Adversarial Objectives for Text Generation

Ankit Vani
New York University
ankit.vani@nyu.edu

Abstract

Language models can be used to generate text by iteratively sampling words conditioned on previously sampled words. In this work, we explore adversarial objectives to obtain good text generations by training a recurrent language model to keep its hidden state statistics during sampling similar to what it has sees during maximum likelihood estimation (MLE) training. We analyze the convergence of these models and discuss the effect of the adversarial objective on word-level and character-level language models. We find that using an adversarial objective assists the MLE objective and results in faster convergence and lower validation perplexities for character-level language models. We also collect sentence quality ratings from human participants and show that character-level language models with the adversarial objective generates qualitatively better sentences than the standard character-level recurrent language model.

1 Introduction

One of the most natural ways to generate text is to build a language model and sample from it. Language models have been used to disambiguate natural language input in tasks like speech and handwriting recognition, and to improve fluency of generations in machine translation and captioning. Models that can better understand the structure of language and the semantics of how the world works can be useful in all of these tasks.

There has been tremendous interest in generative models for images recently, and one of the most active areas of research is to use generative adversarial networks (GANs) (Goodfellow et al. [2014]) to play a min-max game where a discriminator is trained to distinguish between fake image generations and real image data, and a generator is trained to fool the discriminator into believing that its generations are real. The trained generators are then capable of producing very realistic images (Radford et al. [2015]).

Although computer vision has seen a large fraction of related research in adversarial networks, adversarial objectives for natural language generation is less explored. We ask the question of whether adversarial objectives can benefit text generation, and discuss possible approaches to improve language models using them. We focus on language models since they directly model the distribution over natural language sequences and often provide a very simple way to sample sentences from them.

Recurrent language models have been shown to be very effective at the task of language modeling (Mikolov et al. [2010], Zaremba et al. [2014], Gal and Ghahramani [2015]), and sampling from recurrent language models only requires sampling a word from every timestep and providing it as input to the next timestep iteratively. However, unless we have a perfect language model, such iterative sampling can produce sentences that are unlikely to be grammatically or semantically correct. Since during training, the model has only seen sentences from the training data, such a sample can result in a domain shift and the recurrent neural network (RNN) dynamics at this point may not produce reasonable conditional word distributions. In other words, the statistics of the hidden states during sampling can start diverging from the statistics of the hidden states seen during maximum

likelihood estimation (MLE) training of the language model, resulting in poorer language modeling and text generation performance.

We propose a method to use adversarial objectives in conjunction with the standard MLE objective to help retain the same RNN behavior during sampling as seen during MLE training with true inputs. We evaluate the effect of the adversarial objective on recurrent language models and the quality of sentences produced by these models according to human ratings.

2 Related work

Mikolov et al. [2010] introduced RNN-based language models, which are capable of capturing long-term dependencies in text and learn a distributed representation of words. We train an RNN language model using Gated Recurrent Units (GRUs) (Cho et al. [2014]), through an additional adversarial objective.

Generative adversarial networks (GANs) have been very successful in computer vision at producing novel, high-quality and realistic images (Goodfellow et al. [2014], Radford et al. [2015]). In this work, we use adversarial objectives closely related to GANs[1] in the domain of language modeling. GANs require differentiability of the generators, and word outputs sampled from RNN language models are not differentiable due the the discrete nature of the words produced. Thus, we instead discriminate based on the hidden states of the generative RNN, providing a surrogate continuous generator output for the discriminator. The generator in a GAN maps a prior noise distribution to the data distribution. In our work, the noise for the generator comes from the sampling of words from the conditional word or character distributions produced by the RNN at each timestep.

Domain-adversarial neural networks (Ganin et al. [2015]) enable representation learning where training data and test data come from similar but different distributions. An adversarial objective ensures that the model learns features that cannot discriminate between training and testing data distributions, and the training data is also used to learn a supervised task such as classification. The approach presented in our work is very similar, where the RNN hidden states distribution during sampling is forced to be indistinguishable from the RNN hidden states distribution when the RNN gets true inputs.

