Estimating Marginal Probabilities of n-grams for Recurrent Neural Language Models

Thanapon Noraset[*]
Mahidol University
Nakhon Pathom, Thailand

Doug Downey[†]
Northwestern University
Evanston, IL, USA

Lidong Bing[‡]
Tencent AI Lab
Shenzhen, China

Abstract

Recurrent neural network language models (RNNLMs) are the current standard-bearer for statistical language modeling. However, RNNLMs only estimate probabilities for complete sequences of text, whereas some applications require context-independent phrase probabilities instead. In this paper, we study how to compute an RNNLM's *marginal* probability: the probability that the model assigns to a short sequence of text when the preceding context is not known. We introduce a simple method of altering the RNNLM training to make the model more accurate at marginal estimation. Our experiments demonstrate that the technique is effective compared to baselines including the traditional RNNLM probability and an importance sampling approach. Finally, we show how we can use the marginal estimation to improve an RNNLM by training the marginals to match n-gram probabilities from a larger corpus.

1 Introduction

Recurrent neural networks (RNNs) are the state-of-the-art architecture for statistical language modeling (Jozefowicz et al., 2016; Melis et al., 2018), the task of assigning a probability distribution to a sequence of words. The relative likelihoods of the sequences are useful in applications such as speech recognition, machine translation, automated conversation, and summarization (Mikolov et al., 2010; Bahdanau et al., 2014; See et al., 2017; Wen et al., 2017). Typically, RNN language models (RNNLMs) are trained on complete sequences (e.g., a sentence or an utterance), or long sequences (e.g. several documents), and used in the same fashion in applications or testing.

[*] thanapon.nor@mahidol.edu
[†] d-downey@northwestern.edu
[‡] lyndonbig@tencent.com

A question arises when we want to compute the probability of a short sequence without the preceding context. For instance, we may wish to query for how likely the RNNLM is to generate a particular phase aggregated over *all contexts*. We refer to this context-independent probability of a short phrase as a *marginal probability*, or marginal.

These marginal probabilities are useful in three board categories of applications. First, they allow us to inspect the behavior of a given RNNLM. We could check, for example, whether an RNNLM-based generator might ever output a given offensive phrase. Second, the marginals could be used in phrase-based information extraction, such as extracting cities by finding high-probability x's in the phrase "cities such as x" (Soderland et al., 2004; Bhagavatula et al., 2014). Finally, we can use the phrase probabilities to train an RNNLM itself, e.g. updating the RNNLM according to n-gram statistics instead of running text (Chelba et al., 2017; Noraset et al., 2018). In our experiments, we show an example of the last application.

Estimating marginals from an RNNLM is challenging. Unlike an n-gram language model (Chen and Goodman, 1996), an RNNLM does not explicitly store marginal probabilities as its parameters. Instead, previous words are recurrently combined with the RNN's hidden state to produce a new state, which is used to compute a probability distribution of the next word (Elman, 1990; Mikolov et al., 2010). When the preceding context is absent, however, the starting state is also missing. In order to compute the marginal probability, in principle we must marginalize over all possible previous contexts or all continuous-vector states. Both options pose a severe computational challenge.

In this paper, we study how to efficiently approximate marginal probabilities from an RNNLM, without generating a large amount of text. Given an RNNLM and a phrase, our goal is to estimate how frequently the phrase will occur in text generated by the RNNLM. We present two approaches that can be used to estimate the marginal probabilities: sampling for the starting state, and using a single starting state with altered RNNLM training. We show empirically that we can use a zero vector as a starting state of an RNNLM to compute accurate marginal estimates, but we must randomly reset the RNNLM state to zero during training and add a unigram likelihood term to the RNNLM training objective. Finally, we demonstrate that we can use marginal estimation to incorporate n-gram statistics from a larger corpus to improve the perplexity of an RNNLM trained on a similar, but smaller corpus.

2 Marginal Estimation

The goal of marginal estimation is to determine the likelihood of a short phrase where the preceding context is not known; we refer to this likelihood as a *marginal probability*. In other words, the marginal probability of a query refers to how likely a language model will generate a query regardless of context.

