Using Large Corpus N-gram Statistics to Improve Recurrent Neural Language Models

Yiben Yang, Ji-Ping Wang
Department of Statistics
Northwestern University
Evanston, IL, 60208, USA
{yiben.yang, jzwang}@northwestern.edu

Doug Downey
Department of Electrical Engineering
& Computer Science
Northwestern University
Evanston, IL, 60208, USA
{d-downey}@northwestern.edu

Abstract

Recurrent neural network language models (RNNLM) form a valuable foundation for many NLP systems, but training the models can be computationally expensive, and may take days to train on a large corpus. We explore a technique that uses large corpus n-gram statistics as a regularizer for training a neural network LM on a smaller corpus. In experiments with the Billion-Word and Wikitext corpora, we show that the technique is effective, and more time-efficient than simply training on a larger sequential corpus. We also introduce new strategies for selecting the most informative n-grams, and show that these boost efficiency.

1 Introduction

Recurrent neural network models of language (RNNLMs) form a foundation for many natural language processing systems. However, the networks can be expensive to train: training a single model over several million tokens can take hours, and searching through the large hyperparameter space of RNNLMs often entails training and testing hundreds of different models. This makes it burdensome to experiment with new RNNLM architectures on large corpora, or to train RNNLMs for new textual domains.

RNNLMs are typically trained on sequential text. In this paper, we investigate how to efficiently augment the training of RNNLMs by regularizing the models to match n-gram statistics taken from a much larger corpus. The motivation is that large-corpus n-gram statistics may be informative to an RNNLM trained on a smaller sequential corpus, but unlike RNNLM training, n-gram statistics are inexpensive to compute even over large corpora. Moreover, the statistics only need to be computed once and can be re-used for training many different smaller-corpus RNNLMs.

Naively, regularizing an RNNLM to match a given set of n-gram statistics is non-trivial, because the marginal probabilities that n-gram statistics represent are not parameters of the RNNLM. In recent work, Noraset et al. (2018) showed that it was possible to regularize an RNNLM to match given n-gram statistics by training the network, when started from a zero state, to match each n-gram probability. However, the regularization approach in that work had tractability limitations—the time cost of the regularization was sufficiently high that using it was inferior to simply training the RNNLM on more sequential text.

In this paper, we present an efficient n-gram regularization technique and show that the technique can improve RNNLM training. Our method has three distinctions from previous work that provide efficiency. First, we prioritize regularizing only the n-grams that are most likely to improve the model, by focusing on cases where the RNNLM's sequential training corpus diverges significantly from the n-gram statistics. Secondly, we regularize the entire output softmax of the RNN to match given conditional n-gram statistics, which means we can impose a large number of statistical constraints using only one softmax evaluation. Finally, we use an ensemble of multiple loss functions in our regularizer, which provides an additional boost. In experiments, we show how n-gram regularization with these enhancements results in better models using the same amount of training time, compared to standard sequential training. We also plan to release our code base and to the research community.[1]

2 Methods

RNNLMs are trained to optimize a loss function L_d, which is defined as the average negative log-likelihood of a training corpus. We regularize the RNNLM to match n-gram statistics by introduc-

[1] https://github.com/yangyiben/Conditional-N-gram-Regularization

ing another penalty term L_p to the loss function that captures how well the model matches n-gram statistics, giving a combined loss L:

$$L = L_d + \alpha L_p,$$

where α is a hyperparameter to control the regularization strength. We use the term *large corpus* to refer to the text utilized to compute the n-gram statistics. We expect the large corpus to be multiple orders of magnitude larger than the *small corpus* utilized for computing the RNN's sequential loss L_d. In the rest of this section, we first define what we mean by conditional n-gram statistics, and then present our regularization methods.

2.1 Conditional N-gram Statistics

An order-N n-gram is a sequence of N words. For a given corpus $\boldsymbol{c}$, we denote the kth distinct order-N n-gram $w_{k1}, w_{k2}, ..., w_{kN}$ as $\mathbf{w}_k^N$, and denote the corresponding order $N-1$ n-gram formed by the first $N-1$ words as $\mathbf{w}_k^{N-1}$. Here, in order to eliminate ambiguity, the notation $N-1$ is exclusively used to represent the prefix. For instance, $\mathbf{w}_k^3$ is the k-th trigram, while $\mathbf{w}_k^{4-1}$ is the trigram prefix of the k-th 4-gram. We define conditional n-gram statistics as the empirical conditional probabilities of observing each next word w_{kN} given the previous $N-1$ gram $\mathbf{w}_k^{N-1}$ for all ks :