Recently, Lamb et al. [2016] presented Professor Forcing, a way of training RNNs using MLE and an adversarial objective, and showed that the adversarial objective has a regularizing effect in RNN training. They experiment with this approach on language modeling, digit generation, handwriting generation and music synthesis. For language models, they observe that an adversarial objective helps for character-level language models, but not for word-level language models. However, they do not offer much insight into why this happens. This work presents the same idea, although started independently, and contains more detailed analysis of adversarial objectives applied to training language models for text generation.

3 Language modeling with an adversarial objective

3.1 Recurrent language models

A language model learns a distribution over a sequence of words, such that sequences that are plausible under a language have higher likelihood. The probability of a sequence $\mathbf{x} = \langle x_1, \ldots, x_T \rangle$ under a language model parameterized by θ is

$$p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_1, \ldots, x_{t-1}) \tag{1}$$

The language model is trained to maximize the likelihood of the sequence of words in a training corpus, or equivalently, to minimize the negative log-likelihood of training data, by optimizing over θ. This objective is referred to as maximum likelihood estimation (MLE), and is given by

$$\min_\theta \mathcal{L}_{MLE}(\theta) \triangleq \mathop{\mathbb{E}}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [-\log p_\theta(\mathbf{x})] \tag{2}$$

[1] Unlike in the GAN setting, in this work the 'real' data distribution keeps changing as the model learns. However, we still choose to refer to this model component as a GAN in this report.

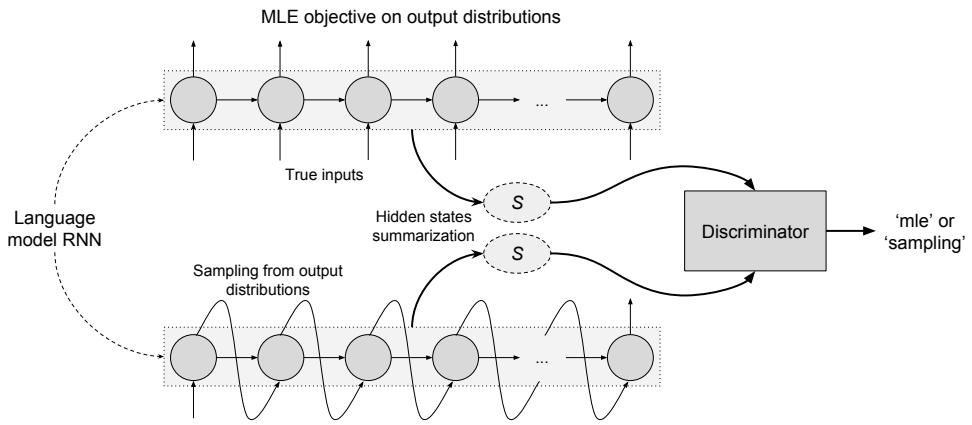


Figure 1: Architecture for training the GAN-RNN language model.

Various language models have been explored in literature, including count-based n-gram language models, feedforward n-gram language models (Bengio et al. [2003]), recurrent language models (Mikolov et al. [2010]), etc. n-gram language models make an assumption that the last word in a sequence of words is independent of all words before the previous n words when conditioned the previous n words. However, recurrent language models do not make this assumption, and can model arbitrarily long sequences. Neural language models can learn continuous representation of words, also known as word embeddings, that are capable of encoding task-specific semantic information about words. Thus, words similar in meaning under a task would also be closer in the embedding space. This enables neural language models to generalize beyond the sequence of words seen during training, to sequence of concepts. RNN-based language models, with their ability to capture long-term dependencies and learn continuous representations of words, are thus extremely powerful and attractive for the task of language modeling.

