are produced. More specifically, we added an extra output r and an extra input $\tilde{r}$ to the decoder. To ensure that r increases gradually while target word characters are generated, we modified calculation of total loss in the model training, allowing an extra (hinge-like) term as follows:

$$L = \max(0, \gamma \cdot (s - \Delta r)) \quad (1)$$

Here Δr is the difference between current and previous r values. Initially, for every predicted word form r is set to zero. Having observed the dynamics of r value in preliminary training experiments, we chose $\gamma = 50; s = 0.05$.

For better exploration of different factors, we tested combinations of the following setting variations:

- Feeding r back to $\tilde{r}$ vs. leaving it unused (letting $\tilde{r} = 0$). We hypothesized that even when an increasing output itself isn't used, computation of its value still affects neural weights at the front layer of the decoder.
- Requiring r to increase vs. leaving it free.
- Scalar vs. vector r (in the latter case, terms according to equation 1 are to be added per each component).

[1] We didn't consider possible irregular (chaotic) looping cases as they are extremely rare.

- Using an externally provided auto-incremented value for r instead of an extra decoder output.

Table 2 presents mode denotations we use in the paper.

We repeated experiments 15 times for each distinct setting. The result figures presented are normalized to single experiment.

denotation	goal for r	$\tilde{r}$ value
n (*"**n**one"*)	none	zero
i (*"**i**ncrement"*)	r is ablated	incrementing
f (*"**f**eedback"*)	none	previous r
u (*"**u**nused"*)	increase	zero
s (*"all **s**et"*)	increase	previous r

Note: if r is a vector, its size is added before a mode symbol: '3f', '3u', '3s'.

Table 2: A summary of explored modes

mode	**nen**	**ru**	mode	**nen**	**ru**
n	0.040	2.313	i	0.020	0.033
	0	1.267		0	0
s	0.017	0.017	3s	0.030	**0.003**
	0	0		0.066	0
u	**0.010**	0.027	3u	0.810	39.87
	0	0.133		0	24.13
f	0.087	5.770	3f	5.823	107.2
	0.066	2.800		2.667	114.7

Table 3: Average looping counts (per epoch) observed at epochs 15..34

mode	**nen**	**ru**	mode	**nen**	**ru**
n	0.725	0.717	i	0.726	0.724
s	**0.732**	0.750	3s	0.716	**0.753**
u	0.727	0.728	3u	0.704	0.669
f	0.432	0.451	3f	0.668	0.574

Table 4: Development set accuracy achieved at different modes

5 Results

The plots given in Fig. 1 present counts of looped predictions at different epochs for the two datasets used (Nen and Russian).[2] It can be observed that

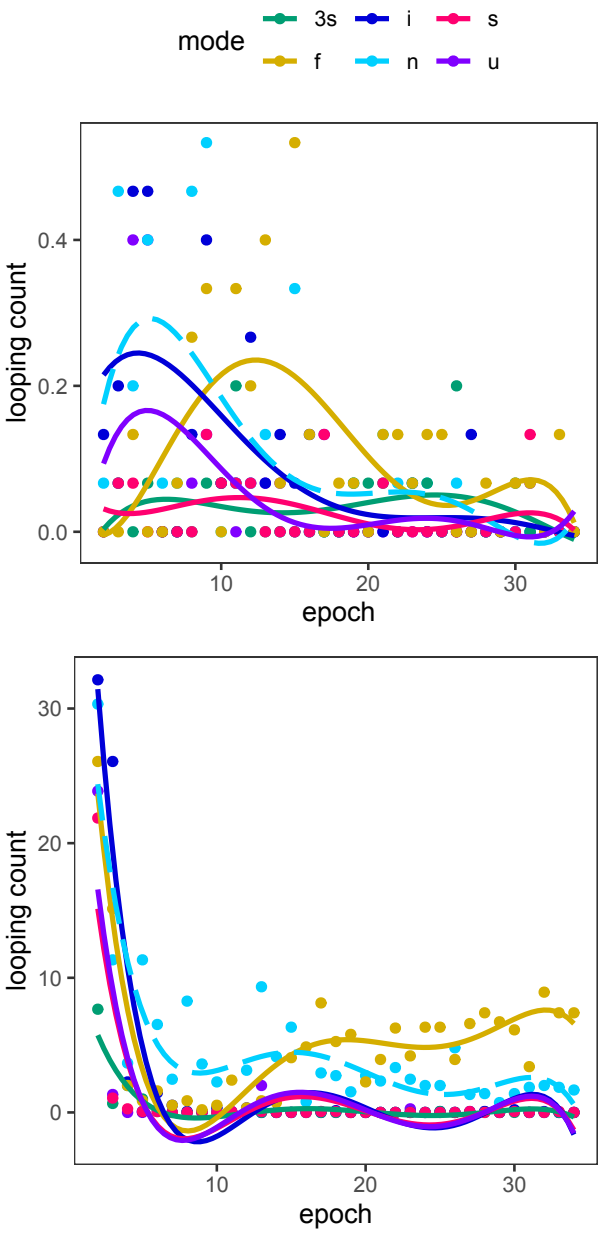

Figure 1: Looping counts observed in training a hard attention model on morphological datasets for Nen (upper plot) and Russian (lower plot)