$$\hat{P}(w_{kN}|\mathbf{w}_k^{N-1}) = \frac{count(\mathbf{w}_k^N)}{count(\mathbf{w}_k^{N-1})}, \forall \mathbf{w}_k^N \in \Omega(\boldsymbol{c}),$$

where w_{kN} and $\mathbf{w}_k^{N-1}$ are the Nth word and the corresponding previous $N-1$ gram of $\mathbf{w}_k^N$ respectively, and $\Omega(\boldsymbol{c})$ is the set of all unique $\mathbf{w}_k^N$s contained in the corpus $\boldsymbol{c}$. For a given RNNLM, the model conditional probability for some n-gram $\mathbf{w}_k^N$ is defined as:

$$P_\theta(w_{kN}|\mathbf{w}_k^{N-1}) = \mathbb{E}_h(P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h)),$$

where h is the model's hidden state prior to encountering the N-gram. However, it is difficult to express that expectation in terms of the model parameters, so we adopt the approach from Noraset et al. (2018), which has shown preliminary evidence of forming an effective regularizer:

$$\mathbb{E}_h(P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h)) \approx P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h_0),$$

where h_0 is a zero hidden state.

2.2 Conditional N-gram Regularization

We propose three forms of regularization loss functions that penalize the divergence of the model's conditional probabilities from the conditional n-gram statistics.

2.2.1 Mean Squared Log Probability Ratio

We denote our first penalty as L_p^{sq}, defined as:

$$L_p^{sq} = \frac{1}{\|R\|} \sum_{\mathbf{w}_k^N \in R} \left(log \frac{\hat{P}(w_{kN}|\mathbf{w}_k^{N-1})}{P_\theta(w_{kN}|\mathbf{w}_k^{N-1})} \right)^2,$$

where R is a set of n-grams $\mathbf{w}_k^N$ to regularize, and $\hat{P}$ and P_θ are conditional n-gram statistics and model conditional probabilities as defined in Section 2.1. This penalty is similar to that of (Noraset et al., 2018). However, instead of computing the loss with multiple forward passes for different $\mathbf{w}_k^N$s that have the same $\mathbf{w}_k^{N-1}$, we propose to only perform one forward pass for each $\mathbf{w}_k^{N-1}$, and regularize all subsequent Nth words. This makes our loss much more computationally efficient. As this penalty only accounts differences in point-wise probabilities for specific words w_N, it can be used for partially specified distributions where we only know the desired probabilities for some Nth words in a given context, but not the entire distribution.

2.2.2 Mean Kullback–Leibler Divergence

We denote our second penalty as L_p^{KL}, defined as:

$$P_{\mathbf{w}_k^{N-1}} = \hat{P}(w|\mathbf{w}_k^{N-1}), Q_{\mathbf{w}_k^{N-1}} = P_\theta(w|\mathbf{w}_k^{N-1})$$
$$L_p^{KL} = \frac{1}{\|R\|} \sum_{\mathbf{w}_k^{N-1} \in R} D_{KL}(P_{\mathbf{w}_k^{N-1}} || Q_{\mathbf{w}_k^{N-1}}),$$

where here R is a set of prefixes $\mathbf{w}_k^{N-1}$ to regularize. This penalty regularizes all possible subsequent words w_N, thus it only works for fully-specified reference distributions.

2.2.3 Combined Penalty

Because the above two penalty functions differ, we hypothesize that they may be complementary and propose a *combined* penalty $L_p^c = L_p^{sq} + L_p^{KL}$.

2.3 N-gram Selection Strategy

Note that we do one forward pass for each unique $\mathbf{w}_k^{N-1}$, so only the number of distinct prefixes $\mathbf{w}_k^{N-1}$ will significantly affect the computational cost of our regularization methods. Naively regularizing all unique prefixes in a large corpus usually requires a large number of forward passes, which could be expensive. We hypothesize that some prefixes are more useful than others, so we attempt to select the ones that will improve the model the most. We propose to select prefixes $\mathbf{w}_k^{N-1}$ that maximize the **E**xpected **L**og-likelihood **C**hange (ELC), defined as:

$$ELC(\mathbf{w}_k^{N-1}) = \sum \hat{P}(\mathbf{w}_k^N) log \frac{\hat{P}(w_{kN}|\mathbf{w}_k^{N-1})}{P_\theta(w_{kN}|\mathbf{w}_k^{N-1})}.$$

Ideally, P_θ would reflect the statistics of the RNNLM, updated during training, but these are expensive to obtain. Thus we propose to train a inexpensive n-gram model (Chen and Goodman, 1999) on the sequential corpus to serve as P_θ, and we use that to select a fixed set of n-grams to regularize.