Training recurrent language models using the MLE objective involves giving the true input at every timestep and maximizing the likelihood of the next word over the model parameters, conditioned on previous true inputs. Simple RNNs are prone to vanishing gradients if the largest eigenvalue of the recurrent weight matrix is less than one, and have exploding gradients if this largest eigenvalue is greater than 1 (Pascanu et al. [2013]). We use GRUs to avoid vanishing gradients, and gradient clipping to a value of 5 to alleviate exploding gradients.

To generate sentences from a language model, one needs to iteratively sample words from the generative model described by Equation (1). In case of RNNs, one can provide an initial word, which is often a special start-of-sentence token, to get a distribution over the next word. We sample a word from this distribution and provide it as input for the next timestep, thus generating a distribution over the next word conditioned on the start-of-sentence token and the first word sampled. Referring to Equation (1), RNNs encode the context $x_1, \ldots, x_{t-1}$ in the hidden state $\mathbf{h}_{t-1}$ at timestep $t-1$.

Character-level RNN language models are similar to the word-level models, except that each timestep produces a distribution over the next character, instead of the next word. We explore text generation through both word-level and character-level RNN language models in this work.

3.2 Adversarial objective

The quality of text generated from a language model relies on the ability of the language model to produce accurate conditional distributions for words. While sampling from a recurrent language model, the sequence of words generated can lead to hidden states whose statistics diverge from the hidden state statistics seen during MLE training. Since these hidden states are not encountered during training, when this happens, the distributions over words produced by the language model may not be close to the true conditional distributions conditioned on the previously generated words. We propose to use an adversarial objective that ensures that the RNN behavior during sampling matches that during MLE training where it gets the true inputs.

We present the GAN-RNN framework, illustrated in Figure 1. While a generative RNN is trained to maximize the log-likelihood of the training data as described in Equation (2), a discriminator is trained to classify whether the hidden states come from MLE training with true inputs or from sampling. The gradients from the discriminator are reversed and backpropagated to the generative RNN, thus training the model to have its hidden state statistics indistinguishable between those during MLE training and those during sampling. The adversarial objective with the language model parameters θ and discriminator parameters ϕ is given by

$$\min_{\theta} \max_{\phi} \mathcal{L}_{GAN}(\theta, \phi) \triangleq \mathop{\mathbb{E}}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D_{\phi}(S_{\theta}(\mathbf{x}))] + \mathop{\mathbb{E}}_{\mathbf{x} \sim p_{\theta}(\mathbf{x})} [\log(1 - D_{\phi}(S_{\theta}(\mathbf{x})))] \quad (3)$$

Here, S_θ is a differentiable summarization function operating on the hidden states of the generative RNN and, as suggested by Lamb et al. [2016], also the input embeddings. The output of S_θ is taken by the discriminator as its input. This summarization function does not use any learnable parameters to map the RNN hidden states to the input space of the discriminator, but is responsible for backpropagating gradients back to the generative RNN.

In practice, we found it useful to systematically control the updates made to the generative RNN and the discriminator during adversarial training. We implemented a scheduler that keeps track of the discriminator accuracies from the past k batches, and uses the maximum value from this list as an optimistic measure of the discriminator's accuracy at any given batch. If this accuracy is greater than 97%, we do not update the discriminator to prevent it from getting too strong. If it gets too strong, no useful gradients are propagated back to the generative RNN, and the discriminator stays at a perfect accuracy for the remainder of training. On the other hand, if this accuracy is lower than 75%, the gradients are of poorer quality and we do not update the generator[2]. We used $k = 8$ in our implementation.

The word or character embedding matrices are not updated when training the generator with the adversarial objective. We leave the optimization of the embedding matrix to the MLE objective, and use the adversarial objective only to alter the behavior of the RNN itself through its hidden states under fixed embeddings.

3.3 Discrimination strategies

Different discrimination strategies can be proposed by altering the formulation of S_θ and the discriminator. Considering the final state of the generative RNN as an inherent summarization of the generation produced, one of the simplest summarization functions S_θ is to consider the final state of the generative RNN, and use a feedforward discriminator on it. However, we found that when discriminating based on the final state, the discriminator accuracy struggles to go beyond 60%, even without any adversarial training steps of the generator.