training a model with increasing r (modes 's', '3s') demonstrates significantly lower rates of looped word generation compared to the baseline mode ('n'). This is true for almost all considered epochs. One may also note that the 'u' mode yields results comparable to ones obtained with the 's' mode. This fact means that the presence of gradually incrementing decoder output is helpful for fighting looping even when the output isn't used. However, if the output is free of constraints and is fed back to the decoder (mode 'f'), the effect is mostly

[2]The curves shown at Fig. 1, 2 are generated by a polynomial smoothing procedure from a dataset with high variance. They may expose some irrelevant artifacts, for example, they fall to negative count values at some points.

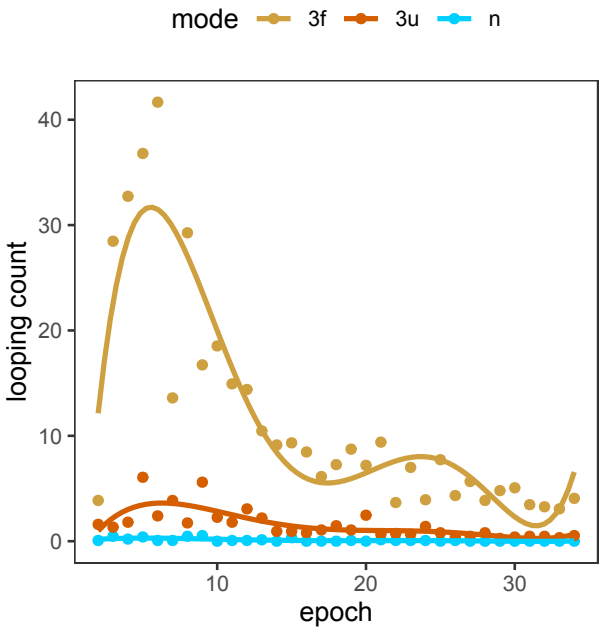

Figure 2: Looping count increase observed at some modes on a Nen language morphological dataset (see Figure 1 for other modes)

negative.

Fig. 1 demonstrates the results of the same kind for the modes that occur to be less looping-prone than the baseline mode. When its components weren't trained to gradually increase (mode '3f'), a vector of 3 feedback values drastically increased looping rate at all epochs. If a vector of 3 increasing components was produced but wasn't fed back as input, the results were still negative. This is surprising because the result for a respective scalar mode ('u') is positive.

Table 3 shows average looping counts for the 'later' epochs (15..34). Those epochs are more significant for the final quality assessment because maximum accuracy is usually achieved at one of them, so they have relatively high probability of producing the best model. Also, the table displays looping counts observed at epochs yielding best prediction accuracy as measured at a respective development set. The figures demonstrate that using modes with gradually increasing r ('s', '3s', 'u', 'i') yields significant reduction of looping rate. The only exception is mode '3u' which causes increase of the rate. As for the 'f' and, especially, '3f' modes (feeding an output back without requirement to grow), they may cause unacceptable high frequency of looped sequence generation. Overall, the digits are in line with the trends shown in the figures.

Increasing the dimensionality of extra decoder output sometimes yields an improvement ('3s' mode) but generally the results suggest that vector size is a factor causing looping rate increase. Finally, scalar seems to be more preferable than vector.

Table 4 shows prediction accuracy figures achieved in the experiments. For each training run, the epoch which produced the highest prediction accuracy against the development set was selected. Then, an average over repeated similar experiments was calculated. According to the figures, 's' mode yields a notable improvement of accuracy. In contrast, sticking to the 'f' mode causes a dramatic decline of accuracy.