3 Experiments

We now present our experiments measuring the effectiveness of conditional n-gram regularization.

3.1 Data and Settings

We experiment on a medium-size (2 layers with 650 hidden states) LSTM language model (Zaremba et al., 2014) over two corpora: Wikitext (Merity et al., 2016) and Google Billion-Word (Chelba et al., 2013) (1B). We adopt weight tying (Inan et al., 2016) and variational dropout (Gal and Ghahramani, 2016). All models are trained by SGD for 30 epochs with batch size 64 and truncated backpropagation (Mikolov et al., 2011) with 35 time steps. The learning rate starts at 20 and then is reduced to 5 at epoch 20. For the 1B corpus, we follow the same procedure in Yang et al. (2017) to generate training, validation and test sets, except that we use only the top 50K vocabulary terms. For the Wikitext corpus, we adopt the Wikitext-2 vocabulary. All of the RNNLM sequential training sets are small subsets sampled from the Wikitext-103 and 1B training corpora.

For each dataset, we use the whole training set as the large corpus for building our reference conditional n-gram statistics. In this study, we only consider conditional trigram regularization for all experiments. The regularization takes additional time during training. To enable a fair experiment, we equalize the training time between regularized models and the baselines, by providing the baselines with more sequential training data than the regularized models. The three proposed penalties are almost equally fast, thus they can be compared against the same baseline. Unlike RNNs, there are no hyperparameter settings or decisions involved in computing the n-gram counts, so this can be done once and re-used across the many RNN training runs that adequate hyperparameter search for RNNs often entails. Moreover, counting n-grams is fast. We approximate that it takes about one minute to obtain n-gram statistics from the Wikitext-103 training corpus, for example. Thus, we set up our experiments to equalize neural network training time, and we ignore the small one-time cost of computing n-gram statistics.

We fix the number of bigrams per batch to be 500, and employ the proposed strategy to select the top X most informative bigrams (X depends on the number of batches in sequential data). Finally, instead of tuning the regularization strength hyperparameter α for each setting, we fix α to be 1.0, 0.75 and 0.5 for the three sequential data sizes based on the heuristic that larger sequential data may need less regularization. More carefully tuning the regularization strength might yield somewhat better results for our methods. Also, using different regularization strengths in the combined penalty might further improve results.

3.2 Results

In Table 1, we compare the performance of our proposed methods against equal-time controlled baselines under different token sizes for both the Wikitext and 1B data sets. In the table, the numbers in the column headings indicate the token count of the sequential corpus used to train the regularized methods. The baselines train on larger corpora, to ensure an equal-time setting as described above. All regularized models outperform their baseline counterparts for all token sizes. Among them, the models regularized by the combined penalty consistently perform best. This illustrates that conditional n-gram regularization is effective at incorporating large-corpus statistics to improve an RNNLM trained on a relatively small corpus. Performing regularization using the combined selection strategy yields more accurate models compared to simply training on a larger sequential corpus.

In Figure 1, we plot validation perplexities after each training epoch for models trained on the 5M-token 1B corpus, against an equal-time baseline. The plot shows that the relative performance of the methods remains similar across training epochs.

Methods	Wikitext			Google 1B		
	500K	1M	2M	2.5M	5M	10M
baseline	161	110	76	110	90	81
sq_log	122	91	72	98	86	80
KL	137	100	77	100	86	78
combined	**112**	**86**	**69**	**94**	**83**	**77**

Table 1: Test perplexities of different methods and sequential training token sizes. sq_log: mean squared log probability ratio penalty. KL: mean Kullback-Leibler divergence penalty. Models with the combined penalty achieve the lowest perplexities.

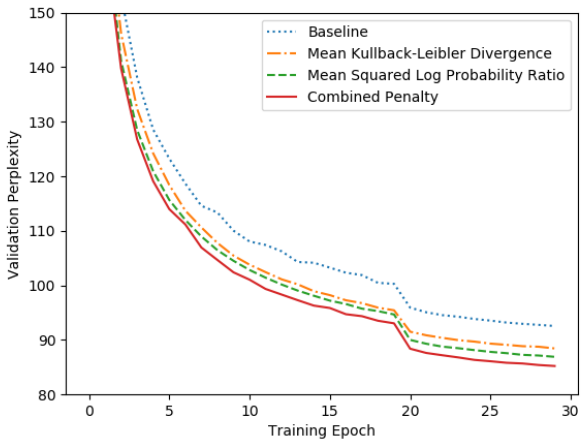

Figure 1: Validation Perplexities on the 5M-token 1B Dataset. The combined penalty performs best.