Thus, we consider discrimination strategies that look at the sequence of hidden states of the generator, which is much more informative of the RNN's behavior as compared to just the final state. We propose a recurrent discriminator, that takes as inputs the hidden states and input embeddings of the generative RNN through S_θ. We experimented with both unidirectional and bidirectional discriminators, and found the bidirectional discriminator to provide a larger gain in perplexity for GAN-RNN.

For the bidirectional discriminator, we concatenate the hidden states of a bidirectional GRU RNN for each timestep, and perform a 1D convolution with the number of filters equal to the hidden unit size of the discriminator RNN divided by the window size[3]. We fix the window size to be 5. The convolution operation helps bring into consideration some local context into the transformed states at each timestep, after which we linearly transform the states following with an Exponential Linear Unit (ELU) non-linearity (Clevert et al. [2015]). The transformed output is then flattened and transformed into a single value, the sigmoid of which gives the value of $D_\phi(S_\theta(\mathbf{x}))$ for the input $\mathbf{x}$.

We also experimented with an energy-based discriminator (Zhao et al. [2016]), where we use the reconstruction cost of a unidirectional discriminator RNN's hidden states as the energy function. The discriminator tries to minimize the energy associated with the input hidden states when the

[2]Lamb et al. [2016] also use a similar strategy, but our implementation was prepared independently and before their publication to arXiv.

[3]Using a convolution operation here helped implement a decent discriminator that could still fit reasonably well on Titan Black GPUs.

language model gets true inputs, and maximize the energy associated with the input hidden states from sampling up to a margin. Lacking the notion of discriminator accuracy in this case, we could not implement the scheduler to decide the schedule of generator and discriminator updates. In our experiments, we could not get the energy-based discriminator to help, and it often hindered MLE training, significantly slowing down its convergence. This can possibly be attributed to the fact that encoding a sequence into a fixed length vector and reconstructing it is hard, especially if each of the elements in such sequences are high-dimensional vectors.

4 Experiments

4.1 Datasets

We run our experiments with two datasets, analyzing language modeling and consequent text generation on one small and one large dataset:

- The Penn Treebank (PTB) corpus, which has 887k words and 4,975k characters for training, 70k words and 389k characters for validation, and 78k words and 438k characters for testing. It has a vocabulary size of 9,966 and an alphabet size of 49.

  This serves as a small dataset for our experiments.

- Project Gutenberg (Gutenberg), which is a collection of free ebooks. We downloaded 3,036 English books and combined them into a dataset after randomizing the global order of sentences. The prepared corpus contains 198,242k words and 1,035,216k characters for training, 79k words and 414k characters for validation, and 118k words and 614k characters for testing. For word-level models, we preprocess this dataset to retain the most frequent 20k words (they cover 97% of the data), and for character-level models, we remove punctuations and miscellaneous symbols, and normalize numbers to '#', retaining an alphabet size of 30.

  This serves as a large dataset for our experiments.

4.2 Language modeling

We compare the performance of character-level RNN and GAN-RNN, word-level RNN and GAN-RNN, feedforward n-gram language models and count-based n-gram models on language modeling for PTB and Gutenberg. In all our GAN-RNN experiments here, we use the bidirectional discriminator.

