Non-Monotonic Sequential Text Generation

Sean Welleck[1], **Kianté Brantley**[2], **Hal Daumé III**[2,3] **and Kyunghyun Cho**[1,4,5]
[1]New York University [2]University of Maryland [3]Microsoft Research
[4]CIFAR Azrieli Global Scholar [5]Facebook AI Research
wellecks@nyu.edu, kdbrant@cs.umd.edu, me@hal3.name, kyunghyun.cho@nyu.edu

1 Introduction

Most sequence-generation models, from n-grams (Bahl, Jelinek, and Mercer, 1983) to neural language models (Bengio, Ducharme, Vincent, and Jauvin, 2003) generate sequences in a purely left-to-right, monotonic order. This raises the question of whether alternative, non-monotonic orders are worth considering (Ford, Duckworth, Norouzi, and Dahl, 2018), especially given the success of "easy first" techniques in natural language tagging (Tsuruoka and Tsujii, 2005), parsing (Goldberg and Elhadad, 2010), and coreference (Stoyanov and Eisner, 2012), which allow a model to effectively learn their own ordering.

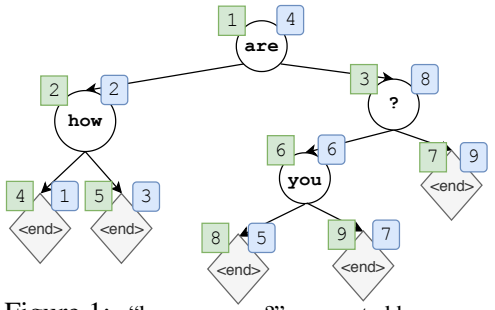

Figure 1: "how are you ?", generated by our approach. The model first generated "are" and then recursively generated left and right subtrees ("how" and "you ?", respectively). At each step, the model may either generate a token, or an ⟨end⟩ token, which indicates that this subtree is complete. The generation is performed in level-order (numbers in left green squares); the output is read off in-order (numbers in right rounded blue squares).

We propose a framework for training sequential text generation models which learn a generation order without having to specify an order in advance. In contrast to the default left-to-right model where the generation order is fixed, our model learns the sequence generation order without any extra supervision. We frame the learning problem as an *imitation learning* problem, in which we aim to learn a generation policy that mimics the actions of an oracle generation policy (§2). Because the tree structure is unknown at training time, the oracle policy cannot know the exact correct actions to take.

Specifically, we consider the problem of sequentially generating a sequence of discrete tokens $Y = (w_1, \ldots, w_N)$, such as a natural language sentence, where $w_i \in V$, a finite vocabulary. Let $\tilde{V} = V \cup \{\langle \text{end} \rangle\}$. The generation process deterministically navigates a state space $\mathcal{S} = \tilde{V}^\star$ where a state $s \in \mathcal{S}$ corresponds to a sequence of tokens from $\tilde{V}$. We interpret this sequence of tokens as a top-down traversal of a binary tree, where $\langle \text{end} \rangle$ terminates a subtree. An action a is an element of $\tilde{V}$ which is appended to the state. Terminal states are those for which all subtrees have been $\langle \text{end} \rangle$'ed.

A policy π is a (possibly) stochastic mapping from states to actions, and we denote the probability of an action $a \in \tilde{V}$ given a state s as $\pi(a|s)$. There are many unique binary trees with an in-order traversal equal to a sequence Y; thus the policy is capable of choosing from many different generation orders for Y. **Note that left-to-right generation can be recovered if $\pi(\langle \text{end} \rangle | s_t) = 1$ if and only if t is odd (or non-zero and even for right-to-left generation).**

This learning problem is challenging because the sequences Y alone only tell us what the final output sequences of words should be, but not what tree(s) should be used to get there. In left-to-right generation, the observed sequence Y fully determines the sequence of actions to take. In our case, however, the tree structure is effectively a latent variable, which will be determined by the policy itself. At training time, we do know *which words* should eventually appear, and their order; this substantially constrains the search space that needs to be explored, suggesting learning-to-search (Daumé, Langford, and Marcu, 2009) and imitation learning (Ross, Gordon, and Bagnell, 2011; Ross and Bagnell, 2014) approaches.

2 Learning to Search

We aim to learn a policy π that mimics an oracle (or "reference") policy π^*. To do so, we define a *roll-in* policy π^{in} and *roll-out* policy π^{out}. The *roll-in* policy determines the state distribution over which

Oracle	%Unique	%Novel	Avg. Span	BLEU
left-right	97.0	17.8	1.0	47.0
uniform	99.9	98.3	1.43	40.0
annealed	98.2	93.1	1.31	56.2
Validation	100	12.1	-	-

Oracle	BLEU (BP)	Test Meteor	YiSi	Ribes
left-right	26.23 (1.00)	27.87	47.58	79.85
uniform	13.17 (0.64)	19.87	36.48	75.36
+⟨end⟩-tuning	17.68 (0.96)	24.53	42.46	74.12
annealed	16.94 (0.72)	23.15	42.39	78.99
+⟨end⟩-tuning	19.19 (0.91)	25.24	43.98	79.24

