The Importance of Generation Order in Language Modeling

Nicolas Ford[*]   **Daniel Duckworth**   **Mohammad Norouzi**   **George E. Dahl**
Google Brain
{`nicf,duckworthd,mnorouzi,gdahl`}`@google.com`

Abstract

Neural language models are a critical component of state-of-the-art systems for machine translation, summarization, audio transcription, and other tasks. These language models are almost universally autoregressive in nature, generating sentences one token at a time from left to right. This paper studies the influence of token generation order on model quality via a novel *two-pass language model* that produces partially-filled sentence "templates" and then fills in missing tokens. We compare various strategies for structuring these two passes and observe a surprisingly large variation in model quality. We find the most effective strategy generates *function* words in the first pass followed by *content* words in the second. We believe these experimental results justify a more extensive investigation of generation order for neural language models.

1   Introduction

Neural networks have been extremely successful statistical models of text in language modeling and machine translation. Despite differences in model architectures, state of the art neural nets generate sequences from left to right (Vaswani et al., 2017; Jozefowicz et al., 2016; Wu et al., 2016). Although in some sense humans produce and consume language from left to right as well, there are many other intuitively appealing ways to generate text. For instance, language is slow enough on a neurological time scale for multiple passes of generation that incorporate feedback to occur. Linguistic intuition might suggest that we should first generate some abstract representation of what we want to say and then serialize it, a process that seems more universally appropriate given the existence of languages with freer word order such as Czech and Polish.

There has been interest in moving beyond the left-to-right generation order by developing alternative multi-stage strategies such as syntax-aware neural language models (Bowman et al., 2016) and latent variable models of text (Wood et al., 2011). Before embarking on a long-term research program to find better generation strategies that improve modern neural networks, one needs evidence that the generation strategy can make a large difference. This paper presents one way of isolating the generation strategy from the general neural network design problem. Our key technical contribution involves developing a flexible and *tractable* architecture that incorporates different generation orders, while enabling *exact* computation of the log-probabilities of a sentence. Our experiments demonstrate that even when using a few simple two-pass generation orders, the differences between good and bad orderings are substantial.

We consider ways of reordering the tokens within a sequence based on their identities. The best ordering we tried generates function words first and content words last, which cuts against the idea of committing to the general topic of a sentence first and only then deciding exactly how to phrase it. We offer some possible explanations in Section 3, and we conclude that our experimental results justify a more extensive investigation of the generation order for language and translation models.

2   Two-pass Language Models

We develop a family of *two-pass language models* that depend on a partitioning of the vocabulary into a set of *first-pass* and *second-pass* tokens to generate sentences. We perform a preprocessing step on each sequence $\mathbf{y}$, creating two new sequences $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$. The sequence $\mathbf{y}^{(1)}$, which we call the *template*, has the same length as $\mathbf{y}$, and consists of the first-pass tokens from $\mathbf{y}$ together with a special *placeholder* token wherever

---

Table 1 (placeholder token indicated by "‿"):

