Controlling Global Statistics in Recurrent Neural Network Text Generation

Thanapon Noraset, David Demeter, and **Doug Downey**
Department of Electrical Engineering & Computer Science
Northwestern University, Evanston IL 60208, USA
{nor, ddemeter}@u.northwestern.edu, d-downey@northwestern.edu

Abstract

Recurrent neural network language models (RNNLMs) are an essential component for many language generation tasks such as machine translation, summarization, and automated conversation. Often, we would like to subject the text generated by the RNNLM to constraints, in order to overcome systemic errors (e.g. word repetition) or achieve application-specific goals (e.g. more positive sentiment). In this paper, we present a method for training RNNLMs to simultaneously optimize likelihood and follow a given set of statistical constraints on text generation. The problem is challenging because the statistical constraints are defined over aggregate model behavior, rather than model parameters, meaning that a straightforward parameter regularization approach is insufficient. We solve this problem using a dynamic regularizer that updates as training proceeds, based on the generative behavior of the RNNLMs. Our experiments show that the dynamic regularizer outperforms both generic training and a static regularization baseline. The approach is successful at improving word-level repetition statistics by a factor of four in RNNLMs on a definition modeling task. It also improves model perplexity when the statistical constraints are n-gram statistics taken from a large corpus.

Introduction

Recurrent neural network language models (RNNLMs) are a critical component of many natural language generation tasks such as machine translation, summarization, automated conversation, and caption generation (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2014; Rush, Chopra, and Weston 2015; Karpathy and Fei-Fei 2015; Li et al. 2016). The models are trained to maximize the likelihood of a training corpus, and evaluated on the likelihood they assign to a held-out test corpus (measured in terms of perplexity). RNNLMs, and in particular Long Short-term Memory Networks (LSTM) (Hochreiter and Schmidhuber 1997), have provided dramatic improvements in the perplexities of language models in recent years (Jozefowicz et al. 2016; Melis, Dyer, and Blunsom 2017).

However, while RNNLMs optimize well on the perplexity metric, when we utilize the models to generate text we often wish to encourage the models to obey additional statistical constraints. For example, one way in which RNNLM text generators tend to fail is by repeating the same words or phrases too often, leading to nonsensical output (See, Liu, and Manning 2017; Paulus, Xiong, and Socher 2017; Noraset et al. 2017). Can we learn a low-perplexity model that avoids this failure mode? Likewise, we may want to adjust our model's output distribution to not reflect undesirable biases in our training corpus, or to utilize a different style, such as shorter sentences or more positive sentiment.

In this paper, we present a regularization technique that allows modelers to specify additional *soft constraints* on language models during the training. For example, the constraints might encourage the model to repeat words less often, or to use shorter sentences, etc. The constraints are stated as a *reference distribution* that gives target marginal probability values for events in the text (e.g., the probability that a certain word will repeat consecutively). The regularizer encourages the global statistics of the model to match the reference distribution. Implementing the regularizer in RNNLMs is challenging because the marginal output probabilities of an RNNLM do not correspond directly to parameters of the model (as they do in simpler n-gram models), and instead must be inferred. In this work, we solve this problem by computing estimates of the model's marginals from a sample of generated text, which is continually updated as training proceeds. We then use the estimates during training to encourage the model to generate text that matches the reference distribution by minimizing the KL-divergence. We refer to this method as *Dynamic KL Regularization.*

To evaluate our approach, we experiment with two types of constraints in RNNLMs. The first type aims to reduce local repetition. Local repetition is a long-standing issue with RNNLMs: the models disproportionately repeat the same word within a short window (see Figure 1). The problem becomes particularly acute when we ask the RNN to generate its estimate of the *high-likelihood* text for a given input. We show how using dynamic KL regularization to encourage the RNNLM to exhibit a similar repetition profile to the training data can reduce repetition with little harm to perplexity. The second type of constraints attempt to match n-gram statistics from a reference corpus. Training an RNNLM on even tens of millions of tokens can be computationally costly, but often we can readily acquire n-gram statistics over even billions of tokens. We show how dynamic KL regularization can be used to incorporate large-corpus statistics that im-

Sampled text from PTB model
... between ***the u.s. and the u.s. and*** <unk> agencies ...
... closed higher in ***paris paris paris paris*** and zurich ...
... the exxon ***aerospace*** and ***aerospace*** firm is n't ...

Sampled text from WordNet model	
samurai:	a ***Japanese Japanese Japanese Japanese*** warrior
vintner:	a person who ***wine wine***
papal:	***associated with or associated with or*** belonging to the papacy

Figure 1: Examples of repetitions generated by models trained with PTB text and WordNet definitions.

prove an RNNLM trained on a smaller corpus from the same distribution.