3.3 Analysis

In Table 2, we demonstrate how the number of regularized bigrams affects the performance. Here, the regularized models always train on wikitext-2 as a sequential corpus, and the baseline trains on larger corpora as the fractions of bigrams increases. The regularized model achieves 66 test perplexity on Wikitext-2 corpus, which is about 2.7 points worse than a state of the art mixture of softmax model (Yang et al., 2017) even though our model has fewer parameters (25M vs. 35M). Including more bigrams helps lower the perplexity, while it also demands extra computational time. ELC performs best when using less than 20% of the bigrams. Regularizing randomly selected n-grams does not outperform the equal-time baseline, indicating that not all n-grams are equally useful. In order to be time efficient, it is important to select informative n-grams, and ELC is an effective measure of informativeness.

% of total bigrams	baseline	random	ELC
0%	86	86	86
5%	76	83	**69**
10%	71	77	**67**
20%	**66**	71	67
40%	**61**	68	66

Table 2: Test perplexities of RNNLMs trained on Wikitext-2 regularized with different numbers of bigrams using the combined penalty. Baseline indicates an equal-time controlled baseline, random is a regularized model with bigrams selected randomly, and ELC indicates our proposed strategy.

In Table 3, we consider ensembling a standard RNN with a KN-smoothed trigram model. This achieves a ppl of 65, but requires 51M more parameters, whereas our regularization with the n-grams achieves most of the perplexity improvement at the cost of zero additional parameters. Further, somewhat surprisingly we find that an ensemble of our regularized RNNLM with the n-gram model achieves much better perplexity of 59.

Models	Test PPL
Unregularized RNNLM	86
Unregularized RNNLM + Trigram KN	65
Regularized RNNLM	67
Regularized RNNLM + Trigram KN	**59**

Table 3: Test perplexities of RNN-LMs trained with and without regularization on the Wikitext-2 corpus, ensembled with a KN-smoothed trigram model trained on the Wikitext-103 corpus.

Another possible way of efficiently utilizing a large corpus would involve training a Word2vec model on the large corpus, and using the pre-trained embeddings within a RNNLM trained on a small corpus. This approach can utilize larger corpora since training a Word2vec model is much faster than training a RNNLM. However, in our preliminary experiments with this approach, we did not observe any improvement when using word embeddings trained on a large corpus. Further experiments with variants of this approach are an item of future work.

4 Related Work

Chelba et al. (2017) trained large-order n-gram models using a recurrent neural network trained over limited context to produce the conditional probability estimates. Our regularizer is trained in a similar way, but by contrast we are focused on how the regularizer can be used in concert with standard sequential RNNLM training to improve the training procedure. We introduce n-gram selection techniques and distinct loss functions that increase the effectiveness of the combined training. Ganchev et al. (2010) presents a posterior regularization method for restricting posterior distributions of probabilistic models with latent variables to obey predefined constraints using the EM algorithm. This approach shares our goal of imposing constraints on probabilistic models, but we focus on RNNLMs, which do not estimate distributions over latent state variables and are not trained using EM. Finally, Mikolov et al. (2011), Józefowicz et al. (2016) and Chelba et al. (2013) trained ensembles of RNNLMs and KN-smoothed n-gram models, and showed that one can obtain a better model when ensembling RNNLMs with n-gram models. Our experiments show that compared to ensemble methods, conditional n-gram regularization achieves similar results at the cost of zero additional parameters, and can perform even better when combined with ensembling.

5 Conclusion and Future Work

In this paper, we have proposed methods to utilize using large corpus n-gram statistics to regularize RNNLMs trained on a smaller corpus. Our experiments demonstrate that the proposed regularization penalties are effective in improving model performance, and can be more time efficient than training RNNLMs on a larger sequential corpus. Selecting informative n-grams is shown to be important. In future work, we would like to obtain a better theoretical understanding of why starting the RNNLM from a zero state forms an effective n-gram regularizer. We would also like to extend our regularization approach to BiLSTMs (Peters et al., 2017) and Transformers (Alec Radford and Sutskever, 2018; Devlin et al., 2018).

Acknowledgments

This work was supported in part by NSF Grant IIS-1351029 and the Allen Institute for Artificial Intelligence.

References

Tim Salimans Alec Radford, Karthik Narasimhan and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI.*

Ciprian Chelba, Tomas 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.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *CoRR*, abs/1703.10724.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531.

Thanapon Noraset, Doug Downey, and Lidong Bing. 2018. Estimating marginal probabilities of n-grams for recurrent neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2930–2935. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, abs/1711.03953.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.