sentence	common first	rare first	function first	content first	odd first
" all you need to do if you want the nation 's press camped on your doorstep is to say you once had a [UNK] in 1947 , " he noted memorably in his diary . [EOS]	" all you ‿ to ‿ if you ‿ the ‿ 's ‿ ‿ on ‿ ‿ is to ‿ you ‿ had a [UNK] in ‿ , " he ‿ ‿ in his ‿ . [EOS]	‿ ‿ ‿ need ‿ do ‿ ‿ want ‿ nation ‿ press camped ‿ your doorstep ‿ ‿ say ‿ once ‿ ‿ ‿ 1947 ‿ ‿ ‿ noted memorably ‿ ‿ diary ‿ [EOS]	" all you ‿ to ‿ if you ‿ the ‿ 's ‿‿ on your ‿ is to ‿ you ‿ ‿ a ‿ in ‿ , " he ‿ ‿ in his ‿ . [EOS]	‿ ‿ ‿ need ‿ do ‿ ‿ want ‿ nation ‿ press camped ‿ ‿ doorstep ‿ ‿ say ‿ once had ‿ [UNK] ‿ 1947 ‿ ‿ ‿ noted memorably ‿ ‿ diary ‿ [EOS]	" all you need ‿ ‿ ‿ you ‿ the nation 's press camped on your doorstep ‿ ‿ say you once had ‿ ‿ ‿ ‿ ‿ " ‿ noted ‿ his ‿ . [EOS]
the team announced thursday that the 6-foot-1 , [UNK] starter will remain in detroit through the 2013 season . [EOS]	the ‿ ‿ ‿ that the ‿ , [UNK] ‿ will ‿ in ‿ ‿ the ‿ ‿ . [EOS]	‿ team announced thursday ‿ ‿ 6-foot-1 ‿ ‿ starter ‿ remain ‿ detroit through ‿ 2013 season ‿ [EOS]	the ‿ ‿ ‿ that the ‿ , ‿ ‿ will ‿ in ‿ through the ‿ ‿ . [EOS]	‿ team announced thursday ‿ ‿ 6-foot-1 ‿ [UNK] starter remain ‿ detroit ‿ ‿ 2013 season ‿ [EOS]	the team announced ‿ ‿ the 6-foot-1 ‿ ‿ ‿ will remain ‿ ‿ through the 2013 ‿ . [EOS]
scotland 's next game is a friendly against the czech republic at hampden on 3 march . [EOS]	‿ 's ‿ ‿ is a ‿ ‿ the ‿ ‿ at ‿ on ‿ ‿ . [EOS]	scotland ‿ next game ‿ ‿ friendly against ‿ czech republic ‿ hampden ‿ 3 march ‿ [EOS]	‿ 's ‿ ‿ is a ‿ against the ‿ ‿ at ‿ on ‿ ‿ . [EOS]	scotland ‿ next game ‿ ‿ friendly ‿ ‿ czech republic ‿ hampden ‿ 3 march ‿ [EOS]	‿ 's next game ‿ ‿ ‿ ‿ the czech republic at hampden on 3 march . [EOS]
of course , millions of additional homeowners did make a big mistake : they took advantage of " liar loans " and other [UNK] deals to buy homes they couldn 't afford . [EOS]	of ‿ , ‿ of ‿ ‿ ‿ ‿ a ‿ ‿ : they ‿ ‿ of " ‿ ‿ " and ‿ [UNK] ‿ to ‿ ‿ they ‿ 't ‿ . [EOS]	‿ course ‿ millions ‿ additional homeowners did make ‿ big mistake ‿ ‿ took advantage ‿ ‿ liar loans ‿ ‿ other ‿ deals ‿ buy homes ‿ couldn ‿ afford ‿ [EOS]	of ‿ , ‿ of ‿ ‿ ‿ a ‿ ‿ : they ‿ ‿ of " ‿ " and ‿ ‿ ‿ to ‿ ‿ they ‿ ‿ ‿ . [EOS]	‿ course ‿ millions ‿ additional homeowners did make ‿ big mistake ‿ ‿ took advantage ‿ ‿ liar loans ‿ ‿ other [UNK] deals ‿ buy homes ‿ couldn 't afford ‿ [EOS]	of ‿ ‿ ‿ of additional ‿ ‿ ‿ ‿ big ‿ ‿ they ‿ advantage of " liar ‿ " and other ‿ deals ‿ buy homes they couldn ‿ afford . [EOS]

Table 1: Some example sentences from the dataset and their corresponding templates. The placeholder token is indicated by "‿".

y had a second-pass token. The sequence $\mathbf{y}^{(2)}$ has length equal to the number of these placeholders, and consists of the second-pass tokens from **y** in order.

We use a neural language model p_1 to generate $\mathbf{y}^{(1)}$, and then a conditional translation model p_2 to generate $\mathbf{y}^{(2)}$ given $\mathbf{y}^{(1)}$. Note that, since the division of the vocabulary into first- and second-pass tokens is decided in advance, there is a one-to-one correspondence between sequences **y** and pairs $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)})$. The total probability of **y** is then

$$p(\mathbf{y}) \;=\; p_1(\mathbf{y}^{(1)})\, p_2(\mathbf{y}^{(2)} \mid \mathbf{y}^{(1)}) . \qquad (1)$$

Two-pass language models present a unique opportunity to study the importance of generation order because, since the template is a deterministic function of **y**, the probability of **y** can be computed exactly. This is in contrast to a language model using a *latent* generation order, which requires a prohibitive marginalization over permutations to compute the exact probabilities. Given the tractable nature of the model, exact learning based on log-likelihood is possible, and we can compare different vocabulary partitioning strategies both against each other and against a single-pass language model.