The rest of the paper proceeds as follows. We first derive our regularizer from the KL-divergence between the reference distribution and the model distribution. We then present dynamic KL regularization, an approximation to our regularizer that is differentiable and uses estimates of marginal probabilities from text generated by the model. Our experiments compare dynamic KL regularization with three baselines, and show that the proposed regularization is better at matching repetition and n-gram statistics on the Penn Treebank dataset. Furthermore, we show how our method can use statistics from the larger WikiText-103 corpus (Merity et al. 2016) to improve an RNNLM trained on a more tractable small corpus from the same distribution (WikiText-2). Finally, in the last set of experiments is an application of our approach to reduce repetition for a conditional language generation task. We choose definition modeling (Noraset et al. 2017), the task of generating definitions from word embeddings, as a benchmark. We find that dynamic KL regularization improves repetition statistics by a factor of four, and also improves BLEU score. Finally, we provide analysis, review related work, and conclude.

Dynamic KL Regularization

Our goal is to train a recurrent neural network language model such that the global statistics of its generated text are similar to a specified set of statistical soft constraints. Each constraint applies to the marginal probability $P(w, c)$, where w is a word and c is a *condition* – an event specified over the context up to and including the word. For each constraint, we specify two quantities, $P_0(c)$ and $P_0(w|c)$ to be defined as a constraint $P_0(w, c) = P_0(w|c)P_0(c)$. As a simple example, if we wish to constrain the probability of "the dog", we would define $P_0(w_i = \text{"dog"}|w_{i-1} = \text{"the"})$ and $P_0(w = \text{"the"})$. The distribution $P_0(\mathcal{W}, \mathcal{C})$ is denoted as the *reference* distribution, where $\mathcal{W}$ is a vocabulary set and $\mathcal{C}$ is a set of conditioning events.

An RNNLM defines a probability distribution over words conditioned on previous words as the following:

$$\begin{aligned} P_\theta(w_t|w_{1:t-1}) &= P_\theta(w_t|h_t; \tau) \\ &\propto \exp(\theta_o^{(i)} h_t/\tau) \\ h_t &= g(h_{t-1}, w_{t-1}; \theta) \end{aligned}$$

where $w_{1:t-1}$ is a sequence of previous words, τ is a temperature (1.0 unless specified), θ denotes the parameters of the model, $\theta_o^{(i)} \subset \theta$ is a set of weights associated with output word i, and $g(\cdot)$ is a recurrent function such as an LSTM unit. When training on a sequence of ground truth tokens, RNNLMs are optimized to maximize the log likelihood of the training data, or equivalently to maximize the summed log likelihood of each token given the previous ones.

Our goal is to train an RNNLM to generate text with statistics similar to the reference distribution, while simultaneously achieving high likelihood on the training data. In principle, matching the reference distribution entails minimizing the KL-divergence from the reference distribution P_0 to the model distribution P_θ. Specifically, we would like to minimize the KL-divergence as the following:

$$\begin{aligned} \mathcal{R}(\theta, P_0) = &\mathop{\mathbb{E}}_{c \sim P_\theta(\mathcal{C})} [D_{KL}(P_\theta(\mathcal{W}|c)||P_0(\mathcal{W}|c)] \\ &+ D_{KL}(P_\theta(\mathcal{C})||P_0(\mathcal{C})) \end{aligned}$$

The first term of $\mathcal{R}$ defines a divergence between the model and the reference conditional distribution of words, and the second term defines the same measure for the distribution of the conditioning events. Since we are minimizing log-likelihood of the training data, we decide to minimize $\mathcal{R}(\theta, P_0)$ using conditioning events with respect to the training data. Our loss function is then:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} -\log P_\theta(w_t|h_t) + \alpha \mathcal{R}(w_{t-l:t}) \quad (1)$$

$$\begin{aligned} \mathcal{R}(w_{t-l:t}; \theta, P_0) = &\sum_{c_t \in \mathcal{K}_t} D_{KL}(P_\theta(\mathcal{W}|c_t)||P_0(\mathcal{W}|c_t)) \\ &+ D_{KL}(P_\theta(\mathcal{C})||P_0(\mathcal{C})) \end{aligned} \quad (2)$$

where $\mathcal{K}_t \subset \mathcal{C}$ is a set of conditioning events occurring in a truncated sequence $w_{t-l:t}$ (i.e. $\{w_{t-1} = \text{"the"}\}$ in our example). α is a weight that controls how strongly the objective favors matching the reference distribution versus maximizing the log likelihood of the training corpus. The sequence is truncated because we only need as many previous tokens as the conditions require. Furthermore, when training with truncated back-propagation through time, we omit any event occurring before the current training sequence.

However, the loss function in Equation 2 is challenging to compute exactly. In a classical n-gram model, obtaining a marginal distribution (i.e. $P_\theta(\mathcal{W}|c)$ or $P_\theta(\mathcal{C})$) would be straightforward. These quantities are parameters of an n-gram model. However, in state-of-the-art RNNLM language models, the situation is more complex. An advantage of RNNLMs is that their output distributions incorporate arbitrarily long context, via the hidden state. But, this makes it difficult to infer marginal probabilities, because doing so requires summing over all possible preceding contexts.