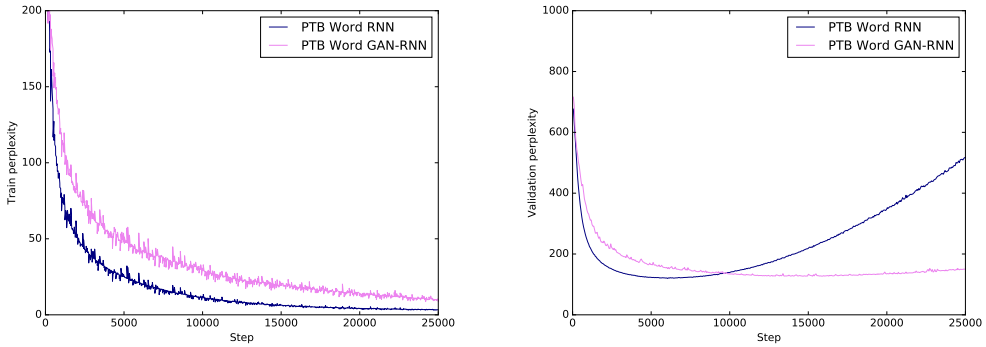
We used the SRI Language Modeling Toolkit (SRILM) (Stolcke [2002]) to perform the count-based n-gram model experiments. All count-based n-gram language modeling experiments implement backoff, and thus increasing the order of the n-gram model does not necessarily hurt the validation perplexity of the model. We do not consider the n-gram models without smoothing, since they cannot generalize beyond the training data n-grams, and their validation perplexity can be ∞.

For the character-level RNN and GAN-RNN, we divided the training set into non-overlapping sequences of length 512, similar to the experimental setting of Lamb et al. [2016]. For the word-level RNN and GAN-RNN, we divided the training set into non-overlapping sequences of length 256. In both cases, the RNN uses single-layer GRUs with 800 hidden units. The character embedding size is 96 and the word embedding size is 224.

For the feedforward n-gram models (Bengio et al. [2003]), we concatenate the n input word embeddings, and project them to a hidden layer through a ReLU non-linearity. Another projection and ReLU non-linearity reduces this to a vector of embedding size, which is used compute the score of the output word based on its inner product with embeddings from an output word embedding matrix. A softmax operation on these scores gives the probability distribution over the $(n+1)$th word given n input words. We searched over the word embedding sizes 32, 64, 96, 128, 192, 224 and 256 and hidden layer sizes of 64, 96, 160, 192, 256, 512 and 1024 for the configurations that give the best validation perplexities and don't immediately overfit. We use a word embedding size of 224 for Gutenberg and 32 for PTB, and we use a hidden layer size of 512 for Gutenberg and 96 for PTB. We prepend each training sentence with n start-of-sentence tokens, so that a sentence can be sampled starting from the first word given n start-of-sentence tokens during generation.

	n-gram order	PTB Valid PPL	PTB WPS	Gutenberg Valid PPL	Gutenberg WPS
Laplace n-gram + backoff	3, 5, 10	1037.7	–	523.2	–
Interpolated n-gram	3, 5, 10	177.5	–	132.6	–
Kneser-Ney n-gram	3	166.5	–	130.3	–
Kneser-Ney n-gram	5, 10	187.4	–	152.4	–
Modified Kneser-Ney n-gram	3	169.5	–	132.5	–
Modified Kneser-Ney n-gram	5, 10	193.6	–	156.8	–
Feedforward	3	214	61k	129	52k
Feedforward	5	209	58k	118	52k
Feedforward	10	209	57k	112	47k
Word-level RNN	–	**119**	27k	**98**	18k
Word-level GAN-RNN	–	127	3.2k	112	1.5k
Character-level RNN	–	333	5.2k	569	5.5k
Character-level GAN-RNN	–	**327**	650	**565**	675

Table 1: Comparison of validation perplexities on PTB and Gutenberg with different language models. For neural models, WPS is the training time in words per second on a Titan Black GPU.



(a) Training perplexities

(b) Validation perplexities

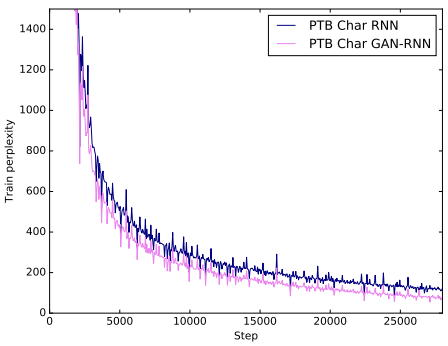
Figure 2: Training and validation perplexities on training word-level RNN and GAN-RNN on PTB.