2.1 Problem settings

An RNNLM (Mikolov et al., 2010) defines a probability distribution over words conditioned on previous words as the following:

$$\begin{aligned} P(w_{1:T}) &= \prod_{t=1}^{T} P(w_t | w_{1:t-1}) \\ P(w_t | w_{1:t-1}) &= P(w_t | h_t) \propto \exp(\theta_o^{(w)} h_t) \\ h_t &= g(h_{t-1}, w_{t-1}) \end{aligned}$$

where $w_{1:t-1}$ is a sequence of previous words, θ_o^w denotes the output weights of a word w, and $g(\cdot)$ is a recurrent function such as an LSTM (Hochreiter and Schmidhuber, 1997) or GRU unit (Cho et al., 2014).

An initial state, h_1 is needed to start the recurrent function $g(h_1, w_1)$, and also defines the probability distribution of the first word $P(w_1|h_1)$. In the standard language model setting, we compute h_1 using a start-of-sentence symbol for w_0 ("<s>"), and a special starting state h_0 (usually set to be a vector of zeros $\vec{0}$). This initialization approach works fine for long sequences, because it is only utilized once and its effect is quickly swamped by the recurrent steps of the network. However, it is not effective for estimating marginal probabilities of a short phrase. For example, if we naively apply the standard approach to compute the probability of the phrase "`of the`", we would obtain:

$$\begin{aligned} P(\texttt{of the}) = &P(\texttt{of} | h_1 = g(\vec{0}, \texttt{<s>})) \times \\ &P(\texttt{the} | h_2 = g(h_1, \texttt{the})) \end{aligned}$$

The initial setting of the network results in low likelihoods of the first few tokens in the evaluation. For instance, the probability $P(\texttt{of the})$ computed in the above fashion will likely be a bad underestimate, because "`of the`" does not usually start a sentence.

We would like to compute the likelihood of standalone phrases, where rather than assuming the starting state we instead marginalize out the preceding context. Let the RNN's state prior to our query sequence be $z \in \mathbb{R}^d$, a vector-valued random variable representing the RNN initial state, and let $w_{1:T}$ be a short sequence of text. The marginal probability is defined as:

$$P(w_{1:T}) = \int P(w_{1:T}|z) P(z) dz \qquad (1)$$

The integral form of the marginal probability is intractable and requires an unknown density estimator of the state, $P(z)$.

2.2 Trace-based approaches

The integral form of the marginal probability in Eq 1 can be approximated by sampling for z. In this approach, we assume that there is a source of samples which asymptotically approaches the true distribution of the RNN states as the number of samples grows. In this work, we use a collection of RNN states generated in an evaluation, called a *trace*.

Given a corpus of text, a trace of an RNNLM is the corresponding list of RNN states, $H^{(tr)} = (h_1^{(tr)}, h_2^{(tr)}, ..., h_M^{(tr)})$, produced when evaluating the corpus. We can estimate the marginal probability by sampling the initial state z from H as follows:

$$P(w_{1:T}) = \mathbb{E}_{z \sim H^{(tr)}} \left[P(w_1|z) \prod_{t=2}^{T} p(w_t|h_t) \right] \quad (2)$$

where $h_2 = g(z_\psi, w_1)$ and $h_t = g(h_{t-1}, w_{t-1})$ for $t > 2$ (i.e. the following states are the deterministic output of the RNN function). Given a large trace this may produce accurate estimates, but it is intractably expensive and also wasteful, since in general there are very few states in the trace that yield a high likelihood for a sequence.

To reduce the number of times we run the model on the query, we use importance sampling over the trace. We train an encoder to output for a given n-gram query a state "near" the starting state(s) of the query, $z_\chi = q_\chi(w_{1:T})$. We define a sampling weight for a state in the trace, $h^{(tr)}$, proportional to the dot product of the state and the output of the encoder, z_χ , as the following:

$$P(h^{(tr)}|w_{1:T}) = \frac{exp(z_\chi h^{(tr)})}{\sum\limits_{h'^{(tr)} \in H^{(tr)}} exp(z_\chi h'^{(tr)})}$$

This distribution is biased to toward states that are likely to precede the query $w_{1:T}$. We can estimate the marginal probability as the following:

$$P(w_{1:T}) = \mathop{\mathbb{E}}_{z \sim P(h^{(tr)}|w_{1:T})} \left[\frac{P(z)}{P(z|w_{1:T})} P(w_1|z) \prod_{t=2}^{T} p(w_t|h_t) \right]$$

Here the choice of the prior $P(z)$ is a uniform distribution over the states in the trace. The encoder, $q_\chi(w_{1:T})$, is a trained RNN with its input reversed, and z_χ is the final output state of q_χ. To train the encoder, we randomly draw sub-strings $w_{i:i+n}$ of random length from the text used to produce the trace, and minimize the mean-squared difference between z_χ and $h_i^{(tr)}$.