6 Discussion

We have found a strong evidence that the presence of a decoder output which is trained to progressively increment reduces the average rate of looping sequences in multiple times. In most cases the positive effect is more significant if this output is fed back to the decoder, although there are exceptions of minor magnitude. Attempts to scale the effect further by increasing dimensionality of progressively incrementing variables are sometimes successful. Still, if we consider an average explored case, the mode 's' seems to be the most effective and consistent in fighting looping. We also observed that presence of an auto-incremented decoder input (mode 'i') leads to looping rate reduction, but the effect is superior if the decoder itself serves to produce a gradually increasing value. Thus, the practical recommendation arising from our research should be (1) adding an extra scalar output to the decoder, (2) endorsing it to increase by inclusion a respective term into a training loss formula, and (3) feeding it back as an encoder input.

Conceptually, it isn't surprising that the presence of an increasing variable helps the decoder to distinguish states rated to different phases of output word production and such a way reduces probability of falling into a loop. Still, the details of this mechanism yet need exploration. In our current work we made no attempt to enforce the usage of the new variable in any way; we only made such a usage potentially possible. A detailed exploration of its effect on the learning process is yet a subject of further research. And, what is even more

practically important, we yet need to find how the system design may be changed to incorporate progressive variables in a more explicit, controllable and efficient way.

The introduction of feedback variables adds elements of RNN architecture to the decoder. We observed highly negative results when such variable values weren't constrained (modes 'f' and, especially, '3f'). This indirectly suggests that RNN schema may not be a good solution for a decoder in terms of looping prevention.

7 Related Work

Holtzman et al. (2019) associated the problem with a more general *degeneration* issues that also includes production of blank and incoherent text. The authors observed that the issue appears during in maximization-based decoding methods such as beam search. As a remedy, they proposed a nucleus sampling technique that truncates unreliable tail of the probability distribution in the decoder part. Kulikov et al. (2019) also compared two search strategies, greedy and beam, proposing a novel iterative beam search strategy that increases diversity of the candidate responses. Contrary to that, Welleck et al. (2019) suggests that the problem cannot be solved by making beam search predictions more diverse. Instead, they propose focusing on likelihood loss, and introduce *"unlikelihood training"* that assigns lower probability to unlikely generations. Finally, following earlier observations on chaotic states w.r.t model parameters in Bertschinger and Natschläger (2004) and Laurent and von Brecht (2016), Vakilipourtakalou and Mou (2020) study chaotic behavior (Kathleen et al., 1996) in RNNs that are defined as iterative maps (Strogatz, 1994).

8 Conclusion

We proposed and explored a simple technique that reduces rate of repetitive loops occurrence in a neural decoder output. Our work was inspired by a hypothesis that looping effects in a neural decoder are caused by its inability to distinguish states related to different positions in a generated word. We both provided a simple and universal practical solution and outlined a promising direction for further research.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada.

Nils Bertschinger and Thomas Natschläger. 2004. Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, 16(7):1413–1436.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared Task—Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

Nicholas Evans. 2017. Quantification in nen. In *Handbook of Quantifiers in Natural Language: Volume II*, pages 571–607. Springer.

Nicholas Evans. 2019. *Waiting for the Word: Distributed Deponency and the Semantic Interpretation of Number in the Nen Verb*, pages 100–123. Edinburgh University Press.

Nicholas Evans and Julia Colleen Miller. 2016. Nen. *Journal of the International Phonetic Association*, 46(3):331–349.

Kyle Gorman, Arya D. McCarthy, Ryan Cotterell, Ekaterina Vylomova, Miikka Silfverberg, and Magdalena Markowska. 2019. Weird inflects but OK: Making sense of morphological generation errors. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 140–151, Hong Kong, China. Association for Computational Linguistics.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Ronald M Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

T Kathleen, D Tim, and A James. 1996. *CHAOS: an introduction to dynamical systems*. Springer, New York, NY, USA.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*, volume 11. University of Helsinki, Department of General Linguistics Helsinki, Finland.

Ilia Kulikov, Alexander Miller, Kyunghyun Cho, and Jason Weston. 2019. Importance of search and evaluation strategies in neural dialogue modeling. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 76–87.

Thomas Laurent and James von Brecht. 2016. A recurrent neural network without chaos. *arXiv preprint arXiv:1612.06212*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

Saliha Muradoglu, Nicholas Evans, and Hanna Suominen. 2020. To compress or not to compress? a finite-state approach to Nen verbal morphology. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 207–213, Online. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418.

Steven H Strogatz. 1994. Nonlinear dynamics and chaos: with applications to physics. *Biology, Chemistry and Engineering*, page 1.

Katsuhito Sudoh, Shinsuke Mori, and Masaaki Nagata. 2013. Noise-aware character alignment for bootstrapping statistical machine transliteration from bilingual corpora. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 204–209.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Pourya Vakilipourtakalou and Lili Mou. 2020. How chaotic are recurrent neural networks? *arXiv preprint arXiv:2004.13838*.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39, Online. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.