We use the Adam (Kingma and Ba [2014]) optimizer with a learning rate of 0.0001 to train the RNN models, and with a learning rate of 0.001 to train the feedforward n-gram models.

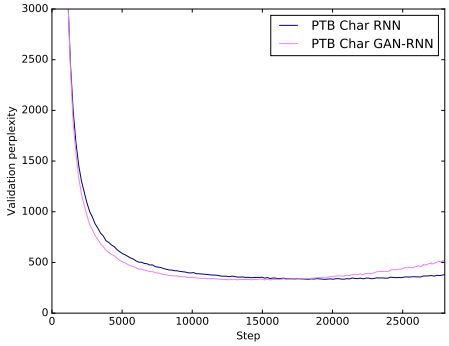
The validation perplexities for the aforementioned models are summarized in Table 1. For both datasets, the best validation perplexity for word-level models is attained by RNNs. This is because RNNs are capable of capturing long-term dependencies in sentences by considering the entire history, whereas n-gram consider only the previous n words to produce the distribution over the current word. Count-based n-gram models also do not generalize beyond the word sequences they've seen in the training data, because they do not learn any semantics about words themselves as the neural models do. The best perplexity for the character-level models is attained by GAN-RNNs, although for Gutenberg the difference is very small.

4.2.1 Effect of the adversarial objective

In our experiments, we noticed that for word-level GAN-RNNs, the adversarial objective slows down the convergence of the MLE loss, and consequently in case of PTB also slows down the validation perplexity from shooting back up due to overfitting as seen in Figure 2. However, it still fails to beat the best validation perplexity of the RNN models. In case of Gutenberg, the word-level GAN-RNN trails behind the RNN in training and validation perplexity both, since the RNN does not overfit the way it does with PTB. For word-level models, it appears that the adversarial objective competes with the MLE objective, for which a plausible reason is presented below.

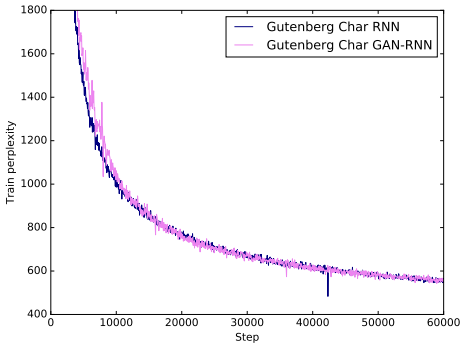


(a) Training perplexities

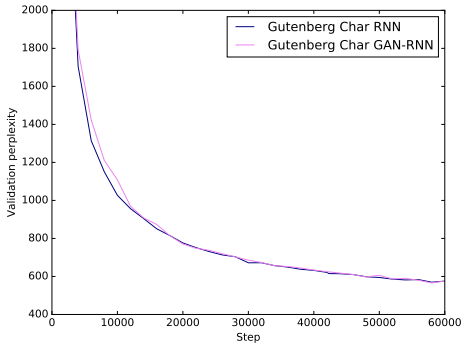


(b) Validation perplexities

Figure 3: Training and validation perplexities on training character-level RNN and GAN-RNN on PTB.