2.3 Fixed-point approaches

While the trace-based approaches work on an existing (already trained) RNNLM, they might take several samples to accurately estimate the marginal probability. We would like to have a single point as the starting state, named z_ψ. We can either train this vector or simply set it to a zero vector. Then the marginal probability in Eq 1 can be estimated with a single run i.e. $p(z_\psi) = 1.0$ and $p(z) = 0.0$ if $z \neq z_\psi$. The computation is reduced to:

$$P(w_{1:T}) = P(w_1|z_\psi) \prod_{t=2}^{T} P(w_t|h_t) \qquad (3)$$

where $h_2 = g(z_\psi, w_1)$ and the rest of the state process is as usual, $h_t = g(h_{t-1}, w_{t-1})$. In this paper, we set z_ψ to be a zero vector, and call this method *Zero*.

As we previously discussed, our fixed-point state, z_ψ, is not a suitable starting state of all n-grams for any given RNNLM, so we need to train an RNNLM to adapt to this state. To achieve this, we use a slight modification of the RNN's truncated back-propagation through time training algorithm. We randomly *reset* the states to z_ψ when computing a new state during the training of the RNNLM (a similar reset was used for a different purpose—regularization—in Melis et al. (2018)). This implies that z_ψ is trained to maximize the likelihood of different subsequent texts of different lengths, and thus is an approximately good starting point for any sequence. Specifically, a new state is computed during the training as follows:

$$h_t = r z_\psi + (1 - r) g(h_{t-1}, w_{t-1})$$

where $r \sim Bern(\rho)$ and ρ is a hyper-parameter for the probability of resetting a state. Larger ρ means more training with z_ψ, but it could disrupt the long-term dependency information captured in the state. We keep ρ relatively small at 0.05.

In addition to the *state reset*, we introduce a *unigram* regularization to improve the accuracy of the marginal estimation. From Eq 3, z_ψ is used to predict the probability distribution of the first token, which should be the unigram distribution. To get this desired behavior, we employ regularization to maximize the likelihood of each token in the training data *independently* (as if reset every step). We call this a unigram regularizer:

$$\mathcal{L}_U = -\sum_{t=1}^{T} \log P(w_t|z_\psi)$$

and we add it to the training objective: $\mathcal{L}_{text} = -\sum_{t=1}^{T} \log P(w_t|h_t)$. Thus, the overall training loss is: $\mathcal{L} = \mathcal{L}_{text} + \mathcal{L}_U$.

3 Experiments and Results

3.1 Experimental Settings

We experiment with a standard medium-size LSTM language model (Zaremba et al., 2014) over 2 datasets: Penn Treebank (PTB) (Mikolov et al., 2010) and WikiText-2 (WT-2) (Merity et al., 2017). We use weight tying (Inan et al., 2017) and train all models with Adam (Kingma and Ba, 2014) for 40 epochs with learning rate starting from 0.003 and decaying every epoch at the rate of 0.85. We use a batch size of 64 and truncated backpropagation with 35 time steps, and

	PTB		WT-2	
	$E(\cdot)$	PPL	$E(\cdot)$	PPL
Zero	3.828	90.18	4.432	104.08
Zero$^{(RU)}$	**0.425**	91.92	**0.801**	106.23
Trace-IW	0.661	-	0.862	-
Zero$^{(R)}$	2.968	93.70	3.519	109.02
Zero$^{(U)}$	1.007	**90.15**	1.713	**102.56**
Trace-Rand	1.105	-	1.742	-
n-grams	26,070		47,130	

Table 1: The average error of different marginal estimation approaches and the testing perplexity. The RNNLM trained with *state-reset (R)* and *unigram (U)* regularization has the lowest error when using a zero starting state to estimate the marginals.

employ variational dropout (Gal and Ghahramani, 2016). The final parameter set is chosen as the one minimizing validation loss. For the query model $q_X(w_{1:T})$ used in importance sampling, we use the same settings for the model and the training procedure as the above.