Our implementation consists of two copies of the Transformer model from Vaswani et al. (2017). The first copy just generates the template, so it has no encoder. The second copy is a sequence-to-sequence model that translates the template into the complete sentence. There are three places in this model where word embeddings appear — the first-phase decoder, the second-phase encoder, and the second-phase decoder — and all three sets of parameters are shared. The output layer also shares the embedding parameters.[1]

For the second pass, we include the entire target sentence, not just the second-pass tokens, on the output side. In this way, when generating a token, the decoder is allowed to examine all tokens to the left of its position. However, only the second-pass tokens count toward the loss, since in the other positions the correct token is already known. Our loss function is then the sum of all of these numbers (from both copies) divided by the length of the original sentence, which is the log-perplexity that our model assigns to the sentence.

We tried five different ways of splitting the vocabulary:

Common First and Rare First: The vocabulary was sorted by frequency and then a cutoff was chosen, splitting the vocabulary into "common" and "rare" tokens. The location of the cutoff[2] was chosen so that the number of common tokens and the number of rare tokens in the average sentence were approximately the same. In "common first"

---

[1] This behavior is enabled in the publicly available implementation of Transformer using the hyperparameter called `shared_embedding_and_softmax_weights`.

[2] In our experiments on LM1B, this is at index 78.

we place the common tokens in the first pass, and in "rare first" we start with the rare tokens.

Function First and Content First: We parsed about 1% of LM1B's training set using Parsey Mc-Parseface (Andor et al., 2016) and assigned each token in the vocabulary to the grammatical role it was assigned most frequently by the parser. We used this data to divide the vocabulary into "function" words and "content" words; punctuation, adpositions, conjunctions, determiners, pronouns, particles, modal verbs, "wh-adverbs" (Penn part-of-speech tag `WRB`), and conjugations of "be" were chosen to be function words. In "function first" we place the function words in the first phase and in "content first" we start with the content words.

Odd First: As a control, we also used a linguistically meaningless split where tokens at an odd index in the frequency-sorted vocabulary list were assigned to the first pass and tokens with an even index were assigned to the second pass.

A few sentences from the dataset are shown in Table 1 together with their templates. Note that the common and function tokens are very similar; the main differences are the "unknown" token, conjugations of "have," and some prepositions.

3   Experimental Results and Discussion

We ran experiments with several different ways of splitting the vocabulary into first-pass and second-pass tokens. We trained all of these models on the One Billion Word Language Modeling benchmark (LM1B) dataset (Chelba et al., 2013). One sixth of the training data was used as a validation set. We used a vocabulary of size 65,536 consisting of whole words (rather than word pieces) converted to lower-case.

We compared the two-pass generation strategies to a baseline version of Transformer without an encoder, which was trained to unconditionally predict the target sentences in the ordinary way. Because the two-pass models contain slightly more trainable parameters than this baseline, we also compare to an "enhanced baseline" in which the size of Transformer's hidden space was increased to make the number of parameters match the two-pass models.

Both the two-pass models and the baselines used the hyperparameters referred to as `base` in the publicly available implementation of Transformer,[3] which has a hidden size of 512, a filter

---

[3] github.com/tensorflow/tensor2tensor

size of 2048, and 8 attention heads, except that the enhanced baseline used a hidden size of 704. We used a batch size of 4096. All models were trained using ADAM (Kingma and Ba, 2014), with $\beta_1 = 0.85$, $\beta_2 = 0.997$, and $\epsilon = 10^{-6}$. The learning rate was tuned by hand separately for each experiment and the experiments that produced the best results on the validation set are reported. Dropout was disabled after some initial experimentation found it to be detrimental to the final validation loss.

Table 2 shows the results for all the two-pass generation strategies we tried as well as the baselines, sorted from worst to best on the validation set. Strikingly, the linguistically meaningless odd first generation strategy that splits words arbitrarily between the two phases is far worse than the baseline, showing that the two-pass setup on its own provides no inherent advantage over a single phase. The common first and closely related function first strategies perform the best of all the two-pass strategies, whereas the rare first and closely related content first strategies are much worse. Since the control, rare first, and content first orderings are all worse than the baseline, the gains seen by the other two orderings cannot be explained by the increase in the number of trainable parameters alone.