(a) Training perplexities

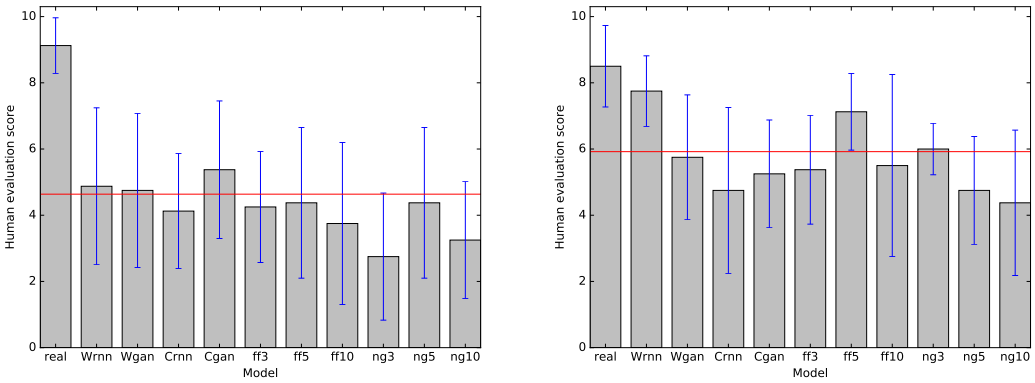


(b) Validation perplexities

Figure 4: Training and validation perplexities on training character-level RNN and GAN-RNN on Gutenberg.

For character-level models on PTB, the adversarial objective assists the MLE objective, resulting in faster convergence of both the training and validation perplexities, as seen in Figure 3. The fact that the adversarial objective helps with a character-level model but not with a word-level model is possibly due the added difficulty of the task of character-level modeling as compared to word-level modeling. The space of possible character-level sequences representing a sentence is much richer than the space of word-level sequences, but the number of plausible sequences remains the same. For simpler models like word-level language models, the MLE objective can be used to maximize the likelihood of the data directly, but the optimization problem becomes more difficult with the complexity of the data space we wish to model. GANs have been very successful in training generative models for images, where a very small fraction of the pixel space represents natural images. Lamb et al. [2016] also report higher gains with their Professor Forcing architecture in modeling more complex data. In character-level language models, the hidden states of an RNN during sampling are more likely to diverge into regions that have not been seen during training as compared to word-level language models. The adversarial objective helps bring these sample-time hidden state statistics closer to the hidden state statistics seen by the model during MLE training with true inputs. If the sampled hidden state statistics are likely to already be close to the statistics seen during training, perhaps due to the MLE objective being good enough, then the adversarial objective can hamper the MLE convergence by discriminating against valid hidden state regions, which may explain the results with word-level models.

However, given huge amounts of data, the performance of training a character-level language model directly with MLE can match that of training with an adversarial objective. Character-level language models are still simple enough that one can prepare such huge datasets in practice. This is apparent



(a) Human ratings for PTB

(b) Human ratings for Gutenberg

Figure 5: Ratings given by human participants for quality of sets of sentences generated by different language models. The blue errorbars show the standard deviation of responses, and the horizontal red line is the mean rating across all models. W[rnn|gan]: word-level RNN or GAN-RNN; C[rnn|gan]: character-level RNN or GAN-RNN; ff[3|5|10]: feedforward n-gram models with $n = 1, 3, 5$; ng[3|5|10]: count-based n-gram models with $n = 1, 3, 5$.

from character-level language modeling results on Gutenberg, where the addition of an adversarial objective makes very little difference to the training or validation perplexities, as seen in Figure 4.

4.3 Quality of generated sentences

To evaluate the quality of sentences generated from different language models, we do a human survey. For both PTB and Gutenberg, we prepared two questions each containing five sentences from the real data, from the word-level RNN, word-level GAN-RNN, character-level RNN, character-level GAN-RNN, feedforward n-gram models with $n = 3, 5, 10$ and count-based n-gram models with $n = 3, 5, 10$. Thus, we had a total of 44 questions, with two questions and ten sentences per model.

We looked at the histogram of the number of words in a sentence from the PTB and Gutenberg datasets, and chose five reasonable lengths per dataset. For each model, we then looked for sampled sentences that are close to these lengths, and picked them randomly to get the set of five sentences for a question. For PTB, this histogram looks like a Gaussian distribution with mean 20 and standard deviation 10, and the selected lengths are 8, 16, 24, 32, 40. For Gutenberg, the histogram of sentence lengths has a large mass around 5 and then decreases linearly till 40. The selected lengths for this dataset are 5, 6, 12, 20, 30.