3.2 Marginal Estimation

In this subsection, we evaluate the accuracy of each approach at estimating marginal probabilities. Given a model and a phrase, we first obtain a target marginal probability, $P_{text}(w_{1:T})$, from a frequency of the phrase occurring in a text generated by the model. Then, we use each approach to estimate the marginal probability of the phrase, $P_{est}(w_{1:T})$. To measure the performance, we compute the absolute value of the log ratio (lower is better) between the target marginal probability and the estimated marginal probability (P_{est}):

$$E(w_{1:T}) = \left|\log(P_{text}(w_{1:T})/P_{est}(w_{1:T}))\right| \quad (4)$$

This evaluation measure gives equal importance to every n-gram regardless of its frequency.

In the following experiments, we generate approximately 2 million and 4 million tokens for PTB and WT-2 models respectively. The probability of phrases occurring in the generated text serves as our target marginal. We form a test set consisting of all n-grams in the generated text for $n \leq 5$ words, excluding n-grams with frequency less than 20 to reduce noise from the generation. For the trace-based estimations, we average the marginal probabilities from 100 samples.

Table 1 shows the average discrepancy between marginal probabilities estimated by generated-text

	1	2	3	4	5
Zero	1.02	4.18	6.15	8.78	10.8
Zero$^{(RU)}$	**0.38**	**0.72**	0.95	1.54	2.10
Trace-IW	0.70	0.81	**0.92**	**1.22**	**1.48**
n-grams	10.7	20.7	10.5	3.7	1.5

Table 2: The error aggregate by n-gram lengths. This shows the same trend as in Table 1, but *Trace-IW* performs better for longer n-grams.

statistics and the methods discussed in Section 2 (Eq 4). From the table, the RNNLM trained with the *state-reset* and the *unigram* regularization (*Zero*$^{(RU)}$) performs better than both zero-start and trace-based approaches on the traditional model. The importance sampling method (*Trace-IW*) has the second lowest error and performs better than random sampling (*Trace-Rand*). Ablation analysis shows that both *state-reset* and the *unigram* regularization contribute to the accuracy. Note that the trace-based methods use the same model as *Zero*.

To show how performance varies depending on the query, we present results aggregated by n-gram lengths. Table 2 shows the errors of the WT-2 dataset. When the n-gram length is greater than 2, *Trace-IW* has better accuracy. This makes sense because the encoder has more evidence to use when inferring the likely start state.

3.3 Training with marginal probabilities

We now turn to an application of the marginal estimation. One way that we can apply our marginal estimation techniques is to train an RNNLM with n-gram probabilities in addition to running text. This is helpful when we want to efficiently incorporate data from a much larger corpus without training the RNNLM on it directly (Chelba et al., 2017; Noraset et al., 2018). In this work, we frame the problem as a regression and use a loss equal to the squared difference of log probabilities:

$$\mathcal{L}_N = \frac{\alpha}{2K}\sum_{k}^{K}(\log P_{text}(x_{1:T}^{(k)}) - \log P_{est}(x_{1:T}^{(k)}))^2$$

where α is a hyper-parameter and set to 0.1. Following the result in Table 1, we use the *Zero* method to estimate $P_{est}(x_{1:T}^{(k)})$ as in Eq 3, and add $\mathcal{L}_N$ to the training losses that use the running text corpus.

To evaluate the approach, we follow the Noraset et al. (2018) experiment in which bi-gram statistics from the training text of WT-103 are used to

Loss	PPL
$\mathcal{L}_{text}^{(2)}$ $(Zero)$	104.08
$\mathcal{L}_{text}^{(2)} + \mathcal{L}_{U}^{(2)}$ $(Zero^{(U)})$	102.56
$\mathcal{L}_{text}^{(2)} + \mathcal{L}_{U}^{(2)} + \mathcal{L}_{N}^{(103)}$	**93.95**

Table 3: Test perplexities of RNNLMs trained on different loss functions. Using n-gram probabilities from a larger corpus (WT-103) improves perplexities.

improve an RNNLM trained using WT-2. In our experiment, we use n-grams up to $n = 5$ with frequency greater than 50. We ignore n-gram containing <unk>, because the vocabulary sets are different. Table 3 shows the result. Since we do not use the same setting as in the original work, we cannot directly compare to that work – they use different optimization settings, more expensive n-gram loss, and Kneser-Ney bi-gram language model. However, we see that the proposed n-gram loss is beneficial when combined with the unigram loss. Importantly, unlike the approach in Noraset et al. (2018), our approach requires no sampling which makes it several times faster.