On the other hand, given a sufficiently large corpus of text generated by an RNNLM, we can easily estimate the model's marginal probabilities by counting. This sampling-based approach has the added benefit that our estimates reflect the model's actual behavior during inference – i.e.

the estimates account for *exposure bias*, the fact that models are never exposed to their own generated text at training time, only at inference time (Ranzato et al. 2015; Bengio et al. 2015). However, we need to sample output from the model to accumulate a large body of generated text, and perform maximum-likelihood estimation over the text (counting) to obtain accurate statistics. This process makes Equation 2 not differentiable and prohibits gradient-based training algorithms.

We propose an approach based on an approximation of the KL-divergence computed in Equation 2 by dynamically updating an estimate of the model's long-run inference behavior during the training. Specifically, we use maximum-likelihood estimate $\hat{P}_\theta$ of the marginal probabilities under the model from its generated text, and compute the KL-divergence terms in Equation 2 using both of these marginals and the model's current prediction at each time step t:

$$\hat{D}_{KL}(P_\theta(\mathcal{C})||P_0(\mathcal{C})) = \sum_{c\in\mathcal{C}} P_\theta(c|h_{t-l:t})\log\frac{\hat{P}_\theta(c)}{P_0(c)} \tag{3}$$

$$\hat{D}_{KL}(P_\theta(\mathcal{W}|c_t)||P_0(\mathcal{W}|c_t)) = \sum_{w\in\mathcal{W}} P_\theta(w|h_t)\log\frac{\hat{P}_\theta(w|c_t)}{P_0(w|c_t)} \tag{4}$$

Intuitively, Equation 3 and 4 minimize expected log-likelihood ratio between the model's estimate and the reference distribution – decreasing the current predicted probability if the log ratio is positive (i.e. if we over-generate) and increasing the predicted probability if the log ratio is negative. In other words, the log-likelihood ratio is a constant (non-differentiable) providing feedback to the model current prediction (differentiable).

Of course, the model's marginal probabilities will change as training proceeds. Thus, as we train, we sample new sequences from the model, dynamically updating the log-likelihood ratio. We label this as ***dynamic***. However, we need a large sample size to accurately estimate the model's marginals ($\hat{P}_\theta$). To efficiently compute the model's marginals, we keep a fixed-size pool of the generated text. As training proceeds, we generate a small portion of the text to replace the oldest tokens, and update the model's marginals every few steps.

Alternate approaches

As discussed above, we infer marginal probabilities of the model by using overall statistics from the model output text. Thus the regularization requires only the aggregated statistics to match the reference distribution. On the other hand, one might argue that similar effect can be achieved by encouraging the model prediction to match the reference distribution (for every context). To justify our choice to sample, we compare against a baseline that uses the current model's output distribution directly for the regularization. This is equivalent to replacing $\hat{P}_\theta$ in Equation 3 and 4 with P_θ. We refer to this alternative as ***static***, as opposed to dynamically updating statistics from the generated text.

We define our soft constraints as a joint probability of a condition and a word event – but in some settings, we may not need to regularize the conditions themselves, only the word event. For example, we may simply want to increase the frequency of "mr. robot", regardless of how many times the model generates the first condition token "mr.". Further, in some cases, computing the probability of the conditioning events from the model might not be straightforward or computationally expensive as Equation 3 requires probability distribution over conditioning events. This leads us to another alternate approach that constrains only the conditional probability $P(\mathcal{W}|c)$. That is, we remove the KL-divergence term of the conditioning event distribution (the last term of Equation 2). We refer to this modified regularization as ***dynamic-$P(\mathcal{C})$***.

Examples of statistical constraints

Local repetition statistics A common problem in generated text from RNNLMs is local repetition, where the same substring repeats multiple times in a short output text. For example, one definition of "fairness" sampled from an unregularized definition model is "the property of being fair and fair." As our first type of soft constraints, we regularize the model such that the probability of words appearing again within a window of tokens is close to that in a given reference text corpus. To construct the constraints, for each $k \in \{1, 2, 3\}$ we define a conditioning event to be that some word repeats after k tokens, i.e. $\mathcal{C} = \{w_{i-k} = w_i : k \in \{1, 2, 3\}\}$. We compute the marginal probabilities as:

$$P(c^k) = \sum_{w\in\mathcal{W}} N_r(w, k)/|\mathcal{W}|$$

$$P(w|c^k) = N_r(w, k)/\sum_{w'\in\mathcal{W}} N_r(w', k)$$

where $N_r(w, k)$ is the number of times w re-occurs after k tokens. For example, a probability of "the" repeats after 2 tokens is $freq(\text{"the} * * \text{ the"})/\sum_w freq(w * * w)$. We apply Witten-Bell estimate (Chen and Goodman 1996) to smooth the distribution to allow a query for unobserved events of both model's and reference marginals. During the training time, the model probability of the conditioning event is then the current probability of previous k words:

$$P_\theta(c^k|h_{t-l:t}) = P_\theta(w_{t-k}|h_t)$$

Using n-gram statistics Another type of the soft constraints is the n-gram distribution where the set of conditioning events is phrases of length $n-1$. These constraints can correct disproportionate n-gram frequencies in the generated text, and can also be used to incorporate statistics from a larger corpus. The marginal probabilities for a conditioning phrase $c = w'_{1:n-1}$ are then:

$$P(c) = P(w'_{1:n-1})$$

$$P(w|c) = P(w_i = w|w_{i-n:i-1} = w'_{1:n-1})$$

From the reference or sampled text, we obtain these marginal probabilities using Kneser-Ney n-gram language models (Kneser and Ney 1995). During training time, the

model probability of the conditioning event is simply the current output probability:

$$P_\theta(c|h_{t-l:t}) = \prod_{k=0}^{n-1} P_\theta(w_{t-k}|h_{t-k})$$

For simplicity of the experiments, we use only bigram constraints in this paper.

Experiments and Results

We now present our experiments evaluating dynamic KL regularization. We begin by providing a comparison between baselines and our regularization on a common benchmark for language modeling, Penn Treebank (PTB).[1] Then, we present our results on practical usage of the regularization on two other datasets for language modeling (WikiText) (Merity et al. 2016) and definition modeling (WordNet definitions) (Miller 1995).

For language models, we use a 2-layer LSTM with 650 hidden units. The embeddings and output logit weights are tied and have 650 units. We adopt the training hyper-parameters from Zaremba et al (2014). Specifically, we use stochastic gradient descent with the standard dropout rate of 50%. The initial learning rate is 1.0, with a constant decay rate of 0.8 starting at the 6$^{\text{th}}$ epoch. Perplexity validation stops significantly improving after around 20 epochs. For definition models, we use the same settings described in (Noraset et al. 2017). Since a model will generate a uniform distribution before any training, we pre-trained all models for 5 epochs before applying the regularization to save training time.

As for hyper-parameters for the regularization, we did a limited exploration and use the following settings throughout the experiments. The weight α on the regularization is set to be 1.0 for repetition constraints and 0.5 for bigram constraints. In addition, the log-likelihood ratio in the approximated KL-divergence (Equation 3 and 4) is clipped at -2.0 to 2.0. This helps reduce variance from the reference distribution and the model's distribution from the generated text. Finally, text is generated every 100 steps to update the model's marginal distribution in an amount equal to 10% of the reference text. The implementation is publicly available.[2]

Baseline Comparison

In the first set of experiments, we compare our regularization with standard (*unregularized*) training on language modeling using Penn Treebank (PTB). We present all comparisons with both repetition constraints where $k = 1, 2$ and 3, and bigram constraints discussed in the previous section. The statistical constraints are computed from the training data of PTB.

To measure performance, we compute total absolute frequencies *mismatch* between the reference text and the generated text:

$$\sum_{w,c \in \mathcal{W},\mathcal{C}} |N_0(w,c) - N_\theta(w,c)|$$

where $N(w,c)$ is the frequency of an event w, c in the reference corpus (N_0) or generated text (N_θ). This error measure shows how much a model's generated text differs from the reference distribution.

First, we present our results on repetition constraints. Table 1 shows the total mismatch of words repeated after k tokens. We can see that *dynamic* can reduce the number of mismatches in word repetition significantly compared to *unregularized* (by more than 50%) at all k, and *dynamic*$-P(\mathcal{C})$ has essentially the same error, while *static* does not work well. This suggests that we do need to sample to estimate the marginals from the model. Interestingly, the mismatch of *static* is due to under-repetition. Finally, we can see that the regularization has only a slight impact on the test perplexity.

For bigram constraints, we measure absolute error in terms of the frequency mismatch of unigrams and bigrams. The result in Table 2 shows that *dynamic* has the lowest error, but the reduction from *unregularized* is less than what we observed in the previous experiments (a 23% and 7% reduction). This could be due to the larger number of constraints we specify ($\sim$ 100k vs. only 200, after filtering out singletons). Again, *dynamic*-$P(\mathcal{C})$ works slightly less well than *dynamic*, and *static* does not work.

Larger corpus statistics

We now demonstrate how to use bigram statistics from a larger corpus to regularize a model trained on a similar, but smaller corpus. For this, we choose WikiText corpora containing WikiText-103 (large) and WikiText-2 (small). Note that WikiText-103 training data does not contain Wikipedia articles that overlap with validation and testing data on WikiText-2. Since WikiText-103 has a much larger vocabulary size, when computing bigram constraints, we use vocabulary from WikiText-2 and set out-of-vocab words to the unknown symbol. The results in Table 3 shows *dynamic-103* model has better perplexity than *unregularized* and *dynamic-2* models.