To select the sentences for the survey, we chose sentences sampled during training at steps where the models have the lowest validation perplexities. This is to ensure that the models have a decent understanding of the language but haven't memorized the training data. This is important for PTB, where most models start overfitting as the training perplexities become very low. For count-based n-gram models, we compared different approaches for a given n, such as simple n-gram, Laplace, interpolation, Kneser-Ney and modified Kneser-Ney, and chose the samples from the model with the best validation perplexity. For both datasets, this was the model with Kneser-Ney smoothing for $n = 3$ and interpolation for $n = 5, 10$.

The participants of the survey were asked to rate the set of sentences they see on a scale of 0 to 10, where 0 means that the sentences make no sense semantically or grammatically and 10 means that the sentences are grammatically correct and the content makes sense. We also suggested that a rating of 5 is around where sentences are either grammatically correct but don't make a lot of sense or you can understand the context but the grammar isn't quite right. The participants were informed of some preprocessing such as lack of punctuations and use of special tokens for numbers. The questions were presented to participants in a random order to ensure that there is no position-dependent bias in the responses.

We considered responses from 10 participants. Figure 5 outlines these responses for both datasets and different models trained on these datasets. We can see that in general, generations for Gutenberg have rated higher than generations for PTB, most likely due to the significantly larger size of the training data and shorter sentences. We can also observe that for both datasets, the character-level GAN-RNN rates better than the character-level RNN, indicating that the adversarial objective for character-level language models has a qualitative benefit. However, the difference is not as significant in case of Gutenberg. These qualitative results are in line with the quantitative language modeling results from the previous section. Another observation, that agrees with the language modeling results, is that the word-level RNN rates better than the word-level GAN-RNN.

5 Conclusion

We hypothesized that generating text by iterative sampling from a recurrent language model may push the hidden state statistics of the RNN into regions that have not been seen during MLE training, and the conditional distributions produced by the model when this happens may not be accurate enough for reasonable generations. To tackle this problem, we presented a way to train recurrent language models with both an MLE objective and an adversarial objective. While the MLE objective maximizes the data likelihood under the model, the adversarial objective encourages the model to retain the same behavior during sampling as it has during MLE training with true inputs at each timestep.

We analyzed the convergence of word-level and character-level GAN-RNNs, and showed that the adversarial objective benefits character-level language modeling. We argued that this might be because the space of sequences in character-level language modeling is much richer, with a smaller fraction of plausible sequences being valid sentences as compared to in the case of word-level language modeling. This makes it more likely for hidden states during sampling to end up in regions not seen during MLE training, such that the generations do not correspond to plausible sentences. The adversarial objective assigns high energies to these hidden state regions, and moves the hidden state statistics to be indistinguishable from the hidden state statistics during MLE training with true inputs. We evaluated the quality of generated sentences from various language models and showed that using an adversarial objective also has a qualitative benefit for character-level language models.

Generative adversarial networks are known to be notoriously hard to train, requiring very precise hyperparameters for optimal operation. Due to limited computing resources, we did not do a detailed hyperparameter search. It is possible that given the right hyperparameters, one may gain benefits from the adversarial objective even for the word-level models and from other discrimination strategies, which can be worth investigating.

Source code

The source code for this project uses TensorFlow (Abadi et al. [2015]), and can be found at `https://github.com/ankitkv/TextGAN/tree/uncond`.

Acknowledgments

I wish to express my gratitude to my advisor, Prof. Kyunghyun Cho, for his advice and invaluable suggestions throughout this project.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944966`.

K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. URL http://arxiv.org/abs/1511.07289.

Y. Gal and Z. Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *ArXiv e-prints*, Dec. 2015.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *ArXiv e-prints*, May 2015.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. *ArXiv e-prints*, Oct. 2016.

T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010. URL http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318, 2013. URL http://jmlr.org/proceedings/papers/v28/pascanu13.html.

Project Gutenberg. https://www.gutenberg.org/.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

A. Stolcke. Srilm - an extensible language modeling toolkit. pages 901–904, 2002.

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014. URL http://arxiv.org/abs/1409.2329.

J. J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016. URL http://arxiv.org/abs/1609.03126.