In addition, we present our preliminary result comparing training with the marginal probability of n-grams to training with the complete data. Given a limited budget of optimization steps, we ask whether training on n-grams is more valuable than training on the full corpus. To keep the results compatible, we use the vocabulary set of WikiText-2 and convert all OOV tokens in the training data of WikiText-103 to the "<unk>" token. Figure 1 shows the loss (average negative log-likelihood) of the validation data as the number of optimization steps increases.

We can see that training with the marginals does not perform as well as training with WikiText-103 training data, but outperforms the model trained only with WikiText-2 training data. This might be due to our choice of n-grams and optimization settings such as a number of n-grams per batch, weight of the n-gram loss, and the learning rate decay rate. We leave exploring these hyper-parameters as an item of future work.

4 Conclusion

We investigated how to estimate marginal probabilities of n-grams from an RNNLM, when the preceding context is absent. We presented a simple method to train an RNNLM in which we occasionally reset the RNN's state and also maximize

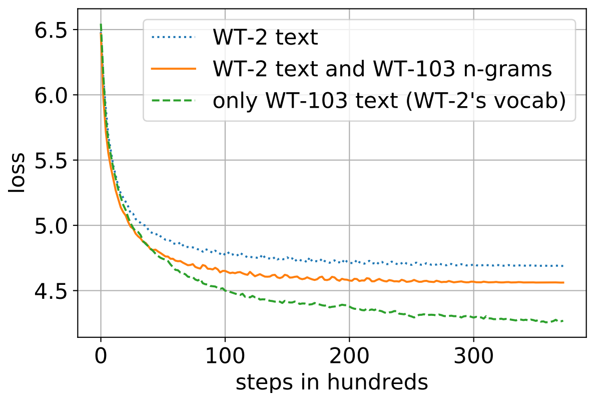

Figure 1: Loss in negative log-likelihood over steps in training. The loss computed using the valid data from WikiText-2 corpus. Training with n-grams from a larger corpus is helpful, but not as well as training with the running text from a larger corpus itself.

unigram likelihood along with the traditional objective. Our experiments showed that an RNNLM trained with our method outperformed other baselines on the marginal estimation task. Finally, we showed how to improve RNNLM perplexity by efficiently using additional n-gram probabilities from a larger corpus.

For future work, we would like to evaluate our approaches in more applications. For example, we can use the marginal statistics for information extraction, or to detect and remove abnormal phrases in text generation. In addition, we would like to continue improving the marginal estimation by experimenting with recent density estimation techniques such as NADE (Uria et al., 2016).

Acknowledgements

This work was supported in part by NSF grant IIS-1351029, the Tencent AI Lab Rhino-Bird Gift Fund, and Faculty of ICT, Mahidol University. We thank the reviewers for their valuable input.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR2014)*. ArXiv e-Prints. 1409.0473.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2014. Textjoiner: On-demand information extraction with multi-pattern queries. In *4th Workshop on Automated Knowledge Base Construction at NIPS 2014*. AKBC.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram Language Modeling using Recurrent Neural Network Estimation. *ArXiv e-prints*, arXiv:1703.10724v2 [cs.CL].

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *International Conference on Learning Representations (ICLR2017)*. ArXiv e-Prints. 1611.01462.

Refal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *ArXiv e-prints*, arXiv:1602.02410v2 [cs.CL].

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

Gbor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations (ICLR2018)*. ArXiv e-Prints. 1707.05589.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations (ICLR2017)*. ArXiv e-Prints. 1609.07843.

Tom Mikolov, Martin Karafit, Luk Burget, Jan ernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Thanapon Noraset, David Demeter, and Doug Downey. 2018. Controlling global statistics in recurrent neural network text generation. In *The Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Stephen Soderland, Oren Etzioni, Tal Shaked, and Daniel Weld. 2004. The use of web-based statistics to validate information extraction. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, pages 21–26.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. 2016. Neural autoregressive distribution estimation. *J. Mach. Learn. Res.*, 17(1):7184–7220.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741, International Convention Centre, Sydney, Australia. PMLR.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *ArXiv e-prints*, arXiv:1409.2329v5 [cs.NE].