Since the reference corpora differ in this experiment, our mismatch measure from the previous experiments is not meaningful. Instead, we measure the KL-divergence of the unigram and bigram distributions to evaluate the efficacy of our constraints. Table 3 shows the KL-divergence from the generated texts to WikiText-103's training data. The *dynamic-103* model is exposed to bigram statistics from WikiText-103, and has the lowest KL-divergence among the generated texts.

Definition generation

Finally, we evaluate our method on a definition modeling. Definition modeling is the task of producing a natural-language definition for a term, given the term and its embedding (Noraset et al. 2017). Definition modeling serves as a benchmark for a simple word-to-sequence language generation task. Certain sequence-to-sequence models have an existing solution to repetition by encouraging models to attend to different positions of an input sequence (Tu et al. 2016; Merity et al. 2016). However, this approach cannot be applied in word-to-sequence models, as there is only one input token.

[1] http://www.fit.vutbr.cz/ imikolov/rnnlm

[2] https://github.com/northanapon/seqmodel/tree/aaai18

	Test PPL	Absolute frequency mismatch k=1		k=2		k=3	
unregularized	77.6	7.07k	%	6.25k	%	8.30k	%
dynamic	78.3	3.00k	(58%)	**2.91k**	**(53%)**	**3.89k**	**(53%)**
dynamic$-P(\mathcal{C})$	78.3	**2.99k**	**(58%)**	3.11k	(50%)	4.67k	(44%)
static	79.9	8.08k	14%	6.20k	(1%)	8.44k	2%

Table 1: Performance of models trained using PTB and regularized with word repetition statistics. Dynamic KL regularization significantly reduce the mismatch in word repetition comparing to training without the regularization. In parentheses, percentage changes are computed relative to *unregularized*.

	Test PPL	Absolute frequency mismatch unigram		bigram	
unregularized	77.6	163k	%	673k	%
dynamic	77.8	**125k**	**(23%)**	**623k**	**(7%)**
dynamic$-P(\mathcal{C})$	77.7	147k	(9%)	640k	(5%)
static	79.8	157k	(4%)	754k	12%

Table 2: Performance of models trained using PTB and regularized with bigram statistics. A model trained with dynamic KL regularization has similar perplexity, and generates text with unigram and bigram frequency closer to the training corpus than an unregularized model.

	Test PPL	KL-divergence unigram		bigram	
unregularized	91.6	0.121	%	1.785	%
dynamic-2	91.4	0.111	(8%)	1.733	(3%)
dynamic-103	**86.8**	**0.072**	**(40%)**	**1.586**	**(11%)**

Table 3: Performance of models trained using WikiText-2, regularized using statistics of the training data from Wikitext-2 and WikiText-103 respectively.

We take all lemma definitions from WordNet and split into training, validating, and testing data. We experiment with repetition constraints from the training data. A common use case of a language generation model is to find a high likelihood sequence. To test whether the proposed regularizer holds up in this setting, we use the greeedy algorithm to generate a definition for each word in the training data and compare the absolute repetition frequency mismatch, as in our first set of experiments. Note that the test perplexities are always computed under normal temperature. Following the original paper, we compute BLEU score as a measure of output definition quality in the Greedy setting. We omit BLEU scores from the Sample setting, because sampled text under a models distribution is often not a high likelihood sequence that we would compare with the reference texts.

Table 4 shows the results of the repetition constraints experiment. Consistent with the baseline comparison, *dynamic* has lower mismatch compared to *unregularized*. Again, the perplexity is similar between the two models. The repetition problem becomes more severe when the text is being generated greedily. In the later part of the table, we can see that mismatch of repetitions increases for both *unregularized* and *dynamic*. However, *dynamic* still has much lower mismatch, especially for tokens that repeat immediately. In addition, *dynamic* results in an increase in BLEU score from *unregularized*.

Dynamic KL regularization can also be adapted to specify constraints over greedily generated text, rather than sampled text. To test this, we trained another model with the same repetition constraints as *dynamic*, but that generates in a near-greedy mode (with temperature of 0.1) during training. As shown in the last row of the table, *dynamic-greedy* can drive the repetition mismatch during the greedy generation further down with slightly worse perplexity (under the usual distribution, $\tau = 1.0$) and BLEU score.

Discussion and Analysis

In this section, we discuss and analyze dynamic KL regularization. We discuss common types of local repetition of the generated definitions and show a few examples that the regularizer solves. Then, we note on the computation cost associated with the regularizer. Finally, we present a preliminary experiment where we have a small set of constraints that conflict with the training data.

Local repetition and n-gram duplication

The results in Table 4 show a total reduction of repeated words in local context windows. We find this indirectly also reduces the number of duplicate n-grams within a definition. Figure 2 shows percentage of n-gram duplicate within a definition from the test data generated greedily from the models. We can see that models trained with the regularizer generate definitions with fewer duplicate n-grams. Note that some part of the reduction is due to the generated definitions being shorter (8.6 vs 8.0 tokens per output sequence).