Table 1: Left:Unconditional language model results; Right:Machine Translation results.

the learned policy π is to be trained. We repeatedly draw states s according to the state distribution induced by π^{in}, and compute cost-to-go under π^{out}. The learned policy π is then trained to choose actions to minimize this cost-to-go estimate. Formally, denote the uniform distribution over $\{1, \dots, T\}$ as $U[T]$; d^t_π as the distribution of states induced by running π; and $\mathcal{C}(\pi; \pi^{\text{out}}, s)$ a scalar cost measuring the loss incurred by π against the cost-to-go estimates under π^{out}. Then, the quantity being optimized is: $\mathbb{E}_{Y\sim D}\mathbb{E}_{t\sim U[2|Y|+1]}\mathbb{E}_{s_t\sim d^t_{\pi^{\text{in}}}}[\mathcal{C}(\pi; \pi^{\text{out}}, s_t)]$. Learning consists of finding a policy which only has access to states s_t but performs as well or better than π^*. There are many ways to measure the prediction cost $\mathcal{C}(\pi; \pi^{\text{out}}, s)$; we use a KL-divergence type loss, measuring the difference between the action distribution produced by π and the action distribution preferred by π^{out}.

All the oracles we consider have access to the ground truth output Y, and the current state s. We interpret the state s as a partial binary tree and a "current node" in that binary tree where the next prediction will go. Given the consecutive subsequence $Y_t = (w'_1, \dots, w'_{N'})$, an oracle policy is:

$$\pi^*(a|s_t) = \mathbf{1}[a = \langle\text{end}\rangle \text{ and } Y_t = \langle\rangle] + p_a \mathbf{1}[a \in Y_t]$$

where the p_as are arbitrary such that $\sum_{a\in Y} p_a = 1$. An oracle policy places positive probability only on valid actions, and forces an $\langle\text{end}\rangle$ output if there are no more words to produce. When an action a is chosen, at s_t, this "splits" the sub-sequence $Y_t = (w'_1, \dots, w'_{N'})$ into left and right sub-sequences, $\overleftarrow{Y}_t = (w'_1, \dots, w'_{i-1})$ and $\overrightarrow{Y}_t = (w'_{i+1}, \dots, w_N)$, where i is the index of a in Y_t. (This split may not be unique due to duplicated words in Y_t, in which case we choose a valid split arbitrarily.)

Uniform Oracle: Motivated by Welleck, Yao, Gai, Mao, Zhang, and Cho (2018), π^*_{uniform} gives uniform probabilities $p_a = 1/n$ for all words in Y_t where n is the number of unique words in Y_t. **Coaching Oracle:** An issue with the uniform oracle is that it does not prefer any specific set of generation orders; motivated by this, we design a coaching oracle as the product of the uniform oracle and current policy π: $\pi^*_{\text{coaching}}(a|s) \propto \pi^*_{\text{uniform}}(a|s)\ \pi(a|s)$. **Annealed Coaching Oracle:** The coaching oracle gives rise to an issue in the early stage of learning, as it does not encourage learning to explore a diverse set of generation orders; we thus design a mixture of the uniform and coaching policies: $\pi^*_{\text{annealed}}(a|s) = \beta\pi^*_{\text{uniform}}(a|s) + (1-\beta)\pi^*_{\text{coaching}}(a|s)$.

3 Experiments

We experiment with our model across two tasks: unconditional generation (e.g. language modeling) using the Persona-Chat (Zhang, Dinan, Urbanek, Szlam, Kiela, and Weston, 2018) dialogue dataset and conditional generation (e.g. machine translation) using IWSLT'16 German $\rightarrow$ English (196k pairs).

For the unconditional language model task we computed statistics over 10,000 samples for each policy trained. In table 1, we see that our models tend to generated more novel sentences (i.e. sentences not in training data) and more unique sentences (i.e. sentences that are not duplicates). The average span (i.e. average number of children per node) is higher than one which means our models generate more bushy trees.

We evaluated our Machine Translation results on four (very) different evaluation measures: BLEU, Meteor (Lavie and Agarwal, 2007), YiSi (Lo, 2018), and Ribes (Isozaki, Hirao, Duh, Sudoh, and Tsukada, 2010). The most dramatic score difference is the drastically superior performance of left-right according to BLEU. We found that the annealed model significantly outperforms the left-right model in 1- and 2-gram precision, ties for 3-gram, and loses for 4-gram; see Table 1, right.

References

Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, 5(2):179–190.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George E Dahl. 2018. The importance of generation order in language modeling. *arXiv preprint arXiv:1808.07910*.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Chi-kiu Lo. 2018. YiSi: A semantic machine translation evaluation metric for evaluating languages with different levels of available resources. Unpublished.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. *Proceedings of COLING 2012*, pages 2519–2534.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 467–474. Association for Computational Linguistics.

Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. 2018. Loss functions for multiset prediction. In *Advances in Neural Information Processing Systems*, pages 5788–5797.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.