The enhanced version of the baseline achieved slightly better perplexity than the best of the two-pass models we trained. Given that state-of-the-art results with Transformer require models larger than the ones we trained, we should expect growing the embedding and hidden size to produce large benefits. However, the two-pass model we proposed in this work is primarily a tool to understand the importance of sequence generation order and was not designed to be parameter efficient. Thus, as these results indicate, increasing the embedding size in Transformer is a more effective use of trainable parameters than having extra copies of the other model parameters for the second pass (recall that the embeddings are shared across both passes).

One potential explanation for why the function first split performed the best is that, in order to generate a sentence, it is easier to first decide something about its syntactic structure. If this is the primary explanation for the observed results, then common first's success can be attributed to how many function words are also common. However, an alternative explanation might

Model	Train	Validation	Test
odd first	39.925	45.377	45.196
rare first	38.283	43.293	43.077
content first	38.321	42.564	42.394
common first	36.525	41.018	40.895
function first	36.126	40.246	40.085
baseline	38.668	41.888	41.721
enhanced baseline	35.945	39.845	39.726

Table 2: The perplexities achieved by the best version of each of our models.

simply be that it is preferable to delay committing to a rare token for as long as possible as all subsequent decisions will then be conditioning on a low-probability event. This is particularly problematic in language modeling where datasets are too small to cover the space of all utterances. We lack sufficient evidence to decide between these hypotheses and believe further investigation is necessary.

Ultimately, our results show that content-dependent generation orders can have a surprisingly large effect on model quality. Moreover, the gaps between different generation strategies can be quite large.

4 Related Work

For tasks conditioning on sequences and sets, it is well known that order significantly affects model quality in applications such as machine translation (Sutskever et al., 2014), program synthesis (Vinyals et al., 2016), and text classification (Yogatama et al., 2016). Experimentally, Khandelwal et al. (2018) show that recurrent neural networks have a memory that degrades with time. Techniques such as attention (Bahdanau et al., 2014) can be seen as augmenting that memory.

Text generation via neural networks, as in language models and machine translation, proceeds almost universally left-to-right (Jozefowicz et al., 2016; Sutskever et al., 2014). This is in stark contrast to phrase-based machine translation systems (Charniak et al., 2003) which traditionally split token translation and "editing" (typically via reordering) into separate stages. This line of work is carried forward in Post-Editing Models (Junczys-Dowmunt and Grundkiewicz, 2016), Deliberation Networks (Xia et al., 2017), and Review Network (Yang et al., 2016) which produce a "draft" decoding that is further edited. As any valid sequence may be used in a draft, calculating perplexity in these models is unfortunately intractable,

and model quality can only be evaluated via external tasks.

In addition to surface-form intermediate representation, syntax-based representations have a rich history in text modeling. Chelba and Jelinek (1998); Yamada and Knight (2001); Graham and Genabith (2010); Shen et al. (2018) integrate parse structures, explicitly designed or automatically learned, into the decoding process.

Similar to the second phase of this work's proposed model, (Fedus et al., 2018) directly tackles the problem of filling in the blank, akin to the second stage of our proposed model. The Multi-Scale version of PixelRNN in (Van Oord et al., 2016) was also an inspiration for the two-pass setup we used here.

5 Conclusion and Future Work

To investigate the question of generation order in language modeling, we proposed a model that generates a sentence in two passes, first generating tokens from left to right while skipping over some positions and then filling in the positions that it skipped. We found that the decision of which tokens to place in the first pass had a strong effect.

Given the success of our function word first generation procedure, we could imagine taking this idea beyond splitting the vocabulary. One could run a parser on each sentence and use the resulting tree to decide on the generation order. Such a scheme might shed light on which aspect of this split was most helpful. Finally, filling in a template with missing words is a task that might be interesting in its own right. One might want to provide partial information about the target sentence as part of scripting flexible responses for a dialogue agent, question answering system, or other system that mixes a hand-designed grammar with learned responses.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1466–1477.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 225–231. Association for Computational Linguistics.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better text generation via filling in the _____. In *International Conference on Learning Representations (ICLR)*.

Yvette Graham and Josef Genabith. 2010. Deep syntax language models and statistical machine translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 118–126.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. *arXiv preprint arXiv:1605.04800*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The sequence memoizer. *Communications of the ACM*, 54(2):91–98.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems*, pages 1782–1792.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. 2016. Review networks for caption generation. In *Advances in Neural Information Processing Systems*, pages 2361–2369.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*.