For further qualitative analysis, we identify common cases of local repetition and provide examples of improvement in the regularized model as shown in Figure 3. We notice that the most common cases of local repetition come after a conjunction, especially "or". This could be due to the fact that a word that follows a conjunction like "or" or "and" often has a meaning similar to the word before the conjunction, and thus the model's hidden states are often similar on either side of the conjunction, increasing the probability of repetition. Another category of local repetition is an error where words, often adjectives, repeat immediately. Interestingly, in this case the unregularized model ends the repetition with a related word after a few repeated tokens. The immediate repetition case is also where regularization has the

	Test PPL	Test BLEU	Absolute frequency mismatch k=1		k=2		k=3	
Sample Generation								
unregularized	48.0	N/A	1.74k	%	4.78k	%	4.28k	%
dynamic	48.0	N/A	**0.41k**	**(76%)**	**2.99k**	**(37%)**	**2.33k**	**(46%)**
Greedy Generation								
unregularized	48.0	18.5	10.73k	%	34.96k	%	36.90k	%
dynamic	48.0	**19.1**	0.72k	(93%)	13.70k	(61%)	14.37k	(61%)
dynamic-greedy	48.8	18.9	**0.69k**	**(94%)**	**11.16k**	**(68%)**	**8.87k**	**(76%)**

Table 4: Performance of models training using WordNet definitions and regularized with word repetition statistics. **Top**: A model trained with our regularization maintains the same perplexity, but generate significantly less repetition. **Bottom**: The definitions are greedily generated from the same models with one additional model. The regularized model generates less repetition under low temperature and has slightly improved BLEU score. In addition, regularized with statistics from low temperature results in similar performance.

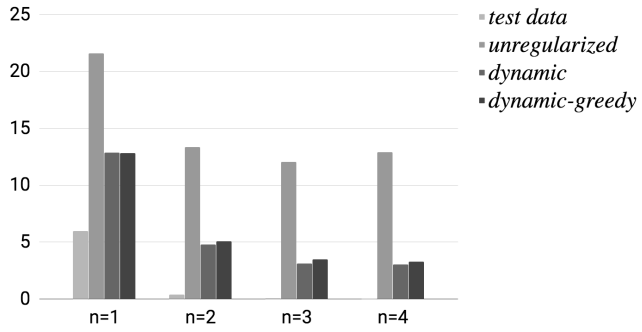

Figure 2: Percent of duplicate n-grams within a definition in test data and definitions generated greedily from models. Reduction in local repetition also reduces n-gram duplication in output sequences.

most positive impact (see Table 4, k=1). The final common repetition type is repeating common phrases, which typically occurs in erroneous definitions — this kind of repetition often results in a sequence reaching a maximum length threshold before generating any content words.

Computational cost

The proposed regularization incurs additional computation to estimate the marginal distribution (generating text) and matching constraints in the running text from the training data. These increased training times are directly proportional to the amount of sampled text generated and the number of soft constraints. With our serial implementation, the training time for the regularized models is approximately two to five times longer than their unregularized counterparts. However, we believe that the additional processes can be done in parallel to the model training.

Conflicting statistics

Most of our experiments use a reference distribution from either the training data itself, or a closely similar distribution. But, we may desire a much different distribution instead. What happens if the constraints conflict dramatically with the training data? We provide a preliminary experiment on a small, artificial set of constraints in order to reveal the effect of the regularizer in such a situation. We create imputed constraints by artificially increasing the probability of "san francisco" and "mr. robot", and decreasing the probability of "new york" from the PTB training data. Hence, the conditioning constraint is the modified unigram probability of {"new", "san", "mr."}, and the conditional constraints include modified probability of "york", "francisco", and "robot" given "new", "san", and "mr." respectively.

Method	NY	SF	MR.R
Training	946	249	0
Reference	455	740	2,136
unregularized	988	215	0
dynamic	792	450	1,089

Table 5: Frequencies of "new york", "san francisco", and "mr. robot" on different texts. The reference text has manually modified frequencies of the bigrams. Text generated from the regularized model has frequency profile closer to the reference text.

As we can see in Table 5, the regularizer can manipulate the frequencies of target bigrams to some certain extent, even though the constraints conflict significantly with the training data. Below are two excerpts of text generated by the regularized model:

> "...grower who has gotten out of san francisco on wall street 's very heavy..."

> "...mr. robot noted that the underlying supply of american companies helped..."

Related Work

A common technique for enforcing a set of pre-defined constraints on a probabilistic model is to modify the model's prediction to follow set of declarative rules. For example, Roth and Yih (2005) add integer linear programming during the inference of CRF model to incorporate constraints. Chang et al. (2008) proposed Constrained Conditional Model to eliminate predicted labels that violate con-

Word	Generated definition
Repeat after conjunctions (most common)	
develop	make a new or new or new
	make a new or more
alleviation	the act of relieving or ... or relieving the body or ... or body
	the act of relieving something
cut	a cut of wood or wood or metal or plastic or plastic or ... or plastic
	a cut of wood or metal
Repeat immediately	
slim	having a thin thin thin thin thin coat
	having a slim or thin shape
samurai	a Japanese Japanese Japanese Japanese warrior
	a Japanese warrior
Repeat common phrases	
telluric	of or relating to or characteristic of or ... or characteristic of a comet
	of or relating to the earth
papal	associated with or associated with or belonging to the papacy
	of or relating to or characteristic of a pope
fatuous	marked by or ... or marked by or characterized by or ... or characterized by
	having or showing a lack of pretensions

Figure 3: Examples of common cases that the regularized model reduces local repetition from definitions generated greedily. Each word shows definitions from unregularized (upper) and regularized (lower) model respectively. "..." indicates a phrase repeating a few times.

straints during both training and inference. In recent neural network models, an output mask is often applied to zero-out probabilities of invalid labels (Williams, Asadi, and Zweig 2017; Liang et al. 2017). For instance, Paulas et al. (2017) masks the probability of a word that will lead to duplicate trigrams during the text generation (with beam search). Distinct from this direction, this paper focuses on *training* a model to obey soft statistical constraints, which are not applied during inference.

Our goal is to regularize an RNNLM's output distribution during training such that the global statistics of the generated text are relatively close to a given set of statistical constraints. This is a very different objective from recent regularization techniques, which are aimed at solving overfitting. For example, variations of dropout regularization randomly mask out activations or parameters of the model to be zero (Wan et al. 2013; Srivastava et al. 2014; Gal and Ghahramani 2016) and they have been successfully applied to train RNNLMs (Zaremba, Sutskever, and Vinyals 2014; Merity, Shirish Keskar, and Socher 2017). Data noising techniques modify the input words directly. This can involve simply randomly dropping off input words (Bowman et al. 2016), or using smoothing and back-off techniques from n-gram language modeling to compute the probability of the noise words (Xie et al. 2017).

The regularization investigated in this paper can be viewed as a *label regularization* or *label smoothing* technique where the output distribution of a model is trained to match a reference distribution. This technique encourages the model to be less confident, and so less overfitted (Szegedy et al. 2015; Reed et al. 2014; Hinton, Vinyals, and Dean 2014). On the other hand, Mann and McCallum (2007) proposed expectation regularization to augment the training with unlabeled data by encouraging model predictions on the unlabeled data to match human-provided label priors. In language modeling, however, label smoothing techniques have not been widely explored. Rosenfeld (1996) applied the maximum entropy principle to train n-gram language models. Recently, Pereyra et al (2017) uses the same principle to penalize overconfident predictions of RNNLMs. They minimize KL-divergence from the uniform distribution to the model output distribution: $D_{KL}(P_\theta||u)$. Extending the previous work, we explore two different non-uniform reference distributions, and introduce a substantially more powerful context-dependent label smoothing technique. Label smoothing can be seen as a static KL regularizer, and we show how our novel dynamic KL regularization performs better than static approaches on our tasks.

Improving the overall quality of the generated text from RNNLMs has been a popular direction in recent research. A general approach is to solve the *exposure bias* by letting the models consume some of its own output predictions during training. This includes scheduled sampling (Bengio et al. 2015) and beam-search optimization (Wiseman and Rush 2016). Many works apply the REINFORCE algorithm (Williams 1992) to directly optimize a sequence-level score, which is often the final evaluation metric such as BLEU or ROUGE score (Ranzato et al. 2015; Bahdanau et al. 2016; Rennie et al. 2017; Paulus, Xiong, and Socher 2017). While BLEU and ROUGE score use n-gram matching similar to our proposed regularization with n-gram constraints (i.e. *bigrams*), they are locally defined per output sequence and might not capture global statistics. In generative adversarial network training, the score is the output prediction of a synchronously trained discriminator (Yu et al. 2017; Che et al. 2017). On the other hand, we use count-based statistics which can be computed more efficiently. It is worth noting that our dynamic KL regularization can be used alongside these approaches.

Conclusion

We investigated how to train RNNLMs to follow a set of soft constraints from a reference distribution. We presented *dynamic KL regularization*, which encourages an RNNLM to match a reference distribution by adjusting the regularizer as training proceeds, based on sampling the model's generated text. We experimented with two types of soft constraints, one for repetition and the other for bigram distributions. Our approach is shown to lower the mismatch in repetition frequency between generated and reference text. This results in a factor of four improvement in local repetition in a definition modeling task. In addition, dynamic KL regularization can utilize the bigram distribution from a large corpus

to decrease the perplexity of a language model trained on a smaller corpus.

While we explore word-level soft constraints in this paper, the approach is more broadly applicable. In future work we would also like to incorporate higher-level constraints such as syntactic or semantic information. Given the proper set of statistical constraints, dynamic KL regularization could be used to encourage models to generate text with a particular sentiment, writing style, reading level, and so on. Constructing reference distributions suitable for applying dynamic KL regularization to these other tasks is an item of future work.

Acknowledgments

This work was supported in part by NSF Grant IIS-1351029, the Allen Institute for Artificial Intelligence, and a gift from Microsoft. We are grateful to the reviewers for their valuable input.

References

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; and Bengio, Y. 2016. An Actor-Critic Algorithm for Sequence Prediction. *ArXiv e-prints* arXiv:1607.07086v3 [cs.LG].

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of 3rd International Conference on Learning Representations*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 1171–1179.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A.; Jozefowicz, R.; and Bengio, S. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 10–21. Berlin, Germany: Association for Computational Linguistics.

Chang, M.-W.; Ratinov, L.; Rizzolo, N.; and Roth, D. 2008. Learning and inference with constraints. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, AAAI'08, 1513–1518. AAAI Press.

Che, T.; Li, Y.; Zhang, R.; Devon Hjelm, R.; Li, W.; Song, Y.; and Bengio, Y. 2017. Maximum-Likelihood Augmented Discrete Generative Adversarial Networks. *ArXiv e-prints* arXiv:1702.07983v1 [cs.AI].

Chen, S. F., and Goodman, J. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 310–318. Santa Cruz, California, USA: Association for Computational Linguistics.

Gal, Y., and Ghahramani, Z. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 1019–1027.

Hinton, G.; Vinyals, O.; and Dean, J. 2014. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop: NIPS 2014*.

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

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the Limits of Language Modeling. *ArXiv e-prints* arXiv:1602.02410v2 [cs.CL].

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39:664–676.

Kneser, R., and Ney, H. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, 181–184 vol.1.

Li, J.; Galley, M.; Brockett, C.; Spithourakis, G.; Gao, J.; and Dolan, B. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 994–1003. Berlin, Germany: Association for Computational Linguistics.

Liang, C.; Berant, J.; Le, Q.; Forbus, K. D.; and Lao, N. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 23–33. Vancouver, Canada: Association for Computational Linguistics.

Mann, G., and McCallum, A. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In Ghahramani, Z., ed., *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, 593–600. Omnipress.

Melis, G.; Dyer, C.; and Blunsom, P. 2017. On the State of the Art of Evaluation in Neural Language Models. *ArXiv e-prints* arXiv:1707.05589v2 [cs.CL].

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models. *ArXiv e-prints* arXiv:1609.07843v1 [cs.CL].

Merity, S.; Shirish Keskar, N.; and Socher, R. 2017. Regularizing and Optimizing LSTM Language Models. *ArXiv e-prints* arXiv:1708.02182v1 [cs.CL].

Miller, G. A. 1995. WordNet: A lexical database for english. *Magazine Communications of the Association for Computing Machinery* 38(11):39–41.

Noraset, T.; Liang, C.; Birnbaum, L.; and Downey, D. 2017. Definition modeling: Learning to define word embeddings in natural language. In *The Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A Deep Reinforced Model for Abstractive Summarization. *ArXiv e-prints* arXiv:1705.04304v3 [cs.CL].

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, .; and Hinton, G. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proceedings of 5th International Conference on Learning Representations*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence Level Training with Recurrent Neural Networks. *ArXiv e-prints* arXiv:1511.06732v7 [cs.LG].

Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2014. Training Deep Neural Networks on Noisy Labels with Bootstrapping. *ArXiv e-prints* arXiv:1412.6596v3 [cs.CV].

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rosenfeld, R. 1996. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language* 10(3):187–228.

Roth, D., and Yih, W.-t. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 736–743. New York, NY, USA: ACM.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 379–389. Association for Computational Linguistics.

See, A.; Liu, P. J.; and Manning, C. D. 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)*, 1073–1083. Vancouver, Canada: Association for Computational Linguistics.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 3104–3112.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints* arXiv:1512.00567v3 [cs.CV].

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 76–85. Berlin, Germany: Association for Computational Linguistics.

Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y. L.; and Fergus, R. 2013. Regularization of neural networks using DropConnect. In *Proceedings of Machine Learning Research*, 1058–1066.

Williams, J. D.; Asadi, K.; and Zweig, G. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 665–677. Vancouver, Canada: Association for Computational Linguistics.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wiseman, S., and Rush, A. M. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1296–1306. Austin, Texas: Association for Computational Linguistics.

Xie, Z.; Wang, S. I.; Li, J.; Lvy, D.; Nie, A.; Jurafsky, D.; and Ng, A. Y. 2017. Data noising as smoothing in neural network language models. In *Proceedings of 5th International Conference on Learning Representations*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent Neural Network Regularization. *ArXiv e-prints* arXiv:1409.2329v5 [cs.NE].